Université Paris II
Michel de Rougemont
http://www.irif.fr/~mdr

Sécurité Informatique: chiffrement et https

- Fonctions de hâchage
- Chiffrement clé secrète
- Chiffrement clé publique

1. Chiffrement et cryptographie
2. Signature Electronique
3. SSL ou https
4. PGP ou S/mime (mail sécurisé)

Sécurité

- Gestion d'accès par mot de passe
  - Login
  - Protection d'un sous-répertoire
- Chiffrement de cette information
  - Protection de login/passwd
  - Protection des messages
  - SSL (https)
- Signature : authentification
  - Théorie unifiée du chiffrement à clé publique : schéma RSA
  - S/MIME
  - Gestion des certificats

Fondements

- Fonction facile ou difficile à calculer
  - Ce qui est autorisé est facile
  - Ce qui est non autorisé est difficile mais pas impossible.

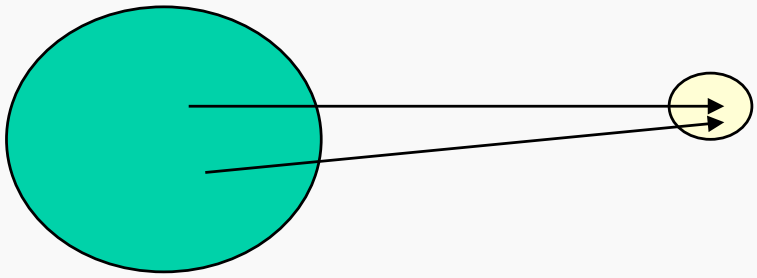
- Facile = calculable en temps polynomial = P-calculable
- Soit f(x)=y et n=|x|

  t_x=nb. d'opérations pour calculer f(x)

$$T_n = Max : t_x \ , n = \left| x \right| \leq c.n^k$$

Fonction de hâchage

Outil important.



h : D → I

Conditions sur h(x)

Facile à calculer

Difficile à inverser

h(x+dx) =/= h(x)

Applications des fonctions de hâchage

Outil important.

- MD4, MD5 (Message Digest) sur 128 bits
- SHA sur 160 bits

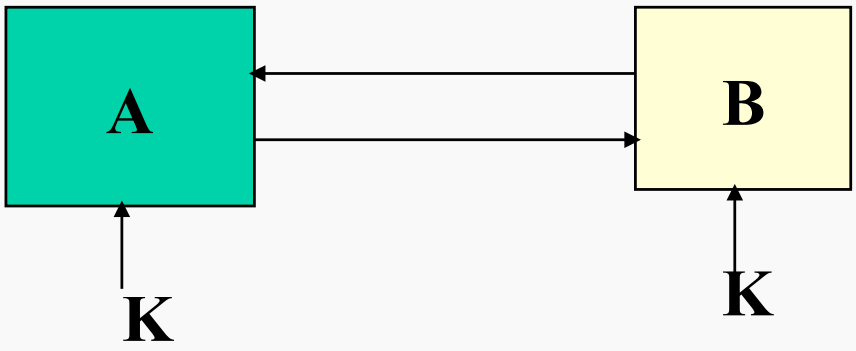
Stocker les mots de passe:

/etc/passwd sous Linux

Partager un secret à distance:
(divorcer à distance)

Cryptographie à Clé secrète

SKC (Secret Key Cryptography)



DES (Data Encryption Standard)

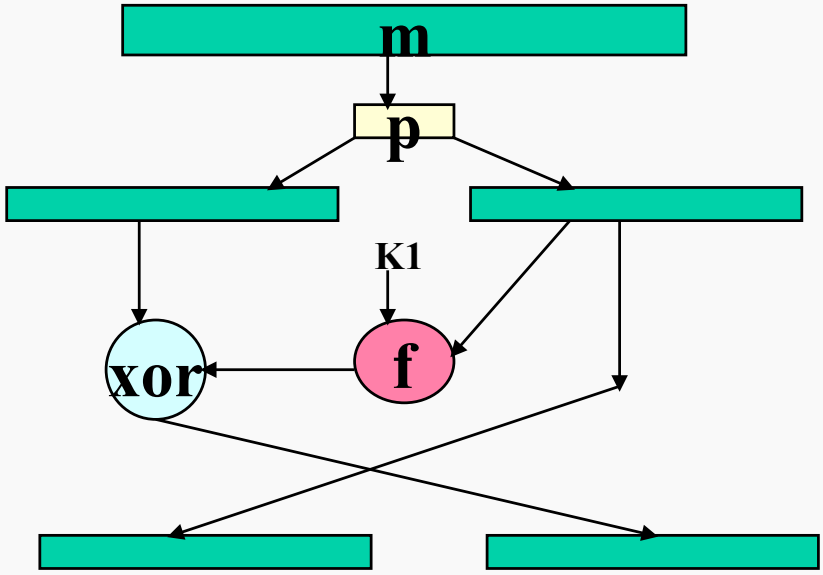
K=56 bits

M=64 bits

Permutations + XOR

Cryptographie à Clé secrète

Des(m,K)=c



Des(c,K) =m

Applications de DES (AES)

1. Identification (militaire Ami/ennemi)

   Message aléatoire r

   Envoyer c=des(r,K)

   Personne testée calcule s=des(c,K) et envoie s.

   Test **r=s ?**

1. Vérifier les mots de passe Client/ Serveur NT

   K=hash(passwd) sur 16 octets=K1.K2.K3

   Message aléatoire =r =challenge

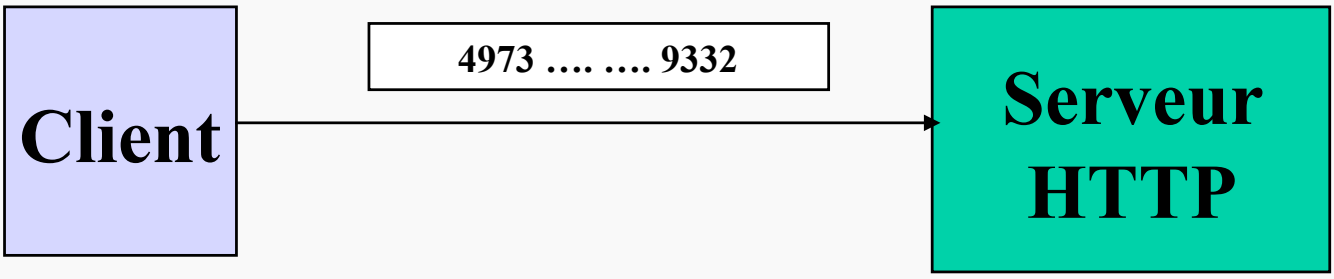
   Client envoie des(r,K1), des(r,K2) et des(r,K3) que le serveur vérifie

   Seules les valeurs cryptées sont transmises.

Chiffrement à clé publique

- **But : Confidentialité et Authenticité**
- **Historique : militaire (1975) puis public**
- **Méthode publique (clé)**
  - **Chiffrement à clé secrète : DES et AES**
  - **Chiffrement à clé publique : RSA**

SSL en pratique

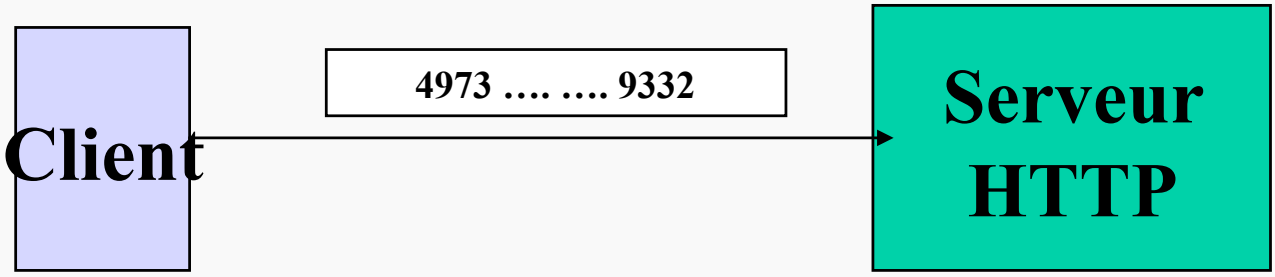


Client veut protéger ses informations: #VISA, montant $, …..

Alice veut protéger le message envoyé à Bob.

Bob veut s ’assurer que le message vient d ’Alice

SSL ou https



- **A vérifie le certificat avec la clé publique du navigateur**
- **A crypte avec (N_b, e_b)**
- **B décrypte avec d_b (qu ’il est seul à connaître).**
- **DES avec K**

1. Chiffrement à clé publique

- **Chaque client possède deux clés**
  - **clé publique (sur une page WEB et donc connue de l ’ensemble)**
  - **clé secrète (connue du seul client)**
- **Les deux clés sont utilisées**

Cryptage à clé publique

- **Facile vs. Difficile**
- **Paramètre : n= longueur des clés**
  - **56 bits**
  - **128 bits**

Solution RSA

- **Solution au système à clé publique**
- **Rivest, Shamir, Adleman**
  - **MIT, 1976**

Factorisation

- **Multiplication A.B = C**
- **Factorisation C=A.B**

Nombres premiers

- **(a,n)=1**
  - **a et n sont premiers entre eux**
  - **Pgcd (a,n)=1**
- **N est premier : seuls diviseurs sont 1 et N.**
- **si a <n (Fermat)**
- **si a <n=p.q (Euler)**

$$a^{n-1} \equiv 1 (n)$$

$$a^{(p-1)(q-1)} \equiv 1 (n)$$

Congruences modulo n

- **x.a = 1 (n) : x,a < n et x.a =k.n+1 par exemple x.3=1 (14)**

- **Solutions si (a,n)=1**
  - **si a=3, alors x=5 car 5.3=15=1.14 +1**
  - **si a=3, alors pas de solution**

N=p.q où p,q sont premiers

- **p=47, q=71, N=3337**
- **(p-1)(q-1)=46.70=3220**
- **choisir e (au hasard) tel que (e,(p-1).(q-1))=1**

Clé publique

- **(N,e) est la clé publique**
- **p=47, q=71, N=3337, (p-1)(q-1)=3220**
- **Soit e =79**

Clé secrète : d

- **Equation RSA détermine le lien entre clés publique et secrète:**
- **e.d = 1 (p-1)(q-1)**
  - **assuré d 'une solution**
- **p=47, q=71, N=3337, (p-1)(q-1)=3220**
- **e =79 impliquent d=1019**

Chiffrement (Cryptage)

- **Alice veut protéger le message envoyé à Bob.**
- **Clés d ’Alice (N_a, e_a) et d_a**
- **Clés de Bob (N_b, e_b) et d_b**

Cryptage d ’Alice

- **M= m_1 m_2 m_3 ..... m_i. ..... m_L**

- **Encodage**

C= c_1 c_2 c_3 ..... c_i. ..... c_L

- **Encodage :**

$$c_i = (m_i)^{eb} \quad (N_b)$$

Exemple de message crypté

- **M= m_1 m_2 m_3 ..... m_i. ..... m_L**
  688 232 687 966 668 3

- **Encodage**

$$c_1 = 688^{79}$$

- **C = 1570 2756 2714 2276 2423 158**

Décryptage de Bob

- **Décodage de**

C= c_1 c_2 c_3 …... c_i. ……. c_L

- **Equation de décodage**

$$S_i = \left(c_i\right)^{db} \left(N_b\right) = \left(m_i\right)^{db.\ eb} \left(N_b\right)$$

- **S= s_1 s_2 s_3 …... s_i. ……. s_L**

Message décodé =message initial

- **Equation de décodage**

$$s_i = \left(c_i\right)^{db} \left(N_b\right) =$$

$$\left(m_i\right)^{db.\ eb} \left(N_b\right) = .\left(m_i\right)^{k.(p-1)(q-1)+1} \left(N_b\right)$$

$$= m_i.\left(m_i\right)^{k.(p-1)(q-1)} \left(N_b\right)$$

$$\mathbf{s_i = m_i}$$

Exemple de message décrypté

- **C = 1570 2756 2714 2276 2423 158**

$$c_1 = 688^{79}$$

$$S_1 = c_1^{1019} = 688$$

- **Décodage**

S = S1 S2 S3 …... Si. …….. SL

S= 688 232 687 966 668 3

2. Authentification

- **Bob veut s ’assurer que le message vient bien d ’Alice.**
- **Clés d ’Alice (N_a, e_a) et d_a**
- **Clés de Bob (N_b, e_b) et d_b**

Authentification d ’Alice

- **Message crypté**

C= c_1 c_2 c_3 …… c_i. …… c_L.

- **Equation d ’authenfication**

$$a_i = (c_i)^{d_a} \ (N_a)$$

Décryptage et Authentification

- **Authentification**

$$c_i = (a_i)^{e_a} \quad (N_a) = (c_i)^{d_a \; e_a} \; (N_a)$$

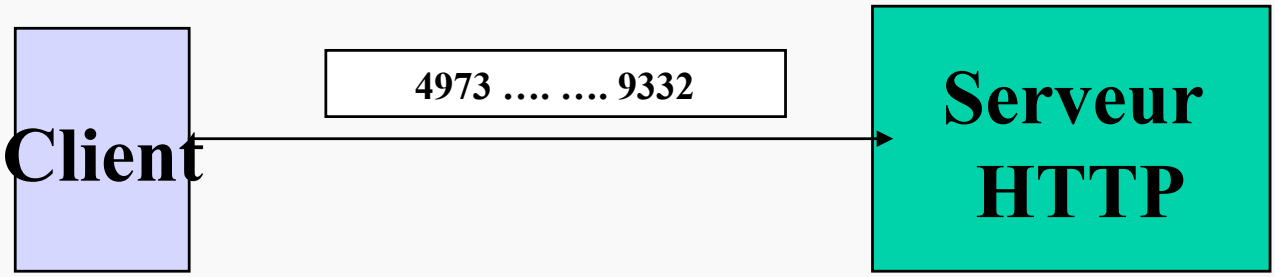
(même principe que pour le cryptage)

- **Décodage de**

C= c_1 c_2 c_3 …… c_i. …… c_L

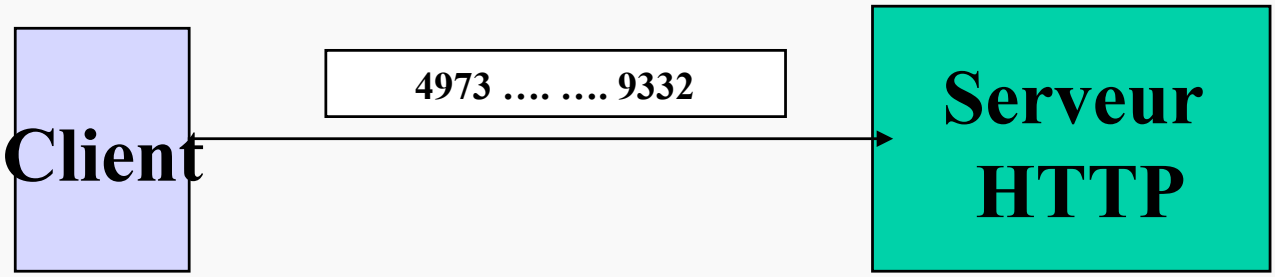
S=M

3. SSL : Secure Socket Layer



- **A vérifie le certificat avec la clé publique du navigateur**
- **A crypte K (clé de session) avec (N_b, e_b)**
- **B décrypte avec d_b (qu 'il est seul à connaître).**
- **DES avec K**

SSL : Secure Socket Layer



- **Serveur transmet un certificat X509 décrivant (N_b, e_b) à A.**

ww.lri.fr

Dates:

k=145GUY6UENEJOE9

Signature: AE56GTd23EZ67A

4.PGP : Pretty Good Privacy

www.pgpi.com

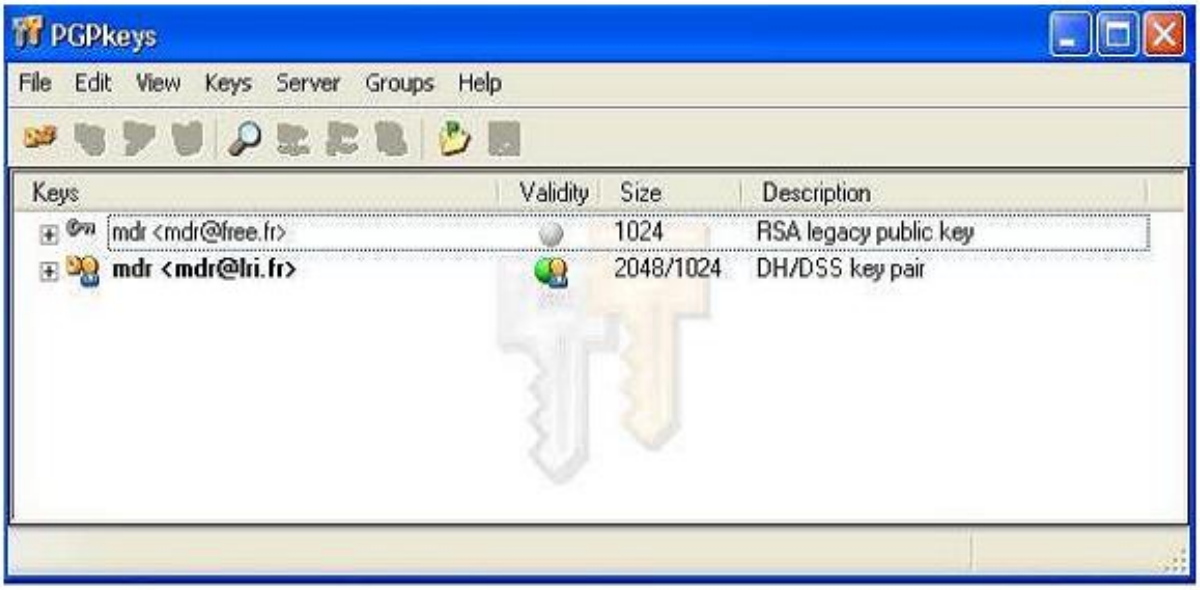
Version 6.5 à 8

Intégration à Outlook

Génération de clés

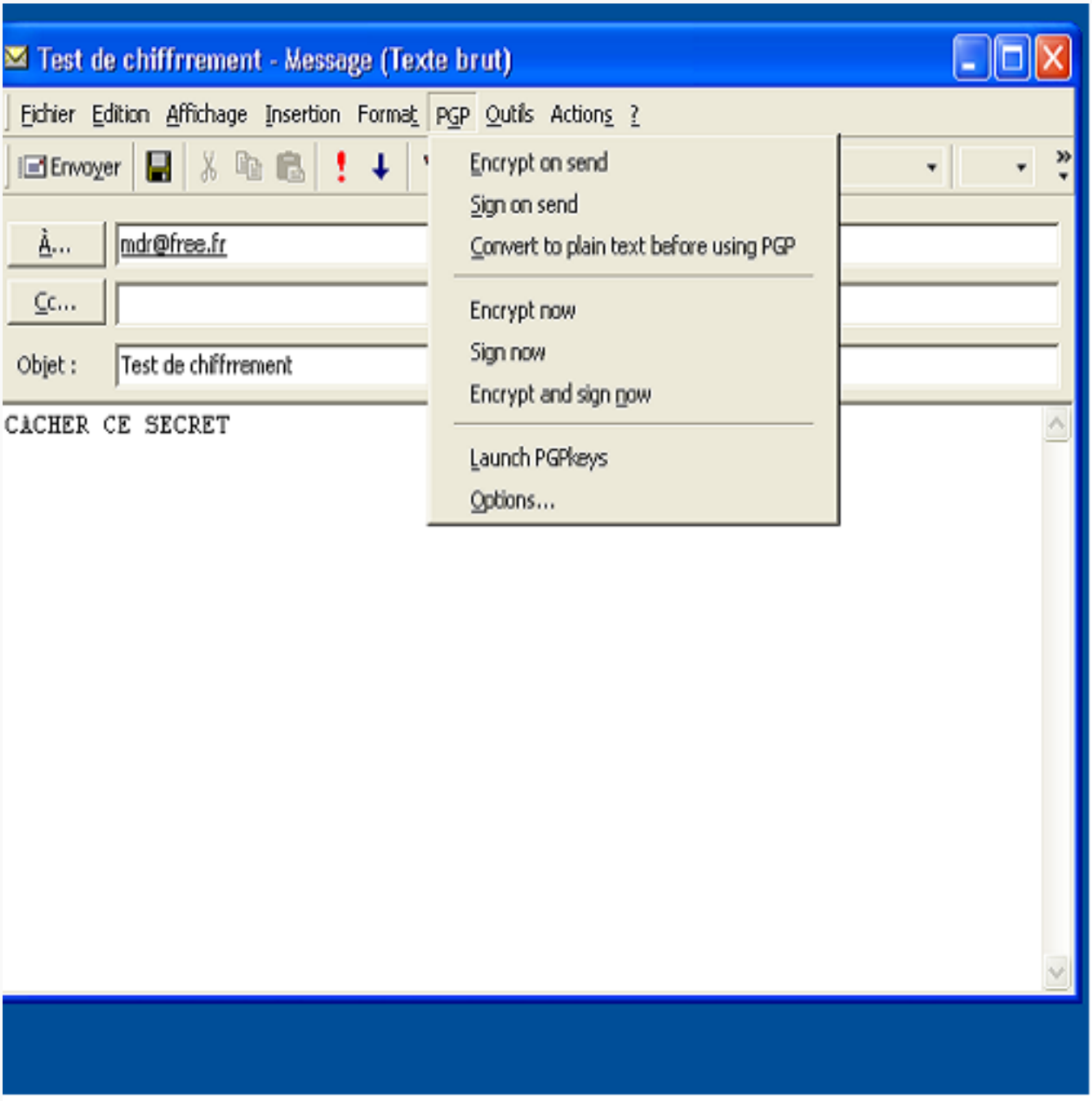
mot de passe pour K (clé privée)

PGP Keys


PGPkeys
File Edit View Keys Server Groups Help
Keys
Validity
Size
Description
mdr <mdr@free.fr>
1024
RSA legacy public key
mdr <mdr@lri.fr>
2048/1024
DH/DSS key pair

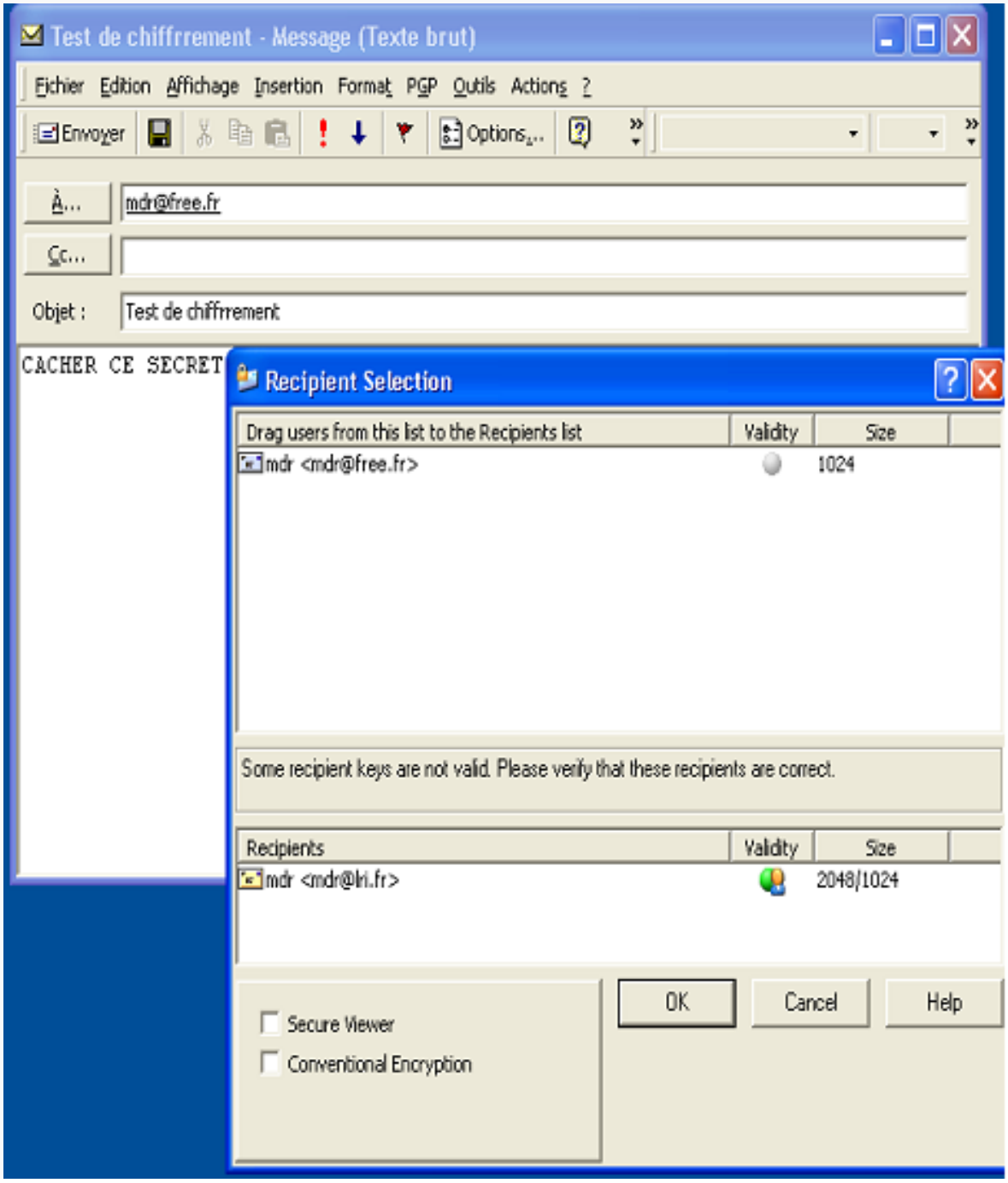
Outlook et PGP

Envoi d'un mail chiffré :



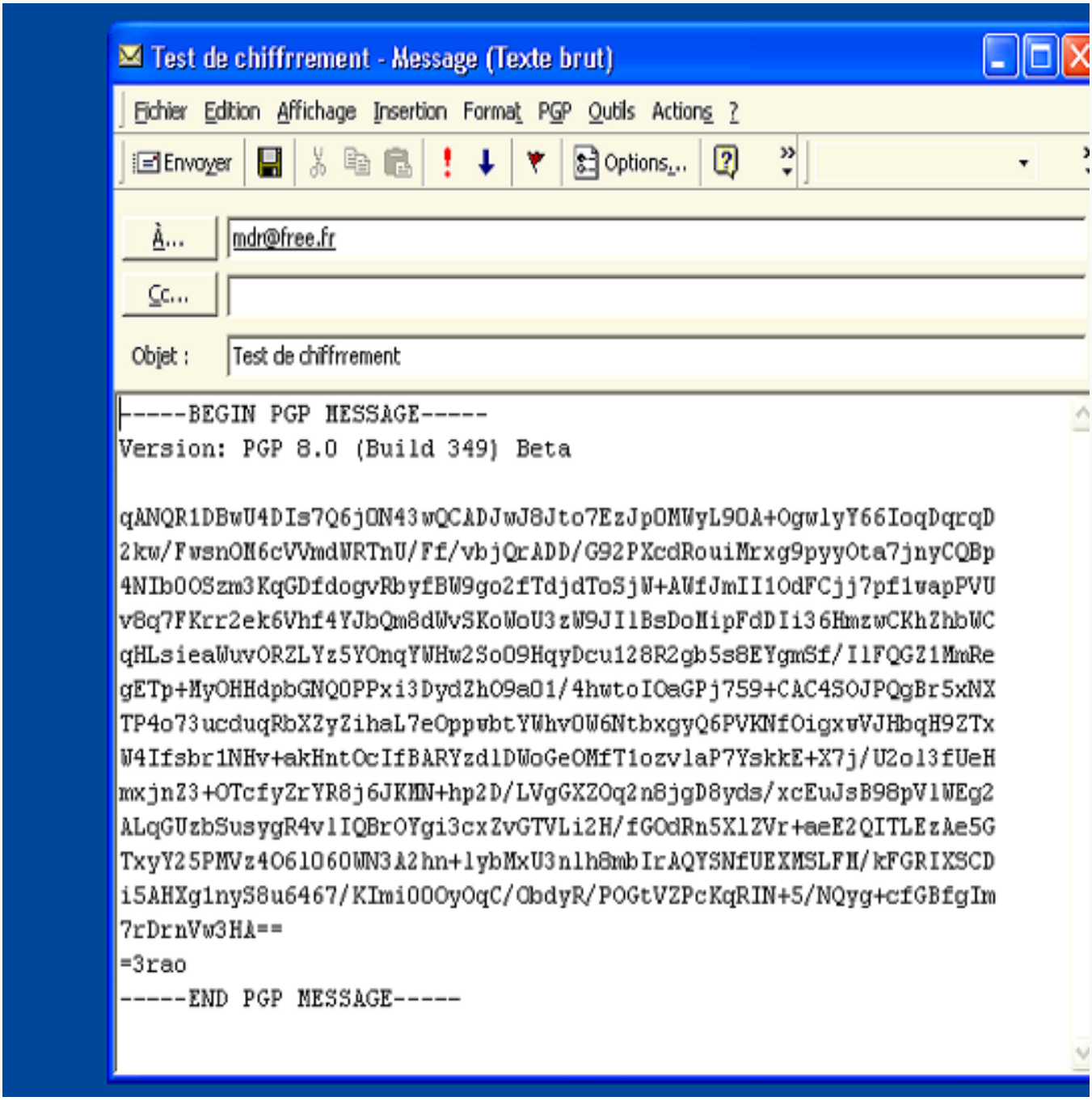
Outlook et PGP

Sélection de la clé publique du destinataire : mdr@free.fr



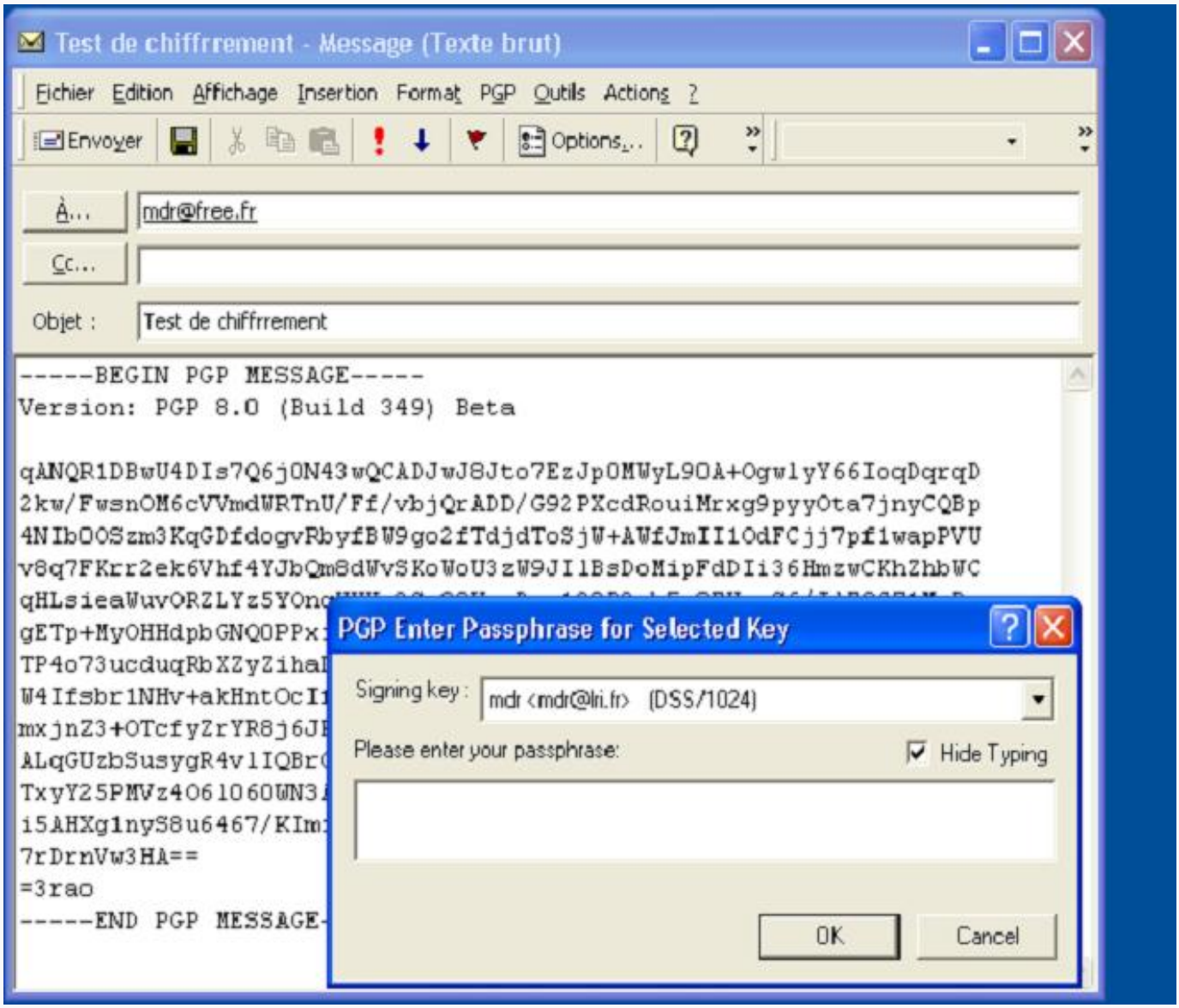
Outlook et PGP

Message chiffré:



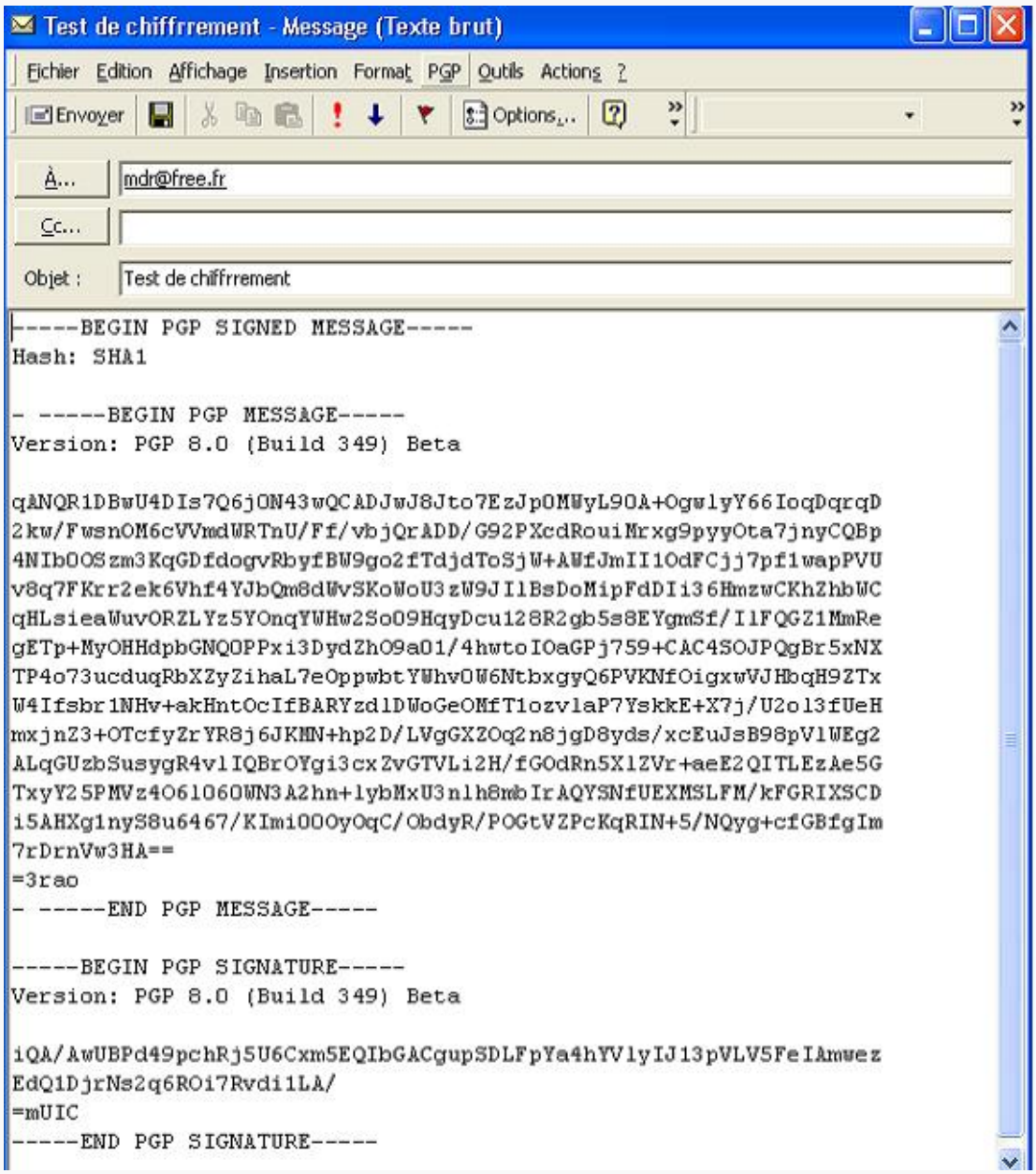
Outlook et PGP

Appel de signature:



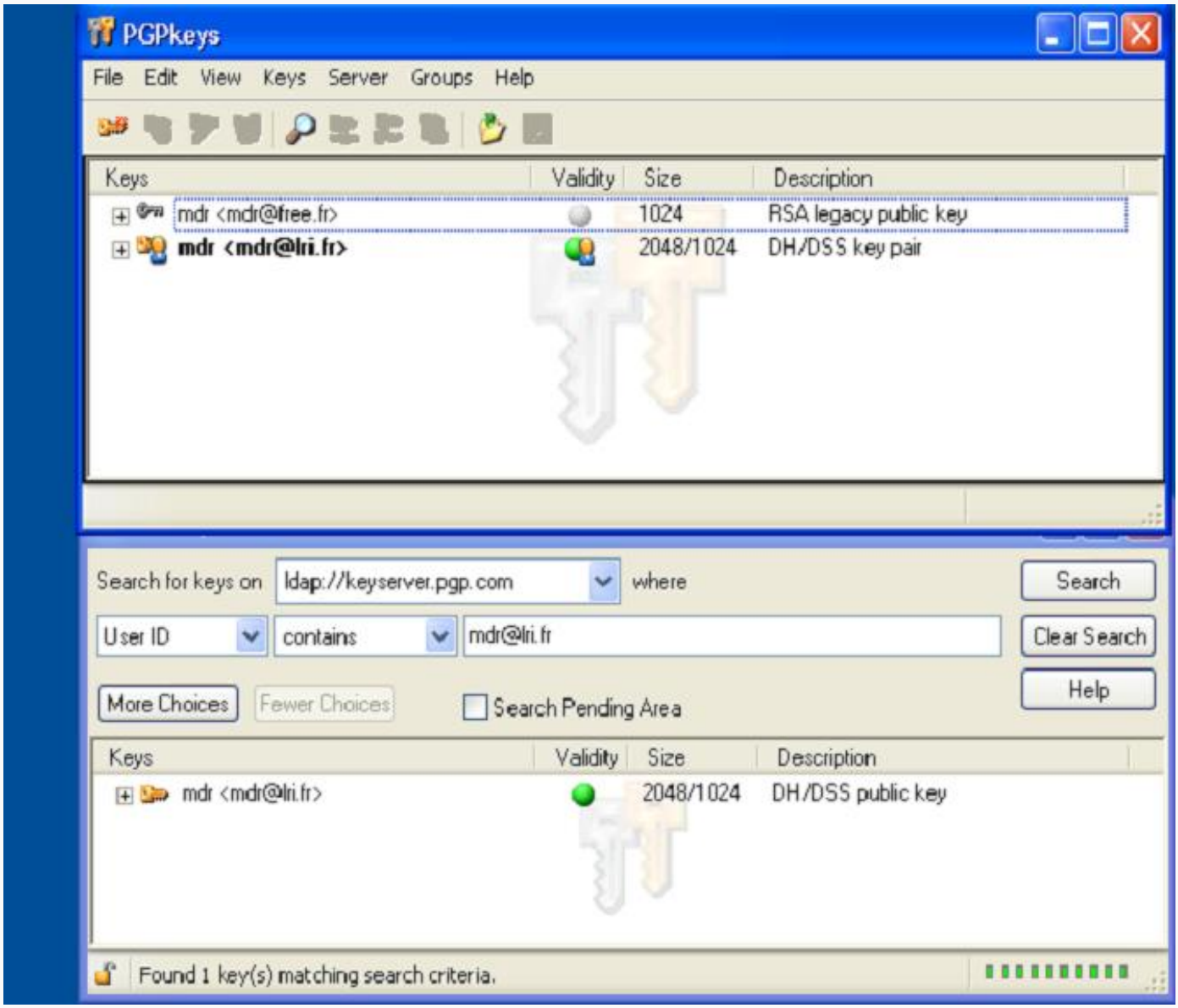
Outlook et PGP

Message chiffré et signé:



PGP Keys

Modifier son anneau de clés:
Server->search
Exemple : mdr@lri.fr



PGPmail vs. S/MIME

- **Clé publique ou certificat X509**

ww.lri.fr

Dates:

k=145GUY6UENEJOE9

Signature: AE56GTd23EZ67A

SSL et S/MIME

- **SSL : Identification du serveur par un certificat SSL**

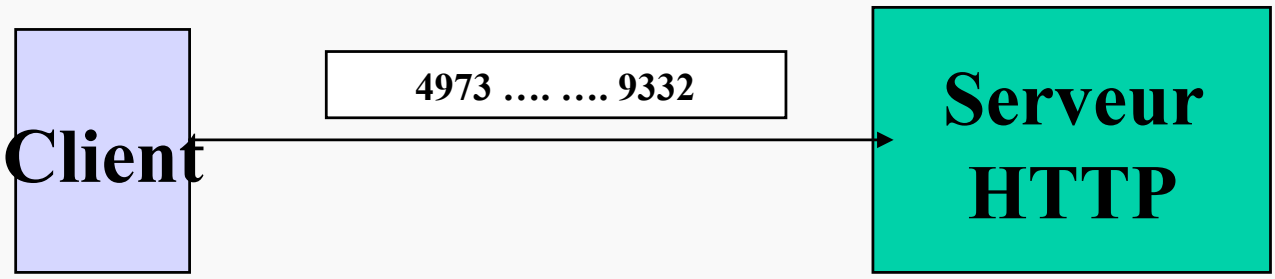
ww.lri.fr

Dates:

k=145GUY6UENEJOE9

Signature: AE56GTd23EZ67A

Chiffrement SSL :



- **A vérifie le certificat avec la clé publique du navigateur**
- **A crypte avec (N_b, e_b)**
- **B décrypte avec d_b (qu ’il est seul à connaître).**
- **DES avec K**

Projet:

- Apache
  - Firewall
  - Dmz
  - https
- Gestion du fichier configuration
- Easyphp ou Xampp
- Environnement Linux

SSL sous Apache

- Apache: http://www.apache.org
- Serveur http depuis 1995
  - Httpd : http daemon
  - « A patchy server »
  - No prisoner…
  - OpenSSL (failles t.q Heartbleed)
- Gestion du fichier configuration
  - https://www.digicert.com/ssl-certificate-installation-apache.htm
  - <VirtualHost 192.168.0.1:**443>DocumentRoot /var/www/ html2ServerName www.yourdomain.comSSLEngine onSSLCertificateFile /path/to/ your_domain_name.crtSSLCertificateKeyFile /path/to/ your_private.keySSLCertificateChainFile /path/to/ DigiCertCA.crt</VirtualHost>**
- Easyphp ou Xampp sous Windows/Linux

Politique de sécurité

Implémentation selon les systèmes d'exploitation.

Linux : Iptables (Ipchains)

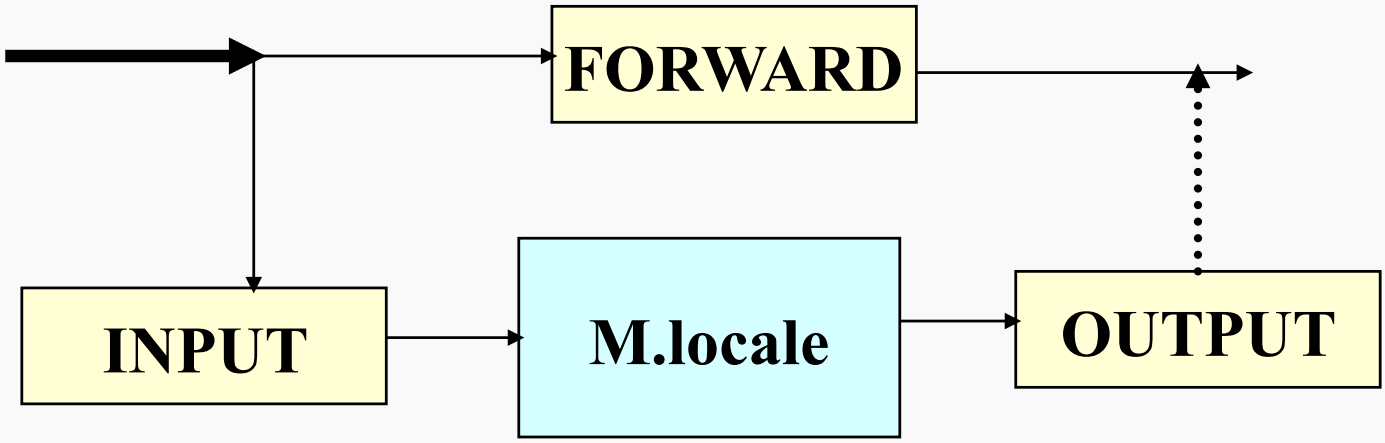
/etc/init.d/iptables

Programme standard sous linux
de gestion des règles de parefeu.

Iptables sur dup2.dyndns.org

```
# Source 'em up
. /etc/init.d/functions

IPTABLES_CONFIG=/etc/sysconfig/iptables

if [ ! -x /sbin/iptables ]; then
        exit 0
fi

KERNELMAJ=`uname -r | sed                    -e 's,\..*,,'`
KERNELMIN=`uname -r | sed -e 's,[^\.]*\.,,' -e 's,\..*,,'`

if [ "$KERNELMAJ" -lt 2 ] ; then
        exit 0
fi
if [ "$KERNELMAJ" -eq 2 -a "$KERNELMIN" -lt 3 ] ; then
        exit 0
fi
……..
```

Iptables



3 chaines : FORWARD, INPUT, OUTPUT

Iptables –P INPUT DROP (plus de paquets ne passent vers la machine)

-A input -s 0/0 -d 0/0 80 -p tcp -y -j ACCEPT (On Ajoute la règle : les paquets TCP du port 80 sont acceptés)

Politique de sécurité

Configuration: /etc/sysconfig/iptables

```
[root@dhcppc2 init.d]# more /etc/sysconfig/ipchains
# Firewall configuration written by lokkit
# Manual customization of this file is not recommended.
# Note: ifup-post will punch the current nameservers through the
#       firewall; such entries will *not* be listed here.
:input ACCEPT
:forward ACCEPT
:output ACCEPT
-A input -s 0/0 -d 0/0 80 -p tcp -y -j ACCEPT
-A input -s 0/0 -d 0/0 21 -p tcp -y -j ACCEPT
-A input -s 0/0 -d 0/0 22 -p tcp -y -j ACCEPT
-A input -s 0/0 -d 0/0 443 -p tcp -y -j ACCEPT
-A input -s 0/0 -d 0/0 3306 -p tcp -y -j ACCEPT
-A input -s 0/0 67:68 -d 0/0 67:68 -p udp -i eth0 -j ACCEPT
-A input -s 0/0 67:68 -d 0/0 67:68 -p udp -i eth1 -j ACCEPT
-A input -s 0/0 -d 0/0 -i lo -j ACCEPT
-A input -p tcp -s 0/0 -d 0/0 0:1023 -y -j REJECT
-A input -p tcp -s 0/0 -d 0/0 2049 -y -j REJECT
-A input -p udp -s 0/0 -d 0/0 0:1023 -j REJECT
-A input -p udp -s 0/0 -d 0/0 2049 -j REJECT
```

Gestion des ports sur un routeur

Routeur Netgear 114