



HERVÉ SCHAUER CONSULTANTS

Cabinet de Consultants en Sécurité Informatique depuis 1989

Spécialisé sur Unix, Windows, TCP/IP et Internet

Fonctionnement des PKI

Franck Davy

<Franck.Davy@hsc.fr>

Fonctionnement des PKI

- 1 -

Bases de cryptographie

- × Introduction
- × Notions de base
- × Algorithmes
 - × à clef secrète
 - × à clef publique
- × Fonctions de hachage
- × Mécanismes
 - × Chiffrement
 - × Signature
- × En pratique
 - × Boîte à outils
 - × Messagerie électronique sécurisée avec S/MIME

Introduction

Vocabulaire et notions de base



Vocabulaire

- × Services de sécurité
 - × Confidentialité
 - × Intégrité
 - × Authentification
 - × Non répudiation
- × Mécanismes
 - × Mécanismes mettant en oeuvre les services précédents, fondés sur les algorithmes cryptographiques
 - × Chiffrement
 - × Signature
 - × Chiffrement et signature étroitement liés (chiffrement d'un condensât)
 - × Scellement
- × Algorithmes
 - × Algorithmes symétriques, à clef secrète
 - × Algorithmes asymétriques, à clef publique



Algorithmes à clef secrète



Algorithmes à clef secrète (1/4)

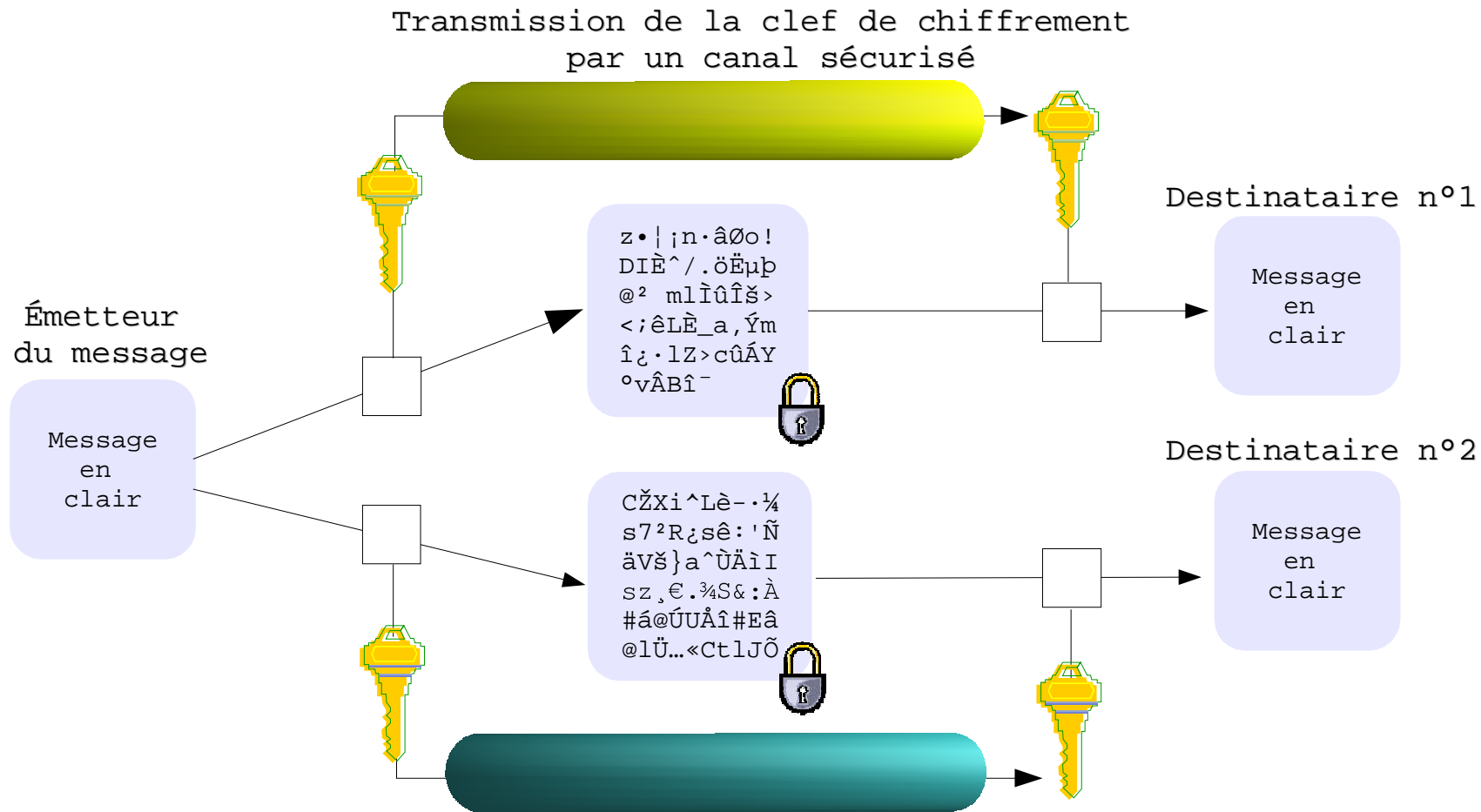
Algorithmes dits « symétriques »

Caractéristiques

- × Opérations de chiffrement et déchiffrement identiques
 - × Clef identique pour les opérations de chiffrement et déchiffrement
- × Utilisation pour le chiffrement de données
- × Rapidité, et facilité de mise en oeuvre sur des circuits « bon marché »
- × Problème de la *distribution* des clefs
 1. *Transport sécurisé* de la clef de chiffrement ?
 - × Problématique de l'échange de clef
 2. *Nombre de clefs échangées* (en n^2)



Schéma de principe





Algorithmes à clef secrète (3/4)

Algorithmes de chiffrement en continu (stream Cipher)

- × Chiffrement bit à bit
- × Exemple : RC4 (RSA Security)
 - × Taille de clef variable (128 bits en pratique)

Algorithmes de chiffrement par blocs (Block Cipher)

- × Chiffrement par blocs de texte clair
 - × 64 bits (DES), 128 bits (AES)
- × Modes ECB, CBC, CFB, OFB
- × DES (56 bits), 3DES (clef de 168 bits en EDE3, 112 bits en EDE)
- × RC2 (128 bits), Blowfish (128bits, jusqu'à 448 bits), AES (128, 192, 256 bits), IDEA (128 bits, brevet ASCOM en Europe et USA pour un usage commercial)





Algorithmes à clef secrète (4/4)

Principal usage

- × Service de confidentialité
 - × Subsiste le problème de la distribution des clefs de chiffrement
- × Peut également assurer un service d'authentification
 - × Sans service de non répudiation
 - × L'utilisateur n'est pas le seul à pouvoir produire la signature !
- × Exemple de la Monétique :
 - × Cybercomm (système d'authentification dynamique)
 - × Clef secrète dépendante du porteur (carte à puce)
 - × Chiffrement (DES 56 bits) des paramètres suivants :
 - × Montant de la transaction
 - × Identifiant du commerçant
 - × Date (contre le rejeu)
 - × Authentification au niveau des DAB
 - × par la piste magnétique



Algorithmes à clef publique



Algorithmes à clef publique (1/16)

Algorithmes dits « asymétriques »

Limitation des algorithmes à clef secrète :

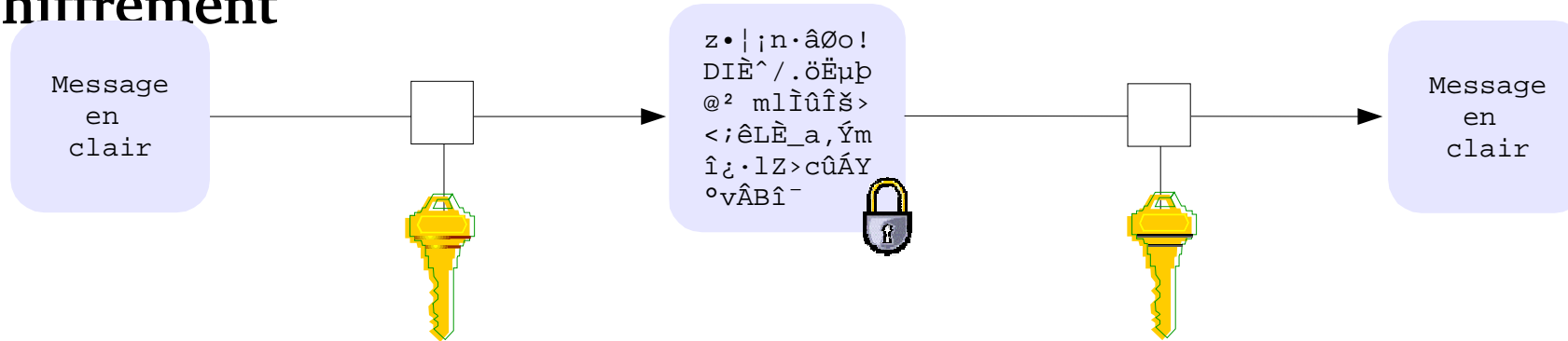
- × Problématique de l'échange de la clef de chiffrement
 - × Établissement préalable d'un canal sûr pour la transmission de la clef

Systemes à clef publique

- × Clef de chiffrement e , rendue publique
- × Clef de déchiffrement d , gardée secrète
 - × d non déductible modulo la connaissance de e
 - × Par exemple : utilisation d'une fonction à sens unique : $f(x) = x^e [n]$
 - × ...sous certaines conditions
- × Chiffrement d'un message à l'aide de la clef publique e
 - × Pas de communication préalable entre les tiers communicants



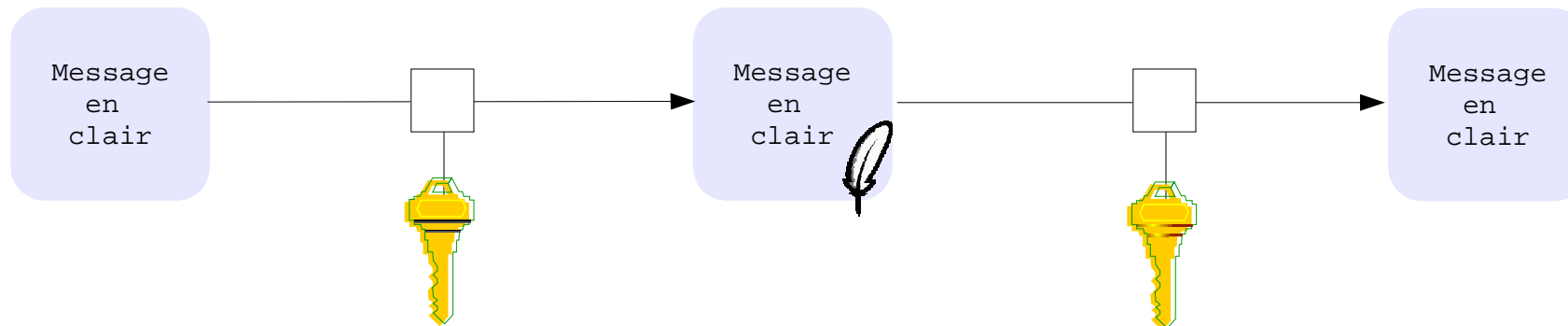
Chiffrement



Chiffrement par l'émetteur avec la clef publique du destinataire

Déchiffrement par le destinataire avec sa clef privée

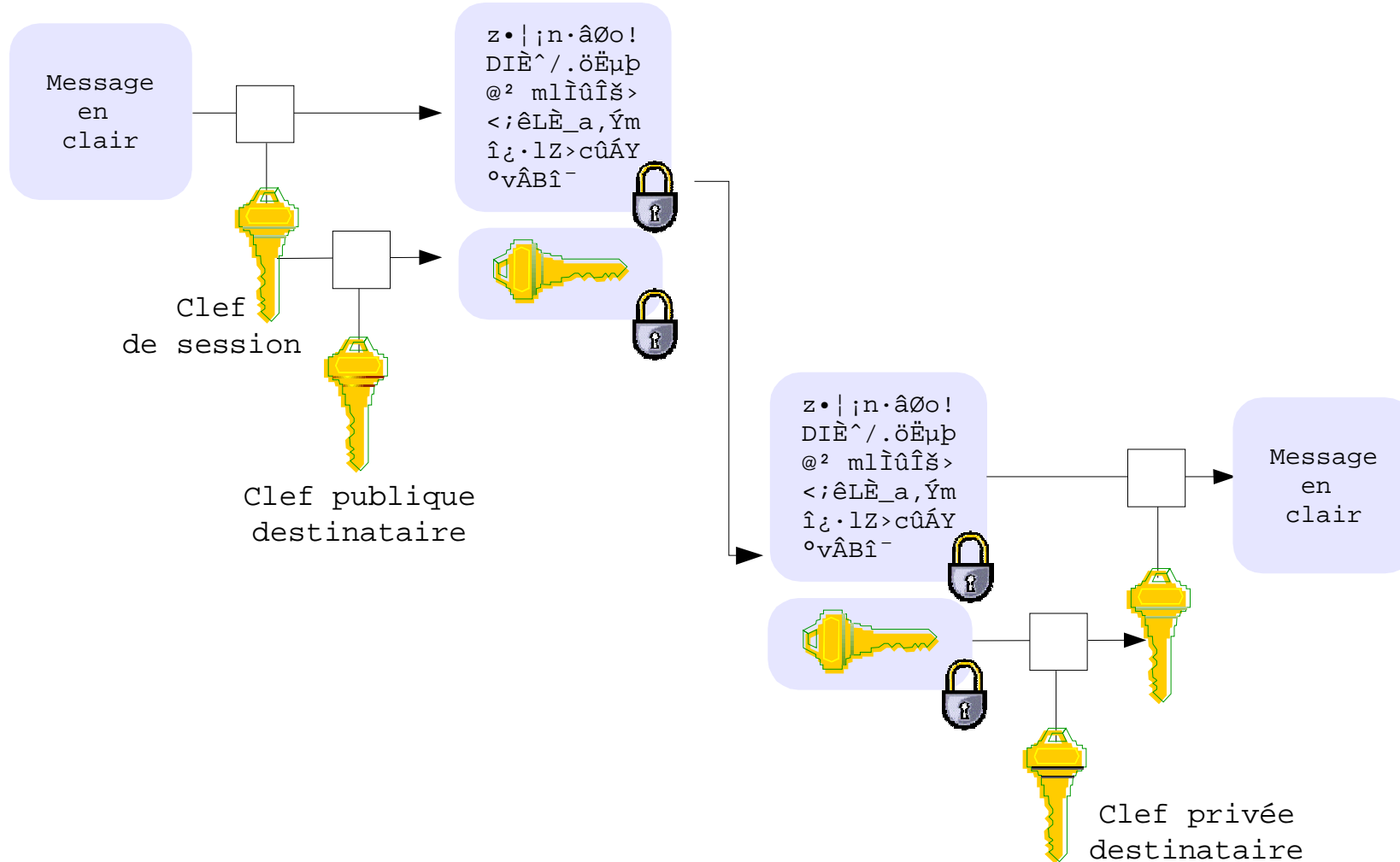
Signature



Chiffrement par l'émetteur avec sa clef privée

Déchiffrement par les destinataires avec la clef publique de l'émetteur

Transport de clef de session





Algorithmes à clef publique (4/16)

Algorithme RSA

- × Publié par Rivest, Shamir et Adelman en 1977
- × Fondé sur la difficulté de la factorisation des grands nombres
 - × Sécurité dite « calculatoire » : système *inconditionnellement sûr*

Schéma

1. On *choisit* p, q deux « grands » nombres premiers
2. On *calcule* $n = p \cdot q$
3. Un entier e est *choisi* tel que premier avec $(p-1)$ et $(q-1)$
4. L'entier d tel que $ed \equiv 1 \pmod{(p-1)(q-1)}$ est *calculé*, avec l'algorithme d'Euclide
 - le couple d'entier (n, e) représente la clef publique
 - L'entier d représente la clef privée





Algorithmes à clef publique (5/16)

Chiffrement RSA

- × Soit le message M , M' est le message chiffré obtenu par exponentiation modulaire
- × $M' = M^e \pmod{n}$
- × Pour le déchiffrement, on calcule $M'^d \pmod{n}$
- × $M'^d = M \pmod{n}$, car $ed = 1 \pmod{(p-1)(q-1)}$ par construction

Signature RSA

- × Opération de chiffrement, en inversant les rôles de e et d
 - × Le message émis est le couple $(M, M'^d \pmod{n})$
 - × Tout le monde peut vérifier la signature par possession de la clef publique
 - × Service de non répudiation
- Sécurité *calculatoire* : repose sur la difficulté de factoriser n





Algorithmes à clef publique (6/16)

Clef RSA principalement sous 2 formats :

- × Format binaire : encodage de la structure ASN.1 suivant la syntaxe de transfert distinctive DER (sans ambiguïté)
- × Format ASCII imprimable : objet binaire précédent encodé au format PEM
- × Possibilité de conserver la clef privée RSA chiffrée
 - × DES, 3DES ou IDEA (en mode CBC) par exemple
 - Informations concernant les algorithmes dans les en-têtes PEM (IV, notamment)
- × Format PEM « *Privacy Enhanced-Mail* » (RFC1421-1424)

1. Executive Summary

This document defines message encryption and authentication procedures, in order to provide privacy-enhanced mail (PEM) services for electronic mail transfer in the Internet. It is intended to become one member of a related set of four RFCs. The procedures defined in the current document are intended to be compatible with a wide range of key management approaches, including both symmetric (secret-key) and asymmetric (public-key) approaches for encryption of data encrypting keys.





Algorithmes à clef publique (7/16)

- × Format général d'un message au format PEM

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----  
Proc-Type: 4, ENCRYPTED  
Content-Domain: RFC822  
DEK-Info: DES-CBC, F8143EDE5960C597  
Originator-ID-Symmetric: linn@zendia.enet.dec.com,  
Recipient-ID-Symmetric: linn@zendia.enet.dec.com,ptf-kmc,3  
Key-Info: DES-ECB,RSA-MD2,9FD3AAD2F2691B9A,B70665BB9BF7CBCDA60195DB94F727D3  
Recipient-ID-Symmetric: pem-dev@tis.com,ptf-kmc,4  
Key-Info: DES-ECB,RSA-MD2,161A3F75DC82EF26,E2EF532C65CBCFF79F83  
A2658132DB47LlrHB0eJzyhP+/fSStdW8okeEnv47jxe7SJ/iN72ohNcUk2jHEU  
SoH1nvNSIWL9MdXd/H5LMDWnonJvPCwOUHt==  
-----END PRIVACY-ENHANCED MESSAGE-----
```

- × Messages chiffrés « *PEM Part III : Algorithms, Modes, and Identifiers* » (RFC1423)
- × Exemple : Chiffrement DES-CBC
 - × En-tête *Proc-Type*
 - × Identifiant de version de la RFC (4), ENCRYPTED (service de sécurité mis en oeuvre)
 - × En-tête *DEK-Info* :
 - × Identifiant d'algorithme (DES-CBC), vecteur d'initialisation (F8143EDE5960C597)





Algorithmes à clef publique (8/16)

- * « RSA Cryptography Specifications » (RFC2437)
 - * Recommandations sur la mise en oeuvre d'un cryptosystème fondé sur RSA
 - * <http://www.ietf.org/rfc/rfc2437.txt>
 - * Anciennement : PKCS#1 « *RSA Cryptography Standard* »
 - * Structure ASN.1

- * Clef publique RSA (« *11.1.1 Public-key syntax* »)

```
RSAPublicKey ::= SEQUENCE {  
    modulus INTEGER, -- n  
    publicExponent INTEGER -- e }
```

- * Clef privée RSA (« *11.1.2 Private-key syntax* »)

```
RSAPrivateKey ::= SEQUENCE {  
    version Version,  
    modulus INTEGER, & -- n  
    publicExponent INTEGER, -- e  
    privateExponent INTEGER, -- d  
    prime1 INTEGER, -- p  
    prime2 INTEGER, -- q  
    exponent1 INTEGER, -- d mod (p-1)  
    exponent2 INTEGER, -- d mod (q-1)  
    coefficient INTEGER -- (inverse of q) mod p }
```

- Clef RSA obtenue par encodage DER de la structure ASN.1 (fichier binaire)





Algorithmes à clef publique (9/16)

- × Exemple de génération d'une clef privée RSA
- × Modulus $n = 512$ bits, exposant public $e = 3$

```
$ openssl genrsa -3 512 | tee rsa.PEM
Generating RSA private key, 512 bit long modulus
e is 3 (0x3)
-----BEGIN RSA PRIVATE KEY-----
MIIBOQIBAAJBALGWJEoGAC6fYqpG9vCx1LEcjFtrhw2WDZ71Ne+Q2ZvgbqDE9j8S
IsjEStKc2UZ/w2bnPc8k+3LTTGCTQ9SAdsmCAwEAAQJAT9VlXvmvn0X7vvzECpSH
P0/MMVT3k73/RZSKuLXvfcj5Lsmce/rMWQkNxC2a5G32UzNvDbOrRIW103QCXZNh
kQIhAOGrWfk4bnjjejGSGwVVAQ3ynTyOIrf179iEnKcZKv71AiEAYXRk9uUa+veK
uL1Oad2kONdZRQdb0sNuOHepinz3HIcCIFTmcmMgp+8zJbWgkinfYRYuQJmXn9gv
2ZxLx+PVxCdBAiAaD4KYJd8tpCQ/7c1dCJ0b9VMziQYp57o0d9Zo4e2dtQIgbxdY
Fsf591Z/Q9sZkoxgUMpuSxBBC8qxWyUGHcEF+DY=
-----END RSA PRIVATE KEY-----
```

- × Clef privée RSA au format PEM (sans chiffrement)
 - × Structure ASN.1 encodée suivant la syntaxe de transfert DER (Distinguished Encoding Rules)
 - La suite d'octets (OCTET STRING) obtenue est encodée en base 64





Algorithmes à clef publique (10/16)

- × Comparaison des formats PEM et DER

- × Conversion avec rsa(1ssl)

```
$ openssl rsa -inform PEM -in ./rsa.PEM -outform DER ./rsa.DER
$ cat ./rsa.DER
0b^G}»Öf³4n
©oî£/)W!×6_é ŽÃ^á      ä;í f Üç Åx<Ü Ñ2Ä TðÔ-    >„ê±<iDKpA7Am
```

- × Encodage en base64 du fichier binaire avec enc(1ssl)

```
$ cat ./rsa.DER | openssl base64 -e
MIIBOQIBAAJBALGWJEoGAC6fYqpG9vCx1LEcjFtrhw2WDZ71Ne+Q2ZvgbqDE9j8S
IsjEStKc2UZ/w2bnPc8k+3LTTGCTQ9SAdsmCAwEAAQJAT9VlXvmvn0X7vvzECpSH
P0/MMVT3k73/RZSKuLXvfcj5Lsmce/rMWQkNxC2a5G32UzNvDbOrRIW103QCXZNh
kQIhAOGGrWfk4bnjjejGSGwVVAQ3ynTyOIrf179iEnKcZKv71AiEAYXRk9uUa+veK
uL1Oad2kONdZrQdb0sNu0Hepinz3HIcCIFtmcMgp+8zJbWgkinfYRYuQJmXn9gv
2ZxLx+PVxCdBAiAaD4KYJd8tpCQ/7c1dCJ0b9VMziQYp57o0d9Zo4e2dtQIgbxdY
Fsf591Z/Q9sZkoxgUMpuSxBBC8qxWyUGHcEF+DY=
```





Algorithmes à clef publique (11/16)

Structure ASN.1

```
$ dumpasn1 ./rsa.DER
0 30 313: SEQUENCE {
  4 02 1: INTEGER 0
  7 02 65: INTEGER
    : 00 B1 96 24 4A 06 00 2E 9F 62 AA 46 F6 F0 B1 D4
    : B1 1C 8C 5B 6B 87 0D 96 0D 9E F5 35 EF 90 D9 9B
    : E0 6E A0 C4 F6 3F 12 22 C8 C4 4A D2 9C D9 46 7F
    : C3 66 E7 3D CF 24 FB 72 D3 4C 60 93 43 D4 80 76
    : C3
  74 02 3: INTEGER 65537
  79 02 64: INTEGER [...]
  145 02 33: INTEGER [...]
  180 02 33: INTEGER [...]
  215 02 32: INTEGER [...]
  249 02 32: INTEGER [...]
  283 02 32: INTEGER
    : 6F 17 58 16 C7 F9 F7 56 7F 43 DB 19 92 8C 60 50
    : CA 6E 4B 10 41 0B CA B1 5B 25 06 1D C1 05 F8 36
    : }
```





Algorithmes à clef publique (12/16)

- × Clef RSA chiffrée
- × Algorithme symétrique DES, en mode CBC

```
$ openssl genrsa -3 -DES 512
Generating RSA private key, 512 bit long modulus
e is 3 (0x3)
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC-3F05FF3C5103E3BB
VjgE0b88PXaA8QzwuqlRyzwVrwtIUu68vXH7FzMtAWaHHrUyK6gjPHh6nHtho+Yy
RB1fn9UKMl2pzJi6dJW9J1kYJafjhuLAM09pAJYJ51F9P37qvu2TGnNq5QjutoGy
Otz0H2s/OeUNmdWcGFrRnjMwXMq8Ssaa28SHy/V0sPG8VQXdHxwBGSr3JP/Tf00C
/j5Y9uzblgw1GT1m4BG5hoWtVSJvbEZ1/R83dvzsBgg=
-----END RSA PRIVATE KEY-----
```

Vecteur d'initialisation

- × En pratique
 - Chiffrement de la clef RSA (au format DER) en DES (mode CBC)
 - Encodage du binaire obtenu en en base 64
 - × La clef de chiffrement DES et le vecteur d'initialisation (IV, utilisé par le mode CBC) sont dérivés du mot de passe fourni par l'utilisateur





Algorithmes à clef publique (13/16)

- × Génération des nombres premiers p et q

```
$ openssl genrsa 1024
Generating RSA private key, 1024 bit long modulus .....+++++
.....+++++
e is 65537 (0x10001)
-----BEGIN RSA PRIVATE KEY...
```

- × Tests de primalité

- × Difficulté d'engendrer de « grands » nombres premiers « aléatoires »

- × Tests de primalité probabilistes

- × Test de Sieve

- × Symbolisé par « . »

- × Test rapide, préliminaire

- × Mis en oeuvre dans PGP

- × Test de Miller-Rabin

- × Symbolisé par « + »

- × Méthode décrite dans « Handbook of Applied Cryptography »

- × <http://www.cacr.math.uwaterloo.ca/hac/about/chap4.ps>

- × Test caractérisé par une probabilité d'erreur

- × $1/2^{80}$ (Cf. `OPENSSL/crypto/bn/bn.h`)





Algorithmes à clef publique (14/16)

Publication de la clef publique RSA

- * Sous-ensemble des paramètres de la clef privée
- * Modulus n, exposant public e

```
$ openssl rsa -in rsa.PEM -pubout

-----BEGIN PUBLIC KEY-----
MfwwDQYJKoZIhvcNAQEBBQADSwwAwSAJBALGWJEoGAC6fYqpG9vCx1LEcjFtrhw2W
DZ71Ne+Q2ZvgbqDE9j8SIsjEStKc2UZ/w2bnPc8k+3LTTGCTQ9SAdsmCAwEAAQ==
-----END PUBLIC KEY-----
```

- * Structure ASN.1 de la clef publique obtenue

```
0 30 92: SEQUENCE {
2 30 13: SEQUENCE {
4 06 9: OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
15 05 0: NULL
: }
17 03 75: BIT STRING 0 unused bits, encapsulates {
20 30 72: SEQUENCE {
22 02 65: INTEGER [...]
89 02 3: INTEGER 3
: }
: }
: }
```





Algorithmes à clef publique (15/16)

Principaux usages

- × Service de confidentialité
 - × Chiffrement avec la clef publique du destinataire
 - × Faible volume de données
 - × Mode opératoire par blocs ?
 - × Protocole TLSv1 : travail sur un et un seul bloc (*chunk*)
 - × PreMasterSecret de 48 octets + bourrage PKCS#1 de 11 octets
 - Taille de clef publique d'un certificat X.509 de **472 bits au minimum**
 - Cas d'un certificat client : **376 bits au minimum** (signature d'empreinte SHA1+MD5)
- × Signature numérique
 - × Chiffrement avec la clef privée du signataire
- × Transport (RSA) ou échange (DH) de clef
 - × Résolution de la problématique de l'échange de clef secrète





Algorithmes à clef publique (16/16)

Récapitulatif

- × Algorithme le plus couramment utilisé : RSA
 - × Cryptosystème décrit dans la RFC 2437
 - × Formats binaire (encodage DER de la structure ASN.1) et ASCII « imprimable » (PEM)
 - × Une clef de chiffrement (n, e)
 - × Transport de clef : mécanisme de chiffrement de la clef de session avec la clef de chiffrement du destinataire
 - × Clef dite « publique », déposée dans un annuaire par exemple
 - × Une clef de signature d
 - × Signature : mécanisme de chiffrement avec la clef de signature par l'émetteur
 - × Service de non-répudiation
 - × Clef privée
- × Décomposition en facteurs premiers du modulus $n = p \cdot q$
 - × p et q , probables grands nombres premiers par construction
 - × Tests de primalité *probabilistes* par les applications (OpenSSL, PGP etc.)



Fonctions de hachage



Fonctions de hachage (1/7)

Fonction de hachage cryptographique

- × Obtention d'une empreinte numérique de taille fixe à partir d'un message de taille arbitraire
 - × Opération par blocs
- × Critères de sécurité
 - × Faibles collisions
 - × Collisions : possibilité d'obtenir une même empreinte pour deux messages distincts
 - × Sécurité calculatoire
 - × Sens unique
 - × Difficulté d'inversion



Fonctions de hachage (2/7)

Principaux algorithmes

- × MD5, « The MD5 Message-Digest Algorithm » (RFC1321)

- × <http://www.ietf.org/rfc/rfc1312.txt>

- × Empreinte de 128 bits

```
$ openssl md5 -c message.txt  
MD5(message.txt)= ef:79:ca:b5:4f:9b:2a:d9:8e:79:ef:46:46:e4:93:63
```

- × SHA, « Secure Hash Algorithm » (NIST)

- × <http://www.itl.nist.gov/fipspubs/fip180-1.htm>

- × Empreinte de 160 bits

```
$ openssl sha1 -c message.txt  
SHA1(message.txt)= af:06:9c:e9:bb:e9:21:29:b1:1a:7e:a8:77:9b:37:09:26:5e:84:b3
```

- × Variantes ou autres : dgst(1ssl)

```
$ openssl dgst -h
```





Fonctions de hachage (3/7)

Exemples d'utilisation :

- × Intégrité et scellement de fichiers
 - × md5sum(1)

```
$ md5sum image.iso  
d41d8cd98f00b204e9800998ecf8427e image.iso
```

- × HIDS (Host-Based Intrusion Detection System)
 - × AIDE
 - × Site officiel : <http://www.cs.tut.fi/~rammer/aide.html>
 - × Brève HSC : <http://www.hsc.fr/ressources/breves/aide.html>
 - × Tripwire
- × Stockage non réversible de mots de passe
 - × Fichier shadow(5), fondé sur l'algorithme md5

```
$ sudo grep davy /etc/shadow  
davy:$1$zdgjjRrg$b9oxpVEDX8I4KgWanwbpd.:11924:0:99999:7:::
```





Fonctions de hachage (4/7)

- × Utilisation de la commande passwd(1ssl)

- × Syntaxe

```
$ openssl passwd [-crypt] [-1] [-apr1] [-salt string] [-in file] [-stdin] [-quiet] [-table] {password}
```

- × Génération de l'empreinte

```
$ echo -n 'secret' | openssl passwd -1 -salt 'zdgjjRrg' -stdin -table  
secret          $1$zdgjjRrg$b9oxpVEDX8I4KgWanwbpd.
```

- × Signification des différents termes

`(1)(zdgjjRrg)$(b9oxpVEDX8I4KgWanwbpd.)`

- × \$1 : Algorithme MD5

- × \$2 : Piment

- × Protection contre les attaques par dictionnaire précalculé

- × Piment en clair (différent d'un MAC)

- × \$3 : Empreinte résultante





Fonctions de hachage (5/7)

Générateurs de nombres pseudo-aléatoires

- × Sources d'entropie multiples
 - × Évènements non déterministes
- × Fonctions de hachage pour « multiplexer » la sortie des sources d'entropie et constituer un « pool d'entropie »
- × Démons collecteurs d'entropie (sous Unix)
 - × Interfaces/pseudo-périphériques
 - × /dev/random (bloquant)
 - × /dev/urandom (non-bloquant)
 - × /dev/inter_rng, (processeurs intel de type i8xx)
 - × PRNGD sous Solaris
 - × http://www.aet.tu-cottbus.de/personen/jaenicke/postfix_tls/prngd.html
- × Analyse de quelques générateurs de nombres pseudo-aléatoires
 - × Phrack 59 « Cryptographic random number generators »
 - × <http://www.phrack.com/show.php?p=59&a=15>
 - × DrMungkee <pub@drmungkee.com>
 - × Intel RNG (SHA-1), Yarrow (SHA-1), /dev/random/ (MD5)





Fonctions de hachage (6/7)

- × Génération d'identifiants de sessions...
 - × <http://lxr.php.net/source/php4/ext/session/session.c>
- × Utilisations multiples !
 - × « Pile ou face »

Influence du changement d'un caractère de texte clair sur l'empreinte résultante

```
$ cat message.txt
Contrôle d'intégrité
$ cat message.txt | openssl md5 -c
ef:79:ca:b5:4f:9b:2a:d9:8e:79:ef:46:46:e4:93:63
$ cat message.txt | tr C c | openssl md5 -c
c5:f1:bf:54:2a:7f:27:04:d2:65:78:a1:bc:47:7c:e7
```



Fonctions de hachage (7/7)

Principaux usages

- × Signature numérique
 - × Norme de fait : RSA
 - × Mécanisme de chiffrement par une clef privée RSA d'une empreinte MD5 ou SHA-1
 - × Service de non-répudiation
- × Scellement
 - × Génération d'un sceau, ou code d'authentification de message
 - × MAC : « *Message Authentication Code* »
 - × Exemple : fonction de hachage à sens unique indexée par une clef secrète
 - × Service d'authenticité des données
 - × Authentification
 - × Intégrité
 - × Pas de service de non-répudiation
 - × Cf. SSL/TLS
 - × Horodatage