Table des matières

1.1) Calcul du hash d'un fichier à l'aide de md5sum

1) Je commence par ouvrir un fichier texte (avec nano par exemple) et je l'édite.

Dans le fichier texte j'écris « salut ».

Maintenant je test la commande suivante :

```
test@213-10 : ~
$ md5sum q1
6aba532a54c9eb9aa30496fa7f22734d  q1
```

Le hash obtenue est celui surligné en rouge ci-dessus.

2)Il y a 32 caractères en hexadécimale soit 32*4=128 bits donc 2^{128} condensés

3) La dernière propriété mathématique énoncée dans Wikipedia est possible en pratique mais est très peu probable.

1.2) Calcul du hash et modification du fichier

1) Je reprend le fichier précédent que je vais renommer :

```
test@213-10 : ~
$ mv q1 q2
```

Je refais ensuite la commande md5sum :

```
test@213-10 : ~
$ md5sum q2
6aba532a54c9eb9aa30496fa7f22734d  q2
```

Je remarque que même en aillant modifié le nom du fichier le hash reste le même.

2) Je modifie maintenant le contenue du fichier et je calcul le hash :

```
test@213-10 : ~
$ md5sum q2
94baaad4d1347ec6e15ae35c88ee8bc8  q2
```

Je remarque cette fois ci que le hash a été modifié comparé au précédent.

3) Il est possible de calculer le hash d'un fichier binaire autre qu'un texte. Je prend l'exemple avec une image.jpg.

```
test@213-10 : ~/Images
$ md5sum bleach.jpg
52edb7a843abacb4ff0f642c8eda1440  bleach.jpg
```

2.1) Créer ses clés de chiffrement

Je commence par générer une paire de clés publique/privée avec la commande suivante :

```
test@213-10 : ~
$ gpg --gen-key
gpg (GnuPG) 2.2.12; Copyright (C) 2018 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.
```

Maintenant je vérifie le porte-clés numérique pour voir s'il contient bien mes clés :

```
test@213-10 : ~
$ gpg --list-keys
gpg: vérification de la base de confiance
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: profondeur : 0  valables :   1  signées :   0
     confiance : 0 i., 0 n.d., 0 j., 0 m., 0 t., 1 u.
```

Enfin je liste ma clé privée :

```
test@213-10 : ~
$ gpg --list-secret-keys
/home/test/.gnupg/pubring.gpg
```

2.2) Diffuser sa clé publique sur internet

Je commence par exporter ma clé sur un serveur de clé :

```
test@213-10 : ~
$ gpg --keyserver 10.213.0.174 --send-keys
1B659201D22186C20877D0AA7D40CADB4F5ACEB4
gpg: envoi de la clef 7D40CADB4F5ACEB4 à hkp://10.213.0.174
```

2.3) Chiffrons un fichier

Je commence par télécharger la clé publique du professeur.

j'importe la clé dans mon trousseau :

```
test@213-10 : ~
$ gpg --import cle_comby
gpg: clef 9E84B61C4CFB6FB4 : clef publique « Fred Comby
<frederic.comby@umontpellier.fr> » importée
gpg:       Quantité totale traitée : 1
gpg:                   importées : 1
```

Je vérifie ensuite que la clé est bien présente dans mon trousseau :

```
test@213-10 : ~
$ gpg -k
/home/test/.gnupg/pubring.gpg
-----------------------------
pub   rsa3072 2019-09-18 [SC] [expire : 2021-09-17]
      1B659201D22186C20877D0AA7D40CADB4F5ACEB4
uid         [  ultime ] samuel <samy271100@gmail.com>
sub   rsa3072 2019-09-18 [E] [expire : 2021-09-17]

pub   rsa3072 2019-09-18 [SC] [expire : 2021-09-17]
      385314CDED6C42ADBD8523A09E84B61C4CFB6FB4
uid         [ inconnue] Fred Comby <frederic.comby@umontpellier.fr>
uid         [ inconnue] La Team <benoit.plessis@umontpellier.fr>
sub   rsa3072 2019-09-18 [E] [expire : 2021-09-17]
```

Maintenant je crée un fichier texte que je vais ensuite éditer (on peut mettre ce que l'on veut).

Une fois édité, je crypte le fichier

```
test@213-10 : ~
$ gpg --armor --recipient frederic.comby@umontpellier.fr --encrypt sam.txt
gpg: DB3BCAECA4C7AD84 : aucune assurance que la clef appartienne
vraiment à l'utilisateur nommé.

sub  rsa3072/DB3BCAECA4C7AD84 2019-09-18 Fred Comby
<frederic.comby@umontpellier.fr>
Empreinte clef princip. : 3853 14CD ED6C 42AD BD85  23A0 9E84 B61C 4CFB
6FB4
 Empreinte de sous-clef : E030 48AB 9772 BC0A 2D5E  14F5 DB3B CAEC A4C7
AD84

La clef n'appartient PAS forcément à la personne nommée
dans l'identité. Si vous savez *vraiment* ce que vous
faites, vous pouvez répondre oui à la prochaine question.


Faut-il quand même utiliser cette clef ? (o/N) o
```

Maintenant le fichier est crypté et je l'envoie au professeur sur l'ent montpellier.

Je reçois maintenant un fichier crypté de la part d'Adrian DELMAS que je vais devoir décrypter avec la commande suivante :

```
 test@213-10 : ~
$ gpg --decrypt coucou-sam.txt.asc > coucou-sam.txt
gpg: chiffré avec une clef RSA de 3072 bits, identifiant 79C011A349FA3921,
créée le 2019-09-18
      « samuel <samy271100@gmail.com> »
```

Une fois décrypté je n'ai plus qu'à lire le fichier

3) Signature d’un fichier

1) Pour cette partie Thomas SEVERAN va signer son fichier et va me l’envoyer sur internet via l’ent de montpellier. Pour se faire il va faire les commandes suivantes :

```
gpg --default-key thomas.severan@etu.umontpellier.fr --clearsign
bonjour_monde.txt
```

Cette commande permet de signer son fichier bonjour_monde.txt

Maintenant il vérifie qu’il a bien son fichier signé :

```
test@210.9 ~ $ ls
bonjour_monde.txt bonjour_monde.txt.asc
```

On peut aussi vérifier que le fichier est bien signé en le lisant :

```
test@210.9 ~ $ cat bonjour_monde.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256
hello world
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2
iQEcBAEBCAAGBQJW9UtTAAoJEBpr/9mFKtWQRMkH/
0H7NCarTzPl7DgPt2PGemJU
WxARNnzcHGF+QREl1yUiJ+nPoA6knS6ilquJFAhKZmyi00MXbp5ptGg2MyZgoe8i
mbuBHPjRQQq47+8qSMKOl1hY8sAAdBpuxrhqgwAG59i/B3DbGjWTNTWEMJG/
ACFt
hFGfn6vRW42A4Phc7U8RhbrrnVQ9LO1DoJFD7QFjxDwwReLnRKzLSPj10sjqyNSe
f6/gI2NH6RsIkGu9Pi5q49vhm4hsnusMEueOdN1ASKo3pI9Vqf1B1KY7McFM3MLp
8FDgOtvLvjw3PlXe55GDAeNL1J6BJy9y3WB97KlcfUuUwJq8Y9FEXd3t85WPAAI=
=bk+0
-----END PGP SIGNATURE-----
```

Maintenant je vais m’occuper de vérifier sa signature. Pour cela je télécharge sa clé publique sur l’ent et je vérifie la signature avec la commande suivante :

```
gpg --verify bonjour_monde.txt.asc
```

4) Signature d'une clé publique (cercle de confiance)

Pour cette partie du TP je serais encore avec Thomas Severan. Je commence par récupérer sa clé publique qu'il a déposé sur internet :

```
test@213.10~ $ gpg --keyserver pgp.mit.edu --recv-keys n°clé
```

Une fois sa clé récupérée je la signe :

```
test@210.9 ~ $ gpg --sign-key n°clé
```

Maintenant j'envoie la clé signée sur le serveur pgp.mit.edu :

```
test@210.9 ~ $ gpg --keyserver pgp.mit.edu --send-key n°clé
```

Pour terminer thomas va sur le serveur pour vérifier que la clé est bien signée.

5) Récupération de sa clé privée

Ma clé publique se trouve sur le serveur et peut être récupéré à tout moment de n'importe où. Pour pouvoir déplacer ma clé privée et la récupérer dans un autre endroit je peux l'exporter dans un fichier nommé secret.key :

```
gpg --export-secret-key --armor > secret.key
```

Une fois exporter je prend le fichier et je peux l'emmener dans un endroit où j'aimerais la récupérer (ici ce sera à la maison). Une fois chez moi pour pouvoir récupérer ma clé je fais la commande suivante :

```
gpg –import secret.key
```

Voilà maintenant j'ai récupéré ma clé privée que je peux utiliser pour envoyer de messages cryptés.