Introduction à la cryptographie

Ghislaine.Labouret@hsc.fr

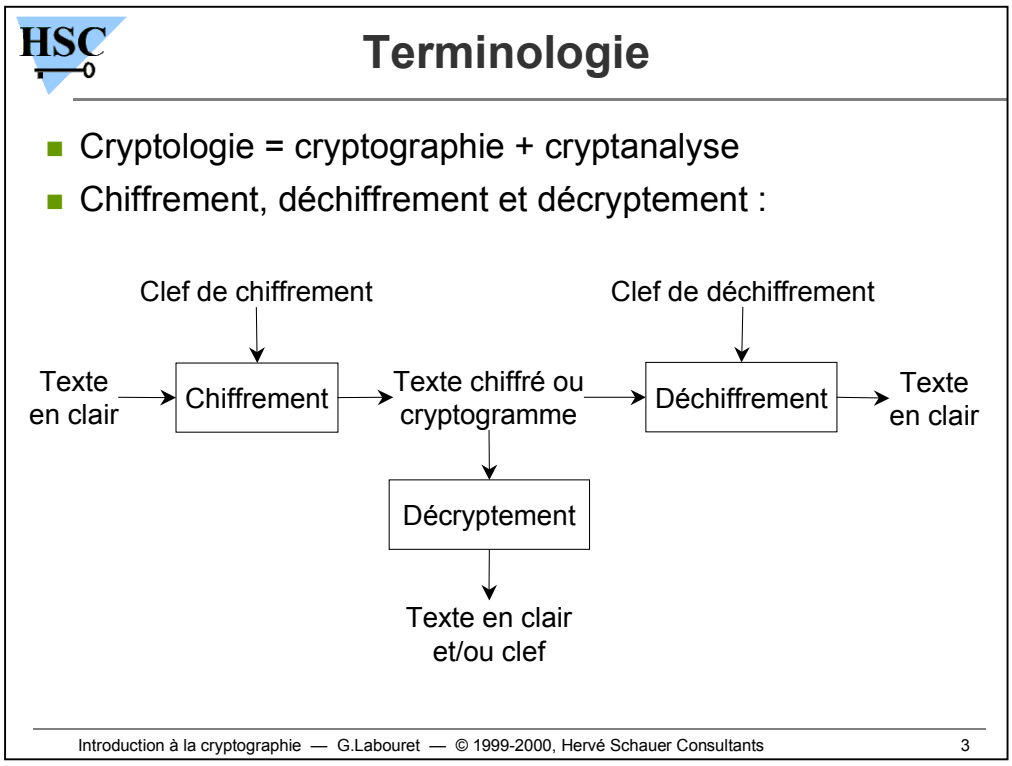
Hervé Schauer Consultants

www.hsc.fr

Support de cours du cabinet Hervé Schauer Consultants (HSC)

Plan

- Terminologie
- Mécanismes et services de sécurité
- Confidentialité et algorithmes de chiffrement
- Fonctions de hachage, scellement et signature
- Authentification mutuelle et échange de clefs de session
- Certificats et PKI
- Législation française
- Pour en savoir plus...

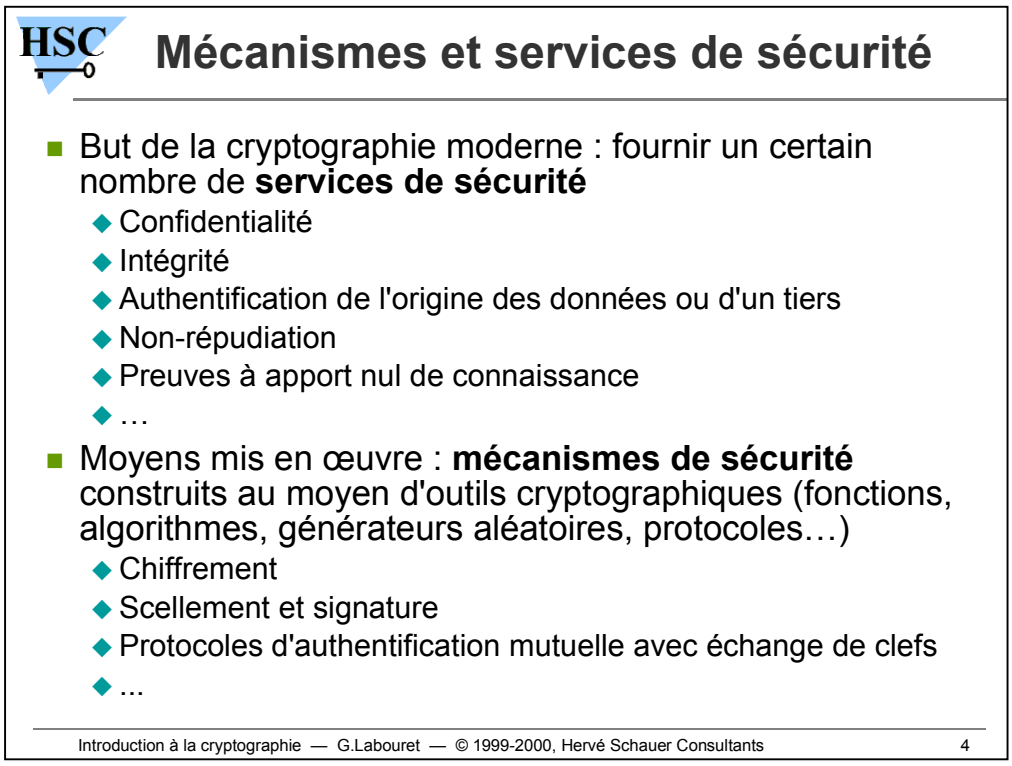


La **cryptologie** est une science mathématique qui comporte deux branches : la cryptographie et la cryptanalyse.

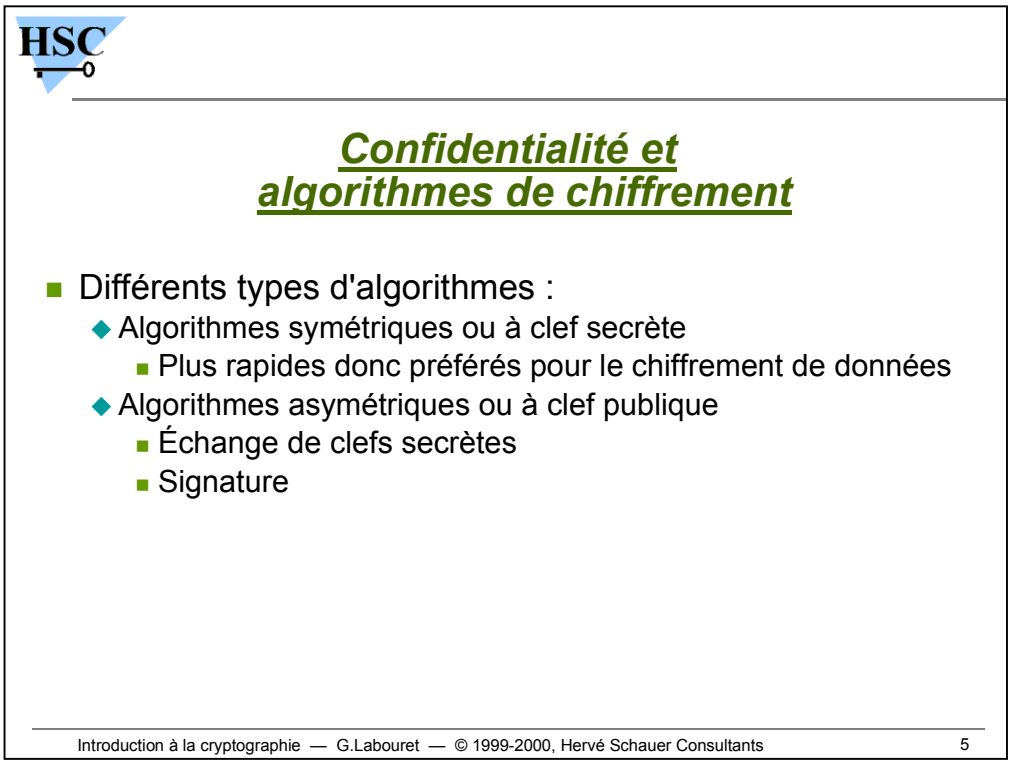
La **cryptographie** traditionnelle est l'étude des méthodes permettant de transmettre des données de manière confidentielle. Afin de protéger un message, on lui applique une transformation qui le rend incompréhensible ; c'est ce qu'on appelle le **chiffrement**, qui, à partir d'un **texte en clair**, 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.

Note : Les termes "cryptage" et "crypter" sont des anglicismes, dérivés de l'anglais *to encrypt*, 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 et de l'action de forger une signature.



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 cours quelles sont les techniques que la cryptographie fournit pour réaliser ces mécanismes.

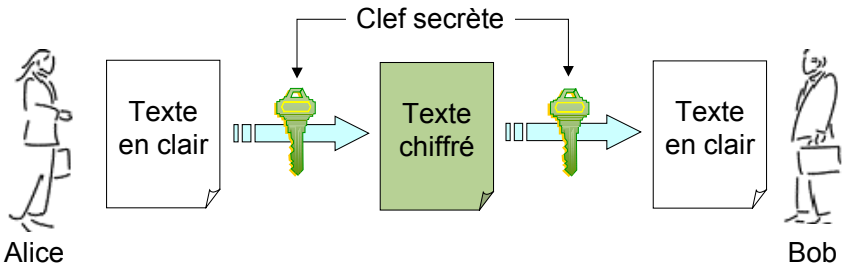


La **confidentialité** est historiquement le premier problème posé à la cryptographie. Il se résout par la notion de chiffrement, mentionnée plus haut.

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**.

Chiffrement symétrique - Principe

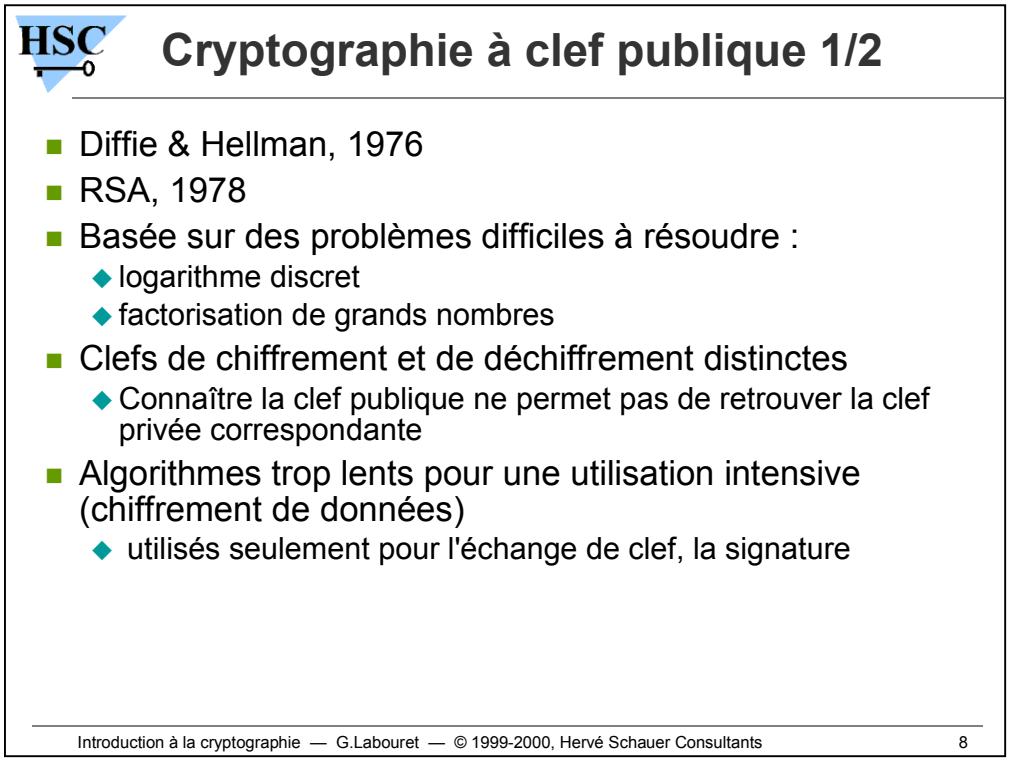
- Clef de chiffrement = clef de déchiffrement, elle doit rester secrète



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.

Chiffrement symétrique - Algorithmes

- Algorithmes de chiffrement en continu (*stream cipher*)
  - Agissent sur un bit à la fois
  - Le plus courant actuellement : RC4 (longueur de clef variable, généralement 128 bits)
- Algorithmes de chiffrement par blocs (*block cipher*)
  - Opèrent sur le texte en clair par blocs (généralement, 64 bits)
  - DES (clef de 56 bits codée sur 64)
  - 3DES
    - EDE : *Encrypt – Decrypt – Encrypt*
    - trois clefs distinctes (168 bits) ou seulement deux (112 bits)
  - IDEA (128 bits)
  - CAST-128 (128 bits)
  - Blowfish (longueur de clef variable, jusqu'à 448 bits)
  - AES (Rijndael, longueur de clef variable : 128, 192, 256)



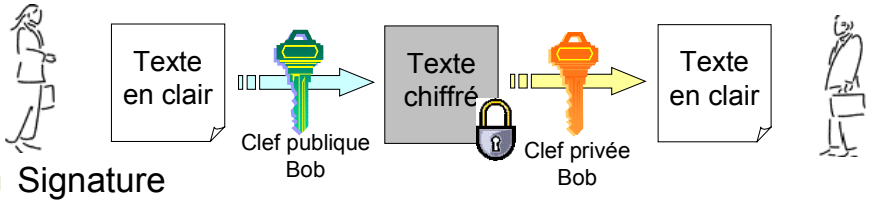
Avec les algorithmes asymétriques, **les clefs de chiffrement et de déchiffrement sont distinctes et ne peuvent se déduire l'une de l'autre**. On peut donc rendre l'une des deux publique tandis que l'autre reste privée. C'est pourquoi on parle de chiffrement à clef publique. Si la clef publique sert 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 a été 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.

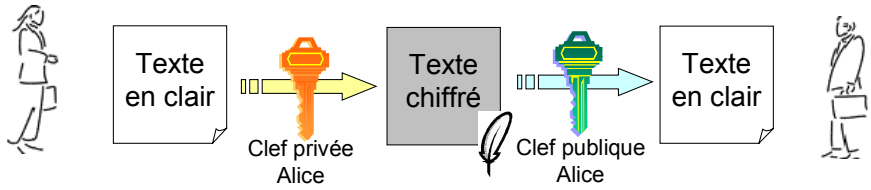
Tous les algorithmes actuels présentent l'inconvénient d'être **bien plus lents que les algorithmes à clef secrète** ; de ce fait, **ils sont souvent utilisés non pour chiffrer directement des données, mais pour chiffrer une clef de session secrète**. Certains algorithmes asymétriques ne sont adaptés 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.

Cryptographie à clef publique 2/2

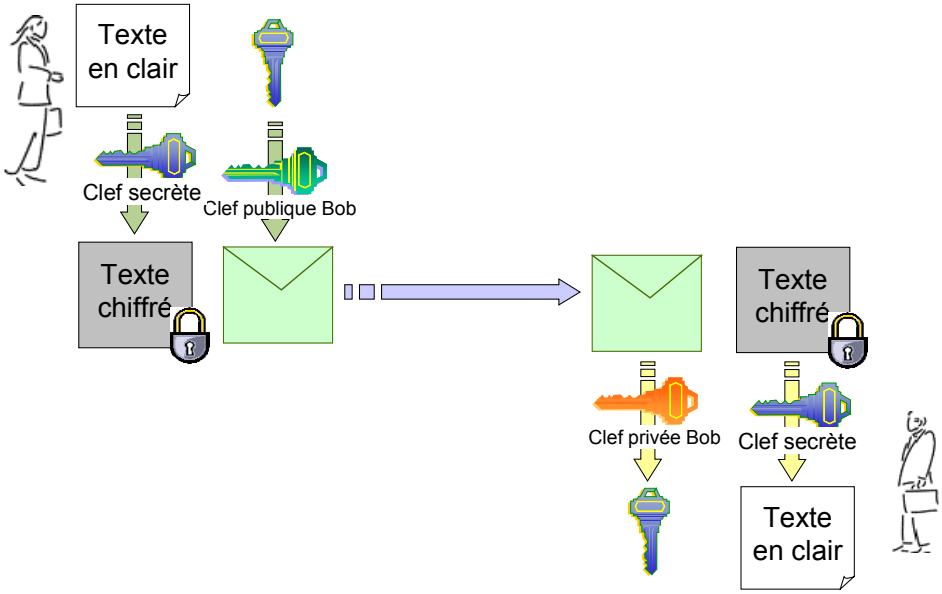
- Chiffrement
  - Clef publique utilisée pour le chiffrement, seul le détenteur de la clef privée peut déchiffrer



- Signature
  - Clef privée utilisée pour le chiffrement, seul son détenteur peut chiffrer, mais tout le monde peut déchiffrer (et donc en fait vérifier la "signature")



Exemple de combinaison clefs publiques / clefs secrètes

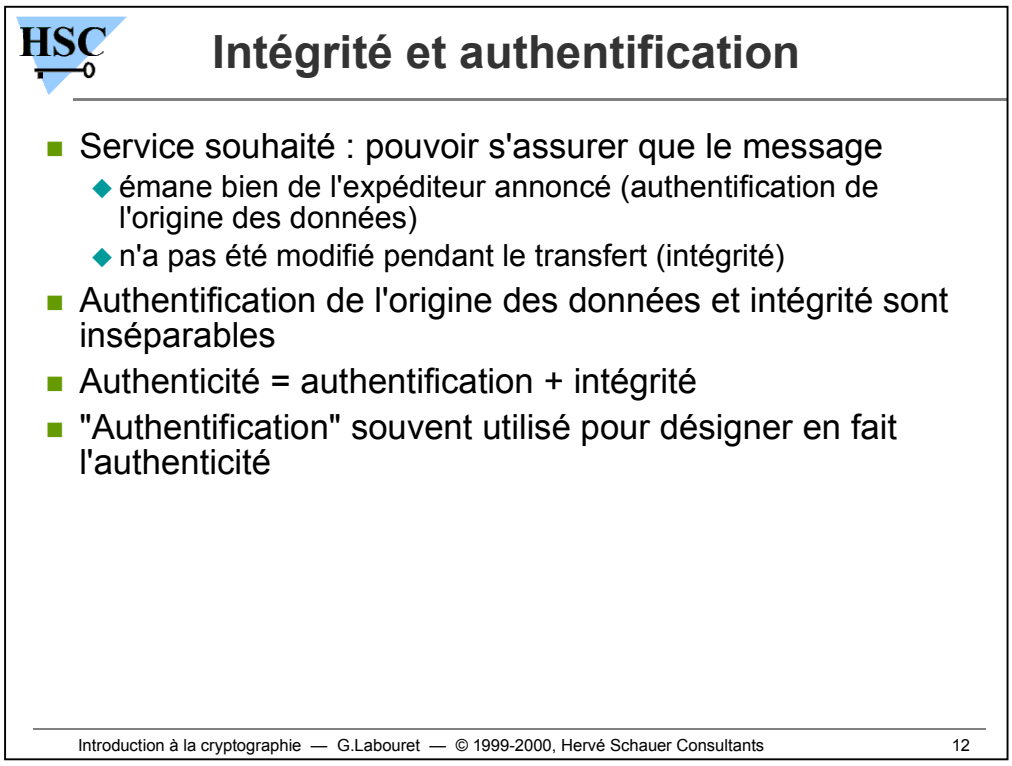


Longueurs de clefs

- Ne pas mélanger les longueurs de clefs publiques et secrètes
- Les algorithmes reposent sur des principes différents et utilisent donc comme clef des éléments présentant des caractéristiques (notamment longueur) différentes
- Cryptanalyse et comparaisons de résistance :
  - Pour les clefs secrètes, la référence est la recherche exhaustive, qui nécessite 2^{n-1} essais en moyenne
  - Pour les clefs publiques, l'attaquant doit résoudre le problème mathématique sur lequel repose l'algorithme
    - Factorisation (RSA) : coût sous-exponentiel, 1024 bits
    - Courbes elliptiques : 160 bits équivalent à RSA-1024

Fonctions de hachage, signature et scellement

- Mécanismes fournissant les services d'intégrité, d'authentification de l'origine des données et de non-répudiation de la source



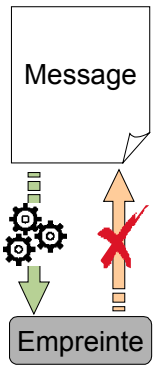
Parmi les problèmes auxquels s'attaque la cryptographie, on trouve l'authentification de l'origine des données et l'intégrité : lorsque l'on communique avec une autre personne au travers d'un canal peu sûr, on aimerait que **le destinataire puisse s'assurer que le message émane bien de l'auteur auquel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

Fonctions de hachage

- Fonction de hachage à sens unique
  - Convertit une chaîne de longueur quelconque en une chaîne de taille inférieure et généralement fixe = empreinte ou condensé
  - A sens unique :
    - Facile à calculer mais difficile à inverser
    - Il est difficile de trouver deux messages ayant la même empreinte
- MD5 (*Message Digest 5*)
  - Empreinte de 128 bits
- SHA (*Secure Hash Algorithm*)
  - Norme NIST
  - Empreinte de 160 bits
  - SHA-1 révision publiée en 1994 (corrige une faiblesse non publique)
  - SHA-1 considéré comme plus sûr que MD5
  - SHA-2 (octobre 2000) agrandit la taille de l'empreinte

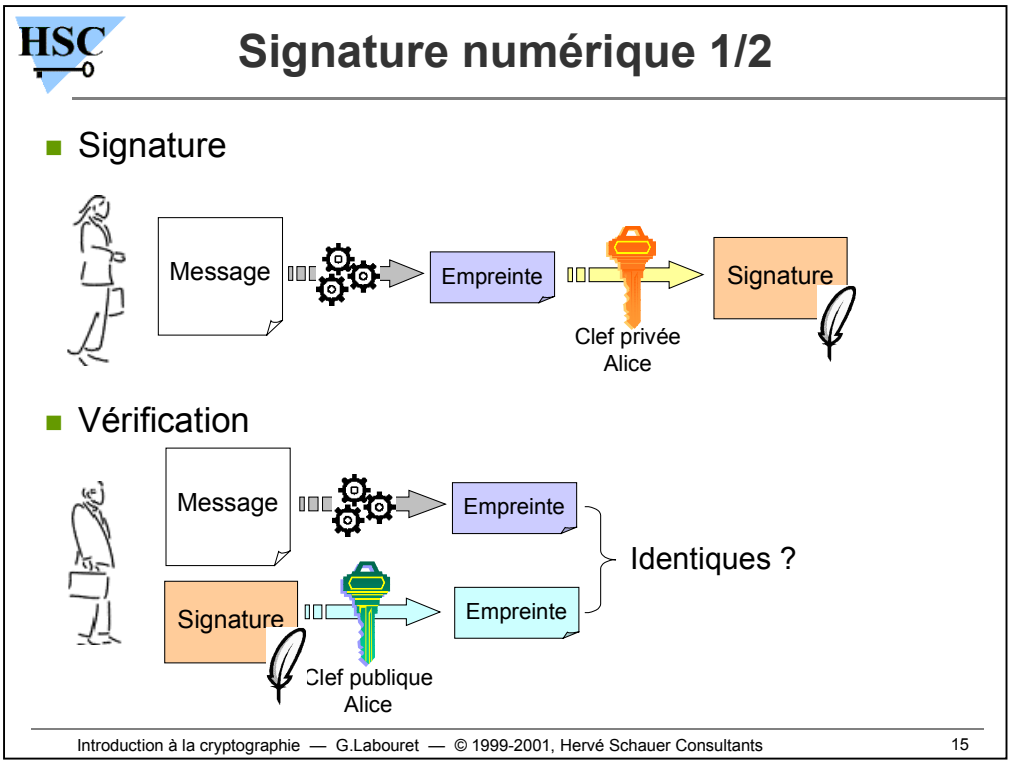


Aussi appelée fonction de condensation, une **fonction de hachage** est une fonction qui convertit une chaîne de longueur quelconque en une chaîne de taille inférieure et généralement fixe ; la chaîne résultante est appelée *empreinte* (*digest* en anglais) ou condensé de la chaîne initiale.

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.



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 transparent précédent), 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.

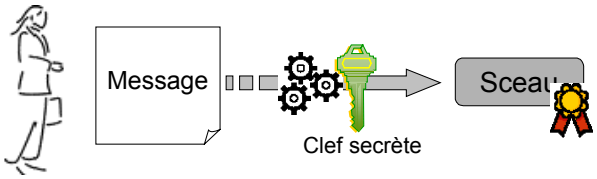
La méthode réellement utilisée pour signer consiste à **calculer une empreinte du message à signer et à ne chiffrer que cette empreinte**. Le calcul d’une empreinte par application d’une fonction de hachage étant rapide et la quantité de données à chiffrer étant fortement réduite, cette solution est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

Signature numérique 2/2

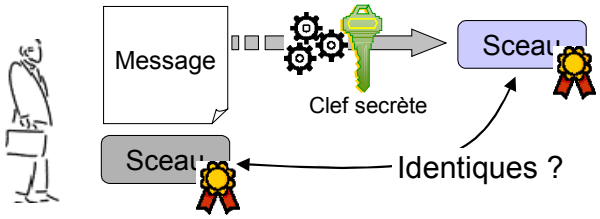
- Mécanisme qui fournit les services suivants :
  - Authentification de l'origine des données
  - Intégrité
  - Non-répudiation de la source
- Algorithmes
  - DSS : standard NIST
    - SHA-1 + El-Gamal
  - RSA : norme de fait
    - MD5 + RSA
    - SHA-1 + RSA

Scellement 1/2

- Scellement



- Vérification



Scellement (2/2)

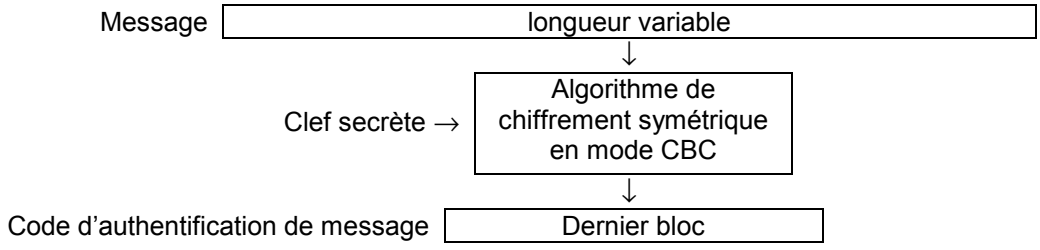
- Mécanisme qui fournit les services suivants :
  - Authentification de l'origine des données
  - Intégrité
- Vocabulaire
  - Sceau ou code d'authentification de message
  - *Message Authentication Code, MAC*
- 2 constructions possibles
  - Dernier bloc du cryptogramme obtenu avec un algorithme de chiffrement en mode CBC
    - DES-MAC
  - Fonction de hachage à sens unique utilisée avec une clef
    - Keyed-MAC (Keyed-MD5, Keyed-SHA-1)
      H(secret, message, secret)
    - HMAC (HMAC-MD5, HMAC-SHA-1)
      H(K⊕opad,H(K ⊕ipad,M))

Introduction à la cryptographie — G.Labouret — © 1999-2001, Hervé Schauer Consultants 18

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.



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

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

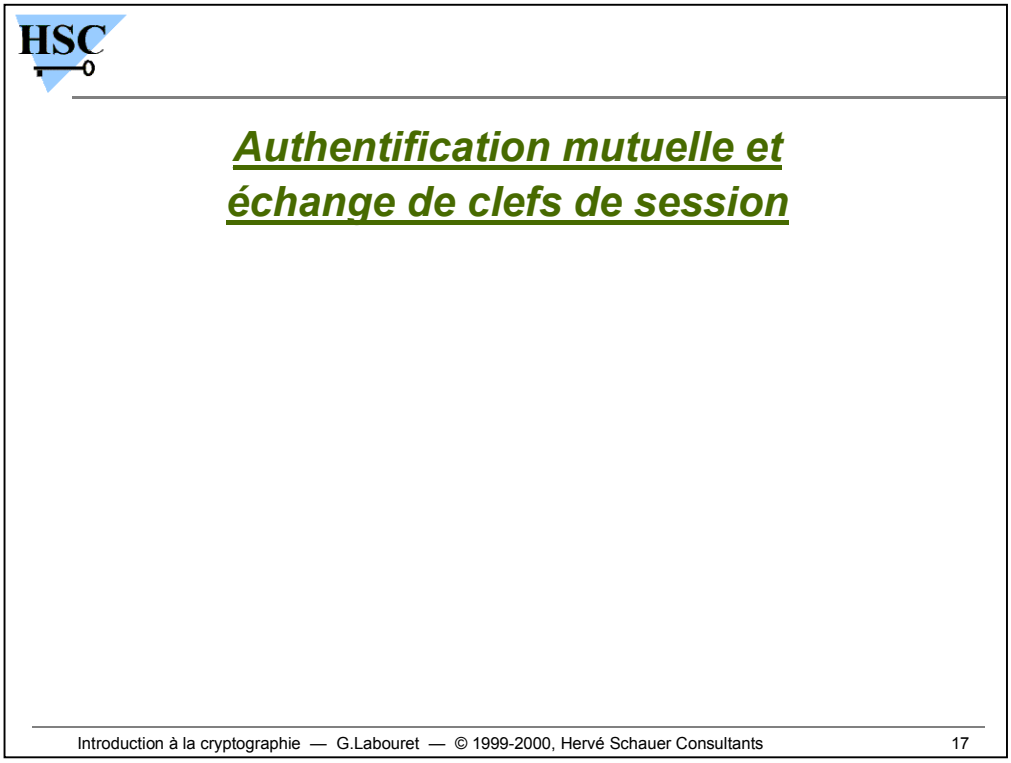
Un exemple simple de cette façon de procéder est de prendre pour MAC des valeurs du type H(secret, message), H(message, secret) ou H(secret, message, secret). Ces méthodes, présentées en 1992 par Gene Tsudik, s'appellent respectivement méthode du préfixe secret, du suffixe secret et de l'enveloppe secrète. 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 la 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-1 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).



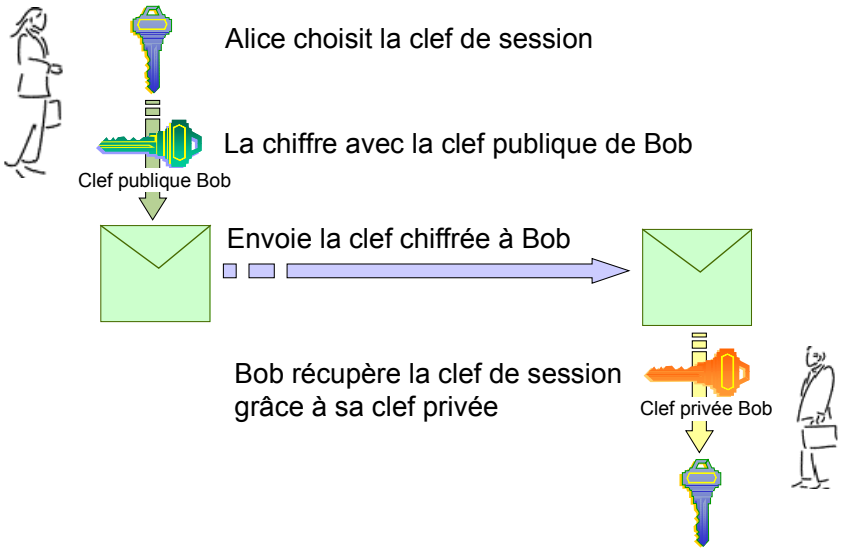
Tout comme les protocoles de communication, les protocoles cryptographiques sont une série d'étapes prédéfinies, basées sur un langage commun, qui permettent à plusieurs participants (généralement deux) d'accomplir une tâche. Dans le cas des protocoles cryptographiques, les tâches en question sont bien 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.

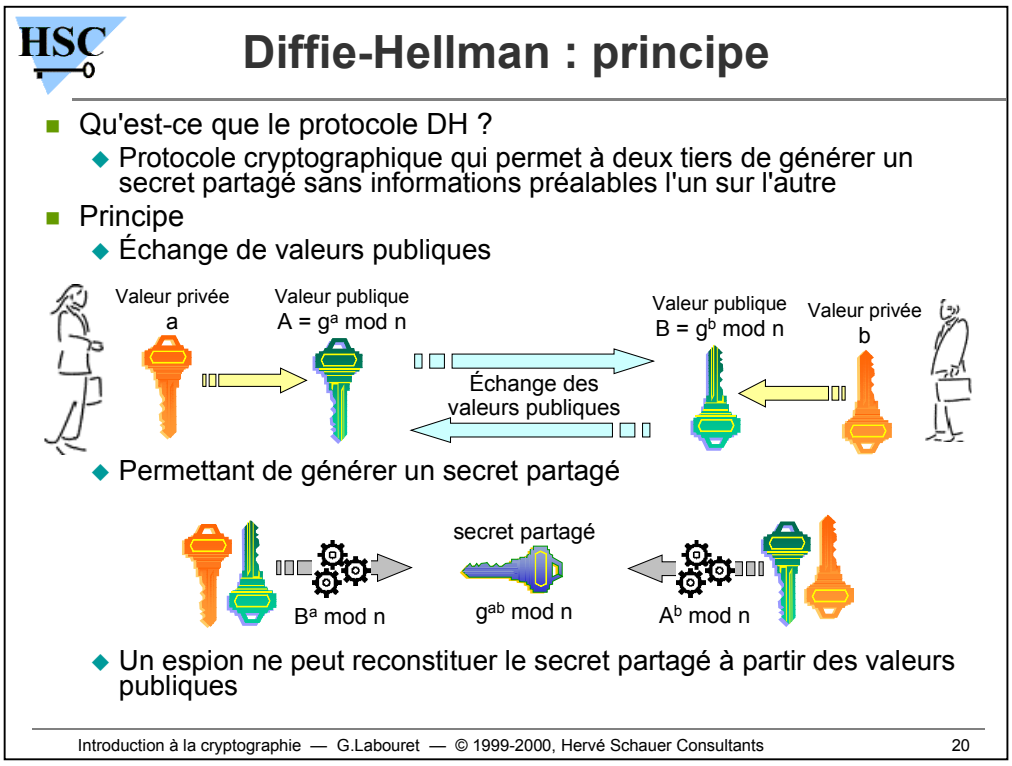
Introduction

- Relations entre échange de clefs et authentification mutuelle
  - L'échange de clefs doit être authentifié pour éviter les attaques
  - Une clef de session permet d'étendre l'authentification à l 'ensemble de la communication
  - Protocole d'authentification mutuelle avec échange de clefs
    - fournit authentification mutuelle et un échange de clefs authentifié tout-en-un
- Types d'échange de clefs
  - Transport
    - Exemple : transport RSA (utilisé par SSL)
  - Génération
    - Exemple : Diffie-Hellman

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.

Transport de clef





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

Une personne qui écoute la communication connaît g, n, $A=g^a \bmod n$ et $B=g^b \bmod n$, ce qui ne lui permet pas de calculer $g^{ab} \bmod n$: 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** .

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 les protocoles Photuris et IKE.

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.

Mais Diffie-Hellman, même authentifié, fournit une propriété intéressante, appelée propriété de ***Perfect Forward Secrecy*** **(PFS)**. Cette propriété 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).

L'utilisation de la cryptographie à clef publique à grande échelle nécessite de pouvoir gérer des listes importantes de clefs publiques, pour des entités souvent réparties dans un réseau. Pour cela, on a recourt à des infrastructures à clefs publiques (*Public Key Infrastructure*, PKI), systèmes de gestion des clefs publiques prévus pour une utilisation à grande échelle. La plupart de ces systèmes utilisent des certificats, mais ce n'est pas une obligation. Ainsi, DNSSEC, qui permet de distribuer des clefs publiques de façon authentifiée, peut être utilisé pour mettre en œuvre une PKI. En général, un amalgame est toutefois effectué entre PKI et système de gestion et de distribution de certificats.

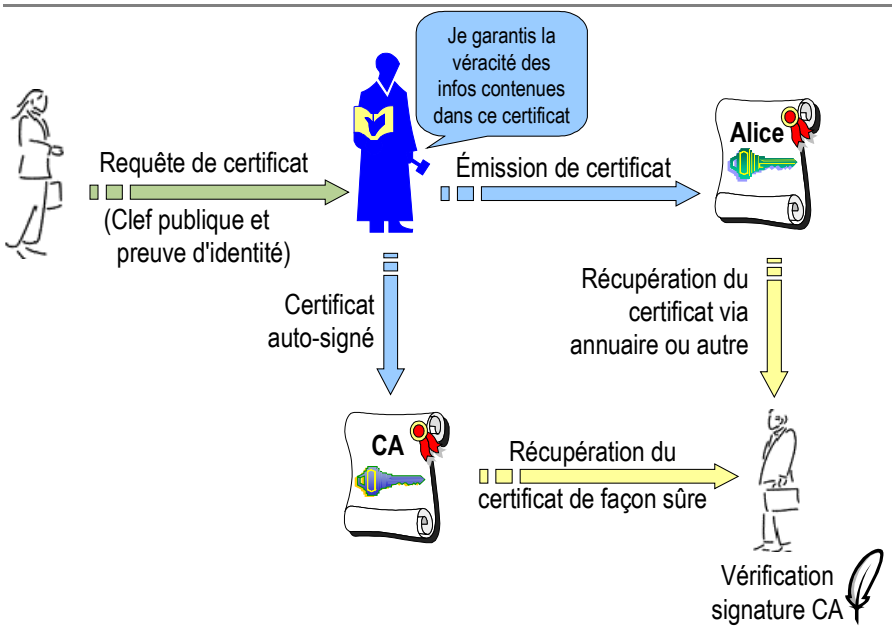
Distribution des clefs publiques

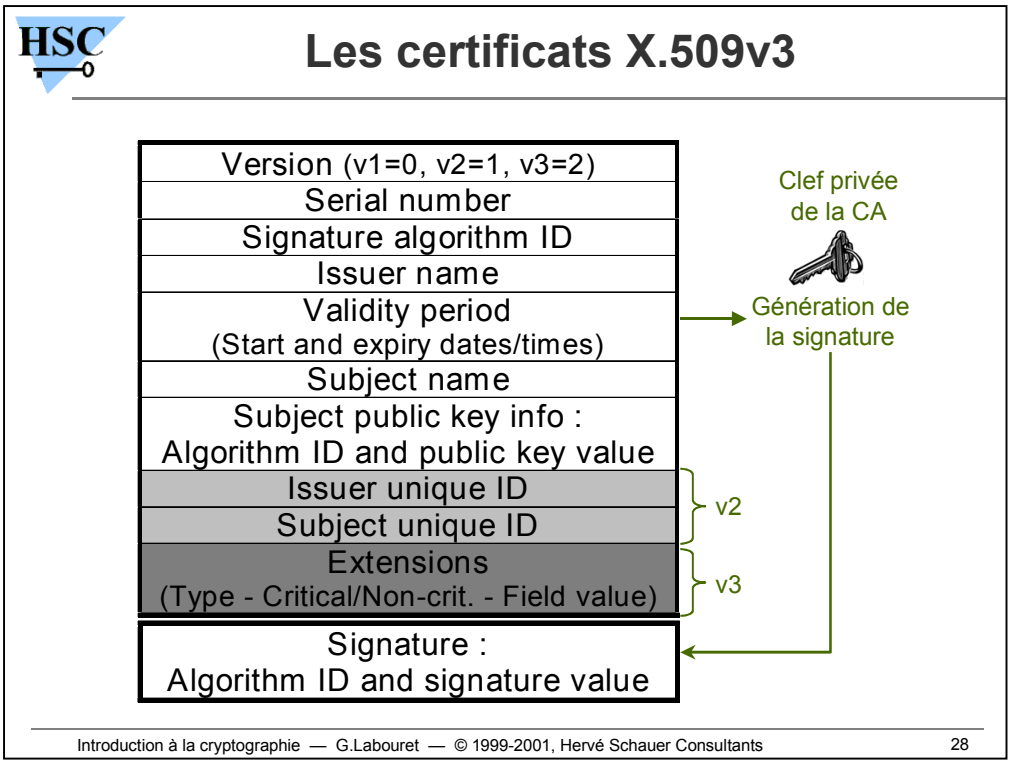
- Idée de départ
  - Simple annuaire des clefs publiques
- Problèmes à résoudre
  - Distribuer les clefs de façon authentifiée et intègre
  - Stocker les clefs de façon sûre (protection en intégrité)
    - Limiter le nombre de clefs à stocker
- Solution = certificats et hiérarchies de certification
  - Élément de transport d'une clef publique, dont l'authenticité est vérifiable de façon autonome
  - Authentification : Lie une clef publique à son possesseur
  - Intégrité : Toute modification du certificat sera détectée

Principe du certificat

- Certificat = Structure de données
  - Permet de lier une clef publique à différents éléments, au moyen de la signature d'une autorité de confiance :
    - Nom du propriétaire de la clef
    - Dates de validité
    - Type d'utilisation autorisée
    - …
  - Format actuel : X.509v3, profil PKIX
- Émis par une autorité de certification (*Certificate Authority* – CA)
  - Garantit l'exactitude des données
  - Certificats vérifiables au moyen de la clef publique de la CA, seule clef à stocker de façon sûre
- Listes de révocation (*Certificate Revocation List* – CRL)
  - Permettent de révoquer des certificats avant leur expiration normale

Émission et vérification des certificats





Version : Indique à quelle version de X.509 correspond ce certificat.
Serial number : Numéro de série du certificat (propre à chaque autorité de certification).
Signature Algorithm ID : Identifiant du type de signature utilisée.
Issuer Name : *Distinguished Name (DN)* de l'autorité de certification qui a émis ce certificat.
Validity period : Période de validité.
Subject Name : *Distinguished Name (DN)* du détenteur de la clef publique.
Subject public key info : Infos sur la clef publique de ce certificat.
Issuer Unique ID / Subject Unique ID : Extensions optionnelles introduites avec la version 2 de X.509.
Extensions : Extensions génériques optionnelles, introduites avec la version 3 de X.509.
Signature : Signature numérique de la CA sur l'ensemble des champs précédents.

Exemple de certificat – client SSL (1/2)

Version: 3 (0x2)
Serial Number:
    5c:e4:85:54:9e:84:f0:59:90:fd:08:7a:62:93:6d:36
Signature Algorithm: md5WithRSAEncryption
Issuer: L=Internet, O=VeriSign, Inc., OU=VeriSign OnSite Admin Demo
Validity
    Not Before: Nov 8 00:00:00 2000 GMT
    Not After : Jan 7 23:59:59 2001 GMT
Subject: O=HSC, OU=www.verisign.com/repository/CPS Incorp. By Ref.,LIAB.LTD(c)96,
    CN=Ghislaine Labouret/Email=ghislaine.labouret@hsc.fr
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
        Modulus (1024 bit):
            00:bd:da:6b:f7:7d:6c:4e:cb:00:9f:61:56:4c:7d:
            …
        Exponent: 65537 (0x10001)

Exemple de certificat – client SSL (2/2)

X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    X509v3 Certificate Policies:
        Policy: 2.16.840.1.113733.1.7.1.1
            CPS: https://www.verisign.com/CPS
            User Notice:
              Organization: VeriSign, Inc.
              Number: 1
              Explicit Text: VeriSign's CPS incorp. by reference liab. ltd. (c)97 VeriSign
    Netscape Cert Type:
        SSL Client
    2.16.840.1.113733.1.6.11:
        . 2f5941537bae90da19e157922c4e3992
    2.16.840.1.113733.1.6.13:
      ....
Signature Algorithm: md5WithRSAEncryption
    85:5a:83:d3:4c:a8:f4:59:69:10:f0:4f:f2:94:87:19:ab:cd:
    …

Exemple de certificat – IPsec

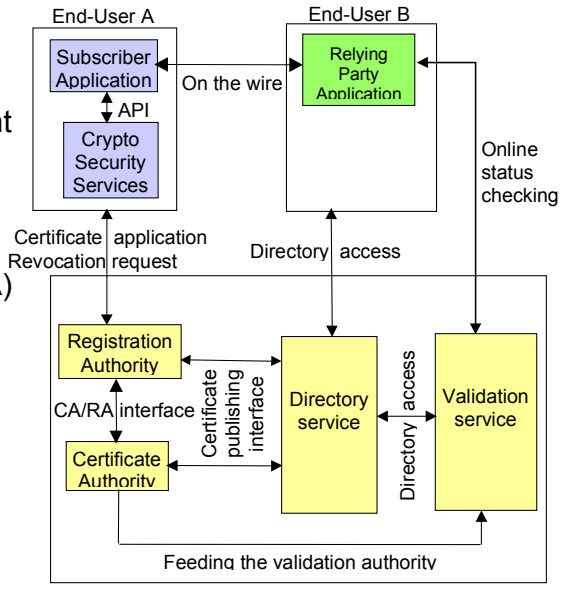
```
Version: 3 (0x2)
Serial Number: 20 (0x14)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=FR, ST=Ile-de-France, L=Levallois-Perret, O=Herv\xE9 Schauer Consultants (HSC),
    OU=Certificate Authority, CN=HSC CA/Email=ca@hsc.fr
Validity
   Not Before: Oct 27 07:29:34 2000 GMT
   Not After : Nov 11 07:29:34 2000 GMT
Subject: C=FR, ST=Ile de France, L=Levallois-Perret, O=HSC, OU=IPsec 2000 testbed,
    CN=kame.ipsec2000.fr
Subject Public Key Info:
   Public Key Algorithm: rsaEncryption
   RSA Public Key: (1024 bit)
      Modulus (1024 bit):
         00:f8:51:89:b7:0c:33:56:74:a5:28:98:ed:60:6c:
         ...
      Exponent: 65537 (0x10001)
X509v3 extensions:
   X509v3 Key Usage: critical
      Digital Signature, Non Repudiation
   X509v3 Subject Alternative Name:
      email:kame@ipsec2000.fr, IP Address:192.168.1.60
Signature Algorithm: md5WithRSAEncryption
   3d:77:f0:f9:9f:9c:6a:f7:1a:ee:2c:bd:0f:57:a2:23:93:35:
   ...
```


Qu'est-ce qu'une PKI ?

- PKI = Public Key Infrastructure = infrastructure à clefs publiques
  - Nature: infrastructure, ensemble d'éléments, protocoles et services
  - Rôle: gestion des clefs publiques à grande échelle
    - Enregistrement et émission
    - Stockage et distribution
    - révocation et vérification de statut
    - Sauvegardes et récupération
  - La plupart des PKI actuelles utilisent des certificats

Composants d'une PKI (1/2)

- Éléments
  - Utilisateurs finaux (possesseurs de clefs)
  - Autorités d'enregistrement (*Registration Authority* – RA, *Local Registration Authorities* – LRAs)
  - Autorité de certification (*Certificate Authority* - CA)
    - Émetteur de certificats
    - Émetteur de listes de révocations
  - Annuaire
  - Service de validation
- Hiérarchie
  - Plusieurs niveaux de certification



Composants d'une PKI (2/2)

- Outils
  - Protocoles de gestion et de communication
    - CMP, PKCS, CRMF, OCSP, SCVP, LDAP…
    - Beaucoup de "standards", certains non complétés, instables ou non encore implémentés
  - APIs cryptographiques
  - Personnel
- Règles
  - Certification Practice Statement (CPS)
    - Définition initiale : déclaration des pratiques utilisées par une autorité de certification pour émettre des certificats
    - Actuellement généralement étendu à l'ensemble de l'infrastructure
  - Certificate Policies (CPs)
    - Règle sous laquelle le certificat a été émis et types d'utilisations autorisées
    - Vérifiées par les utilisateurs finaux (applications)
  - Contrats et autres documents légaux (garanties et limitations de responsabilité)

Législation française en matière de cryptologie

Vue d'ensemble

- Textes législatifs et réglementaires
  - Loi n°90-1170 du 29 décembre 1990, modifiée par la loi n°96-659 du 26 juillet 1996 (décrets d'application début 1998)
  - Deux nouveaux décrets et un arrêté le 17 mars 1999
- La DCSSI (ex-SCSSI)
  - Direction Centrale de la Sécurité des Systèmes d'Information
  - C'est le service à qui adresser les déclarations et demandes d'autorisations
- Principe actuel
  - On distingue différents régimes en fonction de la finalité des moyens ou prestations de cryptologie et du niveau correspondant

Finalités

- Utilisation
  - personnelle
  - collective (pour un domaine ou un type d'utilisateur donné)
  - générale (potentiellement n'importe qui)
- Fourniture
  - vente, location ou fourniture gratuite.
- Importation
  - Hors Communauté Européenne
- Exportation
  - Hors Communauté Européenne

Régimes

- Dispense de toute formalité préalable.
  - La liberté est totale d'utiliser, de fournir, d'importer ou d'exporter le moyen ou la prestation considérés.
- Déclaration
  - Simplifiée ou non
  - Un formulaire administratif doit être transmis à la DCSSI un mois à l'avance
- Autorisation
  - Partie administrative et partie technique
  - La DCSSI dispose de quatre mois pour notifier sa décision. Une absence de notification à l'expiration de ce délai de quatre mois vaut autorisation.

Synthèse

Synthèse du nouveau cadre législatif et réglementaire				
Finalités	**Fonctions offertes**			
	Authentification, signature, intégrité, non répudiation	Confidentialité		
		$L \leq 40$ bits	40 bits < $L \leq$ 128 bits	$L >$ 128 bits
Utilisation	Libre	Libre	Libre ou Déclaration (1)	Autorisation
Fourniture	Déclaration simplifiée	Déclaration	Déclaration	Autorisation
Importation	Libre	Libre	Libre ou Déclaration (1)	Autorisation
Exportation	Libre	Autorisation	Autorisation	Autorisation

(1) La déclaration est nécessaire uniquement pour un matériel ou un logiciel qui n'a pas fait l'objet préalablement d'une déclaration par leur producteur, un fournisseur ou un importateur, et si ledit matériel ou ledit logiciel n'est pas exclusivement destiné à l'usage privé d'une personne physique.

Liens

- www.scssi.gouv.fr
- http://www.internet.gouv.fr/francais/commerce/textesref.htm
  - Copie des principaux textes législatifs et réglementaires
  - Guide pratique (pas à jour mais très clair)
    http://www.internet.gouv.fr/francais/textesref/guidepratique.htm
- http://cwis.kub.nl/~frw/people/koops/lawsurvy.htm
  - Informations sur l'ensemble des lois à travers le monde

Pour en savoir plus...

Lectures

- RSA FAQ
  - RSA Laboratories, *Frequently Asked Questions About Today's Cryptography - Version 4.1*, 2000.
  - Disponible en ligne : http://www.rsasecurity.com/rsalabs/faq/
- Cryptographie appliquée / *Applied Cryptography*
  - Schneier Bruce, *Cryptographie appliquée - Algorithmes, protocoles et codes source en C - 2ème édition*, International Thomson Publishing France, 1997.
  - V.F. disponible dans de nombreuses librairies
  - V.O. disponible chez Amazon ou en PDF sur CD-ROM chez Dr. Dobb's (www.ddj.com)
- *Handbook of Applied Cryptography*
  - Menezes Alfred J., van Oorschot Paul C., Vanstone Scott A., *Handbook of Applied Cryptography*, CRC Press LLC, *Fourth Printing*, July 1999.
  - Disponible en ligne : http://cacr.math.uwaterloo.ca/hac/

Sites

- Counterpane Labs - www.counterpane.com/labs.html
  - Publications de Counterpane Labs (Bruce Schneier)
  - Index d'articles disponibles en ligne
- Cryptography Research - www.cryptography.com
  - Publications de Cryptography Research
  - Index des articles des conférences Crypto et Eurocrypt disponibles en ligne
  - Liens vers les sites de cryptologues
- Cryptome - jya.com/crypto.htm
  - Les nouvelles (brutes)
- Groupes de discussion
  - fr.misc.cryptologie (beaucoup de bruit pour rien)
  - sci.crypt