Sur l'algorithme RSA

Le RSA a été inventé par Rivest, Shamir et Adleman en 1978. C'est l'exemple le plus courant de cryptographie asymétrique, toujours considéré comme sûr, avec la technologie actuelle, pour des clés suffisament grosses (1024, 2048 voire 4096 bits). D'ailleurs le RSA128 (algorithme avec des clés de 128 bits), proposé en 1978 par Rivest, Shamir et Adleman, n'a été "cassé" qu'en 1996, en faisant travailler en parallèle de nombreux ordinateurs sur internet.

Mais le concept de chiffrement asymétrique avec une clef publique était légèrement antérieur (1976). L'idée générale était de trouver deux fonctions f et g sur les entiers, telles que $f \circ g = \mathrm{I}d$, et telle que l'on ne puisse pas trouver f, la fonction de décryptage, à partir de g, la fonction de cryptage. L'on peut alors rendre publique la fonction g (ou clef), qui permettra aux autres de crypter le message à envoyer, tout en étant les seuls à connaître f, donc à pouvoir décrypter. On trouvera un exposé complet sur RSA dans [3].

1 Préliminaires d'arithmétique

Définition On appelle indicatrice d'Euler la fonction qui à un entier n fait correspondre le nombre d'entiers a premiers à n vérifiant $1 \leq a \leq n$. On note cette fonction φ.

Si p est un nombre premier, alors tout entier compris entre 1 et $p-1$ est premier à p, aussi $\varphi(p) = p-1$. On peut également calculer assez facilement $\varphi(p^n)$, toujours pour p premier et $n \geq 2$ entier : les nombres premiers à p^n sont exactement les nombres non multiples de p. Or entre 1 et p^n, il y a exactement p^{n-1} multiples de p. Donc $\varphi(p^n) = p^n - p^{n-1} = p^{n-1}(p-1)$.

Enfin, on va calculer la valeur de φ en un produit de deux nombres premiers distincts. Ceci nous servira en effet dans l'algorithme RSA.

Lemme 1.1 *Soient p et q deux nombres premiers distincts. Alors $\varphi(pq) = (p-1).(q-1)$*

Démonstration Pour calculer $\varphi(pq)$, il nous suffit de calculer le nombre d'entiers compris entre 1 et pq qui ne sont pas premiers à pq. Ce sont bien sûr les multiples respectifs de p et de q.

Or il y a exactement q multiples de p dans l'intervalle $[\![1\,;pq]\!]$, ainsi que p multiples de q. Attention : nous avons ainsi compté deux fois l'entier pq. Le nombre d'entiers non premiers à pq dans l'intervalle $[\![1\,;pq]\!]$ est donc $q+p-1$.

D'où $$\varphi(pq) = pq - (p+q-1) = (p-1).(q-1)$$

□

Théorème 1.2 *(Petit théorème de Fermat) Soit p un nombre premier. Si a est un entier non divisible par p, alors on a $a^{p-1} \equiv 1[p]$.*

Pour tout entier a on a donc $a^p \equiv a[p]$.

Démonstration Ce théorème se déduit immédiatement du théorème 1.3, cependant nous allons en donner une autre preuve, élémentaire et originale, dûe à Philippe Biane (CNRS, ENS).

Considérons l'ensemble des mots de p lettres sur un alphabet de a caractères. Il y en a exactement a^p.

Sous l'action du groupe des permutations circulaires, les orbites obtenues sont réduites à un élément pour ce qui est des mots constants (qui sont au nombre de a), et de cardinal p pour les autres (il faut appliquer p permutations circulaires successives sur les lettres d'un mot pour revenir au mot de départ.)

Toutes ces orbites définissent une partition de l'ensemble des mots. Le cardinal de l'ensemble est donc la somme des cardinaux de toutes ces orbites. Ce qui donne, si n désigne le nombre d'orbites non réduites à un point :

$$a^p = a + np$$

et donc p divise $a^p - a$, ce qui veut dire que l'on a bien $a^p \equiv a[p]$.

L'autre partie du théorème s'en déduit aussitôt : si l'on suppose de plus que p ne divise pas a, comme on sait que p divise $a^p - a = a\left(a^{p-1}-1\right)$, le lemme de Gauss nous assure qu'alors p divise l'autre facteur, c'est-à-dire $a^{p-1}-1$, et donc $a^{p-1} \equiv 1[p]$

□

N.B. La primalité de p semble ne pas intervenir dans cette preuve. En fait, c'est elle qui assure que les orbites non réduites à un seul élément sont de cardinal exactement p, c'est-à-dire que les p mots obtenus à partir d'un mot donné par cette opération de permutation circulaire sont tous distincts. Si p n'est pas premier, par exemple pour $p = 4$, notre mot non constant peut se décomposer en plusieurs blocs identiques. Ainsi le mot ABAB nous donnera une orbite à 2 éléments, qui sont ABAB et BABA. En revanche, pour p premier, on ne peut pas décomposer un mot en succession de blocks identiques. Un tel bloc aurait un cardinal qui diviserait p, donc égal à 1 ou p. Le cas où c'est 1 correspond aux mots constants.

Théorème 1.3 *Soit $n \in \mathbb{N}$ un entier fixé, soit a un entier premier à n. Alors $a^{\varphi(n)} \equiv 1[n]$.*

Démonstration 1 Ce résultat est essentiellement un résultat sur le groupe multiplicatif $(\mathbb{Z}/n\mathbb{Z})^*$: si a est premier à n, la classe de a modulo n est dans $(\mathbb{Z}/n\mathbb{Z})^*$ et son ordre divise l'ordre du groupe (théorème de Lagrange), c'est-à-dire $\varphi(n)$. Donc la classe de $a^{\varphi(n)}$ est la classe de 1. Pour un exposé plus complet, voir par exemple [2].

Démonstration 2 (Inspirée de [1]) Soient $r_1, \ldots, r_{\varphi(n)}$ une énumération des entiers compris entre 1 et n, premiers à n. nous allons montrer que la multiplication par a (modulo n), est une bijection de cet ensemble.

– Pour tout i, on a ar_i encore premier à n (sinon, il y aurait un diviseur premier commun, qui diviserait soit r_i, soit a, contredisant ainsi l'hypothèse, à savoir que a et r_i sont premiers à n).
– Pour i et j distincts, ar_i et ar_j ne sont pas congrus modulo n. Dans le cas contraire, on aurait un entier k tel que $ar_i - ar_j = kn$. De cette égalité, on tire que n divise le produit $a(r_i - r_j)$. Mais comme n et a sont premiers entre eux, le lemme de Gauss nous assure alors que n divise $r_i - r_j$. Ceci est impossible puisque r_i et r_j sont tous deux compris entre 1 et n, leur différence étant donc elle-même plus petite que n.
– Donc les entiers $ar_1[n], \ldots, ar_{\varphi(n)}[n]$ sont distincts deux à deux, premiers à n, et bien sûr au nombre de $\varphi(n)$. A permutation près, ce sont donc les mêmes que $r_1, \ldots, r_{\varphi(n)}$.

En particulier, le produit des r_i est le même que celui de leurs images (par la bijection définie par la multiplication par a). C'est-à-dire que :

$$\prod_{i=1}^{\varphi(n)} (ar_i[n]) \equiv \prod_{i=1}^{\varphi(n)} r_i$$

ce qui signifie également que :

$$a^{\varphi(n)} \prod_{i=1}^{\varphi(n)} r_i \equiv \prod_{i=1}^{\varphi(n)} r_i[n]$$

Autrement dit, n divise la différence :

$$a^{\varphi(n)} \prod_{i=1}^{\varphi(n)} r_i - \prod_{i=1}^{\varphi(n)} r_i = \prod_{i=1}^{\varphi(n)} r_i \left(a^{\varphi(n)} - 1\right)$$

Or le produit $\prod_{i=1}^{\varphi(n)} r_i$ est lui aussi premier à n (même argument que pour les produits ar_i, à répéter $\varphi(n) - 1$ fois). Le lemme de Gauss nous permet alors de conclure que n divise $a^{\varphi(n)} - 1$, c'est-à-dire le résultat attendu :

$$a^{\varphi(n)} \equiv 1[n] \qquad \square$$

2 L'algorithme RSA

2.1 Description du protocole

Le but du jeu est bien sûr de pouvoir transmettre un message codé, que seul le récepteur "officiel" puisse décrypter, c'est-à-dire qui ne puisse pas être décrypté par un tiers qui intercepterait ledit message. Nous appellerons Alice la destinatrice du message, et Bernard l'émetteur.

1. Alice génère deux gros nombres premiers p et q, ainsi qu'un gros nombre d premier avec le produit $w = (p-1)(q-1)$.
2. Alice calcule $n = pq$ et e tel que $de \equiv 1[w]$.
3. Alice diffuse n et e, garde d et oublie w.
4. Bernard crypte un message M par $\mathrm{M} \mapsto \mathrm{M}^e[n]$ et envoie le résultat à Alice.
5. Alice décode alors le message crypté par $\mathrm{C} \mapsto \mathrm{C}^d[n]$

Exemple : Voyons ce qui se passe si l'on prend pour les deux nombres p et q les valeurs 11 et 17. On a alors $n = 187$ et $w = (11-1)(17-1) = 160$. Comme $161 = 7 \times 23$, on peut prendre $e = 7$ et $d = 23$.

Alice va rendre public le couple $(187, 7)$.

Bernard veut transmettre à Alice un message codé plus petit que $n = 187$, mettons la date à laquelle ils vont faire une surprise à Cédric (par exemple, le 10), message qui ne doit pas être intercepté par ledit Cédric, bien sûr.

Bernard va donc calculer $10^7 = 187 \times 53475 + 175$, et envoyer le résultat 175 à Alice.

Alice va calculer le reste de la division euclidienne de 175^{23} par 187 :
– Elle calcule d'abord $175^2 = 30625 = 163 \times 187 + 144$, donc $175^2 \equiv 144[187]$.
– Ensuite, $144^2 = 20736 = 110 \times 187 + 166$, donc $175^4 \equiv 166[187]$.
– Puis, $166^2 = 27556 = 147 \times 187 + 67$ donc $175^8 \equiv 67[187]$.
– Et $67^2 = 4489 = 24 \times 187 + 1$ donc $175^{16} \equiv 1[187]$.
– Enfin, $175^{23} = 175^{16} \times 175^4 \times 175^2 \times 175$ donc

$$175^{23} \equiv 1 \times 166 \times 144 \times 175[187]$$

Or $166 \times 144 \times 175 = 4183200 = 22370 \times 187 + 10$.

Alice retrouve donc bien le message envoyé, à savoir 10.

2.2 Preuve de l'algorithme

Dans la suite, nous allons utiliser constamment la propriété des congruences suivante :

Pour tout quadriplet d'entiers (a, b, c, r), si $a \equiv b[c]$, alors $a^r \equiv b^r[c]$. Cette propriété se démontre aisément si l'on remarque que $a - b$ divise $a^r - b^r$...

Le but du protocole est bien sûr qu'Alice retrouve le message d'origine. Les transformations successives appliquées au message d'origine sont :

$$\mathrm{M} \mapsto \mathrm{M}^e[n] \mapsto (\mathrm{M}^e[n])^d[n]$$

– Si le message M est premier à n :
On a $(\mathrm{M}^e[n])^d[n] \equiv \mathrm{M}^{de}[n]$, et par hypothèse $de \equiv 1[w]$, c'est-à-dire que $de = 1 + kw$, avec k un entier.
Mais alors on peut appliquer le théorème 1.3 : M est premier à n donc $\mathrm{M}^{\varphi(n)} \equiv 1[n]$. Et d'après le lemme 1.1, comme on a $n = pq$, on sait que $\varphi(n) = (p-1)(q-1) = w$. On a donc :

$$\mathrm{M}^w \equiv 1[n], \text{ donc } \mathrm{M}^{de} = \mathrm{M}^{kw+1} \equiv \mathrm{M}.1^k[n] \equiv \mathrm{M}[n]$$

et donc on revient bien ainsi au message originel.
– Sinon, M non premier à $n = pq$, c'est-à-dire que M est multiple de p ou de q. Considérons le cas où M est de la forme $p^\alpha m$, avec m entier non multiple de p..
Comme $\mathrm{M} < n$, on peut affirmer que m est premier à n (sinon, M serait à la fois multiple de p et de q, donc plus grand que n !). Et on a :

$$\mathrm{M}^{de} = (p^\alpha m)^{de}[n] \equiv p^{\alpha de} * m^{de}[n] \equiv p^{\alpha de} * m[n]$$

d'après ce qui précède, appliqué à m.

Or $p^{\alpha de} \equiv p^\alpha [p]$ (c'est évident !) et $p^{\alpha de} \equiv p^\alpha [q]$: par le théorème 1.3, on a $p^{q-1} \equiv 1[q]$ et comme $de = kw + 1$, on obtient $p^{de} \equiv p.1^{k(p-1)}[q] \equiv p[q]$. En élevant la relation à la puissance α, il vient bien $p^{\alpha de} \equiv p^\alpha [q]$.
Mais alors la différence $p^{\alpha de} - p^\alpha$ est à la fois multiple de p et de q, donc multiple de pq en utilisant le lemme de Gauss. On obtient bien $p^{\alpha de} \equiv p^\alpha [n]$ et donc :

$$\mathrm{M}^{de} \equiv p^\alpha m[n] \equiv \mathrm{M}[n]$$

et Alice retrouve bien le message originel.

– Les nombres p et q jouant des rôles identiques, le cas où M est multiple de q est identique.

N.B. Au vu de ce qui précède, on serait tenté d'énoncer un corollaire au théorème 1.3 du type : "Pour tout entier a, on a $a^{\varphi(n)+1} \equiv a[n]$". C'est malheureusement faux lorsque n est multiple d'un carré : $\varphi(4) = 2$ et 2^2 n'est pas congru à 2 modulo 4. Une formulation juste, mais un peu lourde, serait :

"Si n est un entier non multiple d'un carré, alors pour tout entier a, on a $a^{\varphi(n)+1} \equiv a[n]$."

2.3 Force de l'algorithme

Malgré une apparente simplicité, le système RSA reste l'un des plus sûr. Jusqu'à très récemment, la plupart des gens s'accordaient sur l'idée que décoder une message sans connaître la clef était équivalent à factoriser l'entier n (*i.e.* trouver p et q). Un papier récent infirme cette conjecture.

D'autre part, il est très difficile dans la pratique de factoriser n : même s'il existe des méthodes beaucoup plus efficaces que le procédé naturel (tester tous les nombres impairs jusqu'à $\sqrt{n}$), le temps de calcul nécessaire reste incomparablement plus long que celui dont ont besoin Alice et Bernard.

Enfin, signalons que l'algorithme, bien qu'assez sûr, est assez lent. Dans la pratique, il sers le plus souvent à transmettre une clef servant à déchiffrer un message codé selon une autre méthode plus rapide, typiquement une méthode de chiffrement dit symétrique, par exemple IDEA : on code et on décode un message de la même façon, il est donc nécessaire de transmettre le message codé et la clef pour le décoder. C'est ce que fait par exemple PGP (pour Pretty Good Privacy), logiciel de mise en oeuvre de cryptographie publique (RSA) et symétrique (IDEA à une époque).

3 Mise en oeuvre pratique de RSA

Même si le protocole de RSA est assez simple, sa mise en oeuvre pose toutefois quelques problèmes pour Alice, notamment la construction de deux "gros" nombres premiers (p et q), ainsi que la détermination du couple (d, e). Enfin, les deux protagonistes se trouvent confrontés au problème d'élever de façon efficace un "gros" nombre à une "grosse" puissance, modulo n.

Nous ne parlerons pas ici du problème de la génération de "gros" nombres premiers, problème assez complexe qui mériterait que l'on y consacre un texte conséquent (à suivre ?).

3.1 Inversion modulo w

En réalité, Alice peut choisir sa première clef d de façon arbitraire. d doit simplement être premier à w. Mais une telle condition est assez facile à satisfaire, et encore plus à vérifier : il lui suffit de prendre un nombre au hasard, d'utiliser l'algorithme d'Euclide pour savoir s'il est premier à w, et de recommencer si tel n'est pas le cas. Statistiquement, Alice doit ainsi trouver assez rapidement un nombre d premier à w.

Elle doit ensuite trouver un inverse de d modulo w, c'est à dire un entier e tel que :

$$de \equiv 1[w], \text{ c'est-à-dire } de = 1 + kw \ (k \text{ entier}).$$

Mais on reconnaît dans la relation précédente ni plus ni moins qu'une relation de Bezout entre d et w. La méthode la plus classique pour ceci est ce que l'on appelle algorithme d'Euclide étendu, ou algorithme de Bezout. Il s'agit de "remonter" dans l'algorithme d'Euclide appliqué à w et d pour trouver cette relation :

Supposons que l'algorithme de Bezout mène à la suite de divisions euclidiennes successives :

$$\begin{aligned} w &= q_1 d + r_1 \\ d &= q_2 r_1 + r_2 \\ &\vdots \\ r_{n-2} &= q_n r_{n-1} + r_n \end{aligned}$$

où r_n, dernier reste non nul, est ici égal à 1 puisque w et d sont premiers entre eux.

Alors on va partir de la dernière équation pour écrire :

$$1 = r_{n-2} - q_n r_{n-1}$$

dans laquelle on peut remplacer r_{n-1} par l'expression $r_{n-3} - q_{n-1} r_{n-2}$ (on utilise ici l'avant dernière division euclidienne). On a ainsi obtenu une relation de Bezout entre r_{n-2} et r_{n-3}. Il suffit alors de continuer le procédé en remplaçant r_{n-2} à l'aide de l'antépénultième division euclidienne.

On obtient ainsi de proche en proche des relations de Bezout pour les couples d'entiers (r_{n-1}, r_n), puis $(r_{n-2}, r_{n-1}), \ldots$ et à la fin (w, d).

Tel quel, l'exposé de ce procédé nous explique comment déterminer un inverse de d modulo w, mais on voudrait de plus, pour des raisons pratiques, qu'il soit le plus petit possible. En effet, Alice doit pouvoir élever de gros nombres à la puissance e.

Si l'entier obtenu est supérieur à w, il suffit de prendre son reste modulo w (et bien sûr, le produit de reste inchangé modulo w, donc toujours congru à 1).

En réalité, cet algorithme nous permet de trouver un inverse e de d modulo w qui vérifie $|e| < w$. C'est presque la condition la plus forte que l'on puisse imposer à e : en effet il existe un unique inverse de d (modulo w) dans chaque intervalle de longueur w : si $de \equiv 1[w]$ et $de' \equiv 1[w]$, alors $d\,(e - e')$ est multiple de w. Comme d est premier à w, la différence $(e - e')$ est multiple de w d'après le lemme de Gauss. Mais si e et e' sont dans un même intervalle de longueur w, on obtient $|e - e'| < w$, et donc $e = e'$. En conclusion, si l'invers construit e

vérifie $|e| < w$, il ne reste que deux valeurs possibles de e (une positive, l'autre négative). Et il suffit de rajouter w si la valeur obtenue est négative.

Pour montrer que le coefficient de Bezout e obtenu par l'algorithme vérifie $|e| < w$, il nous faut conduire une récurrence un peu soigneuse.

Après i remplacements, on a une relation de la forme :

$$1 = u_i r_{n-i-2} + v_i r_{n-i-1}$$

Soit donc P (i) la propriété : $|u_i| < r_{n-i-1}$ et $|v_i| < r_{n-i-2}$.

Alors P (0) est vraie, car au rang 0 la relation est $1 = r_{n-2} - q_n r_{n-1}$, donc $u_0 = 1 < r_{n-1}$ et $v_0 = -q_n$, donc $|v_0| < r_{n-2}$ (puisque q_n est le quotient de la division de r_{n-2} par r_{n-1}).

Supposons P (i) vraie. Pour passer à l'étape $i+1$, on remplace l'entier r_{n-i-1} par sa valeur $r_{n-i-3} - q_{n-i-1} r_{n-i-2}$ dans l'équation $1 = u_i r_{n-i-2} + v_i r_{n-i-1}$. Ceci nous conduit à la relation :

$$1 = v_i r_{n-i-3} + (u_i - q_{n-i-1} v_i)\, r_{n-i-2}$$

En identifiant les coefficients, on obtient $u_{i+1} = v_i$ et $v_{i+1} = u_i - q_{n-i-1} v_i$. Mais alors, l'hypothèse de récurrence nous assure que $|v_i| < r_{n-i-2}$, et donc $|u_{i+1}| < r_{n-i-2}$.

D'autre part cette hypothèse de récurrence nous donne :

$$|q_{n-i-1} v_i| < q_{n-i-1} r_{n-i-2}$$

Et comme enfin l'autre partie de l'hypothèse de récurrence est la relation $|u_i| < r_{n-i-1}$, on obtient en sommant terme à terme :

$$|u_i - q_{n-i-1} v_i| \leq |u_i| + |q_{n-i-1} v_i| < q_{n-i-1} r_{n-i-2} + r_{n-i-1}$$

c'est-à-dire la majoration :

$$|v_{i+1}| < r_{n-i-3}$$

et l'on a bien établi P $(i+1)$. Par le principe de récurrence, on a bien pour tout i la propriété : $|u_i| < r_{n-i-1}$ et $|v_i| < r_{n-i-2}$. A la dernière étape de l'algorithme, ceci nous donne :

$$|e| < w \text{ et } |k| < d$$

N.B. Dans la pratique, ce procédé est un peu long : faire d'abord l'algorithme d'Euclide puis remonter pas à pas. Cela implique notamment de garder en mémoire toutes les divisions euclidiennes. Il est plus rapide de mener tous ces calculs en parallèle à chaque étape de l'algorithme d'Euclide, en utilisant les relations de récurrences des suites (u_i) et (v_i) : à chaque étape, dans l'expression du couple (u_{n-1}, v_{n-1}) recherché, on remplace les inconnues u_i et v_i par leurs expressions en fonction de u_{i-1} et v_{i-1} qui sont les nouvelles variables. A la fin du procédé, on connaît ainsi l'expression des entiers u_{n-1} et v_{n-1} comme combinaisons linéaires de u_0 et v_0.Et la dernière division euclidienne nous donne u_0 et v_0, ce qui permet de calculer $e = v_{n-1}$.

3.2 Exponentiation rapide

On l'a déjà signalé plus haut, un des problème de la mise en oeuvre de RSA est de pouvoir, pour Bernard, crypter M soit calculer $\mathrm{M}^d[n]$ en un temps

raisonnable, et pour Alice, décrypter C soit calculer $\mathrm{C}^e[n]$ rapidement. Ces deux problèmes sont bien sûr du même type.

Une méthode assez efficace pour calculer $a^k[n]$, où a et k sont deux entiers quelconques, consiste à décomposer l'entier k en base 2. Par exemple, pour élever à la puissance 17, plutôt que de multiplier 16 fois par a et de prendre le reste modulo n à chaque fois, on calcule a^2 modulo n, que l'on élève au carré (en deux étapes on a ainsi $a^4[n]$), et de répéter le procédé. Pour cet exemple, on a donc 4 étapes à effectuer pour obtenir $a^{16}[n]$, plus une cinquième pour arriver à $a^{17}[n]$. Cette méthode est d'autant plus efficace que l'on peut garder en mémoire les résultats intermédiaires lors du calcul du plus grand exposant binaire : pour trouver $a^{20}[n]$, on n'a là aussi besoin que de 5 étapes : lors du calcul de $a^{16}[n]$ on a calculé $a^4[n]$... (voir l'exemple donné au paragraphe 2.1)

De manière générale, si l'entier k se décompose sous la forme :

$$k = 2^{n_1} + 2^{n_2} + \ldots + 2^{n_r}, \text{ où } n_1 > n_2 > \ldots > n_r$$

le nombre d'étapes (multiplication puis reste modulo n) est de cette façon $n_1 + r - 1 < 2n_1$, ce qui est très peu devant k. En effet, on sait que $2^{n_1} < k$, donc $n_1 < \log_2(k)$. Et pour de grandes valeurs de k, ce qui est le cas dans RSA (le produit *de* est au moins $w+1$, donc au mois l'une de ces valeurs est grande), le temps de calcul gagné est considérable, puisque le rapport $\dfrac{\log_2(k)}{k}$ tends vers 0.

Références

[1] H.M. Stark, *An Introduction to number theory*, Markham Publ. Co., 1970

[2] D. Perrin, *Cours d'algèbre*, Ecole Normale supérieure, 1990.

[3] B. Schneier, *Cryptographie appliquée : protocoles, algorithmes et codes sources en C*, J. Wiley, 1997.