Codage RSA

Introduction

R.S.A. signifie Rivest-Shamir-Adleman, en l'honneur de ses inventeurs : Ron Rivest, Adi Shamir et Leonard Adleman qui l'ont inventé en 1977. Le brevet de cet algorithme appartenait jusqu'au 6 Septembre 2000 à la société américaine RSA Data Security, qui fait maintenant partie de Security Dynamics et aux Public Key Partners, (PKP à Sunnyvale, Californie, états-Unis). RSA est un algorithme à clé publique qui sert aussi bien au chiffrement de documents, qu'à l'authentification. Comme il est à clé publique et qu'il a résisté à toutes les attaques jusqu'à présent, l'algorithme RSA est devenu un standard *de facto* dans le monde.

Tout le principe de RSA repose sur le fait qu'il est très difficile et long de décomposer un très grand nombre en deux facteurs premiers.

Exemple

RSA est un algorithme de chiffrement asymétrique, ce qui signifie que le codage et le décodage sont réalisés par des opérations différentes.

Le codage RSA repose sur la *factorisation* d'un entier. Soient *p* et *q* deux entiers, il est facile de calculer leur produit *n* = *pq*. Toutefois, étant donné *n*, il est difficile de retrouver *p* et *q* (i.e. de factoriser *n*).

Le principe du codage par factorisation restant le même quelle que soit la taille du nombre, nous allons nous contenter ici d'un ``tout petit'' nombre, qui constituerait un gros secret pour un enfant mais pas pour un espion. Dans la pratique, on utilise des entiers de plus d'une centaine de chiffres.

Le colonel Alice, adepte du chiffrement RSA, a donné à tous ses agents la clé *n*=187 et le facteur *e*=7. *n* est un nombre composé par le produit des deux facteurs premiers 17 et 11 (17*11=187). On considère l'indicateur d'Euler $\varphi(n) = (p-1) * (q-1) = 16 * 10 = 160$. On

choisit alors le nombre *e* tel que *e* et $\varphi(n)$ soient premiers entre eux[1]. Le colonel a pris *e*=7.

Il doit maintenant déterminer sa clé privée (qui ne sera jamais divulguée). Il va chercher le plus petit *m*>0 tel que $m\varphi(n) + 1$ soit divisible par *e*=7. Pour *m*=1, on

a $m.\varphi(n) + 1 = 1 * 160 + 1 = 161 = 7 * 23$. Sa clé privée est alors d=23.

Le colonel Alice envoie donc à tous ses honorables correspondants la clé (*n*=187, *e*=7), appelée *clé publique*. Il garde pour lui seul *d*=23, appelée *clé privée*.

Justement, Bob doit passer le voir sous peu et veut lui dire ``arrive le 17'', un message ultra-secret qu'il va coder avec la clé publique (*n*, *e*). Pour commencer il convertit son texte en chiffres en prenant par exemple *a*=01, *b*=02, ..., espace=00, 0=30, 1=31, 2=32, ..., ce qui lui donne

011818092205001205003137

Il regroupe ce nombre en tranches ayant moins de chiffres que la clé *n*=187, soit en tranches de deux chiffres :

01 18 18 09 22 05 00 12 05 00 31 37

Chacun de ces nombres représente la variable texte ou *t*[2].

Il va maintenant effectuer son codage par l'opération

$$c \equiv t^7 \bmod 187$$

pour obtenir la succession

1 171 171 70 44 146 0 177 146 0 47 125 181

Ajoutant des 0 si nécessaire pour n'avoir que des nombres de trois chiffres comme la clé *n*, il note

001 171 171 070 044 146 000 177 146 000 047 125 181.

Le message que Bob envoie au colonel Alice est finalement la suite de chiffres :

001171171070044146000177146000047125181,

Celui-ci possède la clé *d*=23, va maintenant utiliser la congruence

$$t \equiv c^{23} \bmod 187$$

après avoir découpé le message en tranches de trois chiffres puisque la clé *n*=187 en a trois. Il va ainsi retrouver 01 pour 001, 18 pour 171 - car $18 \equiv 171^{23} \bmod 187$ - et ainsi de suite. Il ne lui reste plus qu'à prendre son tableau de correspondance alphabétique pour recueillir ``arrive le 17''. Bien entendu, codage et décodage se font sur ordinateur vu la longueur des opérations.

L'adversaire, bien qu'il connaisse la clé publique (*n*=187,*e*=7), ne peut décrypter le message car il lui faudrait avoir la clé secrète *d*=23. Et pour avoir celle-ci, il devrait décomposer *n* en facteurs premiers, ce qui est certes facile avec 187, mais impossible actuellement - à moins d'y consacrer des siècles ou de recourir à la divination - avec un nombre de 150 à 200 chiffres.

En contrepartie, le calcul d'une puissance modulo *n* est à la portée de tout ordinateur, ou même d'une calculatrice de poche programmable. Pour le moment, la sécurité du cryptage RSA repose sur la puissance des ordinateurs et sur les connaissances en arithmétique. Tout progrès dans un domaine comme dans l'autre peut obliger à allonger la clé, ou à chercher une autre fonction qui soit facile dans un sens, et quasiment impraticable dans l'autre.

Principe de Fonctionnement

Le schéma suivant résume le principe du chiffrement RSA:

Soit *n* un entier tel que *n*=*pq* :

1. Génération des clés :
   - clé publique : (*n*, e) tel que *e* soit premier avec $\varphi(n)$
   - clé privée : $d = e^{-1} \bmod \varphi(n)$
2. Chiffrement du message : soit *m* un message, son chiffré *c*est obtenu par $c = m^e \bmod n$
3. Déchiffrement du message codé : *m* est retrouvé à l'aide de la clé secrète par $m = c^d \bmod n$.

Génération des clés

Clé publique

1. Choisir deux nombres premiers *p* et *q* très grands (de l'ordre de 100 chiffres). Il existe pour cela des algorithmes de génération aléatoire de nombres premiers. On pose *n*=*pq* ;
2. Trouver un entier *e* entre 2 et $\varphi(n) = (p-1)(q-1)$ (fonction indicatrice d'Euler, c'est en fait le nombre d'entiers inférieurs à *n* qui sont premiers avec lui) tel que *e* et $\varphi(n)$ soient premiers entre eux.

Les nombres *n* et *e* forment la clé publique avec laquelle n'importe qui pourra crypter un message. On la notera (*n*,*e*).

Clé privée

Il nous faut maintenant calculer le nombre *d* qui sera nécessaire au déchiffrement. Selon la théorie de RSA, nous devons avoir *d* tel que *e**d-1 soit divisible par $\varphi(n)$ (i.e $\exists k$ tel que $ed-1 = k\varphi(n)$), soit :

$$\text{trouver } d \text{ et } k \in Z \text{ tel que } ed + k\varphi(n) = 1$$

Or, comme *e* et $\varphi(n)$ sont premiers entre eux, le théorème de Bezout prouve l'existence de *d* et *k* dans *Z*. Ceci signifie donc que $d \equiv e^{-1} \bmod \varphi(n)$.

Nous voilà prêts à décrypter. Le nombre *d* est notre clé privée. Nous pouvons à présent divulguer la clé publique (*n*,*e*) et garder la clé privée. Quant aux nombres *p*, *q*, et $\varphi(n)$, on doit soit les conserver secrets, soit les détruire car ils ne serviront plus.

Chiffrement avec l'algorithmeRSA

D'après la description initiale, il semble que l'ensemble des lettres en clair $\mathcal{P}$ et des chiffrés $\mathcal{C}$ soient égaux.

En pratique, on travaille sur des blocs de lettres. Si on considère un alphabet de *N* lettres[3], soient $k < l$, tels que N^k et N^l comportent environ 200 chiffres. Un bloc de *k* lettres est un nombre de *k* chiffres pour les entiers en base *N*. Par exemple, la phrase ``arrive le 17'' se décompose de la manière suivante :

$$0.N^{11}+18.N^{10}+18.N^9+9.N^8+22.N^7+5.N^6+0.N^5+12.N^4+4.N^3+0.N^2+27.N^1+33.N^0,$$

avec $a \to 0, \ldots, z \to 18, 0 \to 26, \ldots, 9 \to 35,$ espace $\to 36$. De manière équivalente, on considère le chiffré comme un bloc de *l* chiffres pour les entiers en base *N*.

Les nombres *p* et *q*, tels que *n*=*pq*, doivent alors vérifier :

$$N^k < n < N^l$$

Exemple : L'agent Bob se sert de la nouvelle clé publique du colonel Alice (*n* = 4717, *e* = 5155). Le colonel impose maintenant de coder des blocsde *k* = 2 lettres en blocs de *l*=3 lettres avec un alphabet de *N* = 47 lettres. Bob veut encore dire à Alice qu'il ``arrive le 17''. On obtient donc comme codage des digraphes :

digraphe	valeur du digraphe	chiffré	valeur du trigraphe	trigraphe
``ar''	17	2126	11450	ab3
``ri''	807	681	23140	bw9
``ve''	991	2707	28101	b5u
`` l''	1703	3686	20311	bk2

``e ``	224	3278	35221	aox
``17''	1302	76	2910	a*l

Le colonel Alice reçoit donc comme message ``ab3bw9b5ubk2aoxa*l''.

Déchiffrement avec l'algorithmeRSA

Le déchiffage se fait de la même manière que le chiffrement. On commence par convertir le message en une suite de nombre exprimés sur la bonne base (N^l), puis on inverse ce nombre en se servant de la clé secrète :

$$t \equiv c^d \bmod n,$$

où *c* est le texte chiffré et *d* la clé secrète. On obtient donc le ``texte clair'' *t*, qui est un bloc de lettres exprimé sur la base N^k. Il ne reste donc plus qu'à l'exprimer avec l'alphabet choisi.

Exemple : Reprenons l'exemple précédent. Le colonl Alice a reçu le message ``ab3bw9b5ubk2aoxa*l'' de la part de Bob. Alice commence par découper son message en trigraphe (*l*=3), puis utilise sa clé secrète *d*=4331. Enfin, elle convertit le nombre obtenu sur $47^2$ et obtient des digraphes qui, une fois assemblés, dévoilent le message ``arrive le 8''.

Remarques avancées

Choix des entiers

L'efficacité de cet algorithme repose sur la difficulté à factoriser 2 entiers. Toutefois, le choix de ces 2 entiers n'est pas sans conséquence. Continuons à noter *n*=*pq* où *p* et *q* sont des nombres premiers.

Quelques précautions doivent être prises :

- *p* et *q* ne doivent pas être trop proches ;
- le pgcd de *p*-1 et *q*-1 doit être assez petit alors qu'ils doivent avoir en commun un facteur premier assez grand ;

Rappelons également que la connaissance de la paire $(n, \varphi(n))$ (où $\varphi(n)$) représente la

fonction indicatrice d'Euler) est équilante à la connaissance de la paire (*p*, *q*). En effet, si *n* est pair, alors, *p*=2 et *q*=*n*/2. Considérons donc

En effet, si on connaît (*p*, *q*), on calcule facilement *n*=*pq* et $\varphi(n) = (p-1)(q-1)$,

Inversemment, en connaissant *n* et $\varphi(n)$, on a alors :

$$\begin{aligned} \varphi(n) &= (p-1)(q-1) \\ &= pq-(p+q)+1 \end{aligned}$$

D'où :

$$\begin{aligned} pq &= n \\ p+q &= n+1-\varphi(n) \end{aligned}$$

Donc, *p* et *q* sont les racines de l'équation (on pose 2*b*=*p*+*q*) :

x^2-2bx+n= 0

Soit :

$$p = b + \sqrt{b^2 - n}$$
$$q = b - \sqrt{b^2 - n}$$

Un algorithme probabiliste

Une connaissance plus faible que celle de la paire $(n, \varphi(n))$ peut suffir pour *casser* RSA.

On sait que *n* est le produit de 2 nombres premiers. Supposons connu un entier *m* tel que :

$$\forall a, \text{ avec } \gcd(a,n) = 1, \text{ tel que } a^m \equiv 1 \bmod n \qquad (1)$$

On remarque que *m* est pair (il suffit de le voir pour *a*=-1). On vérifie alors si (1) est encore vérifiée pour *m*/2.

Soient a_1 et a_2 tels que a_1 vérifie (1) mais pas a_2. La produit $a_1 a_2$ ne vérifie pas (1). Supposons donc qu'on trouve un nombre *b* pour lequel (1) est faux, et soit $E = \{a_0, \ldots, a_k\}$ l'ensemble des nombres vérifiant (1). Les éléments de l'ensemble $bE = \{ba_0, \ldots, ba_k\}$ ne vérifient pas (1). L'équation (1) est donc vérifiée pour, au plus, 50% des nombres.

Donc, si on teste (1) pour plusieurs *a*, on obtient une probabilité assez élévée que la propriété soit valide pour tous les nombres *a* premiers avec *n*. Divisons alors m par 2 et testons à nouveau. On continue ainsi jusqu'à ne plus avoir la propriété vérifiée. Deux cas se présentent alors :

1. $\frac{m}{2}$ est un multiple de *p*-1 ou *q*-1 (disons *p*-1) mais pas des 2. On a alors toujours $a^{\frac{m}{2}} \equiv 1 \bmod p$ (théorème des restes chinois et petit théorème de Fermat) et exactement 50% du temps $a^{\frac{m}{2}} \equiv -1 \bmod q$ (au lieu de 1 car les seules racines carrées de 1 sont -1 et 1 modulo un nombre premier) ;
2. $\frac{m}{2}$ n'est un multiple ni de *p*-1, ni de *q*-1. On a donc $a^{\frac{m}{2}} \equiv 1$ modulo *p* et *q* dans 25% des cas, $a^{\frac{m}{2}} \equiv -1$ modulo *p* et *q* dans 25% des cas et pour les 50% valuers restantes possibles de *a*, on a $\equiv 1$ pour l'un et $\equiv -1$ modulo l'autre.

Donc, en essayant aléatoirement des nombres *a*, on doit finir par en trouver un tel que $a^{m/2}$ - 1 soit divisible par *p* ou *q* (disons *p*). Chaque *a* testé a 50% de chances de vérifier cette propriété. Une fois un tel *a* découvert, on connaît un facteur de *n* puisque $\gcd(n, a^{m/2} - 1) = p$.

Footnotes

... eux[1]

Ceci signifie que $\gcd(n,e)=1$, i.e. aucun diviseur de *n* divise *e* et réciproquement.

...*t*[2]

En fait, il faut que *t*<*N*, ce qui est garantie quand *t* comporte moins de chiffres que *n*.

... lettres[3]

Soit $n=\alpha_p\alpha_{p-1}\ldots\alpha_1\alpha_0$ un nombre exprimé en base *B* (i.e.

$\forall i, 0\leq i\leq p, 0\leq \alpha_i < B$), $n=\alpha_p B^p+\alpha_{p-1}B^{p-1}+\ldots+\alpha_1 B+\alpha_0$