Par K. Zayana et P. Dorbec

Les secrets de RSA

L'algorithme RSA est le système cryptographique à clé publique le plus répandu. Basé sur l'arithmétique modulaire, il est bien connu des élèves de terminale des séries scientifiques et de leurs professeurs, mais pas toujours correctement situé dans un contexte didactique.

Karim Zayana est professeur au lycée Hoche (Versailles).

Paul Dorbec est maître de conférence à l'Université de Bordeaux–CNRS.

Quels qu'en soient les motifs (militaires, médicaux, financiers, amoureux…), l'homme a toujours voulu protéger certaines informations. Un premier moyen est de les dissimuler en masquant leur présence ou en les faisant passer pour d'autres. Des exemples courants sont les encres sympathiques, qui se révèlent à la bougie, le recours à un messager dont il faut raser le crâne pour lire le message (inscrit avant que la chevelure ne pousse), ou, plus récemment, les images numériques dont on a imperceptiblement modifié les nuances de rouge–vert–bleu pour y enfouir un message binaire, que l'on retrouve en comparant le code de l'image avec l'image originale. Ces techniques sont du ressort de la stéganographie.

Un second moyen est le cryptage de l'information : il s'agit de coder le texte d'une façon connue uniquement de l'émetteur ou du récepteur, pour le rendre *illisible* et non plus *invisible*. Un exemple classique est le chiffre de César, obtenu par simple transposition des lettres (tel B→G, décalant le B sur le G, puis le C sur le H, *etc.* ; la clé de décodage est alors simplement G→B). Au lieu de transformer une lettre en une autre, on peut aussi permuter les lettres du message selon le principe des anagrammes. Des chiffrements tels DES (Data Encryption Standard) ou AES (Advanced Encryption Standard) exploitent les deux principes. Ces systèmes de chiffrement (ici, la transposition modulaire) sont connus de tous, le secret du message n'étant gardé que par une clé (la règle de chiffrement). C'est d'ailleurs là un élément majeur de la cryptographie moderne. Il est si difficile de garantir la fiabilité d'un système que l'on préfère le savoir analysé par de nombreux mathématiciens plutôt que fondé sur une idée tenue secrète, peu éprouvée, dont le principe aurait fini par filtrer.

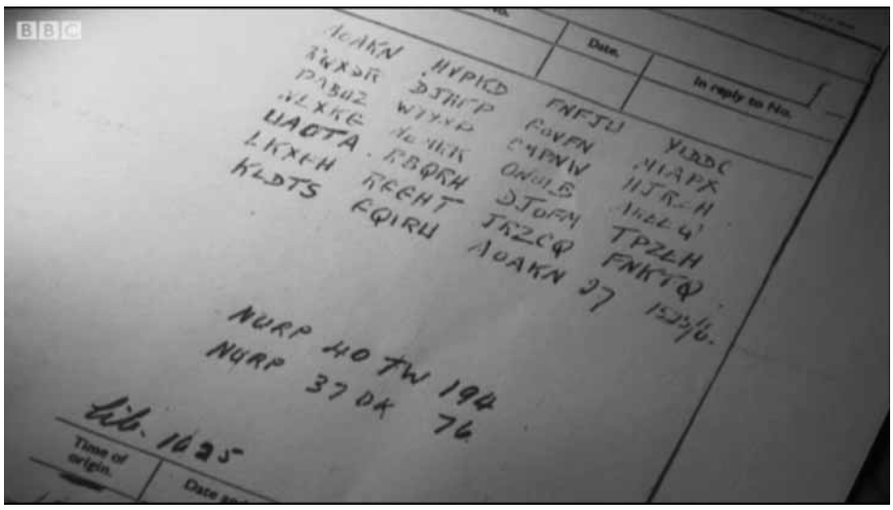
De gauche à droite : Rivest, Shamir et Adleman, les « pères » de RSA.

Des clés secrètes aux clés publiques

Quand les clés de codage–décodage sont similaires, on parle de *système symétrique*. Dans les années 1970, l'algorithme DES était utilisé ; on lui donnait alors une durée de vie d'une quinzaine d'années. Il a résisté remarquablement longtemps, et son obsolescence est née de la trop petite taille de sa clé, longue de 56 *bits*. Il a été remplacé dans les années 2000 par AES, à la clé longue de 128 à 256 *bits*. Ces systèmes sont très puissants et robustes, et les opérations de codage–décodage s'exécutent relativement vite sur des machines dédiées, rendant leur usage facile.

Deux schémas à clés secrètes sont particulièrement élémentaires : la transposition des lettres de l'alphabet, et l'entrelacement des lettres du texte à transmettre. En voici un troisième, adapté aux trames binaires $(b_n)_{n\geq 0}$, comme par exemple un flux de données sons ou un flux de données images. Les suites récurrentes linéaires sont un classique de l'arithmétique. D'habitude, les coef-

ficients de la récurrence sont des nombres réels. Ici, ils sont binaires, c'est-à-dire à valeurs dans {0, 1}, et les calculs sont menés modulo 2.
Voyons un exemple en posant $u_n = u_{n-1} + u_{n-2} + u_{n-4}$. Les coefficients sont 1, 1, 0, 1 (puis des 0). Avec $u_0 = 1$, $u_1 = 0$, $u_2 = 1$ et $u_3 = 1$, nous obtiendrions successivement $u_4 = 1 + 1 + 1 = 1$, $u_5 = 0$, $u_6 = 0$, *etc*. La suite ainsi engendrée est pseudo-aléatoire. Imprévisible de l'extérieur, quiconque serait averti des états initiaux et des coefficients de rétroaction saurait la régénérer à l'identique. La connaissance de cette suite u (relation de récurrence et conditions initiales) joue le rôle d'une clé. L'émetteur « embrouille » son message $(b_n)_{n\geq 0}$ en l'ajoutant modulo 2 à $(u_n)_{n\geq 0}$. À son tour, le récepteur bien informé ajoute à la volée au message reçu le même masque $(u_n)_{n\geq 0}$. Cela fonctionne car, modulo 2, $(b_n + u_n) + u_n = b_n + (u_n + u_n) = b_n$. La clé de chiffrement et la clé de déchiffrement sont identiques. En pratique, la mémoire du système (le nombre de coefficients de la boucle) est « grande », afin de démultiplier les états du registre. Bien entendu, émetteur et récepteur doivent rester synchrones, tout décalage faussant le procédé.



© Document BBC

Des spécialistes d'un service britannique de renseignement électronique tentent toujours de déchiffrer ce message, composé de vingt-sept blocs de cinq lettres, qui date de la Seconde Guerre mondiale et qui a été mis au jour fin 2012.

Le principal défaut des algorithmes à clé symétrique est le besoin d'une clé *secrète*, qu'il faut pouvoir échanger à distance, rapidement, et sûrement. En outre, les communicants devront souvent la modifier : chiffrer un gros volume avec la même clé accroît le risque qu'un tiers ne la brise. D'où l'intérêt des algorithmes à clé *publique*, dont RSA, qui vont notamment aider à crypter avec un haut degré de confidentialité la clé symétrique à partager.

L'identité clé du système RSA

La relation $(x^e \bmod n)^d \bmod n = (x^d \bmod n)^e \bmod n = x$ est au cœur du fonctionnement de RSA. Elle se démontre en exploitant uniquement l'arithmétique élémentaire.
On sait que $(x^e \bmod n)^d = x^{ed} \bmod n$. Or, $ed = 1 + k(p-1)(q-1)$ par définition de e et de d, k étant un certain entier naturel non nul. On en déduit :

$$(x^e \bmod n)^d = (x \bmod n)(x^{k(p-1)(q-1)} \bmod n) \bmod n.$$

Comme $x^{k(p-1)(q-1)} = (x^{p-1})^{k(q-1)}$, on distingue naturellement deux cas :
• x est premier avec p. Dès lors, $x^{p-1} = 1 \bmod p$ par le petit théorème de Fermat. Dans ces conditions, $x^{k(p-1)(q-1)} = 1 \bmod p$, puis $(x^e \bmod n)^d = x \bmod p$.
• x est multiple de p Dans ce cas, $(x^e \bmod n)^d$ l'est aussi, et $(x^e \bmod n)^d = x \bmod p = 0 \bmod p$.
De même, $(x^e \bmod n)^d = x \bmod q$ dans tous les cas.
On en conclut que $(x^e \bmod n)^d = x + k_1 p + k_2 q$, avec k_1 et k_2 deux entiers non nuls.
D'après le théorème de Gauss, k_1 est multiple de q. On peut donc écrire que $(x^e \bmod n)^d = x \bmod n$.
L'entier x n'excédant pas n, on a exactement $(x^e \bmod n)^d = x$.

Pierre de Fermat, par François de Poilly.

RSA : principe de fonctionnement

Le système RSA tient son nom de ses inventeurs, Ronald Rivest, Adi Shamir et Leonard Adleman, ingénieurs du Massachusetts Institute of Technology (Boston, États-Unis), qui l'ont proposé au milieu des années 1970. Sa robustesse vient en partie de la complexité à décomposer un « grand » nombre (de plusieurs centaines de chiffres, par exemple) en facteurs premiers.
Partons d'un entier $n = pq$, produit de deux nombres premiers distincts p et q. Choisissons une paire d'entiers e et d inverses l'un de l'autre modulo l'entier $N = (p - 1)(q - 1)$. Pour un entier quelconque x compris entre 0 et $n - 1$, on constate que $(x^e \bmod n)^d \bmod n$ est égal à x et aussi égal à $(x^d \bmod n)^e \bmod n$. La preuve (voir en page précédente) repose sur des résultats algébriques vus en terminale scientifique (identités de Bezout et de Gauss) ou à titre d'approfondissement (petit théorème de Fermat).

Définissons maintenant comme ci-dessus les nombres p, q, n, N, e et d. Le couple (e, n) forme notre *clé publique*, et on peut la diffuser largement. Le nombre d est notre *clé privée*, que nous gardons secrète.
Quelqu'un souhaite nous écrire (tout un texte, ou simplement une clé symétrique à utiliser pour un échange futur). Un tiers ne devra pas pouvoir lire ce message, même en cas d'interception. Comment procéder ? Il agrège son message (en mettant bout à bout les valeurs ASCII des lettres, par exemple) pour en former un entier $x < n$, ou plusieurs tels entiers si le message est trop long. Il calcule ensuite $y = x^e \bmod n$ et nous envoie le résultat. Nous élevons nous-mêmes à la puissance d (et modulo n) le nombre y reçu, et traduisons le message en clair. Au cours de ces étapes, l'utilisation d'un logiciel de calcul formel est appréciable…
Ce système de codage peut également servir à nous authentifier en tant qu'émetteur : si, de la même façon, nous codons un message x en $x^d \bmod n$ avec notre clé privée, tout le monde pourra décoder le message avec notre clé publique et s'assurer que nous en sommes bien l'émetteur (en le comparant avec le message x initial que nous aurons également transmis).
Un avantage considérable de cet algorithme est que RSA ne requiert pas de clé commune et secrète avant d'être fonctionnel. Deux inconvénients subsistent néanmoins. Déjà, le système ne peut coder les messages que par « petits » paquets, de valeur inférieure à n. Les calculs nécessaires au codage–décodage sont très importants, et ce d'autant plus que n est grand. Aussi n'emploie-t-on pas RSA pour des messages longs ou des applications temps réel.

RSA ne requiert pas de clé commune et secrète avant d'être fonctionnel.

La question de la sécurité

De nos jours, l'entier n choisi s'écrit souvent sur 1 024 à 2 048 *bits*. Il est supérieur à $2^{1\,023}$, qui représente un nombre possédant quelque trois cent quarante chiffres décimaux. Si l'on pouvait factoriser n, qui est public, on pourrait recalculer $N = (p - 1)(q - 1)$ et retrouver d à partir de e. On pourrait par exemple être tenté de diviser successivement n par tous les nombres premiers compris entre 2 et $\sqrt{n}$ (inclus). Or $\sqrt{n}$ dépasse 2^{511}. Un théorème de Tchebychev affirme qu'il existe environ $(2^{511}) / \ln(2^{511})$ nombres premiers inférieurs à $\sqrt{n}$, soit presque 2^{508}, qui dépasse 10^{152}. À raison d'un milliard d'opérations par seconde, cela représente 10^{135} milliards d'années de calcul. Et avec le meilleur algorithme connu, on atteindrait en fait les 10^{14} milliards d'années : cela reste démesuré.
Une autre méthode de force brute consisterait à partir d'un mot codé $y = x^e$, d'en calculer les puissances successives y, y^2, y^3, y^4 *etc.* (modulo n), jusqu'à reconnaître le mot initial x obtenu pour l'exposant d. Cependant, en pratique, d est également de l'ordre de $2^{1\,023}$ et la quantité de calculs reste astronomique. Ces chiffres illustrent com-

bien RSA est gourmand en calculs. Heureusement, des méthodes d'exponentiation rapide existent et permettent d'accélérer certains calculs. Par exemple, pour obtenir une puissance bien déterminée, par exemple y^{21}, on peut d'abord la fractionner en $y(y^{10})^2$, puis en $y((y^5)^2)^2$, puis en $y(((y(y^2)^2)^2)^2)^2$ avant de remonter la pile de calculs. En bout de chaîne, y^{21} est obtenu sans passer par y^{19} ou y^{18}, mais grâce à six multiplications au lieu de vingt. Cette astuce est déjà implantée dans certains logiciels de calcul formel comme Maple (pour lequel elle est appelée par une esperluette &, en notant a &^ d au lieu de a ^ d). Voyons pour terminer deux applications classiques de RSA : le fonctionnement de la carte bleue et celui du protocole SSL.

Dans une transaction par carte bancaire, RSA sert à authentifier l'émetteur de la carte. Le Groupement d'intérêt économique carte bancaire (GIE CB) fixe les entiers n, e et d de tout le réseau. Ce triplet unique est rarement changé. Les entiers n et e sont rendus publics (aucune publicité ne les entoure cependant). Quant à d, il est gardé secret. À la fabrication de la carte, un numéro personnel (embossé au recto de la carte) vous est attribué. Il joue le rôle de l'entier x. Sa signature $y = x^d \bmod n$ est calculée en privé. Les deux nombres sont alors gravés sur la puce de la carte. Puis vous recevez et utilisez votre carte. En caisse, le terminal de payement lit la paire (x, y), calcule y^e et le compare à x. Le cas échéant, le protocole se poursuit (saisie du code PIN, envoi d'un « défi », contrôle en ligne, *etc.*). Au mieux pouvez-vous cloner une carte déjà existante en copiant la paire (x, y). Toutefois, vous ne passerez pas forcément les obstacles suivants.

De même, vous utilisez RSA à chaque visite d'un site Internet sécurisé dont l'URL commence par https:// (par exemple en consultant vos *e-mails* ou vos comptes bancaires). Deux conditions sont essentielles à cette sécurisation : le cryptage des données échangées, et le contrôle que la communication se noue avec le « vrai » destinataire. L'adresse IP (seul élément qui identifie simplement votre interlocuteur) varie en effet souvent pour un même site, et peut être falsifiée. La solution ? Quand commence la connexion, le site « sensible » communique (en clair) un certificat à votre navigateur. Ce certificat contient la clé publique du site ainsi qu'une signature de cette clé par une autorité supérieure (elle-même garantie grâce à RSA). Votre navigateur vérifie que la signature correspond à une autorité en qui il a confiance et qu'elle signe bien la clé publique du site, sans quoi il vous alerte. Ensuite, votre navigateur génère aléatoirement une clé symétrique de session qu'il transmet au site « sensible » avec la clé publique qu'il vous a donnée. Le site décrypte la clé symétrique avec sa clé privée pour initier la communication secrète avec vous.
Faites le test par exemple sur Mozilla depuis Gmail : Outils → Informations sur la page → Sécurité → Afficher le certificat → Détails → Certificat ou info clé publique du sujet.
Vous lirez : RSA, module n (de 1 024 *bits* et affiché en hexadécimal), exposant e valant 65 537.

On peut contrefaire des signatures RSA à partir de signatures connues, le code RSA d'un produit valant le produit des codes. Aussi couple-t-on RSA avec des fonctions de hachage, mais il s'agit là d'un tout autre chapitre de la cryptographie…

K.Z. & P.D.

Références

On écoutera avec intérêt les conférences de Véronique Cortier, directrice de recherche au CNRS. Les vidéos sont en ligne sur les sites du Collège de France et de l'Inria.