Asymmetric Encryption

Miguel Frade – Nuno Rasteiro

Department of Informatics Engineering
School of Technology and Management, Polytechnic Institute of Leiria

November 2019

Asymmetric Encryption - Introduction

- Asymmetric Encryption uses a 2-key system, a private and a public key.
- Each user uses a key pair, where the public key is distributed between all the users you need to communicate with
- The emitter encrypts the message with the public key of the receiver
- Only the private key of the receiver can decrypt the message

Asymmetric Encryption - Introduction

- This protocol solves the key exchange problem of the symmetric encryption
- the emitter and receiver need to exchange the public key (from the receiver)
- This public key can be used for all communications with the receiver
- You will need “*n* “key pairs for “*n* “ persons that can privately communicate

Asymmetric Encryption - Introduction

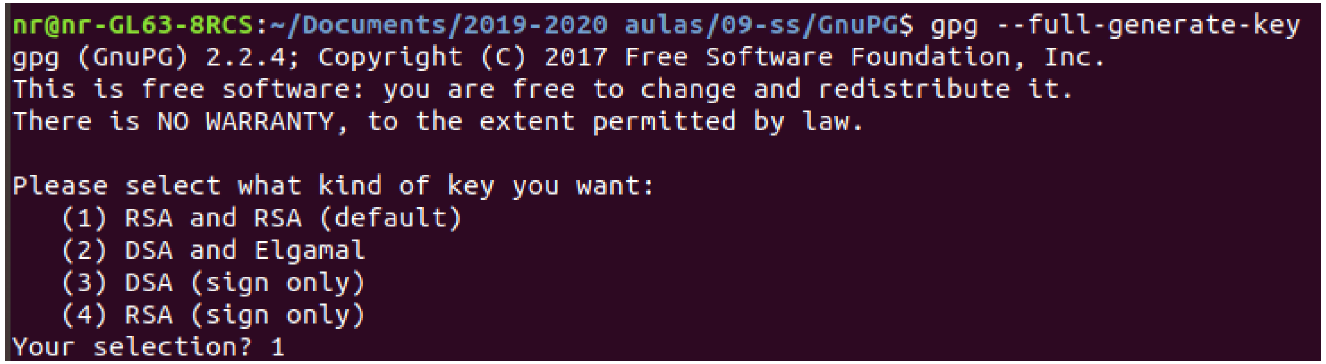
- The security of the symmetric and asymmetric encryptions relies on the keys.
- The length of the key is a good indicator of security, however it is not comparable the between the symmetric and public key system.
- On a brut force attack of a symmetric key with 128 bits, there are 2^{128} possible keys. For a public key with 512 bits, the attacker has to factorize a 512-bit number (up to 155 decimal digits)
- For symmetric keys 128 bits is sufficient, however for the public key, and due to the different existing factorization algorithms, a 1024 bit public key should be used

GnuPG

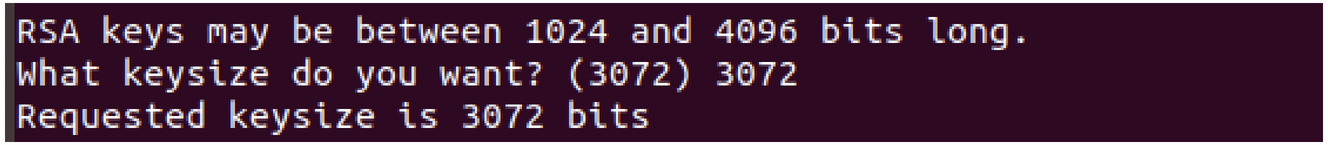
- GnuPG is a tool the store and communicate files in a secure way
- It relies on several cryptographics systems:
  - Symmetric encryption
  - Public key
  - Hashing

GnuPG – Generate keys

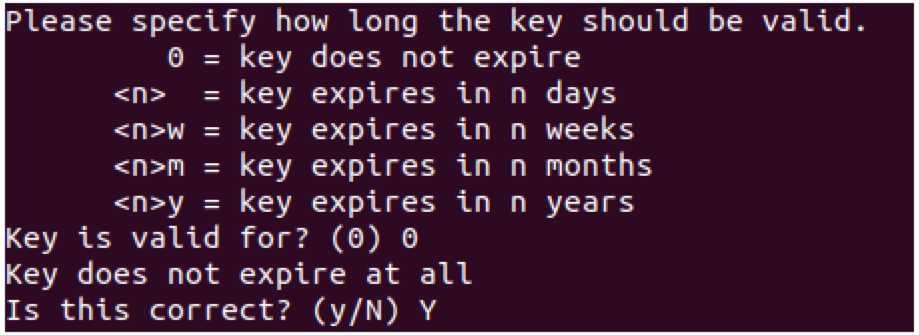
- To create the key pairs:
  - Generate keys → gpg –full-generate-key
  - Choose the option by default (1) to generate 2 RSA key pairs one for the signatures and another to encrypt
  - Define the key length (default 3072): 3072 - the bigger the length, stronger is the key against but force attack but also it will take longer the encryption and decryption method
  - Choose how long the key should be valid



```
nr@nr-GL63-8RCS:~/Documents/2019-2020 aulas/09-ss/GnuPG$ gpg --full-generate-key
gpg (GnuPG) 2.2.4; Copyright (C) 2017 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
```



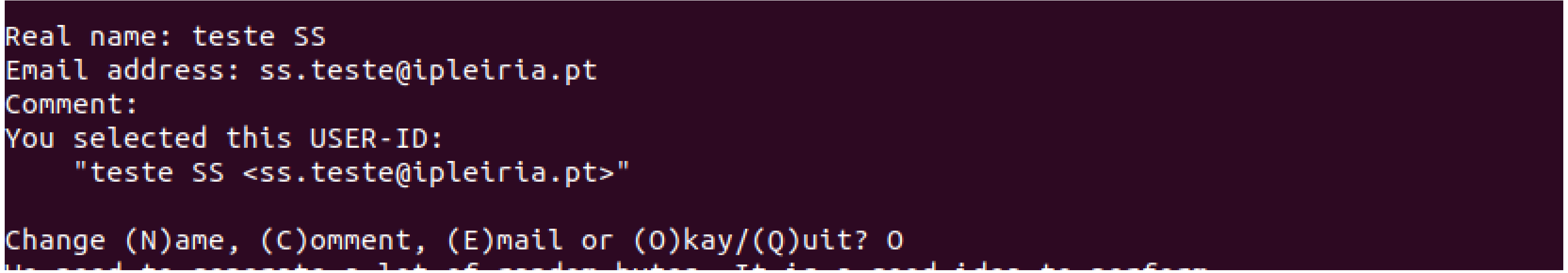
```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 3072
Requested keysize is 3072 bits
```



```
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) 0
Key does not expire at all
Is this correct? (y/N) Y
```

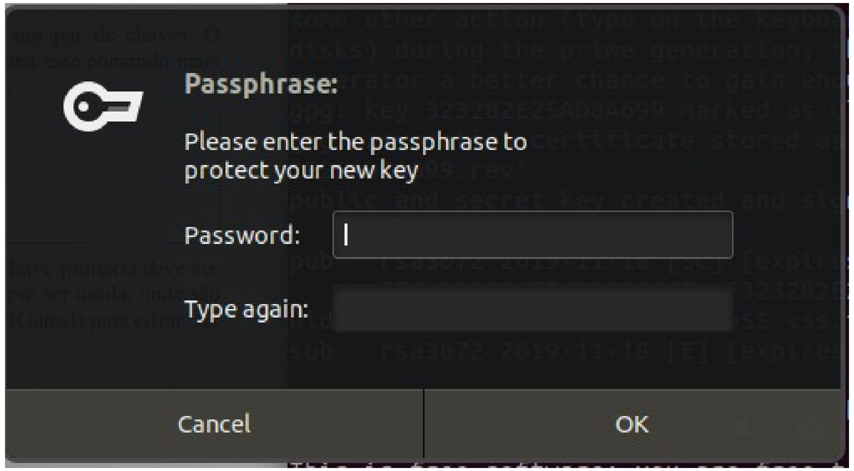
GnuPG – Generate keys

- GnuPG needs to construct a user ID to identify the key



```
Real name: teste SS
Email address: ss.teste@ipleiria.pt
Comment:
You selected this USER-ID:
    "teste SS <ss.teste@ipleiria.pt>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```

The passphrase is requested to protect the private key



Passphrase:

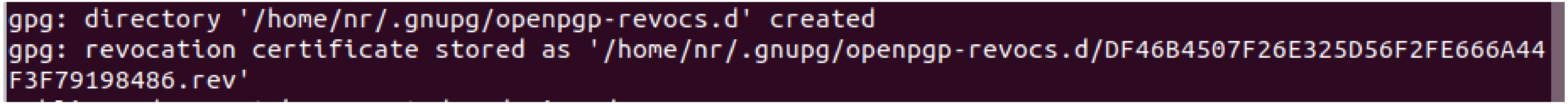
Please enter the passphrase to protect your new key

Password:

Type again:

Create a revoke certificate

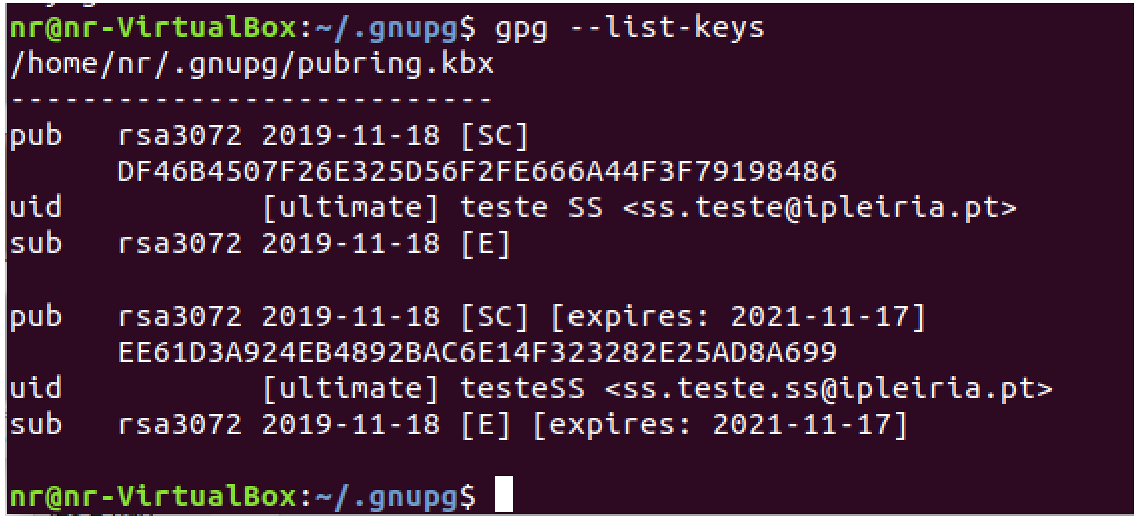
- a revoke key certificate is created automatically to use in case the passphrase is forgoten or if the private key has been compremized
- This certificate allows to verify the digital signitures but does not allow to encrypt
- Th key is normally stored in:



```
gpg: directory '/home/nr/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/nr/.gnupg/openpgp-revocs.d/DF46B4507F26E325D56F2FE666A44
F3F79198486.rev'
```

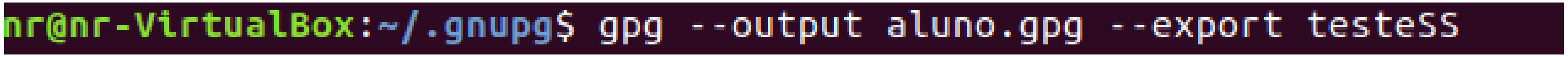
Exchange keys

- To communicate you need to exchange the Public keys. To list the keys:



```
nr@nr-VirtualBox:~/.gnupg$ gpg --list-keys
/home/nr/.gnupg/pubring.kbx
---------------------------
pub   rsa3072 2019-11-18 [SC]
      DF46B4507F26E325D56F2FE666A44F3F79198486
uid           [ultimate] teste SS <ss.teste@ipleiria.pt>
sub   rsa3072 2019-11-18 [E]

pub   rsa3072 2019-11-18 [SC] [expires: 2021-11-17]
      EE61D3A924EB4892BAC6E14F323282E25AD8A699
uid           [ultimate] testeSS <ss.teste.ss@ipleiria.pt>
sub   rsa3072 2019-11-18 [E] [expires: 2021-11-17]

nr@nr-VirtualBox:~/.gnupg$
```

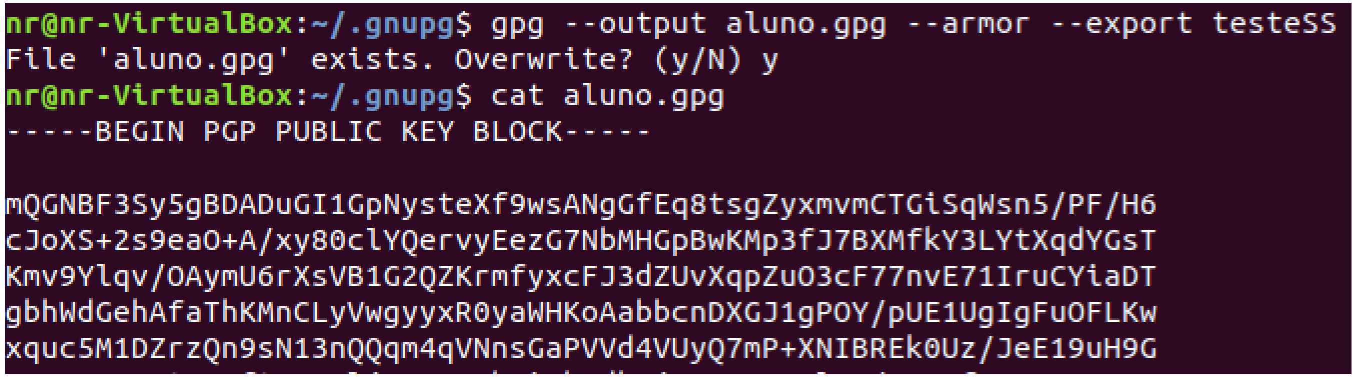
- We need to extract the key:



```
nr@nr-VirtualBox:~/.gnupg$ gpg --output aluno.gpg --export testeSS
```

- To send by e-mail we will need to use the option --armor

  as it will create a ascii format



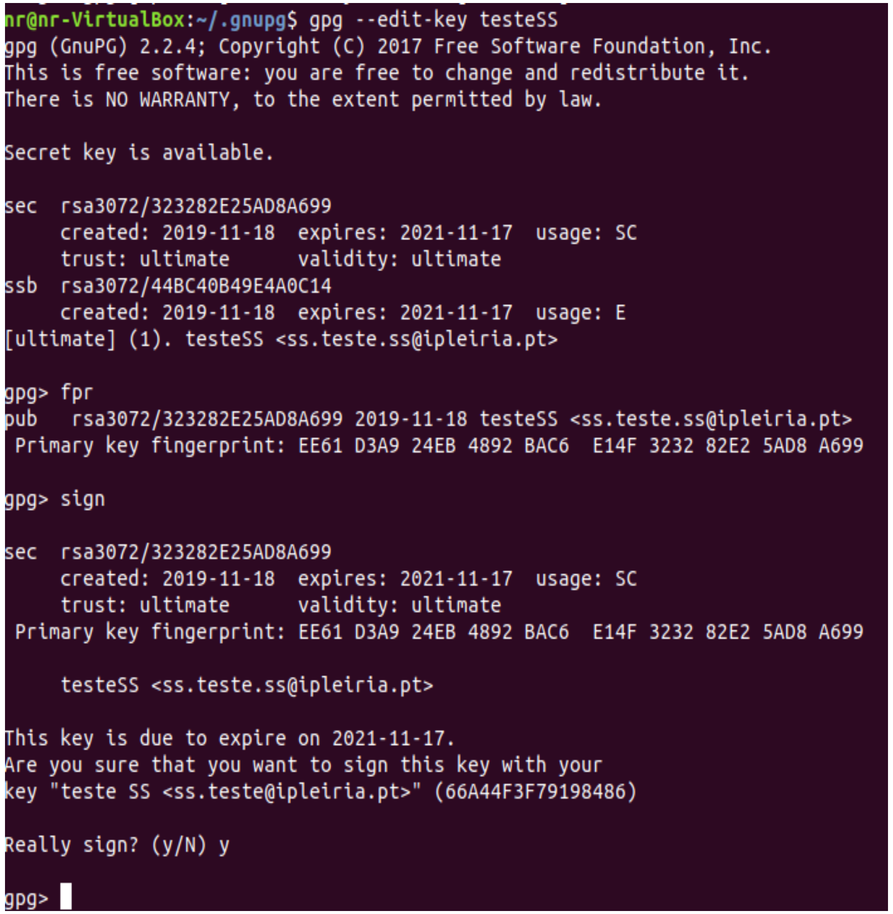
```
nr@nr-VirtualBox:~/.gnupg$ gpg --output aluno.gpg --armor --export testeSS
File 'aluno.gpg' exists. Overwrite? (y/N) y
nr@nr-VirtualBox:~/.gnupg$ cat aluno.gpg
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBF3Sy5gBDADuGI1GpNysteXf9wsANgGfEq8tsgZyxmvmCTGiSqWsn5/PF/H6
cJoXS+2s9eaO+A/xy80clYQervyEezG7NbMHGpBwKMp3fJ7BXMfkY3LYtXqdYGsT
Kmv9Ylqv/OAymU6rXsVB1G2QZKrmfyxcFJ3dZUvXqpZuO3cF77nvE71IruCYiaDT
gbhWdGehAfaThKMnCLyVwgyyxR0yaWHKoAabbcnDXGJ1gPOY/pUE1UgIgFuOFLKw
xquc5M1DZrzQn9sN13nQQqm4qVNnsGaPVVd4VUyQ7mP+XNIBREk0Uz/JeE19uH9G
```

Exchange keys

- To import a key: “gpg --import aluno.gpg”
- You should then verify the key. This consists in authenticity of the key with the command “gpg –edit-key UID”
- Using the “fpr” command it will print

  the fingerprint of the key and will

  need to confirm in a secure way with the one provided by sender
- After confirmimg the fingerpring we should sign the key with the command “sign”

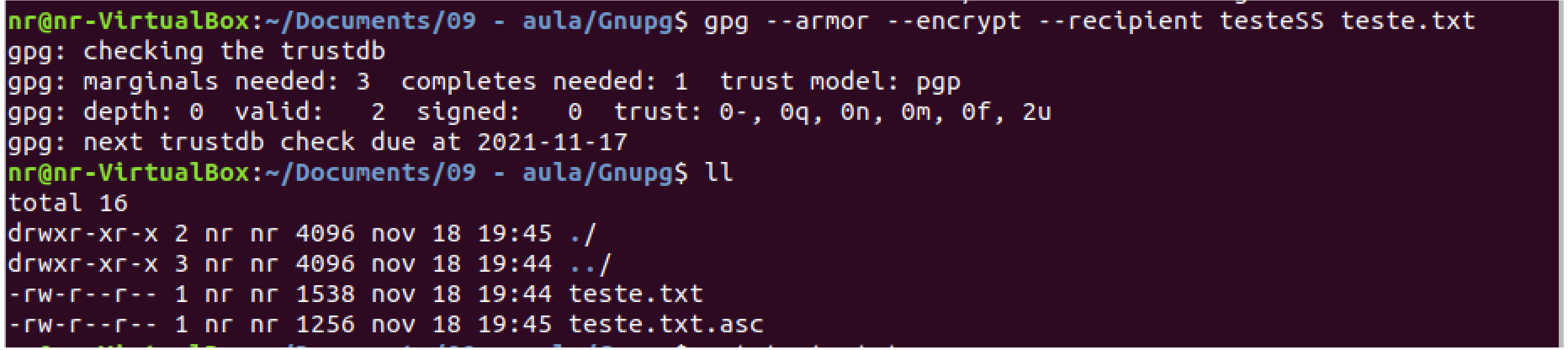


```
nr@nr-VirtualBox:~/.gnupg$ gpg --edit-key testeSS
gpg (GnuPG) 2.2.4; Copyright (C) 2017 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

sec  rsa3072/323282E25AD8A699
     created: 2019-11-18  expires: 2021-11-17  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/44BC40B49E4A0C14
     created: 2019-11-18  expires: 2021-11-17  usage: E
[ultimate] (1). testeSS <ss.teste.ss@ipleiria.pt>

gpg> fpr
pub   rsa3072/323282E25AD8A699 2019-11-18 testeSS <ss.teste.ss@ipleiria.pt>
 Primary key fingerprint: EE61 D3A9 24EB 4892 BAC6  E14F 3232 82E2 5AD8 A699

gpg> sign

sec  rsa3072/323282E25AD8A699
     created: 2019-11-18  expires: 2021-11-17  usage: SC
     trust: ultimate      validity: ultimate
 Primary key fingerprint: EE61 D3A9 24EB 4892 BAC6  E14F 3232 82E2 5AD8 A699

     testeSS <ss.teste.ss@ipleiria.pt>

This key is due to expire on 2021-11-17.
Are you sure that you want to sign this key with your
key "teste SS <ss.teste@ipleiria.pt>" (66A44F3F79198486)

Really sign? (y/N) y

gpg> 
```

Encrypt file

- To encrypt a document we can use the following command:

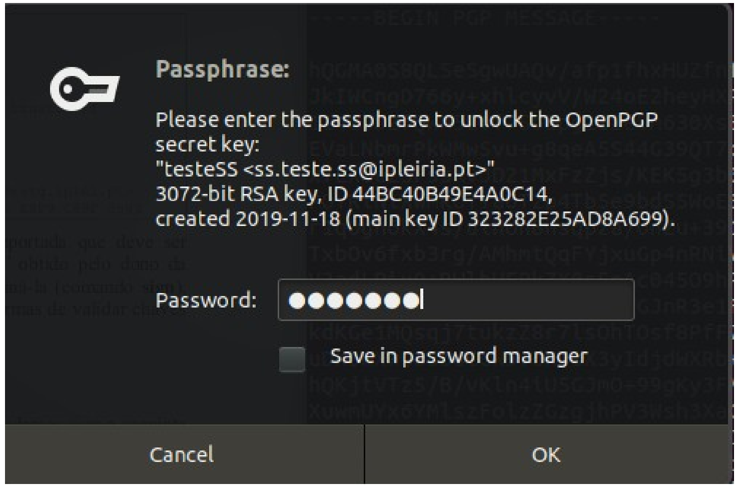
  gpg --armor --encrypt --recipient testeSS teste.txt



```
nr@nr-VirtualBox:~/Documents/09 - aula/Gnupg$ gpg --armor --encrypt --recipient testeSS teste.txt
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2021-11-17
nr@nr-VirtualBox:~/Documents/09 - aula/Gnupg$ ll
total 16
drwxr-xr-x 2 nr nr 4096 nov 18 19:45 ./
drwxr-xr-x 3 nr nr 4096 nov 18 19:44 ../
-rw-r--r-- 1 nr nr 1538 nov 18 19:44 teste.txt
-rw-r--r-- 1 nr nr 1256 nov 18 19:45 teste.txt.asc
```

- The file teste.txt.asc is created encrypted with the public key of testeSS.

Encrypt file



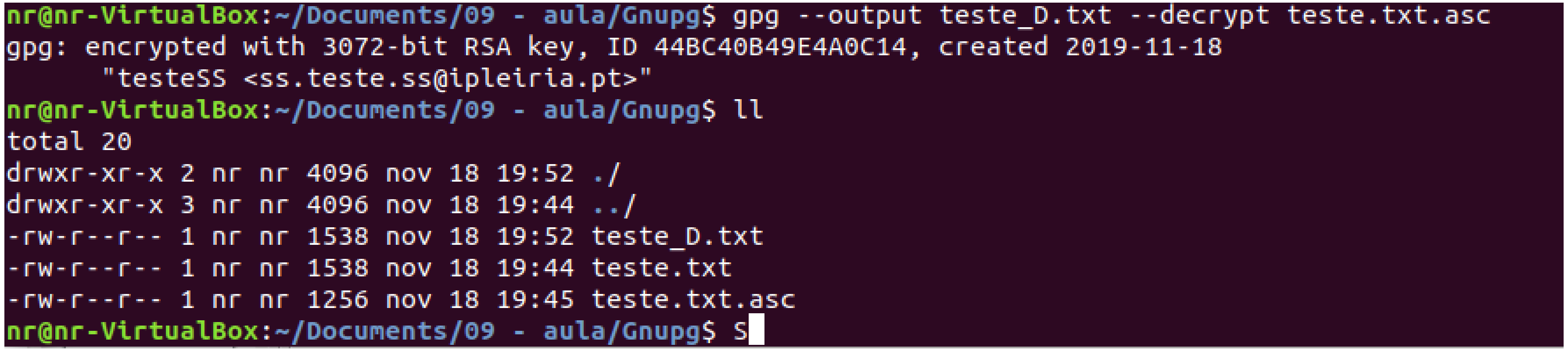
Passphrase:

Please enter the passphrase to unlock the OpenPGP secret key:
"testeSS <ss.teste.ss@ipleiria.pt>"
3072-bit RSA key, ID 44BC40B49E4A0C14,
created 2019-11-18 (main key ID 323282E25AD8A699).

Password: ●●●●●●●

□ Save in password manager

- To read the file you will need a private key and execute:
  - gpg --output teste_D.txt --decrypt teste.txt.asc
  - Passphrase will be requested
- Output:



```
nr@nr-VirtualBox:~/Documents/09 - aula/Gnupg$ gpg --output teste_D.txt --decrypt teste.txt.asc
gpg: encrypted with 3072-bit RSA key, ID 44BC40B49E4A0C14, created 2019-11-18
      "testeSS <ss.teste.ss@ipleiria.pt>"
nr@nr-VirtualBox:~/Documents/09 - aula/Gnupg$ ll
total 20
drwxr-xr-x 2 nr nr 4096 nov 18 19:52 ./
drwxr-xr-x 3 nr nr 4096 nov 18 19:44 ../
-rw-r--r-- 1 nr nr 1538 nov 18 19:52 teste_D.txt
-rw-r--r-- 1 nr nr 1538 nov 18 19:44 teste.txt
-rw-r--r-- 1 nr nr 1256 nov 18 19:45 teste.txt.asc
nr@nr-VirtualBox:~/Documents/09 - aula/Gnupg$ S
```

Encrypt file

- If you wand to create a backup only you can read the option symmetric can be used:
  - gpg --symmetric teste.txt
- And a *.gpg file will be created with the encryption

Sign a Document

- A sign document guaranties intigrity and authenticity of the file without encrypting it
- We will need to sign the document and save the signiture in another seperate file
- To create it:

  -
    ```
    gpg --armor --output ~/doc2.sig --detach-sig ficheiro.txt
    ```

  -
    ```
    gpg --verify doc2.sig ficheiro.txt
    ```

Exercises

1. Generate a pair of keys
2. Export the key to une file, in binary format and ASCII. Verify the differences.
3. Import the public keys from your colleagues
4. Validate the public keys

Exercises

1.Authentication and file security

a)Create a file with the command : ls */etc/* > ls.txt*

b)Sign the file ls.txt

c)Create a hash sha-1 in a seperate file sha1.txt ans sign the file

d)What are the differences between b and c?

2.-share information

a)Share with a colleague the files ls.txt, ls.txt.asc and sha1.txt

b)Validate the information