Stratégies d'attaques

Louis GOUBIN

Depuis l'invention du système RSA, il y a 25 ans, la cryptanalyse de ce système n'a pas révélé de failles majeures, mais elle a montré que des précautions d'utilisation étaient indispensables.

Le RSA est l'algorithme le plus connu et le plus utilisé au monde d'un système de chiffrement à clé publique. Inventé par Ronald Rivest, Adi Shamir et Leonard Adleman, il a été présenté publiquement pour la première fois dans le numéro d'août 1977 de la revue *Scientific American*. On le trouve dans un nombre toujours croissant de produits commerciaux liés à la sécurisation des échanges de données sur Internet, à la protection de la confidentialité et de l'authenticité du courrier électronique ou au paiement électronique au moyen de cartes à puce.

Depuis 25 ans, la communauté cryptographique mondiale étudie la solidité du RSA face à toute une panoplie d'attaques. Ces attaques se divisent en trois catégories : les attaques qui cherchent à trouver une faille dans les fondements mathématiques mêmes de l'algorithme, les attaques sur les protocoles construits autour du RSA, enfin, les attaques sur les implémentations du protocole.

NOMBRE DE MIPS.AN DISPONIBLES	TAILLE MAXIMALE DES CLÉS RSA «FACTORISABLES»
0,048	251
49,6	385
500	438
10000	512
50796	555
512000	620
10^8	784
10^{11}	1035
10^{16}	1551
10^{20}	2057

1. EN SE FONDANT SUR LE FAIT qu'une clé de 512 bits a été factorisée en 1999 avec une capacité de calcul de 10^4 MIPS.an, on peut estimer la capacité de calcul nécessaire pour factoriser une clé d'un nombre de bits donné à l'aide du crible algébrique général, un algorithme de factorisation efficace pour ce genre de nombres.

Toutes ces attaques ont montré que les fondements de RSA sont solides, mais que certaines précautions sont nécessaires, notamment sur la taille des clés à utiliser. Par ailleurs, pour ce qui concerne le protocole, les résultats des attaques ont mis en évidence le danger de chiffrer un même message par plusieurs clés publiques RSA distinctes : la structure du message peut alors apparaître dans les différents chiffrés. Même si toutes les précautions ont été prises (taille des clés adéquate, protocoles sûrs), le dispositif de chiffrement est dans le monde réel : il calcule, il consomme de l'électricité et il commet parfois des erreurs de calculs. Toutes ces manifestations informent un adversaire sur le message qu'il peut éventuellement déchiffrer. Examinons ces attaques et les manières de s'en prémunir.

Le problème mathématique

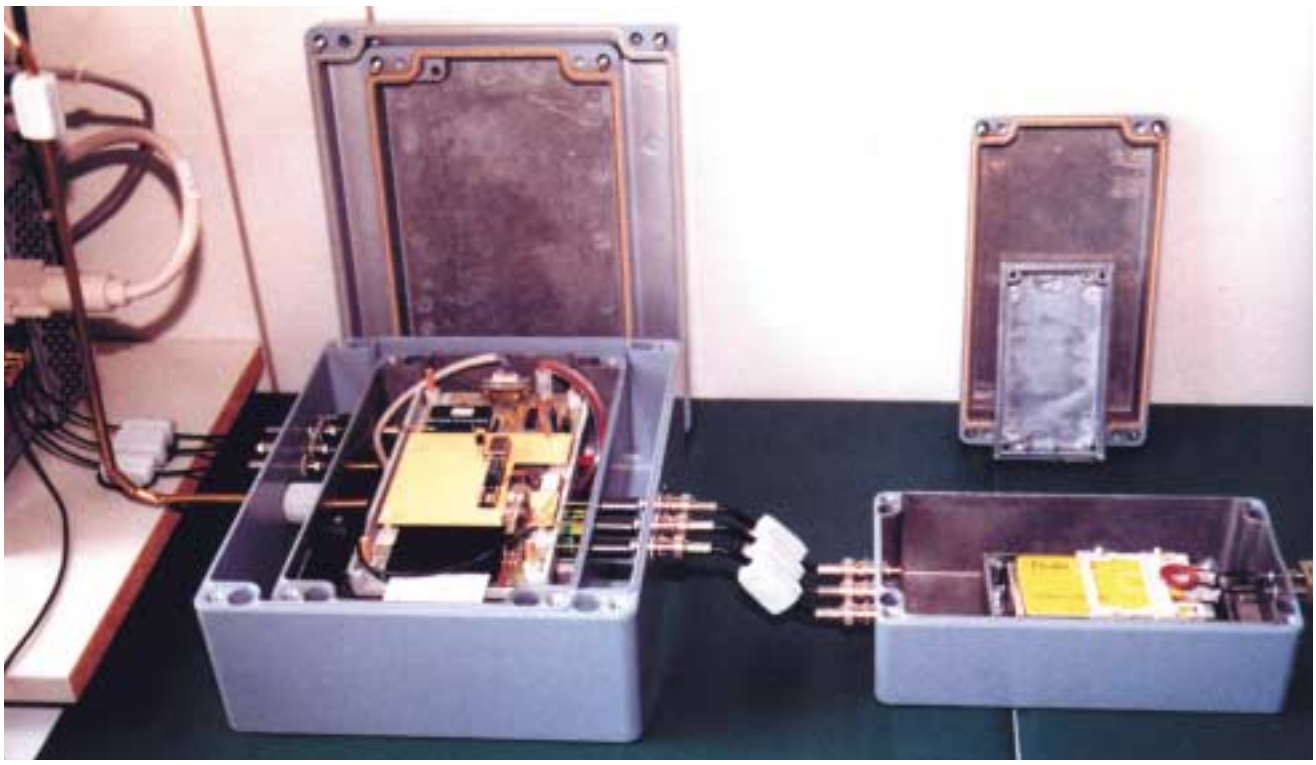
Mathématiquement, on peut décrire l'algorithme RSA de la manière suivante. On commence par choisir l'exposant public e (par exemple 3, 17, 257 ou 65537). On utilise ensuite un générateur de nombres aléatoires pour obtenir deux nombres premiers p et q tels que e soit premier avec p–1 et avec q–1. Si l'on pose $n=pq$, la clé publique est constituée de e et de n, alors que la clé secrète (ou privée) est constituée de p et q. Aujourd'hui on prend typiquement p et q de taille 512 bits, et donc n de taille 1024 bits. La fonction de chiffrement est alors définie par une fonction f qui à x associe y tel que $y = x^e$ modulo n ; la fonction de déchiffrement est f^{-1}, l'inverse de f, qui à y associe x tel que $x = y^d$ modulo n, où d est l'inverse de e modulo $(p–1)\times(q–1)$. Ici d est également une valeur qui doit rester secrète.

La fonction f, aussi nommée fonction RSA, est donc conçue pour être facilement inversible lorsqu'on connaît d. Casser la fonction RSA consiste à trouver un moyen de calculer l'inverse de la fonction f sans recours à l'exposant secret d. Plus exactement, on s'intéresse au problème suivant : étant donnés la clé publique (e, n) et un entier y compris entre 2 et n–1, est-il possible de calculer de façon efficace la racine $e^{\text{ième}}$ de y modulo n?

Plaçons-nous dans la situation où d est *a priori* inconnu et donc la factorisation de n également. La première façon d'attaquer la fonction RSA consiste à essayer de retrouver cette factorisation. De nombreux algorithmes spécialisés ont été inventés pour trouver p et q à partir de n. Dans notre cas, la meilleure méthode connue est le crible algébrique général (NFS). Sachant que cet algorithme a été utilisé avec succès en 1999 pour factoriser un nombre de 512 bits avec 10^4 MIPS.an (c'est-à-dire 10^{10} instructions par seconde d'un ordinateur qui fonctionnerait pendant un an), on peut estimer la capacité de calcul nécessaire pour factoriser un entier n de taille quelconque. Cela permet d'évaluer la puissance nécessaire pour casser la fonction RSA par ce moyen, en fonction de la taille de n *(voir la figure 1)*. De surcroît, si l'on suppose que la puissance des ordinateurs double tous les 18 mois (la loi empirique de Moore), on peut alors prédire que les clés RSA de 1024 bits seront cassées vers l'an 2034 et les clés de 2048 bits vers 2079. Cette prédiction serait bien entendu mise en défaut si des nouvelles techniques de factorisation plus efficaces sont mises au point.

Mais est-il vraiment nécessaire de factoriser n (ou, ce qui est équivalent, de connaître d) pour être en mesure de calculer la racine $e^{\text{ième}}$ d'un entier y modulo n? Le problème n'est pas résolu. Si certains indices laissent penser que casser la fonction RSA serait moins ardu que le problème de la factorisation, on ne

© POUR LA SCIENCE



E.I.S.S.

2. AVEC UN DISPOSITIF MODESTE D'ANALYSE des cartes à puce branché sur un ordinateur, on mesure facilement des temps de calcul ou la consommation électrique, des informations très utiles pour retrouver la clé secrète utilisée dans le cryptosystème RSA de ces cartes.

connaît aujourd'hui aucune autre stratégie générale que de factoriser n, afin d'en déduire l'exposant secret d.

À condition de faire des hypothèses supplémentaires sur l'exposant secret d, d'autres attaques que la factorisation sont envisageables. Ainsi, lorsque l'on suppose que la taille de l'exposant d est relativement petite, on accélère le calcul de la fonction f^{-1}. Par exemple, Michael Wiener de la société canadienne *Entrust* a montré en 1990 qu'il était facile de retrouver la valeur d à partir de e et de n, lorsque d est inférieur à $n^{1/4}$. Cette attaque a été améliorée en 1998 par Dan Boneh et Glen Durfee de l'Université de Stanford, qui ont étendu l'attaque de Wiener à tous les exposants d inférieurs à $n^{0,292}$ (au lieu de $n^{0,25}$). La conjecture généralement admise est qu'une attaque devrait être possible pour tous les d inférieurs à $n^{0,5}$, mais cela n'a pas encore été démontré.

Une deuxième attaque apparaît lorsqu'on suppose que certains bits de la clé d sont connus. D.Boneh et ses collègues ont prouvé en 1998 que si d a une taille de k bits, la connaissance des $k/4$ bits de poids le plus faible de d (c'est-à-dire les bits situés à la fin de la clé, donc les moins significatifs) est suffisante pour reconstituer l'intégralité de d. Dans le même ordre d'idée, Phong Nguyen de l'École normale supérieure de Paris a montré qu'une attaque similaire est possible si l'on connaît les $k/4$ bits de d situés entre les positions $k/4$ et $k/2$. En revanche, on ne sait pas ce que l'on peut dire dans les cas intermédiaires où $k/4$ bits consécutifs sont connus, mais situés ailleurs entre les positions 1 et $k/2$ *(voir la figure 3)*.

Les attaques de protocole

Même si la fonction RSA est solide, la façon dont on l'utilise pour obtenir un cryptosystème capable d'effectuer du chiffrement ou de la signature n'est pas neutre. De manière générale, il doit non seulement être impossible en pratique de retrouver un message clair à partir de son chiffré (ou de forger une fausse signature), hormis pour l'utilisateur légitime du système bien entendu, mais il doit être également impossible de déceler la moindre information sur le message clair (ou sur la signature). C'est ce que l'on nomme la *sécurité sémantique*. Or, la fonction RSA ne vérifie pas cette propriété, car elle possède une propriété de multiplicativité : $f(x{\times}y) = f(x) \times f(y)$. Cette faiblesse ouvre la voie à certaines attaques, que ce soit en mode de chiffrement ou en mode signature.

Illustrons comment cette propriété peut fournir une attaque pour le chiffrement. Supposons que Bernard veuille envoyer le même message M, sous forme chiffrée, à trois interlocuteurs différents, dont les clés publiques RSA sont respectivement (n_1, e=3), (n_2, e=3) et (n_3, e=3). S'il utilise la fonction RSA de façon élémentaire, Bernard envoie donc les valeurs $C_1 = M^3$ modulo n_1, $C_2 = M^3$ modulo n_2 et $C_3 = M^3$ modulo n_3. Si Caroline, une espionne patentée, intercepte les trois chiffrés C_1, C_2 et C_3, elle peut retrouver aisément le message clair M. D'abord, à l'aide du théorème des restes chinois *(voir l'encadré page suivante)*, Caroline calcule la valeur C' telle que $0 \leq C' < n_1n_2n_3$, et $C' \equiv M^3$ modulo $n_1n_2n_3$. Or, par hypothèse, les messages ont une longueur inférieure à la première partie de la clé publique, c'est-à-dire que l'on a $M < n_1$, $M < n_2$ et $M < n_3$. Donc $M^3 < n_1n_2n_3$ et par suite $C' = M^3$, ou encore $M = (C')^{1/3}$. Caroline a donc retrouvé M en faisant une simple racine cubique! Cela montre qu'il est fort risqué de chiffrer le même message M par plusieurs clés publiques RSA distinctes.

Dans une autre direction, D. Coppersmith et ses collègues ont montré en 1996, qu'il est également risqué de chiffrer plusieurs messages «liés» au moyen de la même clé publique RSA. Plus précisément, si Bernard envoie les chiffrés C_1 et C_2 de deux messages M_1 et M_2 liés par une relation de dépendance polynômiale $M_2 = P(M_1)$ modulo n, alors il est facile pour un attaquant de retrouver M_1 et M_2 à partir de C_1 et C_2.

S'il peut sembler artificiel d'avoir à chiffrer deux messages ainsi liés, un scénario assez proche peut se produire dans des applications réelles. Supposons que pour chiffrer un message M de longueur $(k–m)$ bits, on lui ajoute, à la fin, m bits aléatoires avant de lui appliquer la fonction RSA, dont le modulo fait k bits. Il arrive que Bernard doive envoyer deux fois le chiffré du même message M (par exemple, s'il y a eu un problème de transmission sur la ligne). Mathématiquement, cela signifie que Bernard envoie successivement $C_1 = f(M_1)$ et $C_2 = f(M_2)$, avec $M_1 = 2m.M + r_1$ et $M_2 = 2m.M + r_2$, où r_1 et r_2 sont deux valeurs aléatoires de m bits. La présence de ces m bits aléatoires offre-t-elle une protection suffisante? Non, car Don Coppersmith a montré en 1997 que l'on pouvait retrouver M à partir de C_1 et de C_2 à condition que m ne soit pas trop grand : plus précisément l'attaque fonctionne si m est inférieur à k/e^2 (ainsi pour $e = 3$, l'attaque est possible si la partie aléatoire ajoutée au message a une longueur inférieure à 1/9 de celle du modulo).

Toutes ces attaques montrent qu'il faut faire très attention à la manière dont on applique la fonction RSA pour chiffrer un message. Dans la pratique, on commence par «formater»le message M au moyen d'une transformation φ qui fait intervenir un nombre aléatoire. Ainsi le message chiffré est obtenu par $C = f(\varphi(M,r))$, où r est une valeur aléatoire. Pour déchiffrer, on calcule $\varphi(M,r) = f^{-1}(C)$, et φ est conçue pour qu'il soit facile de retrouver M à partir de $\varphi(M,r)$. La théorie des protocoles de chiffrement RSA est aujourd'hui bien maîtrisée, au point qu'on connaît maintenant des transformations φ qui sont prouvées «saines». Cela signifie que, pour ces transformations φ, la sécurité sémantique du schéma global de chiffrement $C = f(\varphi(M,r))$ est équivalente à la solidité de la fonction RSA de base.

En d'autres termes, si la fonction f est mathématiquement solide et qu'on utilise une «bonne» transformation φ, alors il n'existe pas d'attaque mathématique plus simple que celle consistant à trouver un moyen d'inverser la fonction f (voir *Les preuves de sécurité*, par David Pointcheval, dans ce dossier).

Attaques sur le RSA en mode signature

La sécurité sémantique du RSA lorsqu'on l'utilise pour signer des messages ne va pas non plus de soi. La signature d'un message M consiste à l'accompagner d'un nombre S, par exemple $S = M^d$ modulo n, où d est l'exposant secret, un nombre que seul le détenteur de la clé secrète peut calculer.

Supposons que Caroline veuille faire signer à Bernard un message M qu'elle a choisi elle-même (par exemple M peut signifier «Je soussigné, Bernard, donne 1000 euros à Caroline.»). Évidemment, si Caroline envoie ce message tel quel, elle sera démasquée immédiatement, mais elle peut agir de manière plus subtile : elle envoie le message $M' = M.R^e$ modulo n à Bernard, où R est une valeur aléatoire qui sert à dissimuler le vrai message M. Si Bernard n'est pas attentif et renvoie la signature $S' = M'^d$ modulo n, Caroline peut en déduire $S = S'.R^{-1}$ modulo n, qui est la signature du vrai message M. Cette attaque par «camouflage» montre qu'il est dangereux de signer n'importe quoi (même si le message semble inoffensif…).

Pour empêcher un éventuel attaquant d'utiliser la propriété de multiplicativité de la fonction RSA, une méthode naturelle consiste à «compléter» le message M par une valeur fixe P (pour *padding*, soit «remplissage»), avant de lui appliquer la fonction f^{-1}. En pratique, on place P avant le message M (cela s'écrit $P||M$). Dans ce cas, la signature est $S = (P||M)^d$ modulo n.

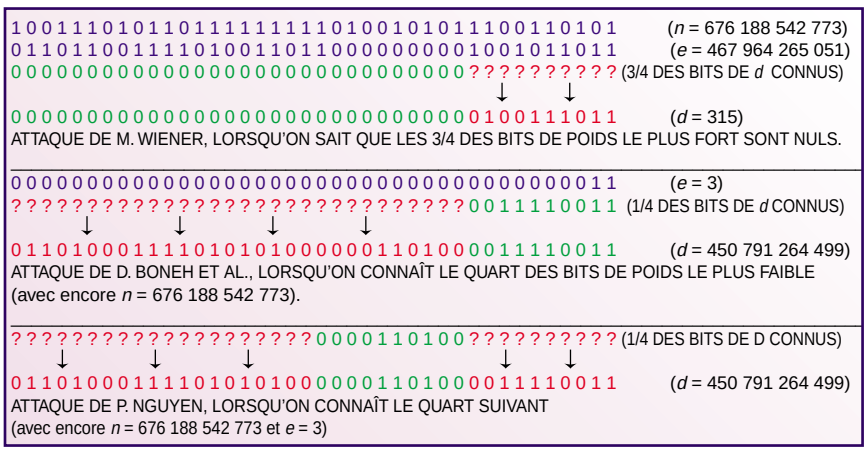
Plus généralement, on calcule la signature par $S = (w.M + a)^d$ modulo n, où w est appelé la redondance multiplicative et a la redondance additive (le *padding* avec P correspond à w=1 et a=$P.2^m$, où m est la taille du message). Toutefois, même si cette structure est censée protéger la signature, de nombreuses attaques sont encore possibles. Elles montrent notamment qu'introduire de la redondance dans le message initial n'est probablement pas suffisant pour cacher la structure multiplicative du RSA.

Pour résoudre le problème, la technique prônée depuis quelques années consiste à appliquer successivement au message une fonction de hachage (c'est-à-dire une fonction qui transforme le message en un résumé de celui-ci ; la fonction de hachage est construite de sorte que deux messages différents ont toujours deux résumés différents), puis une fonction de redondance, et enfin la fonction f^{-1} du RSA. Certains standards internationaux, comme ISO 9796-1, ISO 9796-2, ou PKCS#1 proposent de telles méthodes. Néanmoins, ces protocoles n'étaient jusqu'à présent pas prouvés sémantiquement sûrs. De fait, en 1999, Jean-Sébastien Coron et ses collègues ont découvert une faille dans la norme ISO 9796-1 : on peut forger une signature S sans connaître M.

Heureusement, comme pour le chiffrement, la théorie des protocoles de signature RSA est maintenant bien comprise : on connaît des transformations φ pour lesquelles la sécurité sémantique du schéma global de signature $S = f^{-1}(\varphi(M))$ est équivalente à la solidité de la fonction RSA de base. Deux protocoles de signature ainsi «prouvés sûrs» sont connus : FDH (*Full Domain Hash*) et PSS (*Probabilistic Signature Scheme*). Ce dernier doit prochainement devenir la nouvelle norme de signature dans PKCS#1 v2.1 utilisée sur Internet

Les attaques physiques

Comme nous l'avons vu, si l'on suppose que la fonction RSA est mathématiquement solide, on peut construire des protocoles sûrs, aussi bien pour le chiffrement que pour la signature. Ces protocoles admettent des preuves de sécurité. Néanmoins, dans les applications réelles,



3. SI L'ON SUPPOSE QUE CERTAINS BITS DE LA CLÉ SECRÈTE d sont connus, par exemple le quart des bits correspondant à ceux de poids le plus faible, on peut facilement retrouver d.

© POUR LA SCIENCE

THÉORÈME DES RESTES CHINOIS

Le théorème des restes chinois est très employé en cryptanalyse. Il donne la solution d'un système d'équation modulo. Prenons $m_1,\ldots, m_n$ des entiers supérieurs à 2 deux-à-deux premiers entre eux et des entiers $a_1,\ldots, a_n$. Le théorème stipule que le système d'équations :

$x = a_1$ modulo m_1
…
$x = a_n$ modulo m_n

admet une unique solution modulo $M = m_1 \times \ldots \times m_n$ donnée par la formule :

$$x = \sum_{i=1}^{n} a_i \, Mi \, y_i \text{ modulo } M$$

où $M_i = M/m_i$ et $y_i = M_i^{-1}$ modulo m_i pour i compris entre 1 et n.

Par exemple, supposons que nous ayons trois équations et que $m_1 = 7$, $m_2 = 11$ et $m_3 = 13$. Leur produit, M, est donc égal à 1001. On calcule $M_1 = 143$, $M_2 = 91$, $M_3 = 77$. D'après l'algorithme d'Euclide étendu (qui permet de calculer les inverses des nombres modulo), on trouve $y_1 = 5$, $y_2 = 4$, $y_3 = 12$.

Le système d'équations :

x = 5 modulo 7
x = 3 modulo 11
x = 10 modulo 13

admet une solution unique qui est $x = 5\times143\times5 + 3\times91\times4 + 10\times77\times12$ modulo 1001, soit x = 13907 mod 1001 = 894.

malgré ces preuves, certaines attaques restent possibles. Celles-ci ne s'attaquent pas au problème mathématique, mais elles utilisent le fait que le dispositif électronique qui fait les calculs de chiffrement ou de signature, n'est pas une abstraction mathématique : dans le monde physique, il met un certain temps à calculer, il consomme du courant électrique, il peut faire des erreurs de calcul, etc. Avec ces paramètres supplémentaires, l'attaquant parvient parfois à ses fins.

Examinons trois types d'attaques fondées sur ces données : les attaques sur le temps de calcul, les attaques sur la consommation électrique et les attaques par injection de fautes. Notons que toutes ces attaques «physiques» concernent essentiellement les cartes à puce pour lesquelles, avec un dispositif adéquat, il est facile d'obtenir des données *(voir la figure 2)*.

Les attaques sur le temps de calcul utilisent le fait que, dans le calcul de la fonction RSA, l'algorithme de calcul utilisé est toujours le même. En particulier, il fait intervenir une boucle pour calculer la fonction y^d modulo n, où d est l'exposant secret. L'exposant secret s'écrit en binaire $d = d_{k-1}d_{k-2}\ldots d_1 d_0$, où chaque d_i est égal à 0 ou à 1. Or dans la boucle de calcul, on trouve une instruction du type «si $d_i = 1$, alors faire tel calcul, sinon ne pas faire ce calcul». En mesurant les temps de calcul pour de nombreuses valeurs initiales de y, un attaquant peut déduire si le calcul qui est fait seulement lorsque $d_i = 1$ a été réellement effectué et, de là, retrouver la clé secrète d bit par bit. On contourne ce genre d'attaques en masquant ces différences de temps de calcul. Autrement dit, on programme la fonction RSA de sorte que le temps de calcul soit indépendant de la valeur des bits de la clé d.

Le même type d'attaque est réalisable sur la consommation électrique. Pour chacun des calculs, on mesure la courbe de consommation électrique du composant qui fait le calcul. Là encore, en analysant la statistique de la consommation électrique à chaque étape du calcul, on déduit de proche en proche la valeur de la clé secrète d *(voir la figure 3)*. Des méthodes existent également pour fausser la consommation électrique et rendre cette information inutile.

Pour terminer ce florilège des méthodes du parfait pirate, nous mentionnerons les attaques par injection de fautes. En 1996, D. Boneh et ses collègues ont proposé un nouveau modèle d'attaque physique sur les cartes à puce, qu'ils ont nommé «cryptanalyse en présence d'erreurs de calculs dans le processeur». Dans le cas de la signature RSA, ils ont montré que, d'une part, si l'implémentation du calcul utilise le théorème des restes chinois *(voir l'encadré)*, une signature erronée et la signature correcte correspondante suffisent à factoriser le modulo, et, d'autre part, si l'implémentation n'utilise pas la méthode des restes chinois, l'attaque peut fonctionner avec un nombre de signatures erronés, de l'ordre du nombre de bits du modulo. D'autres cryptologues ont ensuite montré que, dans le cas de l'utilisation des restes chinois, un message et une signature erronée de ce message suffisaient pour retrouver la factorisation du modulo.

Illustrons le cas le plus simple, avec les restes chinois. Pour le calcul de $x = y^d$ modulo n, l'ordinateur calcule séparément $x_p = x$ modulo p et $x_q = x$ modulo q, puis reconstitue x à partir de x_p et de x_q en utilisant le théorème des restes chinois. Supposons que dans la carte à puce, sont stockées les valeurs $d_p = d$ modulo $(p–1)$ et $d_q = d$ modulo $(q–1)$. Les valeurs x_p et x_q sont calculées au moyen des formules suivantes : $x_p = y^{dp}$ modulo p et $x_q = y^{dq}$ modulo q. L'attaquant lance alors le calcul deux fois. La première fois, il n'introduit pas de perturbation. Il obtient alors les bonnes valeurs pour x_p et pour x_q et le théorème des restes chinois donne la valeur correcte x. La seconde fois, il perturbe le calcul (par exemple au moyen de rayonnements électromagnétiques ou en faisant varier l'alimentation électrique). Il obtient par exemple la bonne valeur pour x_p , mais une valeur erronée (x'_q) pour x_q. Le théorème des restes chinois donne alors une valeur x' erronée, qui vérifie $x' \equiv x$ modulo p, mais $x' \neq x$ modulo q. Cela montre que $x'–x$ est divisible par p mais pas par q. On peut alors calculer le PGCD $(x'–x, n)$, qui est en général égal à p. La factorisation de n est donc trouvée, et le système est cassé.

Un moyen simple d'éviter ce type d'attaque consiste à demander au programme une vérification systématique du calcul avant de sortir le résultat. Pour le RSA, c'est particulièrement commode, puisqu'il suffit par exemple de s'assurer que $x^3 = y$ modulo n.

Les nombreuses attaques que nous avons présentées montrent que l'utilisation du RSA réclame un soin particulier, que ce soit dans le choix de la taille des paramètres, dans celui du protocole de chiffrement ou de signature, ou même dans la manière dont le calcul est programmé. Néanmoins, 25 ans après son invention, on constate que le RSA est arrivé à maturité. Si toutes les précautions que nous avons mentionnées sont prises, RSA résistera. Mais n'oublions pas que ce sont justement ces précautions qui font le plus souvent défaut dans le cas où les systèmes ont été attaqués. Sur Internet par exemple, on recense quotidiennement des dizaines d'alertes de sécurité. Dans un environnement aussi complexe, on ne s'improvise pas cryptologue…

Louis GOUBIN est responsable des études cryptographiques pour la division *Cartes à puce* de *Schlumberger-Sema*