Cryptographie (RSA) et primalité

January 26, 2011

(Chapitre pour le cours : groupes et arithmétique, MI4, L2 option info)

1 Cryptographie

1.1 Principes généraux

La cryptographie est l'art (ou la science) des messages secrets : on veut pouvoir envoyer des informations sans qu'une autre personne que le destinaire puisse en bénéficier. Un problème annexe est de pouvoir identifier avec certitude l'auteur du message. On pense communément que le seul moyen est d'utiliser un *code secret*; en fait, l'originalité de la cryptographie *à clefs publiques* réside précisément dans le fait que le code n'est pas secret, mais connu (au moins en grande partie) de tous! Ce n'est pas seulement une curiosité mathématique, c'est aussi le principe régissant les cartes bancaires, les transactions sur Internet, etc.

Le principe général est le suivant : on appelle $\mathcal{M}$ l'ensemble des messages (en pratique on prendra $\mathcal{M} = [0, N-1]$, ou encore $\mathcal{M} = \mathbf{Z}/N\mathbf{Z}$) ; deux personnes A et B souhaitant échanger des messages sans qu'une troisième personne C puisse les déchiffrer choisissent chacun des bijections $f_\mathrm{A}, f_\mathrm{B} : \mathcal{M} \to \mathcal{M}$; l'ensemble $\mathcal{M}$ (disons l'entier N) est connu de tous, de même que f_A et f_B, par contre – et c'est l'idée centrale – la bijection réciproque f_A^{-1} (resp. f_B^{-1}) n'est connue que de A (resp. de B). Cela ne signifie pas bien sûr que, connaissant f_A, il est impossible de calculer f_A^{-1}, mais que ce calcul serait si long qu'il est irréalisable en pratique ; nous verrons plus loin comment construire de telles fonctions.

Quand A veut envoyer à B un message $m \in \mathcal{M}$ (disons un entier modulo N), elle envoie en clair $m' = f_\mathrm{B} \circ f_\mathrm{A}^{-1}(m)$; noter qu'elle connaît f_B (qui est dans l'annuaire) et f_A^{-1} (qui est son secret). Pour déchiffrer le message, B calcule $f_\mathrm{A} \circ f_\mathrm{B}^{-1}(m')$, qui lui redonne m ; noter qu'elle connaît f_A (qui est dans l'annuaire) et f_B^{-1} (qui est son secret). Le système possède un double avantage : non seulement C ne pourra déchiffrer le message qu'en calculant f_B^{-1} (ce qui est supposé impraticable), mais B peut être sûre que c'est bien A qui lui a envoyé le message, puisque celui-ci a dû être codé en utilisant f_A^{-1} que seul A connaît!

Ce procédé est une forme simplifiée de procédures connues sous le nom de protocole de Diffie-Hellman (1976) ; sa sécurité repose sur le choix de fonctions *f à sens unique* , c'est-à-dire telles que f soit facile (rapide) à calculer, mais f^{-1}

soit impossible en pratique à déterminer. Plusieurs constructions de fonctions ont été proposées, mais une des plus robustes et des plus utilisées repose sur le constat simple que si p, q sont de très grands nombres premiers (disons une centaine de chiffres), alors le calcul de leur produit $N := pq$ peut s'effectuer très rapidement (disons dix mille opérations élémentaires), cependant, si l'on ne connaît que N, le calcul de la factorisation est extrêmement long, voire impossible en pratique.

1.2 Complexité des opérations arithmétiques

Pour être précis, il faut donner un sens mathématique aux mots "rapide à calculer" et "trop long pour être caculé". Voila une première approche assez simple.

Notation/Convention. L'écriture $O(f(n))$ (lire "grand o de $f(n)$") désigne une fonction de n que l'on peut majorer par $Cf(n)$ pour une certaine constante C.

Soit n un entier, une fois qu'on a choisi une base $b \geq 2$, on écrit n en base b, c'est-à-dire avec des *chiffres* $a_i \in [0, b-1]$:

$$n = a_0 + a_1 b + \cdots + a_r b^r = \overline{a_r a_{r-1} \dots a_1 a_0}^{\,b}, \qquad \text{avec disons } a_r \neq 0,$$

(les deux choix les plus courants étant $b = 10$ – écriture décimale usuelle – et $b = 2$ – écriture binaire, particulièrement adaptée à la programmation sur machine)[1]. On considèrera une opération sur les chiffres comme une unique opération (ou encore comme une opération nécessitant $O(1)$ temps machine). Il est naturel d'appeler *complexité* du nombre n le nombre de chiffres nécessaires pour le décrire, c'est-à-dire $r+1$; comme $b^r \leq a_r b^r < n \leq b^{r+1}$, on voit que

$$r \leq \frac{\log n}{\log b} < r + 1,$$

et on décrira donc cette complexité comme proportionnelle à $\log n$. Il est clair que la manipulation de nombres quelconques de taille n recquiert au moins $\log n$ opérations élémentaires.

On considère, tant d'un point de vue pratique que théorique, qu'un *bon* algorithme est un algorithme *polynomial*, c'est-à-dire utilisant $O\left((\log n)^\kappa\right)$ opérations élémentaires. Inversement, on considère qu'un algorithme *exponentiel*, c'est-à-dire ayant un temps d'exécution ou requérant un nombre d'opérations supérieur à $\exp(\kappa \log n) = n^\kappa$, est impraticable (pour n grand, bien sûr).

Addition. Pour additionner deux nombres m et n avec au plus r chiffres, on doit faire au plus r additions de deux chiffres et (éventuellement) propager une retenue. Le coût est donc $O\left(\log \max(n, m)\right) = O(r)$. Le coût d'une soustraction est similaire.

Multiplication. Pour calculer $n \times m$, où n et m sont deux nombres avec au plus r chiffres (avec l'algorithme appris à l'école primaire) on effectue au plus r^2

[1] Par exemple $\overline{1001}^{10}$ désigne le nombre écrit usuellement 1001 alors que $\overline{1001}^{2}$ désigne le nombre écrit usuellement $2^4 + 1 = 7$.

multiplications élémentaires et r additions, éventuellement avec retenues, d'où un coût en $O\left((\log\max(n,m))^2\right) = O\left(r^2\right)$.

Division euclidienne. Etant donné a et $b \geq 1$, si on calcule (q,r) tels que $a = qb + r$ et $0 \leq r \leq b-1$ avec (une variante de) l'algorithme appris à l'école primaire, on utilise un nombre d'opérations élémentaires similaire à celui de la multiplication, i.e. $O(\log\max(a,b)^2)$.

Algorithme d'Euclide (étendu). Il s'agit, étant donné deux entiers a, b de calculer $d := \mathrm{pgcd}(a,b)$ et $(u,v) \in \mathbf{Z}^2$ tel que $au + bv = d$ (théorème de Bézout). Rappelons le principe : on effectue la division de a par b, disons $a = bq_1 + r_1$; puis la division de b par r_1, disons $b = r_1q_2 + r_2$ et plus généralement la division de r_n par r_{n+1}, disons $r_n = r_{n+1}q_{n+2} + r_{n+2}$; on observe que la suite r_n est strictement décroissante et on s'arrête quand disons $r_{n+1} = 0$, et alors $\mathrm{pgcd}(a,b) = r_n$. En effet,

$$\mathrm{pgcd}(a,b) = \mathrm{pgcd}(b,r_1) = \mathrm{pgcd}(r_1,r_2) = \cdots = \mathrm{pgcd}(r_n,r_{n+1}) = r_n.$$

Pour le calcul de (u,v) on peut procéder ainsi : on pose $u_0 = 1$, $u_1 = 0$, $v_0 = 0$ et $v_1 = 1$ puis définir par récurrence $u_n = u_{n-2} - q_nu_{n-1}$ et $v_n = v_{n-2} - q_nv_{n-1}$. On vérifie immédiatement par récurrence que $au_n + bv_n = r_n$. Comme nous l'avons déjà vu, le nombre maximal de divisions euclidiennes à effectuer est $O(\log\max\{|a|,|b|\})$. Le coût total est donc $O\left(\log\max\{|a|,|b|\}^3\right)$.

Calculs modulo N (dans $\mathbf{Z}/N\mathbf{Z}$). Il s'agit de faire les opérations d'addition et de multiplication sur deux entiers inférieurs à N puis de prendre le reste pour la division euclidienne par N. Pour calculer l'inverse de a modulo N, on procède ainsi : si a est un entier, l'algorithme d'Euclide étendu nous répond soit que $\mathrm{pgcd}(a,N) > 1$ – auquel cas a n'est pas inversible modulo N – soit qu'il existe u, v (calculés par l'algorithme) tels que $au + Nv = 1$ et alors l'inverse de a est la classe de u modulo N. Le coût est donc le même que celui de l'algorithme d'Euclide étendu.

Exponentiation. Pour calculer a^m on pourrait bien sûr naïvement calculer $a\times a\times\cdots\times a$, mais cela obligerait à effectuer $m-1$ multiplications ; on peut faire beaucoup mieux et effectuer le calcul avec $O(\log m)$ multiplications. Par exemple, si $m = 2^r$ on élevera r fois au carré, i.e. on effectuera r multiplications. Dans le cas général, on écrira m en binaire $m = \epsilon_0 + \epsilon_1 2 + \cdots + \epsilon_r 2^r$ et on calculera

$$a^m = \left(\left(\left(a^{\epsilon_r}\right)^2 a^{\epsilon_{r-1}}\right)^2 a^{\epsilon_{r-2}}\cdots\right)^2 a^{\epsilon_0}.$$

On peut faire le calcul en sens inverse; l'algorithme peut être décrit itérativement. On part à cet effet des données initiales $(u,v,n) := (1,a,m)$ et on itère ainsi : si n est pair, on remplace (u,v,n) par $(u,v^2,n/2)$ et si n est impair, on remplace (u,v,n) par $(uv,v^2,(n-1)/2)$; on s'arrête quand $n = 0$ et on a alors $u = a^m$.

Cela signifie que l'on itère la transformation :

$$(u,v,n) := \begin{cases} (u,v^2,n/2) & \text{si } n \text{ est pair} \\ (uv,v^2,\frac{n-1}{2}) & \text{si } n \text{ est impair} \end{cases}$$

Remarquons que la quantité uv^n est invariante par cette transformation; comme elle vaut a^m au début, on a bien à la dernière étape $u = uv^0 = a^m$.

Comme n est au moins divisé par deux à chaque pas, le nombre r d'étapes est tel que $2^r \leq m$, et l'on doit donc effectuer $O(\log m)$ multiplications. Si on calcule $\mod N$, on réduit chaque résultat $\mod N$, et on fait ainsi à chaque étape la multiplication d'entiers $\leq N$. Le coût total pour calculer $a^m \mod N$ est donc $O\left(\log m (\log N)^2\right)$.

1.3 Le système RSA

Construisons maintenant les fonctions f_A du système RSA. On choisit deux très grands nombres premiers p et q, on calcule $N := pq$ et on choisit également un entier c (de taille moyenne) premier avec $\phi(N) = (p-1)(q-1)$. La clef publique est alors (N, c); par contre, p et q sont secrets et on pose, pour a entier inférieur N,

$$f(a) := a^c \mod N.$$

Pour décoder, on calcule l'inverse d de c modulo $\phi(N)$ et on observe que

$$f^{-1}(b) = b^d \mod N,$$

puisque, d'après le théorème d'Euler

$$(a^c)^d = a^{cd} \equiv a \mod N,$$

car $a^{\phi(N)} \equiv 1 \mod N$.

Ce système présente un grand nombre de qualités.

- Le chiffrement, tout comme le déchiffrement s'effectue très rapidement : il s'agit en effet d'effectuer une exponentiation modulo N.

- Connaissant la factorisation de N, on connait $\phi(N) = (p-1)(q-1)$, le calcul de l'inverse de c modulo $\phi(N)$ se fait par application de l'algorithme d'Euclide-Bézout et est donc très rapide.

- Casser le code (i.e. trouver "d") est essentiellement aussi difficile que factoriser N : plus précisément on ne dispose pas de méthode pour trouver d qui ne passe pas par le calcul de $\phi(N)$ et la connaissance de $\phi(N)$ équivaut (en terme de complexité) à celle de p et q : en effet, si l'on connait $\phi(N)$ et N, on calcule aisément $pq = N$ et $p + q = N + 1 - \phi(N)$, ainsi p et q sont des entiers racines de $X^2 + (\phi(N) - N - 1)X + N = 0$ et un calcul approché des racines de cette équation peut se faire très rapidement.

- Le système repose sur la possibilité de fabriquer facilement de grands nombres premiers "aléatoires" ou "quelconques" : nous allons voir au paragraphe suivant que cela est possible grâce aux "*tests de primalité*" . En gros on choisit un grand nombre au hasard, on teste s'il est premier, si la répnse est oui, on garde ce nombre, si la réponse est non on tire un nouveau nombre au hasard.

Remarque.

La sécurité du système RSA repose sur le fait que la factorisation est un problème "difficile". Cette affirmation, si elle est très bien vérifiée dans la pratique, n'est toutefois pas rigoureusement démontrée! En clair, on ne connait pas

d'algorithme polynomial de factorisation et on pense qu'il n'en existe pas, mais on ne sait pas non plus le démontrer[2]

1.4 Le système El Gamal

On choisit un (grand) nombre premier p ainsi qu'un générateur g du groupe cyclique $(\mathbf{Z}/p\mathbf{Z})^*$. Chaque intervenant(e) choisit une clef *secrète* disons a (resp. b) et calcule $A := g^a \mod p$ (resp. $B := g^b \mod p$). La clef publique de la première personne est (p, g, A), celle de la deuxième (p, g, B).

Pour envoyer à la première personne le mot m, la deuxième calcule $m' = mA^b \mod p$ et envoie (B, m'). Pour déchiffrer il suffit d'observer que $m = m'B^{-a} \mod p$; en effet

$$m'B^{-a} = mA^bB^{-a} = mg^{ab}g^{-ab} = m \mod p.$$

2 Primalité, factorisation

2.1 Tests de primalité

Théorème 1 *("petit" théorème de Fermat). Soit p un nombre premier et a qui n'est pas un multiple de p, alors $a^{p-1} \equiv 1 \mod p$.*

Exemple. Calculons (par exponentiation rapide) $2^{14} \equiv 4 \mod 15$; on peut en conclure (sans avoir factorisé 15 !) que 15 n'est pas premier.

Test de Fermat. Soit N un entier ≥ 2, pour un entier a non divisible par N on teste si la congruence suivante est vraie ou fausse :

$$F(a, N) \quad : \quad a^{N-1} \equiv 1 \mod N.$$

Si l'on trouve une congruence fausse, on peut affirmer que N n'est pas premier; si, pour un certain nombre de valeurs de a on trouve que la congruence est toujours vérifiée, on ne peut affirmer que N est premier, mais seulement qu'il est *probablement* premier.

On a vu que pour certain nombres composés – les nombres de Carmichaël par exemple – le test de Fermat n'est pas très performant; on peut assez simplement l'améliorer grâce à la variante suivante.

Théorème 2 *(Rabin-Miller). Soit p un nombre premier impair, décomposons $p - 1 = 2^s M$ avec M impair. Soit a qui n'est pas un multiple de p, alors*

$$a^M \equiv 1 \mod p \qquad \text{ou bien} \qquad \exists r \in [0, s-1],\ a^{2^r M} \equiv -1 \mod p.$$

[2]Il est peut-être aussi intéressant de signaler que si l'on change de monde informatique, la réponse peut être différente, ainsi, si l'on peut construire un "ordinateur quantique", alors on saura fabriquer une algorithme polynomial de factorisation ... et casser le système RSA (Shor, 1997).

Preuve. L'ordre de $a \mod p$ dans $(\mathbf{Z}/p\mathbf{Z})^*$ est un diviseur de $p-1 = 2^s M$ donc est de la forme $2^t L$ avec $t \leq s$ et L divisant M, c'est-à-dire $M = LL'$. Si $t = 0$ alors $a^M \equiv (a^L)^{L'} \equiv 1 \mod p$; si $t \geq 1$ observons que $a^{2^{t-1}L} \not\equiv 1 \mod p$ mais $\left(a^{2^{t-1}L}\right)^2 \equiv 1 \mod p$ donc $a^{2^{t-1}L} \equiv -1 \mod p$ et donc $a^{2^{t-1}M} \equiv \left(a^{2^{t-1}L}\right)^{L'} \equiv (-1)^{L'} \equiv -1 \mod p$. CQFD

Exemple. Calculons (par exponentiation rapide) $2^{14} \equiv 4 \mod 15$; on peut en conclure (sans avoir factorisé 15 !) que 15 n'est pas premier.

Test de Rabin-Miller. Soit N un entier ≥ 2; on décompose $N-1 = 2^s M$ avec M impair et $s \geq 1$. Pour un entier a non divisible par N on teste si l'une des congruences suivantes est vraie (ou si elles sont toutes fausses) :

$$T(a,N) \ : \ a^M \equiv 1 \mod N \qquad \text{ou bien} \qquad \exists r \in [0, s-1],\ a^{2^r M} \equiv -1 \mod N.$$

Si l'on trouve que $T(a,N)$ est fausse, on peut affirmer que N n'est pas premier; si, pour un certain nombre de valeurs de a on trouve que la congruence est toujours vérifiée, on ne peut affirmer que N est premier, mais seulement qu'il est *très probablement* premier.

Exemple. Soit $N = 529$, on décompose $N-1 = 528 = 2^4.33$ et on calcule $2^{33} \equiv 323 \mod 529$, $2^{66} \equiv 116 \mod 529$, $2^{124} \equiv 231 \mod 529$ et $2^{248} \equiv 461 \mod 529$. On conclut que 529 n'est pas premier (question subsidiaire : factoriser N).

Pour estimer la qualité du test de Rabin-Miller, introduisons le sous-ensemble de $(\mathbf{Z}/N\mathbf{Z})^*$ suivant:

$$S := \left\{ a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^M = 1 \qquad \text{ou bien} \qquad \exists r \in [0, s-1],\ a^{2^r M} = -1 \right\}.$$

On peut montrer le théorème suivant.

Théorème 3 *Soit N un entier impair et $S \subset (\mathbf{Z}/N\mathbf{Z})^*$ l'ensemble défini ci-dessus.*

1. *On a $S = (\mathbf{Z}/N\mathbf{Z})^*$ si et seulement si N est premier.*
2. *Lorsque N n'est pas premier et $N \neq 9$ on a $|S| \leq \phi(N)/4$.*

Preuve (du premier point) Il faut montrer que si N est composé alors il existe un entier modulo N qui n'est pas dans S. Si $N = p^r$ avec $r \geq 2$, on peut choisir a, élément d'ordre p^{r-1} et remarquer que $\text{pgcd}(N-1, p^{r-1}) = 1$ donc $a \notin S$. Si N est divisible par deux premiers distincts, disons p et q, choisissons a tel que $a \equiv 1 \mod p$ et $a \equiv -1 \mod q$ alors $a^M \equiv -1 \mod q$ donc $a^M \not\equiv 1 \mod N$ mais $a^L \equiv 1 \mod p$ donc $a^{2^r M} \not\equiv -1 \mod N$ et $a \notin S$.

Nous n'allons pas montrer le deuxième point mais illustrer comment, si l'on connait la factorisation de N, on peut calculer le cardinal de S (cela permet sur chaque exemple de vérifier le point).

Lemme 1 *Soit $N = p_1^{m_1} \dots p_k^{m_k}$ un nombre impair.*

$$\left|\left\{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^L = 1\right\}\right| = \prod_{i=1}^{k} \operatorname{pgcd}\left(L, p^{m_i-1}(p_i - 1)\right).$$

Lemme 2 *Soit $N = p_1^{m_1} \dots p_k^{m_k}$ un nombre impair. Soit $b \in$, alors le cardinal de l'ensemble $\left\{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^L = b\right\}$ est soit zéro soit $\prod_{i=1}^{k} \operatorname{pgcd}\left(L, p^{m_i-1}(p_i - 1)\right)$.*

Lemme 3 *Soit $N = p_1^{m_1} \dots p_k^{m_k}$ un nombre impair. Décomposons $N-1 = 2^s M$ et $p_i - 1 = 2^{s_i} M_i$ avec M et M_i impair, la congruence $a^{2^r M} \equiv -1 \mod N$ possède une solution si et seulement si $r \leq \min_i(s_i) - 1$.*

Les trois lemmes permettent de calculer le cardinal de l'ensemble S de la manière suivante. Introduisons les ensembles

$$S_j = \left\{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^{2^j M} = 1\right\} \quad \text{et} \quad T_j = \left\{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^{2^j M} = -1\right\}.$$

On peut alors écrire S comme une réunion disjointe

$$S = S_0 \cup T_0 \cup T_1 \cup \dots \cup T_{s-1}.$$

Posons $r = \min(s_i) - 1$, on conclut que, d'après les lemmes 2 et 3

$$|S| = |S_0| + |T_0| + |T_1| + \dots + |T_{s-1}| = |S_0| + \sum_{j=0}^{r} |S_j|,$$

et on conclut le calcul en utilisant le lemme 1.

Exemples. Prenons $N = 629 = 17.37$. On a $N - 1 = 2^2.157$ (i.e. $s = 2$ et $M = 157$) alors que $p_1 - 1 = 2^4$ (i.e. $s_1 = 4$ et $M_1 = 1$) et $p_2 - 1 = 2^2.3^2$ (i.e $s_2 = 1$ et $M_2 = 9$). On trouve que $S = S_0 \cup T_0 \cup T_1$, que $|S_0| = |T_0| = \operatorname{pgcd}(157, 16)\operatorname{pgcd}(157, 36) = 1$ et que $|T_1| = \operatorname{pgcd}(2.157, 16)\operatorname{pgcd}(2.157, 36) = 2.2 = 4$. Ainsi $|S| = 1 + 1 + 4 = 6$. Remarquons que $|S|/\phi(N) = 6/(16.36) = 1/96$ donc la "probabilité" qu'un test de Rabin-Miller $T(a, 629)$ ne détecte pas la non primalité est $\leq 1/96$.

Prenons $N = 8911 = 7.19.67$. On a $N - 1 = 2.4455$ donc $M = 3^4.5.11$, alors que $p_1 - 1 = 2.3$, $p_2 - 1 = 2.9$ et $p_3 - 1 = 2.33$ donc $S = S_0 \cup T_0$ et $|S_0| = |T_0| = \operatorname{pgcd}(M, 2.3)\operatorname{pgcd}(M, 2.9)\operatorname{pgcd}(M, 33) = 3.9.33$ donc $|S| = 2.3.9.33$. Comme $\phi(N) = 6.18.66$ on voit que

$$\frac{|S|}{\phi(N)} = \frac{2.3.9.33}{6.18.66} = \frac{1}{4}.$$

Preuves. Le lemme 1 vient de ce que $(\mathbf{Z}/N\mathbf{Z})^* \cong (\mathbf{Z}/p_1^{m_1}\mathbf{Z})^* \times \dots (\mathbf{Z}/p_k^{m_k}\mathbf{Z})^*$ et le nombre de solution dans $(\mathbf{Z}/p_i^{m_i}\mathbf{Z})^*$ de l'équation $x^L = 1$ est égale à $\operatorname{pgcd}(L, (p_i - 1)p_i^{m_i-1})$. Le lemme 2 vient de ce que, ou bien la congruence n'a aucune solution, ou bien s'il existe a_0 solution, alors l'application $a \mapsto aa_0^{-1}$ fournit une bijection avec les solutions de $x^L = 1$. En fin pour le lemme 3, on observe que le groupe cyclique $(\mathbf{Z}/p_i^{m_i}\mathbf{Z})^*$ est d'ordre $(p_i - 1)p_i^{m_i-1} = 2^{s_i} M_i p_i^{m_i-1}$ donc l'équation $x^{2^r} \equiv -1 \mod p^{m_i}$ possède une solution si et seulement si $r \leq s_i - 1$.

2.2 Factorisation

Le problème de la factorisation (ou plus exactement de la rapidité avec laquelle on peut factoriser un grand nombre N) est essentiellement ouvert : on ne connait pas d'algorithme rapide (polynomial) et on pense plutôt qu'il n'en existe pas mais on ne sait pas le démontrer. L'algorithme naïf consistant à essayer la divisibilité par 2,3,5 etc est clairement exponentiel; on sait néanmoins construire des algorithmes *sous-exponentiels* ayant une complexité bornée par $\exp(c\sqrt{\log N})$ par exemple. Les meilleurs algorithmes connus (crible algébrique ou via les courbes elliptiques) sont assez compliqués; contentons-nous de décrire un algorithme simple qui, dans certains cas (et seulement dans certains cas) factorise rapidement.

Méthode par différence de carrés.

L'idée de départ peut être tracée encore une fois jusqu'à Fermat semble-t-il : si un nombre impair se factorise $N = s.t$ avec disons $t < s$ alors en posant $a := \frac{s+t}{2}$ et $b := \frac{s-t}{2}$ on aura $N = (a+b)(a-b) = a^2 - b^2$; on peut donc chercher à écrire N comme différence de deux carrés. Pour cela on calcule (avec une décimale) $\sqrt{N}$ et on choisit x_1 le plus petit entier supérieur à $\sqrt{N}$ et on pose $z_1 := x_1^2 - N$. Si z_1 est un carré, on a gagné; sinon on essaie $x_2 = x_1 + 1$ et $z_2 := x_2^2 - N$ que l'on calcule par la formule $z_2 = z_1 + 1 + 2x_1$. Si z_2 est un carré on a gagné, sinon on itère le procédé. Un moment de réflexion indiquera que l'on obtiendra rapidement une factorisation à condition qu'il existe une telle factorisation $N = s.t$ avec s relativement proche de t.

Exemple. Essayons de factoriser $N = 4187$.

On note que $64^2 = 4096 < 4187 < 65^2 = 4225$. On pose donc $x_1 := 65$ et on calcule $z_1 := 65^2 - N = 38$ qui n'est pas un carré. On pose $x_2 = 66$ et calcule alors $z_2 = z_1 + 1 + 2x_1 = 38 + 1 + 130 = 169$ qui (chance!) est un carré 13^2. On peut donc écrire

$$N = x_2^2 - z_2 = 66^2 - 13^2 = (66+13)(66-13) = 79.53.$$