Petite introduction à GPG

Pauline POMMERET

Encadrée par JBEN

16 février 2013

Communiquer de façon sûre

- recevoir un email de quelqu'un ne veut pas dire qu'il nous l'a effectivement envoyé ;
- un email est envoyé en clair sur le réseau et les informations envoyées peuvent être lues par n'importe qui.

Le principe du chiffrement est de transformer à l'aide d'une clef un message clair en un message incompréhensible pour que celui qui ne dispose pas de la clef de déchiffrement.
On distingue trois types d'algorithmes utilisés pour le chiffrement :

1. algorithmes de chiffrement simples (code de CÉSAR) ;
2. algorithmes de cryptographie symétrique fondés sur la présence d'une unique clef pour chiffrer et déchiffrer nécessitant autant de clef que de correspondants (AES) ;
3. algorithmes de cryptographie asymétrique fondés sur la présence de 2 clefs, une publique (partageable) et une privée (RSA, DSA).

OpenPGP

OpenPGP est un format de cryptographie qui définit le format des messages, signatures ou certificats que peuvent s'envoyer des logiciels.
C'est un format pour l'échange sécurisé de données.

GNU Privacy Guard

C'est une implémentation du standard OpenPGP, procédé de chiffrement à clef publique. C'est un logiciel très stable, distribué sous la licence GNU GPL et est souvent inclus d'origine sur les systèmes d'exploitation GNU/Linux.

GnuPG est un système cryptographique à clef publique caractérisé par :

- une clef *publique*, distribuée à toutes les personnes avec qui l'utilisateur souhaite communiquer ;
- une clef *privée*, gardée jalousement secrète.

--gen-key

```
pauline@samothrace $ gpg --gen-key
gpg (GnuPG) 1.4.12; Copyright (C) 2012 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Sélectionnez le type de clef désiré :
   (1) RSA et RSA (par défaut)
   (2) DSA et Elgamal
   (3) DSA (signature seule)
   (4) RSA (signature seule)
Quel est votre choix ?
```

- RSA (RIVER SHAMIR ADLEMAN), le plus utilisé dans le commerce électronique ;
- DSA (*Digital Signature Algorithm*) ;
- utiliser RSA (ANSSI : taille minimale de 4096 bits pour usage au delà de 2020).

Taille de la clef

Principe :

- standard entre 2048 et 4096 ;
- plus la clef est longue, plus elle est dure à casser ;
- plus la clef est longue, plus elle est lourde (mais chiffrement hybride) ;
- plus la clef est longue, plus elle est longue à générer (artéfact : `cp Musique/ Musique2/`).

Date d'expiration

Validité d'une clef : temps au bout duquel les correspondants ne pourront plus utiliser cette clef pour chiffrer des données et vérifier les signatures.

```
Veuillez indiquer le temps pendant lequel cette clef devrait être valable.
         0 = la clef n'expire pas
      <n>  = la clef expire dans n jours
      <n>w = la clef expire dans n semaines
      <n>m = la clef expire dans n mois
      <n>y = la clef expire dans n ans
Pendant combien de temps la clef est-elle valable ? (0)
```

Comment choisir ?

- 0 ou temps de vie illimité peu sécurisé : perte clef privée, vol, oubli du mot de passe,...
- possibilité de prolongement temps de vie avant expiration.

Identité de la clef

```
Une identité est nécessaire à la clef ; le programme la construit à partir
du nom réel, d'un commentaire et d'une adresse électronique de cette façon :
   « Heinrich Heine (le poète) <heinrichh@duesseldorf.de> »

Nom réel :
```

Ce sont les informations qui apparaîtront au moment de la vérification des signatures. Attention à l'identité créée et au contexte.

Phrase de passe

À bien choisir !

- **seule** protection de la clef privée si quelqu'un possède le fichier contenant la clef privée, c'est le point faible de GnuPG ;
- ne devrait pas contenir de mot du dictionnaire ;
- devrait mélanger la casse caractères alphabétiques ;
- devrait utiliser des caractères non alphabétiques ;
- taille illimitée.

Générer un certificat de révocation

`--gen-revoke`

`--gen-revoke` génère un certificat de révocation signifiant qu'on ne peut plus utiliser la clef publique. 2 types différents :

- certificat de perte en cas d'oubli du mot de passe ou de perte de la clef ;
- certificat de compromission si la clef privée est compromise.

Voici l'utilisation :

```
pauline@samothrace $ gpg --output revocation_type.asc --gen-revoke identifiant_clef
```

Utilité

Une bonne gestion des clefs est cruciale pour être certain que personne ne lise les messages chiffrés, en émette d'autres. Cela permet d'être sûr de son trousseau et de garantir l'intégrité du trousseau des autres.

Gérer la paire de clefs

Afficher les caractéristiques de la paire

Une clef publique est composée de :

- portion publique de la clef principale de signature ;
- portions publiques des clefs secondaires de signature et de chiffrement ;
- identifiants utilisés pour associer la clef à l'utilisateur (nom, commentaire optionnel, adresse mail, date de création, date d'expiration, degré de confiance,...).

Pour afficher ces informations :

```
pauline@samothrace $ gpg --edit-keys pommeret@crans.org
```

Gérer la paire des clefs

Intégrité des clefs

La distribution des clefs publiques engendre un risque de falsification (substitution clefs, modifications identifiants utilisateurs).
Pour protéger une clef publique, on utilise la partie privée de la clé principale pour signer les composantes publiques et l'identifiant utilisateur : c'est une **auto-signature**.

```
gpg> check
uid  Pauline Pommeret <paulinepommeret@gmail.com>
sig!3        CF875FE1 2012-10-04  [autosignature]
uid  Pauline Pommeret (ma première clé !) <pommeret@crans.org>
```

Gérer la paire des clefs

Ajouter des composantes à une clef

On peut vouloir ajouter différentes composantes :

- identifiants utilisateurs avec `adduid` en cas de multiples identités ;
- sous-clefs avec `addkey` car changer de clef principale nécessite de refaire les certifications, et il est recommandé de changer de sous-clefs régulièrement (3 ans) et d'utiliser des sous-clefs différentes sur des machines différentes.

Gérer la paire des clefs

Retirer des composantes à une clef

Les sous-clefs et les identifiants utilisateurs peuvent être effacés :

1. sélection de l'item à effacer par les sélecteurs `key` et `uid` (`key 2` sélectionne la seconde sous-clef) ;
2. effacement de l'item sélectionné par `delkey` ou `deluid`.

L'effacement complique la distibution des clefs. Lors de l'import ou de l'envoi sur un serveur de la clef publique, la fusion restaure les éléments effacés.

Gérer la paire des clefs

Révoquer les composantes d'une clef

On peut révoquer différentes composantes :

- pour une sous-clef, on utilise `revkey` après avoir sélectionné la sous-clef (auto-signature de révocation) ;
- pour une signature, on utilise `revsig`, l'interface révoquée ;
- pour un identifiant utilisateur, on révoque son auto-signature.

La révocation est toujours visible lors distribution et màj de la clef publique. Cela garantit que les autres aient une version intègre de la clef.

Gérer la paire des clefs

Mettre à jour la date d'expiration de la clef

```
gpg> expire
Modification de la date d'expiration de la clef principale.
gpg: Attention : aucune identité n'a été définie comme principale. Cette commande
           risque de rendre une autre identité principale par défaut.
Veuillez indiquer le temps pendant lequel cette clef devrait être valable.
         0 = la clef n'expire pas
      <n>  = la clef expire dans n jours
      <n>w = la clef expire dans n semaines
      <n>m = la clef expire dans n mois
      <n>y = la clef expire dans n ans
Pendant combien de temps la clef est-elle valable ? (0)
```

`expire` efface la dernière auto-signatur et la remplace. La dernière auto-signature fait référence pour ceux qui ont importé la clef.

Signer une clef

Une clef peut être validée en vérifiant son empreinte. En la signant, on certifie qu'elle est valide. Pour visualiser l'empreinte de la clef on utilise `--fingerprint` ou `fpr` en édition.

L'empreinte de la clef est vérifiée avec son propriétaire, on s'assure ainsi qu'on a une copie correcte de la clef. On s'assure également de l'identité de la personne que l'on a en face de soi.

Pour signer, on utilise alors la commande `sign` sur la clef que l'on veut éditer.

Confiance dans le propriétaire de la clef

Niveaux de confiance

Il existe 5 niveaux de confiance pour les propriétaires de clefs :

- `1` ou *unknown*, on ne sait rien de la façon dont la personne signe ses clefs (valeur par défaut) ;
- `2` ou *none*, on sait que la personne ne vérifie pas soigneusement avant de signer ;
- `3` ou *marginal*, on sait que le propriétaire a conscience de ce qu'il fait quand il signe ;
- `4` ou *full*, le propriétaire sait parfaitement ce qu'il fait et une signature de lui a la même valeur que la votre ;
- `5` ou *réservé exclusivement à ses propres clefs*.

Confiance dans le propriétaire de la clef

`trust`

Le niveau de confiance est une information personnelle et privée, enregistrée sur une base de donnée distincte.
Pour définir le niveau de confiance dans un propriétaire, on utilise l'éditeur de clef avec la commande `trust`.

Utiliser la confiance pour valider une clef

Paramètres du réseau de confiance par défaut

Une clef est considérée comme valide si elle remplit 2 conditions :

1. elle est signée par suffisamment de clefs valides *i.e.*
   - on l'a signée personnellement ;
   - elle a été signée par une clef à laquelle on accorde toute sa confiance ;
   - elle a été signée par 3 clefs auxquelles on accorde une confiance marginale.
2. le chemin des clefs conduisant de cette clef à sa propre clef mesure moins de 5 étapes.

Utiliser la confiance pour valider une clef

Mettre à jour le réseau

Il s'agit des paramètres par défaut, ils sont modifiables avec un fichier de configuration approprié.
Il est possible d'interroger la base de données pour faire apparaître les personnes non signées mais valides dans le trousseau.

Pour mettre à jour le réseau de confiance, on utilise :

```
pauline@samothrace $ gpg --update-trustdb
```

Principe des serveurs de clefs

Idéalement, les clefs sont distribués personnellement. En pratique, les serveurs de clefs publiques sont utilisés pour collecter et distribuer les clefs publiques.
En cas d'envoi de clef :

- ajout de la clef à la base de donnée ;
- fusion de la clef avec la clef existante si elle existe.

En cas de requête de clef, le serveur renvoie la clef publique.

Intérêt des serveurs de clefs

En cas de signature de sa clef, il faut récupérer sa clef signée et la redistribuer à tous ses contacts, pour qu'ils aient une version à jour...
Quand quelqu'un signe une clef, il la renvoie au serveur de clef qui rajoute la signature à sa copie de la clef publique. Les contacts peuvent récupérer de façon autonome la clef mise à jour : le propriétaire est affranchi de la distibution.
Le propriétaire récupère les signatures sur sa clef avec `--recv-keys`.
Les grands serveurs se mettent à jour les uns avec les autres, il suffit d'en sélectionner un.

Intérêt des serveurs de clefs

`--refresh-keys`

Il faut rafraîchir son trousseau régulièrement à cause des révocations et des expirations. Pour cela :

```
pauline@samothrace $ gpg --refresh-keys
```

`--list-keys`

Pour communiquer avec les autres, il faut pouvoir échanger les clefs publiques. Pour afficher le trousseau de clefs publiques on utilise `--list-keys` :

Importer une clef publique

À partir d'un serveur de clefs

Il faut procéder en 3 étapes :

1. trouver la clef publique souhaitée `gpg --search-keys identité` ;
2. choisir la clef de la personne que l'on cherche (et pense avoir trouver) ;
3. absorber la clef `gpg --recv-keys identifiant_clef`.

Importer une clef publique

Lorsque l'on dispose de la clef

Si on a reçu la clef par e-mail, de la main à la main par clef USB ou autre :

```
pauline@samothrace $ gpg --import nom_du_fichier
```

Exporter une clef publique

Sur un serveur de clef

Pour cela on utilise la ligne de commande suivante :

```
pauline@samothrace $ gpg --send-key ID_clef
```

Exporter une clef

Envoyer à un correspondant

Comme le format binaire peut être problématique, on peut utiliser l'option `--armor` qui permet la génération de sorties au format *ASCII-armored* plus pratique d'utilisation.

```
pauline@samothrace $ gpg --armor --export-key ID_clef
```

Trouver l'ID de la clef

Pour commencer, il faut trouver l'ID de la clef avec laquelle on veut travailler :

```
pauline@samothrace $ gpg --list-secret-keys
```

Principe

Le principe est le même pour la clef principale et les sous-clefs, il faut juste remplacer `keys` par `subkeys` et préciser l'ID.

```
pauline@samothrace $ gpg --export-secret-keys (--armor) --output secret.asc
```

Principe

sous-clef : clef de signature ou de chiffrement liée à une clef maîtresse, pouvant être révoquée et stockée indépendamment d'elle/

On travaille exclusivement avec les sous-clefs que l'on publie sur les serveurs de clefs. Cela permet de garder la clef maîtresse à l'abri.

Exceptions

Cependant, on utilise sa clef principale pour :

- signer la clef de quelqu'un,
- créer une nouvelle sous-clef (`addkey`),
- révoquer une sous-clef.

Recommandations Debian

1. récupérer la clef maîtresse `export GNUPGHOME=/media/blabla` ;
2. créer la sous-clef ;
3. copier `$HOME/.gnupg` dans le disque externe ;
4. exporter la sous-clef `gpg --export-secret-subkeys subID > subkeys` ;
5. exporter les clefs publiques `gpg --export ID > pubkeys` ;
6. supprimer la clef principale `gpg --delete-secret-key ID` ;
7. réimporter `gpg --import pubkeys subkeys` ;
8. vérifier que `--list-keys` renvoie `sec#`.

Principe

Si Bob envoie un message à Alice, il le chiffre avec la clef publique d'Alice qui le déchiffrera avec sa clef privée. Et vice-versa.

Chiffrer

Pour chiffrer un message à destination de plusieurs personnes, il faut la clef publique de chacun des destinataires, que l'on précise avec `--recipient`. Si l'on ne s'inclue pas dans les destinataires, on ne pourra pas lire son propre message.

```
$ gpg --output document.gpg --encrypt --recipient ID_clef_dest_1 document
```

Déchiffrer

Pour décrypter un message, il faut la clef privée pour laquelle le message a été chiffré.

```
$ gpg --output document --decrypt document.gpg
```

Principe

Une signature sert à :

- certifier et dater un document ;
- permettre de vérifier que l'on est bien l'expéditeur ;
- permettre de vérifier que le document n'a pas été modifié depuis son envoi.

La signature se fait avec la clef privée de l'expéditeur.

Face à un document signé, on peut :

- vérifier la signature avec `--verify` ;
- vérifier la signature et extraire le document avec `--decrypt doc.sig`.

Documents signés en clair

Par exemple, il n'est pas agréable de recevoir un e-mail compressé, il est préférable de recevoir un message en clair suivi de la signature :

```
pauline@samothrace $ gpg --clearsign doc
```

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

plop

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.12 (GNU/Linux)

iQIcBAEBAgAGBQJRHgoCAAoJEAVQT/DPh1/haL0P/i/SNdHpyBXBn5H2B1kcyNT4
Cp1i8iDdMf6RCVbJIkcVle8Xo6+ZEIZFpIm9jtk1ldISZL11YOWRBXjuNIUUnQLn
XPz9w2R/BLHTRcEIXGgD6Y8iIO3+iK82LC5GfENbHj1wVSiVsEx33Wlejnxs JsQp
aC1aQJm0ICwRX3e+ZndngGkHv2ccc1jET7fk2l4Sl431nCY1L6pFIuuzU6yc17kF
J//M7wM2HhNgqUjbBO8ne157PEA/sx1JrhFp13kftnUAhp6u3PwsU/uhHxiGK6/e
2xonWKjhcpCvcLcia5AsjuA+NEqtEiRNVH7G11wHAHMqbFXU16F0MMeMQllO/h1l
/8dbYRU/oWf6oO+dcKKcF6zK/nqDsOJusiWAaKFuhfgXuMl9TocweOdIrgrvxA6w
VyeL9K0+tmm+uEl7qKIB6/eWXD2QgGbi5D3aODZdScMk7svbkLe7yopClDEn3fFI
RvrfsfmgWZeZVAdSVNr78Ln77sNzoo2xTA30T2E0Goaiham3NUBOxHWV3UnMVejA
tSjizdfrdcHIJwp6s3IfAkNRUgPY104UFxJkfzhLBUIOd8aWojBegnGBQfT0h/L3
EW7MAwunQ50xcuJMaehzY2UaxzEM99NESXbFKzIieLOzk3BAGEJEmtdC+ccaEIzS
F9kIoSjygdynkjhY1oJ4
=N0gF
-----END PGP SIGNATURE-----
```

Signatures détachées

Les utilisateurs doivent récupérer le document à partir de la version signée ou le modifier pour retrouver l'original, ce qui peut être pénible. On peut générer une signature détachée du document, qui peut être vérifiée avec `--verify` :

```
$ gpg --output document.sig --detach-sig document
```

C'est facile, même moi j'ai réussi !

Toujours préférer PGP-MIME

- *Claws mail*, client messagerie supportant GnuPG,
- *Enigmail*, plug-in Mozilla,
- *Evolution*,
- *exmh*,
- *ez-pine-gpg*, ensemble de scripts pour utiliser gpg avec Pine,
- *GNU Anubis*,
- *GNUmail.app*,
- *KMail*,
- *MagicPGP*, ensemble de scripts pour Pine,
- *Mew*,
- *Mutt*, dispose support complet GnuPG/PGP,
- *OpenPGP Webmail*,
- *Scribe*,
- *Sylpheed*,
- *XFmail*,
- pleins d'autres, plus ou moins facile d'accès.

Choisir la taille des clefs

plus c'est gros mieux c'est ?

Protéger sa clef privée

Essentiel :

- si quelqu'un l'obtient, tout pourra être déchiffré et on peut signer en votre nom ;
- si on la perd, on peut plus rien déchiffrer.

Protéger sa clef privée

Il faut idéalement :

- conserver le certificat de révocation et une copie de sauvegarde de la clef publique sur un support protégé en écriture dans un lieu sûr ;
- conserver la clef privée sur un disque amovible protégé en écriture ;
- utiliser la clef privée sur une machine mono-utilisateur déconnectée du réseau ;
- avoir un bon mot de passe.

Conclusion : utiliser des sous-clefs

Définition des dates d'expiration

Selon la clef, les délais d'expiration varient :

1. délai « long » pour la clef principale :
2. délai court pour les sous-clefs :
   - changer régulièrement est plus sécurisé (protection des documents à venir) ;
   - en cas de perte de contrôle de la clef et de perte du certificat de révocation.

Utilisation des clefs secondaires

Les consignes sont :

- changer régulièrement afin de protéger les documents chiffrés ultérieurement ;
- publier la nouvelle clef avant l'expiration de la précédente ;
- faire valider sa clef principale par ses correspondants ;
- aucun intérêt à avoir plusieurs clefs secondaires de chiffrement actives à un temps donné ;
- aucun problème à avoir plusieurs clefs secondaires expirées dans une paire de clef donnée.

Gérer sa toile de confiance

Il faut garder en tête que l'appartenance au réseau de confiance n'est pas une garantie de bonne foi, c'est un indice de validité de l'identité de la personne.
Ce qui compte, ce n'est pas le nombre de signatures, mais la qualité des signatures.
Il existe deux façons de gérer sa confiance :

- modèle PGP ou la validité d'acquière par 1 confiance totale ou 3 confiances marginales ;
- modèle personnalisé en fonction de l'usage des indices de confiances en attribuant des valeurs à `completes-needed` et `marginals-needed` dans `gnupg.conf`.

Faire de la propagande !

- commencer avec les personnes avec qui vous avez appris ;
- introduiser subtilement une signature et répondre aux interrogations soulevées par la mystérieuse pièce-jointe ;
- aller à des *key-signing parties* !

Conclusion

C'était qu'une introduction !

Bibliographie

- http ://doc.ubuntu-fr.org/gnupg
- http ://fr.wikibooks.org/wiki/GPG
- http ://matrix.samizdat.net/crypto/gpg_intro/
- http ://www.gnupg.org/gph/fr/manual.html
- http ://docs.abuledu.org/abuledu/mainteneur/creer_une_cle_gpg#reseaux_de_confiance
- http ://www.legifrance.gouv.fr/
- http ://fr.wikipedia.org/wiki/Cryptographie
- http ://security.stackexchange.com/questions/5096/rsa-vs-dsa-for-ssh-authentication-keys
- http ://www.linuxquestions.org/questions/linux-security-4/gpg-rsa-or-dsa-with-el-gamal-for-new-keys-565242/
- http ://docu.fsugar.be/openpgp/openpgp.html
- http ://wiki.debian.org/subkeys