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).