

## CH.5 SYSTÈMES À CLÉ PUBLIQUE

- 5.1 Les clés publiques : RSA
- 5.2 Les clés publiques : le sac à dos
- 5.3 Les clés publiques : le logarithme discret
- 5.4 L'authentification et la signature électronique
- 5.5 Les preuves sans diffusion de connaissances

Codage ch 5 1

### **5.1 Les clés publiques : RSA**

Diffie et Hellman en 1976 ont aussi proposé l'idée de clé publique pour éviter de communiquer des clés.

Une fonction de chiffrement  $e(M, K)$  et la fonction de déchiffrement  $d(M, L)$  telles que  $d(e(M, K), L) = M$ . Les fonctions  $d$  et  $e$  sont connues de tous, ainsi que  $K$ , appelée clé publique.

On exige que les fonctions soient faciles à calculer, qu'il soit également facile de produire des paires  $(K, L)$  mais qu'il soit impraticable de calculer  $L$ , la clé privée, à partir de  $K$ , mais aussi à partir de couples message en clair – message chiffré.

Mais Diffie et Hellman n'ont pas fourni d'exemple explicites.

Les premiers exemples datent de 1978 et sont dus à Rivest, Shamir et Adleman ainsi qu'à Merkle et Hellman, aussi en 1978.

Codage ch 5 2

Le premier exemple est maintenant connu sous le nom de système RSA, initiales de ses inventeurs.

Il est basé sur les deux faits suivants :

- il est facile de trouver de très grands nombres premiers ;
- il est très difficile de factoriser un très grand nombre composé.

Pour tester si un nombre  $n$  de  $k$  bits est premier, il existe des algorithmes probabilistes (Rabin, Solovay – Strassen). Un nombre est choisi aléatoirement entre 2 et  $n - 1$ . Une fonction de cet entier est calculée en un temps linéaire en  $k$ . Si  $n$  est composé, cette fonction vaut 1 dans au plus un cas sur 4. Mais si  $n$  est premier, elle vaut 1 dans tous les cas. Donc, une fois l'entier choisi aléatoirement, si la fonction ne vaut pas 1, on est certain que  $n$  est composé. Si le résultat vaut 1, il y a une chance sur 4 que  $n$  soit composé. Dans ce cas, on refait le test, jusqu'à ce que la probabilité d'avoir un "faux premier" soit négligeable.

Codage ch 5 3

En utilisant ces tests probabilistes, on peut trouver facilement des nombres premiers de plusieurs centaines de chiffres.

En revanche, les meilleurs algorithmes de factorisation sont exponentiels. La factorisation d'un nombre de même taille requiert un temps de plusieurs milliards d'années.

De façon générale, en cas de progrès dans les performances des machines, on double la taille des données...

Le système RSA est le suivant :

- choisir deux nombres premiers  $p$  et  $q$  et calculer  $n = pq$  ;
- choisir un nombre  $d$  inférieur à  $(p - 1)(q - 1)$  premier avec  $(p - 1)(q - 1)$  ;
- calculer l'inverse  $e$  de  $d$  modulo  $(p - 1)(q - 1)$  ;
- la clé publique est le couple  $(e, n)$ .

Codage ch 5 4

Le message est constitué d'un entier  $M$  inférieur à  $n$ .

Le message est chiffré en  $C = M^e \pmod{n}$ , n'utilisant donc que la clé publique.

Le message est déchiffré en faisant le calcul  $D = C^d \pmod{n}$ .

Propriété :  $D = M$ .

Pour le démontrer, nous allons vérifier que  $D = M \pmod{p}$  et  $\pmod{q}$ .

Si deux nombres sont congrus modulo  $pq$ , ils le sont aussi modulo  $p$ .

Par conséquent,  $D = M^{ed} \pmod{p}$ . Mais  $ed = 1 + k(p-1)(q-1)$ .

Donc  $D = M^{1+k(p-1)(q-1)} \pmod{p}$ . Comme  $M^{p-1} = 1 \pmod{p}$

(Euler-Fermat),  $M^{k(p-1)(q-1)} = 1 \pmod{p}$ . Finalement,  $D = M \pmod{p}$ .

On fait de même pour  $q$ . La propriété est démontrée.

Le système possède les propriétés requises par Diffie et Hellman :

Codage ch 5 5

- il est facile de trouver  $p$  et  $q$  grands et premiers ;
- calculer leur produit  $n$  est simple ;
- trouver un entier  $d$  convenable est facile ;
- calculer  $e$  à partir de  $d$  est facile (algorithme d'Euclide) ;
- le calcul de  $M^e$  se fait en un temps  $O(k^2 \log k \log \log k)$ , si  $k$  est la taille occupée par chaque entier ;
- le déchiffrement se fait dans le même temps.

La condition sur l'impossibilité pratique du décryptage n'est pas prouvée, mais simplement conjecturée, à savoir que trouver  $d$  à partir de  $e$  est au moins aussi difficile que de factoriser  $n$ , et que la factorisation est  $\mathcal{NP}$  difficile.

Ce qui est certain, c'est que la connaissance de  $(p-1)(q-1)$  permettrait de casser le chiffre. Et la connaissance de  $(p-1)(q-1)$  équivaut à savoir factoriser  $n$ .

Codage ch 5 6

Ce système est un des plus répandus dans la pratique, soit pour crypter des messages, soit pour crypter des clés de chiffrement d'autres systèmes (C'est le cas de PGP, pretty good privacy de Zimmermann). Il est aussi largement utilisé pour le cryptage des mots de passe et l'authentification.

Dans la pratique, les nombres premiers  $p$  et  $q$  ont la propriété supplémentaire que  $p - 1$  et  $q - 1$  n'ont pas de petits facteurs premiers. Ce sont ce qu'on appelle des nombres premiers sûrs ("safe primes").

Codage ch 5 7

## 5.2 Les clés publiques : le sac à dos

Autre proposition en 1978, par Merckle et Hellmann, utilisant le problème du sac à dos. L'idée semble meilleure que celle de RSA : le problème est lui prouvé être  $\mathcal{NP}$  complet.

La forme utilisée est la suivante :

on donne des entiers  $A = (a_1, a_2, \dots, a_n)$  et un entier  $t$  ;

Existe-t-il une sous-suite de  $A$  dont  $t$  soit la somme ?

Exemple :

$A = (14, 28, 56, 82, 90, 132, 197, 284, 341, 455)$ .

Peut-on écrire 516 comme somme d'éléments de  $A$  ? Et 515 ?

A noter que, si solution il y a, elle n'est pas nécessairement unique ; 515 a trois représentations comme somme.

Codage ch 5 8

Le problème : "l'entier  $t$  est-il somme d'éléments de  $A$  ?" est un problème  $\mathcal{NP}$  complet en  $n$ , longueur de la suite  $A$ .

Mais, pour certaines suites  $A$ , il est facile, même linéaire en  $n$ , de répondre à cette question. Il s'agit des suites *supercroissantes*.

Ce sont les suites telles que, pour tout  $i$ ,  $a_i$  soit strictement plus grand que la somme des  $a_j$ ,  $1 \leq j < i$ .

Par exemple, (1, 3, 5, 11, 35) est supercroissante.

Pour savoir si  $t$  est somme d'éléments d'une telle suite, il suffit d'appliquer un algorithme glouton. Par exemple,  $47 = 35 + 11 + 1$ .

De plus, la solution éventuelle est unique.

Codage ch 5 9

On peut maintenant décrire le chiffrement par sac à dos.

A choisit une suite supercroissante  $A$  et une paire de nombres premiers entre eux  $w$  et  $N$ ,  $N$  étant strictement plus grand que la somme de tous les éléments de la suite supercroissante  $A$ .

Soit  $w'$  l'inverse de  $w$  modulo  $N$ , calculé au moyen de Bézout.

La suite  $A = (a_1, a_2, \dots, a_n)$  avec  $w'$  et  $N$  constitue la clé privée.

La clé publique est constituée de la suite  $E = (e_1, e_2, \dots, e_n)$ , où  $e_i = w a_i \pmod{N}$ .

Lorsque B veut envoyer un message à A, il le découpe en suites de  $n$  bits.

La suite de bits  $(m_1, m_2, \dots, m_n)$  sert à calculer l'entier

$C = m_1 e_1 + m_2 e_2 + \dots + m_n e_n$  qui est transmis à A.

Le problème du sac à dos étant  $\mathcal{NP}$  complet, on peut supposer que la

Codage ch 5 10

connaissance de  $C$  et de la clé publique ne permet pas de reconstituer les  $m_i$ .

Pour décoder, A calcule  $w' C = m_1 w' e_1 + m_2 w' e_2 + \dots + m_n w' e_n$  modulo  $N$ .

Donc  $w' C = m_1 a_1 + m_2 a_2 + \dots + m_n a_n$  modulo  $N$ . La suite des  $a_i$  est supercroissante et  $N$  assez grand, donc  $w' C$  permet de retrouver les  $m_i$ .

Exemple :

Avec la suite  $A = (1, 3, 5, 11, 35)$  et  $w = 5$ ,  $N = 53$ , on calcule  $w' = 32$ .

La clé publique est la suite  $(5, 15, 25, 2, 16)$ .

Si B veut transmettre 10011, il calcule  $5 + 2 + 16 = 23$ .

Au décodage, A calcule  $23 \times 32 = 47 \pmod{53} = 35 + 11 + 1$  qui redonne bien 10011.

NB : le décryptage est rendu plus difficile si on permute les nombres.

Codage ch 5 11

Le système a comme avantage que l'entier  $N$  est aussi gardé secret. Il est donc en pratique impossible de retrouver la suite supercroissante d'origine.

Malheureusement, Shamir (1982) a montré que presque toutes les réalisations pouvaient être résolues en temps polynomial. Ce n'est pas contradictoire avec le caractère  $\mathcal{NP}$  qui ne concerne que le pire des cas.

En 1983, Adleman a réalisé une attaque sur un chiffre sac à dos qui a justifié l'analyse de Shamir.

La confiance n'étant plus possible, ce système n'a plus été utilisé.

Codage ch 5 12

### 5.3 Les clés publiques : le logarithme discret

Dans le chapitre précédent, on a vu que le calcul du logarithme discret est aussi difficile que la factorisation.

L'idée d'utiliser un logarithme discret pour fabriquer un chiffre à clé publique est donc naturelle. Elle a été formalisée par Elgamal (1985).

Les membres d'un réseau de communication se mettent d'accord sur un grand nombre premier  $p$  et une racine primitive  $a$  modulo  $p$ . A noter que trouver une racine primitive demande un certain calcul...

Un utilisateur B choisit une clé privée qui est un entier  $1 \leq x_b \leq p - 1$ . Il calcule sa clé publique  $y_b = a^{x_b} \pmod{p}$ .

Un expéditeur A veut envoyer à B un message constitué d'un entier  $1 \leq M \leq p - 1$ . Il choisit aléatoirement un entier  $1 \leq k \leq p - 1$ , puis calcule  $K = y_b^k \pmod{p}$ .

Codage ch 5 13

Le message chiffré est  $C = (C_1, C_2)$ , avec  $C_1 = a^k$  et  $C_2 = KM$ . Sa taille est double de celle du message.

Pour déchiffrer, B retrouve  $K = y_b^k = a^{kx_b} = C_1^{x_b}$  qu'il peut calculer avec sa clé privée.

Puis il calcule l'inverse de  $K$  modulo  $p$  par Bézout et donc retrouve  $M$ .

Remarques :

L'entier  $k$  change à chaque transmission de message. Donc un même message  $M$  ne sera jamais chiffré de la même façon.

Le récepteur B trouve  $K$ , mais ne sait pas quel entier  $k$  avait été utilisé par A. La connaissance de  $k$  à partir de  $K$  revient en fait à trouver un logarithme discret.

D'autres techniques, s'apparentant au logarithme discret, utilisent des résultats de géométrie algébrique (courbes elliptiques).

Codage ch 5 14

#### 5.4 L'authentification et la signature électronique

Parmi les premières utilisations des chiffres à clé publique, on mentionne le chiffrement des mots de passe.

Le système fabrique un système à clé publique, puis il efface sa clé privée. Celle-ci peut donc être considérée comme irrécupérable.

L'utilisateur choisit un mot de passe. Celui-ci est crypté au moyen de la clé publique diffusée par le système. Il est stocké sous forme cryptée dans le fichier des mots de passe. Lorsque l'utilisateur s'identifie, son mot de passe est de nouveau crypté et le résultat est comparé à ce qui est stocké. La connaissance du mot de passe crypté ne permet pas de reconstituer le mot de passe en clair. La seule façon raisonnable de trouver un mot de passe est une attaque par dictionnaire.

Codage ch 5 15

Une autre utilisation est la signature électronique.

Supposons que A et B communiquent. Chacun a sa clé privée,  $x_a$  et  $x_b$ , et sa clé publique  $y_a$  et  $y_b$ . Ils ont chacun une signature  $s_a$  et  $s_b$  (un morceau de texte, leur nom,...)

A envoie un message à B et veut prouver qu'il en est bien l'expéditeur.

Pour cela, il signe son message avec  $s_a$  et chiffre avec sa clé privée  $x_a$ , soit  $C(Ms_a, x_a)$ . Il transmet le tout, chiffré avec la clé publique de B :  $C(C(Ms_a, x_a), y_b)$ .

A réception, B déchiffre ce que A a envoyé avec sa clé privée. Il trouve donc  $C(Ms_a, x_a)$ , qu'il code maintenant avec la clé publique de A. Il retrouve ainsi  $Ms_a$ . Seul A a été capable de chiffrer avec sa clé privée. Une interception du message ne permet pas de le décrypter, puisqu'il faut la clé privée de B. Donc aucun tiers n'a pu falsifier le message  $M$ . De plus, aucun tiers n'a pu émettre un message en se faisant passer pour A.

Codage ch 5 16

Une variante de la méthode précédente est utilisée pour *sceller* des données. Elle est utilisée lorsque deux partenaires, A et B, concluent un accord sur un document et veulent s'assurer qu'aucun des deux, ni a fortiori un tiers, ne le modifie. Il est utile, dans ce cas, que le document reste en clair pour être rapidement et facilement accessible sans qu'un déchiffrement soit nécessaire ; ce sceau est aussi appelé un certificat. C'est la technique utilisée pour les déclarations de revenus informatisées, mais aussi dans les actions en justice, lorsqu'on veut s'assurer qu'une preuve ne pourra pas être falsifiée par l'une ou l'autre des parties.

Pour cela, une fonction de hachage est utilisée. A partir du document  $D$ , on calcule le résultat de cette fonction, qui est l'empreinte  $h$  du document. Chacun peut la calculer. Les deux parties A et B, comme plus haut, utilisent une clé privée pour crypter  $h$ .

Codage ch 5 17

L'empreinte  $h$  est chiffrée par A et surchiffrée par B avec leurs clés privées. C'est ce qui constitue le sceau. Les clés privées peuvent alors être oubliées. Avec les clés publiques, chacun peut déchiffrer le sceau. Il peut ensuite comparer le résultat avec l'empreinte du document.

Ni A, ni B, ni un tiers ne peut fabriquer un faux sceau. Une modification du document ne cadrera plus avec le sceau, et celui-ci ne pourra pas être modifié pour correspondre au document modifié.

Les fonctions de hachage produisent des résultats de quelques centaines de bits. Cela fournit une quasi certitude que deux documents différents possèdent effectivement des empreintes distinctes.

Codage ch 5 18

### 5.5 Les preuves sans diffusion de connaissances

Cette dernière application ne ressortit pas à proprement parler de la cryptographie. Mais les techniques utilisées sont du même ordre.

Le problème est le suivant : comment un individu A peut-il prouver à quiconque qu'il possède une certaine information sans pour autant fournir aucun renseignement sur cette information.

Le premier exemple d'une telle procédure est celui des codes à clés publiques : A veut montrer qu'il connaît la clé privée d'un certain code à clé publique sans révéler aucune information sur cette clé privée. Pour cela, B choisit un texte connu de lui seul, qu'il chiffre avec la clé publique. Si A dit quel était le texte choisi par B, il a prouvé (avec une probabilité extrêmement élevée...) qu'il possède effectivement la clé privée.

Codage ch 5 19

En généralisant cette idée, si  $n$  est un entier composé, A peut prouver qu'il en connaît une factorisation sans rien révéler sur ces facteurs.

Pour cela, B choisit un nombre  $b$ . Il choisit un texte  $M$ , qu'il chiffre comme  $C = M^b \pmod{n}$ . Il communique à A le chiffré  $C$  et  $b$ . Si A connaît effectivement la factorisation de  $n$ , il peut calculer l'inverse  $a$  de  $b$  modulo  $\varphi(n)$ . Comme on l'a vu, le calcul de  $a$  est facile si on connaît la factorisation de  $n$ . A peut maintenant calculer  $C^a$  modulo  $n$ , qui vaut  $M$  modulo  $n$  (Euler-Fermat). Il montre donc à B qu'il connaît  $a$  et donc probablement la factorisation de  $n$ . Bien sûr, il y a un problème si  $b$  n'est pas premier avec  $\varphi(n)$ . Il faut prévoir un dispositif spécifique. Par exemple, B fournit une liste de  $b$  et A en choisit un dans cette liste...

Codage ch 5 20

En fait, une telle procédure existe pour tout problème  $\mathcal{NP}$  complet.

C'est une procédure de nature probabiliste. B n'aura jamais la certitude que A connaît une solution de l'instance du problème  $\mathcal{NP}$  complet. Mais cette probabilité pourra être rendue aussi proche de 1 que B peut le souhaiter.

Exemple d'une telle procédure pour un problème de circuit hamiltonien. Un certain graphe  $G$  avec  $n$  sommets est donné. A prétend qu'il connaît un circuit hamiltonien dans ce graphe.

Pour réaliser un test, A fabrique un graphe  $G'$  isomorphe à  $G$ . Les arcs de ce graphe sont cachés. Deux fenêtres peuvent être ouvertes pour chaque paire de sommets de  $G'$ . La première permet de savoir si la paire de sommets est connectée par un arc. La seconde permet de savoir si la paire de sommets est connectée par le circuit hamiltonien.

Codage ch 5 21

B peut demander soit que A ouvre toutes les fenêtres du premier type. il peut ainsi vérifier que  $G'$  est bien isomorphe à  $G$ . Ou alors il demande l'ouverture des deux fenêtres pour le circuit hamiltonien. A ouvre alors les fenêtres du deuxième type réalisant ce circuit, dont B peut vérifier qu'il est hamiltonien et est constitué d'arcs de  $G'$ .

A ne sait pas quelle demande fera B. Il ne peut satisfaire les deux sans connaître un circuit hamiltonien de  $G$ . S'il exhibe un circuit hamiltonien pour  $G'$ , B ne pourra pas retrouver l'isomorphisme avec  $G$  en un temps moindre que  $n!$ , aussi grand que le temps nécessaire pour trouver un circuit hamiltonien dans  $G$ .

Ce test peut être répété autant de fois que B le souhaite.

On peut fabriquer des protocoles identiques pour les autres problèmes  $\mathcal{NP}$  complets, tels que 3-satisfiabilité, nombre chromatique, sac à dos...

Codage ch 5 22