

- [14] T. ElGamal, *A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Trans. Inf. Th., vol. IT-31(4), pp. 469–472, 1985.

Bibliographie

- [1] B. Schneier, *Cryptographie appliquée*, traduction de S. Vaudenay, International Thomson Publishing Company, Paris 1997.
- [2] NBS FIPS PUB 46, *Data Encryption Standard*, National Bureau of Standards, U.S. Department of Commerce, jan. 1997.
- [3] E. Biham, et A. Shamir. *Cryptanalysis of the Full 16-Round DES*, CRYPTO'92, LNCS 740, pp 487–496, 1993.
- [4] J. Stern, *Secret Linear Congruential Generators are Not Cryptographically Secure*, Proc. of the 28th Symposium on Foundations of Computer Science, Pages 421–426, 1987.
- [5] H. Dobbertin. *Cryptanalysis of MD4*, Fast Software Encryption, LNCS 1039, pp 53–69, 1996.
- [6] R. Rivest. *The MD4 Message-Digest Algorithm*, CRYPTO'90, LNCS 537, pp 303–311, 1991.
- [7] *Secure Hash Standard, Federal Information Processing Standard Publication # 180*, U.S. Department of Commerce, National Institute of Standards and Technology, 1993.
- [8] *Secure Hash Standard, Federal Information Processing Standard Publication # 180-1*, U.S. Department of Commerce, National Institute of Standards and Technology, 1995 (addendum to [7]).
- [9] F. Chabaud et A. Joux, *Differential Collisions in SHA-0*, CRYPTO'98, H. Krawczyk ed., LNCS 1462, pp 56–71, 1998.
- [10] R. Rivest. *The MD5 Message-Digest Algorithm*, Network Working Group Request for Comments: 1321, avril 1992. [mailhttp://theory.lcs.mit.edu/rivest/Rivest-MD5.txt](mailto://theory.lcs.mit.edu/rivest/Rivest-MD5.txt)
- [11] B. den Boer, et A. Bosselaers. *Collisions for the compression function of MD5*, EUROCRYPT'93, LNCS 773, pp 293–304, 1994.
- [12] M. Matsui. *Linear cryptanalysis method for DES cipher*, EUROCRYPT'93, LNCS 773, pp 386–397, 1994.
- [13] Ron Rivest, Adi Shamir et Leonard Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, vol. 21(2), pp. 120–126, fév. 1978.

pas une garantie de sa sécurité.

Parmi ces produits, on notera PGP (*Pretty Good Privacy*) qui est un programme gratuit de protection du courrier électronique. Ce programme a valu à son auteur, Philip Zimmermann, quelques déboires avec la justice américaine pour exportation illégale de matériel cryptographique. Le programme était en effet disponible sur Internet. Ce programme utilise RSA pour la signature, MD5 pour le hachage et un algorithme de chiffrement symétrique appelé IDEA. Il est considéré comme l'un des plus sûrs du point de vue cryptographique.

Il existe aussi un certain nombre de normes relatives à l'utilisation de la cryptographie. Ainsi le format de message S/MIME permet d'utiliser des algorithmes cryptographiques pour la protection de messages électroniques. Il ne s'agit pas de protection proprement dite, mais plutôt de garantir une intégration sûre de la cryptographie et un protocole d'échange de messages compris par tous. De même, le format X.509 permet de gérer des certificats de clés publiques.

D'autre part, l'usage de la cryptographie est réglementé dans un certain nombre de pays, dont la France. La nouvelle loi Française [?] prévoit un certain nombre de produits qui sont d'emploi libre. Pour les autres produits, si la cryptographie est utilisée à des fins d'authentification, une déclaration d'emploi est à effectuer. Dans le cas de chiffrement, le contrôle est plus strict et nécessite une demande d'autorisation. L'organisme administratif responsable pour enregistrer les déclarations, délivrer les agréments et autoriser l'utilisation des produits cryptographiques est le

Service Central pour la Sécurité des Systèmes d'Information (SCSSI)
Fort d'Issy
18, rue du Dr. Zamenhoff
92131 Issy-lès-Moulineaux

Bien entendu, ce service, qui dépend directement du Premier Ministre, a pour mission de contrôler l'usage de la cryptographie. Cependant, sa mission est aussi et surtout d'assurer la sécurité des produits cryptologiques. Par le biais de "certifications", le SCSSI garantit à l'utilisateur de produits agréés une sécurité minimale.

5.2.3 Cryptographie et ordinateur quantique

Très récemment, la cryptographie et la physique quantique se sont rejointes sur des sujets d'études communs.

La cryptographie quantique a permis d'élaborer des protocoles d'authentification et d'échange de clés dont la sécurité est basée sur les hypothèses fondamentales de la physique quantique. Des expériences ont même été menées avec un certain succès pour montrer la possibilité de réaliser de tels systèmes. Les hypothèses utilisées sont celles de la dualité onde-corpuscule, du fait qu'une particule isolée est dans un état quantique décrit par une fonction d'onde, et que la mesure de l'état quantique est une opération destructrice de cet état. Ainsi est-on sûr qu'une particule observée n'a pas pu être interceptée par un attaquant éventuel.

Nuançons cependant ce propos, car après tout, rien ne dit que les hypothèses fondamentales de la physique soient plus avérées que celles des mathématiques. En outre, la mise en œuvre de ces systèmes nécessite souvent des artifices (une particule isolée est très difficile à produire et à conserver) qui ruinent le bénéfice de la cryptographie quantique.

À la suite de ces études, le concept d'ordinateur quantique a par ailleurs été introduit. Il s'agit en fait de réaliser une machine de Turing quantique au lieu d'être probabiliste. L'avantage est qu'on peut alors, toujours sous les mêmes hypothèses, faire réaliser à la machine des opérations sur des états quantiques. En choisissant bien ses opérations, et surtout la mesure observée en fin de calcul, il est possible d'obtenir des résultats intéressants. Par exemple, la factorisation ou le logarithme discret sont des problèmes que l'on saurait résoudre en temps polynomial sur un ordinateur quantique.

5.3 Sécurité des implantations

Il faut maintenant attirer l'attention sur un point très important. La cryptographie est utilisée principalement pour assurer une certaine sécurité dans les systèmes d'information. Or la sécurité d'un système, quel qu'il soit, repose toujours sur celle du maillon le plus faible du système. Un système parfait serait conçu de façon telle que la cryptographie soit le maillon faible. En fait, bien souvent, il n'en est rien, et c'est la sécurité informatique du système qui constitue le maillon faible.

Par exemple, nous avons vu que les mots de passe sous UNIX[®] étaient chiffrés à l'aide du DES. La sûreté du DES étant remise en cause, on pourrait en déduire qu'il s'agit là d'une menace contre le système UNIX[®]. En fait, le mode d'utilisation du mot de passe sous UNIX[®] est ainsi fait que la cryptographie n'est absolument pas le maillon faible. En effet, le système d'authentification d'UNIX[®] a été conçu à une époque (pas si lointaine) où les machines n'étaient pas connectées en réseau. Lorsque les réseaux sont apparus, le système d'authentification n'a pas été adapté en conséquence. L'authentification est donc toujours réalisée par la machine sur laquelle on se connecte, et le mot de passe que l'utilisateur tape est donc transmis en clair sur le réseau jusqu'à la machine qui le vérifie. Rien de plus simple dans ces conditions, que de récupérer ce mot de passe sur le réseau, et ceci sans avoir, bien entendu, à casser une instance du DES. Utiliser un produit incorporant de la cryptographie n'est donc absolument

5.2.2 Problèmes NP-complets

La théorie de la complexité permet de classer entre eux des problèmes mathématiques, en définissant des classes d'équivalence. On utilise pour cela la notion de machine de Turing, c'est-à-dire une modélisation de l'ordinateur. Ainsi, la classe des problèmes \mathcal{P} est la classe des problèmes pour lesquels il existe un algorithme qui trouve une solution du problème, et qui s'exécute en temps polynomial sur une machine de Turing. Le fait de s'exécuter en temps polynomial signifie que multiplier par n les dimensions du problème provoque une multiplication par un facteur n^k du temps d'exécution. Par opposition, un algorithme qui s'exécute en temps exponentiel verrait son temps de calcul multiplié par un facteur ℓ^n . La classe \mathcal{P} est donc une classe de problèmes que l'on sait résoudre sur un ordinateur.

On définit aussi la classe \mathcal{NP} des problèmes pour lesquels il existe un algorithme qui s'exécute en temps polynomial et qui permet de vérifier une solution du problème. La nuance est importante. On suppose ici simplement que l'on peut vérifier une solution. Il est clair que $\mathcal{P} \subseteq \mathcal{NP}$. Cependant, le problème suivant reste ouvert :

$$\mathcal{NP} \stackrel{?}{\neq} \mathcal{P}.$$

La conjecture est que les deux classes sont distinctes, car la théorie de la complexité permet de relier entre eux beaucoup des problèmes de la classe \mathcal{NP} . Ces problèmes sont dits \mathcal{NP} -complets, c'est-à-dire que s'il existe un algorithme en temps polynomial qui résout ce problème, alors la conjecture ci-dessus est fautive, et $\mathcal{NP} = \mathcal{P}$. Or le nombre de problèmes \mathcal{NP} -complets est non seulement très important, mais ces problèmes sont très variés et ont été très largement étudiés. Un résultat récent démontre par exemple que le décodage à distance minimale d'un code correcteur d'erreur est un problème \mathcal{NP} -complet.

Les problèmes \mathcal{NP} -complets sont donc parfois utilisés en cryptographie, car ils permettent de vérifier une solution d'un problème en temps polynomial, tout en assurant qu'un algorithme de résolution du problème en temps polynomial a très peu de chances d'exister. Ce type de problème a ainsi été utilisé dans de nombreux protocoles d'authentification à divulgation nulle de connaissance.

Toutefois, le caractère \mathcal{NP} -complet d'un problème ne fait pas forcément la sécurité du système. En effet, un problème peut très bien être \mathcal{NP} -complet mais disposer d'algorithmes polynomiaux qui en résolvent certaines instances. L'exemple du décodage des codes correcteurs est à ce titre significatif, puisqu'il existe des classes entières de codes linéaires que l'on sait décoder en temps polynomial.

En outre, même si on ne connaît que des algorithmes exponentiels de résolution, il arrive souvent que ces algorithmes soient malgré tout suffisants pour résoudre des problèmes de dimension importante. L'utilisation de ces problèmes est alors rendue impossible, car elle conduirait à utiliser des tailles de clés trop importantes.

L'expertise d'un protocole basé sur un problème \mathcal{NP} -complet est donc toujours nécessaire. La seule différence avec un problème réputé difficile est qu'on peut craindre un peu moins l'apparition de méthodes d'attaque révolutionnaire, c'est-à-dire en temps polynomial.

5.1 Les algorithmes empiriques

Les algorithmes tels que RC4 ou SHA-1 sont typiquement des algorithmes empiriques. Leur sécurité n'est pas prouvée, car même si toutes les attaques connues sont inefficaces, rien ne prouve qu'il n'en existe pas d'autre.

5.2 Les algorithmes à sécurité prouvée

5.2.1 Problèmes réputés difficile

Factorisation

L'algorithme RSA est si simple, que le casser semble aisé. Pourtant, connaissant e et n , trouver d implique que l'on sache factoriser n . Or, le problème de la factorisation est étudié depuis les Grecs et il n'est pas si simple qu'il y paraît, dès que l'on cherche à factoriser des grands nombres de 200, 300, 400 chiffres. Le record actuel de factorisation, utilisant un réseau de plusieurs milliers de stations distribuées sur Internet, et fonctionnant pendant plusieurs mois, est de 130 chiffres. Les méthodes employées pour ces factorisations utilisent des techniques d'algèbre très pointues, ce qui montre la difficulté du problème. Les clés actuellement conseillées pour RSA comportent 768 bits ou 1024 bits, soit environ 230 et 300 chiffres. Les méthodes actuelles de factorisation sont donc inopérantes, mais rien ne garantit que de nouvelles méthodes ne voient le jour.

Toutefois, les esprits puristes diront qu'il n'est pas prouvé que casser RSA soit équivalent à effectuer la factorisation du module, puisqu'on pourrait imaginer pouvoir retrouver le message m en fonction de $s = m^e \bmod n$ et n , sans passer par l'obtention de $d = 1/e \bmod pq$. Il conviendrait donc plutôt de classer RSA parmi les algorithmes à sécurité empirique. En fait, l'opinion générale est que malgré tout, une attaque du RSA reviendrait à peu de chose près à une factorisation.

Logarithme discret

Le problème du logarithme discret est par bien des côtés voisin de celui de la factorisation. Les méthodes de calcul sont proches. Cependant, on observe expérimentalement que les dimensions actuellement atteintes pour la résolution d'un calcul de logarithme discret sont de l'ordre de 90 chiffres (au lieu de 130 pour la factorisation).

L'algorithme de ElGamal, le protocole de Diffie-Hellman, le standard DSS sont tous reliés au problème du logarithme discret.

Logarithme discret sur les courbes elliptiques

En fait, le problème du logarithme discret peut se définir sur n'importe quel ensemble fini possédant une structure de groupe. Il est ainsi possible de définir des groupes basés sur la notion de courbe elliptique. Ces groupes sont actuellement très prisés en cryptographie, car ils ne présentent pas la structure de corps indispensable aux attaques connues sur la factorisation ou le logarithme discret. De ce fait les tailles de clés nécessaires sont beaucoup plus faibles que pour les algorithmes cryptographiques basés sur les groupes "classiques".

Chapitre 5

Sécurité des algorithmes cryptographiques

La raison d'être de la cryptologie est la sécurité. Or la sécurité des systèmes cryptographiques peut s'étager en trois niveaux :

1. La sécurité parfaite est liée à la théorie de l'information. Elle repose sur l'incapacité *théorique* de casser certains problèmes, comme par exemple le *One-Time Pad*. Le *One-Time Pad* consiste, pour chiffrer un message m codé sur N bits, à choisir une clé secrète aléatoire k de même longueur codée sur N bits, et à effectuer le ou exclusif des deux quantités $c = m \oplus k$. Le chiffré c est alors une quantité parfaitement aléatoire. Malheureusement, cette même théorie de l'information impose des contraintes très coûteuses à ce type de protocole, à savoir que la taille de la clé doit être de même longueur que le message. La notion de sécurité parfaite n'est donc pas utilisable pour des opérations courantes.
2. La sécurité asymptotique consiste à démontrer le lien entre un procédé cryptographique et un problème mathématique de difficulté connue ou supposée. Casser le système est alors équivalent à la résolution du problème mathématique. En utilisant des résultats de la théorie de la complexité on peut ainsi lier des procédés cryptographiques à des problèmes insolubles en pratique. Ces deux derniers mots sont importants, car il intervient généralement dans ce type de procédé un paramétrage qui fixe la taille du problème mathématique. Or ce paramétrage est le plus souvent déterminé de façon empirique, ce qui nous ramène au troisième type de sécurité.
3. La sécurité empirique consiste à faire expertiser un système *a posteriori* par des cryptanalystes et à fixer un seuil de sécurité. Ce seuil dépend de l'application envisagée. Dans certains cas, on pourra très bien envisager qu'un protocole est sûr si sa cryptanalyse prend 5 minutes sur un ordinateur portable. Dans d'autres on exigera que le système résiste à plusieurs millénaires de temps de calcul.

4.5 Système de chiffrement à clé publique

4.5.1 Algorithme de chiffrement RSA

Il est facile de voir que l'algorithme RSA peut être considéré comme un algorithme de chiffrement. En effet, si nous reprenons les clés publique (n et e) et secrète (d) d'Alice, alors Bob peut chiffrer un message m à destination d'Alice avec la quantité

$$s = m^e \bmod n,$$

Alice déchiffre alors le message m en calculant

$$\begin{aligned} s^d \bmod n &= (m^e)^d \bmod n, \\ &= m^{ed} \bmod n, \\ &= m \bmod n. \end{aligned}$$

4.5.2 Protocole d'échange de clé de Diffie-Hellman

Le protocole d'échange de clé de Diffie-Hellman est conçu pour que deux interlocuteurs puissent se mettre d'accord sur une clé secrète commune sans avoir besoin de se rencontrer auparavant. C'est un système à clé publique basé sur le logarithme discret. On se donne un nombre premier p et un élément g générateur de $\mathbb{Z}/p\mathbb{Z}$, c'est-à-dire tel que tout élément non nul de $\mathbb{Z}/p\mathbb{Z}$ puisse se représenter comme une puissance g^x .

Alors, la clé secrète d'Alice sera un nombre aléatoire x de $\mathbb{Z}/(p-1)\mathbb{Z}$ et sa clé publique

$$y = g^x \bmod p.$$

De même, Bob dispose d'une clé secrète x' de $\mathbb{Z}/(p-1)\mathbb{Z}$ et de la clé publique associée

$$y' = g^{x'} \bmod p.$$

Pour échanger une clé avec Alice, Bob choisit un nombre aléatoire r de $\mathbb{Z}/(p-1)\mathbb{Z}$ et lui envoie $t = g^{rx'}$. Alice calcule alors $t^x = g^{rxx'}$. De son côté, Bob calcule $y^{rx'} = g^{rxx'}$. La clé secrète ainsi échangée est donc $k = g^{rxx'}$. Pourtant, le protocole est sûr car la seule quantité t échangée est un nombre totalement aléatoire dans $\mathbb{Z}/p\mathbb{Z}^*$. Ceci provient du fait que g en est un générateur.

On calcule ensuite

$$\begin{aligned} a &= xr \bmod q, \\ b &= k^{-1} \bmod q, \\ s &= b(\text{SHA}(m) + a) \bmod q. \end{aligned}$$

La signature est constituée du couple (r, s) .

Pour vérifier la signature on calcule successivement

$$\begin{aligned} u &= s^{-1} \bmod q, \\ v &= \text{SHA}(m)u \bmod q, \\ w &= ru \bmod q. \end{aligned}$$

On vérifie ensuite les conditions suivantes :

$$\begin{aligned} 0 &< r < q, \\ 0 &< s < q, \\ r &= (\alpha^v y^w \bmod p) \bmod q. \end{aligned}$$

4.4 Système de certification

La cryptographie asymétrique permet de résoudre de façon élégante le problème de la signature numérique. Toutefois nous avons pour l'instant escamoté une difficulté de mise en œuvre. En effet, dans un système symétrique, la sécurité repose sur le secret des clés. Dans le cas d'un système asymétrique, ce secret n'existe plus pour la partie publique des clés. Or c'est à partir de cette donnée publique que s'effectue la vérification de la signature. Il est donc important que cette partie publique soit **intègre**. Le problème de la garantie de secret des clés a donc été remplacé par un problème de garantie d'intégrité.

Pour assurer cette intégrité des clés publiques, on utilise des systèmes de certification basés sur la signature numérique. Le principe est le suivant :

- Alice s'adresse à une autorité de certification et lui fournit sa clé publique. L'autorité constitue un certificat avec cette clé publique et les informations nécessaires sur l'identité d'Alice et le signe. C'est ce certificat qui est rendu public.
- De son côté, Bob n'a confiance que dans certaines autorités de certification, c'est-à-dire qu'il ne dispose que d'un certain nombre de clés publiques d'autorités de certification. S'il dispose de la clé publique de l'autorité d'Alice, alors il peut directement vérifier le certificat de clé publique d'Alice. Sinon, il doit vérifier le certificat, puis vérifier le certificat de clé publique de l'autorité d'Alice signé par l'une des autorités qu'il reconnaît.

On peut ainsi mettre en place des structures hiérarchiques de certification, aussi appelées infrastructures de clés publiques. Il est important de comprendre que la qualité de la protection en intégrité de ces clés publiques d'intégrité doit être égale ou supérieure à la qualité de la protection en confidentialité des clés privées de chaque usager, car c'est sur cette intégrité que repose tout l'édifice.

Ce problème est donc le calcul d'un logarithme, ce que toute bonne calculatrice est censée savoir faire. Cependant, le problème apparaît beaucoup plus complexe qu'il n'y paraît si au lieu de considérer des éléments dans le corps des réels positifs, on se place dans un groupe multiplicatif fini. On parle alors de "problème du logarithme discret". Ainsi, le schéma de signature de ElGamal [14] repose sur la difficulté de calculer un logarithme discret dans le corps $\mathbb{Z}/p\mathbb{Z}$, où p est un nombre premier.

On se donne un nombre premier p et deux éléments g et x de $\mathbb{Z}/p\mathbb{Z}$ aléatoires. Alors, la clé publique d'Alice sera constituée de p , g et

$$y = g^x \bmod p.$$

La clé privée d'Alice est bien entendu x .

Pour signer un message m , on choisit k aléatoire premier avec $p - 1$ et on calcule

$$a = g^k \bmod p.$$

On utilise ensuite l'algorithme d'Euclide étendu pour déterminer b tel que

$$m = (xa + kb) \bmod (p - 1).$$

La signature est constituée de a et b . La valeur de k ne doit pas être communiquée. Il est alors possible de vérifier la signature du message m en posant :

$$\begin{aligned} y^a a^b \bmod p &= (g^x)^a (g^k)^b \bmod p, \\ &= g^{xa+kb} \bmod p, \\ &= g^m \bmod p. \end{aligned}$$

4.3.3 DSS

L'administration américaine avait son standard de chiffrement et son standard de hachage: il lui manquait un standard de signature! Le *Digital Signature Standard* corrige cette lacune. Il utilise la fonction de hachage SHA-1 et est basé sur des principes similaires à ceux de l'algorithme de ElGamal.

On se donne deux nombres premiers p et q tels que q divise $p - 1$ ¹ et un élément α de $\mathbb{Z}/p\mathbb{Z}$ d'ordre q ². Soit x de $\mathbb{Z}/q\mathbb{Z}$ aléatoire, alors, la clé publique d'Alice sera constituée de p , q , et

$$y = \alpha^x \bmod p.$$

La clé privée d'Alice est bien entendu x .

Pour signer un message m , on choisit k aléatoire dans $\mathbb{Z}/q\mathbb{Z}$ et on calcule

$$r = (\alpha^k \bmod p) \bmod q.$$

¹La fonction de hachage utilisée par le DSS est SHA. Cette fonction fabriquant un haché sur 160 bits, la taille "naturelle" pour q est aussi de 160 bits, car la fonction de hachage est utilisée modulo q . Cependant, rien n'empêche d'utiliser dans un schéma de type DSS une autre fonction de hachage et donc une taille de q différente.

²L'ordre d'un élément est le plus petit entier r non nul tel que $\alpha^r = 1$. Il est facile de voir que dans $\mathbb{Z}/p\mathbb{Z}$, ce nombre est forcément inférieur à $p - 1$. On peut aussi montrer que l'ordre d'un élément divise $p - 1$, d'où la condition sur q .

Bob, de vérifier la signature d'Alice. Toutefois, pour atteindre ses objectifs, le système doit absolument garantir l'intégrité de la clé publique. En effet, dans le cas contraire, il suffirait de se forger un couple de clé privée et publique et de faire passer la fausse clé publique auprès de Bob pour celle d'Alice, pour lui faire croire que c'est Alice qui lui écrit des messages. Il est donc facile de voir que l'exigence de secret absolu observée dans un système symétrique est remplacée dans un système asymétrique par l'exigence d'intégrité absolue.

Voyons maintenant quelques exemples d'algorithmes permettant la réalisation sûre d'un tel concept clé privée, clé publique.

4.3 Algorithmes de signature

4.3.1 RSA

L'algorithme RSA peut être utilisé soit comme algorithme de chiffrement, soit comme algorithme de signature [13]. Il est dû à Ron Rivest, Adi Shamir et Leonard Adleman et porte les initiales de ses auteurs. Sa sécurité repose sur celle de la factorisation des nombres entiers.

Plus précisément, soient p et q deux nombres premiers, on considère

$$n = pq.$$

Alors, connaissant p et q , pour tout entier e premier avec $(p-1)(q-1)$, il est possible de calculer d tel que

$$ed = 1 \pmod{(q-1)(p-1)}.$$

Cette opération s'effectue par l'algorithme d'Euclide dans chacun des corps fini $\mathbb{Z}/p\mathbb{Z}$ et $\mathbb{Z}/q\mathbb{Z}$. Les deux résultats sont alors combinés par le théorème des restes chinois pour fournir d , l'inverse de e dans $\mathbb{Z}/n\mathbb{Z}$, isomorphe à $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$.

La clé publique d'Alice sera constituée de n et e . Sa clé secrète sera d . On considère un message à signer m , dont la représentation puisse être considérée comme un nombre de $\mathbb{Z}/n\mathbb{Z}$. Alors, Alice signe le message m avec la quantité

$$s = m^d \pmod{n},$$

et Bob vérifie la signature du message m en calculant

$$\begin{aligned} s^e \pmod{n} &= (m^d)^e \pmod{n}, \\ &= m^{de} \pmod{n}, \\ &= m \pmod{n}. \end{aligned}$$

4.3.2 ElGamal

Le schéma de signature de ElGamal repose sur le problème mathématique suivant :

Connaissant g et y , déterminer x tel que y soit l'élevation à la puissance x de g .

Chapitre 4

Les systèmes d'authentification et de signature – Concept de cryptographie à clé publique

4.1 Problématique de l'authentification

Dans la vie courante, le besoin d'authentification apparaît souvent. Il est résolu par l'emploi de méthodes anciennes mais qui ont fait leurs preuves... et aussi montré leurs limites (signature, sceau, tampon, etc...). Les méthodes d'authentification classiques reposent sur la difficulté à reproduire ou à falsifier un élément reconnaissable par tous. Lors d'une fraude, la détection est rendue possible par l'expertise physique de la preuve (expertise graphologique, vérification d'un sceau ou d'un tampon).

Le développement des transmissions électronique a fait apparaître un nouveau problème : comment garantir qu'un message a été effectivement écrit par une personne, et comment garantir que cette même personne ne puisse pas nier l'avoir écrit. Cette problématique est celle de la signature électronique.

Pour résoudre ce problème la cryptographie symétrique est totalement impuissante, car elle suppose la possession par deux ou plusieurs entités d'une même clé secrète. Rien n'empêche donc l'une des parties de se faire passer pour une autre personne disposant de la même clé.

4.2 Le concept de cryptographie asymétrique

Le principe de cryptographie asymétrique va au contraire permettre de résoudre ce problème. En effet, en cryptographie asymétrique, Alice qui souhaite s'authentifier va disposer d'un secret qu'elle est seule à connaître. Associée à cette clé privée, une clé publique va permettre à tout le monde, et en particulier à

Pour résumer, l'architecture de SHA peut être illustrée par la figure 3.1.

Le standard est bien sûr complété par la présentation de vecteurs de tests qui permettent de contrôler la validité d'une implantation.

3.1.2 Deuxième version SHA-1

Deux ans après son adoption, l'administration américaine annonce une modification du standard [8] qui prend le nom de SHA-1 (la version initiale s'appellera désormais SHA-0). La seule explication donnée à l'époque est une rumeur sur une attaque du standard SHA-0, qui aurait été découverte par la NSA.

Mais quelle est donc la modification apportée? Elle est minime pour ne pas dire minimaliste. Elle consiste à remplacer dans le processus d'expansion décrit plus haut, l'équation de récurrence (3.1) par l'équation

$$W^{(i)} = \text{ROL}_1 \left(W^{(i-3)} \oplus W^{(i-8)} \oplus W^{(i-14)} \oplus W^{(i-16)} \right), \quad \forall i, 16 \leq i < 80. \quad (3.2)$$

La seule différence est donc cette rotation de un bit vers la gauche.

3.1.3 Recherches de collisions

Rappelons que le paradoxe des anniversaires fournit une attaque générique théorique de toute fonction de hachage. La taille d'un haché dans le cas de SHA est de 160 bits. L'attaque générique est donc en $\sqrt{2^{160}}$. Le fait de trouver une collision n'est pas forcément gênant pour une fonction de hachage, puisqu'il en existe toujours par définition, mais trouver une méthode de construction de telles collisions est plus ennuyeux, car ceci indique une faiblesse de la fonction.

La fonction SHA a été dérivée d'une fonction de hachage proposée par Ron Rivest, MD4¹ [6]. Cette fonction de hachage a depuis été cassée [5]. Cependant, bien que très proche de MD4, les attaques étudiées ne fonctionnaient pas sur SHA.

Récemment, une nouvelle méthode de recherche de collisions sur SHA est apparue [9]. Sa complexité de 2^{61} est inférieure au paradoxe des anniversaires. Elle a donc permis de montrer que SHA-0 présentait une faiblesse. D'autre part, la modification apportée sur SHA-1, pour mineure qu'elle soit, empêche la méthode de s'appliquer.

3.2 MD5

Parallèlement, Ron Rivest a modifié sa fonction initiale MD4, pour obtenir MD5 [10]. Des collisions ont été obtenues sur la fonction de compression de MD5 [11]. Bien que ne s'appliquant pas à la fonction de hachage globale de MD5, ces attaques partielles indiquent des faiblesses de conception.

¹MD signifie *Message Digest*.

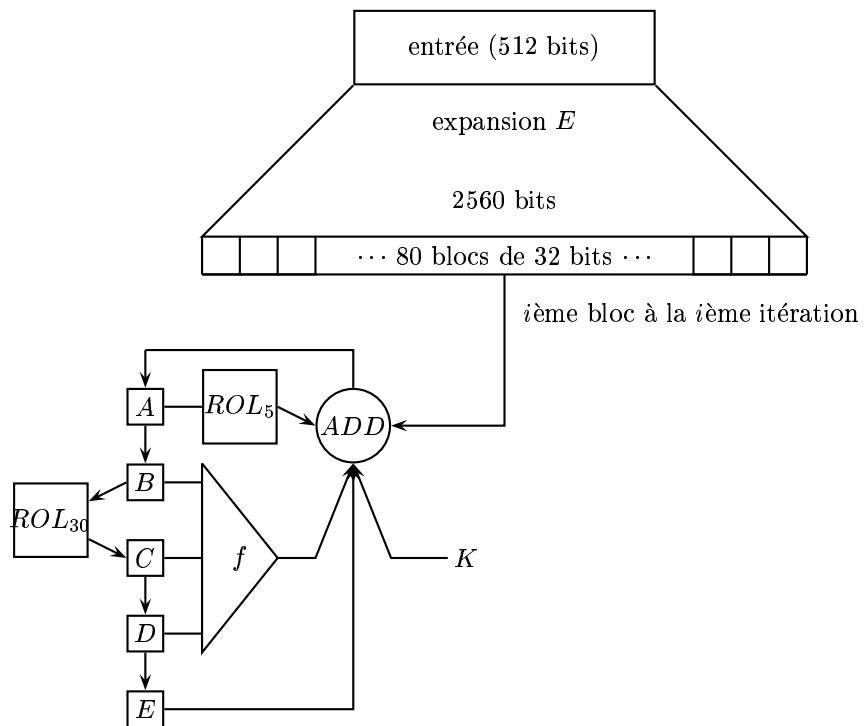


Figure 3.1: Architecture de SHA

Tableau 3.1: Définition de $f^{(i)}(X, Y, Z)$, et des constantes $K^{(i)}$.

Tour i	Fonction $f^{(i)}$		Constante $K^{(i)}$
	Intitulé	Définition	
0–19	IF	$(X \wedge Y) \vee (\bar{X} \wedge Z)$	0x5A827999
20–39	XOR	$(X \oplus Y \oplus Z)$	0x6ED9EBA1
40–59	MAJ	$(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$	0x8F1BCCDC
60–79	XOR	$(X \oplus Y \oplus Z)$	0xCA62C1D6

- $E = 0xC3D2E1F0$

- Pour chaque bloc m_i , copier les registres A, B, C, D et E dans AA, BB, CC, DD et EE et appliquer la fonction de compression H à $(AA, BB, CC, DD, EE, m_i)$. Le résultat obtenu AA', BB', CC', DD' et EE' est alors ajouté à A, B, C, D and E (addition modulo 2^{32}).
- Le résultat du hachage est obtenu par la concaténation des registres A, B, C, D et E obtenus après compression du dernier bloc m_n du message.

Description de la fonction de compression

Voyons maintenant le cœur de la fonction SHA, sa fonction de compression. On note tout d'abord $\langle W^{(0)}, \dots, W^{(15)} \rangle$ les 512 bits de message entrés dans H , représentés par 16 mots de 32 bits. La première étape consiste à effectuer une expansion de ces 512 bits :

$$W^{(i)} = W^{(i-3)} \oplus W^{(i-8)} \oplus W^{(i-14)} \oplus W^{(i-16)}, \forall i, 16 \leq i < 80. \quad (3.1)$$

Ces 80 mots de 32 bits sont utilisés pour altérer l'état interne de l'automate H constitué des cinq registres de 32 bits $A^{(i)}, B^{(i)}, C^{(i)}, D^{(i)}, E^{(i)}$. L'état initial $\langle A^{(0)}, B^{(0)}, C^{(0)}, D^{(0)}, E^{(0)} \rangle$ de H correspond à l'entrée $\langle AA, BB, CC, DD, EE \rangle$ de la fonction de compression.

La modification de l'état interne $\langle A^{(i)}, B^{(i)}, C^{(i)}, D^{(i)}, E^{(i)} \rangle$ est effectuée de la façon suivante :

for $i = 0$ to 79

$$\begin{aligned} A^{(i+1)} &= ADD(W^{(i)}, \text{ROL}_5(A^{(i)}), f^{(i)}(B^{(i)}, C^{(i)}, D^{(i)}), E^{(i)}, K^{(i)}) \\ B^{(i+1)} &= A^{(i)} \\ C^{(i+1)} &= \text{ROL}_{30}(B^{(i)}) \\ D^{(i+1)} &= C^{(i)} \\ E^{(i+1)} &= D^{(i)} \end{aligned}$$

La fonction $f^{(i)}$ et la constante $K^{(i)}$ sont définis par le tableau 3.1, et $ADD(U, V, W, X, Y) = U + V + W + X + Y \pmod{2^{32}}$. Enfin ROL_i désigne la fonction qui sur un entier de 32 bits i effectue une rotation à gauche de i bits.

La sortie de la fonction de compression est donnée par les 160 bits obtenus à partir de l'état final $\langle A^{(80)}, B^{(80)}, C^{(80)}, D^{(80)}, E^{(80)} \rangle$.

Chapitre 3

Les fonctions de hachage

3.1 SHA

3.1.1 Première version SHA-0

De la même manière que le DES pour le chiffrement symétrique, l'administration américaine a proposé un standard de fonction de hachage. Ce standard appelé SHA (*Secure Hash Algorithm*) a été proclamé en 1993 [7]. Nous allons décrire plus complètement ce standard, car il est toujours d'actualité et il est plus simple que le DES. Ceci permettra de mieux appréhender le niveau de détail atteint.

Description de la fonction de hachage

La fonction de hachage SHA considère les messages comme des blocs de 512 bits. Elle fournit un haché de 160 bits et utilise une fonction de compression :

$$\begin{aligned} H : (GF(2)^{32})^5 \times GF(2)^{512} &\rightarrow (GF(2)^{32})^5 \\ (AA, BB, CC, DD, EE, m) &\mapsto (AA', BB', CC', DD', EE') \\ &= H(AA, BB, CC, DD, EE, m). \end{aligned}$$

Pour hacher un message de longueur quelconque on le considère comme une chaîne de caractères codés sur des octets (8 bits) et on effectue la séquence suivante :

1. Ajouter à la fin du message à hacher un octet valant 1, suivi d'un nombre N de 0 et d'un entier codé sur 64 bits représentant la longueur initiale en octets du message. Le nombre N est le nombre minimal d'octets permettant à l'issue du processus d'avoir un message (y compris la taille rajoutée à la fin) codé sur un nombre entier n de blocs de 512 bits (m_1, \dots, m_n) .
2. Initialiser 5 registres de 32 bits notés A , B , C , D et E avec les constantes :
 - $A = 0x67452301$
 - $B = 0xEFCDAB89$
 - $C = 0x98BADCFE$
 - $D = 0x10325476$

L'état final donne la permutation K à utiliser :

$$K = S^{(N)}.$$

Il n'existe pas de cryptanalyse publiée de RC4. En outre, il n'est pas prouvé que le processus de génération soit parfait, c'est-à-dire qu'on doit supposer que le sous-ensemble de l'ensemble des permutations générés par la mise à la clé est indistinguable d'un sous-ensemble aléatoire. À part cela, l'algorithme RC4 semble sûr.

1. Incrémentation du registre i :

$$i^{(\tau+1)} = i^{(\tau)} + 1.$$

2. Ajout de S_i à j :

$$j^{(\tau+1)} = j^{(\tau)} + S_{i^{(\tau+1)}}^{(\tau)}$$

3. Permutation de S_i et S_j :

$$\begin{aligned} S_{i^{(\tau+1)}}^{(\tau+1)} &= S_{j^{(\tau+1)}}^{(\tau)}; \\ S_{j^{(\tau+1)}}^{(\tau+1)} &= S_{i^{(\tau+1)}}^{(\tau)}. \end{aligned}$$

4. Calcul de la somme t de S_i et S_j :

$$t^{(\tau+1)} = S_{i^{(\tau+1)}}^{(\tau+1)} + S_{j^{(\tau+1)}}^{(\tau+1)}.$$

5. Sortie de l'aléa S_t :

$$R^{(\tau+1)} = S_{t^{(\tau+1)}}^{(\tau+1)}.$$

Ce fonctionnement est du type mémoire chiffrante. En effet, la table S est une mémoire dont l'état évolue lentement et qui est utilisée pour en extraire de l'aléa.

2.3.4 Mise à la clé

La mise à la clé de RC4 est une opération à ne pas négliger. En effet, il y a $2^k! = 256! \sim 2^{1684}$ permutations possibles pour la clé interne K . Ceci conduirait à des clés de taille beaucoup trop importantes pour une utilisation pratique, surtout si on utilise un codage par table qui occuperait $256 \times 8 = 2048$ bits.

Pour utiliser une clé plus petite χ de taille quelconque, la méthode proposée est de répéter cette clé dans une table T de N éléments de \mathbb{Z}_k . L'état interne de RC4 est alors initialisé avec l'état suivant :

$$S^{(0)} = Id, \quad i^{(0)} = N - 1, \quad j^{(0)} = 0,$$

où Id est la permutation identité $Id_i = i$ pour tout $0 \leq i \leq N - 1$. Le processus suivant (une altération du RC4) est alors effectué N fois :

1. Incrémentation du registre i :

$$i^{(\tau'+1)} = i^{(\tau')} + 1.$$

2. Ajout de S_i et T_i au registre j :

$$j^{(\tau'+1)} = j^{(\tau')} + S_{i^{(\tau'+1)}}^{(\tau')} + T_{i^{(\tau'+1)}}.$$

3. Permutation de S_i et S_j :

$$\begin{aligned} S_{i^{(\tau'+1)}}^{(\tau'+1)} &= S_{j^{(\tau'+1)}}^{(\tau')}; \\ S_{j^{(\tau'+1)}}^{(\tau'+1)} &= S_{i^{(\tau'+1)}}^{(\tau')}. \end{aligned}$$

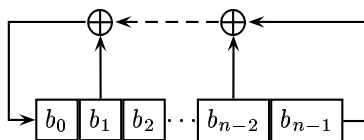


Figure 2.4: Description générale d'un registre à décalage à rétroaction

Le registre à décalage est typiquement destiné à une réalisation sous forme de composant électronique. Une implantation logicielle est en effet souvent très lente.

En outre, l'utilisation de cet outil est difficile à maîtriser, car paradoxalement, un registre à décalage implique une structure mathématique sous-jacente forte, ce qui donne des outils au cryptanalyste. Il est donc généralement nécessaire de combiner plusieurs registres à décalage à l'aide de fonctions bien choisies. Plus encore que dans le cas des systèmes de chiffrement par bloc, l'improvisation n'est pas possible et l'adaptation ou la modification d'un algorithme existant très dangereuse. C'est peut-être pour cette raison que les conceptions de ces algorithmes sont pour la plupart restées secrètes.

2.3.3 RC4

Parmi les rares algorithmes publiés³, l'algorithme de chiffrement RC4 est l'un des plus sérieux, et il est pourtant beaucoup plus simple que DES. Il comporte d'une part une mise à la clé, puis l'algorithme de chiffrement proprement dit. Nous commençons par décrire ce dernier, car la procédure de mise à la clé est une variante de l'algorithme de chiffrement.

Processus de chiffrement

L'algorithme RC4 utilise un paramètre de sécurité $k = 8$. Les opérations s'effectuent dans l'ensemble $\mathbb{Z}_k = \mathbb{Z}/2^k\mathbb{Z}$.

La clé secrète K est une permutation aléatoire des éléments de \mathbb{Z}_k . Il est nécessaire d'introduire quelques notations :

- une permutation Π sur N éléments est représentée par sa table Π_0, \dots, Π_{N-1} ;
- la variable de temps τ est incrémentée à chaque itération ;
- l'état interne de RC4 à l'itération τ , est une permutation $S^{(\tau)}$ des éléments de \mathbb{Z}_k , ainsi que deux registres de \mathbb{Z}_k , notés $i^{(\tau)}$ et $j^{(\tau)}$;
- enfin, la sortie de RC4 à l'itération τ est notée $R^{(\tau)}$.

L'état interne initial de RC4 est

$$S^{(0)} = K, i^{(0)} = 0, j^{(0)} = 0.$$

À chaque itération voici ce qui est effectué :

³Cette publication a d'ailleurs été faite à l'insu des propriétaires de l'algorithme, la société RSA DATA SECURITY.

un chiffrement, déchiffrement, chiffrement². On a donc

$$c = C_B(D_B(C_B(m, K_1), K_2), K_3).$$

Cette méthode est générale pour de nombreux algorithmes, mais ne renforce pas forcément l'algorithme! Dans le cas du DES, il est prouvé que cette méthode apporte une sécurité supplémentaire, même si la sécurité obtenue ne correspond pas au triplement de la longueur de clé.

2.2.2 AES

Le standard DES ne sera pas reconduit en 1998. En effet, les différentes percées évoquées ci-dessus ont montré la nécessité de préparer un nouveau standard, appelé (pour l'instant) *Advanced Encryption Standard*. L'AES est actuellement en cours de choix parmi 15 candidats, présentés en août 1998 lors d'une conférence internationale en Californie. L'histoire dira si l'AES aura une aussi longue durée de vie que le DES, mais ne l'enterrons pas trop vite, car il sera encore largement utilisé, en particulier sous sa forme encore sûre, Triple-DES.

2.3 Systèmes de chiffrement par flot

2.3.1 Générateur de pseudo-aléa

Définir un système de chiffrement par flot en mode OFB revient à fabriquer des suites de bits pseudo-aléatoires. Le terme "pseudo" indique que ces suites sont reproductibles (ce qui est indispensable au déchiffrement), mais le but est bien entendu de conserver les propriétés statistiques d'une suite réellement aléatoire. Pour des applications cryptographiques, une autre caractéristique souhaitable est qu'ils soient non prédictibles, c'est-à-dire que l'observation d'une séquence de bits de sortie ne permette pas de déterminer la suite de la séquence.

Les générateurs aléatoires disponibles de façon native dans la plupart des ordinateurs sont basés sur le principe du générateur linéaire congruentiel. Ils génèrent à chaque itération un nombre aléatoire compris entre 0 et $m - 1$, où m est le module du générateur. La séquence est définie par récurrence à partir d'une semence initiale X_0 par

$$X_n = (aX_{n-1} + b) \bmod m.$$

Les constantes a et b ne doivent pas être choisies au hasard si l'on souhaite obtenir de bonnes propriétés statistiques. En outre, il est prouvé [4] que les générateurs congruentiels ne sont pas sûrs, même si les paramètres utilisés sont tenus secrets. Les utiliser pour du chiffrement ou de la génération d'aléa cryptographique, n'est donc pas possible.

2.3.2 Registres à décalage

Le registre à décalage, dont l'abréviation anglaise LFSR (*Linear Feedback Shift Register*) est souvent utilisée, est un outil beaucoup plus utile à la cryptographie. Sa description générale est donnée par la figure 2.4.

²Le fait d'alterner chiffrement et déchiffrement permet d'assurer la compatibilité entre DES et Triple-DES, puisque l'utilisation de la triple clé K_s, K_s, K_s dans un triple-DES est équivalent à utiliser la clé K_s dans un DES.

Un bit d'information est une application linéaire sur les bits d'un bloc définie par la donnée d'un masque de même taille. Par exemple, le bit d'information correspondant à la somme modulo 2 des bits 0 et 1 est représenté par le masque $3 = 00 \dots 011$. La connaissance de 64 bits d'information linéairement indépendants permet donc de constituer un système linéaire de 64 équations à 64 inconnues donc de retrouver le bloc de 64 bits.

Reprenons notre fonction de chiffrement C_B parfaite qui chiffre par blocs de 64 bits, comme le DES. Alors, par définition, elle est telle que

$$\forall m \forall M, \Pr [C_B(M(A)) = m(A)] = \frac{1}{2}.$$

C'est-à-dire que tous les bits d'informations sont équiprobables et qu'il n'existe aucun lien entre un bit d'information du chiffré et un bit d'information du message clair.

En fait, le DES laisse là encore fuir une faible quantité d'information sur sa clé qui peut être mise en évidence lorsque l'on dispose d'un grand nombre de couples clairs-chiffrés.

Sur le DES à 16 tours, la cryptanalyse linéaire permet de trouver la clé de 56 bits en utilisant 2^{43} couples de textes clairs/chiffrés **connus**.

De la difficulté de concevoir une fonction cryptographique

Les critères de conception du DES ayant été entourés de mystère et en tout cas du secret le plus absolu, des candidats au remplacement du DES ont vu le jour, soit pour en améliorer la rapidité (16 tours semblaient superflus), soit pour garantir l'absence de brèches cachées par la NSA, soit pour essayer de faire mieux qu'IBM. L'exemple le plus fameux est celui de FEAL, conçu par deux japonais de NTT, pour être beaucoup plus rapide et plus sûr que DES.

Utilisant initialement 4 tours (au lieu de 16 pour le DES), une clé de 64 bits (au lieu de 56 pour le DES), des blocs de 64 bits et une architecture générale comparable à celle du DES, FEAL permettait en effet de chiffrer à des débits beaucoup plus élevés. Le seul problème est qu'une attaque différentielle à 4 textes choisis permet de déterminer la clé de chiffrement. Mieux, une attaque linéaire à 5 textes connus donne le même résultat !

Les concepteurs définirent alors un FEAL-8 à 8 tours... qui fut lui aussi cassé. La généralisation FEAL-N fut ensuite proposée avec $N=16$ tours... et connut le même sort. La taille de la clé fut portée à 128 bits avec la variante FEAL-NX, mais il est désormais prouvé que casser FEAL-NX est équivalent à casser FEAL-N, bien que la clé ait vu sa taille doublée.

Il est important de noter que FEAL adopte pourtant une architecture générale de type semblable à celle de DES, appelée "schéma de Feistel". Pourtant, les résultats des différentes méthodes de cryptanalyse sur ces deux algorithmes sont bien différentes !

Triple-DES

Une méthode classique pour renforcer le DES consiste à effectuer un triple chiffrement. Plus exactement, on utilise trois clés de 56 bits K_1, K_2, K_3 et on effectue

- primo, de trouver la première cryptanalyse théorique plus efficace que la recherche exhaustive ;
- secundo, de prouver que les concepteurs de DES étaient parfaitement au courant de cette cryptanalyse, et ceci plus de 15 ans avant Biham et Shamir.

Principe de la cryptanalyse différentielle La cryptanalyse différentielle est une méthode statistique. Une fonction cryptographique parfaite fournit théoriquement un chiffré aléatoire, c'est-à-dire qu'une suite de chiffrés n'est pas distinguable d'une suite de blocs aléatoires. En fait, il existe des biais dont on peut tirer parti, à condition de disposer d'un nombre suffisants d'échantillons pour éliminer statistiquement l'aléa. La cryptanalyse différentielle a permis de montrer l'existence dans le DES d'un biais.

L'idée de base est relativement simple. Considérons la fonction de chiffrement C_B parfaite qui chiffre par blocs de 64 bits, comme le DES. Alors, par définition, elle est telle que

$$\forall \delta \forall B, \Pr [C_B(A \oplus B) \oplus C_B(A) = \delta] = 2^{-64}.$$

En effet, la fonction étant parfaite, les deux quantités $C_B(A \oplus B)$ et $C_B(A)$ sont aléatoires, et la valeur de leur somme est donc aléatoire. Comme la somme comporte 64 bits, il y a 2^{64} possibilités et donc chaque valeur a pour probabilité 2^{-64} .

Or que se passe-t-il avec le DES? La fonction n'est pas parfaite et plus précisément on peut montrer qu'il existe une valeur δ plus probable que les autres et qui dépend de la clé. Cette valeur correspond au biais de la fonction DES. Ainsi, en accumulant des chiffrements $C_B(A \oplus B)$, il est possible de dégager statistiquement la valeur δ la plus probable et d'en déduire une partie de la clé. La fin de la cryptanalyse consiste à énumérer le reste de la clé jusqu'à retrouver la bonne.

Les performances de la cryptanalyse différentielle sur le DES à 16 tours permettent de retrouver une clé de 56 bits à l'aide de 2^{47} textes clairs choisis.

Cryptanalyse linéaire

Encore une révolution En 1993 parut une nouvelle cryptanalyse du DES [12] qui sembla au premier abord assez proche de la cryptanalyse différentielle dans sa présentation. En fait, on s'aperçut vite que si les deux méthodes étaient effectivement liées de façon théorique, la résistance d'une fonction cryptographique à l'une des méthodes n'entraînait pas *ipso facto* la résistance à l'autre méthode. Mieux, la résistance à la cryptanalyse linéaire du DES n'apparaît pas comme ayant été un critère de sa conception, contrairement à la cryptanalyse différentielle.

Principe de la cryptanalyse linéaire Tout comme l'attaque différentielle, la cryptanalyse linéaire est une méthode statistique de mise en évidence d'un biais. Mais au lieu de regarder la différence entre deux chiffrements, la cryptanalyse linéaire considère un seul chiffré, et même un seul bit d'information de ce chiffré.

un niveau de sécurité encore acceptable (mais pour combien de temps?) pour un certain nombre d'applications.

Recherche exhaustive La clé DES comportant 56 bits, la recherche exhaustive sur l'espace des clés à pour complexité moyenne 2^{55} . Ce chiffre en lui-même est peu parlant. Supposons qu'il existe un composant à 300 MHz (la fréquence actuelle commune d'un PC) capable d'effectuer un chiffrement DES en une instruction (un "clock" d'horloge). Alors, cet ordinateur sera capable d'effectuer 300000 vérifications de clés à la seconde. Connaissant un couple clair chiffré (par exemple un mot de passe UNIX[®] crypté, puisque le clair est connu et fixé), combien de temps un tel ordinateur mettra-t-il pour trouver la bonne clé (c'est-à-dire le bon mot de passe)? La réponse est plus de 3800 ans. Or, bien que certains microprocesseurs tournent plus vite encore, un chiffrement DES nécessite beaucoup plus d'une instruction machine!

Seulement, voilà, il est toujours possible d'utiliser beaucoup de machines. La preuve en a été donnée récemment puisque *X machines connectées sur Internet ont permis le premier cassage public d'une clé DES, en X jours*. En outre, il est aussi possible de réaliser des composants dédiés au cassage du DES. Une telle machine a récemment été fabriquée par une association américaine à but non lucratif. Elle permet de casser une clé DES en trois jours¹. Le DES est actuellement l'algorithme de chiffrement le plus courant au monde; croire qu'une agence gouvernementale comme la NSA ne s'est pas donnée les moyens de casser une clé DES en des délais beaucoup plus courts relève de la naïveté.

On voit donc que le cassage du DES, s'il n'est pas encore à la portée du premier venu, n'est pas en soi une opération difficile, et que les progrès technologiques en matière de composants électroniques le rendent aujourd'hui peu sûr. Il pourra cependant être encore utilisé pour la protection de communications privées.

Voyons maintenant pourquoi le DES reste un algorithme très intéressant: encore aujourd'hui la meilleure cryptanalyse pratique connue reste cette fameuse recherche exhaustive. C'est là un résultat remarquable que beaucoup de candidats malheureux à la succession de DES n'ont pas atteint! L'algorithme DES reste un algorithme bien conçu, car sa sécurité est peu ou prou celle garantie par la longueur de sa clé.

Cryptanalyse différentielle

Anecdote En 1990, deux chercheurs utilisent pour la première fois sur le DES une nouvelle technique cryptanalytique appelée cryptanalyse différentielle [3]. Eli Biham et Adi Shamir considèrent en effet, non plus un couple clair-chiffré seul, comme dans la recherche exhaustive, mais deux. Plus précisément, ils étudient l'impact qu'à une différence entre deux textes clairs et la différence observée sur les deux textes chiffrés. Or cette révolution va permettre

¹Histoire de se convaincre que DES n'est plus sûr, on pourra consulter les sites suivants:

- <http://www.awti.com>
- <http://www.cryptography.com>
- <http://www.eff.com>

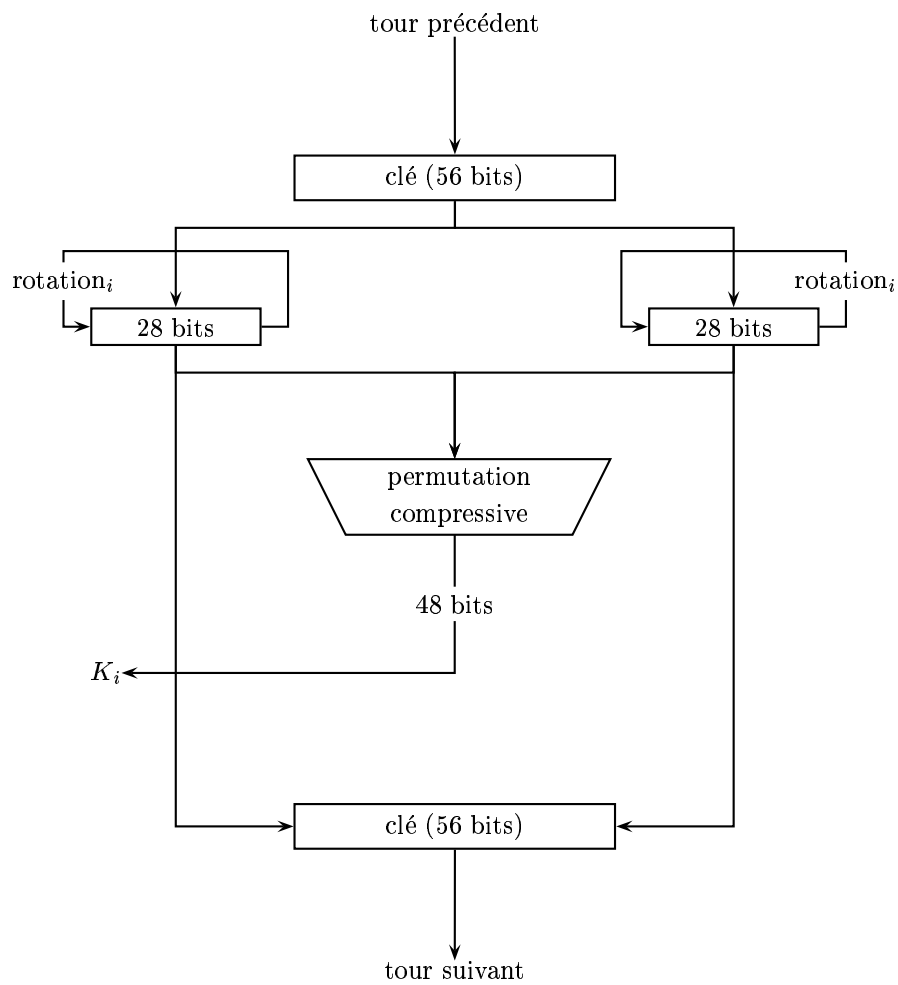


Figure 2.3: Mise à la clé du DES

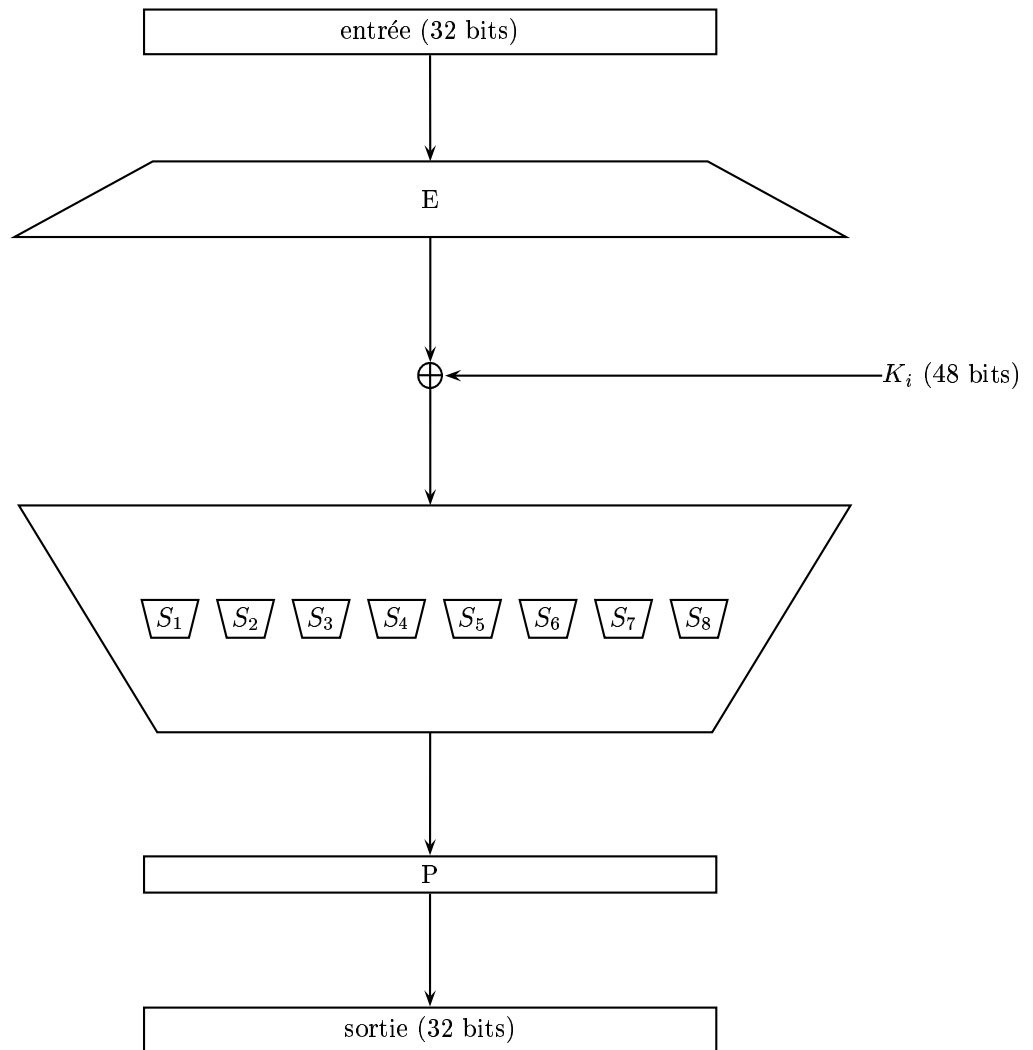


Figure 2.2: Fonction de brouillage f

Permutation IP La permutation IP a longtemps fait couler beaucoup d'encre, car son utilité n'est pas évidente, et elle était soupçonnée d'avoir été placée par la NSA pour affaiblir le DES. En fait, il n'en est rien, et il semble bien que sa seule utilité soit, en séparant les bits consécutifs, de faciliter l'entrée d'un bloc de 64 bits dans un composant électronique implantant le DES.

Description des tours du DES

Description générale La fonction de brouillage f permet de brouiller des blocs de 32 bits. Elle utilise une sous-clé de 48 bits. Le bloc de 32 bits est d'abord étendu à 48 bits par une permutation expansive E . La sous-clé est ensuite combinée au résultat. La compression à 32 bits s'effectue ensuite par l'intermédiaire de 8 boîtes "S". Ces boîtes sont tabulées dans le standard DES. Une dernière permutation est effectuée sur les 32 bits.

Génération des sous-clés La mise à la clé du DES est décrite figure 2.3. Elle utilise des rotations de 1 ou 2 bits à chaque tour. Ces rotations sont définies de telle manière qu'au bout de 16 tours, la clé placée dans le registre de 56 bits soit identique à la clé de départ.

Déchiffrement du DES

Le déchiffrement du DES est très facile du fait de la conception de l'algorithme : il suffit d'utiliser le même algorithme en générant les sous-clés de manière inversée. C'est là l'un des atouts du DES, puisque le même composant peut servir à la fois au chiffrement et au déchiffrement.

Modes opératoires

Le standard officiel [2] prévoyait quatre modes opératoires possibles : ECB, CBC, OFB et CFB. D'autres ont pu être employés dans des applications particulières mais seuls ces quatre modes sont conformes au standard. Il faut noter que le mode ECB, bien que notoirement plus faible que les autres, est souvent utilisé pour sa simplicité de mise en œuvre, alors que le mode CBC, à peine plus complexe, devrait être préféré.

Enfin notons que l'une des applications du DES a été et reste encore la protection des mots de passe UNIX[®]. Le mode utilisé pour cela est un peu particulier, puisque le mot de passe est utilisé comme clé (56 bits correspondent à 8 caractères codés sur 7 bits) pour chiffrer une chaîne constante de 64 bits (généralement la chaîne nulle). Le résultat est stocké dans le fichier de mots de passe. Ce mode de fonctionnement s'apparente à l'utilisation du DES comme fonction de hachage.

Cryptanalyses

Comme toute fonction cryptographique, la sécurité du DES se mesure à l'aune de ses cryptanalyses. L'étude du DES est encore aujourd'hui très utile pour prendre conscience des faiblesses et des forces d'une fonction cryptographique bien conçue. En effet, le DES est actuellement une fonction vieillissante qui ne doit plus être considérée comme sûre. Nous allons voir qu'elle présente cependant

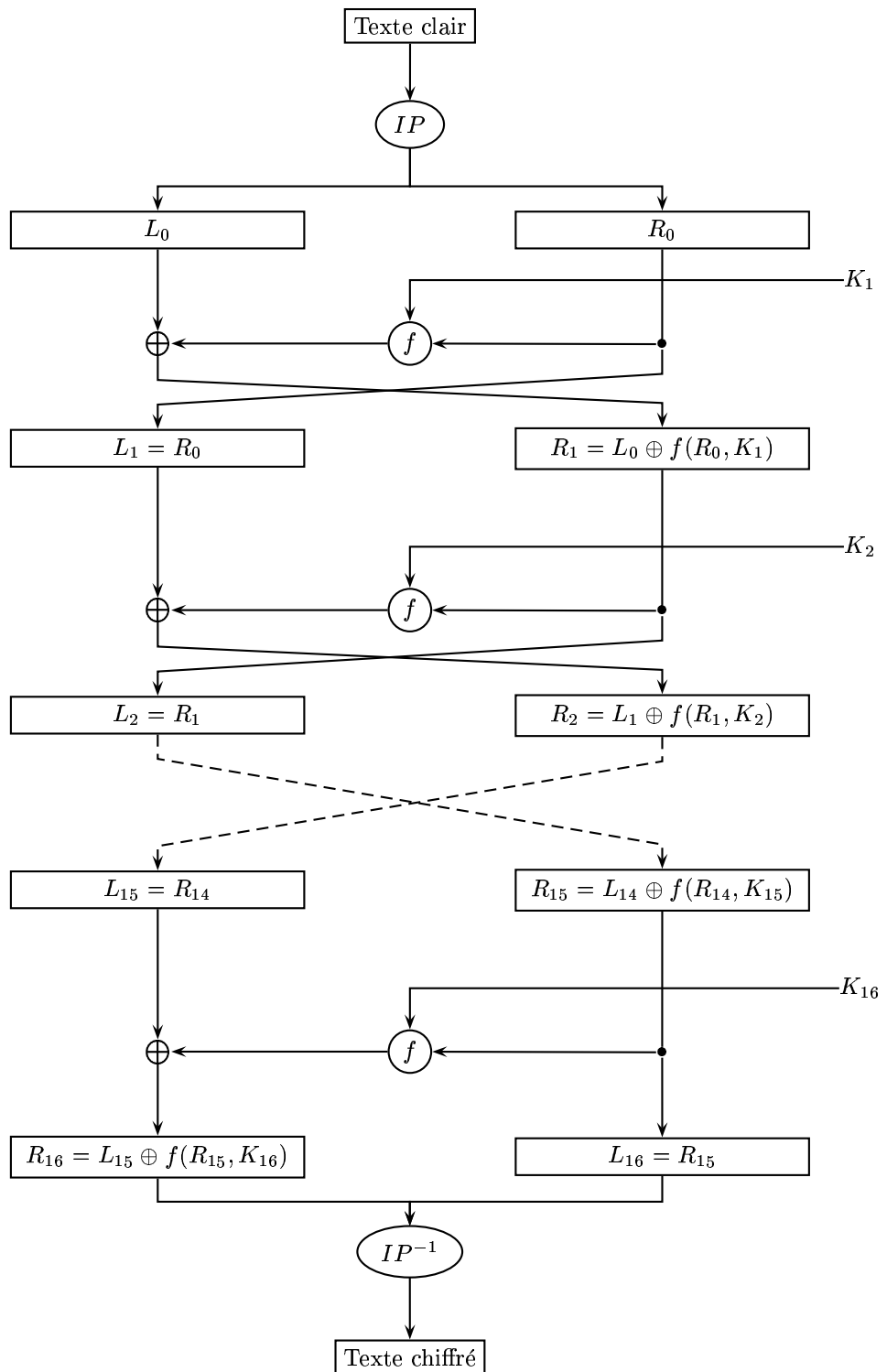


Figure 2.1: Description générale du DES

clé. Les systèmes de chiffrement par flot sont donc plutôt adaptés à des communications en continu, pour lesquelles une nouvelle clé sera utilisée à chaque transaction. Les communications GSM sont ainsi chiffrées entre le poste et la borne par un algorithme de chiffrement par flot appelé A5.

On notera aussi que le mode OFB peut être considéré comme un cas particulier du chiffrement par flot. Il est possible de définir des modes de chiffrement par flot de type CFB, mais nous ne les évoquerons pas ici.

2.1.3 Choix d'un système de chiffrement symétrique

Le choix de tel ou tel mode, ou de tel ou tel système dépend surtout des contraintes de l'application à sécuriser. Chaque mode présente des avantages et des inconvénients en terme de sécurité, mais aussi et surtout en terme de synchronisation et de tolérance aux erreurs. Selon les cas on privilégiera donc tel ou tel mode, ou bien même on utilisera des variantes de ces modes.

2.2 Systèmes de chiffrement par bloc

2.2.1 DES

Le "*Digital Encryption Standard*" est certainement l'algorithme cryptographique le plus connu au monde. En effet, il fut établi en 1976 comme standard pour les communications gouvernementales américaines non classifiées. Repris comme norme ANSI ("*American National Standard Institute*"), puis dans un grand nombre de standards, en particulier financiers, il est resté incontournable pendant de nombreuses années, car il a résisté très longtemps aux attaques de la communauté scientifique.

Son adoption donna lieu à de multiples critiques, en particulier parce que la NSA ("*National Security Agency*"), l'agence gouvernementale américaine souveraine en matière de cryptographie, avait imposé la réduction de la taille des clés de l'algorithme à 56 bits.

Le standard initial prévoyait une révision tous les cinq ans. Dès 1987, la NSA émit des réserves quant à la reconduction du standard. Toutefois, faute de remplaçant crédible, le standard DES fut reconduit en 1987 et même en 1993.

Description

Description générale Le DES utilise des blocs de 64 bits et une clé de 56 bits. Sa description globale est donnée figure 2.1. Le bloc de clair de 64 bits est d'abord permuté par la transformation IP puis modifié par une succession de 16 transformations identiques, appelées communément "tours", et faisant chacune intervenir une sous-clé de 48 bits déduite de la clé K_s . Le bloc obtenu est alors permuté par IP^{-1} pour donner le bloc de chiffré de 64 bits.

La fonction de brouillage f d'un tour du DES est décrite plus précisément figure 2.2. Toutes les "boîtes" utilisées (IP , P , S_x , E) sont fixes et précisément décrites dans le standard. Nous n'incluons pas dans ce cours leurs descriptions précises (qui se présentent sous la forme rébarbative de tableaux de nombres) mais le lecteur intéressé pourra se reporter utilement, soit au standard lui-même [2], soit à [1].

ECB Le premier mode, appelé “*Electronic CodeBook*”, consiste tout simplement à considérer chaque message comme une suite de blocs de taille i et à appliquer pour chaque bloc la fonction C_B en utilisant la clé secrète K_s .

Il est clair que dans un tel mode, un attaquant peut retirer, répéter ou échanger des blocs sans connaître la clé secrète.

CBC Le mode suivant, appelé “*Cipher Block Chaining*”, consiste à chaîner le chiffrement des blocs consécutifs du message $m = (m_1, m_2, \dots, m_t)$. Il suppose l'égalité des tailles des blocs chiffrés et clairs $j = i$. Le message chiffré possède donc le même nombre de blocs que m , soit $c = (c_1, c_2, \dots, c_t)$. Le bloc chiffré c_k est chiffré en utilisant la fonction de chiffrement C_B :

$$c_k = C_B(m_k \oplus c_{k-1}, K_s).$$

Le premier bloc quant à lui, est chiffré à l'aide d'un bloc particulier appelé vecteur d'initialisation IV :

$$c_1 = C_B(m_1 \oplus IV, K_s).$$

Ce vecteur d'initialisation n'est pas secret. Il doit cependant être communiqué au destinataire. Il permet de rendre chaque message unique, et ne doit donc pas être réutilisé. Deux messages identiques pourront donc être chiffrés de façons différentes par l'utilisation de vecteurs d'initialisation différents.

CFB Le chiffrement à rétroaction, appelé “*Cipher FeedBack*”, nécessite comme le mode CBC un vecteur d'initialisation IV . Le chiffrement consiste à effectuer :

$$\begin{aligned} c_1 &= m_1 \oplus C_B(IV, K_s) \\ c_k &= m_k \oplus C_B(c_{k-1}, K_s) \end{aligned}$$

Le vecteur IV doit être unique pour chaque message, car sinon $c_1 \oplus c'_1 = m_1 \oplus m'_1$, ce qui s'avère très gênant. En effet, par analyse statistique sur une collection de message, il est alors possible de déterminer m_1 .

OFB Un autre chiffrement à rétroaction est appelé “*Output FeedBack*”. Comme les deux précédents modes, il nécessite un vecteur IV , mais il ne s'agit plus ici de rendre le chiffrement du k ème bloc dépendant des précédents bloc du message :

$$c_k = m_k \oplus C_B^k(IV, K_s).$$

Chiffrement par flot

On appelle algorithme de chiffrement par flot un générateur de blocs pseudo-aléatoires dépendant de la clé K_s . Un tel générateur fournit pour chaque clé K_s une séquence infinie de blocs a_1, a_2, a_3, \dots . Le chiffré c est alors obtenu en additionnant cette séquence au message :

$$c_i = m_i \oplus a_i.$$

Bien entendu, la connaissance d'un couple message clair, message chiffré, permet de décrypter tout message de longueur inférieure qui serait chiffré avec la même

Chapitre 2

Les systèmes de chiffrement symétriques

2.1 Fondements théoriques

2.1.1 Principe du chiffrement symétrique

Traditionnellement, il est d'usage d'introduire en cryptographie les deux personnages Alice et Bob. Supposons donc qu'Alice souhaite envoyer un message m à Bob, de telle façon que Bob soit le seul à pouvoir prendre connaissance du message. Pour cela, Alice peut utiliser un système de chiffrement C symétrique, et utiliser une clé de chiffrement secrète K_s . Le système de chiffrement symétrique permet à Alice de fabriquer un message chiffré $c = C(m, K_s)$. C'est ce message c qui est transmis à Bob. Au préalable, Bob a reçu par un moyen sûr la clé secrète K_s et peut donc utiliser le mécanisme de déchiffrement C^{-1} pour déchiffrer le message et obtenir $m = C^{-1}(c, K_s)$.

Un tel système de chiffrement est dit symétrique car Alice et Bob doivent tous les deux posséder la même clé K_s pour communiquer. La sécurité du système repose donc sur le secret de la clé, qui ne doit pas être divulguée.

2.1.2 Types de chiffrement symétrique

Chiffrement par bloc

On appelle algorithme de chiffrement par bloc, une fonction

$$\begin{aligned} C_B : GF(2)^i \times GF(2)^s &\rightarrow GF(2)^j \\ (m, K_s) &\mapsto c = C_B(m, K_s). \end{aligned}$$

La taille j du bloc de sortie est obligatoirement supérieure, mais généralement égale à la taille i du bloc d'entrée. La taille s de la clé varie selon les algorithmes.

Modes de fonctionnement

Un algorithme de chiffrement par bloc est utilisable de différentes façons, appelées modes. Il existe un très grand nombre de modes de fonctionnement. Nous en décrivons succinctement quatre, parmi les plus importants et les plus courants.

Comme toujours, il existe une attaque de référence qui consiste à répondre au hasard aux questions posées en espérant obtenir les bonnes réponses. La complexité de cette attaque dépend ici beaucoup du protocole.

1.3.4 Signature

Comme dans le cas du chiffrement et de l'authentification, on distingue les attaques totales, universelles ou existentielles, et celles à *messages connus* ou à *messages choisis*. L'attaque de référence consiste à choisir au hasard la signature du message ou la clé de signature.

1.3 Les objectifs du cryptanalyste

1.3.1 Chiffrement

En matière de chiffrement les objectifs du cryptanalyste ne se résument pas simplement à l'obtention du message clair à partir du message chiffré. Il existe en effet toute une gamme d'attaques qui ne sont pas toutes aussi performantes les unes que les autres.

En effet, si l'attaque permet de découvrir la clé secrète de déchiffrement, alors on parlera de *cassage total* ou de *cryptanalyse totale*. Si l'attaque permet seulement de retrouver le message clair, on parlera de *cassage universel*. Enfin, si l'attaque ne permet de déchiffrer que certains messages, on parlera de *cassage existentiel*.

On distingue aussi les attaques selon la quantité d'information nécessaire au cryptanalyste pour la mener à bien. Une attaque à *chiffré seul* est l'attaque qui demande le moins d'information. Une attaque à *clair connu* suppose que le cryptanalyste dispose par avance d'un certain nombre de couples de messages clairs/chiffrés, avant de pouvoir casser le système. Une attaque à *clair choisi* suppose en outre que l'attaquant a les moyens de demander le chiffrement de messages particuliers.

Pour évaluer l'efficacité d'une attaque, on cherche souvent à comparer celle-ci à l'attaque qui consiste à énumérer toutes les clés, car cette *attaque exhaustive* est toujours possible. Si la taille de la clé est de k bits, alors cette attaque a une complexité moyenne de 2^{k-1} .

1.3.2 Intégrité

Casser une fonction de hachage revient à la recherche de *collisions*. En effet, une fonction de hachage envoie un ensemble de taille infinie (l'ensemble de tous les messages) dans un ensemble de taille finie, puisque le haché du message a une taille fixe. Par définition, des collisions existent donc.

Là encore, on distingue différents types d'attaque, selon que l'attaque permet de partiellement choisir ses messages ou bien exhibe seulement un couple de messages inintelligibles ayant le même haché.

Dans le cas des fonctions de hachage, il existe aussi une attaque toujours possible basée sur le *paradoxe des anniversaires*. Elle consiste à énumérer des messages jusqu'à l'obtention d'une collision. Si la taille du haché est de k bits, on peut prouver que la collision est alors obtenue avec une complexité moyenne de $\sqrt{2^k}$.

1.3.3 Authentification

Un protocole d'authentification met en relation un prouveur et un vérifieur. Il consiste d'une façon générale pour le prouveur à répondre à un certain nombre de questions du vérifieur. Les réponses supposent la connaissance d'un secret. L'objectif du cryptanalyste peut donc être soit de retrouver le secret nécessaire à l'authentification, soit de procéder à une authentification réussie sans connaissance de ce secret. Comme dans le cas du chiffrement, on distingue les attaques totales, universelles ou existentielles, et celles à *questions connues* ou à *questions choisies*.

américains. Il faut noter que si les améliorations techniques en matière de calcul furent importantes à cette occasion, les plus grandes avancées furent d'origine théorique et furent le fait des dizaines de mathématiciens de très haut niveau qui se penchèrent sur ces problèmes. Les résultats obtenus alors sont à la base de la théorie de l'information développée par Shannon.

1.2 Fonctionnalités de la cryptographie

1.2.1 Chiffrement

L'application la plus ancienne de la cryptographie concerne la confidentialité. Elle consiste, au moyen d'une information appelée *clé*, à réaliser le *chiffrement* de l'information confidentielle. Le *déchiffrement* réalise l'opération inverse, connaissant la *clé secrète*. Le *décryptage* constitue l'attaque cryptanalytique du chiffrement, c'est-à-dire le recouvrement du message, sans connaissance de la clé secrète.

1.2.2 Intégrité

Avec l'émergence des technologies informatiques, le problème de l'intégrité d'une donnée est apparu de façon nouvelle. En effet, autant la falsification d'un document papier laisse des traces de l'agression physique, autant la modification d'une donnée informatique mémorisée peut se faire de manière absolument invisible.

La cryptographie a donc développé des *fonctions de hachage* qui permettent d'obtenir à partir d'un message un *haché* (ou *condensat*), de taille réduite par rapport au message initial. La particularité de ces fonctions est qu'il est très difficile de modifier le message sans modifier le haché.

1.2.3 Authentification

L'*authentification* consiste à prouver son identité ou plus généralement sa qualité. En d'autres termes, elle vise à empêcher autrui de se faire passer pour qui il n'est pas. Un exemple classique d'authentification est celui des cartes bancaires : dans un distributeur de billets, il y a avant toute chose, un *protocole d'authentification* qui vise à s'assurer qu'il s'agit bien d'une carte bancaire. L'authentification en elle même est obtenue par la preuve de possession d'un secret, en l'occurrence le code du porteur de la carte.

1.2.4 Signature

La *signature* consiste à associer de manière sûre une *identité* à un message et à empêcher toute *répudiation* ultérieure. Une signature suppose une action volontaire de la part de la personne qui signe. Le besoin de signature numérique est lui aussi apparu récemment.

Chapitre 1

Introduction à la cryptologie

1.1 Historique (très) succinct

Le terme de cryptologie (du grec *κρυπτός*, caché) recouvre un domaine qui ne se limite plus à la simple protection d'une communication au moyen d'une écriture conventionnelle secrète¹. Les objectifs de la cryptologie d'une part, les moyens qu'elle utilise d'autre part, ont beaucoup évolué depuis une cinquantaine d'année. Le développement des transmissions électroniques, qui va s'accéléralant, fait d'autre part apparaître un grand nombre de besoins cryptologiques.

Une particularité de la cryptologie est qu'elle présente un double aspect.

- la *cryptographie* consiste à construire des outils pour assurer un certain nombre de fonctionnalités ;
- la *cryptanalyse* s'attaque aux outils précédents pour en trouver les faiblesses.

Ces deux approches sont indissociables.

Les premiers témoignages d'utilisation de la cryptographie remontent aux Phéniciens. Ils utilisaient un bâton de diamètre fixé sur lequel était enroulée une lanière de cuir. L'écriture et la lecture se faisaient dans le sens de la longueur du bâton. Une fois déroulée, la lanière ne présentait plus qu'un assemblage de caractères permutés du message initial et il fallait un bâton (qui jouait le rôle de clé) de même diamètre pour pouvoir relire le message.

Le chiffre de César est une autre anecdote connue. Il consistait à transposer chaque caractère par un autre.

Au dix-septième siècle, le cabinet noir de Louis XIV était réputé pour être capable de décrypter la plupart des systèmes de chiffrement de l'époque.

Durant la première guerre mondiale, et surtout durant la deuxième, la bataille du chiffre fut l'un des enjeux majeurs du conflit. Ainsi, pour décrypter les messages de la machine cryptographique allemande Enigma, les anglo-américains développèrent la première machine de calcul automatique, ancêtre de tous les ordinateurs modernes. Le chiffre japonais fut de même cassé par les services

¹Définition du dictionnaire Larousse !

4	Authentification et signature	29
4.1	Problématique de l'authentification	29
4.2	Le concept de cryptographie asymétrique	29
4.3	Algorithmes de signature	30
4.3.1	RSA	30
4.3.2	ElGamal	30
4.3.3	DSS	31
4.4	Système de certification	32
4.5	Système de chiffrement à clé publique	33
4.5.1	Algorithme de chiffrement RSA	33
4.5.2	Protocole d'échange de clé de Diffie-Hellman	33
5	Sécurité des algorithmes cryptographiques	35
5.1	Les algorithmes empiriques	36
5.2	Les algorithmes à sécurité prouvée	36
5.2.1	Problèmes réputés difficile	36
5.2.2	Problèmes NP-complets	37
5.2.3	Cryptographie et ordinateur quantique	38
5.3	Sécurité des implantations	38

Table des matières

Avertissement	3
Table des matières	5
1 Introduction à la cryptologie	7
1.1 Historique (très) succinct	7
1.2 Fonctionnalités de la cryptographie	8
1.2.1 Chiffrement	8
1.2.2 Intégrité	8
1.2.3 Authentification	8
1.2.4 Signature	8
1.3 Les objectifs du cryptanalyste	9
1.3.1 Chiffrement	9
1.3.2 Intégrité	9
1.3.3 Authentification	9
1.3.4 Signature	10
2 Les systèmes de chiffrement symétriques	11
2.1 Fondements théoriques	11
2.1.1 Principe du chiffrement symétrique	11
2.1.2 Types de chiffrement symétrique	11
2.1.3 Choix d'un système de chiffrement symétrique	13
2.2 Systèmes de chiffrement par bloc	13
2.2.1 DES	13
2.2.2 AES	21
2.3 Systèmes de chiffrement par flot	21
2.3.1 Générateur de pseudo-aléa	21
2.3.2 Registres à décalage	21
2.3.3 RC4	22
2.3.4 Mise à la clé	23
3 Les fonctions de hachage	25
3.1 SHA	25
3.1.1 Première version SHA-0	25
3.1.2 Deuxième version SHA-1	28
3.1.3 Recherches de collisions	28
3.2 MD5	28

Avertissement

Ce cours se veut une simple introduction à la cryptologie. Il présente quelques algorithmes parmi les plus connus de cryptographie, et cherche à démontrer l'intérêt qu'il y a à les utiliser d'une part, à les bien utiliser d'autre part, c'est-à-dire à comprendre leurs limites.

Le lecteur intéressé par la cryptographie et qui souhaiterait consulter un ouvrage *quasi* exhaustif du domaine pourra consulter [1].

Enfin, la cryptologie est une science en perpétuelle évolution et l'Histoire montre à quel point elle a pu évoluer, rendant totalement obsolètes des algorithmes pourtant soigneusement élaborés. Ce qui suit aura donc forcément une durée de vie très brève...

École Supérieure d'Électricité
master "Systèmes d'Information"

Introduction à la cryptographie

par

Florent CHABAUD



Ingénieur de l'Armement
Centre d'Électronique de l'Armement (CELAR)
CASSI/CA
35998 Rennes Armées

École Supérieure d'Électricité
master "Systèmes d'Information"

Introduction à la cryptographie

par

Florent CHABAUD