GnuPG HOWTO v2.5

Ou comment chiffrer ses mails sous Linux

par cho7

Dernière modification le 05/01/2004. La version la plus récente de ce document est disponible sur http://gpglinux.free.fr

Préliminaires

Pour les décideurs préssés, un Memento est disponible en dernières pages, mais sans lire le reste, c'est pas simple !

Ca me parait evident, il faut installer GPG sur son ordinateur et, moins evident, celui ci doit tourner sous Linux.
Pour la partie Configurer son client de messagerie (Evolution, sylpheed, etc), c'est dans la dernière partie de ce howto, mais je conseille très sincèrement la lecture de tout le document, sinon vous allez être largué, la partie Evolution etant rédigée en partant du principe que vous avez lu le reste du document)
Toutes les opérations dans ce mini-howto ont été réalisées sur Debian Sid (http://www.debian.org)
Commencez donc par installez GPG (et evolution/sylpheed si ca vous interesse) comme n'importe quel programme :

```
# apt-get install gnupg
# apt-get install evolution
```

Etape 1 : création d'une paire de clés

“Euh attend un peu là, pas si vite, c'est quoi une paire de clés ??”
Bon, quelques rappels élémentaires sur le chiffrage a clé publique/clé privé :
Ce systeme repose sur 2 clés. la clé publique sert a chiffrer un message, et la clé privé sert a déchiffrer un message chiffré a l'aide de la clé publique. Dans la pratique, chacun garde sa clé privée bien préciseusement chez lui, et donne sa clé publique a tout ses contacts. Ainsi si je veux chiffrer mon mail et l'envoyer a Albert, il faut que j'ai en stock la clé publique d'albert, si je ne l'ai pas, c'est que albert ne me l'a pas donné, ou bien tout simplement qu'Albert n'utilise pas GPG (et il a le droit), auquel cas, il m'est impossible d'envoyer un mail chiffré a albert !!

Donc pour résumé, si un de mes contacts veut m'envoyer un mail chiffré, il devra d'abord avoir ma clé publique dans son trousseau de clé, il chiffrera son mail avec, et dès lors, le mail chiffré n'est déchiffrable QUE par MA clé privée (et pas la clé privée du voisin). L'expediteur lui meme serait infoutu de déchiffrer le mail qu'il vient d'envoyer, c'est aboslument irreversible, un mail chiffré a l'aide d'une clé publique, ne peut etre déchiffré que par la clé privée générée en meme temps que la clé publique

Voilà pour le bref rappel théorique !

Maintenant la pratique, ouvrez un shell et tappez :

```
cho7@cho7land:~$ gpg --gen-key
gpg (GnuPG) 1.2.5; Copyright (C) 2004 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.
```

```
Sélectionnez le type de clé désiré:
   (1) DSA et ElGamal (par défaut)
   (2) DSA (signature seule)
   (4) RSA (signature seule)
Votre choix ?
1
La paire de clés DSA fera 1024 bits.
Préparation à la génération d'une nouvelle paire de clés ELG-E.
            la taille minimale est  768 bits
          la taille par défaut est 1024 bits
 la taille maximale conseillée est 2048 bits
Quelle taille de clé désirez-vous ? (1024)
1024
La taille demandée est 1024 bits
Spécifiez combien de temps cette clé devrait être valide.
         0 = la clé n'expire pas
      <n>  = la clé expire dans n jours
      <n>w = la clé expire dans n semaines
      <n>m = la clé expire dans n mois
      <n>y = la clé expire dans n années
La clé est valide pour ? (0)
0
Key n'expire pas du tout
Est-ce correct (o/n) ?
o
Vous avez besoin d'un nom d'utilisateur pour identifier votre clé; le
programme le construit à partir du nom réel, d'un commentaire et d'une
adresse e-mail de cette manière:
   « Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de> »

Nom réel:
Cho7 Kipu
Adresse e-mail:
cho7@dlfp.org
Commentaire:
facultatif
Vous avez sélectionné ce nom d'utilisateur:
    "Cho7 Kipu <cho7@dlfp.org>"

Changer le (N)om, le (C)ommentaire, l'(E)-mail ou (O)K/(Q)uitter ?
o
Vous avez besoin d'un mot de passe pour protéger votre clé secrète.
Password :
votre password (a repeter ensuite une seconde fois)

Un grand nombre d'octets aléatoires doit être généré. Vous devriez faire
autre-chose (taper au clavier, déplacer la souris, utiliser les disques)
pendant la génération de nombres premiers; cela donne au générateur de
nombres aléatoires une meilleure chance d'avoir assez d'entropie.
```

```
+++++.+++++++++++++++.++++++++++...+++++++++++++++..+++++..+++++++++++++++.++
+++
++++++++++.++++++++++.++++++++++++++++++++++++++++++++++++++++>+++++..+++
++..... ..>+++++.....................+++++
Un grand nombre d'octets aléatoires doit être généré. Vous devriez faire
autre-chose (taper au clavier, déplacer la souris, utiliser les disques)
pendant la génération de nombres premiers; cela donne au générateur de
nombres aléatoires une meilleure chance d'avoir assez d'entropie.
++++++++++.+++++...++++++++++++++++++..++++++++++++++++++++++++++++++++++++
+++++
+.++++++++++++++++.++++++++++++++++.++++++++++.++++++++++++++++>+++++..... ..
.....+++++^^^
les clés publique et secrète ont été créées et signées.
clé marquée comme ayant une confiance ultime.

pub  1024D/0C792AD8 2004-09-29 Cho7 Kipu <cho7@dlfp.org>
 Empreinte de la clé = 999A 46D0 5E2D D616 2974  DD86 D20C AA55 0C79 2AD8
sub  1024g/51741FEF 2004-09-29

cho7@cho7land:~$
```

Voilà, votre paire de clé publique/privée est générée avec succès !

Pour s'en assurer:

```
cho7@cho7land:~$ gpg --list-keys
/home/cho7/.gnupg/pubring.gpg
------------------------------
pub  1024D/0C792AD8 2004-09-29 Cho7 Kipu <cho7@dlfp.org>
sub  1024g/51741FEF 2004-09-29
```

Ici on voit que ma clé publique a été créée, et possede l'id 0C792AD8
Veuillez notez que la petite taille de cette id (8 caracteres), laisse présager un risque de doublon avec une autre clé publique générée, donc pour enlever tout doute possible sur l'identification de la clé(enfin techniquement il est là aussi possible d'avoir un doublon mais bon...), nous préfererons utiliser le fingerprint de la clé, littéralement une “empreinte digitale”:

```
cho7@cho7land:~$ gpg --fingerprint
/home/cho7/.gnupg/pubring.gpg
------------------------------
pub  1024D/0C792AD8 2004-09-29 Cho7 Kipu <cho7@dlfp.org>
 Empreinte de la clé = 999A 46D0 5E2D D616 2974  DD86 D20C AA55 0C79 2AD8
sub  1024g/51741FEF 2004-09-29
```

Voilà, vous pouvez donc lire ici l'empreinte de votre clé publique (999A 46D0 5E2D D616 2974 DD86 D20C AA55 0C79 2AD8), se terminant par les 8 chiffres de votre id précédemment vu.
Notez cette empreinte sur un bout de papier, elle nous servira plus tard.

S'enregistrer auprès d'un serveur de clés

Une fois vos clés générées, il faut stocker sur un serveur votre clé publique, afin que vos contacts puissent trouver votre clé publique et ainsi vous envoyer un mail chiffré !
pour se faire, nous allons utiliser le serveur http://pgp.mit.edu, mais il y en a d'autres bien

evidemment. La plupart se réactualisent entre eux, mais pas toujours, donc l'idéal et de garder toujours le meme serveur, et de le préciser a ses contacts, pour etre sûr qu'ils vous trouve.
Autre détail, la création d'un certificat de revocation est essentielle, pour prévenir par exemple le serveur de clé que votre clé publique n'est plus bonne et que vos contacts doivent cesser de l'utiliser !
Allez hop, on va créé ce certificat de revocation

```
cho7@cho7land:~/Gpg$ gpg --gen-revoke cho7@dlfp.org > revoc_cho7@dlfp.org.txt
sec  1024D/0C792AD8 2004-09-29   Cho7 Kipu <cho7@dlfp.org>

Générer un certificat de révocation pour cette clé ?
o
choisissez la cause de la révocation:
  0 = Aucune raison spécifiée
  1 = La clé a été compromise
  2 = La clé a été remplacée
  3 = La clé n'est plus utilisée
  Q = Annuler
(Vous devriez sûrement sélectionner 1 ici)
Votre décision ?
3
Entrez une description optionnelle ; terminez-là par une ligne vide:
> J'ai perdu mon mot de passe
>
Cause de révocation: La clé n'est plus utilisée
J'ai perdu mon mot de passe
Est-ce d'accord ?
o

Vous avez besoin d'un mot de passe pour déverrouiller la clé secrète pour
l'utilisateur: "Cho7 Kipu <cho7@dlfp.org>"
clé de 1024 bits DSA, ID 0C792AD8, créée le 2004-09-29
votre password de clé secrète
sortie avec armure ASCII forcée.
Certificat de révocation créé.
```

Voilà, votre certificat est créé, garder le fichier texte précieusement, car si quelqu'un l'utilise, il peut invalider vos clés en prevenant le serveur qu'elles n'existe plus, alors que ce sera faux !! Nous verrons plus tard comment s'en servir...

Pour revenir a nos moutons, nous devons nous enregistrer sur le serveur de clé.
pour celà, c'est très simple :

```
cho7@cho7land:~$ gpg --keyserver pgp.mit.edu --send-keys cho7@dlfp.org
gpg: l'envoi à `pgp.mit.edu' s'est déroulé avec succès (résultat=200)
```

Voilà, votre clé a été exporté sur le serveur de clé, vous pouvez allez l'admirer sur pgp.mit.edu en tappant dans le champ String votre id sur 8 chiffres précédé d'un '0x', ou encore votre nom, votre adresse mail, etc.
pour la mienne : http://pgp.mit.edu:11371/pks/lookup?op=get&search=0x0C792AD8

Revoquer sa clé sur le serveur de clé :

Si comme indiqué plus haut vous ne vous servez plus de vos clé pour une raison ou une autre (compromise par quelqu'un qui possede votre clé privée, vous avez perdu votre mot de passe, etc), vous devez impérativement prévenir le serveur de clé de cette revocation !
Pour se faire, munissez vous du fichier de revocation que vous avez créé plus haut, et importez le dans votre porte-clé via la commande

```
gpg –import revoc_cho7@dlfp.org.txt
```

Verifiez que votre trousseau de clé a bien enregistrer la revocation en listant vos clés :

```
gpg –list-keys
```

Votre clé doit maintenant etre marquée comme [révoquée]
Vous pouvez donc là ré-envoyer sur le serveur de clé pour mettre a jour ce dernier :

```
gpg --keyserver pgp.mit.edu --send-keys cho7@dlfp.org
```

Voilà, votre clé est révoquée, et donc inutilisable.
Vous pouvez donc supprimer vos clés publique et privée de votre trousseau.
Pour se faire :
supprimez d'abord la clé secrète :

```
gpg –delete-secret-keys cho7@dlfp.org
```

puis la ou les clés publiques ratachées :

```
gpg –delete-keys cho7@dlfp.org
```

S'envoyer un mail chiffré

1ère chose a faire une fois la paire de clés créée, s'envoyer a soi-meme un mail chiffré. Un mail c'est quoi ? un fichier, rien de plus, que l'on s'envoit par internet. Donc ici nous appelerons notre mail *test.txt*
Notez que la methode décrite ci-dessous est brute, il est possible avec des clients de messagerie evolués tels que Evolution, de gerer le chiffrage/signature de mail “a la volée”, sans passer par la ligne de commande (cf. annexes)

Pour que ce soit plus propre, j'ai créé un dossier Gpg dans mon home directory. Toutes les actions qui vont suivre s'y situeront.

Tout d'abord, on va ecrire notre mail (prenez votre editeur de texte favori :)

```
cho7@cho7land:~/Gpg$ vi test.txt
```

Mettez ce que vous voulez dedans on s'en fiche pas mal :)
Ensuite passez votre fichier a la moulinette gpg, et passez en meme temps les arguments -e (pour encrypt) et r pour spécifier la clé publique de votre trousseau a utiliser pour chiffrer
Vous noterez encore une fois que la situation dans laquelle vous envoyez un mail chiffré a un destinataire n'utilisant pas Gpg est impossible, car il faut impérativement la clé publique du destinataire dans son trousseau au moment du chiffrage !

```
cho7@cho7land:~/Gpg$ gpg -er cho7@dlfp.org test.txt
cho7@cho7land:~/Gpg$ ls -l
total 8
-rw-r--r--  1 cho7 cho7  50 2004-09-29 15:00 test.txt
-rw-r--r--  1 cho7 cho7 389 2004-09-29 15:01 test.txt.gpg
```

Oh tiens, c'est quoi ce nouveau fichier test.txt.gpg qui est apparu subitement ?? Notre mail chiffré peut etre ? Allez lire son contenu vous verrez bien le sens du mot “chiffré” :)

Bon dès lors on peut supprimer notre fichier test.txt; il ne nous est plus utile...

```
cho7@cho7land:~/Gpg$ rm test.txt
```

Dechiffrer un mail

Et là on fait quoi ? d'abord on va faire un truc que plus tard vous ne pourrez plus faire : dechiffrer le fichier test.txt.gpg !
Et oui, pour dechiffrer ce fichier, il faut utiliser la clé secrete du destinataire, que vous n'aurez jamais en temps normal, sauf là, puisque vous vous etes envoyé ce fichier a vous meme, avec votre clé publique, donc cette fois, et seulement cette fois, votre clé secrete va vous permettre de dechiffrer le fichier que vous avez envoyé (alors que normalement seul votre destinataire pourra déchiffrer votre mail, pas vous !)

```
cho7@cho7land:~/Gpg$ gpg test.txt.gpg
Vous avez besoin d'un mot de passe pour déverrouiller la clé secrète pour
l'utilisateur: "Cho7 Kipu <cho7@dlfp.org>"
clé de 1024 bits ELG-E, ID 51741FEF, créée le 2004-09-29 (ID clé principale 0C792AD8)
Password :
saisissez le mot de passe de votre clé secrete
gpg: chiffré avec une clé de 1024 bits ELG-E, ID 51741FEF, créée le 2004-09-29
      "Cho7 Kipu <cho7@dlfp.org>"
cho7@cho7land:~/Gpg$
cho7@cho7land:~/Gpg$ ls -l
total 8
-rw-r--r--  1 cho7 cho7  50 2004-09-29 15:13 test.txt
-rw-r--r--  1 cho7 cho7 389 2004-09-29 15:01 test.txt.gpg
```

Oh bah tiens, un revenant... On l'avait pas viré lui ? Bon bah on a fait l'opération en sens inverse, vous avez retrouvé le fichier d'origine ! mais encore une fois, je ne le repeterai jamais assez, ce sera dans l'avenir possible uniquement avec un fichier mail que vous avez RECU, ou alors ENVOYé A VOUS MEME !

Donc dans l'absolu, vous envoyer ce fichier .gpg avec le client de messagerie de votre choix, et votre destinataire (ou vous meme, il suffit d'inverser les roles), n'aura qu'a enregistrer le fichier .gpg recu sur son disque dur, ouvrir un shell, taper gpg mon_fichier.gpg et fournir son mot de passe de clé privée, et hop, le fichier sera déchiffré dans le meme dossier sous le nom “mon_fichier”

Vous pouvez donc désormais ecrire un fichier, le chiffrer a l'aide de la clé publique de votre destinataire, et envoyer le fichier .gpg comme bon vous semble.
Vous pouvez aussi désormais recevoir un fichier .gpg par mail, et vous dire, tiens j'ai recu un mail chiffré, je vais aller le déchiffrer comme on m'a dit plus haut :)

Ce que vous ne savez pas encore :
“On m'a dit que pour envoyer un mail chiffré a quelqu'un je devais avoir sa clé publique, mais jla trouve où ??”
Très bonne question, si votre destinataire n'utilise pas GPG, il n'en a pas, donc pas de mail chiffré. S'il en a une, alors il doit vous la communiquer.
Pour celà, plusieurs methodes, la plus sûre etant d'etre face a la personne, et qu'il vous remette un papier avec le fingerprint de sa clé publique noté dessus, et le nom d'un serveur de clé sur lequel sa clé est repertorié (n'aillez pas peur des termes, ce sera expliqué plus bas)
Ainsi, il va vous remettre par exemple :

Empreinte de la clé = 999A 46D0 5E2D D616 2974 DD86 D20C AA55 0C79 2AD8
sur http://pgp.mit.edu

Ok c'est cool, j'en fais quoi moi de ce bou de papier ??
C'est facile, vous allez tapper presque la meme chose que pour vous enregistrer sur le serveur de clé . Notez ici que AAAAAAA correspond aux 8 derniers caracteres du finger print, soit votre id a 8 chiffres.

```
cho7@cho7land:~/Gpg$ gpg --keyserver pgp.mit.edu --recv-keys AAAAAAAA
gpg: clé AAAAAAAA: clé publique "Plop Plop <plop@plop.org>" importée
gpg:        Quantité totale traitée: 1
gpg:                    importée: 1
```

En cas de doublons (ce qui ne m'est jamais arrivé), il vous proposera sans doute 2 clés publiques et leur fingerprint, vous pourrez donc choisir la bonne (bien que de toute facon, les noms associés aux 2 clés risque a 99% d'être différent, mais sait-on jamais, le fingerprint est la seule solution pour distinguer 2 clés avec le meme id 8 chiffres et le meme nom de propriétaire...)

Vous pouvez ensuite faire un

```
cho7@cho7land:~/Gpg$ gpg –list-keys
```

pour constater que la clé a bien été importée dans votre trousseau.
Vous pouvez désormais envoyer des mails chiffrés à plop@plop.org !

Prevenir ses contacts

Vous pouvez dès maintenant commencer a faire suivre votre clé publique, en donnant votre id et votre serveur de clé (facultatif, mais je vous ai dit que parfois les serveurs n'etaient pas a jour, donc je préfère donner celui ou j'ai moi meme envoyer ma clé)
Vos contacts pourront donc (pour ceux etant equipés en GPG) vous envoyez des mails chiffrés dès qu'ils auront importé votre clé publique dans leur trousseau.
De votre coté, récuperer aussi les clés publiques de vos contacts, afin de pouvoir, vous aussi, leur envoyer des mails chiffrés !

Signer ses mails

“Signer ses mails ? ca sert a quoi ?”
Tout d'abord, chiffrer ses mails a un inconvénient, a savoir que le destinataire doit saisir un mot de passe pour déchiffrer, et parfois ca peut enerver. D'autant que beaucoup de gens se serve de gpg non pas pour chiffrer, mais juste pour signer leur message. De plus, il est possible d'envoyer des mails signés, meme a des destinataires n'utilisant pas GPG !
Signer un message, c'est lui joindre un certificat, attestant que la personne qui a envoyé le mail, et bien la personne qui apparait dans le champs “From : “ (ou “De: “)
N'importe quel bouzin sait en effet comment envoyer un mail anonyme en changeant le champ From et en envoyant des mails au nom de bilou@microsoft.com pour se la jouer hacker, ou encore les milliers de virus se propageant via les carnets de contacts et qui abusent de la crédulité des gens qui pensent ouvrir un “jeu” envoyé par leur copain, alors que c'est en fait le virus qui a fait une “usurpation d'identité”
Signer ses mails ne requiert donc pas de dechiffrage de la part du destinataire, pas plus qu'un mot de passe, lorsqu'il ouvrira son mail avec gpg, ca lui indiquera simplement qu'il a été signé numériquement, et lui dira si oui ou non la signature est valide.
Ainsi si le contenu du mail est modifié, la signature ne sera plus bonne, et le destinataire en sera averti, pour lui dire “Attention, signature invalide, le contenu n'est peut etre pas sur, car ou bien il a été altéré, ou bien s'il n'y a pas de signature du tout, et auquel cas si vous prenez l'habitude de signer vos mails, alors votre contact se doutera d'une entourloupe en se disant :

“Tiens c'est marrant, lui normalement il signe ses mails, et pas là... Peut etre un virus... J'ouvre pas cette drole de pièce jointe pour la peine...”
Et hop, un virus de plus qui ne se propagera pas grace a moi.

Voilà pour la théorie, venons en a la pratique...
Reprenons notre fichier test.txt du début.

Avant toute chose, sachez qu'il existe 2 methodes pour signer son mail. Le certificat peut etre fusionné au mail lui meme, ce qui le rend illisible pour les simples clients de messageries non compatibles, ou bien le certificat peut etre fourni en piece jointe (un fichier d'extension .sig)
Auquel cas, lors de la lecture du mail par un client de messagerie non compatible, le message apparaitra normalement, et la signature apparaitra en pièce jointe sous la forme d'un fichier d'extension .sig)
Pour les clients de messageries evolués tels que Evolution, le fichier sera normalement détécté, et un pied de page en bas du mail sera rajouté pour signifier que le mail est signé numériquement et le logiciel indiquera aussi si cette signature est valide ou non.

Notez que pour signer un mail, il faut impérativement choisir une clé pour savoir “au nom de qui nous signons”, et en saisir le mot de passe de la clé secrète, ce qui “prouve” que c'est bien nous qui avons signé, et pas un virus ou mon voisin cherchant a se faire passer pour moi :-)

1ere methode – signature fusionnée au mail

```
gpg -su cho7@dlfp.org test.txt
```

va générer un fichier test.txt.gpg, qui sera signé (et aussi “chiffré” pour les clients de messagerie non compatible, mais ca ce n'est pas voulu :-)
Lors du dechiffrage en ligne de commande, l'indication de la signature sera faite naturellement :

```
cho7@cho7land:~/Gpg$ gpg test.txt.gpg
gpg: Signature faite mer 29 sep 2004 16:58:55 CEST avec la clé DSA ID 1E8F63AF
gpg: Bonne signature de "Cho7 Kipu <cho7@dlfp.org>"
gpg: vérifier la base de confiance
gpg: vérification à la profondeur 0 signé=1 ot(-/q/n/m/f/u)=0/0/0/0/0/3
gpg: vérification à la profondeur 1 signé=0 ot(-/q/n/m/f/u)=0/0/0/1/0/0
```

2eme methode – signature en pièce jointe

```
gpg -bu cho7@dlfp.org test.txt
```

Ne va absolument pas modifier le fichier test.txt, mais va créér un fichier test.txt.sig, qu'il faudra joindre en meme temps que le mail lui meme.
La cette fois, les clients de messageries non-compatible avec gpg, se contenteront d'afficher la pièce jointe...

Testez l'altération des données vous verrez :
Si vous souhaitez verifier simplement l'intégrité d'une signature tappez :

```
cho7@cho7land:~/Gpg$ gpg --verify test.txt.sig
gpg: Signature faite mer 29 sep 2004 17:31:00 CEST avec la clé DSA ID 0C792AD8
gpg: Bonne signature de "Cho7 Kipu <cho7@dlfp.org>"
```

maintenant on va modifier un seul caractere de notre fichier test.txt et on va reverifier la signature :

```
gpg: Signature faite mer 29 sep 2004 17:31:00 CEST avec la clé DSA ID 0C792AD8
gpg: MAUVAISE signature de "Cho7 Kipu <cho7@dlfp.org>"
```

Voilà, le message a été altéré, donc la signature numérique n'est plus bonne, CQFD

Réseaux de confiance

Qu'est ce que c'est ? Tout d'abord allez sur n'importe quel seveur de clé et faite une recherche sur bill gates ou jacques chirac, et vous verrez que plein de gens ont référencés des clés publiques loufoques. D'ou la question : comment distinguer les vrais clés des fausses ? En regardant celles qui ont le plus de signatures. Il est en effet possible de signer la clé de quelqu'un, comme pour dire “Moi cho7, atteste que la clé 12345678 est bien celle de plop, hop je signe avec ma clé privée” Plus une clé publique a de signatures, plus elle est fiable. De meme, il est possible de donner un niveau de confiance a une clé, pour dire par exemple “Moi cho7 ne fait pas trop confiance a la clé 12345678, car jtrouve que son propriétaire n'a pas trop saisi l'interet des réseaux de confiance et signe un peu les clés des autres à la légère, et qu'il peut mettre en péril le réseau de confiance en signant la clé d'une personne sans s'assurer que ce soit bien elle” (notez que je peux attester la clé en la signant, tout en nuancant sur la confiance que je porte au PORTEUR de la clé publique)

Pour donner ma confiance a la clé 12345

```
gpg –edit-key 12345
```

Là on tappe

```
sign
```

suivit d'un indice de validité (3 etant le summum, j'en suis sûr et certain, c'est bien la bonne personne)
Signer une clé nécéssite le mot de passe de la clé secrete (sinon n'importe quelle personne ayant accès a votre pc pourrait signer les clés sans votre consentement)

Pour donner un niveau de confiance a une clé:
toujours dans le mode d'edition (gpg –edit-key 12345), tappez

```
trust
```

puis un indice de confiance.
Validez par votre mot de passe de clé secrete.
une fois vos clés signées et “trustés”, vous pouvez mettre a jour le serveur de clé, pour que les prochaines personnes qui vont importés les clés de vos contacts, puissent voir que vous-meme avait approuvées la ou les signatures.
pour se faire :

```
gpg –keyserver pgp.mit.edu –send-keys
```

Et confirmez.
Pour mettre a jour son trousseau (a faire régulièrement, car les clés de votre trousseau peuvent etre signées et trustées par d'autres tous les jours, faites

```
gpg –keyserver pgp.mit.edu –refresh-keys
```

Là, gpg va vous indiquer si des clés ont été signées par de nouvelles personnes ou bien si elles ont été révoquées.
Vous pouvez lister les clés de votre trousseau ainsi que les signatures récoltées pour chacune en tappant :

```
gpg –list-sigs
```

Configurer son client de messagerie

Bon nous voilà a la 10ème page, nous allons enfin pouvoir parler de client de messagerie, afin de rendre GPG beaucoup plus convivial :-)

Pour Evolution

C'est bon vous avez créé votre paire de clé ? Vous voulez dire a evolution que vous etes un grand garcon et que vous voulez chiffrer/signer vos mails ?
Ok, c'est pas sorcier vous allez voir.

Demarrer Evolution et aller dans le menu Outils> Parametrages
Sélectionnez le compte qui va utiliser GPG (il doit avoir une clé dans votre trousseau portant cette adresse mail !) et cliquez sur Editer, et enfin rendez-vous sur l'onglet Securité.

Vous l'aurez compris, il suffit d'indiquer l'id 8 chiffres de la clé secrete correspondant au compte evolution, vous pouvez eventuellement mettre des options comme celle de signer automatiquement tout courrier sortant de ce compte, ou de toujours chiffre le mail pour vous-meme, ce qui vous permettra de RELIRE par la suite les mails cryptés contenus dans le dossier Elements Envoyés

Validez tout, et vous revoilà sous la fenetre principale d'evolution.

Désormais, pour envoyer un mail chiffré il vous suffit de rédiger un nouveau mail normalement, mais avant de l'envoyer, séléctionnez le menu Sécurité de votre message, et vous pourrez alors choisir de chiffre et/ou signer votre message, a condition que le compte choisi pour le champ From soit un compte paramatré pour tourner avec GPG (cf un peu plu haut)

Pour verifier a nouveau le principe de “Vous ne pouvez dechiffrer un mail qu'avec la clé secrete du destinataire” essayer donc d'aller dans le dossier “Envoyés” et essayer de lire le mail que vous venez d'envoyer a votre contact. Evolution ne pourra pas vous l'ouvrir, car il vous faudra la clé secrete du destinataire et son mot de passe, il est donc impossible de réutiliser un message envoyé s'il a été chiffré, il fallait penser a en faire une sauvegarde avant :)

Pour Sylpheed

pour le client de messagerie Sylpheed, c'est a peu près la meme chose :
Menu Configuration des comptes > Edition des comptes > on séléctionne le compte > Editer > onglet Confidentialité > Indiquez manuellement la clé > on saisit l'id a 8 chiffres > On valide (on peut là aussi regler des trucs comme la signature systematique des mails)
Avant d'envoyer le mail : Menu Message > Chiffrer et/ou Signer

Pour Mutt

La distribution de Mutt est fournie avec des exemple qui font tout ce qui faut. Vous devez avoir dans votre fichier .muttrc quelque chose comme ce qui suit (sinon ajoutez-le):

```
set     pgp_sign_as=0xFFFFFFFF          # indiquer ici l'ID de votre clef

set     pgp_timeout=1800                # Mutt retient votre passphrase quand vous la tapez.
                                        # Au bout de combien de temps (secondes) doit-il l'oublier ?
```

```
set     pgp_autosign                    # signer automatiquement tous les mails sortant
set     pgp_replyencrypt                # chiffrer automatiquement les réponses aux mails chiffrés
set     pgp_replysign                   # signer automatiquement les réponses aux mails signés
set     pgp_replysignencrypted          # signer automatiquement les réponse au type sig/crypt

set     pgp_retainable_sigs             # signer les différentes parties du message indépendemment
                                        # (si vous attachez une pièce jointe, celle-ci sera signée
                                        # séparement, ce qui permet au destinataire de prouver que
                                        # vous l'avez signé sans divulguer pour autant le reste du
                                        # message)

set     pgp_sort_keys=trust             # tri des clefs

#set pgp_create_traditional             # utiliser l'ancien format mime (déconseillé, obsolète)

# les lignes ci-dessous ne devraient pas être modifiées,
# sauf cas particulier (par exemple pour ajouter un chiffre en module à gpg).

        # comment déchiffrer le type mime application/pgp
set     pgp_decode_command=\
        "gpg --charset utf-8 %?p?--passphrase-fd 0? --no-verbose --batch --output - %f"
        # verifier une signature pgp/mime
set     pgp_verify_command=\
        "gpg --no-verbose --batch --output - --verify %s %f"
        # dechiffrer une pièce jointe pgp/mime
set     pgp_decrypt_command=\
        "gpg --passphrase-fd 0 --no-verbose --batch --output - %f"
        # créer un message signé pgp/mime
set     pgp_sign_command=\
        "gpg --no-verbose --batch --output - --passphrase-fd 0 \
        --armor --detach-sign --textmode %?a?-u %a? %f"
        # créer un message signé application/pgp
        # (obsolète, utilisé si pgp_create_traditional est positionnée)
set     pgp_clearsign_command=\
        "gpg --no-verbose --batch --output - --passphrase-fd 0 \
        --armor --textmode --clearsign %?a?-u %a? %f"
        # créer un message chiffré pgp/mime
set     pgp_encrypt_only_command=\
        "pgpewrap gpg -v --batch --output - --encrypt --textmode \
        --armor --always-trust -- -r %r -- %f"
        # créer un message signé et chiffré pgp/mime
set     pgp_encrypt_sign_command=\
        "pgpewrap gpg --passphrase-fd 0 -v --batch --output - --encrypt \
        --sign %?a?-u %a? --armor --always-trust -- -r %r -- %f"
        # Importer une clef dans votre trousseau  de clefs publiques
set     pgp_import_command="gpg --no-verbose --import -v %f"
        # Exporter une clef de votre trousseau  de clefs publiques
set     pgp_export_command="gpg --no-verbose --export --armor %r"
        # vérifier une clef
set     pgp_verify_key_command=\
        "gpg --no-verbose --batch --fingerprint --check-sigs %r"
        # Afficher les clefs de votre trousseau  de clefs publiques
set     pgp_list_pubring_command=\
        "gpg --no-verbose --batch --with-colons --list-keys %r"
        # Afficher les clefs de votre trousseau  de clefs privées
set     pgp_list_secring_command=\
        "gpg --no-verbose --batch --with-colons --list-secret-keys %r"
        # recevoir les clefs du serveur de clefs.
set     pgp_getkeys_command=""
```

```
        # Motif pour les bonnes signatures (à changer avec la locale).
set     pgp_good_sign="`gettext -d gnupg -s 'Bonne signature de "' | tr -d '"'`"
```

L'envoi des mails se fait comme auparavant, simplement avant de presser sur 'y' pour envoyer le message une fois celui-ci composé, vous avez la possibilité d'utiliser 'p' pour sélectionner le menu pgp: vous avez alors le choix entre 'c' pour chiffrer uniquement, 's' pour signer uniquement, 'd' pour signer ET chiffrer, 'e' pour changer de clef privée utiliser pour la signature (par rapport à la clef indiquée par la variable "pgp_sign_as"), et 'o' pour ne pas utiliser gpg pour ce mail.

Quand vous tapez finalement sur la touche d'envoi du message ('y' par défaut), si vous avez demandé que ce message soit signé et si vous n'avez pas entré votre phrase de passe récemment, Mutt vous demander alors de la taper, sinon le message par directement.

Note: si vous faites une erreur en tapant votre phrase de passe, Mutt va essayer de signer le message mais gpg va renvoyer une erreur car la phrase n'est pas bonne, Mutt revient alors à l'écran d'envoi de message: taper alors <Ctrl>-f pour effacer la phrase de passe en mémoire, ainsi quand vous appuierz à nouveau sur 'y', Mutt vous redemandera la phrase.

Pour la réception, il n'y a aucune différence si ce n'est que l'on vous demande votre phrase de passe pour les mails chiffrés que vous recevez. Dans le pager, des balises indiquent les parties chiffrées et ou signées des messages si nécessaire, ainsi que, dans le cas des messages signés le status de la signature (bonne, mauvaise, vérification impossible car clef absente du trousseau).

Astuce: avec la configuration telle que décrite ci-dessus, les mails que vous chiffrez ne le sont qu'avec la clef du destinataire, ce qui fait que bien qu'ils soient lors de l'envoi copié dans la boîte sent-mail, vous ne pouvez pas les relire ultérieurement. La solution consiste à indiquer à gpg de chiffrer tous les messges également avec votre clef. Cela se fait en ajoutant dans votre fichier de configuration de gpg (~/.gnupg/options) une ligne

```
encrypt-to FFFFFFFF
```

si FFFFFFFF est l'ID de votre clef.

Pour Kmail

Kmail possède un manuel en ligne en français qui explique comment l'utiliser avec gpg.
Néanmoins, si vous êtes pressé, voici en quelques mots comment faire...
(Ces instructions ont été écrites à partir de Kmail 1.5, mais devraient pouvoir être adaptées facilement pour toutes les versions)
Lancez Kmail.
Dans le menu **Configuration** choisissez **Configurer Kmail**. Dans la fenêtre qui s'affiche alors, cliquez sur l'icône **Sécurité** et allez dans l'onglet **OpenPGP**. Là, dans la liste des **Outils de Cryptage** (sic) choisissez **GNU Privacy Guard (GnuPG)**. Sélectionnez également en dessous les options que vous désirez utiliser.
Revenez ensuite à l'icône **Identités**. Sélectionner votre identité et cliquez sur **modifier**. Dans l'onglet **Options Avancées**, indiquez votre clef pgp (**modifier**).

Par la suite, l'utilisation est très simple: il suffit, lorsque l'on compose un message, d'activer les icônes pour chiffrer (cadenas) et signer (plume et feuille de papier). Une boîte de dialogue apparaît lorsque c'est nécessaire pour que l'on entre sa phrase de passe.

GPG sans utiliser de pièce jointe

Vous aurez sans doute remarqué que gpg implique d'envoyer ses messages chiffrés en pièce jointe d'un message normal (transparent si le client connait gpg, mais si vous envoyez un mail chiffré a partir d'evolution vers un client outlook, ce dernier affichera un mail vide, et des pièces jointes étranges contenant le mail lui meme, et eventuellement sa signature...)
Vous pouvez demander a ne pas utiliser de pièce jointe, et baliser votre message de balises PGP, afin de le transmettre directement tel quel (utile sur les mailings listes ou les pièces jointes n'apparaissent pas)
Si c'est pas clair, voici un exemple :

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

ceci est un message signé numériquement !!
plop plop
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.5 (GNU/Linux)

iD8DBQFBfr0a0gyqVQx5KtgRAmrQAJ0ZrlKrx595ZmAAe1LEpS5/2KN39wCghHvN
5yhdIJcAOplRHTKriDYRWrQ=
=B0DP
-----END PGP SIGNATURE-----
```

Pour réaliser ce type de mail, saisissez dans un shell :

```
gpg –clear-sign -u cho7@dlfp.org
Vous avez besoin d'un mot de passe pour déverrouiller la clé secrète pour
l'utilisateur: "Cho7 Kipu <cho7@dlfp.org>"
clé de 1024 bits DSA, ID 0C792AD8, créée le 2004-09-29
```

Une fois le mot de passe saisi, vous pouvez saisir le corps de votre message.
Une fois votre message saisi, faite un ctrl+D pour valider et generer le message signé.
Vous n'avez plus qu'a copier-coller le message ainsi balisé dans votre client messagerie, et à l'envoyer.

Si vous recevez un message de cette forme, vous pouvez tapper dans un shell

```
gpg
```

sans argument, et ensuite copier-coller le message recu, et valider 2 fois par ctrl+D.
Si le message est signé, gpg vous le dira, et s'il est chiffré, gpg vous demandera en plus votre mot de passe de clé secrète.

Vous pouvez réaliser un mail chiffré sans pièce jointe de la même manière :

```
gpg -ear destinataire@plop.com
```

puis ctrl + D

pour dechiffrer :

```
gpg
```

on copie colle le message, on saisit le mot de passe de sa clé secrète, puis ctrl + D

C'est fini !

Voilà, c'est la fin de ce howto dans lequel je vous ai epargné la longue philosophie GnuPG et ses réseaux de confiance, mais quelques recherches avec notre ami google devrait vous apporter toutes les réponses aux questions que vous vous posez !
Donc merci a vous de m'avoir lu, et a une prochaine fois peut etre...
Je pense que vous savez désormais où me contacter !!

Petit rappel sur la licence de ce document :
Celui ci est parfaitement libre au niveau de la diffusion, donc imprimez, faites suivre autant que vous voulez, mais pour la modification il faut initialement passer par moi (en plus je réagis vite aux suggestions profitez en), ca me permet de recentraliser les demandes de chacun et d'en faire profiter tout le monde en modifiant le document.

Quelques liens :

Le site officiel de GnuPG : http://www.gnupg.org
Mailchiffr : http://mailchiffr.sourceforge.net/
Gnus : http://www.emacswiki.org/cgi-bin/wiki/CategoryGnus
Editer un fichier chiffrer directement avec Vi : http://www.vim.org/scripts/script.php?script_id=661
Un tutorial très bien fait de J.Francoz et G. Dartiguelongue : http://francoz.net/doc/gpg/gpg.html
Un editeur de clé en mode graphique pour Gnome, seahorse (merci à Etienne Bersac) : http://seahorse.sourceforge.net/

Et un remerciement tout particuliers aux linuxfriens du site http://linuxfr.org
pour leurs critiques et suggestions lors de l'élaboration de ce document,
ainsi qu'à Nicolas Bernard, rédacteur de la partie concernant la configuration de Mutt et Kmail

Memento GPG

Général

Créer une paire de clés :

```
gpg –gen-key
```

Créer un certificat de révocation :

```
gpg –gen-revoke [clé] > fichier
```

Revoquer sa clé :

```
gpg –import [fichier_contenant_certificat_de_revocation_généré_précédemment]
```

puis on met a jour le serveur en renvoyant notre trousseau au complet

```
gpg –keyserver pgp.mit.edu –send-keys
```

ou bien simplement la clé révoquée :

```
gpg –keyserver pgp.mit.edu –send-keys [clé]
```

Lister son trousseau de clé publiques :

```
gpg –list-keys (--list-secret-keys pour les clés privées)
```

S'inscrire a un serveur de clé :

```
gpg –keyserver pgp.mit.edu –send-keys [clé]
```

Ajouter une clé publique a son trousseau :

```
gpg –keyserver pgp.mit.edu –recv-keys [id à 8 chiffres de la clé a recevoir]
```

Mettre à jour son trousseau de clés publiques (afin de savoir si certaines ont été signées ou révoquées)

```
gpg –keyserver pgp.mit.edu –refresh-keys
```

puis pour lire les nouvelles signatures

```
gpg –list-sigs
```

Si des clés du trousseau ont été révoqués, on peut alors les supprimer :

```
gpg –delete-keys [clé]
```

et pour supprimer une clé secrete de son trousseau :

```
gpg –delete-secret-keys [clé]
```

Sauver des clés ou des clés secretes (pour les importer dans un autre trousseau de clé situé sur un autre PC par exemple)

```
gpg --export -a [clé] > [fichier_clé] (si l'on précise pas de clé, tout le trousseau est stocké dans le fichier, pour les clés secrètes –export-secret-keys)
```

puis on réimporte dans le trousseau de destination via

```
gpg –import [fichier_clé]
```

Edition d'une clé publique de son trousseau

```
gpg –edit-key [clé]
```

puis

```
sign
```

ou

```
trust
```

Une fois son trousseau modifié, on met a jour le serveur de clé :

```
gpg –keyserver pgp.mit.edu –send-keys
```

Traitement de fichiers

Signer un fichier :

```
gpg -su [clé] [nom du fichier]
```

ou

```
gpg -bu [clé] [nom du fichier]
```

pour séparer la signature du fichier (création d'un fichier .sig)

Chiffrer un fichier :

```
gpg -er [clé du destinataire] [nom du fichier]
```

génère un fichier .gpg

Signer et Chiffrer un fichier :

```
gpg -su [clé] -er [clé du destinataire] [nom du fichier]
```

génère un fichier .gpg

Dechiffrer un fichier :

```
gpg [fichier chiffré]
```

regénère le fichier d'origine dans le repertoire courant

Verifier la signature d'un fichier :

```
gpg –verify [fichier signé ou fichier original + fichier .sig]
```