Les révolutions cryptographiques autour du XX^{e} siècle

Hervé Lehning

Cacher ses messages semble une attitude vieille comme le monde. Les anciens ont inventé deux techniques pour cela : la stéganographie qui cache l'existence même d'un message et la cryptographie qui en dissimule le sens. Dans cet article, nous nous focalisons sur la cryptographie moderne tout en évoquant la cryptographie classique qui explique les idées modernes.

Les chiffrements classiques

Les chiffrements classiques reposent sur deux méthodes : substitutions et transpositions. L'idée des chiffrements par substitution est associée à Jules César, qui chiffrait ses messages en décalant les lettres. Par exemple, un décalage de trois lettres consiste à remplacer A par D, B pas E, etc. La méthode fut généralisée ensuite en utilisant des alphabets chiffrés, où chaque lettre est remplacée par un symbole différent. Dès le IX^{e} siècle de notre ère, Abu Yusuf Al-Kindi décrit la méthode des fréquences, qui permet de décrypter un texte chiffré ainsi, à condition qu'il soit assez long. Par exemple, en français, on repère vite le symbole remplaçant la lettre e. Un scientifique italien de la Renaissance, Giambattista della Porta, inventa une autre méthode de décryptement, celle du mot probable : un mot dont on attend la présence. Armé de ces deux méthodes, les chiffres par substitution simple s'écroulent vite.

Dès la Renaissance, Blaise de Vigenère conçut un système de substitution poly-alphabétique dépendant d'une clef. Dans celui-ci, la clef ABC transforme *cryptographie* en *csapuqgscpike*, la première lettre étant conservée (A), la seconde décalée d'un cran (B), la troisième de deux (C), etc. Une autre méthode consiste à permuter les lettres d'un message. Pour ce faire, une idée consiste à disposer le texte à chiffrer en colonnes, en ajoutant quelques lettres pour obtenir un rectangle :

```
ledecryptemen
tdunmessagech
iffrepartrans
positionestpl
usdelicatquec
eluidunmessag
echiffreparsu
bstitutionxxx
```

On modifie ensuite l'ordre des colonnes puis on réécrit le texte obtenu par lignes. La clef est ainsi le nouvel ordre des colonnes. Par exemple, si l'on écrit d'abord les colonnes paires puis les impaires, on obtient : *eerpeeldcytmndne… tttoxx*. Le décryptement pour qui ignore la clef est délicat mais loin d'être impossible. Les mathématiques permettent de trouver le rectangle utilisé, un raisonnement linguistique est ensuite nécessaire pour déterminer la transposition mise en œuvre.

En 1883, Auguste Kerckhoffs a formalisé un principe étonnant, généralisant les deux méthodes que nous venons d'exposer : la sûreté d'un système cryptographique ne peut pas reposer sur le secret de la méthode de chiffrement. Pour être valide, le système doit dépendre d'un paramètre facilement modifiable : sa clef. En 1949, le grand cryptologue Claude Shannon a résumé ce principe en une phrase : « l'ennemi connaît le système ».

L'ère numérique et le masque jetable

De nos jours, tout message, quel que soit son contenu, est une suite de bits qu'on peut considérer comme un nombre écrit en binaire ou comme un mot sur l'alphabet {0, 1}. Dans ce cadre, la méthode de Vigenère consiste à additionner des nombres (avec la règle 1 + 1 = 0) comme le montre le tableau suivant où nous avons chiffré le mot 100111001010 avec la clef 110, pour trouver 010001111100 :

clair	1	0	0	1	1	1	0	0	1	0	1	0
clef	1	1	0	1	1	0	1	1	0	1	1	0
chiffré	0	1	0	0	0	1	1	1	1	1	0	0

En 1949, Claude Shannon a montré que cette méthode, utilisée avec une clef aléatoire aussi longue que le message et jetée après usage, est inviolable. Pourquoi la jeter ? Tout simplement parce que si on ne le fait pas, en gardant les messages successifs, la longueur de la clef diminue à chaque usage. Avant la Seconde Guerre mondiale, d'après les archives britanniques, les Soviétiques utilisèrent cette méthode en gardant la même clef deux fois, les Britanniques surent utiliser cette erreur pour décrypter leurs messages. Il est probable que les Soviétiques aient continué longtemps à agir ainsi sans que personne ne le leur dise, bien entendu. D'autre part, cette méthode qu'on nomme le masque jetable

était le chiffrement utilisé par le téléphone rouge et dans la machine TAREC de l'OTAN.

Suites pseudo-aléatoires

La sécurité du masque jetable, et d'autres méthodes que nous verrons plus loin, dépend du côté aléatoire de la clef, ce qui amène la question : comment fabriquer de l'aléatoire ? Souvent on se contente de pseudo-aléatoire, c'est-à-dire de suites déterministes dont seul le premier terme est aléatoire. En voici une qui fut utilisée autrefois, et qui est typique d'un grand nombre de suites pseudo-aléatoires. On part d'un nombre pris au hasard, que l'on nomme traditionnellement le germe, on le multiplie par 16 807 puis on prend le reste du résultat dans la division par 2 147 483 647. Cela donne une suite de nombres compris entre 0 et 2 147 483 646 qui prend bien des aspects du hasard. Par exemple, si nous utilisons 21 comme germe, nous obtenons la suite 21, 352 947, 1 637 012 935, 1 863 396 828, 1 356 463 995, etc. Nous pouvons convertir ces nombres en binaire et les accoler les uns aux autres, nous obtenons 10101 10101 10001 01011 00111 10000 11001 00101 10101 01110 00111 11011 11000 10001 00101 10111 01110 01010 00011 01100 11111 11110 11110 11.

Une telle suite ressemble à une suite aléatoire. Pourtant, si nous connaissons la règle, elle est prévisible dès que l'on connaît son premier terme. Les masques jetables pseudo-aléatoires ne sont donc pas sûrs. C'est le cas du système RC4 au cœur des clefs WEP et WPA du système Wifi. Ils sont cassables en moins de deux minutes dès lors qu'on est muni du matériel adéquat. Si vous désirez utiliser un algorithme encore sûr à l'heure actuelle, il faut vous tourner vers le système WPA-2 dont le cœur est l'algorithme A.E.S. qui sera évoqué plus loin. Les communications des téléphones portables GSM sont chiffrées de façon comparable, par un algorithme portant le nom d'A5/1. De nos jours, une communication peut être décryptée en temps réel dès lors qu'elle dure au moins deux minutes. Pour cela, il faut cependant disposer des informations et de l'équipement nécessaires… et réussir à capter la conversation qui nous intéresse.

Création d'aléatoire et cryptographie quantique

Pour éviter le type de décryptement que nous venons d'examiner, il est nécessaire d'injecter le maximum de vrai hasard dans la construction de la suite, pas seulement dans le germe. Pour cela, on utilise l'heure actuelle en secondes, le temps d'accès au disque dur, le bruit créé par un micro enregistrant le vent, etc. À chaque étape de la suite, nous introduisons un aléa de ce type. Il devient alors difficile de déterminer la clef. D'autre part, une faiblesse du masque jetable est le transfert de cette clef. Les ambassades utilisent la valise diplomatique pour cela mais la méthode ne peut pas fonctionner pour les opérations routinières, comme le commerce sur Internet par exemple. Pour cela, de nos jours, on utilise d'autres types de chiffrement.

Cependant, la cryptographie quantique permet cette transmission et permet de plus de créer de vraies clefs aléatoires. Sans rentrer dans les détails (voir [1]), cela vient du principe d'incertitude d'Heisenberg. La méthode autorise une transmission sûre des clefs du masque jetable ou de tout autre système cryptographique. Elle permet également la création de clefs véritablement aléatoires. Contrairement à l'ordinateur quantique que nous verrons plus loin, la cryptographie quantique est déjà une réalité. Cependant, vitesse et distance de transfert restent limitées : une centaine de bits par seconde pour la distance maximale de 300 kilomètres. Pour lever cette limitation, la voie poursuivie est l'utilisation de satellites. Malgré les annonces chinoises, les difficultés restent nombreuses et, comme toujours dans le domaine du secret, il est difficile de connaître tous les résultats des recherches entreprises.

Les chiffrements par blocs

Le masque jetable fonctionne par flot continu. Pour être sûr, il exige une clef aléatoire aussi longue que le message. Une autre méthode consiste à découper le message par blocs, de 64 bits en général, et de traiter chaque bloc indépendamment. Elle correspond à une adaptation des chiffres de l'époque de la Première Guerre mondiale. On effectue d'abord une permutation sur le bloc puis des substitutions de type Vigenère mélangées avec d'autres permutations… et on recommence un plus ou moins grand nombre de fois selon les méthodes. Elles ont l'avantage d'être rapides puisqu'elles correspondent à des additions. Nous n'en détaillerons aucune ici (voir [1] et [3] pour des détails au niveau de la vulgarisation, [2] et [4] pour une étude mathématique plus poussée).

Quand, en mai 1973, le National Bureau of Standards lança un appel d'offres pour un système de chiffrement, IBM proposa son chiffre nommé Lucifer, un procédé créé par Horst Feistel. Il remporta le concours et fut modifié pour devenir le D.E.S. Ce chiffre scinde d'abord le message en blocs de 64 bits, pour les chiffrer grâce à une clef de 56 bits, utilisée de façon subtile avec une suite de permutations et de substitutions ce qui, mathématiquement, correspond bien à des additions.

L'algorithme de chiffrement de D.E.S. étant entièrement connu, comme Kerckhoffs l'avait prévu, sa seule véritable protection est sa clef qui, comportant 56 bits, peut prendre

2^{56} valeurs distinctes. Malgré l'énormité de ce nombre, on peut imaginer d'essayer toutes les clefs possibles l'une après l'autre et examiner le texte obtenu à chaque fois pour voir si la clef est la bonne. On parle de recherche exhaustive ou d'attaque par force brute. Une machine dédiée a été fabriquée uniquement pour montrer que l'algorithme D.E.S. n'offre plus la sécurité requise. À elle seule, cette machine de moins de 200 000 € peut décrypter un message en quatre jours. Pour cette raison, D.E.S. a été abandonné au début des années 2000. Il reste utilisé pour certaines applications, comme le cryptage de chaînes de télévision.

Pour pallier cette faiblesse du D.E.S., on a imaginé de le tripler en utilisant deux clefs D.E.S. ce qui le met à l'abri des attaques exhaustives. Cependant, la lenteur de l'algorithme a fait qu'il a laissé la place à un nouvel algorithme : l'A.E.S. Celui-ci opère sur des blocs de 128 bits avec une clef de 128 bits. Il est donc pour l'instant à l'abri d'une recherche exhaustive. Bien que fondé sur des principes similaires au code D.E.S., il donne un algorithme plus rapide (voir [2]). Sauf erreur de protocole, il peut être considéré comme sûr. Il existe énormément d'autres algorithmes de ce type, parmi lesquels I.D.E.A. est très utilisé.

La symétrie des clefs

Dans tous les chiffres rencontrés jusqu'à présent, savoir chiffrer implique savoir déchiffrer. Chiffrement et déchiffrement sont symétriques. Il peut sembler étrange qu'il puisse en être autrement. Pourtant, il existe des chiffres où savoir chiffrer n'implique pas que l'on sache déchiffrer. Ces chiffres sont dits asymétriques. Cela est très utile pour les échanges de clefs symétriques (pour le négoce sur Internet et la sécurisation des courriels) ou l'utilisation des cartes bancaires.

Prenons ce dernier exemple. Quand vous payez avec votre carte, le terminal envoie les données de manière chiffrée au centre de traitement. Le chiffre utilisé est connu puisque les algorithmes et la clef sont contenus dans le terminal. Vouloir les garder secrets serait vain comme l'a souligné Kerckhoffs. Si ces données suffisaient pour déchiffrer le message, il serait facile à un escroc de vous voler. Le même problème se trouve dans le négoce sur Internet et les échanges de courriels. Pour cette raison, on utilise des chiffres à clefs asymétriques. Même en connaissant la clef de chiffrement, l'escroc ne pourra pas déchiffrer. Dans ce cas d'asymétrie, on parle de clef publique pour chiffrer et de clef privée (ou secrète) pour déchiffrer.

Le système asymétrique le plus utilisé correspond à l'élévation à une puissance donc est coûteux en temps et en énergie, il s'agit du système R.S.A. nommé d'après les initiales des trois inventeurs : Ronald Rivest, Adi Shamir et Leonard Adleman, qui l'imaginèrent en 1977. Comme pour tous les systèmes asymétriques, l'idée de départ repose sur une difficulté mathématique : ici, celle de factoriser les nombres. La clef publique est essentiellement le produit de deux nombres premiers et la clef privée est constituée de ces deux nombres. En théorie, la clef publique mène à la clef privée mais, en pratique, si les nombres sont suffisamment grands, personne n'est capable aujourd'hui de la trouver. S'il est simple de trouver que 6 est égal à 2 fois 3, il est plus difficile de factoriser 221 091 801 607 en 470 201 fois 470 207. Un ordinateur bien programmé est nécessaire pour cela. En revanche, il est facile de vérifier que ces deux derniers nombres sont premiers et que leur produit est bien 221 091 801 607. En 1983, le groupement des cartes bancaires a adopté la cryptographie R.S.A. avec une clef publique de 320 bits que personne à l'époque n'était capable de factoriser.

S'interrogeant sur la fiabilité des cartes bancaires, un ingénieur, Serge Humpich est parvenu à comprendre les mécanismes d'authentification sous-jacents et, en 1997, factorisa la clef. Il conçut alors de fausses cartes bleues dont la puce répondait « oui » à n'importe quelle demande. Pour cette raison, elles ont été nommées « yescards ». Humpich contacta alors le groupement interbancaire pour négocier sa découverte d'une faiblesse dans le protocole des cartes bancaires. De façon très logique, le groupement lui demanda de prouver ses dires. Humpich acheta 10 carnets de tickets de métro auprès d'un distributeur de la RATP à l'aide de 10 numéros de cartes bancaires inexistantes. Le groupement feignit alors de vouloir négocier mais mena une enquête pour remonter à l'auteur de ces pratiques « frauduleuses ». Il ordonna une perquisition du domicile de Serge Humpich. En représailles, sa découverte fut alors diffusée publiquement sur Internet en juin 1999. Serge Humpich fut ensuite condamné à dix mois de prison avec sursis en février 2000. Malgré tout, le groupement interbancaire comprit la leçon puisque, pour combler la faille découverte par Humpich, il utilise dorénavant un nombre de 768 bits. Cela sera suffisant jusqu'à ce que quelqu'un parvienne à factoriser ce nombre. Sachant que l'on sait aujourd'hui factoriser des nombres de 700 bits, le compte à rebours est lancé mais il est peu probable qu'un autre ingénieur indépendant prévienne le groupement interbancaire.

Détails de R.S.A.

La méthode R.S.A. consiste à choisir deux nombres premiers, 5 et 11 pour rendre les calculs faciles à suivre même si on utilise de très grands nombres dans la pratique ! On considère alors le produit des deux nombres qui les précèdent, soit 40 ici (4 x 10), et on choisit un nombre premier avec 40,

comme 3. La clef publique est prête, il s'agit du couple (55, 3). Voici comment elle fonctionne sur l'exemple du nombre 28. On l'élève à la puissance 3, on trouve 21952 dont on garde le reste dans la division par 55, soit 7 qui est donc le chiffré de 28. Ce calcul de puissance explique pourquoi le système R.S.A. est gourmand en temps et en énergie. C'est ainsi qu'il ne peut être utilisé dans les petits objets connectés, en particulier ceux qu'on implante dans le corps humain comme les *pacemakers* : la chaleur dégagée par l'objet serait une cause d'inconfort. Ces objets sont difficiles à sécuriser, et peuvent donc être piratés.

Le déchiffrement demande la factorisation du nombre 55, qui est 5 x 11 et un nombre b tel que $3\,b - 1$ soit divisible par 40 (3 et 40 sont les deux nombres rencontrés précédemment). Un tel nombre existe car 3 et 40 sont premiers entre eux. On trouve que 27 convient puisque 3 x 27 – 1 = 80. Le déchiffrement consiste à reprendre les opérations de chiffrement en remplaçant 3 par 27. On élève donc 7 à la puissance 27, on trouve 657123623635342801395 43 dont le reste dans la division est 28. Nous avons donc bien déchiffré 7.

Décrypter R.S.A. avec les moyens actuels est considéré comme hors de portée dès que la clef atteint les 2048 bits. L'avènement d'un ordinateur quantique puissant changerait la donne. Un algorithme fonctionnant sur un tel ordinateur, l'algorithme de Shor, permettrait de décrypter R.S.A. pour des clefs bien plus grandes car son temps d'exécution est polynomial et non exponentiel (en le nombre de chiffres) et donc plus rapide et moins sensible à la taille de la clef. Cependant, la mise au point d'un ordinateur quantique pose des problèmes théoriques profonds et il n'est pas certain qu'il voie le jour dans un avenir prévisible.

Les courbes elliptiques

La recherche actuelle s'oriente vers des chiffrements du type R.S.A., mais encore plus subtils. Ils s'appuient tous sur la théorie des groupes. Les groupes les plus prometteurs en matière de cryptographie ont un fondement géométrique. Ils portent le nom de courbes elliptiques. Le rapport avec les ellipses est indirect puisqu'il concerne le calcul de leurs longueurs. Nous n'insisterons pas sur ce point car il n'a aucun rapport avec la cryptographie. Une courbe elliptique est une courbe dont l'équation est telle que $y^2 = x^3 - 2\,x + 1$ complétée par un point, dit à l'infini, O. Sur une telle courbe, un procédé géométrique permet d'associer à deux points P et Q un autre que l'on note additivement P + Q. Cette « addition » possède certaines des propriétés de l'addition usuelle. La courbe elliptique munie de cette opération est un groupe commutatif.

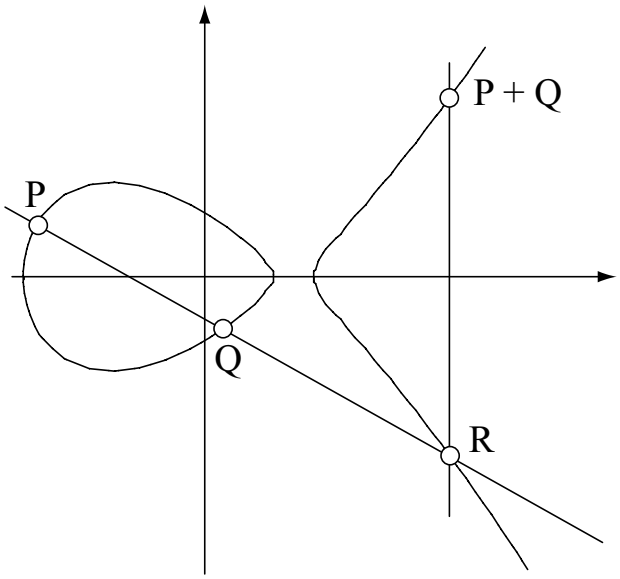

Figure 1 : Une courbe elliptique et la loi de groupe qu'elle définit.

Pour l'utiliser en cryptographie, la courbe est limitée à un nombre fini de points, formant également un groupe. La détermination de ces points passe par des calculs algébriques qu'il est inutile d'effectuer pour comprendre l'esprit de la méthode. L'idée de départ est qu'un texte peut être transformé en une suite de points de la courbe. Cela revient à écrire avec un alphabet ayant autant de signes que la courbe a de points. Notons que le problème sous-jacent n'a rien de simple mais, théoriquement, le chiffrement consiste alors à transformer un point de la courbe en un autre. La clef secrète est constituée d'un point P de la courbe et d'un nombre entier, comme 3 par exemple. On calcule ensuite P ' = 3 P = P + P + P. La clef publique est alors le couple de points (P, P '). Pour chiffrer un point M, le chiffreur choisit un entier, 23 par exemple, et transmet le couple (U, V) défini par U = 23 P et V = M + 23 P '. La connaissance du premier nombre, ici 3, suffit pour retrouver M car M = V – 3 U.

Logarithme discret

Pour retrouver le nombre choisi, 3 dans notre exemple, connaissant P et P ', il suffit de savoir résoudre l'équation P ' = 3 P. L'utilisation du verbe « suffire » ne doit pas tromper. Cela ne signifie absolument pas que cela soit facile mais que, si vous savez le faire, vous savez décrypter. Le nombre 3 est alors appelé un logarithme discret ce qui n'est guère intuitif si on utilise la notation additive ci-dessus. Avec une notation multiplicative de l'opération de groupe, cela devient plus habituel puisque l'équation s'écrit alors $P' = P^3$. Dans l'ensemble des nombres usuels, 3 correspondrait au logarithme de base P de P ' d'où le nom dans le cadre d'un groupe fini. À l'heure actuelle, ce problème est considéré

comme très difficile. On estime qu'une clef de 200 bits pour les courbes elliptiques est plus sûre qu'une clef de 1024 bits pour la méthode R.S.A. Comme les calculs sur les courbes elliptiques ne sont pas compliqués à réaliser, c'est un gros avantage pour les cartes à puces où on dispose de peu de puissance, et où la taille de la clef influe beaucoup sur les performances. Les inconvénients sont de deux ordres. D'une part, la théorie des fonctions elliptiques est complexe et relativement récente. Il n'est pas exclu que l'on puisse contourner le problème du logarithme discret. D'autre part, la technologie de cryptographie par courbe elliptique a fait l'objet du dépôt de nombreux brevets à travers le monde. Cela pourrait rendre son utilisation coûteuse ! De façon plus générale, le problème du logarithme discret débouche sur le cryptosystème El Gamal (voir [2]), qui n'a fait l'objet d'aucun dépôt de brevet.

Signature électronique et hachage

Pour être sûr de ce que l'on signe, une fois signé, un texte ne doit pas pouvoir être modifié. De même, une signature ne doit pas pouvoir être reniée. Ce problème a pris un tour cryptographique depuis l'an 2000 quand il est devenu possible de signer électroniquement, sa déclaration de revenus par exemple. À quoi correspond une telle signature ? L'idée est d'établir des résumés électroniques du document signé afin qu'ils servent de preuves de l'absence de modification. L'élément technique utilisé est nommé une fonction de hachage. Il s'agit d'une application transformant une suite de bits en un résumé de longueur fixe, que certains nomment « haché ». Ces fonctions doivent répondre à quelques impératifs de sécurité. Tout d'abord, comme les résumés sont connus ou faciles à découvrir, ils ne doivent pas permettre de remonter aux originaux. Dans le cas contraire, cela pourrait entraîner des falsifications de signatures. Cette propriété indispensable est appelée la résistance à la pré-image. Cette résistance ne suffit pas pour éviter la fabrication de faux. Une bonne fonction de hachage doit également assurer la résistance à la seconde pré-image. Connaissant le message, il doit être difficile de trouver un autre message ayant le même résumé. Si ce n'est pas le cas, connaissant un de vos contrats signés, on peut en fabriquer un autre et prétendre que vous l'avez signé puisqu'il a le même résumé. Cette résistance est nécessaire pour éviter la contrefaçon. Enfin, une fonction de hachage doit assurer la résistance aux collisions. Il doit être difficile de trouver deux messages ayant le même résumé. Si ce n'est pas le cas, il est possible de vous faire signer un contrat et de prétendre que vous en avez signé un autre. Remarquez que la résistance aux collisions implique la résistance à la seconde pré-image.

Ces principes étant posés, comment réaliser une fonction de hachage ? Comme souvent en informatique, l'idée est de la construire de façon progressive. Imaginons que nous disposions d'une fonction h fournissant un résumé sûr de 128 bits d'un message de 256 bits. Nous découpons alors le message donné M en blocs de 128 bits : M_1, M_2, etc. Nous combinons un message initial de 128 bits à M_1 pour obtenir, grâce à h, un résumé de 128 bits que nous combinons à M_2. Nous obtenons un nouveau résumé de 128 bits et nous recommençons avec M_3. En continuant ainsi, nous obtenons un résumé final de 128 bits, c'est-à-dire une fonction de hachage H résumant tout message en 128 bits.

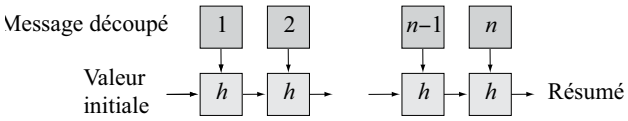

Figure 2 : Construction de Merkle-Damgård.

Merkle et Damgård ont montré que si h est résistante aux collisions alors H l'est aussi ! On peut réaliser une telle fonction de hachage au moyen d'un chiffrement par blocs comme A.E.S., puisqu'il combine une clef secrète de 128 bits à un bloc de 128 bits pour obtenir un bloc de 128 bits. Il suffit d'introduire la clef secrète comme valeur initiale de l'itération. La fonction de hachage obtenue est aussi résistante que le chiffrement utilisé. L'ennui est que A.E.S. ou les fonctions de chiffrement en général sont longues à calculer. C'est pourquoi on a introduit des fonctions de hachage plus rapides tels MD5 sur 128 bits ou SHA-1 sur 160 qui utilisent des chiffrements simplifiés.

Malheureusement les clefs des chiffrements sont trop courtes, ce qui autorise une attaque exhaustive. Le nombre de cas à analyser pour MD5 est de 2^{16} ce qui est bien trop faible à l'heure actuelle. Pour SHA-1, il est de 2^{60} ce qui peut être atteignable en mettant un très grand nombre d'ordinateurs en réseau. Il est possible d'améliorer ces attaques. Cependant, SHA-1 est encore relativement sûr.

Authentification

Les signatures électroniques permettent ainsi d'authentifier l'origine et le contenu des messages sur Internet, à en assurer l'intégrité. Les chiffrements à clefs asymétriques, comme le chiffre R.S.A., fournissent une autre méthode. Imaginons que vous soyez en relation habituelle avec un site marchand. Vous vous connectez. Comment votre navigateur peut-il savoir qu'il n'a pas été piraté et que c'est bien à lui qu'il s'adresse ? Une réponse est dans l'échange de clefs. Le site marchand aura conservé votre clef publique.

Votre navigateur lui envoie un message à chiffrer, vous déchiffrez le message renvoyé avec votre clef privée. Vous vérifiez ainsi que le site possède bien votre clef publique. Inversement, le site peut vérifier votre identité. L'inconvénient est que cette méthode ne fonctionne que si vous êtes déjà client du site. Si ce n'est pas le cas, on fait intervenir une autorité de certification, la plus connue étant *Verisign*. Votre navigateur produit auprès de l'autorité différentes informations personnelles ainsi que votre clef de chiffrement publique. L'autorité émet alors un certificat regroupant ces informations, un numéro de série, une période de validité ainsi qu'une signature numérique qu'elle aura réalisée à l'aide de sa propre clef privée. Lors d'une transaction avec vous, le site commercial reçoit ce certificat. Il prend connaissance de votre clef publique et peut vérifier la validité du certificat à l'aide de la clef publique de l'autorité de certification. En faisant confiance à cette autorité, le site peut vous faire confiance. Bien entendu, le procédé fonctionne dans l'autre sens. Ces différents protocoles sont les bases permettant les échanges d'informations sécurisées sur Internet. Le protocole PGP est de ce type.

Les chiffrements homomorphes et les nuages

Les nuages de l'informatique permettent d'externaliser certaines ressources. Plus besoin de disposer de tous les logiciels sur son ordinateur, il suffit d'utiliser ceux se trouvant dans le nuage qui, de plus, sont toujours à jour. De même, vous disposez de toute la mémoire nécessaire. Bien des gens le font de nos jours en déposant des vidéos sur des sites comme *YouTube*, par exemple. Il en est de même de la puissance de calcul. Malgré ces avantages, le *cloud computing* a deux gros inconvénients. Le premier est de dépendre d'Internet. En cas de coupure du réseau, vous n'avez plus accès à ce que vous avez externalisé. Le second est plus grave, il s'agit du manque de confidentialité. Vous ne savez plus où transitent vos données. Difficile de répondre totalement au premier inconvénient. Certaines fonctions vitales ne devraient jamais passer par les nuages. En revanche, on peut tenter de résoudre le second problème.

Pour sécuriser cette méthode de travail, il est important de pouvoir chiffrer les données envoyées. Cela correspond à une fonction f transformant un nombre en un autre, soit $x \mapsto f(x)$. Les calculs vont alors se faire sur $f(x)$ et non sur x, c'est-à-dire sur le chiffré et non sur le clair. Par exemple, s'il s'agit d'une addition, nous calculons $f(x) + f(y)$. Quelle fonction choisir pour obtenir $x + y$ quand nous déchiffrons $f(x) + f(y)$? La condition la plus simple est que f vérifie $f(x + y) = f(x) + f(y)$ pour tout x et y c'est-à-dire que f soit un homomorphisme additif. Le déchiffrement sera encore possible si $f(x + y) = f(x) * f(y)$ où * est une autre loi, par exemple une multiplication. Nous parlons encore d'homomorphisme additif dans ce cas. Un chiffrement homomorphe additif peut être à la base d'un système de vote électronique. Chaque vote est chiffré. Le déchiffrement de la somme des chiffrés donne le résultat sans que l'on déchiffre pour autant chaque vote.

De manière plus subtile, on peut également imaginer consulter une base de données dans le nuage (c'est-à-dire à distance, via Internet) sans révéler la requête faite. Pour être plus précis, supposons que nous disposions d'une base de n éléments $M_1, M_2, \ldots, M_n$, que nous pouvons considérer comme des nombres entiers, puisque tout message est une suite de 0 et de 1. Nous considérons le polynôme à n indéterminées $P(X_1, X_2, \ldots, X_n) = M_1.X_1 + M_2.X_2 + \ldots + M_n.X_n$.

Si nous voulons atteindre l'élément numéro i, il suffit d'évaluer ce polynôme pour $X_i = 1$ et les autres X nuls car il donne M_i. Si nous disposons d'un chiffrement f homomorphe pour l'addition $P[f(X_1), f(X_2), \ldots, f(X_n)] = f[P(X_1, X_2, \ldots, X_n)]$. On fait donc évaluer le polynôme P pour $f(X_1)$, $f(X_2)$, …, $f(X_n)$, en déchiffrant le résultat, on obtient $P(X_1, X_2, \ldots, X_n)$ c'est-à-dire M_i. Bien entendu, cette méthode est difficilement applicable telle quelle si la base de données est importante car les calculs deviennent gigantesques… mais l'idée est améliorable.

De même, admettons que nous voulions faire effectuer le calcul $x\,y + z$ dans le nuage sans révéler les valeurs de ces trois


Photo Philippe Matsas @ Flammarion

Après des études classiques (Ecole normale supérieure, agrégation de mathématiques, maîtrise d'histoire des religions), **Hervé Lehning** *a enseigné les mathématiques, l'informatique et la cryptologie en écoles d'ingénieurs et en classe de mathématiques spéciales. Il est commandant de réserve, membre de l'Association des réservistes du chiffre et de la sécurité de l'information ainsi que de la Société mathématique de France.*

variables. Si nous disposons d'une méthode de chiffrement homomorphe pour les deux lois f, nous envoyons $f(x)$, $f(y)$ et $f(z)$ dans le nuage via Internet. Le calcul $t = f(x)\,f(y) + f(z)$ est effectué dans le nuage et la valeur de t est renvoyée. Cette valeur est déchiffrée, c'est-à-dire que l'on trouve u tel que $t = f(u)$. Comme $f(u) = f(x)\,f(y) + f(z)$, l'homomorphie de f implique que $u = x\,y + z$. De façon générale, cette méthode permet d'évaluer tout polynôme de plusieurs variables et donc d'effectuer discrètement bien des requêtes possibles à un serveur.

Le chiffrement RSA, est homomorphe, mais pour une seule loi puisqu'il consiste à élever à une puissance. L'existence d'un chiffrement doublement homomorphe sûr a été conjecturée dès 1978 par Rivest, Adleman et Dertouzos. Un cap théorique a été franchi en 2009 quand Craig Gentry inventa le premier chiffrement doublement homomorphe sûr. Malheureusement, l'avancée n'est que théorique car la clef de son chiffre est très longue. Plus de deux gigabits qui le rendent impraticable ! La recherche a donc encore bien des progrès à faire pour que l'on puisse utiliser le *cloud computing* en confiance. ■

Références

[1] Hervé Lehning, *L'univers des codes secrets de l'Antiquité à Internet*, Ixelles éditions, 2012.
[2] Douglas R. Stinson, *Cryptography theory and practice*, Chapman & Hall, 2006.
[3] http://blogs.futura-sciences.com/lehning/ (articles de vulgarisation).
[4] www.lehning.eu (articles plus techniques).
[5] Hervé Lehning, *Toutes les mathématiques du monde*, Flammarion, 2017.