PEUT-ON AVOIR CONFIANCE EN LE COMMERCE ÉLECTRONIQUE ?

MICHEL RIGO

1. Résumé

Pour de bonnes ou de mauvaises raisons, l'homme a toujours eu besoin d'échanger des messages secrets. Avec l'avénement du e-commerce, les paiements à distance sont de plus en plus fréquents. Se pose donc la question suivante : peut-on sans crainte transmettre son numéro de carte de crédit sur Internet ? Quels sont les concepts mathématiques sous-jacents à une telle transaction ?

2. Introduction à la cryptographie

Si on se réfère au dictionnaire, on peut dire que la cryptographie est l'ensemble des principes, méthodes et techniques dont l'application assure le chiffrement et le déchiffrement de données afin d'en préserver la confidentialité et l'authenticité.

Les principales applications de la cryptographie sont de quatre types :

(**a**) assurer la *confidentialité* des données transmises (rendre possible des communications secrètes et ce, même si une tierce personne intercepte ces communications),

(**b**) ne pas permettre la *répudiation* d'un message envoyé (signature d'un message),

(**c**) préserver l'*intégrité* du message transmis (être certain que le message n'a pas été altéré par une tierce personne),

(**d**) permettre l'*authentification* (par exemple, accès par mot de passe).

Les applications de la cryptographie sont nombreuses : l'armée, les transactions bancaires, le commerce électronique, la téléphonie mobile, la télévision à la carte, les cartes d'identité électroniques, le vote électronique, ...

Exemple. Prenons l'exemple de Bob, champion de musculation, qui désire acheter un livre de poésie sur un site de vente en ligne tenu par Alice. Bob ne veut pas que ses amis de la salle de musculation et surtout pas son éternel rival Oscar puissent découvrir son penchant pour la

M.Rigo@ulg.ac.be, http://www.discmath.ulg.ac.be/
Université de Liège, Institut de Mathématiques,
Grande traverse 12 (B37), 4000 Liège.

poésie ! Il faut donc pouvoir assurer que sa commande soit uniquement connue d'Alice et qu'Oscar ne puisse pas découvrir le contenu de celle-ci en espionnant le trafic Internet (**a**). Alice veut quant à elle être certaine que la commande vient bien de Bob et que celui-ci ne retournera pas le livre (disons, après l'avoir photocopié) en prétextant qu'il n'a jamais rien commandé (**b**). Il faut également être certain qu'Oscar n'a pas intercepté et modifié la commande de Bob, par exemple, en changeant l'adresse de livraison ou le contenu de la commande (**c**). Enfin, Bob a la possibilité de consulter en ligne le suivi de sa commande. Pour cela, lui seul doit pouvoir s'identifier sur le site de vente d'Alice pour avoir accès à ses données personnelles (**d**).

3. Les deux types de chiffrement

On distingue deux types de chiffrement : le chiffrement à clé secrète et le chiffrement à clé publique. Ils ont tous deux leurs avantages et leurs inconvénients. Il est dès lors souvent pratique de les combiner dans les applications réelles.

Dans le *chiffrement à clé secrète*, pour communiquer secrètement, Alice et Bob doivent partager une clé commune connue d'eux seuls. Cette clé e sert tant au chiffrement qu'au déchiffrement de messages.

$$\text{“bonjour”} \xrightarrow{e} \text{“?91aa2x”} \xrightarrow{e} \text{“bonjour”}$$

Cela pose un problème parfois insoluble : comment échanger cette clé sans être espionné par Oscar. En effet, il faudrait par exemple qu'Alice et Bob se rencontrent pour se mettre d'accord sur la clé. Ceci n'est pas envisageable lors de connexions Internet. A titre d'exemple, pour le téléphone rouge, on utilisait une valise diplomatique. La sécurité d'un tel système repose intégralement sur la clé. Les systèmes à clé secrète les plus connus sont le DES, le triple DES[1], l'AES[2] (successeur de DES) et le masque jetable. Les trois premiers sont essentiellement basés sur un mélange des bits du message avec les bits de la clé (c'est en tout cas l'idée).

Les *systèmes à clé publique* évite cet écueil mais sont souvent (beaucoup) plus lents que les systèmes à clé secrète. Nous donnerons plus loin un aperçu détaillé du système RSA mais il existe aussi d'autres systèmes basés sur des courbes elliptiques. L'idée est la suivante. Alice possède deux clés, une clé publique e_A connue de tous et publiée sur son site de vente en ligne. Cette clé est utilisée par les acheteurs et elle sert à chiffrer les messages qui lui sont destinés. La seconde clé d_A est

[1]Plusieurs banques belges utilisent le triple DES dans leurs applications de “home-banking”.

[2]mis au point par deux Belges.

quant à elle conservée secrètement par Alice. Elle lui sert à déchiffrer les messages chiffrés avec e_A.

$$\text{“bonjour”} \xrightarrow{e_A} \text{“a1x\# z !a”} \xrightarrow{d_A} \text{“bonjour”}$$

Toute l'astuce des systèmes à clé publique réside dans le fait qu'il est “pratiquement” impossible de retrouver la clé d_A à partir de e_A. Ainsi, tout le monde peut envoyer des messages chiffrés à Alice et elle seule pourra les déchiffrer !

En pratique, on utilise un chiffrement à clé publique pour échanger secrètement une clé qui sera utilisée par la suite pour un chiffrement à clé secrète. Ainsi, on combine l'avantage certain du chiffrement à clé publique avec la rapidité du chiffrement à clé secrète.

4. RSA

Ce cryptosystème du nom de ses concepteurs Rivest, Shamir et Adleman est un système à clé publique. Grossièrement, il repose sur le fait qu'il est facile de calculer le produit de deux (grands) nombres premiers alors qu'il est beaucoup plus long d'effectuer l'opération inverse pour retrouver les facteurs de ce produit.

Détaillons-en les ingrédients. Alice choisit deux grands nombres premiers distincts p et q. Elle en calcule le produit

$$n = p.q.$$

Puisque p et q sont premiers, il est clair que la valeur en n de la fonction indicatrice d'Euleur[3] est

$$\varphi(n) = (p-1).(q-1).$$

Alice choisit à présent deux nombres e et d tels que

$$d.e \equiv 1 \mod \varphi(n).$$

Pour ce faire, elle choisit e tel que

$$1 < e < \varphi(n) \quad \text{et} \quad \text{pgcd}(e, \varphi(n)) = 1.$$

Ensuite, Alice obtient d grâce à l'algorithme d'Euclide étendu. Alice publie n et e et conserve secret les autres éléments $d, p, q, \varphi(n)$. On appelle e (resp. d) l'*exposant de chiffrement* (resp. *de déchiffrement*). On dira de plus que le couple $k = (e, n)$ est la clé du système. Si l'ensemble des textes clairs est $\mathcal{P} = \mathbb{Z}/n\mathbb{Z}$, alors

$$\forall x \in \mathbb{Z}/n\mathbb{Z},\ e_k(x) := x^e \mod n.$$

[3]La fonction d'Euleur compte le nombre de nombres premiers avec n et inférieurs à n, $\varphi(1) = 1$ et, si $n \geq 2$, $\varphi(n) = \#\{x \mid 1 \leq x \leq n,\ \text{pcgd}(x, n) = 1\}$.

Le chiffrement s'obtient aisément au moyen de l'*exponentiation modulaire*[4].

Proposition. Avec les notations précédentes, si $k = (e, n)$ et si $y = e_k(x)$, alors

$$y^d \mod n = x.$$

Cette proposition nous montre que pour déchiffrer un message, il suffit d'élever le texte chiffré à la puissance d et on posera donc

$$d_k(y) := y^d \mod n.$$

Connaissant d, le déchiffrement est donc semblable au chiffrement et utilise une fois encore l'*exponentiation modulaire*.

En résumé, on a les données suivantes :

- Alice publie : $k = (e, n)$,
- Alice conserve secret : d, p, q, $\varphi(n)$,
- fonction de chiffrement : $e_k(x) = x^e \mod n$,
- fonction de déchiffrement : $d_k(y) = y^d \mod n$.

Remarque. Le malhonnête Oscar a bien évidemment lui aussi accès à la clé publique (e, n) d'Alice. Pour retrouver l'exposant d de déchiffrement, il lui faut connaître $\varphi(n)$ pour ensuite calculer l'inverse de e modulo $\varphi(n)$. Un moyen lui permettant de trouver $\varphi(n)$ est de factoriser n. Ainsi, la sécurité du RSA réside dans le fait qu'il est facile de faire le produit n de deux grands nombres premiers p et q mais qu'il est par contre *supposé difficile* de factoriser n. On peut en outre montrer que la connaissance de d ou de $\varphi(n)$ revient à la factorisation de n.

Exemple. Considérons un exemple rudimentaire (et très peu réaliste). Soient les entiers :

$$p = 11,\ q = 23,\ n = p.q = 253,\ e = 3.$$

Puisque $220 = 3.73 + 1$, on en déduit que l'exposant de déchiffrement d est tel que

$$3^{-1} = -73 \mod 220 = 147 \mod 220.$$

Alice publie $k = (3, 253)$. Si Bob veut envoyer le texte clair $x = 165$ à Alice, il calcule

$$e_k(x) = 165^3 \mod 253 = 110.$$

Si Alice reçoit le message $y = 110$, il lui suffit de calculer

$$d_k(y) = 110^{147} = 110^{128}.110^{16}.110^2.110^1 \mod 253 = 165.$$

[4]cf. annexe.

5. Le RSA en pratique (ou presque)

Pour assurer la sécurité du RSA, il est de coutûme de prendre des nombres premiers p et q de l'ordre de 2^{512}. Un nombre de cette taille écrit en base 10 possède environ

$$\lfloor \log_{10} 2^{512} \rfloor = \left\lfloor 512 \underbrace{\log_{10} 2}_{\sim 0,3} \right\rfloor = 154$$

chiffres décimaux. Si nous supposons travailler sur un alphabet Σ de N symboles (on utilise un N majuscule pour le distinguer de $n = pq$), alors une valeur $N = 256$ pour le codage ASCII est bien trop petite par rapport à $n \sim 2^{1024}$. En effet, si on utilisait comme ensemble de textes clairs $\mathbb{Z}/N\mathbb{Z}$ au lieu de $\mathbb{Z}/n\mathbb{Z}$, alors une recherche exhaustive de toutes les images $x^e \mod n$ pour $x \in \mathbb{Z}/N\mathbb{Z}$ permettrait de réduire à néant la sécurité du RSA. Avec un N petit, une telle recherche est rapide à réaliser. On implémente dès lors en général le RSA en considérant comme unité de base, les blocs de k symboles consécutifs de l'alphabet Σ avec k défini par

$$k := \left\lfloor \log_N n \right\rfloor,$$

autrement dit, $N^k \leq n < N^{k+1}$. De plus, k éléments consécutifs de $\mathbb{Z}/N\mathbb{Z}$ correspondent à un nombre $x \in \mathbb{Z}/N^k\mathbb{Z}$ écrit en base N et compris entre 0 et $N^k - 1$. En fait, il est clair que k est la plus grande valeur possible permettant ce codage. Ainsi, si $x \in \mathbb{Z}/N^k\mathbb{Z}$, le chiffrement de x est donné par

$$y = x^e \mod n$$

qui est un entier $y < n$ et $n < N^{k+1}$. Donc la représentation de y en base N peut être de longueur $k + 1$. Ainsi, les blocs de k éléments consécutifs de $\mathbb{Z}/N\mathbb{Z}$ sont envoyés de manière injective[5] sur des blocs de longueur $k + 1$. Il suffit donc dans l'implémentation du RSA d'utiliser les conventions ad hoc[6].

Exemple (suite). Considérons encore une fois un exemple simpliste. Comme dans l'exemple précédent, on prend $n = 253$ et $e = 3$. L'alphabet utilisé est $\Sigma = \{a, b, c, d\}$ et le codage de Σ fait correspondre a (resp. b, c, d) à 0 (resp. 1, 2, 3). Ici, $N = \#\Sigma = 4$ et donc

$$k = \left\lfloor \log_4 253 \right\rfloor = 3$$

[5]Le caractère injectif est bien évidemment primordial pour le décodage !

[6]En particulier, le choix de k est univoquement déterminé par n et par la taille N l'alphabet.

car $4^3 = 64$ et $4^4 = 256$. Si Bob désire envoyer à Alice le message

$$bccadb,$$

il procède comme suit. Tout d'abord, le premier bloc de longueur 3, *bcc*, correspond à l'entier

$$1.4^2 + 2.4 + 2.1 = 26 < 64$$

et son chiffrement est donné par

$$26^3 \mod 253 = 119 = 1.4^3 + 3.4^2 + 1.4^2 + 3.4^0$$

qui correspond au mot *bdbd*. Le second bloc de texte clair, *adb*, correspond à

$$0.4^2 + 3.4 + 1 = 13$$

et son chiffrement

$$13^3 \mod 253 = 173 = 2.4^3 + 2.4^2 + 3.4^1 + 1.4^0$$

fournit le mot *ccdb*. Ainsi, Bob transmet le texte chiffré

$$bdbdccdb.$$

Quant à Alice, elle sait qu'elle doit redécouper le texte chiffré en blocs de longueur $k + 1 = 4$ avant déchiffrement.

6. Grands nombres premiers ?

Pour mettre en oeuvre le RSA, il est nécessaire de pouvoir générer de grands nombres premiers. Pratiquement, si on désire trouver un nombre premier de k bits lorsqu'il est écrit en base 2, on considère un vecteur appartenant à $\{0,1\}^k$ dont les première et dernière composantes valent 1. En effet, voulant obtenir un nombre ayant exactement k chiffres, celui-ci ne commence donc pas par 0 et pour avoir un nombre premier, le dernier chiffre doit être 1 sinon le nombre serait pair. Une fois ce nombre obtenu, on effectue des tests de primalité (test de Fermat, de Miller-Rabin, ...) pour déterminer si le nombre tiré est bel et bien premier. Le théorème de raréfaction des nombres premiers[7] nous donne une approximation du nombre de tirages à réaliser pour obtenir un nombre premier de k bits : le nombre de nombres premiers de k bits est

$$\pi(2^k) - \pi(2^{k-1}) \sim \frac{2^k}{\ln 2^k} - \frac{2^{k-1}}{\ln 2^{k-1}}$$

et le nombre total d'entiers impairs de k bits est 2^{k-2}. Ainsi, en quotientant les deux grandeurs, pour $k = 100$, la probabilité de tirer au

[7]Si $\pi(n)$ désigne le nombre de nombres premiers inférieurs ou égaux à n, alors

$$\lim_{n\to\infty} \frac{\pi(n)\ln n}{n} = 1.$$

Autrement dit, le quotient $\pi(n)/n$ se comporte asymptotiquement comme $1/\ln n$.

hasard un nombre premier est proche de $0,0285$ et pour $k = 512$, cette probabilité passe à $0,0056$.

On pourra noter que la plupart des tests employés pour tester la primalité d'un nombre sont des tests probabilistes. Cela signifie par exemple que, lorsque le test répond "*oui, ce nombre est premier*", c'est avec une probabilité supérieure à $1/2$ (par contre, lorsque la réponse est "*non, ce nombre n'est pas premier*", c'est de manière définitive). En pratique, on applique successivement plusieurs tests de façon à obtenir une probabilité très élevée que le nombre "considéré" soit premier. Ainsi, bien que les nombres premiers soient connus depuis l'antiquité et qu'ils ne posent aucun problème d'un point de vue conceptuel, il est frappant qu'il n'est pas aisé d'exhiber de grands nombres premiers !

Nous avons dit que la sécurité du RSA reposait sur l'impossibilité pratique actuelle de factoriser de grands entiers (on pourrait imaginer des avancées frappantes de la recherche dans le domaine de la factorisation). Voici un petit aperçu de l'état actuel de la factorisation des grands nombres premiers. (Rappelons que 2^{1024} comporte un peu plus de 300 chiffres décimaux).

- En 1998, des nombres de taille 10^{70} pouvaient être factorisés en près de 10 heures sur une station de travail.
- A cette même époque, il fallait un an pour factoriser un nombre de 100 chiffres décimaux sur une station.
- En 1999, un nombre de 155 chiffres a été factorisé en plus de 5 mois par 292 ordinateurs en réseau. Des calculs préalables à la factorisation ont pris près de 4 mois à des super-ordinateurs CRAY.
- En décembre 2003, le "*RSA Challenge number*" `RSA-576` de 576 bits (174 chiffres décimaux) a été factorisé[8]. Ce concours consiste à trouver la factorisation de grands nombres premiers et veut ainsi démontrer la sécurité du RSA pour des choix de clés appropriés.
- Le "*RSA Challenge number*" `RSA-640` de 640 bits suivant

  3107418240490043721350750035888567930037346022842727545720
  1619488232064405180815045563468296717232867824379162728380
  3341547107310850191954852900733772482278352574238645401469
  1736602477652346609

  vient d'être factorisé le 5 novembre 2005 (193 chiffres décimaux). A titre indicatif, on propose 100000 dollars pour le challenge number `RSA-1024` ...

On peut souvent lire des messages comme ceux-ci :

[8] `http ://www.rsasecurity.com/rsalabs/node.asp ?id=2092`

"*Using a minimal key length of* 1024 *bits, it is guaranteed that RSA can be considered safe for the near future as long as thre will be no fundemantal advance in the factoring of large numbers.*"

"*Clearly, the factoring of a challenge-number of specific length does not mean that the RSA cryptosystem is "broken." It does not even mean, necessarily, that keys of the same length as the factored challenge number must be discarded. It simply gives us an idea of the amount of work required to factor a modulus of a given size. This can be translated into an estimate of the cost of breaking a particular RSA key pair.*

Suppose, for example, that in the year 2010 a factorization of RSA-768 is announced that requires 6 months of effort on 100,000 workstations. In this hypothetical situation, would all 768-bit RSA keys need to be replaced ? The answer is no. If the data being protected needs security for significantly less than six months, and its value is considerably less than the cost of running 100,000 workstations for that period, then 768-bit keys may continue to be used."

7. Dans le secondaire ?

Bien que le présent exposé ne relève pas directement de la matière enseignée dans le secondaire, on pourrait imaginer que la présentation du RSA pourrait être facilement assimilée par des élèves rhétoriciens par exemple, dans un projet de fin d'année, dans un cours de renforcement ou encore en relation directe avec un cours d'informatique ou de programmation. Il n'y a pas si longtemps, début des années 90, la notion de groupe (abstrait) était encore au programme de la quatrième année du secondaire ! De même, pour les classes françaises de terminale, la cryptographie est explicitement citée dans le programme comme application directe de l'arithmétique. La cryptographie présente à la fois une base théorique, l'arithmétique, et une base algorithmique. Ceci exige un enseignement par l'exemple et l'expérimentation. En outre, la mise en oeuvre du RSA montre une application réelle des mathématiques à des élèves qui se demandent souvent à quoi servent les mathématiques.

Pour mettre en oeuvre cette courte introduction à la cryptographie, les notions suivantes devraient être présentées :

- les entiers modulo (par exemple, par le biais de l'arithmétique 12/24 heures, on introduit facilement la congruence modulo 12),
- la notion d'inverse dans un anneau (pour les exposants de chiffrement et de déchiffrement),
- éventuellement, la présentation de l'algorithme d'Euclide,
- la représentation des entiers en base $k \geq 2$ (cela peut par exemple déboucher sur la présentation du symbole sommatoire, sur l'étude des critères de divisibilité, ...).

Signalons enfin, que pour des étudiants se destinant à des études d'ingénieur ou de mathématicien, on retrouve les notions de groupes, d'anneaux et d'entiers modulo très tôt dans le cursus (aussi bien dans les cours d'algèbre actuels de premier bachelier en sciences mathématiques ou ingénieur civil).

8. Annexe : l'exponentiation modulaire

Nous allons voir comment l'écriture en base 2 permet de calculer rapidement une exponentiation modulo m. Ainsi, nous voulons rechercher

$$x^e \mod m \quad \text{avec} \quad e = \sum_{i=0}^{k} e_i\, 2^i.$$

Il est clair que

$$x^e = x^{\sum_{i=0}^{k} e_i\, 2^i} = \prod_{i=0}^{k} (x^{2^i})^{e_i} = \prod_{\substack{0 \le i \le k \\ e_i = 1}} x^{2^i}.$$

Si on remarque que $x^{2^{i+1}} = (x^{2^i})^2$, pour calculer $x^e \mod m$, il suffit donc de calculer les $x^{2^i} \mod m$ au moyen de k élévations successives au carré (modulo m) ainsi que k produits (modulo m).

Exemple. Pour calculer $6^{73} \mod 100$, on a tout d'abord

$$73 = 2^0 + 2^3 + 2^6, \quad \rho_2(73) = 1001001.$$

Ensuite, on calcule le tableau suivant

i	0	1	2	3	4	5	6
$(6^{2^{i-1}})^2$		6^2	36^2	$(-4)^2$	16^2	56^2	36^2
$6^{2^i} \mod 100$	6	36	-4	16	56	36	-4

Enfin, $6^{73} \mod 100 = 6^{2^0}.6^{2^3}.6^{2^6} = 6.16.(-4) = 16 \mod 100$. Pour effectuer ce calcul, nous avons donc eu besoin de 6 élévations au carré et de deux produits dans $\mathbb{Z}/100\,\mathbb{Z}$. Ceci est bien plus efficace qu'une méthode naïve qui aurait nécessité 73 multiplications.