



**AVIGNON**  
UNIVERSITÉ

# Rendu TP3 : Certificats

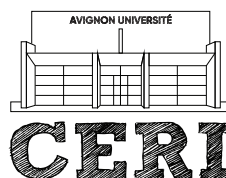
Groupe TD4, B  
**RHADI Meryam**

**28 avril 2021**

**Ingénierie Logicielle**  
**UE** Sécurité informatique

**Responsable**  
Majed haddad

**UFR**  
**SCIENCES**  
**TECHNOLOGIES**  
**SANTÉ**



**CENTRE**  
**D'ENSEIGNEMENT**  
**ET DE RECHERCHE**  
**EN INFORMATIQUE**  
[ceri.univ-avignon.fr](http://ceri.univ-avignon.fr)

## Sommaire

<b>Titre</b>	<b>1</b>
<b>Sommaire</b>	<b>2</b>
<b>1 Exercice 1 : Empreintes MD5/SHA-1/ SHA-2</b>	<b>3</b>
1.1 Question 1, 2, 3 :	3
1.2 Question 4 :	3
1.3 Question 5 :	3
1.4 Question 6 :	4
<b>2 Exercice 2 : GPG GnuPG</b>	<b>4</b>
2.1 Question 1 :	4
2.2 Question 2 :	5
2.3 Question 3 :	5
2.4 Question 4 :	7
2.5 Question 5 :	7
2.6 Question 6 :	7
2.7 Question 7 :	8
2.8 Question 8 : Chiffrement	9
2.9 Question 9 : Signature d'un message	10
2.10 Question 10 : Signature et chiffrement	11
<b>3 Exercice 3 : Certificats</b>	<b>11</b>
3.1 Question 1 :	11
3.2 Question 2 :	11
3.3 Question 3 :	12
3.4 Question 4 :	13
3.5 Question 5 :	14
3.6 Question 6 :	14
3.7 Question 7 :	15
3.8 Question 8 :	16
3.9 Question 9 :	16

## 1 Exercice 1 : Empreintes MD5/SHA-1/ SHA-2

### 1.1 Question 1, 2, 3 :

Dans cette partie, il a été demandé de travailler avec des outils qui permet de générer des empreintes pour un fichier quelconque, c'est outils sont **md5sum**, **sha1sum** et **sha256sum**, ils sont déjà installé sous Linux, et on a travaillé avec un fichier téléchargeable **Licence.txt**, il est demandé de générer l'empreinte de ce fichier avec les différentes outils.

```
stud@ubuntu:~/Desktop$ md5sum LICENSE.txt
d33aaa5246e1ce0a94fa15ba0c407ae2  LICENSE.txt
stud@ubuntu:~/Desktop$ █
```

Figure 1. Résultat de l'empreinte avec md5sum

```
stud@ubuntu:~/Desktop$ sha1sum LICENSE.txt
11d197acb61361657d638154a9416dc3249ec9fb  LICENSE.txt
stud@ubuntu:~/Desktop$
```

Figure 2. Résultat de l'empreinte avec sha1sum

```
stud@ubuntu:~/Desktop$ sha256sum LICENSE.txt
1d4ff95ce9c6e21fe4a4ff3b41e7a0df88638dd449d909a7b46974d3dfab7311  LICENSE.txt
stud@ubuntu:~/Desktop$
```

Figure 3. Résultat de l'empreinte avec sha256sum

### 1.2 Question 4 :

La taille des empreintes obtenue par les outils **md5sum**, **sha1sum** et **sha256sum**, se défère d'un outil à l'autre, ils est comme suite :

- **md5sum** : Les empreints génères par MD5 contient 32 caractères, et en bits sa donne 128 bits.
- **sha1sum** : Les empreints génères par SHA1 contient 40 caractères, et en bits sa donne 160 bits.
- **sha256sum** : Les empreints génères par MD5 contient 64 caractères, et en bits sa donne 256 bits.

### 1.3 Question 5 :

Résultat après la modification d'un caractère dans le fichier texte **LICENCE.txt** :

```
root@ubuntu:~/Desktop# nano LICENSE.txt
root@ubuntu:~/Desktop# md5sum LICENSE.txt
9a1c55c053acf942d3897531422b5523  LICENSE.txt
root@ubuntu:~/Desktop# sha1sum LICENSE.txt
8fc7b25860c55569997b047541071528749ac032  LICENSE.txt
root@ubuntu:~/Desktop# sha256sum LICENSE.txt
78eb9442250a0c80f6268333418bc576f778514a43f26b417fd5bb188c52b6fc  LICENSE.txt
root@ubuntu:~/Desktop# █
```

**Figure 4.** Résultat des empreintes avec les différents outils après la modification de fichier source

On remarque que les empreintes sont différentes, même si on a changé un seul caractère dans le fichier texte, donc l'empreinte assure l'intégrité des données en peut savoir si le fichier de départ a été changé ou non, en comparant les empreintes.

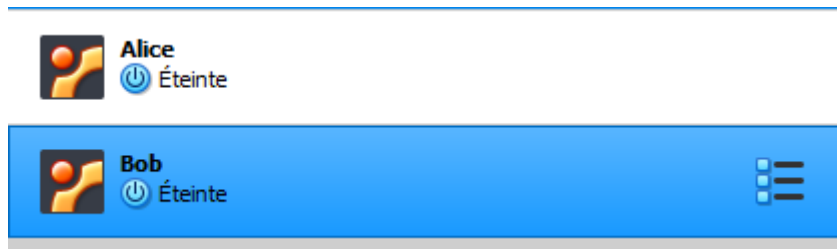
#### 1.4 Question 6 :

La méthode d'empreinte numérique parmi (MD5, SHA-1, SHA2) qui est recommandée aujourd'hui est **SHA2**, parce qu'il contient plus de bits donc il renforce la sécurité des informations, et pour **MD5 et SHA1** on a la possibilité d'être en collision, c'est à dire avec deux entrées différentes, on peut obtenir une empreinte identique.

## 2 Exercice 2 : GPG GnuPG

### 2.1 Question 1 :

Lors de cette étape, j'ai créé deux machines virtuelles :



**Figure 5.** Création de Alice et Bob

Pour la communication entre ces deux machines via un réseau interne, j'ai utilisé pour les deux un réseau de type **Réseau interne**, et affecter des adresses IP 192.168.100.1 pour Alice et 192.168.100.2 pour Bob avec adapter2 et dans adapter1 j'ai travaillé avec réseau NAT pour avoir la capacité pour accéder à internet.

```

root@ubuntu:~# ifconfig eth0 192.168.100.1
root@ubuntu:~# hostname -I
192.168.100.1
root@ubuntu:~#

root@ubuntu:~# ifconfig eth0 192.168.100.2
root@ubuntu:~# hostname -I
192.168.100.2
root@ubuntu:~# █

```

Figure 6. Changement des paramètres réseaux pour Bob et Alice

## 2.2 Question 2 :

Pour utiliser la commande **gpg2 --help**, qui permet d'obtenir des informations sur GNU Privacy Guard, il faut premièrement installer gnupg2 avec la commande **apt install gnupg2**

```

root@ubuntu:~# apt install gnupg2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  dirmngr gnupg-agent libassuan0 libksba8 libnpth0 pinentry-curses
Suggested packages:
  gnupg-doc parcimonie xloadimage pinentry-doc
The following NEW packages will be installed:
  dirmngr gnupg-agent gnupg2 libassuan0 libksba8 libnpth0 pinentry-curses
0 upgraded, 7 newly installed, 0 to remove and 382 not upgraded.
Need to get 1,488 kB of archives.
After this operation, 4,448 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main i386 libassuan0 i386 2.4.2-2 [36.6 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial/main i386 pinentry-curses i386 0.9.7-3 [32.1 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial/main i386 libnpth0 i386 1.2-3 [8,322 B]
Get:4 http://us.archive.ubuntu.com/ubuntu xenial-updates/main i386 gnupg-agent i386 2.1.11-6ubuntu2.1 [260 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu xenial-updates/main i386 libksba8 i386 1.3.3-1ubuntu0.16.04.1 [98.7 k
B]

```

```

root@ubuntu:~# gpg2 --help
gpg (GnuPG) 2.1.11
libgcrypt 1.6.5
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://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: ~/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2

Syntax: gpg [options] [files]
Sign, check, encrypt or decrypt
Default operation depends on the input data

Commands:

-s, --sign                make a signature
--clearsign               make a clear text signature

```

Figure 7. Résultat de la commande gpg2 --help

## 2.3 Question 3 :

Pour la génération des clés, on vas utiliser la commande **gpg --gen-key**, et après on vas répondre sur les différents questions posées.

```

root@ubuntu:~# gpg --gen-key
gpg (GnuPG) 1.4.20; Copyright (C) 2015 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.

Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
  0 = key does not expire
  <n> = key expires in n days
  <n>w = key expires in n weeks
  <n>m = key expires in n months
  <n>y = key expires in n years
Key is valid for? (0) 22
Key expires at Fri 14 May 2021 07:53:08 PM CEST
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
  "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: meryam rhadi
Email address: meryamrhadi@gmail.com
Comment: ceri
You selected this USER-ID:
  "meryam rhadi (ceri) <meryamrhadi@gmail.com>"

```

```

...+****
gpg: /home/stud/.gnupg/trustdb.gpg: trustdb created
gpg: key E8EECFE7 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2021-05-14
pub  2048R/E8EECFE7 2021-04-22 [expires: 2021-05-14]
      Key fingerprint = 2336 11E1 3E07 D12B FCB0  DA9A EEAE 547F E8EE CFE7
uid          meryam rhadi (ceri) <meryamrhadi@gmail.com>
sub  2048R/0C1EFF1F 2021-04-22 [expires: 2021-05-14]

```

Pour l'affichage de trousseau de clés, on utilise la commande : **gpg --list-keys**

```

root@ubuntu:~# gpg --list-keys
/home/stud/.gnupg/pubring.gpg
-----
pub  2048R/E8EECFE7 2021-04-22 [expires: 2021-05-14]
uid          meryam rhadi (ceri) <meryamrhadi@gmail.com>
sub  2048R/0C1EFF1F 2021-04-22 [expires: 2021-05-14]

```

**Figure 8.** Affichage de trousseau de clé publique

```

root@ubuntu:~# gpg --list-secret-keys
/home/stud/.gnupg/secring.gpg
-----
sec  2048R/E8EECFE7 2021-04-22 [expires: 2021-05-14]
uid          meryam rhadi (ceri) <meryamrhadi@gmail.com>
ssb  2048R/0C1EFF1F 2021-04-22

```

**Figure 9.** Affichage de trousseau de clé privé

On remarque que les clés (publique et privée), qu'en a crée auparavant a été ajouté au trousseau de la machine.

## 2.4 Question 4 :

Pour l'exportation des clés dans des fichiers différents, on va utiliser les commandes :

```
root@ubuntu:~# gpg --export -o /home/stud/AlicePubKey.gpg
```

```
root@ubuntu:~# gpg --export-secret-key -o /home/stud/AlicePrivKey.gpg
```



## 2.5 Question 5 :

Pour le transfert du clé public d'Alice vers Bob, j'ai utilisé email pour faire cela.

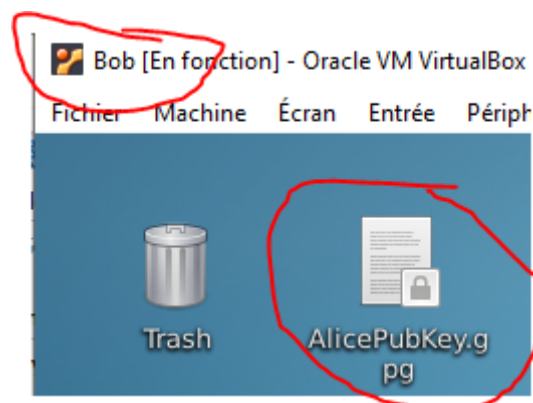


Figure 10. Transfère du clé public d'Alice vers Bob

## 2.6 Question 6 :

Pour cette question, on va refaire les même étapes fait à Alice pour Bob, en commence par la génération des clés, les afficher dans le trousseau de clés de Bob, exporter les clés publiques et privées dans des fichiers différents (BobPubKey.gpg et BobPrivKey.gpg), et finalement transférer la clé public de bob vers Alice.

```
gpg: /home/stud/.gnupg/trustdb.gpg: trustdb created
gpg: key 5501CCA3 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2021-05-14
pub 2048R/5501CCA3 2021-04-22 [expires: 2021-05-14]
    Key fingerprint = 8A4F AE0F BDCB 8B4F 5F1D C9BC 3CA4 D762 5501 CCA3
uid          meryam rh (ceri2) <meryamrh@gmail.com>
sub 2048R/543B6138 2021-04-22 [expires: 2021-05-14]
```

Figure 11. Création des clés pour Bob

```

root@ubuntu:~# gpg --list-keys
/home/stud/.gnupg/pubring.gpg
-----
pub   2048R/5501CCA3 2021-04-22 [expires: 2021-05-14]
uid           meryam rh (ceri2) <meryamrh@gmail.com>
sub   2048R/543B6138 2021-04-22 [expires: 2021-05-14]

root@ubuntu:~# gpg --list-secret-keys
/home/stud/.gnupg/secring.gpg
-----
sec   2048R/5501CCA3 2021-04-22 [expires: 2021-05-14]
uid           meryam rh (ceri2) <meryamrh@gmail.com>
ssb   2048R/543B6138 2021-04-22

```

Figure 12. Affichage de trousseau de clés de Bob

```

root@ubuntu:~# gpg --export -o /home/stud/BobPubkey.gpg
root@ubuntu:~# gpg --export-secret-key -o /home/stud/BobPrivkey.gpg
root@ubuntu:~# ls
BobPrivkey.gpg  BobPubkey.gpg  Desktop  Documents  Downloads  GNS3
root@ubuntu:~#

```

Figure 13. Exportation des clés de Bob

Et pour le transfère de la clé public de bob vers Alice, j'ai utilisé gmail.

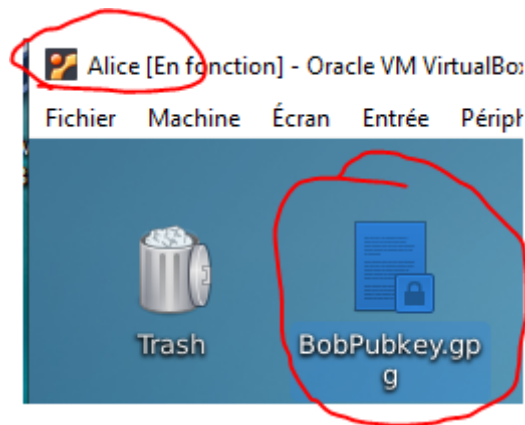


Figure 14. Transfère du clé public de bob vers Alice

## 2.7 Question 7 :

Pour l'ajoute de la clé publique d'Alice dans la machine de Bob, on vas utiliser la commande suivante : **gpg --import le chemin vers la clé public d'Alice**

```

root@ubuntu:~/Desktop# gpg --import AlicePubKey.gpg
gpg: key E8EECFE7: public key "meryam rhadi (ceri) <meryamrhadi@gmail.com>" imported
gpg: Total number processed: 1
gpg:             imported: 1 (RSA: 1)
root@ubuntu:~/Desktop# gpg --list-keys
/home/stud/.gnupg/pubring.gpg
-----
pub   2048R/5501CCA3 2021-04-22 [expires: 2021-05-14]
uid           meryam rh (ceri2) <meryamrh@gmail.com>
sub   2048R/543B6138 2021-04-22 [expires: 2021-05-14]

pub   2048R/E8EECFE7 2021-04-22 [expires: 2021-05-14]
uid           meryam rhadi (ceri) <meryamrhadi@gmail.com>
sub   2048R/0C1EFF1F 2021-04-22 [expires: 2021-05-14]

```

Figure 15. Ajoute et affichage de la clé public d'Alice dans le trousseau de la machine de Bob

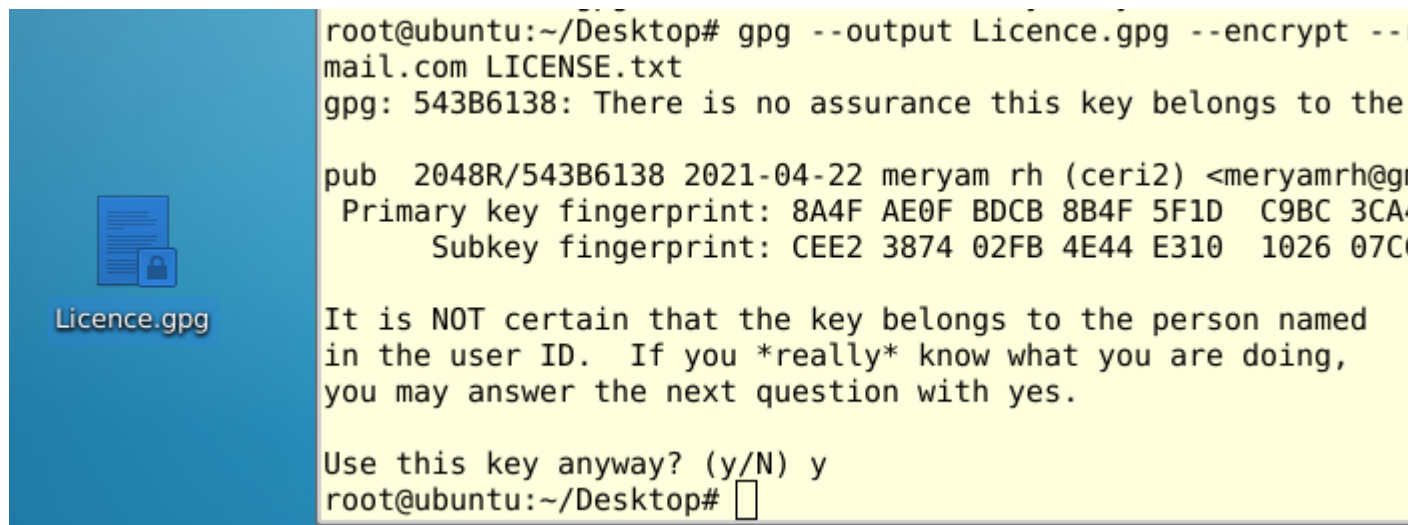


On remarque ici, quand on a fait la commande pour afficher le trousseau de Bob, que la clé d'Alice est bien ajoutée dans cette dernière.

## 2.8 Question 8 : Chiffrement

a- Le but de cette partie est de chiffrer le fichier **LICENSE.txt** dans la machine d'Alice avec la clé public de bob, qu'en a déjà ajouté dans le trousseau d'Alice, pour que seulement Bob qui peut déchiffrer se message avec sa clé privé, donc le chiffrer j'ai utilisé la commande de gpg :

**gpg --output Licence.gpg --encrypt --recipient meryamrh@gmail.com LICENSE.txt**. On voit dans la capture le résultat de chiffrement **LICENSE.gpg**.



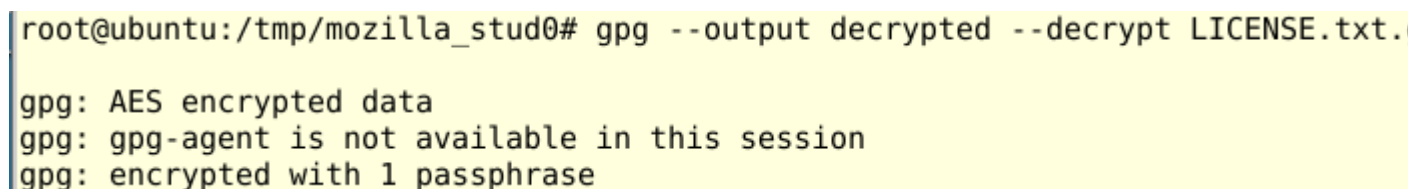
```

root@ubuntu:~/Desktop# gpg --output Licence.gpg --encrypt --recipient meryamrh@gmail.com LICENSE.txt
gpg: 543B6138: There is no assurance this key belongs to the
pub 2048R/543B6138 2021-04-22 meryam rh (ceri2) <meryamrh@gmail.com>
Primary key fingerprint: 8A4F AE0F BDCB 8B4F 5F1D C9BC 3CA
Subkey fingerprint: CEE2 3874 02FB 4E44 E310 1026 07C
It is NOT certain that the key belongs to the person named
in the user ID. If you *really* know what you are doing,
you may answer the next question with yes.
Use this key anyway? (y/N) y
root@ubuntu:~/Desktop#

```

Figure 16. Chiffrement du fichier

b- Sur la machine de bob, on va déchiffrer **License.gpg**, avec la commande **gpg --output decrypted --decrypt LICENSE.txt.gpg**



```

root@ubuntu:/tmp/mozilla_stud0# gpg --output decrypted --decrypt LICENSE.txt.gpg
gpg: AES encrypted data
gpg: gpg-agent is not available in this session
gpg: encrypted with 1 passphrase_

```

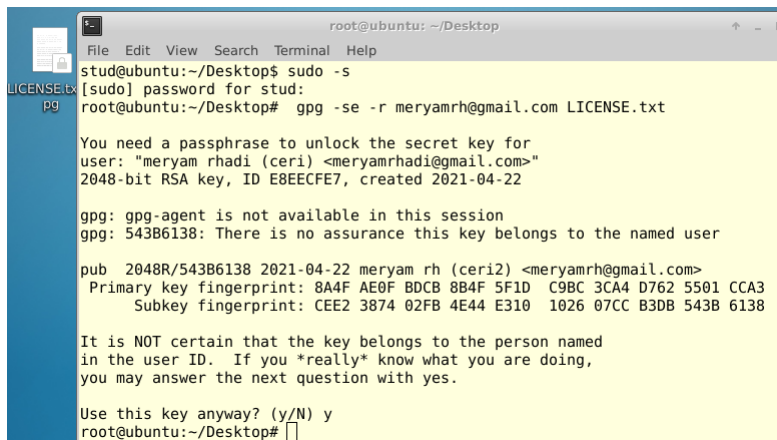
Figure 17. Déchiffrement du fichier

Et on obtient comme résultat le fichier d'origine **License.txt**, donc on remarque que on obtient le fichier source qui a été chiffré par Alice.



## 2.10 Question 10 : Signature et chiffrement

Pour cette dernière partie, on va chiffrer et signer au même temps le fichier LICENSE.txt, avec la commande :



```

root@ubuntu: ~/Desktop
File Edit View Search Terminal Help
stud@ubuntu:~/Desktop$ sudo -s
[sudo] password for stud:
root@ubuntu:~/Desktop# gpg -se -r meryamrh@gmail.com LICENSE.txt

You need a passphrase to unlock the secret key for
user: "meryam rhadi (ceri) <meryamrhadi@gmail.com>"
2048-bit RSA key, ID E8EECFE7, created 2021-04-22

gpg: gpg-agent is not available in this session
gpg: 543B6138: There is no assurance this key belongs to the named user

pub 2048R/543B6138 2021-04-22 meryam rh (ceri2) <meryamrh@gmail.com>
Primary key fingerprint: 8A4F AE0F BDCB 8B4F 5F1D C9BC 3CA4 D762 5501 CCA3
Subkey fingerprint: CEE2 3874 02FB 4E44 E310 1026 07CC B3DB 543B 6138

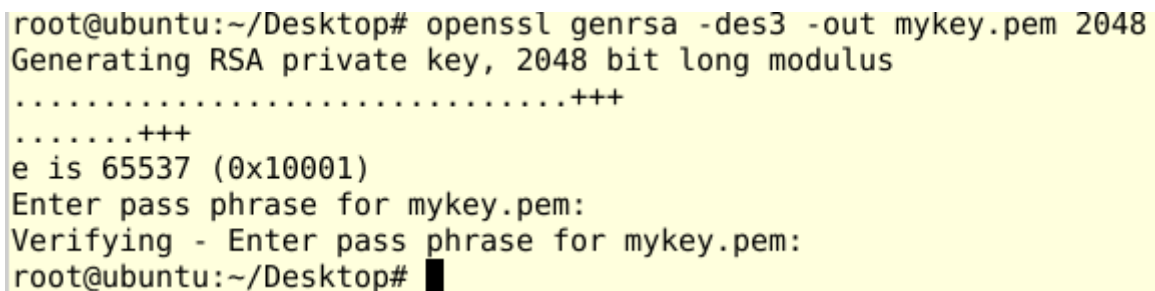
It is NOT certain that the key belongs to the person named
in the user ID. If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
root@ubuntu:~/Desktop#
  
```

## 3 Exercice 3 : Certificats

### 3.1 Question 1 :

Pour la génération du paire de clés RSA, d'une taille de 2048 bits, protégée par un mot de passe et nommé myKey.pem, on va utiliser la commande **openssl genrsa -des3 -out mykey.pem 2048**

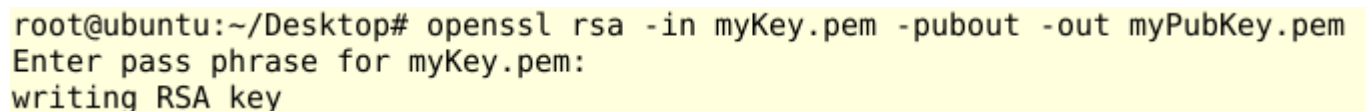


```

root@ubuntu:~/Desktop# openssl genrsa -des3 -out mykey.pem 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for mykey.pem:
Verifying - Enter pass phrase for mykey.pem:
root@ubuntu:~/Desktop#
  
```

### 3.2 Question 2 :

Pour la création du fichier, qui ne contient que la partie publique du clé RSA, on va utiliser la commande **openssl rsa -in myKey.pem -pubout -out myPubKey.pem**



```

root@ubuntu:~/Desktop# openssl rsa -in myKey.pem -pubout -out myPubKey.pem
Enter pass phrase for myKey.pem:
writing RSA key
  
```

Figure 22. Commande pour la création d'un fichier de la clé public

```

GNU nano 2.5.3                               File: myPubKey.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtld13XAt4Tk3Hu2eil9x
PEN/Sni2nnVw31i28+KBjmq8lyZ78YUtl0CP086HzhiRmHcrDeX0lb8iNgCPWWH3
/GdB9zEnovhwI2LAmPs1JnBUdCe4S6xWHYXa//9naDPTV69YmjQ2q+vHyAT07tfu
07ZWC32Qg7CBF5Y3G1ZARSt1PF8Gddh6vKRzpkimC8k4WRDuq2htAzSGoPl/6x6V
xKx2v/oSVNNZCsn2RJN6Bl9wfBziJCbp9cAaWwfttxAJ2EiD3S8cld/JuWIFjSlH
+/C1W7hk7Uhmt8oL0ubeVAGXUZt/6t5pypNGQBNnFtVmJTzVHWZ0PFyiGvo+Xl5c
8wIDAQAB
-----END PUBLIC KEY-----

```

Figure 23. Le contenu du fichier myPubKey.pem

### 3.3 Question 3 :

Pour la création du certificat, on vas utiliser la commande **openssl req -new -key my-Key.pem -out myRequest.pem**

```

root@ubuntu:~/Desktop# openssl req -new -key mykey.pem -out maRequete.pem
Enter pass phrase for mykey.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:fr
State or Province Name (full name) [Some-State]:vaucluse
Locality Name (eg, city) []:avignon
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ceri
Organizational Unit Name (eg, section) []:openssl
Common Name (e.g. server FQDN or YOUR name) []:rhadi
Email Address []:meryamrhadi@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:maryame2
An optional company name []:ceri2
root@ubuntu:~/Desktop# █

```

```

root@ubuntu:~/Desktop# cat myRequest.pem
-----BEGIN CERTIFICATE REQUEST-----
MIIC/TCCAeUCAQAwYgxCzAJBgNVBAYTAMZyMREwDwYDVQQIDAh2YXVjbHVzZTEQ
MA4GA1UEBwwHYXZpZ25vb2JENMAsGA1UECgwEY2VyaTE0MAwGA1UECwwFcmhhZGKx
DzANBgNVBAMMBm1lcnlhbTEkMCIGCSqGSIb3DQEJARYVbWVyeWFtcmhhZGZlAZ21h
aWwuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtld13XAt4Tk3
Hu2eil9xPEN/Sni2nnVw3li28+KBjmq8lyZ78YUtl0CP086HzhiRmHcrDeX0lb8i
NgCPWwH3/GdB9zEnovhwI2LAmPs1JnBUdCe4S6xWHYXa//9naDPTV69YmjQ2q+vH
yAT07tfu07ZWC32Qg7CBF5Y3G1ZARSt1PF8Gddh6vKRzpkimC8k4WRDuq2htAzSG
oPl/6x6VxKx2v/oSVNNZCsn2RJN6BL9wfBziJCbp9cAaWwfttxAJ2EiD3S8cld/J
uWIFjSLH+/C1W7hk7UhmT8oL0ubeVAGXUZt/6t5pypNGQBNnFtVmJTzVHWZOPFyi
Gvo+Xl5c8wIDAQABOC8wFAYJKoZIhvcNAQkCMQcMBWNlcmkyMBcGCSqGSIb3DQEJ
BzEKDAhtYXJ5Yw1lMjANBgkqhkiG9w0BAQsFAAOCAQEAEihEvjKzLhJyExTzndxB
dUDQJxPmc1nTIZhnE6Fh2fZ+2Nlv8+StaFdjZZGlstYRUlgqDurnG3FWLCJLvbTx
+3IdbLYONXMKqaKTEKCV6FoQ8WxOKIZGUUD4QKCVqU6tRLEudNtCumb+f0/GJX6V
vzCQx3Z7FfVMdo+PQzV1Qf0KiW57iVY7w6YA80inqUSI5/M2cyrrVPLxgZIprfVg
aUAdwuBG8+DG8ddAyd4ZtBA4iJiNGG7I00j3HS45glyg3+c9NonoXio/Iixp0mpv
8UKJBN85lSmj7B8wpXKuhRcJkCepo6GmFouTeugSy+IX0Nhy5/Rfx3B9K0jYHA9J
ig==
-----END CERTIFICATE REQUEST-----

```

Figure 24. Résultat de la commande cat pour voir le Contenu de myRequest

### 3.4 Question 4 :

Pour la commande **openssl req -in myRequest.pem -text -noout**, on a l'option **-text** demande l'affichage décodé de la paire de clés et l'option **-noout** supprime la sortie normalement produite par la commande **rsa**, en général on a obtenu des informations concernant notre certificat, et on a aussi la présence de la clé public, par contre la clé privé ne figure pas sur ces informations.

```

root@ubuntu:~/Desktop# openssl req -in myRequest.pem -text -noout
Certificate Request:
Data:
  Version: 0 (0x0)
  Subject: C=fr, ST=vaucuse, L=avignon, O=ceri, OU=ueo, CN=rhadi/emailAddress=meryamrhadi@gmail.com
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
        00:9c:09:63:4f:12:ca:a2:38:65:21:56:1d:ea:3d:
        e9:29:0f:30:b0:e6:36:04:90:98:fe:84:fc:a4:c2:
        a7:10:d1:13:9b:2c:fb:d3:85:16:f5:fa:63:33:03:
        e0:39:4c:43:7a:36:92:b8:08:9f:2d:d7:97:30:e9:
        2a:60:2b:f6:ce:f5:51:f6:60:ef:38:fe:74:03:b9:
        6b:e0:f7:1b:10:48:48:87:c1:26:1a:c8:a4:7d:6e:
        9a:fc:07:c9:d4:5a:e8:53:3b:e6:01:f9:3c:82:64:
        08:66:f6:37:77:f1:ff:1a:48:a2:e9:6b:4d:36:ca:
        fa:b9:70:1c:76:a8:75:82:33:ea:59:55:5f:b4:5c:
        ac:71:5e:6e:64:e9:e3:2d:38:86:0a:6f:f5:c7:22:
        f1:1a:2a:ee:44:57:25:4f:f5:4e:31:fa:b5:fe:67:
        d6:42:69:05:54:84:e9:c1:70:24:cc:d6:d2:26:97:
        69:6f:1b:a4:ad:46:f9:bd:49:a9:b9:07:7b:d6:36:
        52:65:62:3b:b8:a8:ad:f5:98:9c:02:0d:57:eb:94:
        67:95:d2:d9:55:9e:1e:d7:b3:a1:d5:11:63:8a:75:
        44:93:53:57:66:64:94:6e:c4:b9:ce:c2:67:ba:24:
        0b:7b:61:da:b0:8b:c0:b0:bf:b5:e7:6e:3c:22:31:
        28:a5
      Exponent: 65537 (0x10001)

```



```

      Exponent: 65537 (0x10001)
    Attributes:
      unstructuredName      :unable to print attribute
      challengePassword     :unable to print attribute
    Signature Algorithm: sha256WithRSAEncryption
      93:41:a9:3c:db:e7:a8:ec:26:df:59:02:31:a7:c3:e6:c4:c7:
      2a:2f:7f:8f:ec:88:cb:39:56:32:d5:2f:92:bd:06:f7:4d:55:
      9c:85:e6:3e:20:f6:70:6f:fe:82:19:8c:fe:b9:38:44:68:51:
      e3:fa:43:41:d3:05:29:ef:0d:11:ff:e0:b0:e4:57:79:84:d9:
      0a:a3:3b:eb:ea:c7:17:7f:78:45:1f:a5:f3:e6:45:03:81:ca:
      03:82:07:82:c3:4e:c1:1d:7e:16:69:26:6a:d3:35:d3:7e:7b:
      8c:c6:d9:2c:03:83:cc:70:c9:fc:83:52:25:cc:d4:c4:99:9d:
      5b:54:9d:8c:f6:3a:a5:99:aa:b2:ab:d4:54:89:18:c8:8c:08:
      b6:08:ba:8e:75:50:8b:4d:3a:2e:16:53:70:d2:fe:71:76:77:
      dd:ee:45:44:31:1c:cf:13:b2:b4:4b:1e:ee:43:29:c0:18:09:
      9c:44:df:a7:39:69:3d:a2:c7:b5:ef:14:59:17:b5:45:7a:d5:
      38:f6:7e:c6:e2:e0:d8:81:fd:3f:43:7c:e7:9c:e8:08:f3:2f:
      67:01:b6:65:e6:d8:ee:ff:01:04:5f:60:21:c8:d0:a1:41:bf:
      c5:f0:c7:38:22:a7:55:48:22:34:54:61:a6:1c:7e:09:e8:07:
      67:dd:b4:03

```

### 3.5 Question 5 :

Pour cette question, après la finalisation de l'établissement de la requête de certificat, il faut maintenant qu'en signe nous même notre certificat, en commençant par la création de la clé privée de l'autorité de certification, avec la commande : **openssl genrsa -des3 1024 > CA.key**

```

root@ubuntu:~/Desktop# openssl genrsa -des3 1024 > CA.key
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
Enter pass phrase:
Verifying - Enter pass phrase:
root@ubuntu:~/Desktop# ls
CA.key  LICENSE.txt  maRequete.pem  mykey.pem  myPubKey.pem  myRequest.pem
root@ubuntu:~/Desktop# █

```

Figure 25. La commande pour la création de la clé privée de l'autorité de certification

### 3.6 Question 6 :

Ensuite, à partir de la clé privée qu'en a généré précédemment , on crée un certificat x509 pour une durée de validité d'un an auto-signé avec la commande **openssl req -new -x509 -days 365 -key CA.key > CA.crt**

```

root@ubuntu:~/Desktop# openssl req -new -x509 -days 365 -key CA.key > CA.crt
Enter pass phrase for CA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:fr
State or Province Name (full name) [Some-State]:vacluse
Locality Name (eg, city) []:avignon
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ceri
Organizational Unit Name (eg, section) []:security
Common Name (e.g. server FQDN or YOUR name) []:meryam rhadi
Email Address []:meryamrhadi@gmail.com
root@ubuntu:~/Desktop# ls
CA.crt CA.key LICENSE.txt maRequete.pem mykey.pem myPubKey.pem myReques
root@ubuntu:~/Desktop#

```

Figure 26. La commande pour la création d'un certificat x509 à partir de la clé privée

### 3.7 Question 7 :

Maintenant à partir de myKey.pem, on vas crée un fichier de demande de signature de certificat, avec la commande **openssl req -new -key mykey.pem > myPubKey.csr**

```

root@ubuntu:~/Desktop# openssl req -new -key myKey.pem > myPubKey.csr
Enter pass phrase for myKey.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:fr
State or Province Name (full name) [Some-State]:vacluse
Locality Name (eg, city) []:avignon
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ceri
Organizational Unit Name (eg, section) []:security
Common Name (e.g. server FQDN or YOUR name) []:rhadi
Email Address []:meryamrhadi@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:maryame2
An optional company name []:.

```

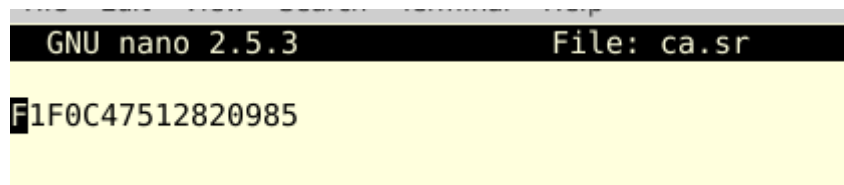
### 3.8 Question 8 :

Pour signer notre demande de signature de certificat, on utilisera la commande suivante : **openssl x509 -req -in myPubKey.csr -out myPubKey.crt -CA CA.crt -CAkey CA.key -CAcreateserial -CAserial ca.srl**,

### 3.9 Question 9 :

il nous donne comme résultat :

```
root@ubuntu:~/Desktop# openssl x509 -req -in myPubKey.csr -out myPubKey.crt
-C A CA.crt -CAkey CA.key -CAcreateserial -CAserial ca.sr
Signature ok
subject=/C=fr/ST=vaucluse/L=avignon/O=ceri/OU=security/CN=rhadi/emailAddress=meryamrhadi@gmail.com
Getting CA Private Key
Enter pass phrase for CA.key:
root@ubuntu:~/Desktop# ls
CA.crt  ca.sr      mykey.pem    myPubKey.csr  myRequest.pem
CA.key  LICENSE.txt myPubKey.crt myPubKey.pem
root@ubuntu:~/Desktop# nano ca.sr
```



```
GNU nano 2.5.3      File: ca.sr
F1F0C47512820985
```

Figure 27. Contenu de ca.sr