Introduction à la cryptologie

Sujet :	Document d'introduction à la cryptologie
Date :	5 Novembre 1998
Auteur :	Ghislaine Labouret

Droits d'auteur :	Reproduction autorisée à condition de mentionner l'auteur original et de donner l'URL de ce document.

Table des matières

1. Terminologie
2. Cryptographie
   - 2.1. Confidentialité et algorithmes de chiffrement
     - 2.1.1. Chiffrement symétrique ou à clef secrète
       - Modes de chiffrement par blocs
       - Chiffrement par blocs avec itération
       - Exemples d'algorithmes symétriques
     - 2.1.2. Chiffrement asymétrique ou à clef publique
       - Algorithmes à empilement
       - RSA (Rivest, Shamir, Adleman)
   - 2.2. Générateurs aléatoires et pseudo-aléatoires
   - 2.3. Fonctions de hachage à sens unique, signature numérique et scellement
     - 2.3.1. Fonctions de hachage à sens unique
       - MD5 (Message Digest 5)
       - SHA (Secure Hash Algorithm)
       - RIPE-MD
     - 2.3.2. Intégrité et authentification de l'origine des données
     - 2.3.3. Signature numérique
       - DSS
       - RSA
     - 2.3.4. Scellement
       - Keyed-Hash
       - HMAC
   - 2.4. Protocoles cryptographiques
     - 2.4.1. Protocoles à apport nul de connaissance
     - 2.4.2. La notion de tiers de confiance
     - 2.4.3. Échange de clef
       - Diffie–Hellman
       - Échange de clef et authentification mutuelle
       - Propriétés des protocoles d'échange de clef
       - Hiérarchie des clefs
3. Cryptanalyse

3.1. Attaques des fonctions de chiffrement
3.1.1. Classement des attaques en fonction des données dont dispose le cryptanalyste
3.1.2. Attaques sur les algorithmes symétriques
Attaques au niveau des clefs
Cryptanalyse différentielle
Cryptanalyse linéaire
3.1.3. Attaques sur les algorithmes asymétriques
Attaques au niveau des clefs
Attaque à texte en clair deviné et problème de la faible entropie
Attaque à texte chiffré choisi
Attaque temporelle
3.2. Attaques des générateurs pseudo-aléatoires
3.3. Attaques des fonctions de hachage à sens unique
Attaque des anniversaires
3.4. Attaques des protocoles cryptographiques
3.4.1. Attaque par mascarade
3.4.2. Attaque par rejeu
3.4.3. Attaque par réflexion
3.4.4. Attaque de l'intercepteur

Sigles et acronymes

Glossaire

Bibliographie commentée

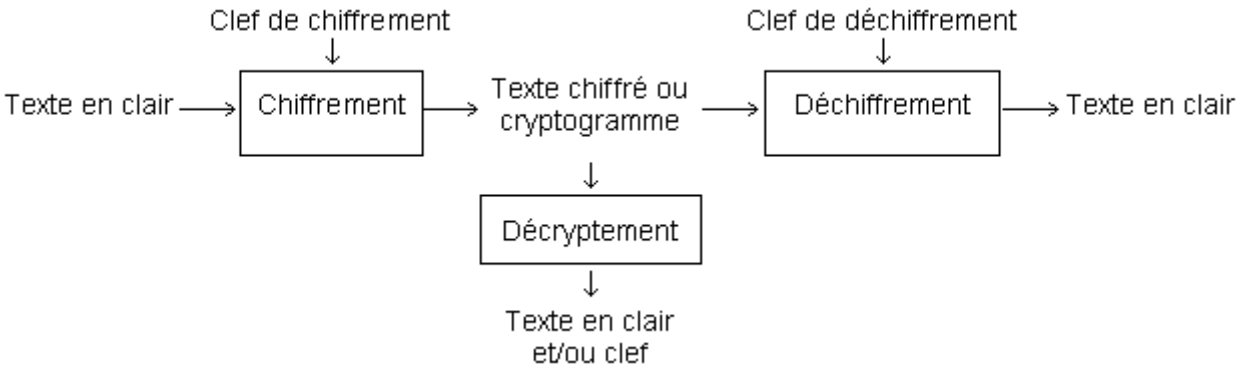
Généralités
Codes d'authentification de messages
Échange de clef

1. Terminologie

La *cryptologie* est une science mathématique qui comporte deux branches : la *cryptographie* et la *cryptanalyse*.

La **cryptographie** traditionnelle est l'**étude des méthodes permettant de transmettre des données de manière confidentielle**. Afin de protéger un message, on lui applique une transformation qui le rend incompréhensible ; c'est ce qu'on appelle le *chiffrement*, qui, à partir d'un *texte en clair*, donne un *texte chiffré* ou *cryptogramme*. Inversement, le *déchiffrement* est l'action qui permet de reconstruire le texte en clair à partir du texte chiffré. Dans la cryptographie moderne, les transformations en question sont des fonctions mathématiques, appelées *algorithmes cryptographiques*, qui dépendent d'un paramètre appelé *clef*.

La **cryptanalyse**, à l'inverse, est l'**étude des procédés cryptographiques dans le but de trouver des faiblesses** et, en particulier, de pouvoir décrypter des textes chiffrés. Le *décryptement* est l'action consistant à retrouver le texte en clair sans connaître la clef de déchiffrement.



- Figure 1. Principes de base de la cryptologie -

2. Cryptographie

Si le but traditionnel de la cryptographie est d'élaborer des méthodes permettant de transmettre des données de manière confidentielle, **la cryptographie moderne s'attaque en fait plus généralement aux problèmes de sécurité des communications**. Le but est d'offrir un certain nombre de services de sécurité comme la *confidentialité*, l'*intégrité*, l'*authentification* des données transmises et l'authentification d'un tiers. Pour cela, on utilise un certain nombre de mécanismes basés sur des *algorithmes cryptographiques*. Nous allons voir dans ce chapitre quelles sont les techniques que la cryptographie fournit pour réaliser ces mécanismes.

2.1. Confidentialité et algorithmes de chiffrement

La confidentialité est historiquement le premier problème posé à la cryptographie. Il se résout par la notion de chiffrement, mentionnée plus haut. Il existe deux grandes familles d'algorithmes cryptographiques à base de clefs : les algorithmes à *clef secrète* ou algorithmes symétriques, et les algorithmes à *clef publique* ou algorithmes asymétriques.

2.1.1. Chiffrement symétrique ou à clef secrète

Dans la cryptographie conventionnelle, les **clefs de chiffrement et de déchiffrement sont identiques** : c'est la clef secrète, qui doit être connue des tiers communiquants et d'eux seuls. Le procédé de chiffrement est dit symétrique.

Les algorithmes symétriques sont de deux types :

- les algorithmes de **chiffrement en continu**, qui agissent sur le texte en clair un bit à la fois ;
- les algorithmes de **chiffrement par blocs**, qui opèrent sur le texte en clair par groupes de bits appelés blocs.

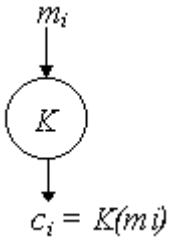
Modes de chiffrement par blocs

Les algorithmes de chiffrement par blocs peuvent être utilisés suivant différents modes, dont les deux principaux sont le mode ECB (*Electronic CodeBook*) et le mode CBC (*Cipher Block Chaining*).

Le mode ECB (Electronic CodeBook)

Le mode du "carnet de codage électronique" (*Electronic CodeBook*) est la méthode la plus évidente pour utiliser un algorithme de chiffrement par blocs : **un bloc du texte en clair se chiffre,**

indépendamment de tout, en un bloc de texte chiffré. L'avantage de ce mode est qu'il permet le chiffrement en parallèle des différents blocs composant un message.



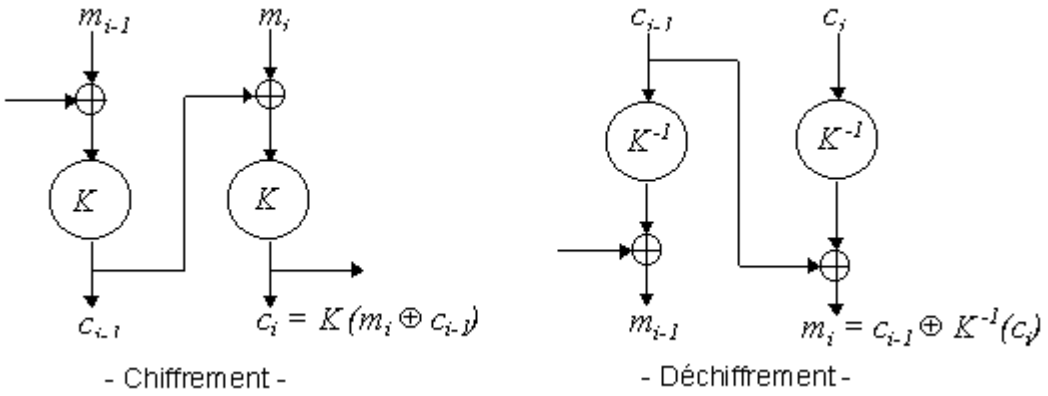
- Figure 2. Le mode ECB -

L'inconvénient de ce mode est qu'**un même bloc de texte en clair sera toujours chiffré en un même bloc de texte chiffré**. Or, dans le chiffrement sur un réseau par exemple, les données à chiffrer ont des structures régulières facilement repérables par un cryptanalyste, qui pourra donc **obtenir beaucoup d'informations**. D'autre part, un attaquant actif pourra facilement **manipuler les messages chiffrés** en retirant, répétant ou interchangeant des blocs. Un autre inconvénient, qui s'applique au chiffrement par blocs en général, est l'**amplification d'erreur** : si un bit du texte chiffré est modifié pendant le transfert, tout le bloc de texte en clair correspondant sera faux.

Le mode CBC (Cipher Block Chaining)

La solution aux problèmes posés par le mode ECB est d'utiliser une technique dite de **chaînage**, dans laquelle **chaque bloc du cryptogramme dépend non seulement du bloc de texte en clair correspondant, mais aussi de tous les blocs précédents**.

En mode de "chiffrement avec chaînage de blocs" (*Cipher Block Chaining*), **chaque bloc de texte en clair est combiné par *ou exclusif* avec le bloc chiffré précédent avant d'être chiffré**. Le premier bloc du texte en clair est, quant à lui, combiné avec un bloc appelé ***vecteur d'initialisation***. L'utilisation d'un vecteur d'initialisation différent pour chaque message permet de s'assurer que deux messages identiques (ou dont les premiers blocs sont identiques) donneront des cryptogrammes totalement différents.



- Figure 3. Le mode CBC -

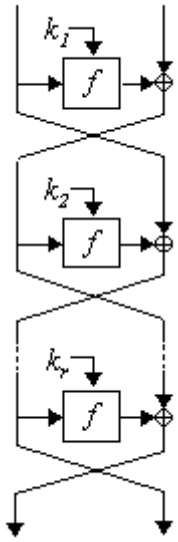
Le gros avantage du mode CBC est donc que la structure du texte en clair est masquée par le chaînage. Un attaquant ne peut plus manipuler le cryptogramme, excepté en retirant des blocs au début ou à la fin. Un inconvénient est qu'il n'est plus possible de paralléliser le chiffrement des différents blocs (le déchiffrement reste parallélisable).

On pourrait craindre que le chaînage de bloc n'entraîne une propagation d'erreur importante. De fait, une erreur d'un bit sur le texte en clair affectera tous les blocs chiffrés suivants. Par contre, si un bit du texte chiffré est modifié au cours du transfert, seul le bloc de texte en clair correspondant et un bit du bloc de texte en clair suivant seront endommagés : le mode CBC est dit **auto-réparateur**.

Chiffrement par blocs avec itération

Un algorithme de chiffrement par blocs avec itération est un algorithme qui chiffre les blocs par un processus comportant plusieurs rondes. **Dans chaque ronde, la même transformation est appliquée au bloc**, en utilisant une sous-clef dérivée de la clef de chiffrement. En général, un nombre de rondes plus élevé garantit une meilleure sécurité, au détriment des performances.

Un cas particulier d'algorithmes de chiffrement par blocs avec itération est la famille des **chiffres de Feistel.** Dans un chiffre de Feistel, un bloc de texte en clair est découpé en deux ; la transformation de ronde est appliquée à une des deux moitiés, et le résultat est combiné avec l'autre moitié par *ou exclusif*. Les deux moitiés sont alors inversées pour l'application de la ronde suivante. Un avantage de ce type d'algorithmes est que chiffrement et déchiffrement sont structurellement identiques.



- Figure 4. Principe du chiffre de Feistel -

Exemples d'algorithmes symétriques

La méthode la plus employée pour concevoir un procédé de chiffrement est de chercher à réaliser une transformation suffisamment compliquée et irrégulière pour que son analyse soit difficile. Cette méthode empirique ne fournit aucune garantie quant à la résistance de l'algorithme résultant.

Des exemples d'algorithmes asymétriques d'utilisation courante aujourd'hui sont le DES (*Data Encryption Standard*), le DES triple à deux ou trois clefs, IDEA (*International Data Encryption Algorithm*), RC5 (*Rivest's Code 5*),...

Masque jetable (one-time pad)

Il existe un **algorithme de chiffrement prouvé inconditionnellement sûr** : le chiffre de Vernam ou masque jetable (*one-time pad* en anglais), inventé en 1917.

Dans ce système, la "clef" a la même taille que le texte à chiffrer et est appelée masque jetable.

"Masque", car cette clef est combinée par *ou exclusif* avec le texte en clair pour obtenir le texte chiffré ; "jetable", car **une clef ne doit servir qu'une seule fois**, sans quoi un cryptanalyste pourrait retrouver le texte en clair. Le point important de ce système est que la **clef est générée de façon totalement aléatoire** ; tout masque est alors équiprobable, si bien qu'un attaquant n'a aucune information pour baser sa cryptanalyse. En effet, si M est le message à chiffrer, C le cryptogramme correspondant et K le masque jetable, nous avons $C = M \oplus K$. Supposons que le cryptanalyste connaisse C ; quel que soit M', il existe K' tel que $C = M' \oplus K'$, c'est $K' = M' \oplus C$: tous les messages en clair sont équiprobables ! Sans information sur K, **le cryptanalyste n'a donc aucun moyen de savoir quel est le bon texte en clair**.

Les principaux inconvénients de ce système sont la taille de la clef et la nécessité de posséder un générateur aléatoire pour sa création. La conséquence en est que ce système est inutilisable dans le cadre du chiffrement d'un flux important de données et reste limité à des applications extrêmes.

DES (Data Encryption Standard)

Le gouvernement américain a adopté, en 1977, la fonction DES (*Data Encryption Standard*) comme algorithme de chiffrement standard officiel.

Le DES est un algorithme de chiffrement par blocs qui agit sur des **blocs de 64 bits**. C'est un chiffre de Feistel à 16 rondes. **La longueur de la clef est de 56 bits**. Généralement, celle-ci est représentée sous la forme d'un nombre de 64 bits, mais un bit par octet est utilisé pour le contrôle de parité. Les sous-clefs utilisées par chaque ronde ont une longueur de 48 bits.

Le **DES triple** est une variante du DES qui consiste à appliquer l'algorithme trois fois à chaque bloc : chiffrement, déchiffrement puis de nouveau chiffrement. Les clefs utilisées pour chaque application du DES peuvent être toutes les trois distinctes, ou bien on peut utiliser seulement deux clefs distinctes : une pour le chiffrement et une pour le déchiffrement. Dans tous les cas, la longueur efficace de la clef du triple DES ne dépasse pas 112 bits.

Un nouvel algorithme, qui vise à remplacer le DES, est en cours de normalisation auprès du NIST (*National Institute of Standards and Technology*). Il sera désigné sous le sigle AES (*Advanced Encryption Standard*).

IDEA (International Data Encryption Algorithm)

Un autre algorithme de chiffrement répandu est IDEA (International Data Encryption Algorithm), qui a vu le jour en 1991. IDEA est un algorithme de chiffrement par blocs avec itération qui utilise une clef de 128 bits et comporte 8 rondes.

2.1.2. Chiffrement asymétrique ou à clef publique

Avec les algorithmes asymétriques, **les clefs de chiffrement et de déchiffrement sont distinctes et ne peuvent se déduire l'une de l'autre**. On peut donc rendre l'une des deux publique tandis que l'autre reste privée. C'est pourquoi on parle de chiffrement à clef publique. Si la clef publique sert au chiffrement, tout le monde peut chiffrer un message, que seul le propriétaire de la *clef privée* pourra déchiffrer. On assure ainsi la confidentialité. Certains algorithmes permettent d'utiliser la clef privée pour chiffrer. Dans ce cas, n'importe qui pourra déchiffrer, mais seul le possesseur de la clef privée peut chiffrer. Cela permet donc la signature de messages.

Le concept de cryptographie à clef publique fut inventé par Whitfield Diffie et Martin Hellman en 1976, dans le but de résoudre le problème de distribution des clefs posé par la cryptographie à clef secrète. De nombreux algorithmes permettant de réaliser un cryptosystème à clef publique ont été proposés depuis. Ils sont le plus souvent **basés sur des problèmes mathématiques difficiles à**

résoudre, donc leur sécurité est conditionnée par ces problèmes, sur lesquels on a maintenant une vaste expertise. Mais, si quelqu'un trouve un jour le moyen de simplifier la résolution d'un de ces problèmes, l'algorithme correspondant s'écroulera.

Nombre des algorithmes proposés pour la cryptographie à clef publique se sont révélés rapidement non sûrs, ou non réalisables sur le plan pratique. Tous les algorithmes actuels présentent l'inconvénient d'être **bien plus lents que les algorithmes à clef secrète** ; de ce fait, **ils sont souvent utilisés non pour chiffrer directement des données, mais pour chiffrer une *clef de session* secrète**. Certains algorithmes asymétriques ne sont adaptés qu'au chiffrement, tandis que d'autres ne permettent que la signature. Seuls trois algorithmes sont utilisables à la fois pour le chiffrement et pour la signature : RSA, ElGamal et Rabin.

Le paragraphe "3.1.3. Attaques sur les algorithmes asymétriques" décrit les attaques possibles contre les algorithmes asymétriques et les précautions à prendre en conséquence. Un point particulièrement important est que l'ensemble des données à chiffrer ne doit pas être de faible entropie.

Algorithmes à empilement

Historiquement, le premier algorithme asymétrique fut proposé par Merkle et Hellman en 1978. Il repose sur le problème d'empilement ou problème du sac à dos (*knapsack problem* en anglais), qui est un problème NP-complet. Cet algorithme fut très vite démontré non sûr, ainsi que la plupart des algorithmes à empilement qui virent le jour par la suite.

RSA (Rivest, Shamir, Adleman)

Inventé par Rivest, Shamir et Adleman en 1978, RSA permet le chiffrement et la signature. Il est aujourd'hui encore très largement utilisé. Cet algorithme repose sur la difficulté de factoriser des grands nombres.

Voici comment se fait la génération des paires de clefs :

1. On commence par choisir deux grands nombres premiers, p et q, et on calcule $n = pq$. n est rendu public ; p et q doivent rester secrets et sont donc détruits une fois les clefs générées.
2. On choisit ensuite aléatoirement une clef publique e telle que e et $(p-1)(q-1)$ soient premiers entre eux.
3. La clef privée d est obtenue grâce à l'algorithme d'Euclide : $ed \equiv 1 \mod (p-1)(q-1)$.

Soit m le message en clair et c le cryptogramme. La fonction de chiffrement est, de façon simplifiée, $c = m^e \mod n$ (si m est plus grand que n, il est séparé en morceaux de valeur inférieure à n et chaque morceau est chiffré séparément suivant cette formule). Du fait de la relation entre e et d, la fonction de déchiffrement correspondante est $m = c^d \mod n$. La signature se fait de manière similaire, en inversant e et d, c'est-à-dire en chiffrant avec une clef privée et en déchiffrant avec la clef publique correspondante : $s = m^d \mod n$ et $m = s^e \mod n$.

Pour un cryptanalyste, retrouver la clef privée à partir de la clef publique nécessite de connaître $(p-1)(q-1) = pq-p-q+1 = n+1-p-q$, donc de connaître p et q. Pour cela, il doit factoriser le grand nombre n. Donc n doit être suffisamment grand pour que cela ne soit pas possible dans un temps raisonnable par rapport au niveau de sécurité requis. Actuellement, la longueur du module n varie généralement de 512 à 2048 bits suivant les utilisations. Compte tenu de l'augmentation des vitesses de calcul des ordinateurs et des avancées mathématiques en matière de factorisation des grands nombres, la longueur minimale des clefs doit augmenter au cours du temps.

Clefs			
Clef publique	$n = pq$, où p et q sont deux grands nombres premiers tenus secrets e telle que e et $(p-1)(q-1)$ soient premiers entre eux		
Clef privée	$d \equiv e^{-1} \bmod (p-1)(q-1)$		
Algorithmes			
Chiffrement	$c = m^e \bmod n$	Déchiffrement	$m = c^d \bmod n$
Signature	$s = m^d \bmod n$	Vérification	$m = s^e \bmod n$

- Figure 5. RSA -

2.2. Générateurs aléatoires et pseudo-aléatoires

La cryptographie a souvent recours à des nombres aléatoires. Ainsi, lorsqu'une personne génère une clef secrète ou privée, elle doit faire intervenir le hasard de façon à empêcher un adversaire de deviner la clef. De même, certains protocoles cryptographiques nécessitent, pour éviter la rejouabilité par exemple, l'utilisation d'aléas imprévisibles par les opposants.

Malheureusement, il est impossible de produire des suites aléatoires à l'aide uniquement d'un ordinateur : le générateur sera toujours périodique, donc prévisible. On a donc recours à des générateurs dits pseudo-aléatoires et cryptographiquement sûrs. Un tel générateur doit présenter les caractéristiques suivantes :

1. La période de la suite doit être suffisamment grande pour que les sous-suites finies utilisées avec l'algorithme ou le protocole cryptographique ne soient pas périodiques.
2. Ces sous-suites doivent, sur le plan statistique, sembler aléatoires.
3. Le générateur doit être imprévisible, au sens où il doit être impossible de prédire le prochain aléa à partir des aléas précédents.

La plupart des générateurs pseudo-aléatoires sont construits en utilisant des registres à décalage (*shift registers* en anglais) et, en particulier, les registres à décalage à rétroaction linéaire (*Linear Feedback Shift Registers, LFSR)*. Ces derniers présentent l'inconvénient de générer des suites linéaires, si bien que des grands nombres générés à partir de sous-suites sont fortement corrélés. C'est pourquoi les générateurs pseudo-aléatoires sont généralement construits en combinant, à l'aide d'une fonction non linéaire, plusieurs registres à décalage de tailles différentes. Ce type de générateur est très utilisé par les algorithmes de chiffrement en continu.

Si l'on veut vraiment générer des suites aléatoires, au sens où ces suites sont de plus non reproductibles, on a généralement recours à des éléments extérieurs comme les déplacements de la souris, la vitesse de frappe, l'entrée d'un micro enregistrant le bruit atmosphérique,...

2.3. Fonctions de hachage à sens unique, signature numérique et scellement

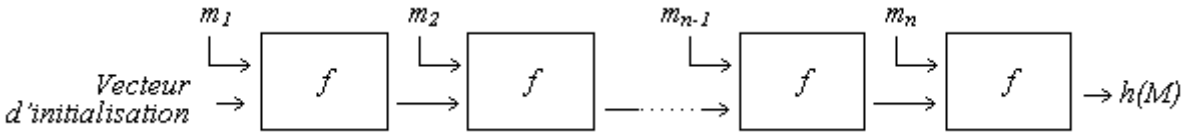
2.3.1. Fonctions de hachage à sens unique

Aussi appelée fonction de condensation, une **fonction de hachage** est une fonction qui convertit une chaîne de longueur quelconque en une chaîne de taille inférieure et généralement fixe ; la chaîne résultante est appelée *empreinte* (*digest* en anglais) ou condensé de la chaîne initiale.

Une **fonction à sens unique** est une fonction facile à calculer mais difficile à inverser. La cryptographie à clef publique repose sur l'utilisation de fonctions à sens unique à brèche secrète : pour qui connaît le secret (i.e. la clef privée), la fonction devient facile à inverser.

Une **fonction de hachage à sens unique** est une fonction de hachage qui est en plus une fonction à sens unique : il est aisé de calculer l'empreinte d'une chaîne donnée, mais il est difficile d'engendrer des chaînes qui ont une empreinte donnée, et donc de déduire la chaîne initiale à partir de l'empreinte. On demande généralement en plus à une telle fonction d'être **sans collision**, c'est-à-dire qu'il soit impossible de trouver deux messages ayant la même empreinte. On utilise souvent le terme fonction de hachage pour désigner, en fait, une fonction de hachage à sens unique sans collision.

La plupart des fonctions de hachage à sens unique sans collision sont construites par itération d'une fonction de compression : le message M est décomposé en n blocs $m_1,...,m_n$, puis une fonction de compression f est appliquée à chaque bloc et au résultat de la compression du bloc précédent ; l'empreinte $h(M)$ est le résultat de la dernière compression.



- Figure 6. Fonction de hachage itérative -

Des exemples de fonctions ainsi conçues sont MD5, SHA et RIPE-MD.

MD5 (Message Digest 5)

Développé par Rivest en 1991, MD5 produit une empreinte de 128 bits à partir d'un texte de taille arbitraire. MD5 manipule le texte d'entrée par blocs de 512 bits.

SHA (Secure Hash Algorithm)

SHA est la fonction de hachage utilisée par SHS (*Secure Hash Standard*), la norme du gouvernement Américain pour le hachage. SHA-1 est une amélioration de SHA publiée en 1994. SHA-1 produit une empreinte de 160 bits à partir d'un message de longueur maximale 2^{64} bits. Tout comme MD5, SHA-1 travaille sur des blocs de 512 bits.

RIPE-MD

Développée dans le cadre du projet RIPE (*RACE Integrity Primitives Evaluation*) de la communauté Européenne, RIPE-MD fournit une empreinte de 128 bits. RIPE-MD-160 est une version renforcée de RIPE-MD qui fournit une empreinte de 160 bits.

2.3.2. Intégrité et authentification de l'origine des données

Parmi les problèmes auxquels s'attaque la cryptographie, on trouve l'authentification de l'origine des données et l'intégrité : lorsque l'on communique avec une autre personne au travers d'un canal peu sûr, on aimerait que **le destinataire puisse s'assurer que le message émane bien de l'auteur auquel il est attribué et qu'il n'a pas été altéré pendant le transfert**.

Les fonctions de hachage à sens unique interviennent dans la résolution de ces problèmes. Si l'on dispose d'un canal sûr (mais plus coûteux) en parallèle du canal de communication normal, on peut communiquer l'empreinte des messages par l'intermédiaire de ce canal. On assure ainsi l'intégrité des

données transférées.

Sans canal sûr, le problème se complique : si l'on transfère l'empreinte sur un canal de communication non sûr, un intercepteur peut modifier les données puis recalculer l'empreinte. Il convient donc de trouver une méthode pour s'assurer que seul l'expéditeur est capable de calculer l'empreinte. Pour cela, on peut utiliser, par exemple, une fonction de hachage à sens unique qui fonctionne de plus avec une clef secrète ou privée. On remarquera que, ce faisant, on fournit également l'authentification de l'origine des données. Inversement, si on désire fournir l'authentification de l'origine des données et que l'on utilise pour cela un moyen qui ne garantit pas l'intégrité des données authentifiées, un intrus peut modifier le message et donc faire accepter comme authentifiées des données qu'il a choisies. C'est pourquoi **intégrité et authentification de l'origine des données sont généralement fournies conjointement par un même mécanisme**. J'utiliserai parfois le terme d'*authenticité* pour désigner l'intégrité jointe à l'authentification des données.

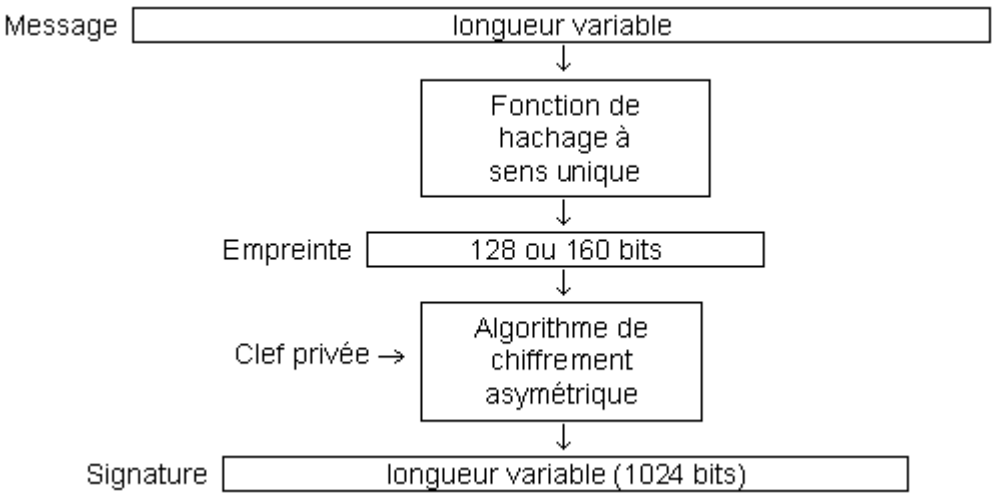
2.3.3. Signature numérique

La norme [ISO 7498-2] définit la signature numérique comme des "données ajoutées à une unité de données, ou transformation cryptographique d'une unité de données, permettant à un destinataire de prouver la source et l'intégrité de l'unité de données et protégeant contre la contrefaçon (par le destinataire, par exemple)". La mention "protégeant contre la contrefaçon" implique que seul l'expéditeur doit être capable de générer la signature.

Une signature numérique fournit donc les services d'**authentification de l'origine des données, d'intégrité des données et de non-répudiation**. Ce dernier point la différencie des *codes d'authentification de message* (voir paragraphe suivant), et a pour conséquence que la plupart des algorithmes de signature utilisent la cryptographie à clef publique.

Sur le plan conceptuel, la façon la plus simple de signer un message consiste à chiffrer celui-ci à l'aide d'une clef privée : seul le possesseur de cette clef est capable de générer la signature, mais toute personne ayant accès à la clef publique correspondante peut la vérifier. Dans la pratique, cette méthode est peu utilisée du fait de sa lenteur.

Une autre méthode utilisée pour signer consiste à **calculer une empreinte du message à signer et à ne chiffrer que cette empreinte**. Le calcul d'une empreinte par application d'une fonction de hachage étant rapide et la quantité de données à chiffrer étant fortement réduite, cette solution est bien plus rapide que la précédente.



- Figure 7. Signature -

DSS

Proposé par le NIST en 1991, puis finalement adopté en 1994, DSS (*Digital Signature Standard*) est la norme de signature numérique du gouvernement Américain. Elle se base sur l'algorithme DSA (*Digital Signature Algorithm*), qui utilise SHA comme fonction de hachage à sens unique et Elgamal pour la génération et la vérification de la signature.

RSA

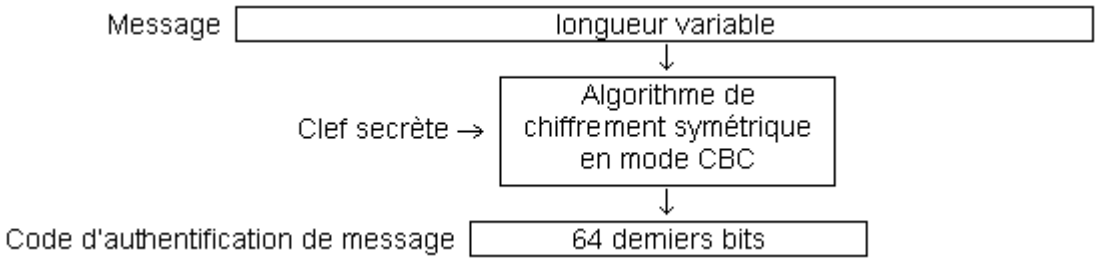
Si DSS est la norme officielle aux U.S.A., RSA est en revanche une norme de fait, qui est en pratique beaucoup plus utilisé pour la signature que DSA.

2.3.4. Scellement

Tout comme la signature numérique, le scellement fournit les services d'**authentification de l'origine des données et d'intégrité des données**, mais il ne fournit pas la non-répudiation. Ceci permet l'utilisation de la cryptographie à clef secrète pour la génération du sceau ou *code d'authentification de message*.

Un code d'authentification de message (*Message Authentication Code,* MAC) est le résultat d'une fonction de hachage à sens unique dépendant d'une clef secrète : l'empreinte dépend à la fois de l'entrée et de la clef. On peut construire un MAC à partir d'une fonction de hachage ou d'un algorithme de chiffrement par blocs. Il existe aussi des fonctions spécialement conçues pour faire un MAC.

Une façon courante de générer un code d'authentification de message consiste à appliquer un algorithme de chiffrement symétrique en mode CBC au message ; le MAC est le dernier bloc du cryptogramme.



- Figure 8. Scellement à l'aide d'un algorithme de chiffrement symétrique -

Un moyen simple de transformer une fonction de hachage à sens unique en un MAC consiste à chiffrer l'empreinte avec un algorithme à clef secrète. Une autre méthode consiste à appliquer la fonction de hachage non pas simplement aux données à protéger, mais à un ensemble dépendant à la fois des données et d'un secret.

Keyed-Hash

Un exemple simple de cette façon de procéder est de prendre pour MAC des valeurs du type H (secret, message), H(message, secret) ou H(secret, message, secret). Ces méthodes, présentées en 1992 par Gene Tsudik dans [Tsu92], s'appellent respectivement méthode du préfixe secret, du suffixe secret et de l'enveloppe secrète. Elles ont une sécurité limitée.

HMAC

Une méthode de calcul de MAC à base de fonction de hachage plus élaborée et plus sûre est HMAC, présenté dans [RFC 2104]. La méthode HMAC peut être utilisée avec n'importe quelle fonction de hachage itérative telle que MD5, SHA-1 ou encore RIPE-MD.

Soit H une telle fonction, K le secret et M le message à protéger. H travaille sur des blocs de longueur *b* octets (64 en général) et génère une empreinte de longueur *l* octets (16 pour MD5, 20 pour SHA et RIPE-MD-160). Il est conseillé d'utiliser un secret de taille au moins égale à *l* octets. On définit deux chaînes, *ipad* (*inner padding data*) et *opad* (*outer padding data*), de la façon suivante : *ipad* = l'octet 0x36 répété *b* fois, *opad* = l'octet 0x5C répété *b* fois. Le MAC se calcule alors suivant la formule suivante : $HMAC_K(M) = H (K \oplus opad, H(K \oplus ipad, M))$.

Une pratique courante avec les fonctions de calcul de MAC est de tronquer la sortie pour ne garder comme MAC qu'un nombre réduit de bits. Avec HMAC, on peut ainsi choisir de ne retenir que les *t* bits de gauche, où *t* doit être supérieur à *l/2* et 80. On désigne alors sous la forme HMAC-H-*t* l'utilisation de HMAC avec la fonction H, tronqué à *t* bits (par exemple, HMAC-SHA1-96).

2.4. Protocoles cryptographiques

Tout comme les protocoles de communication, les protocoles cryptographiques sont une série d'étapes prédéfinies, basées sur un langage commun, qui permettent à plusieurs participants (généralement deux) d'accomplir une tâche. Dans le cas des protocoles cryptographiques, les tâches en question sont bien sûr liées à la cryptographie : ce peut être une authentification, un échange de clef,... Une particularité des protocoles cryptographiques est que les tiers en présence ne se font généralement pas confiance et que le protocole a donc pour but d'empêcher l'espionnage et la tricherie.

2.4.1. Protocoles à apport nul de connaissance

Une situation fréquente est celle où un tiers doit **prouver à un autre la connaissance d'un secret sans rien révéler sur ce secret**. Les protocoles qui permettent de répondre à ce problème sont appelés protocoles à apport nul de connaissance ou preuves à divulgation nulle (*0-knowledge* en anglais).

Les protocoles à apport nul de connaissance prennent la forme de protocoles interactifs au cours desquels Victor, le vérificateur, pose à Patricia, le plaideur, une série de questions. Si Patricia connaît le secret, elle saura répondre correctement à toutes les questions ; sinon elle aura 50% de chances de répondre correctement à chaque question. La probabilité que Patricia connaisse effectivement le secret après *n* réponses correctes est donc de $1 - \frac{1}{2^n}$. Après 10 questions, cette probabilité sera donc de 0,999.

Une application de tels protocoles est leur utilisation pour prouver son identité. Ainsi, le schéma d'identification Feige-Fiat-Shamir est un système de preuve d'identité à divulgation nulle. Soit *n* un module aléatoire, produit de deux grands nombres premiers. La clef publique de Patricia est *v*, un résidu quadratique modulo *n* (i.e. $x^2 \equiv v \bmod n$ admet une solution et $v^{-1} \bmod n$ existe). La clef privée correspondante est le plus petit *s* tel que $s \equiv \sqrt{v^{-1}} \bmod n$. Le déroulement d'une ronde du protocole est le suivant :

1. Patricia choisit un nombre aléatoire $r \leq n$, puis calcule $x = r^2 \bmod n$ et envoie *x* à Victor.
2. Victor envoie un bit aléatoire $b = 0$ ou 1 à Patricia.

3. Patricia envoie $y = r \times s^b \bmod n$ à Victor.
   Si $b = 0$, $y = r$, donc Victor vérifie que $y^2 \equiv x \bmod n$; cela prouve que Patricia connaît $r = \sqrt{x}$
   Si $b = 1$, $y = r \times s$, donc Victor vérifie que $y^2 \times v \equiv x \bmod n$; cela prouve que Patricia connaît s.

Si Patricia ne connaît pas s, pour tromper Victor lorsque $b = 1$, elle doit lui envoyer $x = v^{-1}$; mais cela implique que $r = s$, donc Patricia ne saura pas répondre si $b = 0$. Inversement, si Patricia envoie effectivement une valeur x générée par $x = r^2 \bmod n$, elle saura répondre lorsque $b = 0$ ($y = r$), mais elle ne pourra pas répondre lorsque $b = 1$, car il lui faudrait connaître s. Dans chaque cas, Patricia a donc une chance sur deux de savoir répondre à la question.

Le principal problème des protocoles à apport nul de connaissance est le nombre important d'échanges nécessaires, qui les rend peu utilisés en pratique.

2.4.2. La notion de tiers de confiance

Beaucoup de protocoles cryptographiques, notamment ceux visant à sécuriser des environnements distribués, ont recours à la notion de *tiers de confiance*. Dans un tel cadre, pour établir une communication sécurisée, **Alice et Bob se font aider de Norbert le notaire, qui est une personne en qui ils ont confiance**. Norbert va jouer un rôle dans la sécurisation des échanges entre Alice et Bob en participant à la mise en œuvre de mécanismes de sécurité, notamment en intervenant dans les protocoles d'authentification et d'échange de clef. De nombreux protocoles et systèmes ayant recours à des tiers de confiance ont vu le jour, parmi lesquels Kerberos et Sesame par exemple.

2.4.3. échange de clef

Les deux méthodes les plus courantes d'établissement de clef sont le **transport** de clef et la **génération** de clef. Un exemple de transport de clef est l'utilisation d'un algorithme à clef publique pour chiffrer une clef de session générée aléatoirement. Un exemple de génération de clef est le protocole Diffie–Hellman, qui permet de générer un secret partagé à partir d'informations publiques (voir ci-dessous).

Le seul protocole décrit ici est le principe de génération de clef de Diffie–Hellman et les façons de l'étendre pour contrer les attaques de l'intercepteur. Diffie–Hellman est à la base de nombreux protocoles d'échange de clef.

Diffie–Hellman

Inventé en 1976 par Diffie et Hellman, ce protocole permet à deux tiers de **générer un secret partagé sans avoir aucune information préalable l'un sur l'autre**. Il est basé sur la cryptologie à clef publique (dont il est d'ailleurs à l'origine), car il fait intervenir des valeurs publiques et des valeurs privées. Sa sécurité dépend de la difficulté de calculer des logarithmes discrets sur un corps fini. Le secret généré à l'aide de cet algorithme peut ensuite être utilisé pour dériver une ou plusieurs clefs (clef secrète, *clef de chiffrement de clefs*,...).

Voici le déroulement de l'algorithme :

1. Alice et Bob se mettent d'accord sur un grand entier n tel que $(n-1)/2$ soit premier et sur un entier g primitif par rapport à n. Ces deux entiers sont publics.
2. Alice choisit de manière aléatoire un grand nombre entier a, qu'elle garde secret, et calcule sa valeur publique, $A = g^a \bmod n$. Bob fait de même et génère b et $B = g^b \bmod n$.
3. Alice envoie A à Bob ; Bob envoie B à Alice.

4. Alice calcule $K_{AB} = B^a \ mod \ n$; Bob calcule $K_{BA} = A^b \ mod \ n$. $K_{AB} = K_{BA} = g^{ab} \ mod \ n$ est le secret partagé par Alice et Bob.

Une personne qui écoute la communication connaît g, n, $A=g^a \ mod \ n$ et $B=g^b \ mod \ n$, ce qui ne lui permet pas de calculer $g^{ab} \ mod \ n$: il lui faudrait pour cela calculer le logarithme de A ou B pour retrouver a ou b.

En revanche, **Diffie–Hellman est vulnérable à l'attaque active dite attaque de l'intercepteur** (voir "3.4.3. Attaque de l'intercepteur").

Une façon de contourner le problème de l'attaque de l'intercepteur sur Diffie–Hellman est d'**authentifier les valeurs publiques utilisées pour la génération du secret partagé**. On parle alors de **Diffie–Hellman authentifié**. L'authentification peut se faire à deux niveaux :

- En utilisant des valeurs publiques authentifiées, à l'aide de certificats par exemple. Cette méthode est notamment à la base du protocole SKIP.
- En authentifiant les valeurs publiques après les avoir échangées, en les signant par exemple. Cette méthode est utilisée entre autres par le protocole Photuris.

L'inconvénient, dans les deux cas, est que l'on perd un gros avantage de Diffie–Hellman, qui est la possibilité de générer un secret partagé sans aucune information préalable sur l'interlocuteur.

Échange de clef et authentification mutuelle

Pour établir une communication sécurisée, on procède généralement, en premier lieu, à une authentification à des fins de contrôle d'accès. Puis, un échange de clef permet l'utilisation d'un mécanisme de sécurisation des échanges : l'authentification est ainsi étendue à la suite de la communication. L'exemple de Diffie–Hellman montre comme il est important que l'échange de clef soit authentifié. On parle alors de **protocole d'authentification mutuelle avec échange de clef**.

Propriétés des protocoles d'échange de clef

Diffie, Van Oorschot et Wiener définissent, dans [DOW92], la notion de protocole d'authentification mutuelle avec échange de clef sûr. Un protocole est dit sûr si les deux conditions suivantes sont valables dans chaque instance du protocole où l'un des deux tiers, par exemple Alice, exécute le protocole honnêtement et accepte l'identité de l'autre tiers :

- Au moment où Alice accepte l'identité de Bob, les enregistrements des messages échangés par les deux tiers se correspondent (i.e. les messages n'ont pas été altérés en route).
- Il est matériellement impossible pour toute personne autre que les tiers en présence de retrouver la clef échangée.

La propriété évoquée ci-dessus représente le minimum requis pour tout protocole. Cependant, d'autres propriétés des protocoles d'authentification avec échange de clef peuvent être souhaitables :

- La propriété dite de ***Perfect Forward Secrecy*** *(PFS)* est garantie si **la découverte par un adversaire du ou des secrets à long terme ne compromet pas les clefs de session** générées précédemment : les clefs de session passées ne pourront pas être retrouvées à partir des secrets à long terme. On considère généralement que cette propriété assure également que **la découverte d'une clef de session ne compromet ni les secrets à long terme ni les autres clefs de session**.
  A ce sujet, on parle d'attaque de Denning-Sacco lorsqu'un attaquant récupère une clef de session et utilise celle-ci, soit pour se faire passer pour un utilisateur légitime, soit pour

rechercher les secrets à long terme (par attaque exhaustive ou attaque par dictionnaire).

- La propriété dite de ***Back Traffic Protection*** est fournie si la génération de chaque clef de session se fait de manière indépendante : les nouvelles clefs ne dépendent pas des clefs précédentes et **la découverte d'une clef de session ne permet ni de retrouver les clefs de session passées ni d'en déduire les clefs à venir**.

- On dit qu'il y a **authentification directe** (*Direct Authentication*) si, **à la fin du protocole, les valeurs servant à générer le secret partagé sont authentifiées ou si chaque tiers a prouvé qu'il connaissait la clef de session**. Par opposition, l'authentification est dite indirecte (Indirect authentication) si elle n'est pas garantie à la fin du protocole, mais dépend de la capacité de chaque tiers à utiliser, dans la suite des échanges, la ou les clefs mises en place précédemment.

- On parle de protection de l'identité (*Identity Protection*) lorsque le protocole garantit qu'un attaquant espionnant les échanges ne pourra pas connaître les identités des tiers communiquants.

- Enfin, l'utilisation du temps (*Timestamps*) afin d'éviter la rejouabilité est très controversée du fait, principalement, de sa trop grande dépendance d'horloges synchronisées.

Hiérarchie des clefs

On peut définir plusieurs types de clefs suivant leurs rôles :

- **Clefs de chiffrement de clefs**
  Situées en haut de la hiérarchie, ces clefs servent exclusivement à chiffrer d'autres clefs et ont généralement une durée de vie longue. On notera que leur utilisation, restreinte au chiffrement de valeurs aléatoires que sont des clefs, rend les attaques par cryptanalyse plus difficiles à leur niveau. **La cryptographie à clef publique est souvent utilisée pour le transport de clefs en chiffrant la clef à transporter à l'aide d'une clef publique**.

- **Clefs maîtresses**
  Les clefs maîtresses sont des clefs qui ne servent pas à chiffrer mais uniquement à **générer d'autres clefs par dérivation**. Une clef maîtresse peut ainsi être utilisée, par exemple, pour générer deux clefs : une pour le chiffrement et une pour la signature.

- **Clefs de session ou de chiffrement de données** D'une **durée de vie généralement faible** (elle peut aller jusqu'à changer à chaque message), une telle clef, par opposition aux précédentes, sert à chiffrer des données et se situe donc au bas de la hiérarchie. Du fait de leur lenteur, les algorithmes asymétriques sont très peu utilisés en chiffrement de données, et **les clefs de session sont donc généralement des clefs secrètes**.

Il est à noter que des abus de langage ont souvent lieu avec ces termes. Ainsi, on parle parfois de clef de session pour désigner en fait une clef maîtresse de même durée de vie et servant à générer plusieurs clefs : une pour l'authentification, une pour le chiffrement,...

3. Cryptanalyse

Si le but de la cryptographie est d'élaborer des méthodes de protection, le but de la cryptanalyse est au contraire de casser ces protections. Une tentative de cryptanalyse d'un système est appelée une attaque, et elle peut conduire à différents résultats :

- Cassage complet : le cryptanalyste retrouve la clef de déchiffrement.
- Obtention globale : le cryptanalyste trouve un algorithme équivalent à l'algorithme de déchiffrement, mais qui ne nécessite pas la connaissance de la clef de déchiffrement.
- Obtention locale : le cryptanalyste retrouve le texte en clair correspondant à un message chiffré.
- Obtention d'information : le cryptanalyste obtient quelques indications sur le texte en clair ou la clef (certains bits de la clef, un renseignement sur la forme du texte en clair,...)

D'une manière générale, on suppose toujours que le cryptanalyste connaît le détail des algorithmes, fonctions mathématiques ou protocoles employés. Même si ce n'est pas toujours le cas en pratique, il serait risqué de se baser sur le secret des mécanismes utilisés pour assurer la sécurité d'un système, d'autant plus que l'usage grandissant de l'informatique rend de plus en plus facile la reconstitution de l'algorithme à partir du programme.

3.1. Attaques des fonctions de chiffrement

Les fonctions de chiffrement sont supposées rendre impossible le décryptement, c'est-à-dire la récupération d'un message clair sans la clef. A fortiori, elles doivent protéger le secret des clefs.

3.1.1. Classement des attaques en fonction des données dont dispose le cryptanalyste

On distingue plusieurs types d'attaques suivant les informations que peut obtenir le cryptanalyste. Ce sont :

- **L'attaque à texte chiffré seulement**, où le cryptanalyste ne connaît qu'un ensemble de textes chiffrés ; il peut soit retrouver seulement les textes en clair, soit retrouver la clef. En pratique, il est très souvent possible de deviner certaines propriétés du texte en clair (format ASCII, présence d'un mot particulier,...), ce qui permet de valider ou non le décryptement.

- **L'attaque à texte en clair connu**, où le cryptanalyste connaît non seulement les textes chiffrés, mais aussi les textes en clair correspondants ; son but est alors de retrouver la clef. Du fait de la présence, dans la plupart des textes chiffrés, de parties connues (en-têtes de paquets, champs communs à tous les fichiers d'un type donné,...), ce type d'attaques est très courant.

- **L'attaque à texte en clair choisi**, où le cryptanalyste peut, de plus, choisir des textes en clair à chiffrer et donc utiliser des textes apportant plus d'informations sur la clef. Si le cryptanalyste peut de plus adapter ses choix en fonction des textes chiffrés précédents, on parle d'**attaque adaptative**.

- **L'attaque à texte chiffré choisi**, qui est l'inverse de la précédente : le cryptanalyste peut choisir des textes chiffrés pour lesquels il connaîtra le texte en clair correspondant ; sa tâche est alors de retrouver la clef. Ce type d'attaques est principalement utilisé contre les systèmes à clef publique, pour retrouver la clef privée.

3.1.2. Attaques sur les algorithmes symétriques

Attaques au niveau des clefs

L'attaque la plus simple sur le plan conceptuel est l'**attaque exhaustive** ou attaque en force brute, qui consiste à essayer toutes les clefs possibles. Avec une clef de 56 bits par exemple, cela nécessite 2^{56} tests.

Si la clef recherchée est en fait un mot de passe ou si elle dérive d'un mot de passe, on a des chances de trouver la clef en testant des mots susceptibles d'avoir été choisis comme mot de passe. On parle alors d'**attaque par dictionnaire**, car le cryptanalyste se constitue un dictionnaire de mots à tester (ce peut être une liste de prénoms, l'ensemble des mots d'une langue donnée,...). Ce type d'attaques est bien sûr beaucoup plus rapide qu'une attaque exhaustive.

Cryptanalyse différentielle

La **cryptanalyse différentielle** est un type d'attaques qui peut-être utilisé contre les algorithmes de chiffrement par blocs itératifs. Elle utilise une attaque à texte en clair choisi et se base sur l'observation de l'évolution des différences entre deux textes lorsqu'ils sont chiffrés avec la même clef. En analysant ces différences entre paires de textes, il est possible d'attribuer des probabilités à chaque clef possible. à force d'analyser des paires de textes, on finit soit par trouver la clef recherchée, soit par réduire suffisamment le nombre de clefs possibles pour pouvoir mener une attaque exhaustive rapide. La cryptanalyse différentielle peut également être utilisée pour attaquer d'autres types d'algorithmes comme les fonctions de hachage.

Ce type d'attaques a été utilisé pour la première fois par Murphy en 1990 contre l'algorithme FEAL-4. Biham et Shamir ont mené des attaques différentielles sur le DES, mais leur efficacité reste limitée du fait de la conception des tables-S, qui avaient été optimisées contre ce type d'attaques. La meilleure attaque différentielle contre le DES complet à 16 rondes nécessite 2^{47} textes en clair choisis. Cela reste énorme et demande beaucoup trop de chiffrements, si bien qu'il est couramment admis que le DES est encore sûr par rapport à la cryptanalyse différentielle.

Cryptanalyse linéaire

La **cryptanalyse linéaire** utilise une attaque à texte en clair connu et consiste à modéliser l'algorithme de chiffrement par blocs par une approximation linéaire. Avec un nombre suffisant de paires (texte en clair, texte chiffré), on peut deviner certains bits de la clef.

La cryptanalyse linéaire fut utilisée pour la première fois en 1992 par Matsui et Yamagishi pour attaquer FEAL. Elle fut étendue par Matsui en 1993 pour attaquer DES. Les tables-S du DES ne sont pas optimisées pour contrer la cryptanalyse linéaire, et la meilleure attaque différentielle actuelle contre le DES nécessite 2^{43} textes en clair connus en moyenne. Une réalisation logicielle de cette attaque a découvert une clef DES en 50 jours avec 12 stations de travail.

En 1994, Langford et Hellman introduisirent une attaque appelée cryptanalyse différentielle linéaire qui combine des éléments des deux méthodes précédentes.

3.1.3. Attaques sur les algorithmes asymétriques

Attaques au niveau des clefs

Avec les algorithmes asymétriques, le problème n'est pas de trouver la bonne clef par attaque exhaustive, mais de **dériver la clef secrète à partir de la clef publique**. La plupart des algorithmes asymétriques reposant sur des problèmes mathématiques difficiles à résoudre, **cela revient généralement à résoudre ce problème**. C'est pourquoi les paramètres qui influencent la difficulté de résolution du problème sont les éléments déterminant la sécurité. Généralement, cela se traduit par la nécessité d'utiliser de grands nombres, la taille de ces nombres ayant une répercussion sur la longueur des clefs. Cela explique que les clefs utilisées par la cryptographie à clef publique soient généralement bien plus longues que celles utilisées par la cryptographie à clef secrète.

Par exemple, dans le cas de RSA, l'élément déterminant est la taille du module. La factorisation d'un

module de 512 bits est à la portée d'une agence gouvernementale, 1024 bits est considéré comme sûr actuellement, et 2048 bits garantit une sécurité à long terme.

Attaque à texte en clair deviné et problème de la faible entropie

Un point faible des algorithmes à clef publique est le caractère public de la clef de chiffrement : le cryptanalyste ayant connaissance de cette clef, il peut mener une **attaque à texte en clair deviné**, qui consiste à tenter de deviner le texte en clair et à le chiffrer pour vérifier son exactitude. Cette particularité implique donc une **restriction sur l'utilisation des algorithmes asymétriques** : il faut absolument éviter de les utiliser avec un ensemble de textes en clair possibles restreint (i.e. de faible entropie). En effet, si tel était le cas, un attaquant pourrait aisément se constituer une liste exhaustive des textes en clair possibles et des textes chiffrés correspondants. Il n'aurait alors aucun mal à retrouver le texte en clair correspondant à un cryptogramme donné.

Attaque à texte chiffré choisi

D'autre part, la plupart des algorithmes asymétriques sont sensibles aux **attaques à texte chiffré choisi**. Il convient donc, si l'on utilise le même algorithme pour le chiffrement et pour la signature, de s'assurer que les protocoles employés ne permettent pas à un adversaire de faire signer n'importe quel texte. Mieux, on peut avoir recours à des couples (clef publique, clef privée) distincts pour ces deux opérations.

Attaque temporelle

Un type d'attaque apparu récemment est l'**attaque temporelle**, qui utilise la mesure du temps pris par des opérations cryptographiques pour retrouver les clefs privées utilisées. Cette attaque a été réalisée avec succès contre des cartes à microcircuits, des calculettes de sécurité et contre des serveurs de commerce électronique à travers l'Internet. La société Counterpane Systems, entre autres, a généralisé ces méthodes pour y inclure des **attaques sur des systèmes en mesurant la consommation, les émissions radioélectriques, et autres "canaux latéraux"** ; ils les ont mises en œuvre contre plusieurs types d'algorithmes à clef publique ou à clef secrète utilisés dans des calculettes. Une recherche voisine s'est intéressée à l'introduction délibérée d'erreurs dans les processeurs cryptographiques pour tenter d'obtenir des informations sur la clef.

3.2. Attaques des générateurs pseudo-aléatoires

Les générateurs pseudo-aléatoires sont supposés être imprévisibles. Un modèle d'attaque courant consiste à observer les aléas engendrés pendant un certain temps pour deviner la suite, voire l'état du générateur.

3.3. Attaques des fonctions de hachage à sens unique

Les deux attaques exhaustives de base que l'on peut mener contre une fonction de hachage à sens unique consistent à tester des entrées aléatoires avec la fonction jusqu'à :

- trouver une entrée qui donne en sortie une empreinte donnée (contredisant de ce fait la propriété "à sens unique") ;
- trouver deux entrées qui produisent la même sortie (contredisant de ce fait la propriété "sans collision").

Supposons que la fonction de hachage génère une empreinte de n bits. Si l'on veut trouver une entrée qui produira une valeur de sortie h donnée, alors, chaque sortie étant équiprobable, il faudra essayer 2^n valeurs possibles d'entrée pour avoir une bonne chance de trouver.

Attaque des anniversaires

Le paradoxe des anniversaires est un résultat célèbre du problème de probabilité suivant : combien faut-il de personnes dans une pièce pour qu'il y ait plus d'une chance sur deux pour que deux personnes au moins soient nées le même jour de l'année. La réponse est surprenante : 23 personnes suffisent ! D'une façon plus générale, considérons une fonction qui, lorsqu'on lui fournit une entrée aléatoire, retourne, de façon équiprobable, une valeur parmi k valeurs possibles (k grand). Si l'on applique cette fonction à différentes valeurs d'entrée, on a plus d'une chance sur deux de retomber sur une sortie déjà obtenue après avoir effectué un nombre d'essais de l'ordre de $\sqrt{k}$.

Si l'on applique ce résultat à une fonction de hachage générant une empreinte de n bits, il y a $k=2^n$ valeurs possibles en sortie, donc il faudra essayer seulement de l'ordre de $2^{n/2}$ valeurs d'entrées. Au lieu d'être en $O(2^n)$, la complexité de l'attaque n'est plus qu'en $O(2^{n/2})$!

L'attaque des anniversaires est surtout intéressante avec les algorithmes de signature basés sur une fonction de hachage : elle permet de faire signer à un tiers un message correct, mais auquel correspond un message frauduleux ayant la même empreinte. Yuval a proposé la stratégie suivante :

- L'adversaire choisit un message frauduleux qu'il désire faire signer, et un message inoffensif qu'Alice acceptera sûrement de signer.
- L'adversaire génère $2^{n/2}$ variantes du message inoffensif (en faisant, par exemple, des modifications rédactionnelles mineures) et les empreintes correspondantes. Il génère ensuite un nombre égal de variantes du message frauduleux.
- La probabilité qu'une des variantes du message inoffensif et une des variantes du message frauduleux donnent la même empreinte est supérieure à 1/2 selon le paradoxe des anniversaires.
- L'adversaire fait alors signer la variante du message inoffensif à Alice.
- La signature du message inoffensif est alors retirée et attachée à la variante du message frauduleux qui donne la même empreinte. L'adversaire a donc forgé un faux message signé sans avoir besoin de connaître les clefs de chiffrement.

Pour éviter de telles attaques, il convient de choisir une longueur d'empreinte assez élevée. Les empreintes de 128 bits générées par MD5 sont désormais trop justes ; les 160 bits de SHA-1 et RIPE-MD-160 sont plus sûrs.

3.4. Attaques des protocoles cryptographiques

On distingue deux types d'attaques sur les protocoles : les **attaques passives** et les **attaques actives**. Dans le premier cas, l'attaquant ne peut qu'espionner les données échangées par les tiers communiquants, alors que dans le second cas il peut modifier ou supprimer des messages, en ajouter des nouveaux ou des anciens qu'il rejoue.

3.4.1. Attaque par mascarade

On parle d'attaque par mascarade (*spoofing attack*) lorsqu'un attaquant **essaye de se faire passer pour un utilisateur légitime** en exécutant un protocole à la place de celui-ci. La conception d'un protocole a bien sûr pour but principal de contrecarrer ce type d'attaques.

3.4.2. Attaque par rejeu

Une attaque par rejeu (*replay attack*) consiste à **envoyer, dans une communication, des messages interceptés au cours d'une autre communication ou plus tôt dans la communication**. Ce type

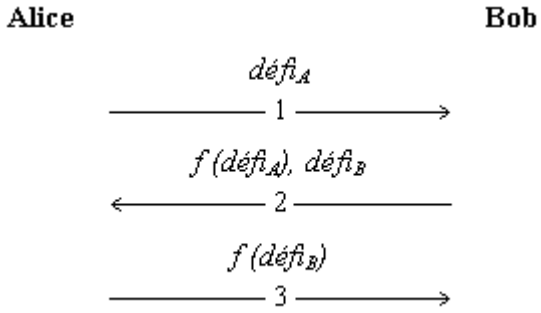
d'attaques permet de contourner des protocoles simples comme, par exemple, une authentification par mot de passe : il suffit à un adversaire d'avoir espionné un échange pour connaître le mot de passe et donc pour pouvoir se faire passer pour un utilisateur légitime.

Les méthodes pour se prémunir contre ce type d'attaques sont l'utilisation de marquages temporels (*timestamps*) ou de défis imprévisibles et à usage unique.

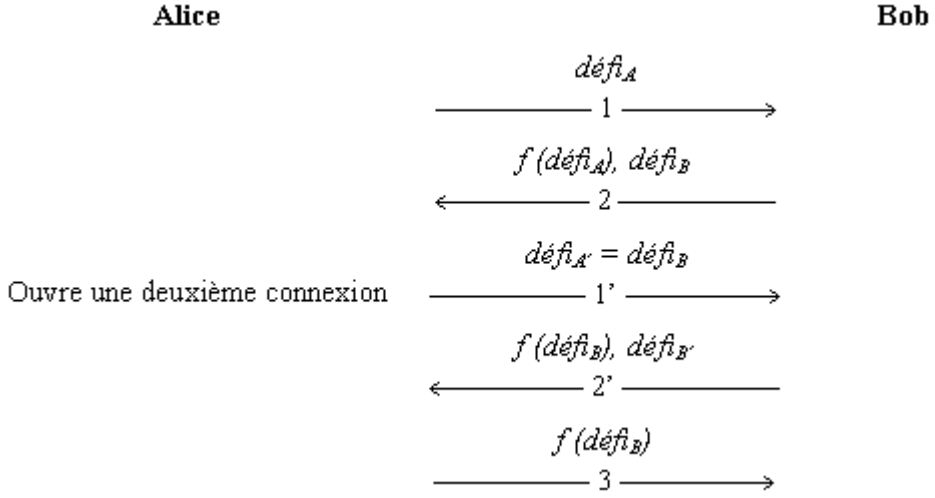
3.4.3. Attaque par réflexion

Une attaque par réflexion (*reflection attack*) est une attaque pour laquelle **l'adversaire exploite le caractère symétrique d'un protocole** pour répondre aux défis de son interlocuteur en utilisant des réponses fournies par l'interlocuteur lui-même.

Considérons par exemple le protocole d'authentification mutuelle suivant :



Pour se faire passer pour Alice auprès de Bob, il suffit à un adversaire d'établir deux connexions en parallèle avec Bob, et d'envoyer $défi_{A'} = défi_B$ comme second défi à Bob. Celui-ci fournira alors la réponse attendue à son premier défi :



3.4.4. Attaque de l'intercepteur

Le principe de l'attaque de l'intercepteur (*man-in-the-middle attack*) est que, pendant que deux tiers procèdent à un échange de clef, en utilisant un algorithme du type Diffie–Hellman par exemple, **un adversaire se positionne entre les deux tiers et intercepte les échanges**. De cette façon, il procède à un échange de clef avec chaque tiers. à la fin du protocole, chaque tiers utilisera donc une clef différente, chacune de ces clefs étant connue de l'intercepteur. Pour chaque message transmis par la suite, l'intercepteur procédera à son déchiffrement avec la clef correspondante puis le rechiffrera

avec l'autre clef avant de l'envoyer à son destinataire : les deux tiers croiront communiquer de façon sûre alors que l'intercepteur pourra en fait lire tous les messages, voire même forger de faux messages.

Voici comment se déroule cette attaque dans le cas de Diffie–Hellman :

1. Alice envoie sa valeur publique $A = g^a \ mod \ n$ à Bob ; Ingrid l'intercepteur remplace cette valeur publique par la sienne. Bob reçoit donc $I = g^i \ mod \ n$.
2. Bob envoie sa valeur publique à Alice ; Ingrid remplace là aussi cette valeur par la sienne.
3. Alice génère le "secret" $K_{AI} = I^a \ mod \ n$. Ingrid génère le même secret en calculant $A^i \ mod \ n$.
4. Bob génère le "secret" $K_{BI} = I^b \ mod \ n$. Ingrid génère le même secret en calculant $B^i \ mod \ n$.

Alice et Bob croient tous les deux être en possession d'un secret partagé, mais en fait chacun d'eux partage un secret différent avec Ingrid.

Sigles et acronymes

3DESE	Triple-DES Encryption protocol
AFNOR	Association Française de Normalisation
CAM	Code d'Authentification de Message
CBC	Cipher Block Chaining
DES	Data Encryption Standard
DESE	DES Encryption protocol
DH	Diffie–Hellman
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECB	Electronic CodeBook
FAQ	Frequently Asked Questions
FEAL	Fast Encryption ALgorithm
HMAC	a MAC mechanism based on cryptographic Hash functions
ICV	Integrity Check Value
IDEA	International Data Encryption Algorithm
ISO	International Standardization Organization
IV	Initialization Vector
KDC	Key Distribution Center
MAC	Message Authentication Code
MD*n*	Message Digest *n*
NIST	National Institute of Standards and Technology
OSI	Open Systems Interconnection

PFS	Perfect Forward Secrecy
PIN	Personal Identification Number
RACE	Research and development in Advanced Communication technologies in Europe
RC*n*	Rivest's Code *n*
RFC	Request For Comments
RIPE	RACE Integrity Primitives Evaluation
RIPE-MD	RIPE Message Digest
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm

Glossaire

Algorithme cryptographique ou de chiffrement

Procédé ou fonction mathématique utilisée pour le *chiffrement* et le *déchiffrement*. Dans la cryptographie moderne, l'algorithme est souvent public et le secret du chiffre dépend d'un paramètre appelé *clef*.

Voir le chapitre "2.1. Confidentialité et algorithmes de chiffrement".

Analyse du trafic

Observation des caractéristiques extérieures du trafic transitant sur un réseau afin de tenter d'en tirer des informations : fréquence des transmissions, identités des tiers communiquants, quantités de données transférées. Associées à des informations de nature différente (date de rendez-vous, actualité,...) ces éléments peuvent permettre aux adversaires de faire des déductions intéressantes.

Authenticité

Terme utilisé dans ce document pour désigner le service de sécurité qui consiste à assurer à la fois l'*intégrité* et l'*authentification de l'origine des données*.

Voir le chapitre "2.3.2. Intégrité et authentification de l'origine des données".

Authentification

On distingue deux types d'authentification :

- Authentification d'un tiers
  C'est l'action qui consiste à prouver son identité.
  Ce service est généralement rendu par l'utilisation d'un "échange d'authentification" qui implique un certain dialogue entre les tiers communiquants.

- Authentification de l'origine des données
  Elle sert à prouver que les données reçues ont bien été émises par l'émetteur déclaré.
  Dans ce cas, l'authentification désigne souvent la combinaison de deux services : authentification et intégrité en mode non connecté. Ces deux services n'ont en effet pas de sens séute;parément et sont souvent fournis conjointement.

Voir le chapitre "2.3.2. Intégrité et authentification de l'origine des données".

CAM - Voir *Code d'authentification de message*

Certificat

Document électronique qui renferme la clef publique d'une entité, ainsi qu'un certain nombre d'informations la concernant, comme son identité. Ce document est signé par une autorité de certification ayant vérifié les informations qu'il contient.

Chiffrement, chiffrer

Application d'un *algorithme cryptographique* à un ensemble de données appelées *texte en clair* afin d'obtenir un *texte chiffré*.
Le chiffrement est un mécanisme de sécurité permettant d'assurer la confidentialité des données.

Voir le chapitre "1. Terminologie".

Clef (secrète, publique, privée)

Paramètre d'un algorithme de *chiffrement* ou de *déchiffrement*, sur lequel repose le secret.
On distingue deux types de clefs :

- les clefs secrètes, utilisées par les algorithmes symétriques, pour lesquels la clef de *chiffrement* et de *déchiffrement* sont égales.
- les couples (clef publique, clef privée), utilisés par les algorithmes asymétriques, pour lesquels clef de *chiffrement* et de *déchiffrement* sont distinctes.

Voir le chapitre "2.1. Confidentialité et algorithmes de chiffrement".

Clef de chiffrement de clefs

Clef utilisée exclusivement pour chiffrer d'autres clefs, afin de les faire parvenir à un interlocuteur. Une clef de chiffrement de clef a généralement une durée de vie assez longue, par opposition aux clefs qu'elle sert à chiffrer.

Voir le chapitre "Hiérarchie des clefs".

Clef de session

Clef ayant une durée de vie très limitée, généralement à une session.
Les clefs de session sont généralement des clefs secrètes, utilisées pour chiffrer les données transmises, et que les tiers communiquants génèrent en début de communication.

Voir le chapitre "Hiérarchie des clefs".

Clef maîtresse

Clef servant à générer d'autres clefs.

Voir le chapitre "Hiérarchie des clefs".

Code d'authentification de message (CAM)

Résultat d'une *fonction de hachage à sens unique* à clef secrète. L'empreinte dépend à la fois des données et de la clef ; elle n'est donc calculable que par les personnes connaissant la clef. Adjointe aux données sur lesquelles elle a été calculée, elle permet de vérifier leur authenticité (authentification + intégrité).

Voir le chapitre "2.3.4. Scellement".

Confidentialité

Service de sécurité qui consiste à s'assurer que seules les personnes autorisées peuvent prendre connaissance d'un ensemble de données.
Le mécanisme qui permet d'obtenir ce service est généralement le chiffrement des données concernées à l'aide d'un algorithme cryptographique.
On parle aussi de confidentialité du trafic lorsqu'on désire empêcher l'analyse du trafic en cachant les adresses source et destination, la taille des paquets, la fréquence des échanges,...

Contrôle d'accès

Service de sécurité permettant de déterminer, après avoir authentifié un utilisateur, quels sont ses privilèges et de les appliquer. Ce service a pour but d'empêcher l'utilisation d'une ressource (réseau, machine, données,...) sans autorisation appropriée.

Cryptage, crypter

Termes dérivés de l'anglais *to encrypt* et souvent employés incorrectement à la place de *chiffrement* et *chiffrer*. En toute rigueur, ces termes n'existent pas dans la langue française.
Si le "cryptage" existait, il pourrait être défini comme l'inverse du *décryptage*, c'est-à-dire comme l'action consistant à obtenir un *texte chiffré* à partir d'un *texte en clair* sans connaître la *clef*. Un exemple concret pourrait être de signer un texte choisi en reproduisant un chiffrement avec la *clef privée* de la victime. Mais on préfère parler dans ce cas de contrefaçon.

Cryptanalyse ou analyse cryptographique

Science qui étudie la sécurité des procédés cryptographiques pour tenter de trouver des faiblesses et pouvoir en particulier effectuer un *décryptement* avec succès.

"Analyse d'un système cryptographique, et/ou de ses entrées et sorties, pour en déduire des variables confidentielles et/ou des données sensibles (y compris un *texte en clair*)." [ISO 7498-2]

Voir le chapitre "3. Cryptanalyse".

Cryptogramme

Aussi appelé *texte chiffré*. Données obtenues par application d'un algorithme de chiffrement. Le contenu sémantique de ces données n'est pas compréhensible.

Voir le chapitre "1. Terminologie".

Cryptographie

Étude du *chiffrement* et du *déchiffrement*, ainsi que des procédés permettant d'assurer l'intégrité, l'authentification,...

"Discipline incluant les principes, moyens et méthodes de transformation des données, dans le but de cacher leur contenu, d'empêcher que leur modification passe inaperçue et/ou d'empêcher leur utilisation non autorisée." [ISO 7498-2]

Voir le chapitre "2. Cryptographie".

Cryptologie

Étude scientifique de la *cryptographie* et de la *cryptanalyse*.

Déchiffrement

Action inverse du *chiffrement*, lorsque celui-ci est réversible : à l'aide d'un *algorithme cryptographique* et d'une *clef*, on reconstruit le *texte en clair* à partir du *texte chiffré*.

Voir le chapitre "1. Terminologie".

Décryptement, décryptage

Action qui consiste à "casser" le chiffrement d'un texte de façon à retrouver le *texte en clair* sans connaître la *clef* qui permet son *déchiffrement* normal.

Voir le chapitre "1. Terminologie".

Déni de service

"Impossibilité d'accès à des ressources pour des utilisateurs autorisés ou introduction d'un retard pour le traitement d'opérations critiques." [ISO 7498-2]

Disponibilité

Service de sécurité qui assure une protection contre les attaques visant à dégrader ou rendre impossible l'accès à un service.

Empreinte (digest)

Aussi appelé condensé.
Chaîne de taille fixe obtenue par application d'une *fonction de hachage* à un ensemble de données.

Voir le chapitre "2.3.1. Fonctions de hachage à sens unique".

Fonction à sens unique

Une fonction à sens unique est une fonction facile à calculer mais difficile à inverser.
La cryptographie à clef publique repose sur l'utilisation de fonctions à sens unique à brèche secrète : pour qui connaît le secret (i.e. la clef privée), la fonction devient facile à inverser.

Voir le chapitre "2.3.1. Fonctions de hachage à sens unique".

Fonction de hachage

Aussi appelée fonction de condensation.

Fonction qui convertit une chaîne de longueur quelconque en une chaîne de taille inférieure et généralement fixe ; cette chaîne est appelée empreinte (*digest* en anglais) ou condensé de la chaîne initiale.

Voir le chapitre "2.3.1. Fonctions de hachage à sens unique".

Fonction de hachage à sens unique

Fonction de hachage qui est en plus une fonction à sens unique : il est aisé de calculer l'empreinte d'une chaîne donnée, mais il est difficile d'engendrer des chaînes qui ont une empreinte donnée. On demande généralement en plus à une telle fonction d'être sans collision, c'est-à-dire qu'il soit impossible de trouver deux messages ayant la même empreinte.

Voir le chapitre "2.3.1. Fonctions de hachage à sens unique".

ICV (Integrity Check Value)

"Valeur de vérification d'intégrité". Cette valeur est calculée par l'expéditeur sur l'ensemble des données à protéger. L'ICV est alors envoyée avec les données protégées. En utilisant le même algorithme, le destinataire recalcule l'ICV sur les données reçues et la compare à l'ICV originale. Si elles se correspondent, il en déduit que les données n'ont pas été modifiées.

Implémenter

Anglicisme, de *to implement* : mettre en œuvre, réaliser.

Intégrité

Service de sécurité qui consiste à s'assurer que seules les personnes autorisées pourront modifier un ensemble de données. Dans le cadre de communications, ce service consiste à permettre la détection de l'altération des données durant le transfert.
On distingue deux types d'intégrité :

- L'intégrité en mode non connecté permet de détecter des modifications sur un datagramme individuel, mais pas sur l'ordre des datagrammes.
- L'intégrité en mode connecté permet en plus de détecter la perte de paquets ou leur réordonnancement.

L'intégrité est très liée à l'authentification de l'origine des données, et les deux services sont souvent fournis conjointement.

Liaison

Ensemble de matériels (câbles, modems, concentrateurs, routeurs,...) qui relient physiquement deux équipements terminaux.

MAC (Message Authentication Code) - Voir *Code d'authentification de message*

Mascarade (Masquerade, spoofing)

Acte de prétendre être une autre entité dans le but d'accéder aux ressources de celle-ci ; incident au cours duquel un tiers non autorisé prétend être le véritable utilisateur.

Message

Dans le monde des réseaux, un message est une suite de données binaires formant un tout logique pour les tiers communiquants.
Lorsqu'un message est trop long pour être transmis d'un seul bloc, il est segmenté et chaque segment est envoyé séparément dans un *paquet* distinct.

Non-rejouabilité

Garantie qu'un adversaire ayant intercepté des messages au cours d'une communication ne pourra pas les faire passer pour des messages valides en les injectant soit dans une autre communication, soit plus tard dans la même communication.

Perfect Forward Secrecy (PFS)

Propriété d'un protocole d'échange de clef selon laquelle la découverte, par un attaquant, du ou des secrets à long terme utilisés ne permet pas de retrouver les clefs de sessions.

Voir le chapitre "Propriétés des protocoles d'échange de clef".

PIN (Personal Identification Number)

Code secret, généralement de petite taille, incorporé dans une carte à puce et qui empêche une personne non autorisée d'utiliser la carte. Un mécanisme bloque la carte après un certain nombre d'essais infructueux.

Rejeu

Action consistant à envoyer un message intercepté précédemment, en espérant qu'il sera accepté comme valide par le destinataire.

Répudiation

"Le fait, pour une des entités impliquées dans la communication, de nier avoir participé aux échanges, totalement ou en partie." [ISO 7498-2]

RFC (Request For Comment)

Littéralement, "Appel à commentaires". C'est en fait un document décrivant un des aspects d'Internet de façon relativement formelle (généralement, spécification d'un protocole). Ces documents sont destinés à être diffusés à grande échelle dans la communauté Internet et servent souvent de référence. On peut les trouver sur la plupart des sites FTP.

Sans connexion - Voir *Connexion*

Scellement (seal) - Voir aussi *Code d'authentification de message*

Mécanisme de sécurité permettant d'assurer l'intégrité et l'authentification de l'origine des données.

Voir le chapitre "2.3.4. Scellement".

Signature numérique

"Données ajoutées à une unité de données, ou transformation cryptographique d'une unité de

données, permettant à un destinataire de prouver la source et l'intégrité de l'unité de données et protégeant contre la contrefaçon (par le destinataire, par exemple)." [ISO 7498-2].
Une signature numérique fournit donc les services d'authentification de l'origine des données, d'intégrité des données et de non-répudiation. Ce dernier point la différencie des *codes d'authentification de message*, et a pour conséquence que la plupart des algorithmes de signature utilisent la cryptographie à clef publique.
D'autre part, la signature peut prendre deux formes :

- "transformation chiffrée" : un algorithme cryptographique modifie directement le message (par exemple chiffrement du message avec une clef privée).
- "données annexées" : des données supplémentaires sont adjointes au message (par exemple une empreinte, chiffrée avec une clef privée).

Voir le chapitre "2.3.3. Signature numérique".

Somme de contrôle

Condensé d'un ensemble de données, calculé par l'expéditeur avant l'envoi des données et recalculé par le destinataire à la réception pour vérifier l'intégrité des données transmises.

Texte chiffré

Aussi appelé *cryptogramme*.
Données obtenues par application d'un algorithme de chiffrement. Le contenu sémantique de ces données n'est pas compréhensible.

Voir le chapitre "1. Terminologie".

Texte en clair

Données intelligibles, dont la sémantique est compréhensible.

Voir le chapitre "1. Terminologie".

Tierce partie (ou tiers) de confiance (Trusted Third Party)

Tiers jouant un rôle dans la sécurisation des échanges entre deux partenaires en participant à la mise en œuvre de mécanismes de sécurité. On parle aussi de notarisation.

Voir le chapitre "2.4.2. La notion de tiers de confiance".

Tunneling

Technique consistant à créer un "tunnel" entre deux points du réseau en appliquant une transformation aux paquets à une extrémité (généralement, une *encapsulation* dans un protocole approprié) et en les reconstituant à l'autre extrémité.

Vecteur d'initialisation (Initialization Vector, IV)

Bloc de texte de valeur quelconque servant à initialiser un chiffrement avec chaînage de blocs, et donc à faire en sorte que deux messages identiques donnent des cryptogrammes distincts.

Voir le chapitre "Le mode CBC (Cipher Block Chaining)".

Bibliographie commentée

Cette bibliographie liste les principaux documents utilisés pour la réalisation de ce document.

1. Généralités

[Sch96]

Schneier Bruce, *Cryptographie appliquée - Algorithmes, protocoles et codes source en C - 2ème édition*, International Thomson Publishing France, 1997.

Ce livre est une traduction, le titre original est *Applied Cryptography - Protocols, Algorithms, and Source Code in C - 2nd Edition*.

▷ Livre de référence sur la cryptographie moderne.

[RSA - FAQ]

RSA Laboratories, *Frequently Asked Questions About Today's Cryptography - Version 4.0*, 1998.

▷ Présentation rapide de nombreux points de cryptographie sous forme de FAQ.

[ISO 7498-2]

International Standardization Organization, *Systèmes de traitement de l'information : interconnexion de systèmes ouverts - Modèle de référence de base - Partie 2 : architecture de sécurité*, NOR Z 70-102 (NF ISO 7498-2), AFNOR, 1989.

▷ Description générale des services de sécurité OSI et des mécanismes associés, et notamment définitions formelles des termes relatifs à la sécurité des réseaux et des échanges.

2. Codes d'authentification de messages

[Tsu92]

Tsudik Gene, *Message Authentication with One-Way Hash Functions*, ACM Computer Communication Review, Vol. 22, pp. 29-38, 1992.

▷ Méthodes du préfixe secret, du suffixe secret et de l'enveloppe secrète.

[RFC 2104]

Krawczyk Hugo, Bellare Mihir, Canetti R, *HMAC: Keyed-Hashing for Message Authentication*, RFC 2104, Février 1997.

▷ HMAC

3. échange de clef

[DOW92]

Diffie Whitfield, Van Oorschot Paul C., Wiener Michael J., *Authentication and Authenticated Key Exchanges*, Designs, Codes and Cryptography, 2, pp. 107-125, Kluwer Academic

Publishers, 1992.

▷ Les protocoles d'authentification mutuelle avec échange de clef : définition d'un protocole sûr, les propriétés souhaitées pour de tels protocoles, présentation du protocole STS.