se rencontrer pour convenir d'une clef secrète, qui devra rester connue d'eux seuls, ni n'ont besoin de faire circuler – sur un réseau informatique ou autre – une clef secrète, transmission qui bien sûr engendre un risque. Seule la clef publique, insuffisante pour le décryptage, est connue préalablement aux échanges cryptés entre les deux interlocuteurs.

Le RSA est un système à clef publique qui permet de signer – on dit aussi authentifier – des messages : pour cela, *Émetteur* transforme, avec sa clef privée, le texte à signer en un message codé. *Destinataire* reçoit le message et la signature qu'il peut décoder avec la clef publique d'*Émetteur* (le fait de pouvoir décoder la signature assure que c'est bien *Émetteur* qui a signé le message). On peut combiner les méthodes de codage et d'identification, comme indiqué sur la figure 2.

L'algorithme RSA est moins rapide que les algorithmes classiques à une seule clef, ce qui est un handicap lorsque l'on doit coder des messages volumineux. Aussi est-il fréquemment utilisé en combinaison avec un algorithme classique. Dans une première phase, le RSA (avec ses deux clefs) permet à *Émetteur* de transmettre secrètement à *Destinataire* une troisième clef secrète qu'ils seront seuls à connaître : *Émetteur* la fait parvenir à *Destinataire* en la cryptant avec la clef publique de *Destinataire*, qui, à l'arrivée, la décrypte avec sa clef secrète.

Dans un second temps, la troisième clef est utilisée avec un algorithme symétrique convenu pour envoyer rapidement à *Destinataire* le long message. Les algorithmes à clefs symétriques étant de cent à dix mille fois plus rapides que RSA (selon qu'il s'agit de versions matérielles ou logicielles), ce procédé à deux étages est aujourd'hui d'utilisation très commune.

VOUS L'UTILISEZ... SANS LE SAVOIR

Insistons sur le fait que ce jeu numérique élémentaire est aujourd'hui un système universel servant dans une multitude d'applications. Au cours des années, il a supplanté tous ses concurrents, qui sont abandonnés (au profit du RSA) soit parce qu'on y a trouvé des points faibles, soit parce qu'ils n'ont pas été assez étudiés pour qu'on soit convaincu de leur solidité.

La technologie RSA est protégée par un brevet dans certains pays (aux États-Unis, ce brevet expire en septembre 2000 ; en France, il n'y a pas de brevet). Elle a été commercialisée par plus de 350 entreprises, et l'on estime que plus de 300 millions de programmes installés utilisent le RSA : votre ordinateur contient sans doute de tels programmes, qui sont activés – sans même que vous le sachiez – lorsque, par exemple, vous souhaitez mener des transactions sécurisées sur l'Internet.

Le RSA est inclus dans de nombreux standards informatiques, dont le *Society for Worldwide Interbank Financial Telecommunication Standard* ou le *French Financial Industry's ETEBAC-5 Standard*, ou encore le *X9.44 draft Standard for the U.S. Banking Industry*. Le RSA est programmé aussi dans les systèmes d'exploitation de *Microsoft*, d'*Apple*, de *Sun* et de *Novel*. Il est intégré dans les cartes Ethernet et dans certaines cartes à puce bancaires. Un très grand nombre d'institutions gouvernementales, universitaires ou militaires l'utilisent de façon interne. Le réseau Internet en fait un usage systématique pour assurer la confidentialité du courrier électronique et authentifier les utilisateurs. Bref, le RSA est partout.

Nous avons vu que le système RSA est fondé sur la particularité qu'une opération inverse est plus difficile que l'opération directe. Examinons alors les opérations arithmétiques de RSA qui ont cette propriété.

Protocole RSA pour envoyer un message crypté :

(a) Création des clefs. Destinataire construit un quadruplet de nombres *(p, q, e, d)* tel que : p et q sont deux nombres premiers ; on pose $n = pq$; e est un entier premier avec le produit $(p-1)(q-1)$; d est un entier positif tel que $ed-1$ est un multiple de $(p-1)(q-1)$, c'est-à-dire tel que $ed = 1 \bmod (p-1)(q-1)$.

On sait alors d'après l'énoncé du théorème du RSA que, si A est un entier quelconque, alors : $A^{ed} = A \pmod n$, et c'est cette identité qui va tout faire fonctionner.

Grâce aux algorithmes probabilistes, on sait engendrer des nombres premiers de plusieurs milliers de chiffres de long en peu de temps ; le calcul de p et q est donc rapide. Trouver e et d est aussi une opération rapide, car l'algorithme d'Euclide qui permet de calculer ces deux nombres est un algorithme peu coûteux à exécuter. Au total, la constitution d'un quadruplet *(p, q, e, d)* est donc une opération rapide pour un ordinateur, même si l'on souhaite que p et q aient quelques centaines ou quelques milliers de chiffres.

2. SIGNATURE ET CODAGE SIMULTANÉS

SI *Émetteur*, non content d'envoyer un message, désire aussi le signer, de manière que *Destinataire* soit certain que c'est *Émetteur* qui l'a écrit, il code un texte de signature avec sa clef secrète **Pri'** et obtient le message codé B *(d)*. Il adjoint alors le message B au texte initial et il code le tout avec la clef publique **Pub** de *Destinataire (e)*. Il obtient ainsi le message A'. *Destinataire* décode le message A' avec sa clef privée **Pri**, puis le message B avec la clef publique d'*Émetteur* **Pub'**. Cette technique de codage repose sur la propriété : **Pub'**(**Pri**(**Pub**(**Pri'**(texte)))) = (texte).

3. LE THÉORÈME DU RSA

Le protocole RSA se fonde sur le théorème suivant.
Soient p et q deux nombres premiers. On pose $n = pq$. Si le nombre e est un entier premier avec le nombre $(p-1)(q-1)$, alors il existe un entier d positif, tel que $ed = 1 \bmod (p-1)(q-1)$, et, pour cet entier d et un entier A quelconque, on a : $A^{ed} = A \pmod n$.
La démonstration n'utilise que des mathématiques parfaitement connues depuis le XVIIIe siècle. Il est amusant de voir que les mathématiques du RSA attendaient depuis deux siècles qu'on leur trouve une application : les mathématiques étaient en avance. Ce n'est pas toujours le cas, et, aujourd'hui, en matière de preuve de complexité, la cryptographie attend des résultats mathématiques qui ne viennent pas (concernant les liens entre RSA et factorisation, ou concernant la difficulté de la factorisation) : les mathématiques sont en retard!

(b) Destinataire rend publics n et e, qui constituent la clef publique. Il la publie dans un annuaire ou la communique à *Émetteur*, qui la lui demande. Il ne communique surtout pas p, q ou d. Les nombres p et q peuvent être oubliés, car ils ne serviront plus à personne. Le nombre d constitue la clef secrète de *Destinataire*.

(c) Émetteur, qui veut transmettre une information secrète à *Destinataire*, transforme son information en un nombre entier A, inférieur à n (ou en plusieurs si nécessaire), en utilisant des conventions connues de tous (provenant, par exemple, des codes numériques des caractères typographiques, ou en prenant $a = 01$, $b = 02$, etc.).

(d) Émetteur calcule, *(voir la figure 4)*, grâce à la méthode d'exponentiation rapide, $B = A^e \pmod n$, envoie B à *Destinataire* par un canal qui n'a pas besoin d'être protégé (par exemple, le courrier électronique).

(e) Destinataire, pour décoder B, calcule $B^d \pmod n$, ce qui lui redonne A, car, d'après le théorème du RSA, on a $B^d = A^{ed} = A \pmod n$.

UTILISATION ET SÉCURITÉ DU RSA

Avec les techniques utilisées aujourd'hui pour programmer les systèmes RSA, les spécialistes estiment que doubler la longueur des clefs multiplie le temps de création des clefs par 16 (croissance en L^4, L étant la longueur des clefs) et multiplie, en revanche, le temps des opérations de codage, décodage, signature et authentification seulement par 4 (croissance en L^2). En se fondant sur l'hypothèse que casser le RSA demanderait des algorithmes non polynomiaux, c'est-à-dire plus longs, certains déduisent que, plus les machines seront puissantes, plus on pourra utiliser facilement de longues clefs, et que donc plus l'écart entre la puissance disponible et celle nécessaire pour casser le RSA sera grand. Autrement dit, plus le temps passe, plus RSA serait robuste et sûr. Nous verrons plus loin que cette présentation des choses est très optimiste et loin d'être mathématiquement établie.

Sur le plan pratique, les experts recommandent aujourd'hui d'utiliser des nombres n de 768 bits (232 chiffres décimaux) pour mettre en œuvre le RSA dans le cas de données pas trop importantes, mais ils conseillent 1 024 bits (309 chiffres décimaux) pour un usage commercial et 2 048 bits (617 chiffres décimaux) pour avoir une garantie se prolongeant sur une longue période de temps. Les clefs de 512 bits (155 chiffres décimaux), encore très utilisées, ne devraient plus l'être, car on a réussi, en août 1999, à factoriser un nombre n (produit de deux grands nombres premiers) de 512 bits.

La confiance dans la sécurité du RSA n'est pas due à la démonstration théorique que ce système est sûr, car une telle démonstration n'existe pas (nous allons y revenir). La confiance affichée provient de l'échec, répété depuis plus de 20 ans, de toutes les tentatives entreprises pour casser ce système, tentatives qui n'ont conduit qu'à la formulation de quelques recommandations pour le choix des paramètres p, q, e, d. D'autres systèmes concurrents du RSA sont peut-être aussi bons ou même meilleurs, mais aucun ne bénéficie de la validation par l'usage et la robustesse aux attaques dont le RSA peut se prévaloir. Le fait d'avoir été l'un des premiers lui a donné un avantage qui, puisqu'il résiste aux tentatives d'effraction, le maintient en tête.

Sur le plan théorique, la situation est décevante et le restera certainement longtemps. Celui qui sait factoriser $n = pq$ retrouve ensuite facilement d. Inversement, les mathématiques montrent que celui qui connaît n, e et d peut trouver rapidement p et q. La robustesse du RSA apparaît donc liée à la difficulté de la factorisation. Malheureusement, il n'est pas exclu que quelqu'un, sans réussir à obtenir d ni p ni q, puisse décrypter un message : autrement dit, il n'a pas été prouvé que la difficulté du RSA est équivalente à celle de la factorisation. Deux attaques théoriques du RSA sont donc envisageables. Celle passant par la factorisation : quiconque sait factoriser les nombres de la taille de $n = pq$ sait lire comme à livre ouvert tous les messages cryptés par le RSA. Celle contournant la factorisation, dont personne n'a su établir qu'elle était impossible et pour laquelle, au contraire, on a proposé récemment des arguments indiquant qu'elle devait être sérieusement crainte. Dan Boneh et Ramarathnam Venkatesan ont en effet établi en 1998 que casser le RSA, lorsqu'il est utilisé avec des exposants e trop petits, n'est pas équivalent à factoriser n.

TENTATIVES POUR BRISER LE RSA

D'autres consignes et mises en garde sont d'ailleurs formulées à destination des utilisateurs du RSA. Certains nombres premiers p et q doivent être évités : on conseille de prendre p et q de taille proche, et d'éviter que les nombres $p + 1$, $p - 1$, $q + 1$, $q - 1$ soient faciles à factoriser (car alors n risquerait de l'être aussi par l'utilisation d'algorithmes spécialisés de factorisation qui savent tirer parti des factorisations de $p + 1$, $p - 1$, $q + 1$, $q - 1$). Un conseil évident, mais à ne pas oublier, est que le nombre n ne doit pas être choisi par plus d'une entité : sinon, l'utilisation des diverses clefs publiques des utilisateurs du même nombre n risquerait de donner accès aux facteurs de n.

Plus subtil, mais très important, il ne faut jamais accepter de signer un message en apparence aléatoire soumis par un inconnu. En effet, en choisissant bien ce message, l'inconnu peut en déduire la signature d'un message qu'il aura choisi lui-même au préalable, et vous vous retrouveriez donc à devoir assumer une signature que vous n'avez pas donnée.

L'utilisation d'exposants e trop petits doit être évitée.

Il faut compléter les textes, avant de les coder, par des bouts choisis aléatoirement et surtout pas par de longues séries de blancs ou de zéros.

Une attaque astucieuse proposée par P. Kocher consiste à mesurer minutieusement le temps de calcul (ou la consommation électrique) demandé pour le codage, ce qui permet, si l'on dispose de suffisamment de données, de retrouver les clefs utilisées. Appliquée aux cartes à puce, cette méthode se révélerait catastrophique. Heureusement, se prémunir contre cette attaque est assez facile : il suffit de s'arranger pour que le temps de calcul (ou la consommation électrique)