



Certificado de Profesionalidad en Seguridad Informática

IronHack - SOC

Módulo 4

Criptografía - Certificados

Práctica 3

Alumno: Julián Gordon

Indice

Enunciado.....	3
Actividad 3 - Certificados.....	3
Introducción.....	4
Certificados PGP.....	5
Linux.....	5
GPA The Gnu Privacy Assistant.....	10
Creación de fichero y cifrado.....	14
Windows.....	16
Instalación y uso de Kleopatra.....	16
Creación de fichero y cifrado.....	22
Creación de un Certificado X.509 Autofirmado.....	25
Conceptos teóricos.....	25
Certificado AutoFirmado.....	25
Autoridad Certificadora (CA).....	26
OpenSSL.....	26
Certificado X.509 Autofirmado con OpenSSL.....	26
Instalación de OpenSSL.....	28
Windows.....	28
Linux.....	35
Conclusiones.....	38
Certificados PGP.....	38
Certificados X.509 Autofirmados.....	38
Aprendizajes Clave.....	39

Enunciado

Actividad 3 - Certificados

- Trabajaremos con certificados PGP. Para ello, el profesor te entregará un certificado con su clave pública, que tendrás que importar en tu programa, GPG4Win o PGP en Linux, y tendrás que crear un fichero de texto que contenga el texto “Solución P3” y cifrarlo para el profesor. Hay que realizar la práctica en ambos sistemas operativos.
- Adicionalmente, e igualmente en ambos sistemas operativos, procederemos a la creación de un certificado x509 autofirmado, para lo que crearemos primero la solicitud y luego la firmaremos con la CA que se crea con la instalación de la suite OpenSSL.

Introducción

En el ámbito de la seguridad informática y la criptografía, los certificados digitales y las claves públicas juegan un papel fundamental en la protección y autenticidad de los datos. En este ejercicio, nos centraremos en la utilización de certificados PGP (Pretty Good Privacy) y certificados X.509, dos estándares ampliamente reconocidos para la encriptación y firma digital.

PGP es un método de cifrado que proporciona privacidad y autenticidad a los datos a través de la criptografía de clave pública. El proceso involucra la utilización de un par de claves: una clave pública, que puede ser distribuida abiertamente, y una clave privada, que se mantiene en secreto. La clave pública se usa para cifrar los datos, y solo la clave privada correspondiente puede descifrarlos. Este método garantiza que sólo el destinatario previsto pueda acceder a la información cifrada. En este ejercicio, utilizaremos GPG4Win en Windows y PGP en Linux para manejar certificados PGP. La tarea implica importar un certificado con la clave pública del profesor, crear un archivo de texto con el contenido "Solución P3" y cifrarlo, utilizando la clave pública del profesor. Esta práctica asegura que el mensaje pueda ser leído únicamente por el destinatario, proporcionando una experiencia práctica en la utilización de PGP en diferentes sistemas operativos.

Además, abordaremos la creación de certificados X.509 autofirmados utilizando OpenSSL, una herramienta robusta y ampliamente utilizada para la criptografía de clave pública. Los certificados X.509 son esenciales para asegurar la comunicación en redes y sistemas, proporcionando una forma de autenticación y encriptación. La creación de un certificado autofirmado implica generar una solicitud de firma de certificado (CSR) y luego firmar con una autoridad certificadora (CA). En nuestro caso, la CA será creada como parte de la suite OpenSSL. Este ejercicio se realizará tanto en Windows como en Linux, proporcionando una comprensión integral del proceso en ambas plataformas.

Este ejercicio no solo refuerza el conocimiento teórico sobre criptografía y seguridad de la información, sino que también proporciona habilidades prácticas en el manejo de herramientas y tecnologías cruciales para proteger y autenticar datos. Al finalizar, tendremos una experiencia concreta en el uso de PGP y certificados X.509, preparándonos mejor para enfrentar desafíos de seguridad en entornos profesionales.

Certificados PGP

Linux

Comenzaremos este proceso desde nuestra máquina virtual de Kali Linux. Como norma general, empezamos haciendo un apt update y upgrade para confirmar si el sistema necesita actualizaciones. Una vez tenemos nuestro sistema actualizado, procederemos a instalar el programa gnupg2, que sirve para implementar PGP en Linux.

```
(root@kali)-[/home/kali]
# apt-get install gap gnupg2
```

Una vez instalado veremos el siguiente mensaje.

```
(root@kali)-[/home/kali]
# apt-get install gap gnupg2
Reading package lists... Done
Building dependency tree ... Done
Reading state information... Done
gap is already the newest version (4.13.0-2).
gnupg2 is already the newest version (2.2.40-3).
The following packages were automatically installed and are no longer required:
 libabsl20220623 libadwaita-1-0 libaio1 libappstream5 libatk-adaptor libboost-dev libboost1.83-dev libgphoto2-l10n libnsl-dev libopenblas-dev
 libopenblas-pthread-dev libopenblas0 libpthread-stubs0-dev libpython3-all-dev libpython3.12 libpython3.12-dev libstemmer0d libtirpc-dev libxmlb2
 libxsimd-dev python3-all-dev python3-anyjson python3-beniget python3-editables python3-gast python3-pyatspi python3-pypdf2 python3-pyppeteer
 python3-pyrsistent python3-pythran python3.12-dev xtl-dev zenity zenity-common
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 774 not upgraded.
```

El siguiente paso que vamos a realizar será la creación de la llave. Para esto usamos el comando: `gpg2 --gen-key`

```
(root@kali)-[/home/kali]
# gpg2 --gen-key
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: directory '/root/.gnupg' created
gpg: keybox '/root/.gnupg/pubring.kbx' created
Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Julius Hacker
Email address: julius.hacker88@gmail.com
You selected this USER-ID:
    "Julius Hacker <julius.hacker88@gmail.com>"

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

Una vez le damos a Okay, nos pedirá una passphrase para proteger esta key.

```
Please enter the passphrase to
protect your new key

Passphrase: *****█

      <OK>                                <Cancel>
```

Nos dice que debe ser mayor de 8 caracteres para que sea segura.

```
Warning: You have entered an insecure passphrase.

A passphrase should be at least 8 characters long.
A passphrase should contain at least 1 digit or
special character.

<Take this one anyway>                                <Enter new passphrase>
```

Volvemos a escribir la passphrase y terminamos el primer proceso.

```
└─# gpg2 --gen-key
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: directory '/root/.gnupg' created
gpg: keybox '/root/.gnupg/pubring.kbx' created
Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Julius Hacker
Email address: julius.hacker88@gmail.com
You selected this USER-ID:
    "Julius Hacker <julius.hacker88@gmail.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? 0
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: directory '/root/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs.d/29BF118F7DCC9B4420DFE6AFB11C5DDB9E3D3190.rev'
public and secret key created and signed.

pub   rsa3072 2024-06-05 [SC] [expires: 2026-06-05]
       29BF118F7DCC9B4420DFE6AFB11C5DDB9E3D3190
uid     [ultimate] Julius Hacker <julius.hacker88@gmail.com>
sub   rsa3072 2024-06-05 [E] [expires: 2026-06-05]
```

Con el comando `gpg2 --list-key` podemos ver el listado de key guardadas en el sistema.

```
(root@kali)-[/home/kali]
└─# gpg2 --list-key
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid: 1  signed: 0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2026-06-05
/root/.gnupg/pubring.kbx

pub   rsa3072 2024-06-05 [SC] [expires: 2026-06-05]
       29BF118F7DCC9B4420DFE6AFB11C5DDB9E3D3190
uid     [ultimate] Julius Hacker <julius.hacker88@gmail.com>
sub   rsa3072 2024-06-05 [E] [expires: 2026-06-05]
```

El siguiente paso que debemos realizar es exportar a un fichero la clave pública y privada. Para ello vamos a usar el comando `gpg2 --export -a {clave pública} > public-key.asc`

```
(root@kali)-[/home/kali]
└─# gpg2 --export -a
```

+ tab y nos autocompleta con la pub.

```
(root@kali)-[/home/kali]
# gpg2 --export -a 29BF118F7DCC9B4420DFE6AFB11C5DDB9E3D3190 > public_key.asc
```

Ahora haremos lo mismo con la clave privada. Para ello usamos el comando:

Gpg2 --export-secret-key -a {clave publica} > private_key.asc

```
(root@kali)-[/home/kali]
# gpg2 --export-secret-keys -a 29BF118F7DCC9B4420DFE6AFB11C5DDB9E3D3190 > private_key.asc
```

Ahora nos pedirá la passphrase que elegimos anteriormente como protección.



Verificamos que están creados los dos ficheros en la carpeta que estábamos.



En la siguiente imagen podemos ver el contenido de nuestra clave pública creada.

└─# cat public_key.asc

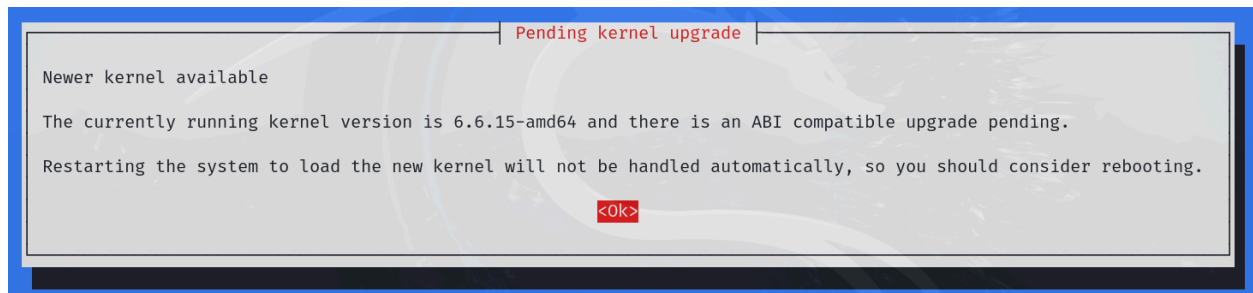
-----BEGIN PGP PUBLIC KEY BLOCK-----

```
mQGNBGZgk5cBDACcgC2cC907UERTYezjd1nTKYBrRXtkaDnp5dAvDJV3NrRINiyJ
36xHHb3UCXcQ65hLS46kDASJrVRC5CV2A2mV53bg3uXs4iYgkmsljC60wyheQlhA
5Rz9n79VaBuj5u0bU6x+y4mXR9ptFQ7oso8jTCdyw/eHaIP20P0Kw0bhXNHnj612
ndlG/gGCnW6ZHWBfo4A6tjQhbW4H/NTaJcLAeQk77quUoxyLYn1xZeAY+s4HI17h
WUWK7mgWBnEreCOKq8wqQ+zQJBwC10ous16cvGhNzg6E/hiPMevUitah+X4fDLol
aeX00IGZl5tNh//8DB5mZ+K8zAX524m+26RCxkilNABVkwDdZf9Q4B7sewcfAaEy
4uKqL8QzQ2jtxms52LOW/OgIwayFEjy+ezkIClsd7mLZWg4vDQ50w0vrefmjsQhl
ZQFlEaj9uybNPMh5eIHh3Iz80S4rmjPLPEw5q3c3+DdCtCmyes9n/Wfsv1MpfytB
rMakbCLKaUgqi0sAEQEAAbQpSnVsaXVzIEhhY2ticiA8anVsaXVzLmhhY2tlicjg4
QGdtYWlsLmNvbT6JAdQEEwEKAD4WIQQpvxGPfcybRCDf5q+xHF3bnj0xkAUCZmCT
lwIbAwUJA8JnAAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRCxHF3bnj0xkAJ3X
C/9sXCPaWRhNUXvgbVqLybo7Spk8kEuJK7NfZzr0mbXjdZHvaXfPrG6qp5kwM0Om
8eCN8fxr/S7eNecnrX9gXBwnanSv2q6MVv1fj7ZC1m/MCMSsXUZpAdjwor/9BQH
wrpCYg1V8TGBzd0uu+3PB8N2l8+i2wn2GRD04R0qWQqLvLZFDbaUA7DPB9B/2rGLl
fb4RCrr3f812UsU/sgF+buDDsN3K0kzjz0ct2Pv6meuYRPbql50H9F5Ta74B7rkM
mwED2K9YE38n2mU9EtpmIQubYyJibxt8A20I4827atHAVPZKlMISpXbg4nA1BB1J
EbNuewrjy1vVgwbHTP58Dx2D2uCAKGom31/vQ9gcMxHT9X7HzCFr7Hd/mNBSEY0e
N9t/nkJ75VtOIRnVy2uDmyrQ4Ez++rCD8A+nAEBRwXl4ph4TiSVwHNqJt3bKhcnX
pcE24Vk1ujtG4ntk4J7lehTR5opTk+h7abyuranDZP8opC2JR8yeHaRHtJzqzJ79
cdq5AY0EZmCTlwEMAMT3kHko1hTVSrayxbblZyefCItmT0IzkPaqZWMgdoW0kOLv
c22VIU5351K1QLfffcNehk4WjGT653Bhck9T8zVj8kGQ7X1dBQeJAnuqH4JC5SSG9
lPnW43dj4E3HPxVNX217Zz0sSolP9DIqU3VnaqijqhEtGNg0V/PdMeZ10bFawhdL
5+B8+01xrrWXjpYpzcCFMPCL7sR3La87grNzbCnQr3S/QqGWye1l9hMgLnRnhmPN
UC/7pLcJ6/LD5Kj2eysNavJphl0q2n0QCoCYyk3bRlnZxMlLP7v4borYk6VEk9Bt
UM0tIy4AoypXJF03YmUSYQz4H7xBLleRR1Ynx5fEyDDe1Ma9NGWOH/jztd1nG6S0
jlPpCv8wFm3RZhOK1VpYHEKqld6S/41ZF10jzf/Ez+gUC5dAvEviumi8LWOpL01n
XOcb/IyHBdEUzMoHi9GPJ0SX1JSG3bHWj4QZ6ltMr/M/xUDXOFwjQE45wANQKpqy
lhoY4HMFUlnskmrjDQARAQABiQG8BBgBCgAmFiEEKb8Rj33Mm0Qg3+avsRxd2549
MZAFAmZgk5cCGwwFCQPCZwAACgkQsRxd2549MZDfNwv/UFxdJBj9mAf8Ge67KD4L
jhoHtUbhMD7y9FqvMMlRVBqnk1F8ReAWcQZhtjuUxrLCX675X5yZZIYaGippTGsb
TUhd9Z5Fyqt0fro2JLzdj0Lcp1FqggylBBRaScbJ9DvGNBjbnuC6lhmX9jYJk4+2
IiMn3Y5BZ4aSNCL+tLC8g7LMEaTN7qpbK71rtpqqbeNd6IEH/5xLcn0BMazfbtFH
00t/B0YCZxo4iP/G087duvVL25jQsflXUK61cckSIIDuM1EqVC10PAB2JMsYJNH
tWesX6+6M/zqjgaYSP50kEiaPIbxwifCtSE0dcIqXscuGlQTnf8wqfw5ZdwwvLx3
pvs5KMILf4xdA+wwTZNcS+rWHocoEuXkA/ftK76bR4wAcy7nSZDfg5WjK0sqHvNb
4qbXHCAIY54953yRC2mMdhKPPqRNLffmgONePM8JalkQvjDNeQ0KZP2dKdPN8HFA
faicjHIwIV6iWDhg9PNmt5JKOpouts6w++Pw7RGYIfKn
=J5P/
```

GPA The Gnu Privacy Assistant

El siguiente paso que realizaremos en este ejercicio será la instalación de una herramienta con interfaz gráfica para el gnuPG, GNU Privacy Guard. Para ello usaremos el comando: `apt-get install gpa` .

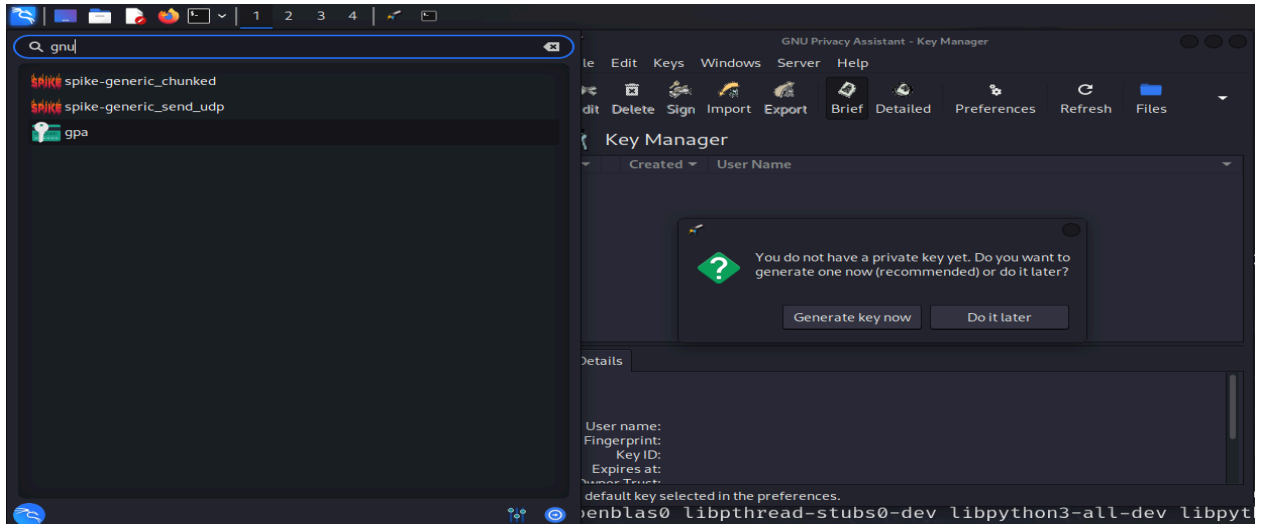
Al empezar la instalación nos aparece la siguiente ventana:



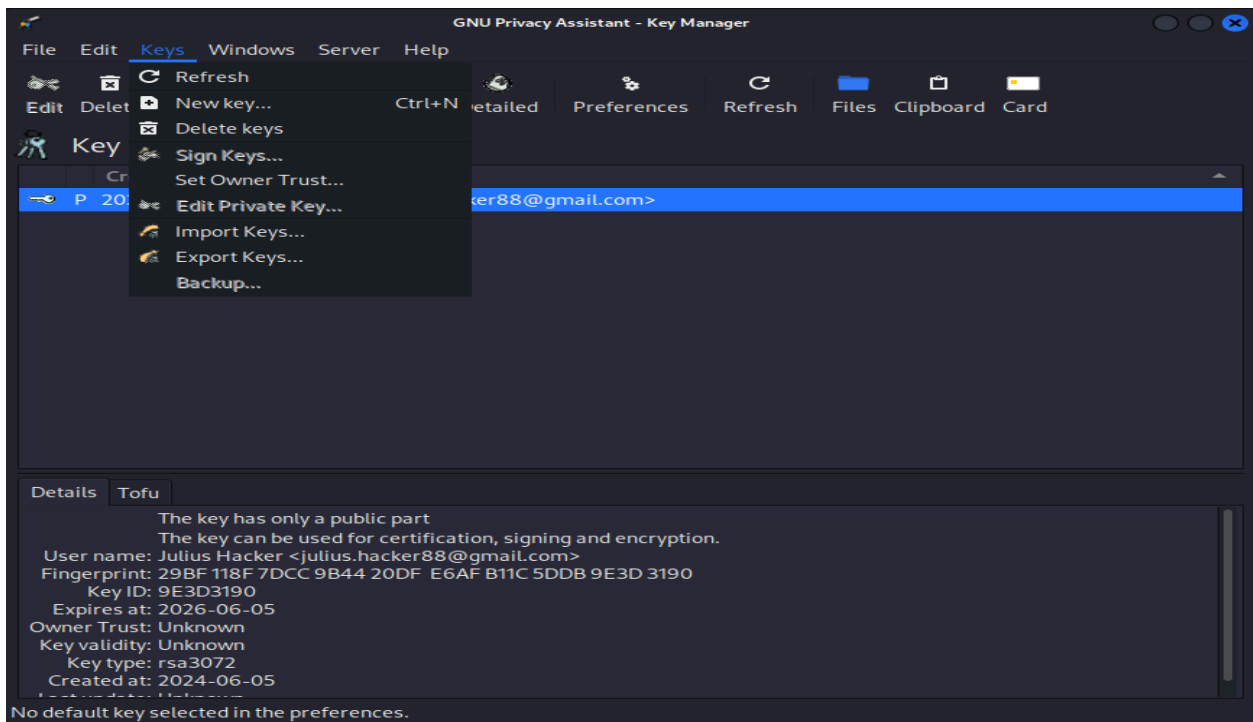
Le damos ok. Y verificamos si se instaló correctamente ejecutando nuevamente el mismo comando.

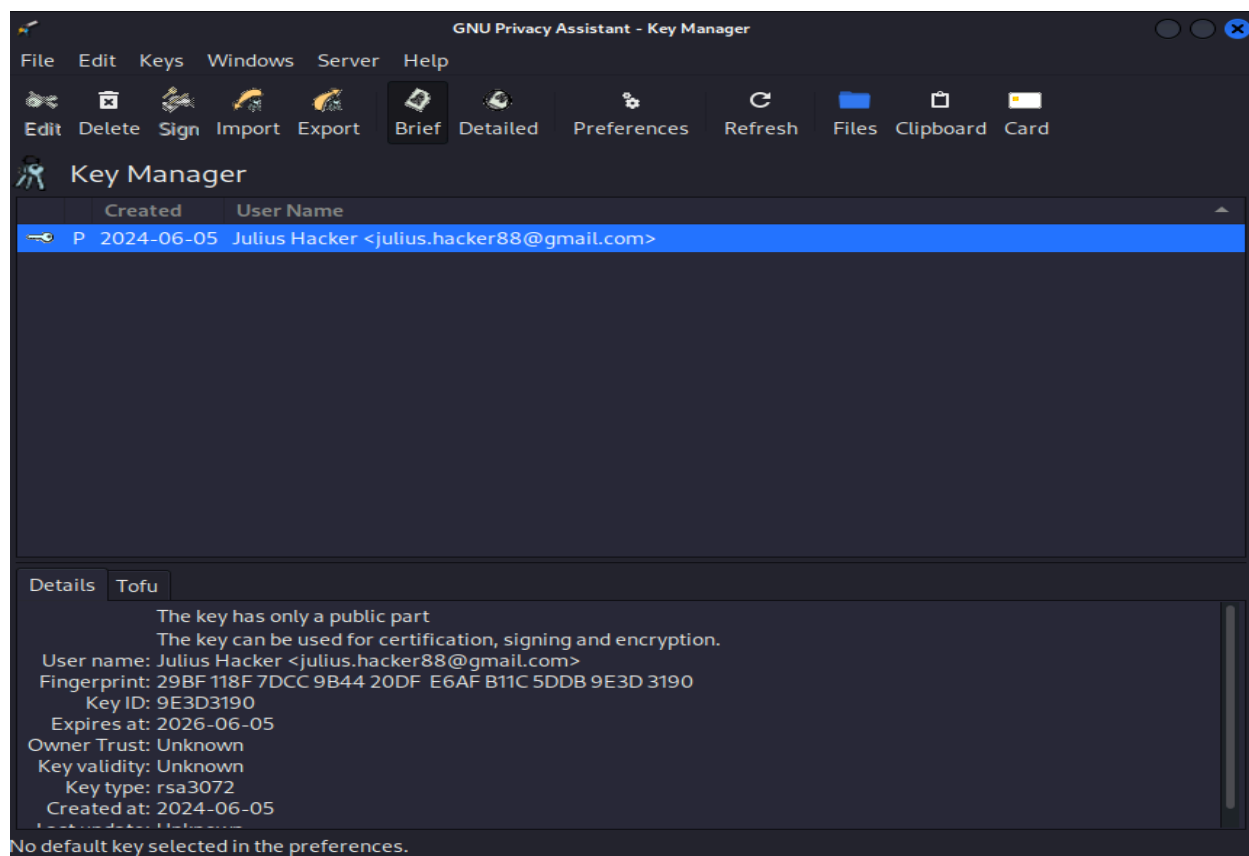
```
(root@kali)~[/home/kali]
# apt-get install gpa
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gpa is already the newest version (0.10.0-6).
The following packages were automatically installed and are no longer required:
 fonts-noto-color-emoji libabsl20220623 libadwaita-1-0 libaio1 libappstream5 libatk-adaptor libboost-dev libboost1.83-dev libgphoto2-l10n libnsl-dev
 libopenblas-dev libopenblas-pthread-dev libopenblas0 libpthread-stubs0-dev libpython3-all-dev libpython3.12 libpython3.12-dev libstemmer0d libtirpc-dev
 libxmlb2 libxsimd-dev python3-all-dev python3-anyjson python3-beniget python3-editables python3-gast python3-pyatspi python3-pypdf2 python3-pyppeteer
 python3-pyrsistent python3-pythran python3.12-dev xtl-dev zenity zenity-common
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 762 not upgraded.
```

Ya tenemos instalada la herramienta y ahora vamos a ejecutarla desde la lista de herramientas de nuestro Kali Linux.



Luego importamos la Key creada. Para ello vamos a Keys > Import Keys





El siguiente paso que vamos a realizar será descargar el fichero con la clave pública PGP que nos envió el profesor.

Una vez descargado el fichero que se llama Profe.asc , lo abrimos para verificar su contenido.

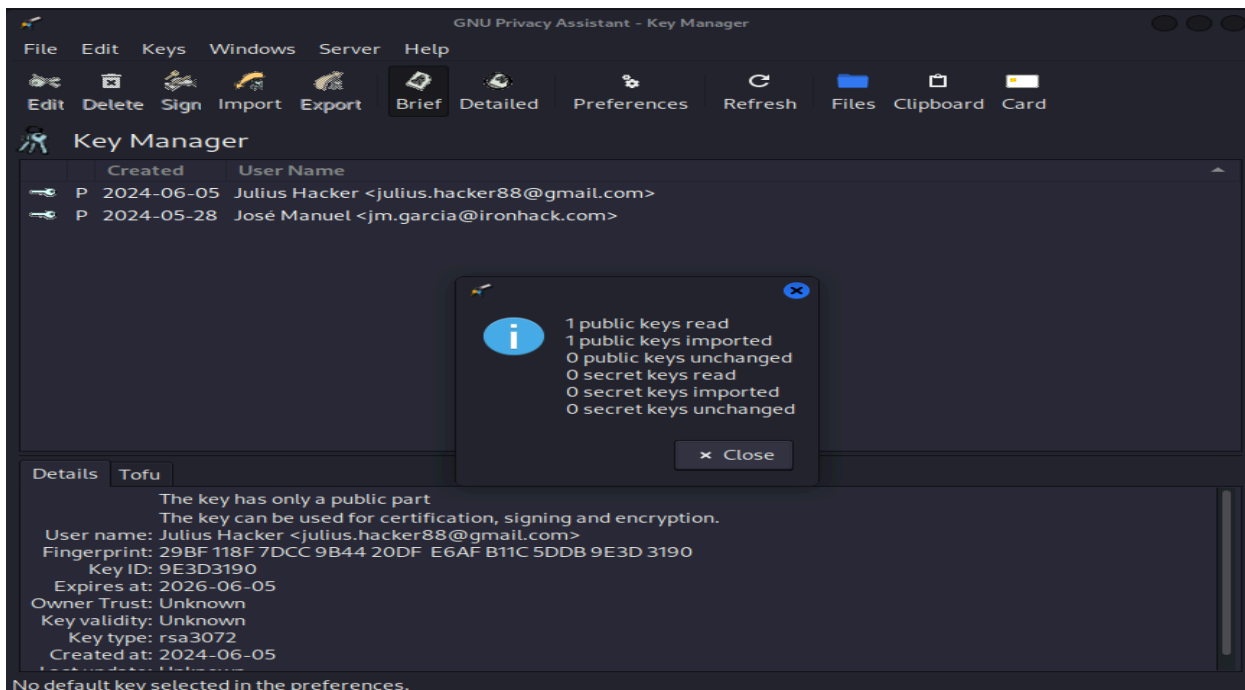
```

(root@kali)-[/home/kali]
# cat Profe.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----

mDMEZLWWORYJKwYBBAHaRw8BAQdAcurDz/ogkPm0RQ/hhpJ3J7JCU0/tQ7GlyIPL
GM/pfqK0JUUpvc80pIE1hbnVlbCA8am0uZ2FyY2lhQGlyb25oYWNrLmNvbT6ImQQT
FgoAQRyHBKu+T8i4eCVZgDagc+A4L5eC6URzBQJmVZY5AhsDBQkFo69nBQsJCAcC
AiICBhUKCQgLAGQWAgMBAh4HAheAAoJEOA4L5eC6URzJdAA/Ai2FwhZpTr292mL
Ttjj5GK9uvGsYFP0mhLpCQy0UNzUAP45rl0e5KdqhrbDUZtniUxrZegQCSNjzF2e
8g4hJ6tPArg4BGZVljKSCisGAQQBl1UBBQEBB0AgBdX0xkUvTrP8X8rFfoG7ITve
9xlKp90UNHQxlthAfgMBCAeIfgQYFgoAJhYhBKu+T8i4eCVZgDagc+A4L5eC6URz
BQJmVZY5AhsMBQkFo69nAAoJEOA4L5eC6URz4yQBAJnI30Z1EIYjUojPqj2LEa/H
PNvJx7bpbUk1Kqvv7k/GAQDre4L4mVE8EE3P9a+aEpbFPjUYk2iQdmdL4PVWVNpz
CQ=
=1jRj
-----END PGP PUBLIC KEY BLOCK-----

```

Ahora haremos el mismo proceso para importarlo en gnuPA.



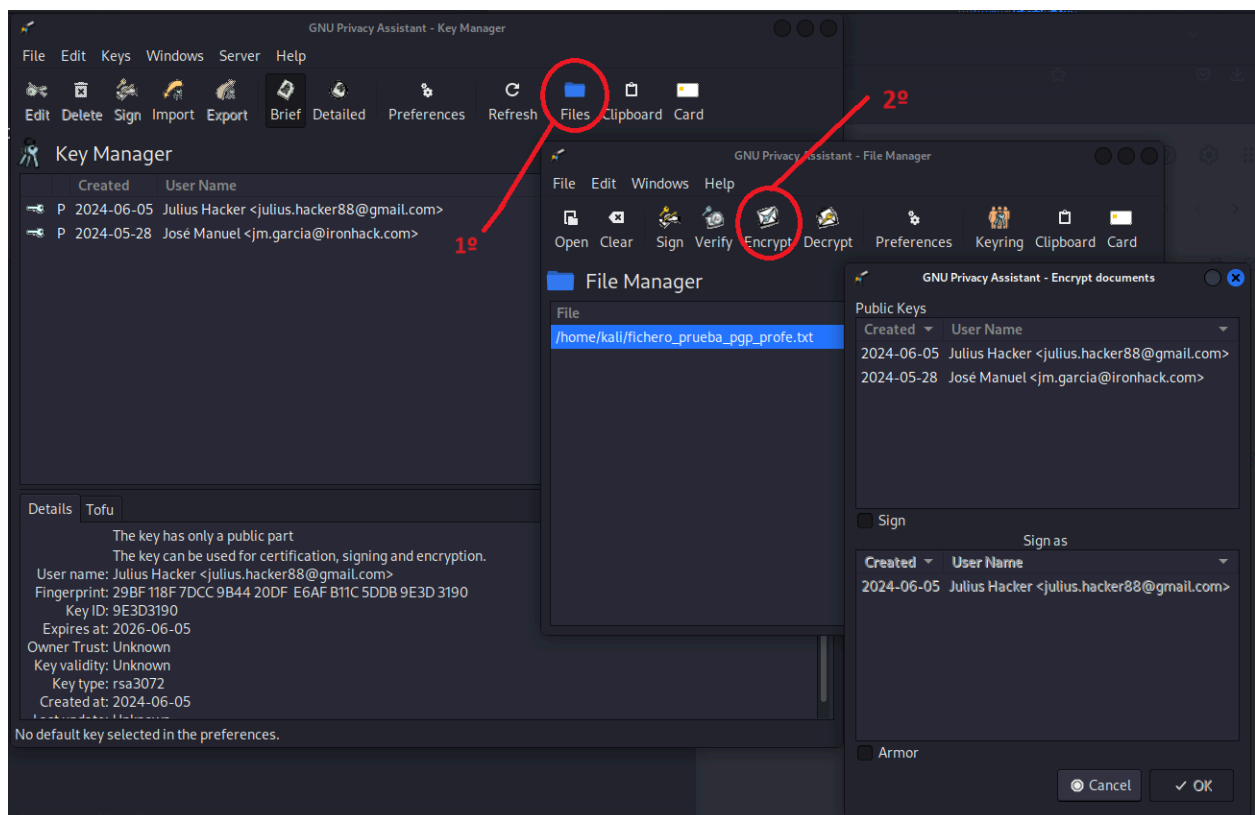
Creación de fichero y cifrado

El siguiente paso en este ejercicio será crear un fichero con el contenido “Solución P3”.

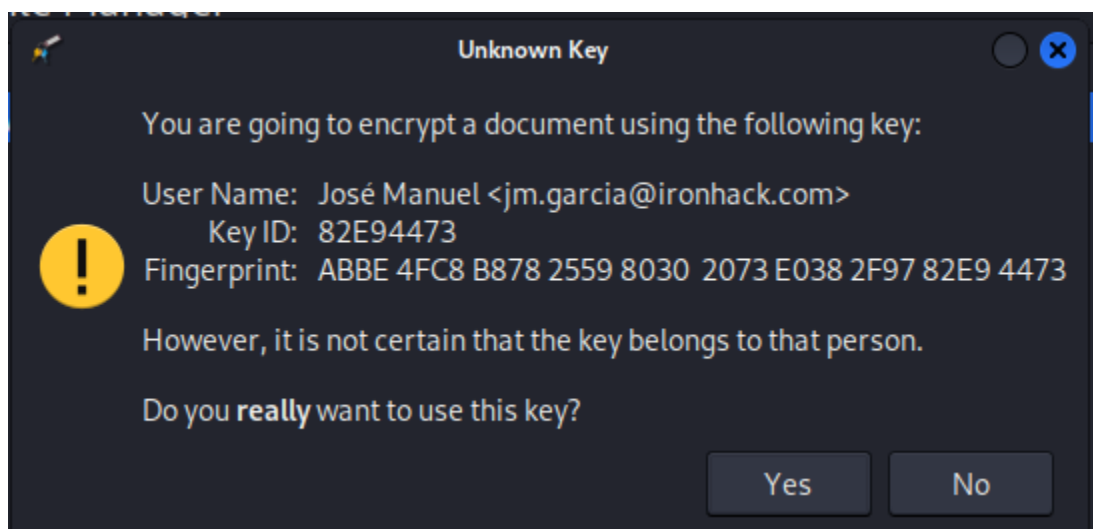
```
(root@kali)-[/home/kali]
# echo "Solución P3" > fichero_prueba_pgp_profe.txt

(root@kali)-[/home/kali]
# cat fichero_prueba_pgp_profe.txt
Solución P3
```

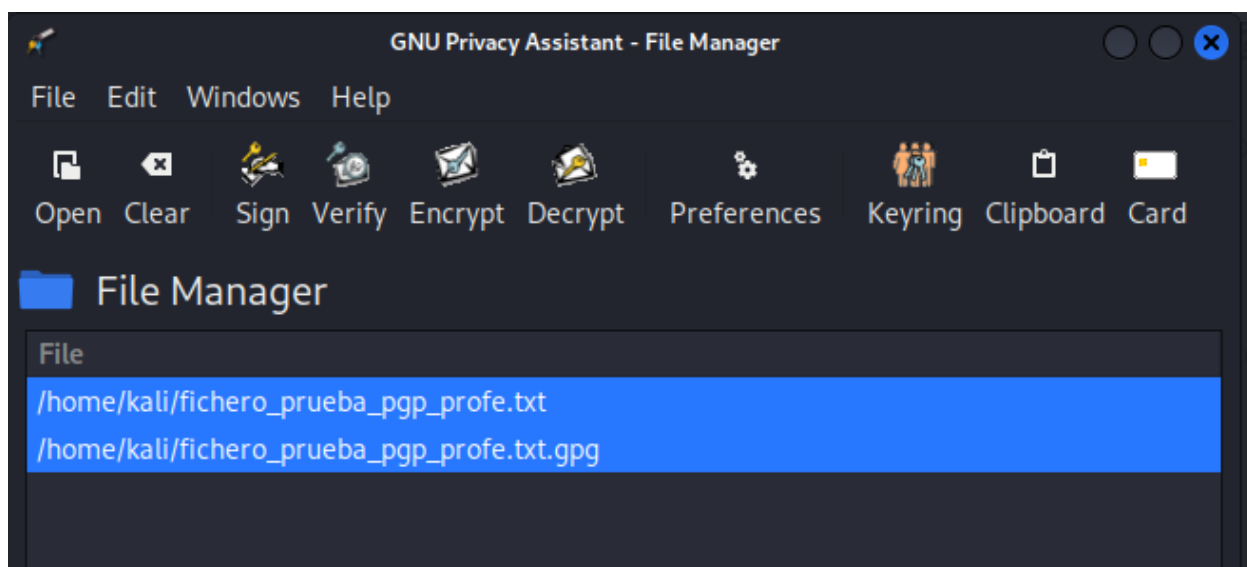
Ahora procederemos a encriptar el fichero con la clave que nos pasó el profesor. Para ello volvemos a gnuPA e importamos el fichero ‘fichero_prueba_pgp_profe.txt’ .



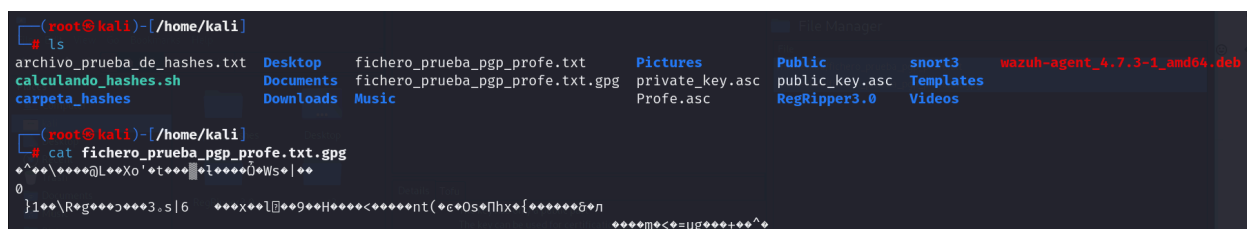
Seleccionamos la clave que nos pasó el profesor y nos aparecerá el siguiente mensaje.



Podemos ver que se generó un fichero con el mismo nombre pero .gpg.



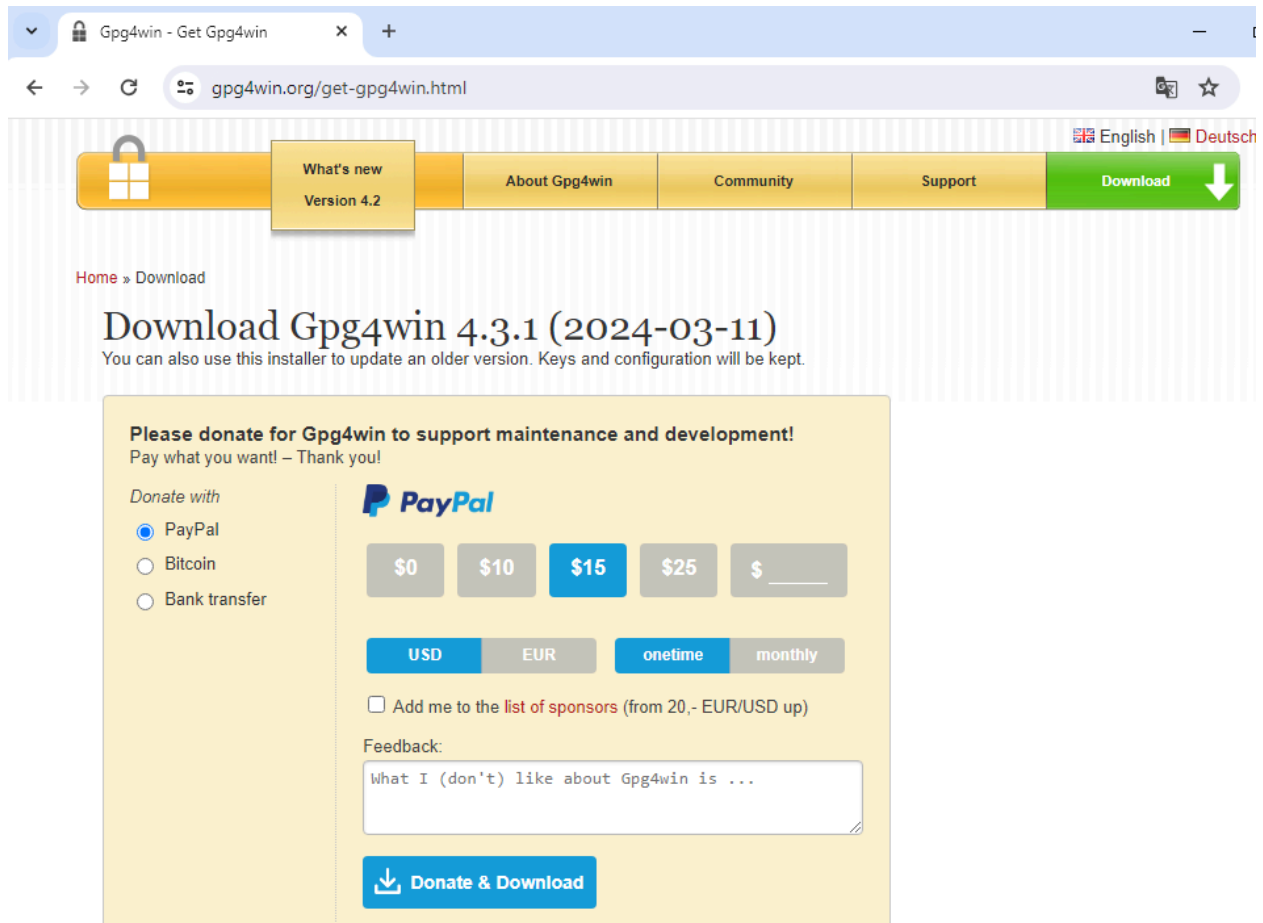
Ahora intentaremos abrir el fichero para ver si podemos ver el contenido, ya que era un fichero en texto plano. Podemos confirmar que está encriptado y no podemos leerlo.



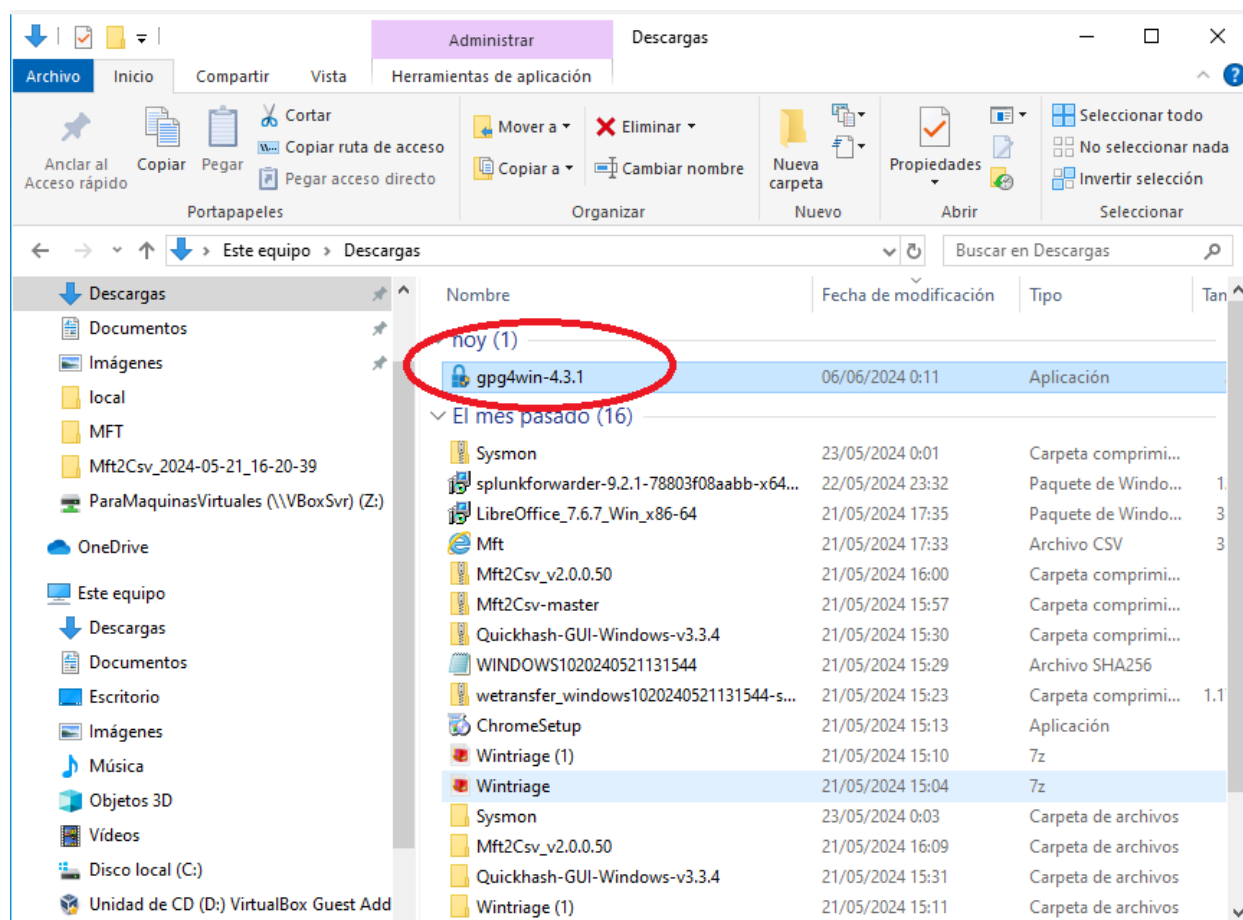
Windows

Instalación y uso de Kleopatra

Ahora realizaremos el mismo proceso que anteriormente, pero en nuestra máquina Windows 10. Empezaremos por descargarnos el programa GpG4Win de su página web oficial <https://www.gpg4win.org/get-gpg4win.html> .



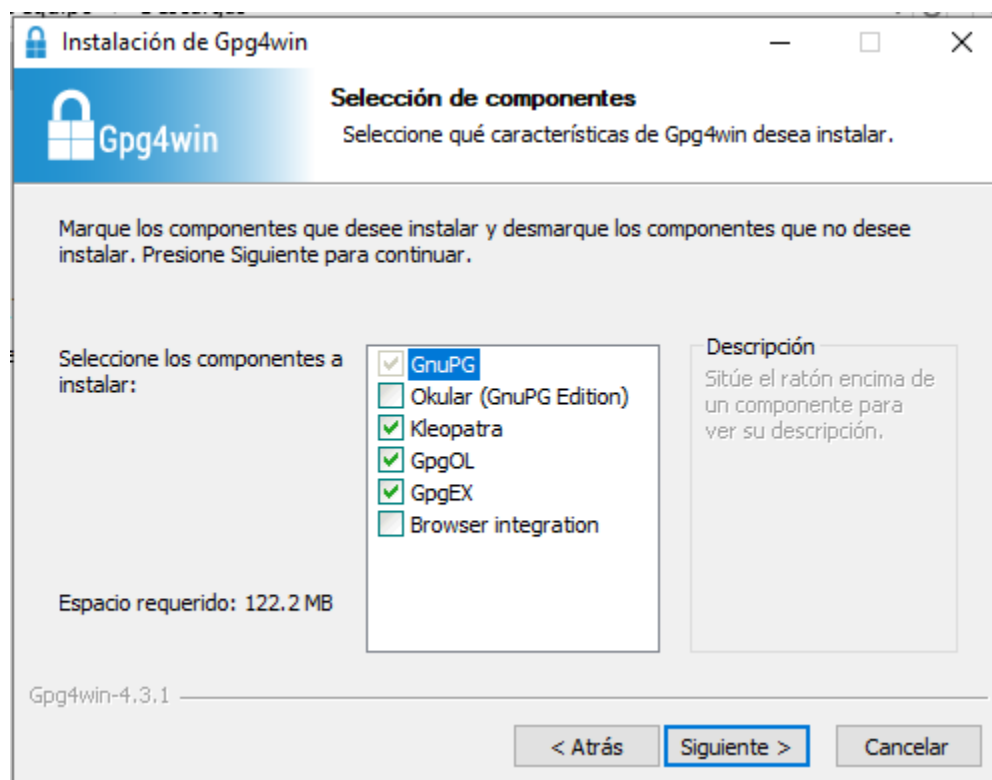
Una vez descargado el fichero lo ejecutamos.



Se nos abrirá el instalador.



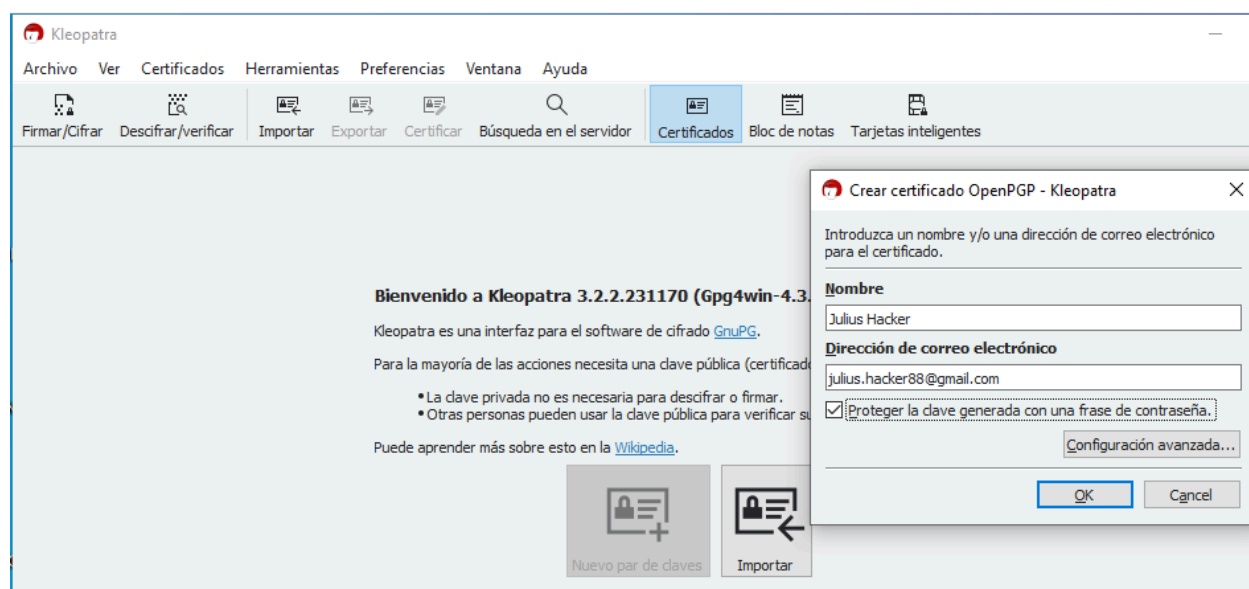
Dejamos por defecto los componentes que vienen seleccionados.



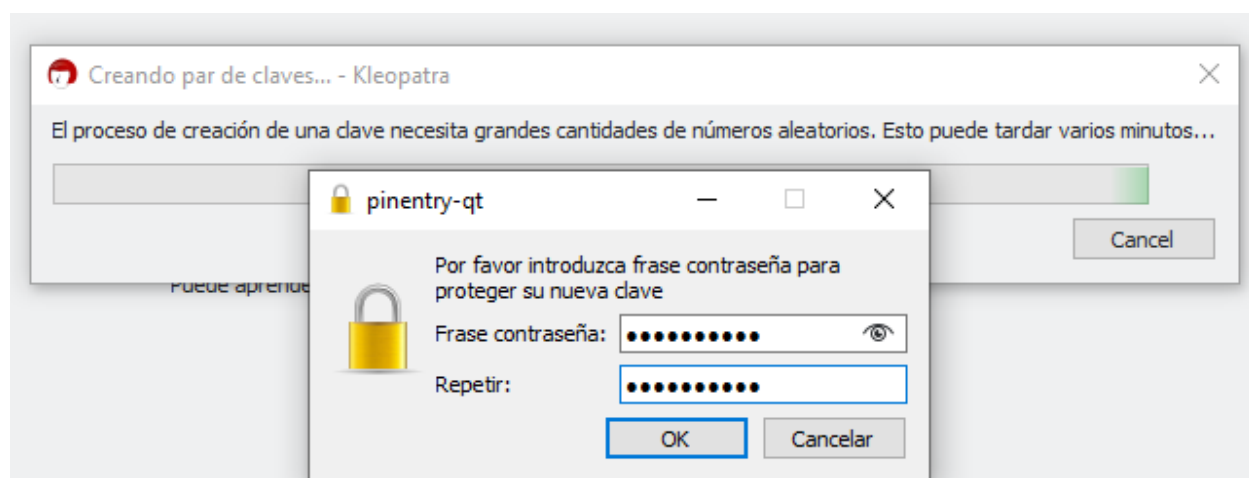
Finalizamos la instalación.



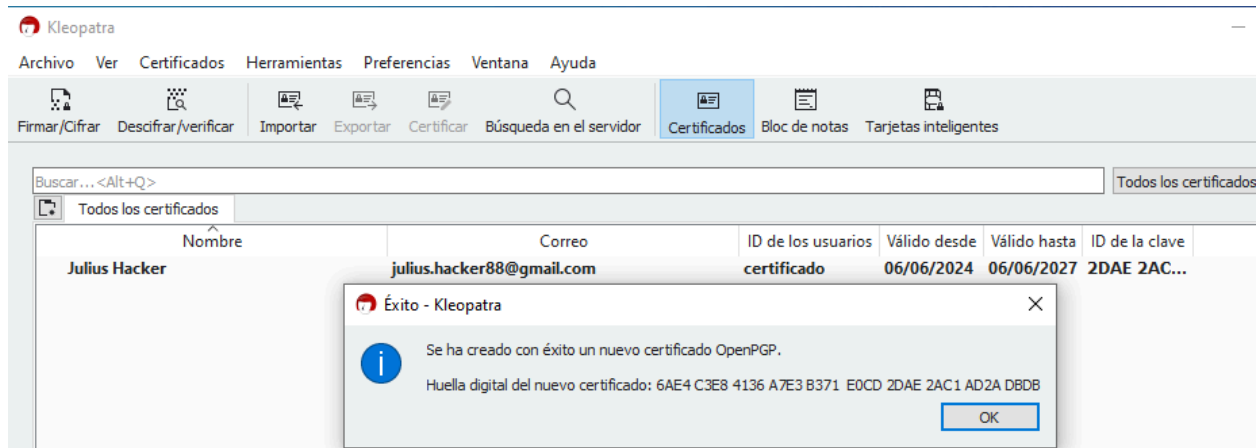
El siguiente paso será abrir Kleopatra para generar un par de claves PGP (clave pública y privada).



Completamos con nombre y correo electrónico y tildamos la casilla de proteger la Clave generada con una frase de contraseña. Al igual que anteriormente en el proceso que hicimos con Linux, nos pedirá una frase de contraseña.

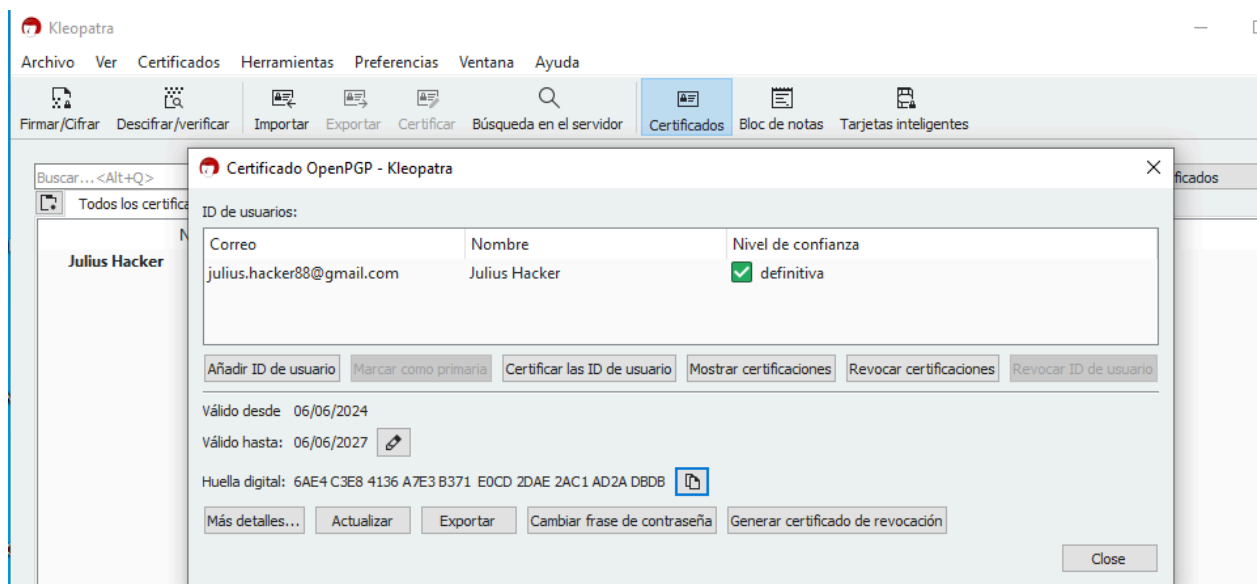


Se ha creado con éxito y nos da la huella digital(hash) del nuevo certificado.

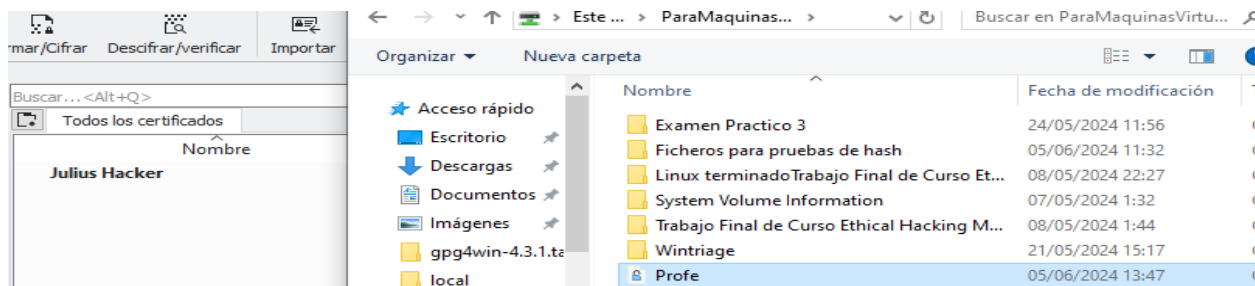


Es la siguiente:

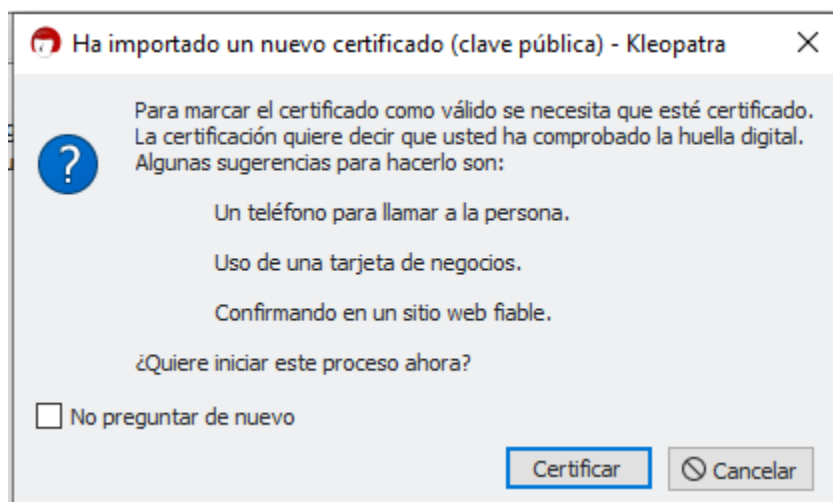
6AE4C3E84136A7E3B371E0CD2DAE2AC1AD2ADBDB



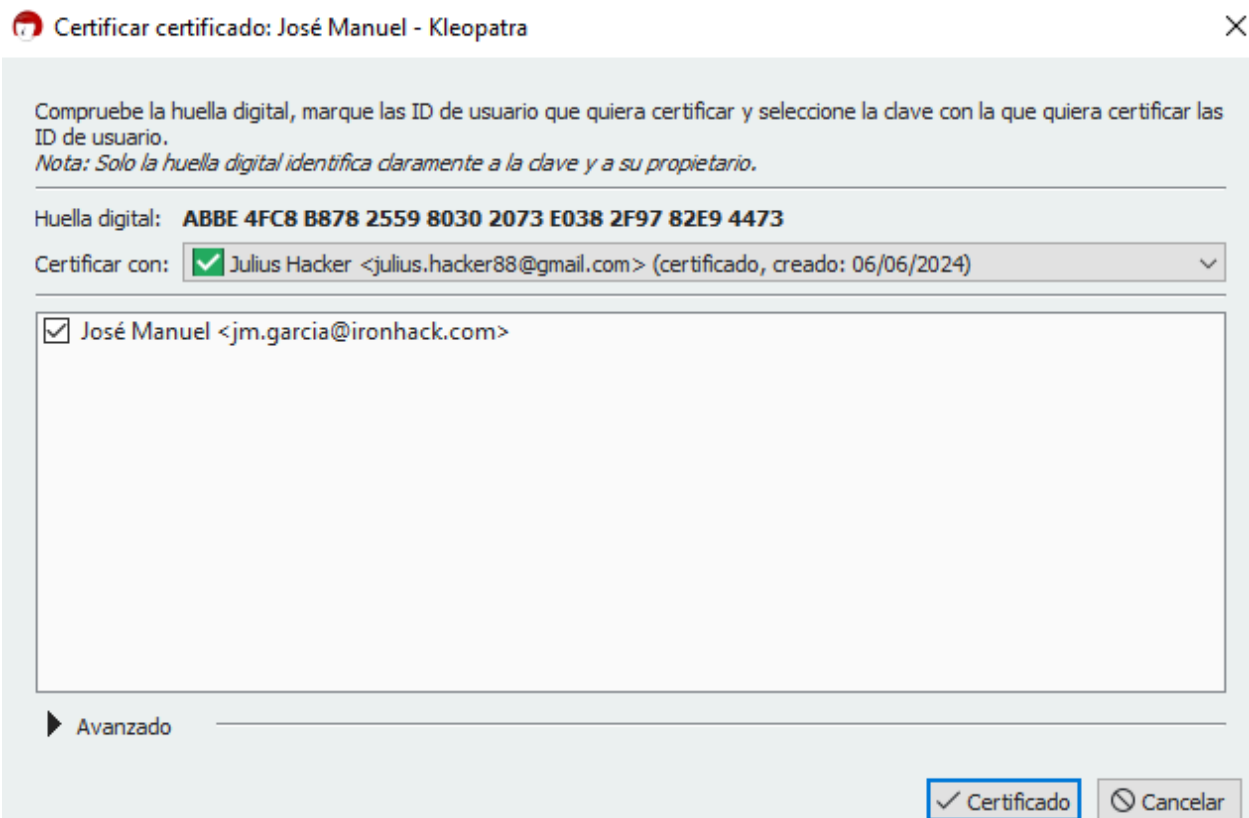
Ahora importamos la clave que nos dió el profesor, al igual que hicimos con Linux.



Se nos abre la siguiente ventana.



Le decimos que si, queremos Certificar.

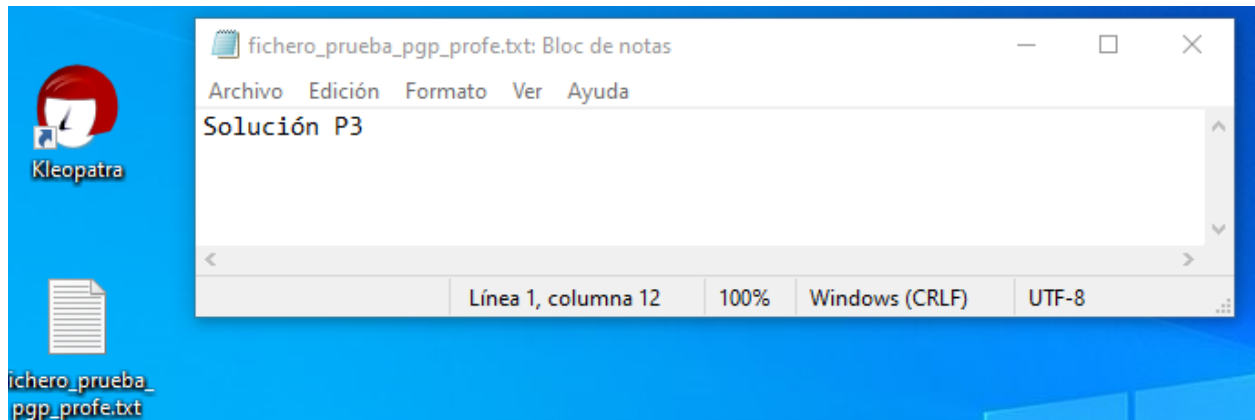


Ahora ya tenemos los 2 certificados creados.

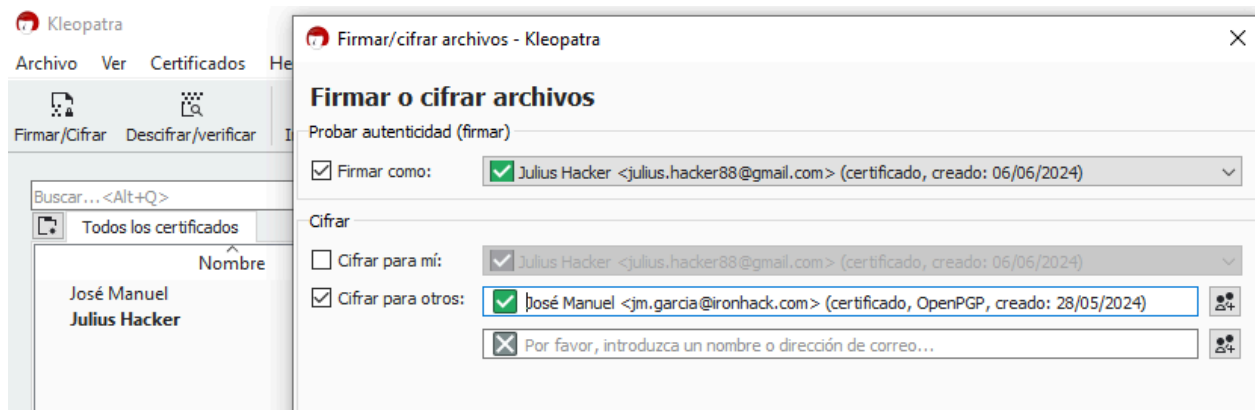


Creación de fichero y cifrado

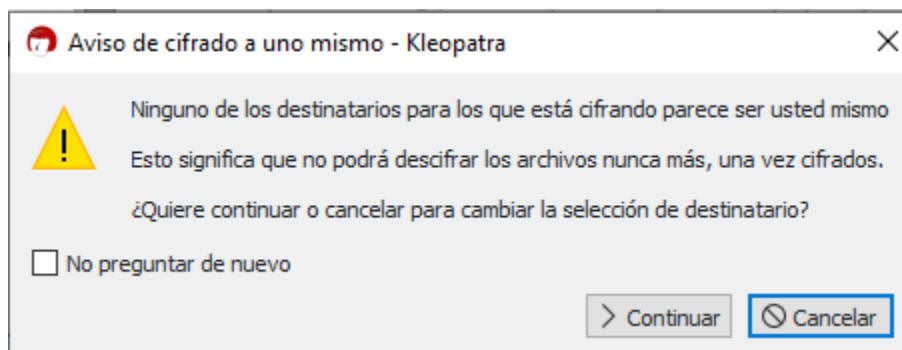
Ahora siguiendo el proceso anterior, crearemos un fichero txt con el contenido "Solución P3".



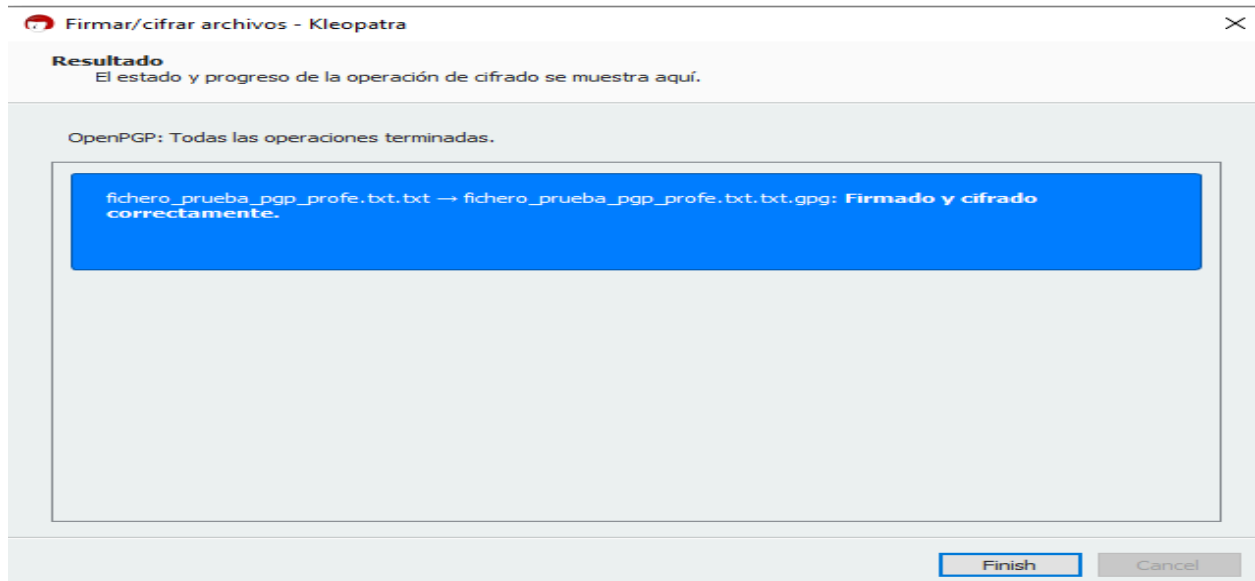
Una vez creado el fichero, volvemos a Kleopatra para cifrarlo. Marcamos la opción Cifrar para Otros y seleccionamos el certificado del profesor.



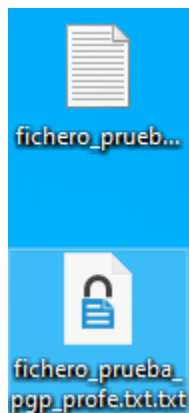
Nos saldrá la siguiente ventana y le damos a continuar.



Se firmó y cifró correctamente el fichero.



Verificamos que se creó un fichero binario, de tipo "OpenPGP binary file", cifrado y sólo puede ser abierto con la clave privada correspondiente al certificado del profesor.



Creación de un Certificado X.509 Autofirmado

Conceptos teóricos

A continuación explicaremos un poco los conceptos de los que trataremos posteriormente en la parte práctica de este ejercicio.

Un certificado X.509 es un estándar ampliamente utilizado para la infraestructura de clave pública (PKI). Proporciona un formato estandarizado para los certificados digitales y las listas de revocación de certificados. Los certificados X.509 son utilizados para verificar la identidad de las entidades en una red (como servidores, usuarios o dispositivos) y para cifrar la comunicación entre ellas, garantizando así la seguridad y la privacidad de los datos transmitidos.

Un certificado X.509 contiene varias piezas de información crítica, incluyendo:

- El sujeto del certificado: Identifica a la entidad que posee el certificado.
- La clave pública del sujeto: Utilizada para verificar firmas digitales y cifrar datos.
- La autoridad certificadora (CA): La entidad que emite el certificado.
- El período de validez: Define las fechas de inicio y expiración del certificado.
- El uso del certificado: Define para qué puede ser utilizado el certificado (por ejemplo, autenticación de cliente, autenticación de servidor, etc.).

Certificado AutoFirmado

Un certificado autofirmado es un certificado X.509 firmado por la misma entidad que lo emite. Esto significa que la entidad emite y firma su propio certificado en lugar de depender de una autoridad certificadora (CA) externa. Los certificados autofirmados se utilizan comúnmente para pruebas internas, desarrollo o para establecer una CA privada dentro de una organización.

Las principales características de un certificado autofirmado son:

- Autenticación y confianza limitada: Ya que no ha sido verificado por una CA externa, su autenticidad sólo puede ser verificada por entidades que confían explícitamente en la entidad que lo emite.
- Uso en entornos controlados: Adecuado para pruebas y desarrollo, o en situaciones donde se puede gestionar manualmente la confianza (por ejemplo, dentro de una red privada).

Autoridad Certificadora (CA)

Una autoridad certificadora (CA) es una entidad responsable de emitir y gestionar certificados digitales. La CA verifica la identidad de las entidades que solicitan certificados y luego firma los certificados para garantizar su autenticidad. En una infraestructura de clave pública (PKI), las CAs desempeñan un papel importante al actuar como una tercera parte de confianza que garantiza la integridad y validez de los certificados emitidos.

Funciones principales de una CA:

- Verificación de identidad: Confirmar la identidad de las entidades antes de emitir un certificado.
- Emisión de certificados: Generar y firmar certificados digitales que vinculan una clave pública con una identidad específica.
- Gestión de certificados: Mantener y publicar listas de certificados revocados (CRLs) y responder a las consultas sobre la validez de los certificados.

OpenSSL

OpenSSL es una biblioteca de software robusta y ampliamente utilizada que proporciona funciones criptográficas y herramientas para la implementación de protocolos de seguridad. Es una herramienta fundamental en la creación y gestión de certificados X.509 y la configuración de la seguridad de las comunicaciones en redes.

Características y usos de OpenSSL:

- Generación de claves y certificados: Permite crear pares de claves y certificados digitales.
- Firmado de certificados: Facilita la creación de certificados firmados por una CA.
- Cifrado y descifrado: Proporciona herramientas para cifrar y descifrar datos utilizando diversos algoritmos criptográficos.
- Soporte de protocolos SSL/TLS: Implementa los protocolos Secure Sockets Layer (SSL) y Transport Layer Security (TLS) para asegurar las comunicaciones en red.

Certificado X.509 Autofirmado con OpenSSL

El proceso para crear un certificado X.509 autofirmado generalmente incluye los siguientes pasos:

1. Generación de una clave privada: La clave privada es utilizada para firmar el certificado y debe mantenerse en secreto.
2. Creación de una solicitud de firma de certificado (CSR): El CSR es un archivo que contiene información sobre la entidad que solicita el certificado, incluyendo su clave pública. Este CSR será firmado por la CA para emitir el certificado.
3. Firmar el CSR con la clave privada para crear un certificado autofirmado: Este paso genera el certificado X.509 que será utilizado para identificar y asegurar la entidad.

En un entorno real, este proceso puede ser automatizado y manejado por scripts o herramientas especializadas, facilitando la gestión de certificados en grandes infraestructuras.

Instalación de OpenSSL

Windows


Empezaremos la parte práctica de esta parte del ejercicio descargando el programa OpenSSL del sitio web: <https://slproweb.com/products/Win32OpenSSL.html>. Descargamos el fichero ejecutable .exe.

Win32/Win64 OpenSSL Installer

slproweb.com/products/Win32OpenSSL.html

directly? Please send me the bank details with a quote." My bank account is 000012345 and my routing number is 000012345. I look forward to your donation. Huh...that's amazing! I've got the same combination on my luggage!

Win32/Win64 OpenSSL Screenshot



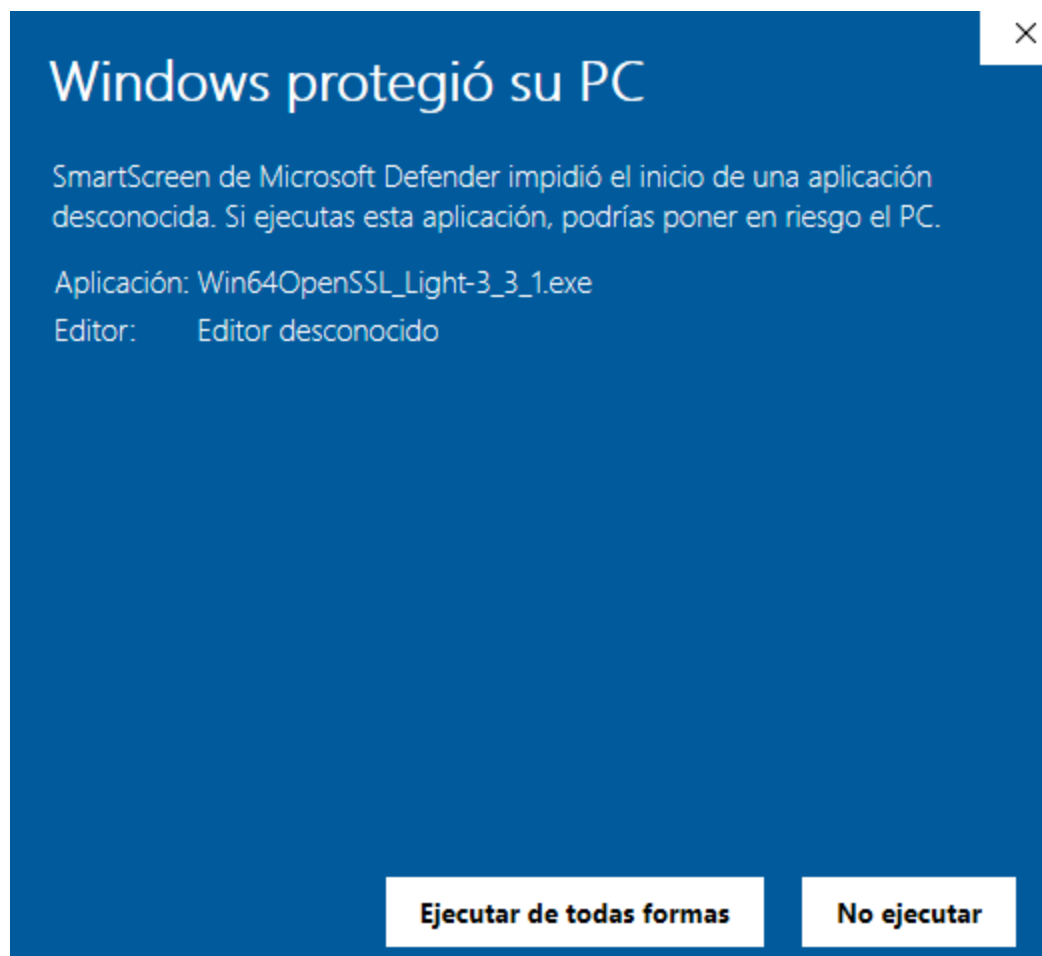
[Screenshot of the Win32/Win64 OpenSSL Installer.](#)

Download Win32/Win64 OpenSSL

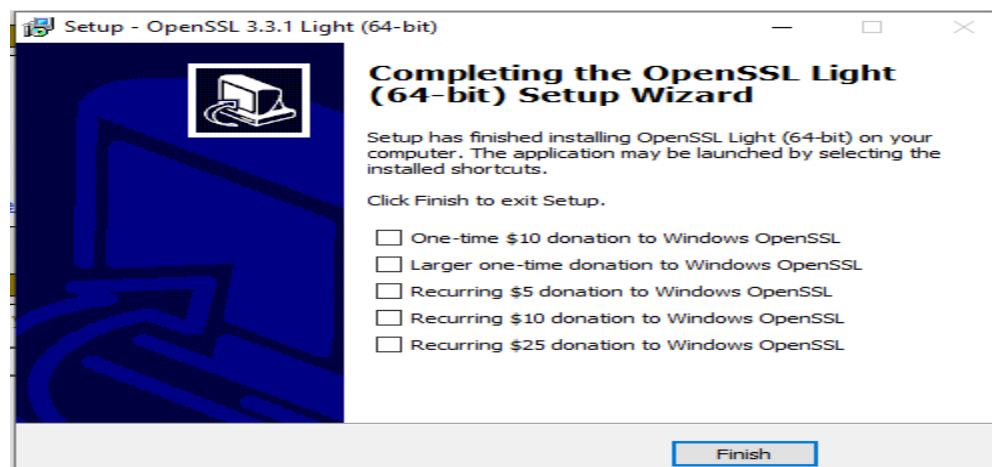
Download Win32/Win64 OpenSSL today using the links below!

File	Type	Description
Win64 OpenSSL v3.3.1 Light EXE MSI	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.3.1 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

Windows Defender lo detecta como malicioso y nos pregunta si queremos ejecutarlo igualmente, le diremos que queremos ejecutar de todas formas, porque el profesor nos enseñó que debemos ser valientes!



Finaliza la instalación dejando todo por default.



Verificamos que se nos creó esta carpeta en archivos de programa.

Nombre	Fecha de modificación	Tipo
bin	06/06/2024 0:56	Carpeta de archivos
acknowledgements	04/06/2024 12:26	Documento de te...
authors	04/06/2024 12:26	Documento de te...
c_rehash.pl	04/06/2024 12:26	Archivo PL
changes	04/06/2024 12:26	Documento de te...
libcrypto-3-x64.dll	04/06/2024 12:26	Extensión de la ap...
libssl-3-x64.dll	04/06/2024 12:26	Extensión de la ap...
license	04/06/2024 12:26	Documento de te...
news	04/06/2024 12:26	Documento de te...
readme	04/06/2024 12:26	Documento de te...
start	04/06/2024 12:26	Archivo por lotes ...
unins000.dat	06/06/2024 0:56	Archivo DAT
unins000	06/06/2024 0:56	Aplicación

El siguiente paso será crear un certificado X.509 autofirmado. Para ello abriremos un terminal cmd como administrador y ejecutaremos el siguiente comando:

```
openssl genpkey -algorithm RSA -out private.key -pkeyopt rsa_keygen_bits:4096
```

[illegible]

Este comando utiliza OpenSSL para generar una clave privada RSA de 4096 bits y la guarda en un archivo llamado **private.key**.

Explicación del comando:

- **openssl genpkey**: Este es el comando de OpenSSL para generar una clave privada.
- **-algorithm RSA**: Especifica que se desea generar una clave RSA.
- **-out private.key**: Especifica el nombre del archivo de salida donde se guardará la clave privada.

- `-pkeyopt rsa_keygen_bits:4096`: Define el tamaño de la clave, en este caso, 4096 bits.

Ahora vamos a crear un fichero con datos. Para ello ejecutamos el siguiente comando:
`openssl req -new -key private.key -out request.csr`

```
C:\Program Files\OpenSSL-Win64\bin>openssl req -new -key private.key -out request.csr
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]:
```

Explicación del comando:

- `openssl req -new`: Indica a OpenSSL que se desea crear una nueva solicitud de certificado (CSR).
- `-key private.key`: Especifica la clave privada que se utilizará para generar la solicitud de certificado.
- `-out request.csr`: Define el nombre del archivo de salida donde se guardará la solicitud de certificado.

Este comando generará un archivo `request.csr` que contiene la solicitud de certificado. Debemos asegurarnos que `private.key` es la clave privada que generamos anteriormente. Nos pedirá información para completar y luego nos generará los ficheros en la carpeta que le indicamos.

```
C:\Program Files\OpenSSL-Win64\bin>openssl req -new -key private.key -out request.csr
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]:ES
State or Province Name (full name) [Some-State]:Catalunya
Locality Name (eg, city) []:Barcelona
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IronHack
Organizational Unit Name (eg, section) []:JM_THE_KING
Common Name (e.g. server FQDN or YOUR name) []:Jose Manuel
Email Address []:julius.hacker88@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:julian1234
An optional company name []:Ironhack
```

Ahora vamos a crear el certificado autofirmado, para ello ejecutamos el siguiente comando:

```
openssl x509 -req -in request.csr -signkey private.key -out certificate.crt -days 365
```

Explicación del comando:

- `openssl x509`: Indica a OpenSSL que se desea trabajar con certificados X.509.
- `-req`: Especifica que se utilizará una solicitud de certificado (CSR) para generar el certificado.
- `-in request.csr`: Define el archivo de solicitud de certificado de entrada.
- `-signkey private.key`: Especifica la clave privada que se utilizará para firmar el certificado.
- `-out certificate.crt`: Define el nombre del archivo de salida donde se guardará el certificado autofirmado.
- `-days 365`: Especifica la duración del certificado en días.

Este comando generará un archivo `certificate.crt` que contiene el certificado autofirmado, válido por 365 días.

```
C:\Program Files\OpenSSL-Win64\bin>openssl x509 -req -in request.csr -signkey private.key -out certificate.crt -days 365
Certificate request self-signature ok
subject=C=ES, ST=Catalunya, L=Barcelona, O=IronHack, OU=JM_THE_KING, CN=Jose Manuel, emailAddress=julius.hacker88@gmail.com
```

Ahora vamos a abrir el fichero generado que contiene el certificado autofirmado.


```

C:\Program Files\OpenSSL-Win64\bin>openssl x509 -in certificate.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            67:c8:66:59:cb:c9:d0:1e:88:44:58:6a:a3:86:e7:b2:15:7a:9e:32
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=ES, ST=Catalunya, L=Barcelona, O=IronHack, OU=JM_THE_KING, CN=Jose Manuel, emailAddress=julius.hacker88@gmail.com
        Validity
            Not Before: Jun  5 23:19:25 2024 GMT
            Not After : Jun  5 23:19:25 2025 GMT
        Subject: C=ES, ST=Catalunya, L=Barcelona, O=IronHack, OU=JM_THE_KING, CN=Jose Manuel, emailAddress=julius.hacker88@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (4096 bit)
            Modulus:
                00:bc:39:bb:15:35:0b:c7:3f:9b:a6:43:ec:87:06:
                16:c1:94:f8:7f:24:f4:71:ca:a0:a3:eb:c4:3e:99:
                96:ba:c1:79:b1:a0:62:bc:50:a9:ca:cb:3a:e7:ca:
                1c:fa:eb:ae:75:f7:46:6e:26:3a:49:3d:62:17:ad:
                d7:f5:ca:c1:de:81:59:c9:6a:63:6a:c6:3d:bd:99:
                2f:98:8c:46:3d:1b:f4:36:4f:7b:fb:2e:0b:44:17:
                20:1e:83:0e:92:f5:38:80:93:02:7b:3e:68:4d:11:
                1a:a8:9d:68:ae:f3:05:1a:91:c5:97:34:06:0d:dc:
                20:e3:e0:77:a7:78:a0:1a:95:04:b1:cc:c0:9f:01:
                83:85:75:56:7c:28:e5:b8:5f:fe:b0:e9:9a:1c:15:
                b3:41:60:b1:7d:86:99:1b:43:bb:aa:5e:44:00:d3:
                d8:84:26:93:ad:66:5e:94:5f:15:64:c4:a1:82:63:
                31:c2:8e:b9:5c:d6:50:08:e8:97:a2:78:25:12:b5:
                41:28:43:08:16:fd:8e:28:85:05:82:3c:3a:92:1e:
                8a:02:4e:a1:77:81:a8:32:cd:0f:79:eb:7c:18:34:
                8f:95:25:a8:15:63:89:cc:20:4b:56:3a:ee:f3:3c:
                c2:ad:f1:37:45:a4:bb:be:30:05:8d:dd:ac:d4:0c:
                34:ea:56:16:7d:30:80:de:95:18:7f:e3:07:e4:0f:
                ec:b7:ea:07:43:67:23:ec:90:4c:21:74:8a:89:31:
                7e:cc:d4:59:fd:d5:d6:88:3e:29:6c:55:23:e7:5b:
                5a:53:3e:8f:f8:92:a4:c6:e3:42:78:c4:a5:b4:b7:
                05:f6:c1:c1:e9:45:85:75:12:3c:85:75:51:66:b9:
                46:89:22:5a:fc:14:b9:fa:0b:23:7f:d6:a2:9d:b8:
                f9:26:91:6b:cb:f1:78:6b:30:5c:87:08:40:1a:34:

```

```
8d:aa:5d:c7:1f:4e:df:1f:96:3b:a6:bf:c5:9c:11:
80:8e:9f:94:e1:e0:e0:ea:bf:d3:19:61:3e:6b:fc:
7b:ca:55:1e:ec:72:03:0f:5f:e9:99:02:c0:ec:ad:
77:20:13:ce:91:61:f7:3b:2a:b3:da:d8:c7:5c:6f:
9a:bd:54:8a:e1:5a:6f:e5:96:a4:3e:84:c9:63:a5:
4b:89:ad:17:ca:b9:cd:e0:da:c1:42:66:6d:c7:4e:
04:aa:f9:9b:4a:e4:49:bb:fc:09:b0:68:60:17:ed:
19:59:07:e4:e5:bf:1c:88:31:c9:f2:96:54:59:d3:
76:37:39:f8:2b:d5:77:0a:dd:2a:2c:b6:dc:ca:2a:
e4:a9:56:03:ac:89:38:01:b1:60:f0:d7:67:ea:92:
39:dd:2b
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Key Identifier:
FD:09:72:28:09:A4:2B:C0:0F:94:30:4A:60:26:4C:8F:A6:C3:8B:42
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
2a:98:53:80:39:a1:79:54:ac:bb:28:2c:f9:f3:85:b5:4d:af:
00:15:08:dc:8d:af:6e:5b:77:1c:6e:61:fd:30:fb:24:96:ff:
d7:9f:e2:a9:72:54:76:28:42:f8:23:67:7e:d5:6c:ad:e2:a5:
79:f9:f7:0f:64:7b:fb:bf:39:af:ed:0b:2b:9c:d4:60:a2:74:
a5:64:0d:ff:a3:4c:2e:f2:2c:34:c4:5c:e4:2f:27:b0:b9:d0:
f8:36:68:a4:b5:86:a6:10:66:aa:4c:c6:4c:e5:a1:86:16:37:
38:35:d7:8b:cc:8e:94:c1:b4:4e:fa:68:23:ea:11:c8:47:1a:
6b:80:a8:2a:60:16:f8:77:8f:39:98:a8:1b:f9:90:e2:9f:da:
ef:72:c8:9d:c0:28:ef:33:e0:98:7a:d1:7f:c3:85:3c:5a:9f:
c5:26:43:cd:08:03:e4:19:dc:3e:e9:4a:71:28:4d:75:2c:6e:
7c:67:21:3c:3d:6a:6e:57:fe:00:85:93:9e:c6:0b:99:86:cf:
1c:36:73:7b:e2:42:2e:b8:cd:93:57:ea:6e:03:60:07:da:a7:
6c:f2:60:89:bb:eb:6b:3b:6b:e0:58:96:a4:3d:aa:8f:c0:17:
82:ed:46:3f:ac:5f:a9:8b:32:70:af:6b:e9:b5:9f:22:75:a3:
87:4b:50:c9:1d:d6:9a:4e:5b:78:23:c2:4a:f5:98:f7:47:bb:
b4:32:99:0b:d5:e7:64:4a:0c:3f:87:a9:3c:22:ee:7c:ad:f6:
68:77:16:63:41:e1:69:23:d9:d9:2a:06:3a:01:73:7b:ed:ae:
1b:77:10:7b:9f:fa:4f:f6:73:a0:97:d6:7c:96:c7:01:c0:4f:
c2:1d:27:ff:2b:a2:7c:c4:a7:a7:6d:38:18:aa:b5:26:2b:43:
92:0d:e7:92:1d:71:df:ba:82:b1:59:84:db:ac:f3:69:be:1b:
5d:73:5c:70:87:37:7e:cd:83:ab:04:c6:65:59:84:24:f6:c4:
9f:66:40:e9:0b:89:79:4c:9b:40:83:21:aa:53:5c:be:b9:84:
19:1f:73:8a:0a:13:3b:0f:67:ac:03:24:06:1d:df:c0:8d:81:
42:50:fd:b0:5b:0d:6d:39:aa:31:00:cb:a5:27:78:9e:e5:8b:
52:bb:83:fe:cb:d5:47:64:31:b0:41:cb:6e:70:a0:a8:46:ba:
```

```
52:7f:26:7b:5c:a5:91:e6:73:5c:f9:51:c2:56:06:c9:b1:a8:
3c:8b:88:4f:37:35:c2:2f:d4:46:be:97:43:d2:30:a6:25:25:
45:36:f0:17:34:1a:fd:d1:e3:69:16:99:fa:7e:21:1c:f0:a5:
5a:7d:0e:fa:91:bc:8d:4f
```

Linux

En Kali Linux, que es la máquina virtual que utilizamos en nuestro laboratorio de Virtualbox para el desarrollo de todo el curso, openssl ya viene instalado por lo que vamos directo a la ejecución de la herramienta.

Crearemos un certificado X.509 autofirmado y una clave privada. Para ello utilizamos el comando:

```
openssl req -x509 -nodes -days 365 -newkey rsa:4096 -keyout selfsigned.key -out selfsigned.crt
```

Este comando generará una nueva clave privada RSA de 4096 bits y un certificado autofirmado válido por 365 días. La clave privada se guardará en el archivo selfsigned.key y el certificado autofirmado se guardará en el archivo selfsigned.crt.

Explicación de los parámetros:

- `openssl req`: Indica a OpenSSL que se va a generar una solicitud de certificado (CSR).
- `-x509`: Genera un certificado autofirmado en lugar de una solicitud de firma de certificado (CSR).
- `-nodes`: Omite la opción de cifrar la clave privada, lo que significa que no se solicitará una contraseña.
- `-days 365`: Establece la validez del certificado en 365 días.
- `-newkey rsa:4096`: Genera una nueva clave RSA de 4096 bits junto con el certificado.
- `-keyout selfsigned.key`: Especifica el archivo en el que se guardará la clave privada generada.
- `-out selfsigned.crt`: Especifica el archivo en el que se guardará el certificado autofirmado.

```
(root@kali)-[/home/kali]
# openssl req -x509 -nodes -days 365 -newkey rsa:4096 -keyout selfsigned.key -out selfsigned.crt

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]:ES
State or Province Name (full name) [Some-State]:Catalunya
Locality Name (eg, city) []:Barcelona
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Ironhack
Organizational Unit Name (eg, section) []:Ironhack
Common Name (e.g. server FQDN or YOUR name) []:Julius
Email Address []:julius.hacker88@gmail.com
```

Podemos verificar que nos genera lo mismo que anteriormente con Windows y debemos completar las preguntas que se van generando.

Para verificar el fichero usamos el comando: 'openssl x509 -in selfsigned.crt -text -noout'

```
(root@kali)-[/home/kali]
# openssl x509 -in selfsigned.crt -text -noout

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5c:fd:ae:96:2b:03:6a:67:c2:d0:ad:5f:63:2b:0d:07:63:e4:fd
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = ES, ST = Catalunya, L = Barcelona, O = Ironhack, OU = Ironhack, CN = Julius, emailAddress = julius.hacker88@gmail.com
        Validity
            Not Before: Jun  5 23:56:10 2024 GMT
            Not After : Jun  5 23:56:10 2025 GMT
        Subject: C = ES, ST = Catalunya, L = Barcelona, O = Ironhack, OU = Ironhack, CN = Julius, emailAddress = julius.hacker88@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (4096 bit)
            Modulus:
                00:b2:5e:b1:25:70:78:f9:5e:ca:34:bd:95:a6:90:
                c6:54:4c:be:40:e4:65:73:19:cb:73:d5:b6:0f:c9:
                da:95:c0:78:b2:bb:38:40:e4:68:11:44:4b:83:83:
                af:f9:a0:b9:dd:6f:30:c9:b9:35:b3:80:0b:bb:42:
                32:b5:51:00:6d:fc:56:4a:b7:9d:9f:e6:b2:22:a1:
                e6:03:27:ff:96:3c:2f:8d:d2:94:05:eb:d6:c6:1a:
                cc:77:bb:12:f8:ae:44:f3:5b:2d:e1:0f:28:12:e1:
                ad:a1:b5:72:4a:29:1b:fc:91:ea:7b:87:88:8d:cf:
                5b:65:dc:ee:07:55:65:5b:e9:04:ef:8a:9e:f7:39:
                92:5d:a8:fd:eb:b7:fb:4f:f3:df:fd:7b:20:a4:14:
                b3:d1:ad:a1:96:af:ff:40:e9:f3:ba:dc:c1:83:b8:
                32:4b:f7:0b:ea:a0:e2:a1:00:4e:31:65:f8:be:d2:
                87:5b:45:48:5a:58:68:68:71:cd:85:14:b5:48:91:
                cf:d6:5c:51:8a:6a:b4:a6:a0:1b:1b:fd:97:79:b1:
                07:37:df:9c:be:00:21:11:7f:58:a5:5f:ae:ca:34:
                61:82:19:ec:81:ba:12:71:0c:b3:72:76:b3:2e:f2:
                3d:fa:9f:62:ee:59:5e:1e:ae:40:ec:81:5c:bd:90:
                74:14:b3:92:cd:2b:1a:f3:89:81:6d:5d:a7:43:aa:
                c0:b7:58:4e:1d:56:39:cf:b2:7b:fe:86:42:ce:d9:
                48:fa:a0:0d:e8:e3:91:25:37:dd:c3:06:a7:d4:1c:
                fc:8e:ad:97:7c:c5:b3:af:b8:fa:de:2c:f4:7c:18:
                52:55:85:32:3c:88:77:68:17:16:c2:9d:b3:7d:2a:
                97:ab:aa:a9:d1:5d:e7:e1:f1:1e:7e:79:a5:1a:56:
                49:e5:03:76:c3:f2:88:3e:1d:25:e8:83:7b:99:8a:
                3a:e8:3d:59:b0:59:f5:2d:32:00:6a:47:4c:11:43:
                82:b9:c7:42:0b:71:84:46:66:ff:c0:77:66:34:cf:
                8c:65:85:c2:7a:e6:43:5e:de:ed:50:d2:85:60:7c:
                29:5a:51:7f:b4:f0:7b:08:06:67:94:fe:c4:87:8e:
                fe:ba:f3:b2:32:62:29:f0:94:b6:d3:a6:e4:2e:b7:
                e3:5f:2b:7d:98:40:95:44:01:66:47:fe:02:61:79:
                eb:0d:26:06:2d:6b:85:f6:9e:f9:e7:06:5d:25:7a:
```

```

Trash      86:5f:52:a5:77:4c:58:bd:8c:f8:ec:71:d0:6b:c5:
           3f:79:81:9a:0c:6c:9c:46:74:1d:a8:57:b0:27:f3:
           6e:79:97:51:3a:28:58:15:e3:8b:53:0f:66:92:be:
           ad:9b:7b
           Exponent: 65537 (0x10001)
           X509v3 extensions:
File System X509v3 Subject Key Identifier:
           CA:34:DE:A4:07:10:04:AB:BC:AE:7D:03:9C:1E:04:2B:9E:AB:00:2C
           X509v3 Authority Key Identifier:
           CA:34:DE:A4:07:10:04:AB:BC:AE:7D:03:9C:1E:04:2B:9E:AB:00:2C
           X509v3 Basic Constraints: critical
           CA:TRUE
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
6d:16:b0:d3:7a:b2:8b:9d:f0:34:af:ad:5d:8a:c5:60:00:1a:
Ho b1:14:57:d4:17:e2:3e:1a:70:32:53:2c:d9:b7:9b:2d:47:89:
18:3f:65:09:a9:87:b3:3c:ee:56:7b:b9:02:38:5a:4b:b6:c9:
e3:99:41:07:5c:dc:15:1a:15:ee:1e:23:f7:0f:90:12:0f:45:
ca:8c:35:fc:f4:c7:f6:b0:3b:23:34:e1:52:3f:00:3d:04:b6:
06:3d:db:5c:42:f6:8f:01:e5:c9:0d:52:1b:ab:67:c5:32:2b:
95:3a:c8:e4:84:3c:56:e3:a9:3e:24:62:44:85:e6:04:c7:92:
62:9a:d8:5d:34:29:65:1b:8a:9a:cd:dd:e3:7b:db:1e:2b:1b:
f9:78:03:fe:49:b7:83:42:b0:a7:38:e5:ae:49:88:d9:5e:55:
18:85:92:6c:6a:47:12:f3:a3:e4:5b:4c:65:ff:c5:d3:4f:24:
5e:ab:86:94:94:c2:d9:72:bc:da:03:06:9d:20:af:97:2f:59:
fd:1c:d5:f8:8b:0f:a0:51:a7:5c:ec:36:86:6f:70:42:f6:70:
50:43:b5:5f:16:3e:f1:cc:cd:e9:fe:1f:36:e4:94:19:cb:ad:
da:b9:d3:3e:56:fe:d4:2b:89:fe:1d:55:4e:2b:13:c3:85:46:
a5:3e:fd:55:a2:96:e1:c6:78:d1:40:07:09:19:e1:f4:00:b1:
30:e8:8e:8b:26:d7:41:c0:68:6d:d8:06:f6:c7:20:da:3a:0e:
9a:60:5e:73:8a:0a:37:31:02:fd:4d:d8:84:c5:a8:fd:1a:91:
dc:98:df:8e:46:9c:94:dd:ec:be:50:04:a2:ca:c7:ca:4d:8d:
07:52:09:91:a7:c2:9b:3c:8b:62:7e:b0:90:82:88:45:e3:3a:
5a:90:60:84:a7:35:24:cc:51:d2:32:46:0e:c7:d8:3d:61:8a:
95:d9:9a:b6:25:ff:57:f4:bc:f5:e4:7e:8b:31:e0:9e:cd:0c:
c0:dd:59:8f:fc:3d:3a:7c:57:eb:82:e9:c7:e8:36:24:fe:20:
21:f7:23:02:b5:9e:f6:81:07:6f:6f:5d:f4:ef:94:4a:21:70:
05:a7:c5:03:65:a7:0f:75:a7:bd:85:f5:76:48:c9:0b:fb:6a:
0f:71:76:9a:75:29:2c:a3:70:59:12:bb:ad:75:97:9a:0e:0e:
70:89:1c:ce:1b:bd:61:9d:ab:fa:06:08:15:56:6a:cc:5e:2b:
dc:18:f2:69:65:d0:26:9c:30:07:33:58:bd:15:00:be:97:99:
de:24:fe:81:43:3e:4d:d4:96:92:c7:4a:5f:0a:7d:01:3d:aa:
36:bd:cb:26:19:bb:75:f4

```

Conclusiones

En esta actividad, nos enfocamos en la práctica y comprensión de la utilización de certificados PGP y X.509 en entornos tanto Windows como Linux. A lo largo del proceso, realizamos una serie de tareas que reforzaron nuestro conocimiento y habilidades en la criptografía de clave pública y la gestión de certificados digitales.

Certificados PGP

La primera parte del trabajo consistió en manejar certificados PGP. Importamos la clave pública del profesor y creamos un archivo cifrado con el contenido "Solución P3" en ambos sistemas operativos. Este ejercicio destacó la importancia de la criptografía de clave pública para asegurar la privacidad y autenticidad de la información:

- **Windows:** Utilizamos GPG4Win y su interfaz gráfica Kleopatra, lo que facilitó la generación y manejo de claves, así como el cifrado de archivos. La interfaz gráfica proporcionó una experiencia más accesible para usuarios no familiarizados con la línea de comandos.
- **Linux:** Implementamos PGP a través de la herramienta gnupg2, usando comandos de terminal para generar claves y cifrar archivos. La experiencia en Linux enfatizó la versatilidad y potencia de las herramientas de línea de comandos.

Certificados X.509 Autofirmados

La segunda parte del trabajo implicó la creación de certificados X.509 autofirmados utilizando OpenSSL:

- **Conceptos Teóricos:** Comprendimos que un certificado X.509 es un estándar crucial en la infraestructura de clave pública (PKI), utilizado para autenticar identidades y cifrar comunicaciones. Los certificados autofirmados, aunque menos confiables que aquellos firmados por una autoridad certificadora externa, son útiles para pruebas y entornos controlados.

- **Windows:** Instalamos OpenSSL y generamos un certificado X.509 autofirmado mediante comandos específicos en la terminal. Esta práctica demostró la importancia de la clave privada en el proceso de firma y la necesidad de mantenerla segura.
- **Linux:** En Kali Linux, ejecutamos comandos de OpenSSL para generar un certificado autofirmado, destacando la eficiencia y robustez de las herramientas criptográficas disponibles en sistemas operativos basados en Unix.

Aprendizajes Clave

1. **Criptografía de Clave Pública:** La utilización de PGP y certificados X.509 subraya la relevancia de la criptografía de clave pública en la protección de datos y la autenticación de identidades.
2. **Herramientas Multiplataforma:** Aprendimos a manejar herramientas criptográficas en diferentes sistemas operativos, desarrollando una comprensión integral de sus funcionalidades y diferencias.
3. **Seguridad y Confianza:** La creación de certificados digitales enfatiza la necesidad de gestionar adecuadamente las claves privadas y entender el papel de las autoridades certificadoras en la infraestructura de clave pública.

Esta práctica nos permitió consolidar conocimientos teóricos y prácticos en la gestión de certificados digitales, preparándonos para enfrentar desafíos de seguridad en entornos profesionales. La experiencia adquirida es fundamental para garantizar la integridad y autenticidad de la información en el ámbito de la seguridad informática.