Authentification GPG

Nous allons généré un trousseau de clé GPG afin de certifier, chiffrer, signer et authentifier des fichiers et autres.

Configuration de GPG

Créons une VM, démarrons la, on lui change son IP :

```
nano /etc/network/interfaces
```

On change l'IP du fichier par 10.31.208.10 on save et on quit.

```
systemctl restart networking
```

Maintenant on change le nom d'hôte

```
hostnamectl set-hostname gpg
reboot
```

Maintenant si ce n'est pas fait installons gpg

```
apt update
apt install gpg
```

Générer une clé GPG

Vérifions la version de gpg

```
gpg --version
```

```
root@testgpg:~# gpg --version
gpg (GnuPG) 2.2.40
libgcrypt 1.10.1
Copyright (C) 2022 g10 Code GmbH
License GNU GPL-3.0-or-later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /root/.gnupg
Algorithmes pris en charge :
Clef publique : RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Chiffrement : IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256,
              TWOFISH, CAMELLIA128, CAMELLIA192, CAMELLIA256
Hachage : SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression : Non compressé, ZIP, ZLIB, BZIP2
```

Nous allons généré la clé principal de gpg

```
gpg --full-generate-key --expert
```

```
root@testgpg:~# gpg --full-generate-key --expert
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: répertoire « /root/.gnupg » créé
gpg: le trousseau local « /root/.gnupg/pubring.kbx » a été créé
Sélectionnez le type de clef désiré :
   (1) RSA et RSA (par défaut)
   (2) DSA et Elgamal
   (3) DSA (signature seule)
   (4) RSA (signature seule)
   (7) DSA (indiquez vous-même les capacités)
   (8) RSA (indiquez vous-même les capacités)
   (9) ECC et ECC
  (10) ECC (signature seule)
  (11) ECC (indiquez vous-même les capacités)
  (13) Clef existante
  (14) Existing key from card
Quel est votre choix ?
```

Ici on va prendre le choix **8** (**RSA**)

```
Quel est votre choix ? 8

Actions possibles pour une clef RSA : Signer Certifier Chiffrer Authentifier
Actions actuellement permises : Signer Certifier Chiffrer

   (S) Inverser la capacité de signature
   (C) Inverser la capacité de chiffrement
   (A) Inverser la capacité d'authentification
   (Q) Terminé

Quel est votre choix ?
```

Pour la clé principal nous allons uniquement garder l'option **certifier**. Donc en rentre 'S' puis 'C' puis 'A'.

```
Actions possibles pour une clef RSA : Signer Certifier Chiffrer Authentifier
Actions actuellement permises : Certifier

   (S) Inverser la capacité de signature
   (C) Inverser la capacité de chiffrement
   (A) Inverser la capacité d'authentification
   (Q) Terminé

Quel est votre choix ?
```

On peut maintenant rentrer 'Q' pour terminer.

```
les clefs RSA peuvent faire une taille comprise entre 1024 et 4096 bits.
Quelle taille de clef désirez-vous ? (3072) 4096
```

Ici on nous demande le nombre de bit de notre clef RSA, on choisit 4096 (pour un max de sécu).

```
La taille demandée est 4096 bits
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) 1y
```

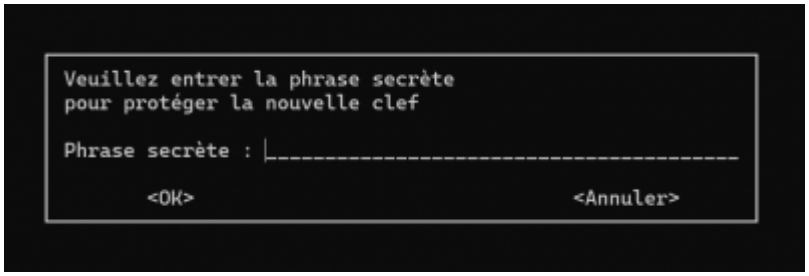
Ici on nous demande combien de temps on veut que notre clé sois valide (note : on peut la renouveler si on veut garder la même), on choisis 1 an en rentrant **1y**.

Maintenant on nous demande notre identité :

```
GnuPG doit construire une identité pour identifier la clef.

Nom réel : Paul
Le nom doit contenir au moins cinq caractères
Nom réel : Paulia
Adresse électronique : skibidi.fortnite@apagnan.com
Commentaire :
Vous avez sélectionné cette identité :
    « Paulia <skibidi.fortnite@apagnan.com> »
```

Pour finir, on nous demande de rentrer une passphrase :

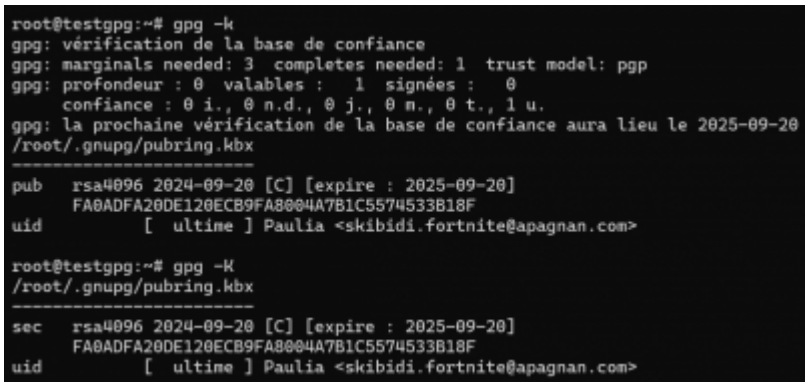


Voila, notre clé principal est finalisé :

Générer des sous-clés GPG

Vérifions au préalable si nous avons des clés :

```
gpg -k # affiche les clés publiques possédées
gpg -K # affiche les clés privées possédées
```



Nous pouvons créer une sous-clé :

```
gpg --expert --edit-key Paulia # éditer la clé master
gpg> addkey # ajouter une sous-clé
```

```
root@testgpg:~#  gpg --expert --edit-key Paulia
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

La clef secrète est disponible.

sec  rsa4096/A7B1C5574533B18F
     créé : 2024-09-20  expire : 2025-09-20  utilisation : C
     confiance : ultime        validité : ultime
[  ultime ] (1). Paulia <skibidi.fortnite@apagnan.com>

gpg> addkey
Sélectionnez le type de clef désiré :
   (3) DSA (signature seule)
   (4) RSA (signature seule)
   (5) Elgamal (chiffrement seul)
   (6) RSA (chiffrement seul)
   (7) DSA (indiquez vous-même les capacités)
   (8) RSA (indiquez vous-même les capacités)
  (10) ECC (signature seule)
  (11) ECC (indiquez vous-même les capacités)
  (12) ECC (chiffrement seul)
  (13) Clef existante
  (14) Existing key from card
Quel est votre choix ?
```

On va remettre les mêmes valeurs pour la taille de clé et sa durée de validité, mais pour chacune des 3 sous clés, on va associé uniquement 'Signer' 'Chiffrer' 'Authentifier'.

```
gpg> save
```

```
gpg -k
```

```
root@testgpg:~# gpg -k
/root/.gnupg/pubring.kbx
------------------------
pub   rsa4096 2024-09-20 [C] [expire : 2025-09-20]
      FA0ADFA20DE120ECB9FA8004A7B1C5574533B18F
uid           [  ultime ] Paulia <skibidi.fortnite@apagnan.com>
sub   rsa4096 2024-09-20 [E] [expire : 2025-09-20]
sub   rsa4096 2024-09-20 [S] [expire : 2025-09-20]
sub   rsa4096 2024-09-20 [A] [expire : 2025-09-20]
```

```
gpg -K
```

```
root@testgpg:~# gpg -K
/root/.gnupg/pubring.kbx
------------------------
sec   rsa4096 2024-09-20 [C] [expire : 2025-09-20]
      FA0ADFA20DE120ECB9FA8004A7B1C5574533B18F
uid           [  ultime ] Paulia <skibidi.fortnite@apagnan.com>
ssb   rsa4096 2024-09-20 [E] [expire : 2025-09-20]
ssb   rsa4096 2024-09-20 [S] [expire : 2025-09-20]
ssb   rsa4096 2024-09-20 [A] [expire : 2025-09-20]
```

Test GPG

Voila toute les commandes :

```
# Affiche la liste des clés publiques possédées
gpg -k

# Affiche la liste des clés privées possédées
gpg -K
```

```
# Création d'une paire de clés
gpg --full-gen-key --expert

# Editer une clé (pour modification par exemple)
gpg --expert --edit-key 310144D6
# La commande précédent ouvre un prompt gpg>
    # On peut taper addkey pour ajouter une sous-clé à la clé éditée
    gpg> addkey
    # save permet d'enregistrer les modifications effectuées sur la clé.
    gpg> save

# Création d'un certificat de révocation
gpg --output ./revoc.asc --gen-revoke 310144D6

# Export de la clé secrète en format ASCII
gpg -a --export-secret-key 310144D6 > secret.asc

# Export de la clé secrète en format binaire
gpg --export-secret-key 310144D6 > secret.gpg

# Export de la clé publique en format ASCII
gpg -a --export 310144D6 > public.asc

# Export de la clé publique en format binaire
gpg --export 310144D6 > public.gpg

# Export de sous-clés en format ASCII
gpg -a --export-secret-subkeys 033B83FF > sub_keys.asc

# Export de sous-clés en format binaire
gpg --export-secret-subkeys 033B83FF > sub_keys.gpg

# Supprime une clé privée
gpg --delete-secret-keys 310144D

# Supprime une clé publique
gpg --delete-keys 310144D

# Supprime les deux en une seule commande
gpg --delete-secret-and-public-keys 310144D6

# Après suppression d'une clé il convient de redémarrer l'agent gpg
gpgconf --kill gpg-agent

# Chiffre le fichier bonjour.txt pour tompouce (il faut avoir sa clé
publique)
gpg -r tompouce@beaup.com --encrypt bonjour.txt

# Déchiffre le fichier bonjour.txt.gpg (il faut avoir la clé privée)
gpg --output bonjour.txt --decrypt bonjour.txt.gpg
```

```
# Signe et compresse le fichier bonjour.txt
gpg --output bonjour.sig --sign bonjour.txt

# Décompresse et vérifie la signature d'un fichier signé
gpg --output bonjour.txt --decrypt bonjour.sig

# Signe et garde en clair le fichier bonjour.txt
gpg --clearsign bonjour.txt

# Vérifie la signature du fichier bonjour.txt.asc
gpg --verify bonjour.txt.asc

# Signe le fichier et place la signature dans un fichier à part.
gpg --detach-sign bonjour.txt

# Chiffre pour tompouce et signe en même temps le fichier bonjour.txt
gpg -r tompouce@beaup.com -a --sign --encrypt bonjour.txt
```