UT02: – Técnicas De Cifrado.

Nombre: Ricardo Camacho Diaz-Roncero y Miguel Lillo Romero

Curso: 2º ASIR.

Índice

Introducción

En esta práctica aprenderemos sobre el cifrado, para ello realizaremos técnicas de cifrado y veremos la historia que tienen.

a) CIFRADO: Historia.

- Realizar CISCO CCNA Security 2.0. Laboratorio: Explorando métodos de cifrado.

El cifrado Vigenère es un cifrado basado en diferentes series de caracteres o letras del cifrado César formando estos caracteres una tabla, llamada tabla de Vigenère, que se usa como clave. El cifrado de Vigenère es un cifrado de sustitución simple poli alfabético. El cifrado Vigenère se ha reinventado muchas veces.

Encriptación

Cipher Keyword	F	R	A	N	V	I	G	S	A	D
Encrypted Message	R	O	U	T	E	R	R	O	U	T
Decrypted Message	W	F	U	G	Z	Z	X	G	U	W

Desencriptación

Cipher Keyword	W	F	U	G	Z	Z	X	G	U	W
Encrypted Message	R	O	U	T	E	R	R	O	U	T
Decrypted Message	F	R	A	N	V	I	G	S	A	D

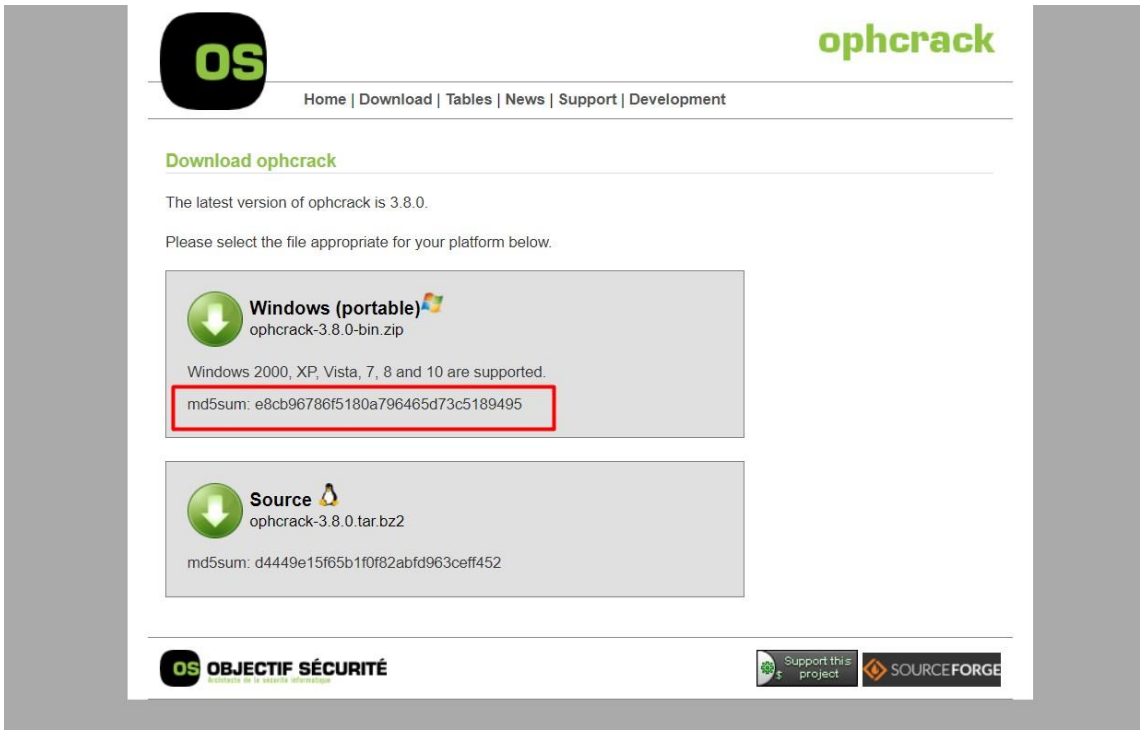
Página que recomienda Cisco para probar esto: Vignere Cipher

b) CIFRADO: Funciones HASH--> Integridad y Autenticidad.

- 1. Utiliza un programa en Windows o GNU/Linux para simular la integridad, utilizando MD5 y SHA1.

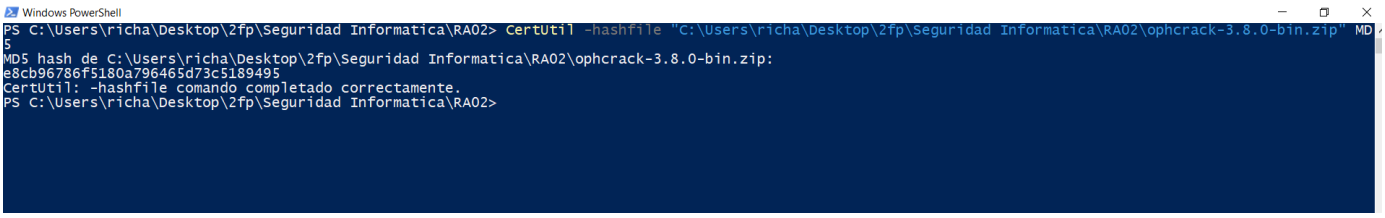
Para realizar esto usare PowerShell. PowerShell es una interfaz de consola (CLI) con posibilidad de escritura y unión de comandos por medio de instrucciones (script)). Esta interfaz de consola está diseñada para su uso por parte de administradores de sistemas, con el propósito de automatizar tareas o realizarlas de forma más controlada.

En mi caso lo usare comprobare la integridad de la versión portable de Ophcrack. Podemos ver el md5 desde su misma página web.



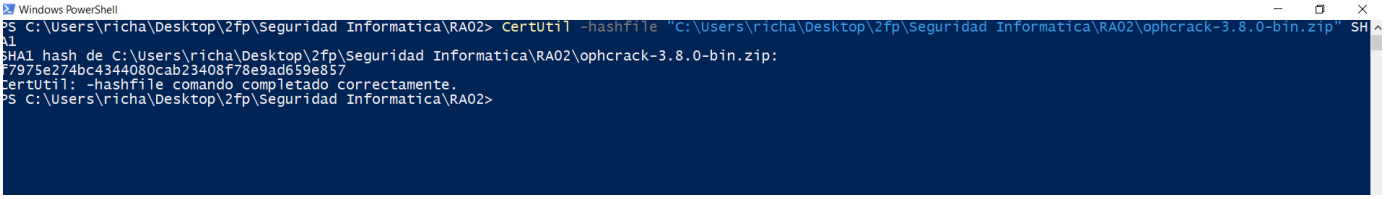
Una vez guardado el programa simplemente abrimos PowerShell y ejecutamos el siguiente comando para sacar el hash del archivo:

CertUtil -hashfile "C:\Users\richa\Desktop\2fp\Seguridad Informatica\RA02\ophcrack-3.8.0-bin.zip" MD5

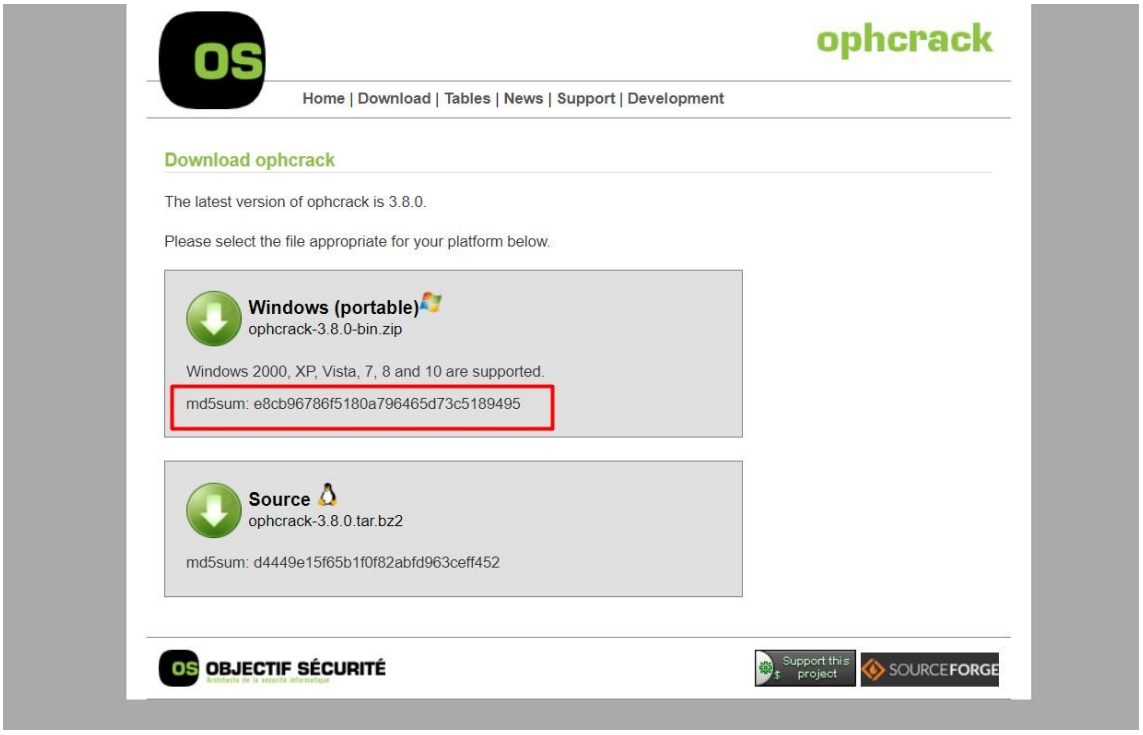


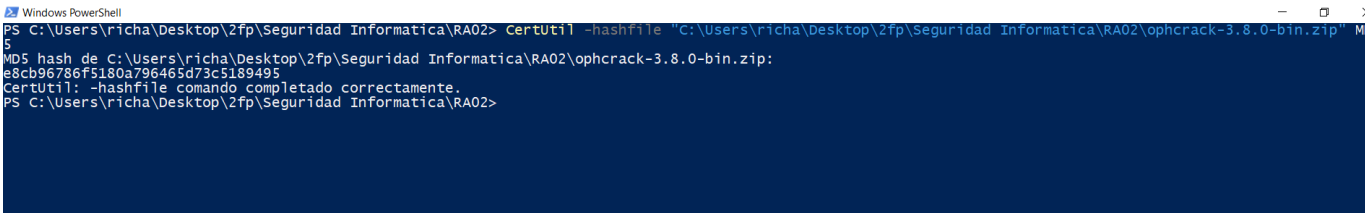
Vemos como nos genera un hash.

Podremos hacer lo mismo en SHA1.



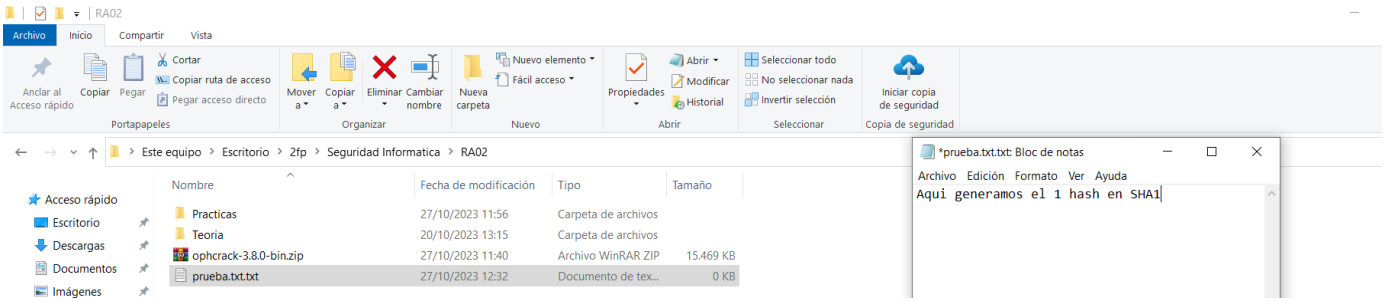
Ahora simplemente tenemos que comprar los dos hashes (el de PowerShell con el que ofrece la página Ophcrack. Vemos que son iguales por lo cual la integridad es correcta.

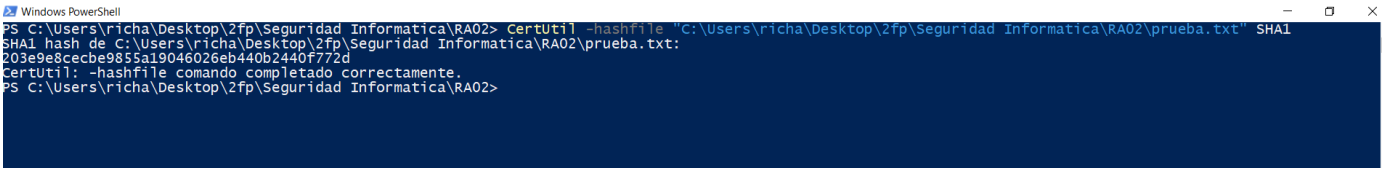




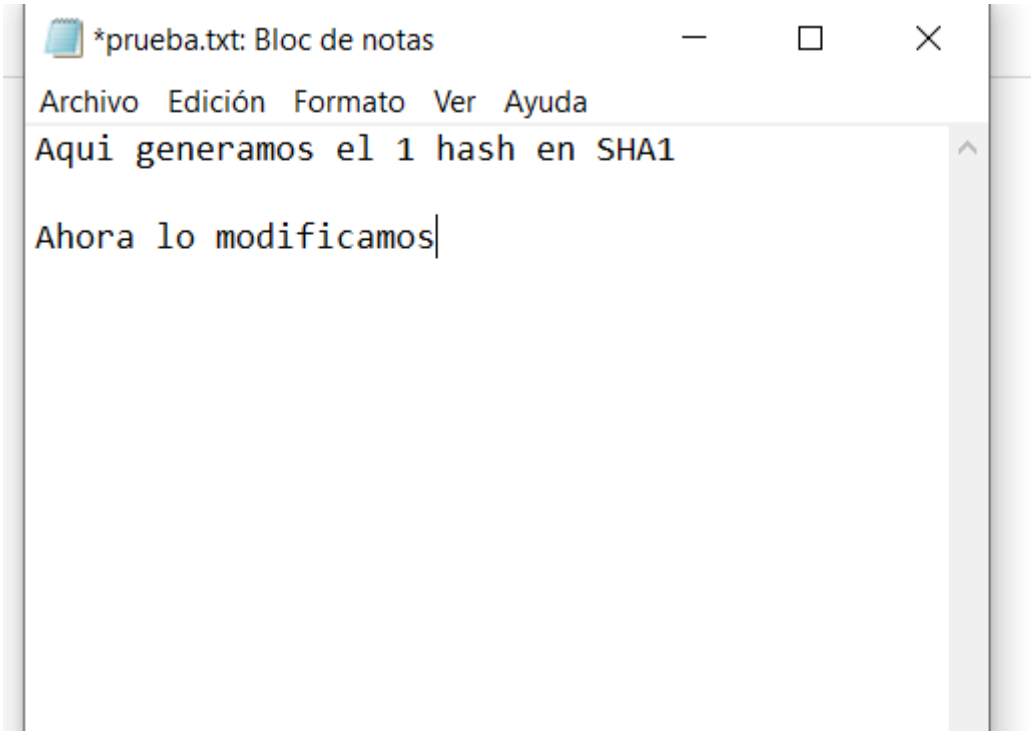
Para probar SHA1 al no encontrar programas que brinden en su página web el SHA1 para comprobar la integridad lo que haré es crear un archivo txt con algo escrito, generar su SHA1 y después modificarlo para que se vea cómo cambia.

Crearemos un documento de texto y generamos su SHA1.


RA02
Archivo
Inicio
Compartir
Vista
Anclar al Acceso rápido
Copiar
Pegar
Cortar
Copiar ruta de acceso
Pegar acceso directo
Portapapeles
Mover a
Copiar a
Eliminar
Cambiar nombre
Organizar
Nueva carpeta
Nuevo elemento
Fácil acceso
Nuevo
Propiedades
Abrir
Modificar
Historial
Abrir
Seleccionar todo
No seleccionar nada
Invertir selección
Seleccionar
Iniciar copia de seguridad
Copia de seguridad
Este equipo
Escritorio
2fp
Seguridad Informatica
RA02
Acceso rápido
Escritorio
Descargas
Documentos
Imágenes
Nombre
Fecha de modificación
Tipo
Tamaño
Practicas
27/10/2023 11:56
Carpeta de archivos
Teoria
20/10/2023 13:15
Carpeta de archivos
ophcrack-3.8.0-bin.zip
27/10/2023 11:40
Archivo WinRAR ZIP
15.469 KB
prueba.txt.txt
27/10/2023 12:32
Documento de tex...
0 KB
*prueba.txt.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
Aqui generamos el 1 hash en SHA1


Windows PowerShell
PS C:\Users\richa\Desktop\2fp\Seguridad Informatica\RA02> CertUtil -hashfile "C:\Users\richa\Desktop\2fp\Seguridad Informatica\RA02\prueba.txt" SHA1
SHA1 hash de C:\Users\richa\Desktop\2fp\Seguridad Informatica\RA02\prueba.txt:
203e9e8cecbe9855a19046026eb440b2440f772d
CertUtil: -hashfile comando completado correctamente.
PS C:\Users\richa\Desktop\2fp\Seguridad Informatica\RA02>

Si editamos el contenido vemos cómo cambia el SHA1.



2. Utiliza un programa en Windows o GNU/Linux para simular la autenticidad utilizando HMAC-MD5, HMAC-SHA1.

En términos criptográficos, es un tipo de código para autentificar mensajes en clave-hash (HMAC), es decir, es un diseño específico para generar un código de autentificación de mensaje (MAC) lo cual implica una función hash criptográfica junto con una llave criptográfica secreta.

También puede ser usada para verificar simultáneamente la autentificación de un mensaje y la integridad de los datos. El poder criptográfico del HMAC va a depender de la cantidad de caracteres de la clave secreta que se utilice.

Utilizaré dos páginas web conocidas para realizar esta tarea. La primera es la siguiente (Link). Introduciremos un mensaje con su clave y generaremos el HMAC-MD5.

This tool will allow you to test a variety of HMAC Hashing algorithms for a variety of purposes.

The String to be Encrypted

Hola somos Miguel y Ricardo

Secret Key

inves

Select Hash Algorithm to Use

md5

Generate Clear

This tool will allow you to test a variety of HMAC Hashing algorithms for a variety of purposes.

The String to be Encrypted

Hola somos Miguel y Ricardo

Secret Key

inves

Select Hash Algorithm to Use

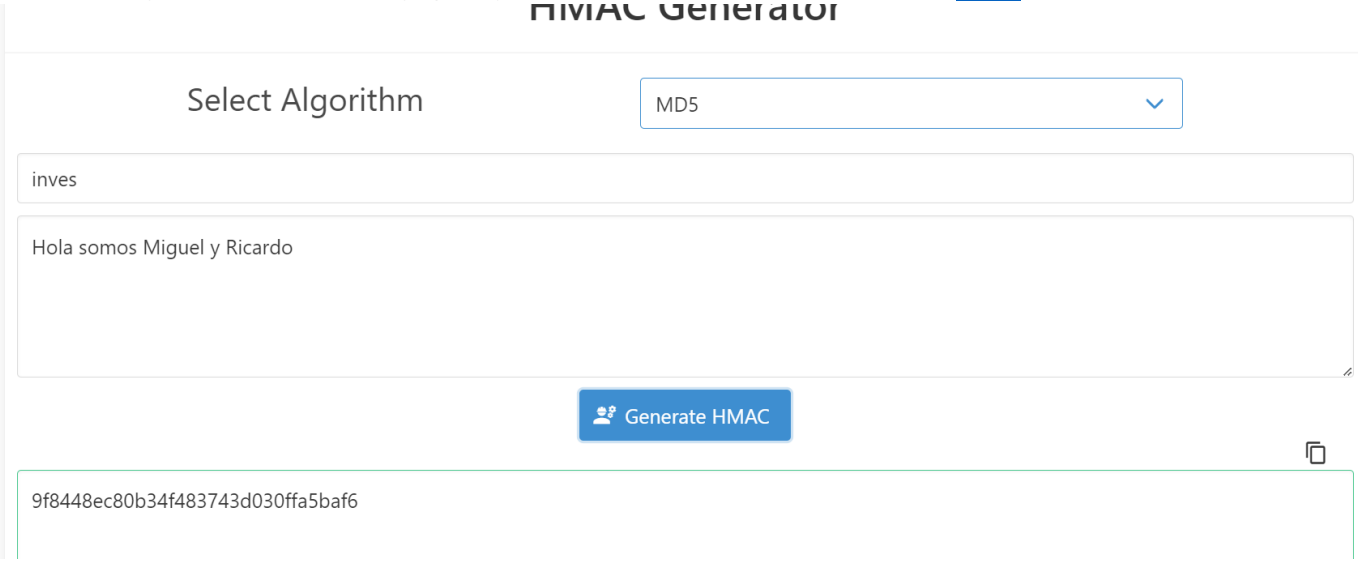
md5

Generate Clear

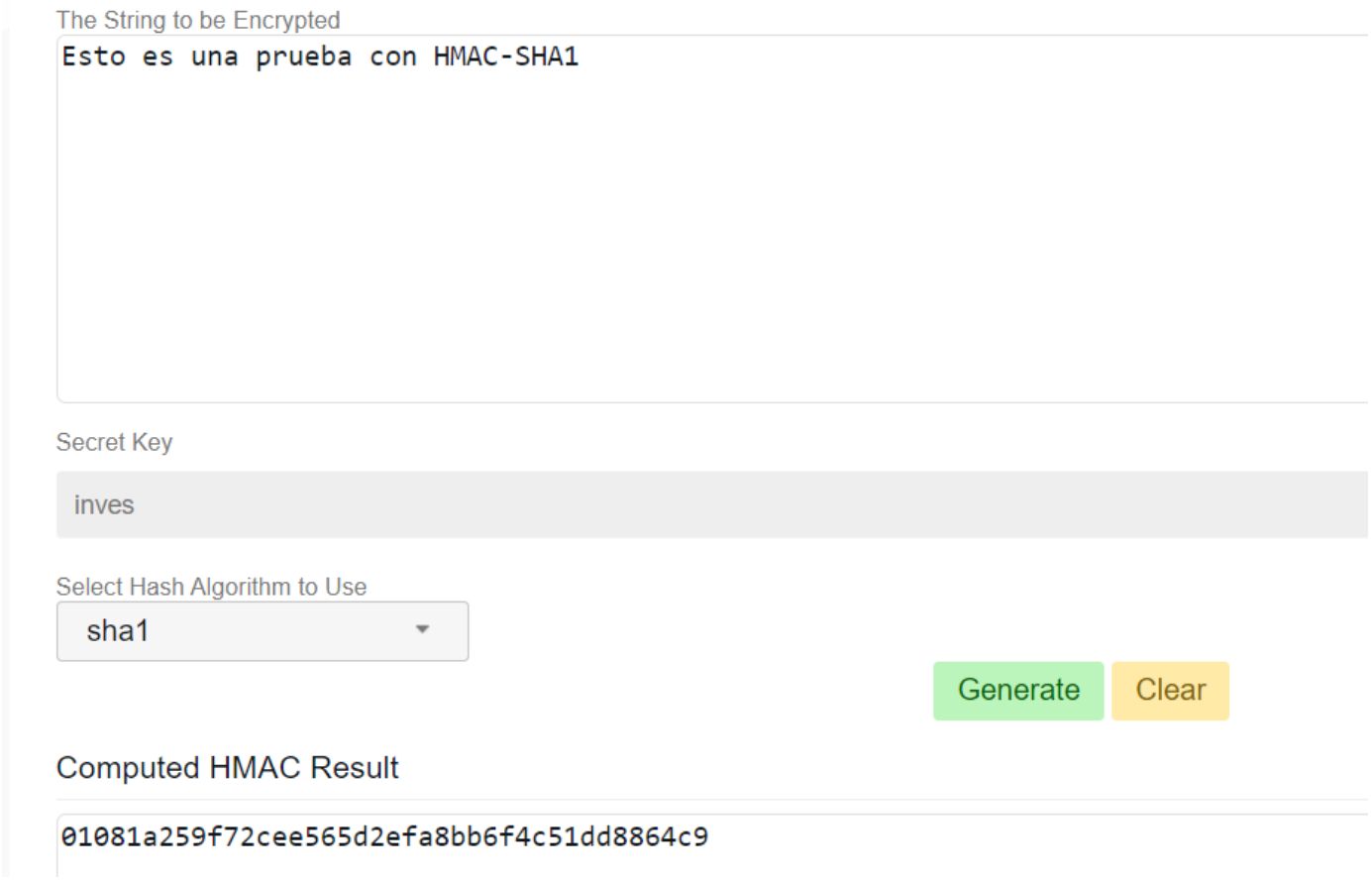
Computed HMAC Result

9f8448ec80b34f483743d030ffa5baf6

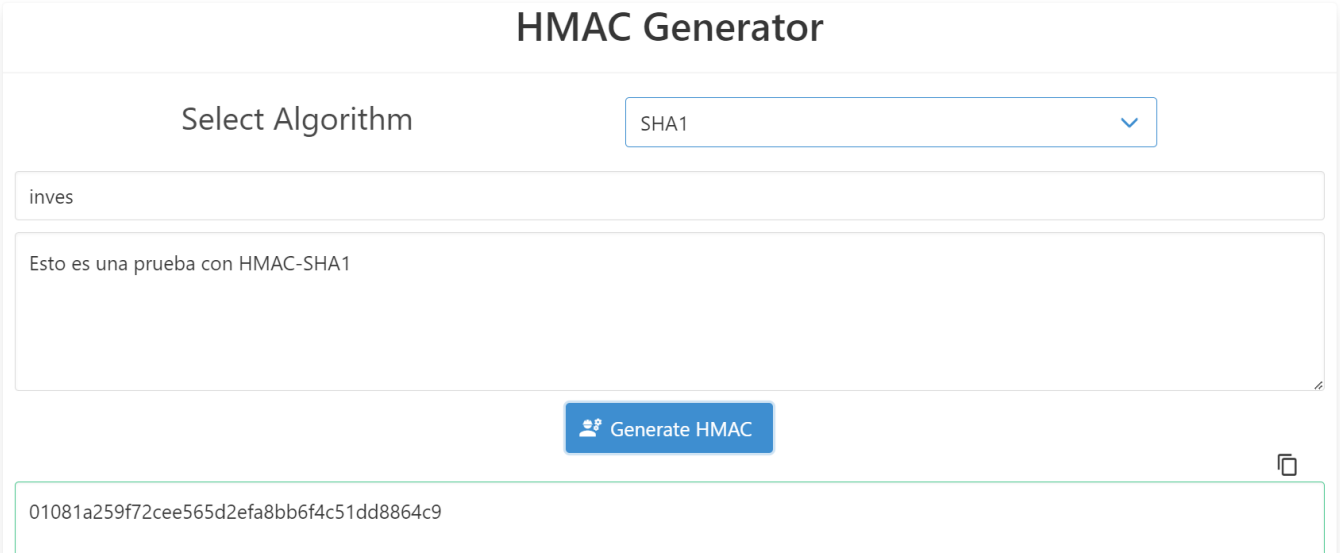
Comprobaremos con otra página que el resultado es realmente correcto. (Link)



Podemos ver que los dos hmac-hashes son iguales por lo cual la autenticidad es correcta. Lo siguiente es hacer lo mismo, pero en HMAC-SHA1, lo realizaremos en las páginas web de antes.



Lo compararemos con la otra página web. Vemos que es igual que la anterior página por lo cual la autenticidad es correcta.


HMAC Generator
Select Algorithm
SHA1
inves
Esto es una prueba con HMAC-SHA1
Generate HMAC
01081a259f72cee565d2efa8bb6f4c51dd8864c9

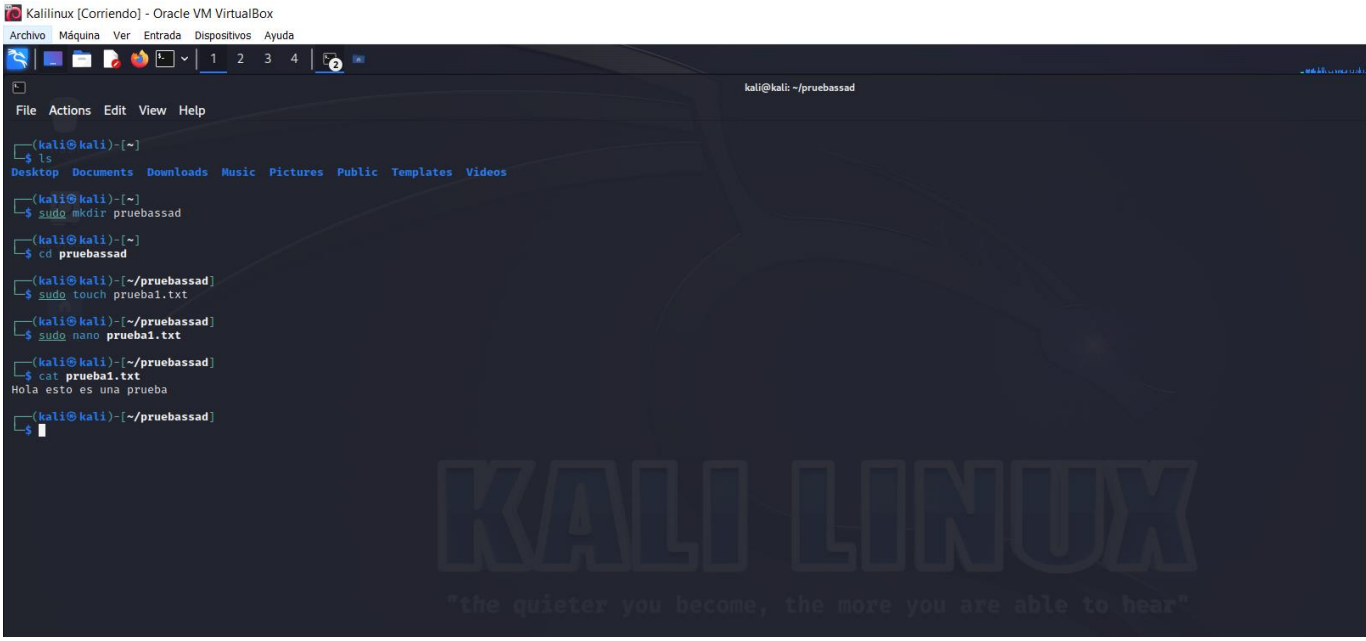
c) CIFRADO: Simétrico y Asimétrico--> Confidencialidad, Integridad y Autenticidad.

- 1. Utiliza un programa en Windows o GNU/Linux para simular la confidencialidad mediante “cifrado simétrico”.

La criptografía de clave simétrica, también llamada criptografía de clave secreta o criptografía de una clave, es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y el receptor.

Utilizaremos para llevar acabo esta práctica Linux con Kalilinux El programa que usare es pgp.

Primero crearemos un documento y lo cifraremos con una clave simétrica.

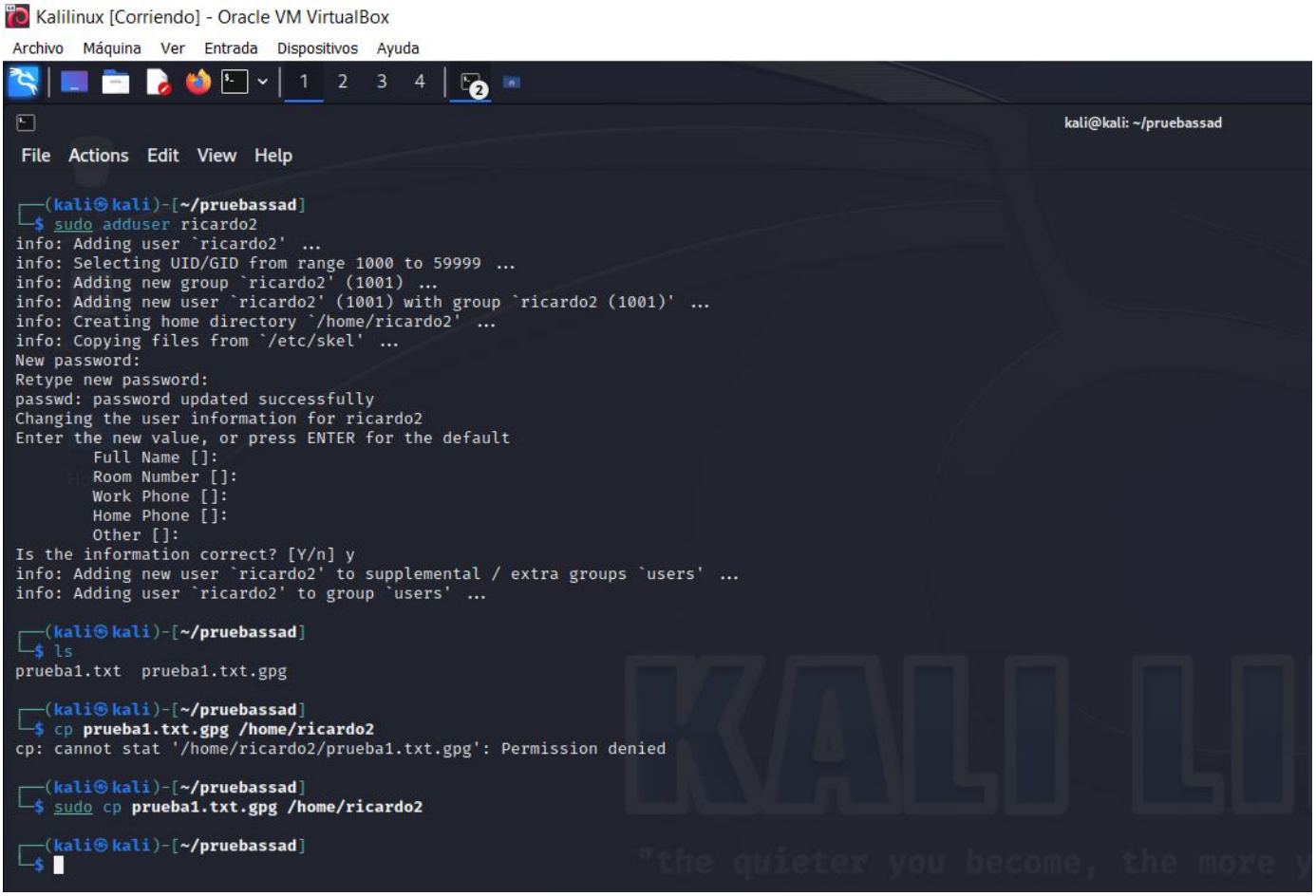


Ahora lo cifraremos de manera simétrica con gpg.

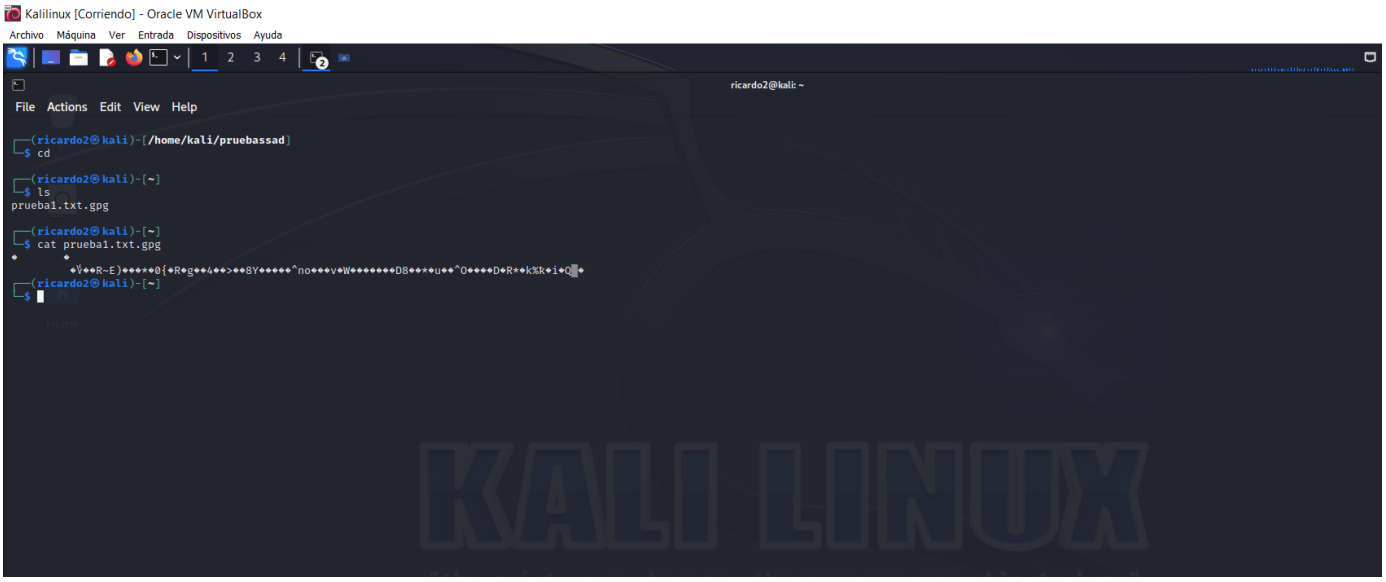
Ponemos una clave.

Vemos como nos generó el mismo archivo, pero con extensión. gpg

Ahora simplemente enviaremos el archivo cifrado con clave simétrica al /home del segundo usuario que tenemos.



Iniciaremos sesión con el segundo usuario e intentaremos ver el contenido con un simple cat. Vemos como lógicamente no deja al estar cifrado.



Simplemente lo descifraremos con gpg, nos pedirá la contraseña que pusimos al cifrar el archivo.

$ sudo gpg –decrypt prueba1.txt.gpg

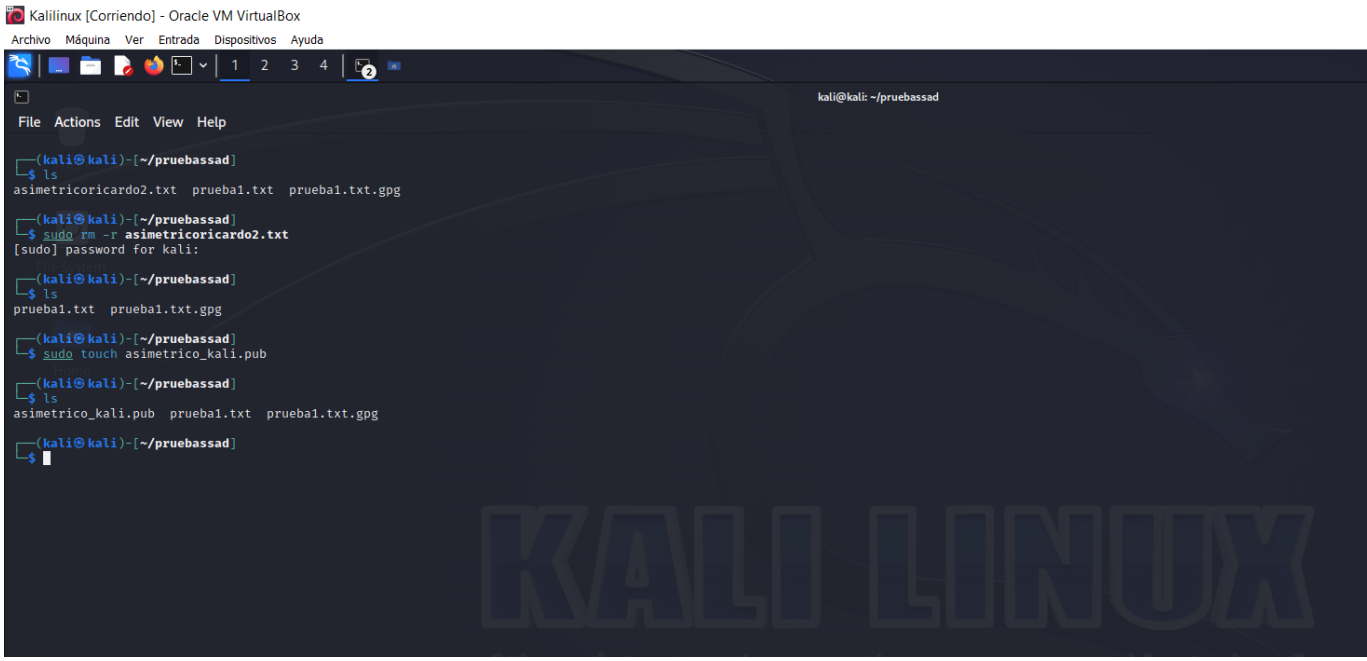
Vemos como nos descifra el archivo y podemos ver su contenido.

- 2. Utiliza un programa en Windows o GNU/Linux para simular la confidencialidad mediante “cifrado asimétrico”.

La criptografía asimétrica, también llamada criptografía de clave pública o criptografía de dos claves, es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que recibirá el mensaje.

Para realizar esta práctica, igual que antes, utilizaremos gpg en Kalilinux.

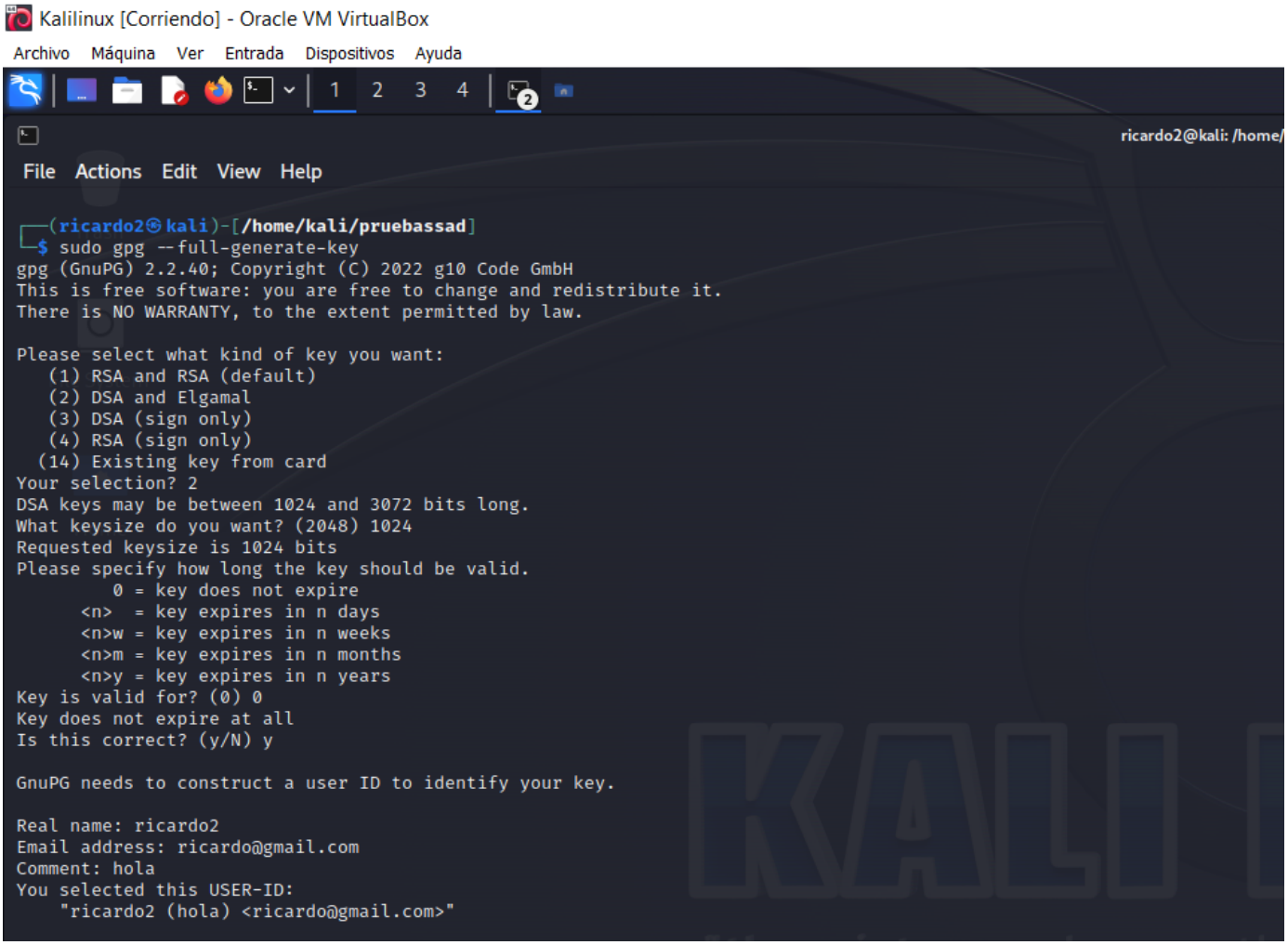
Crearemos para ello un nuevo archivo con la cuenta ricardo2.


Kalilinux [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
1 2 3 4
kali@kali: ~/pruebassad
File Actions Edit View Help
(kali㉿kali)-[~/pruebassad]
$ ls
asimetricoricardo2.txt prueba1.txt prueba1.txt.gpg
(kali㉿kali)-[~/pruebassad]
$ sudo rm -r asimetricoricardo2.txt
[sudo] password for kali:
(kali㉿kali)-[~/pruebassad]
$ ls
prueba1.txt prueba1.txt.gpg
(kali㉿kali)-[~/pruebassad]
$ sudo touch asimetrico_kali.pub
(kali㉿kali)-[~/pruebassad]
$ ls
asimetrico_kali.pub prueba1.txt prueba1.txt.gpg
(kali㉿kali)-[~/pruebassad]
$
KALI LINUX

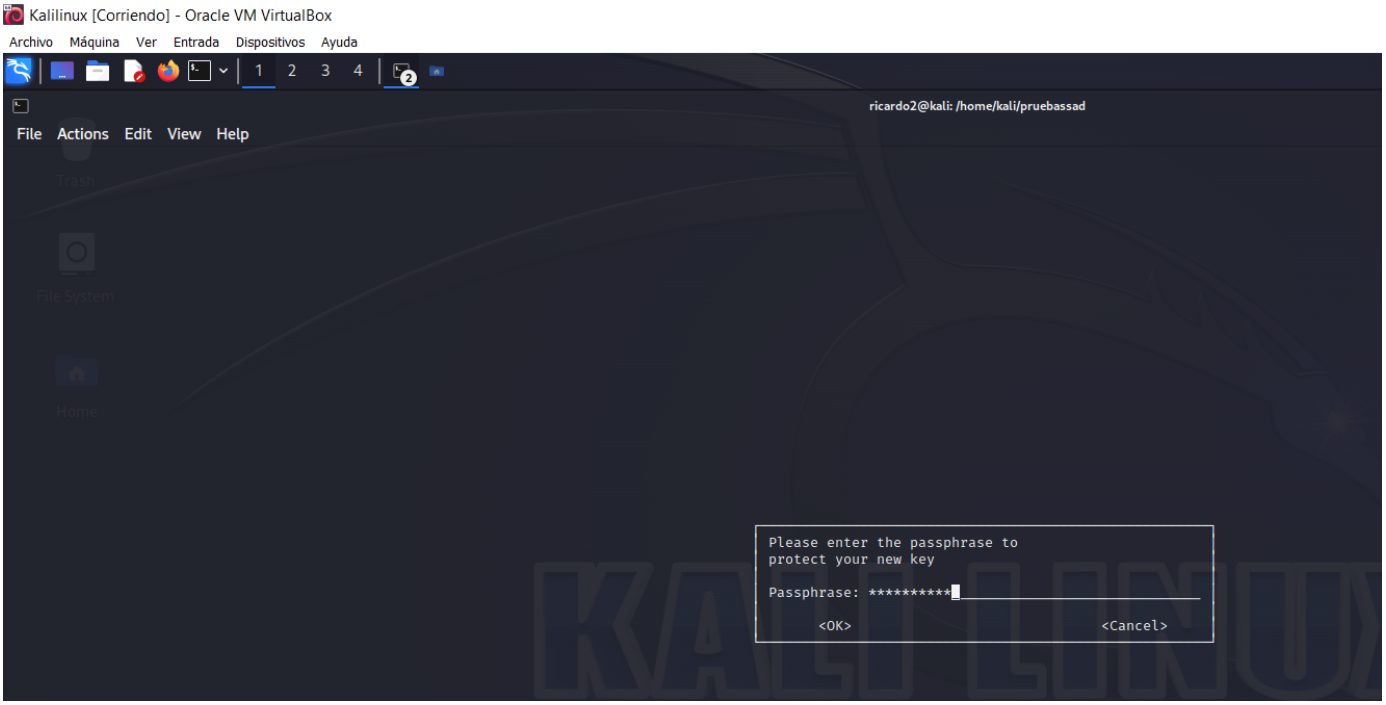
Ahora generaremos las claves necesarias (dos, una publica y otra privada) para cada usuario.

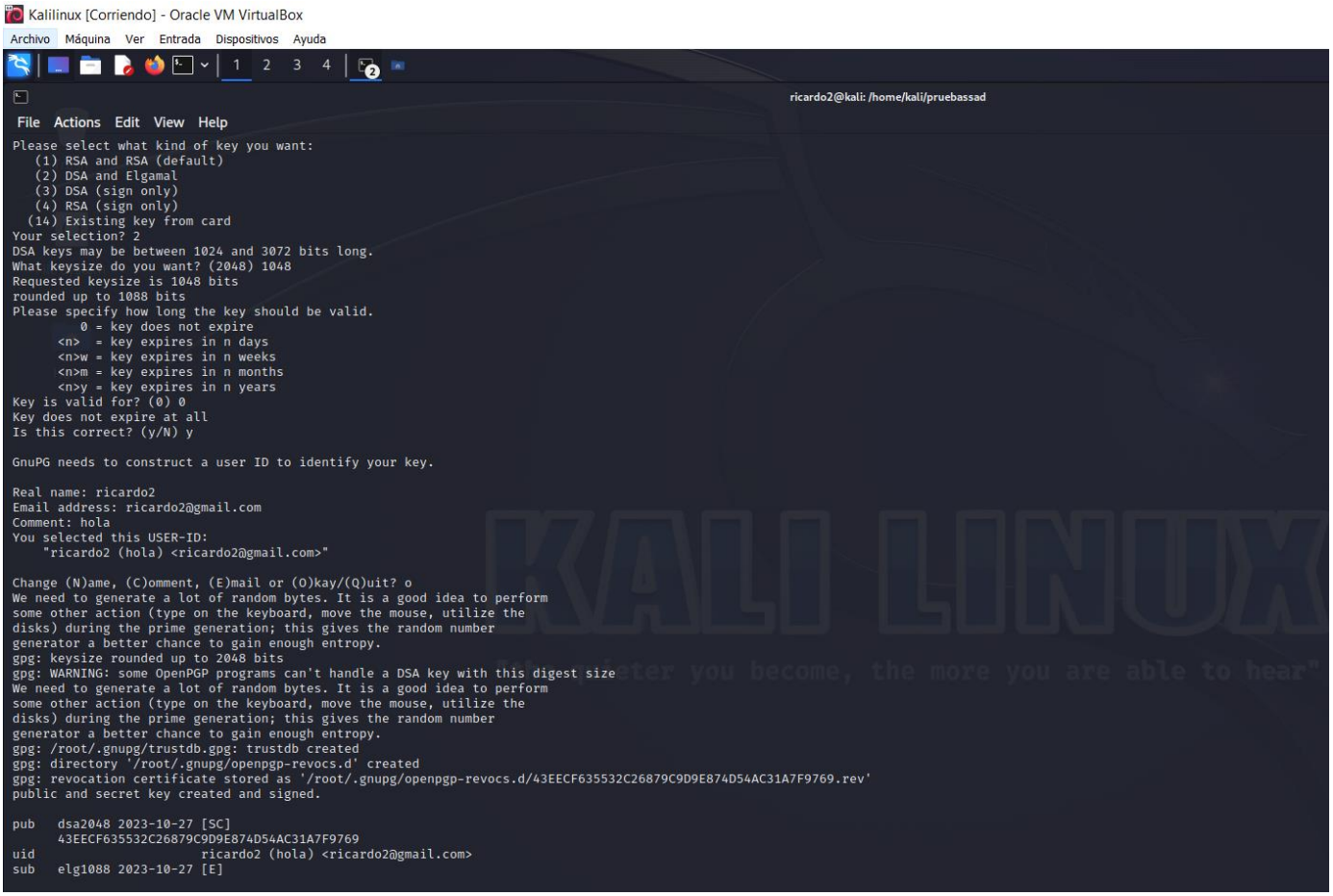
Primero crearemos las claves del primer usuario.

$ gpg –full-generate-key



Le ponemos una contraseña.

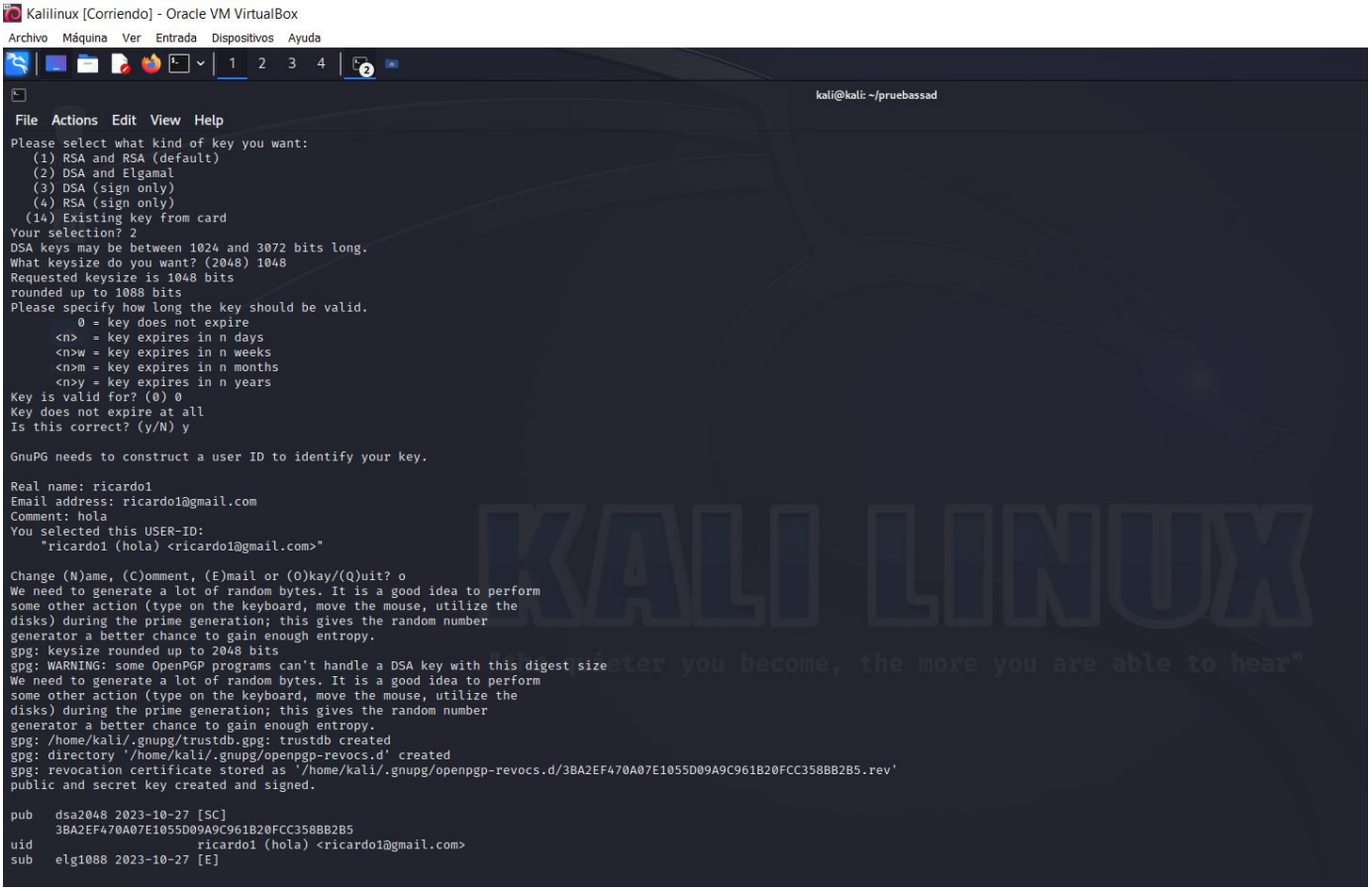

Kalilinux [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
1 2 3 4
ricardo2@kali: /home/kali/pruebassad
File Actions Edit View Help
Please enter the passphrase to
protect your new key
Passphrase: **********
<OK>
<Cancel>


Kalilinux [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
1 2 3 4
ricardo2@kali: /home/kali/pruebassad
File Actions Edit View Help
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)
(14) Existing key from card
Your selection? 2
DSA keys may be between 1024 and 3072 bits long.
What keysize do you want? (2048) 1048
Requested keysize is 1048 bits
rounded up to 1088 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 needs to construct a user ID to identify your key.
Real name: ricardo2
Email address: ricardo2@gmail.com
Comment: hola
You selected this USER-ID:
"ricardo2 (hola) <ricardo2@gmail.com>"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
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: keysize rounded up to 2048 bits
gpg: WARNING: some OpenPGP programs can't handle a DSA key with this digest size
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/43EECF635532C26879C9D9E874D54AC31A7F9769.rev'
public and secret key created and signed.
pub dsa2048 2023-10-27 [SC]
43EECF635532C26879C9D9E874D54AC31A7F9769
uid ricardo2 (hola) <ricardo2@gmail.com>
sub elg1088 2023-10-27 [E]

Ahora, vamos al otro usuario (kali) y generaremos su clave igual que hicimos con ricardo2.

Le ponemos una contraseña.

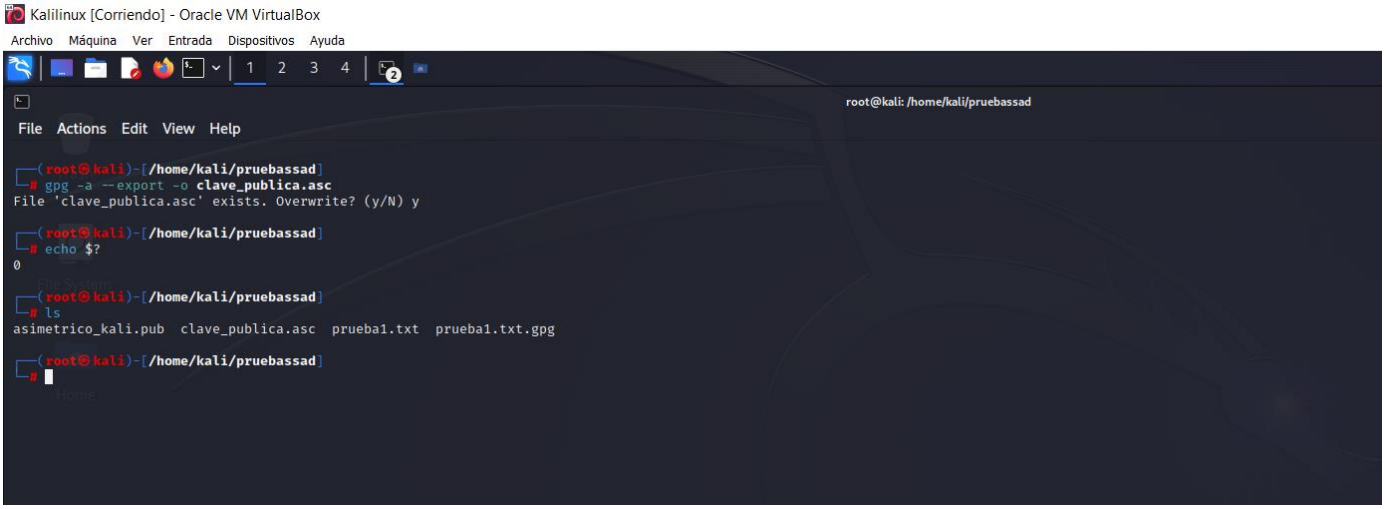
Ya la tendremos generada.


Kalilinux [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
1 2 3 4
kali@kali: ~/pruebassad
File Actions Edit View Help
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)
(14) Existing key from card
Your selection? 2
DSA keys may be between 1024 and 3072 bits long.
What keysize do you want? (2048) 1048
Requested keysize is 1048 bits
rounded up to 1088 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 needs to construct a user ID to identify your key.
Real name: ricardo1
Email address: ricardo1@gmail.com
Comment: hola
You selected this USER-ID:
"ricardo1 (hola) <ricardo1@gmail.com>"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
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: keysize rounded up to 2048 bits
gpg: WARNING: some OpenPGP programs can't handle a DSA key with this digest size
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: /home/kali/.gnupg/trustdb.gpg: trustdb created
gpg: directory '/home/kali/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/kali/.gnupg/openpgp-revocs.d/3BA2EF470A07E1055D09A9C961B20FCC358BB2B5.rev'
public and secret key created and signed.
pub dsa2048 2023-10-27 [SC]
3BA2EF470A07E1055D09A9C961B20FCC358BB2B5
uid ricardo1 (hola) <ricardo1@gmail.com>
sub elg1088 2023-10-27 [E]

Ahora simplemente exportaremos las claves publicas de los dos usuarios.

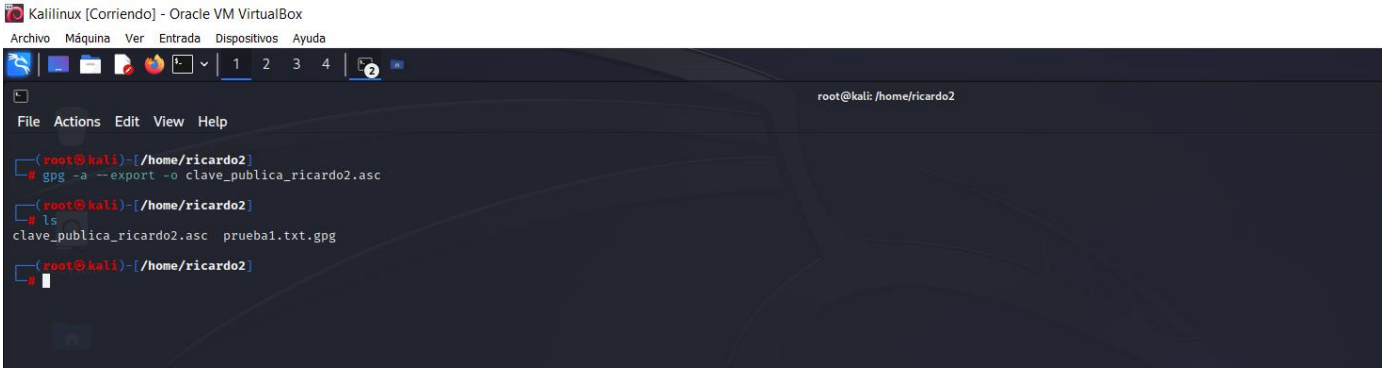
kali:

$ gpg -a --export -o clavepublica.asc



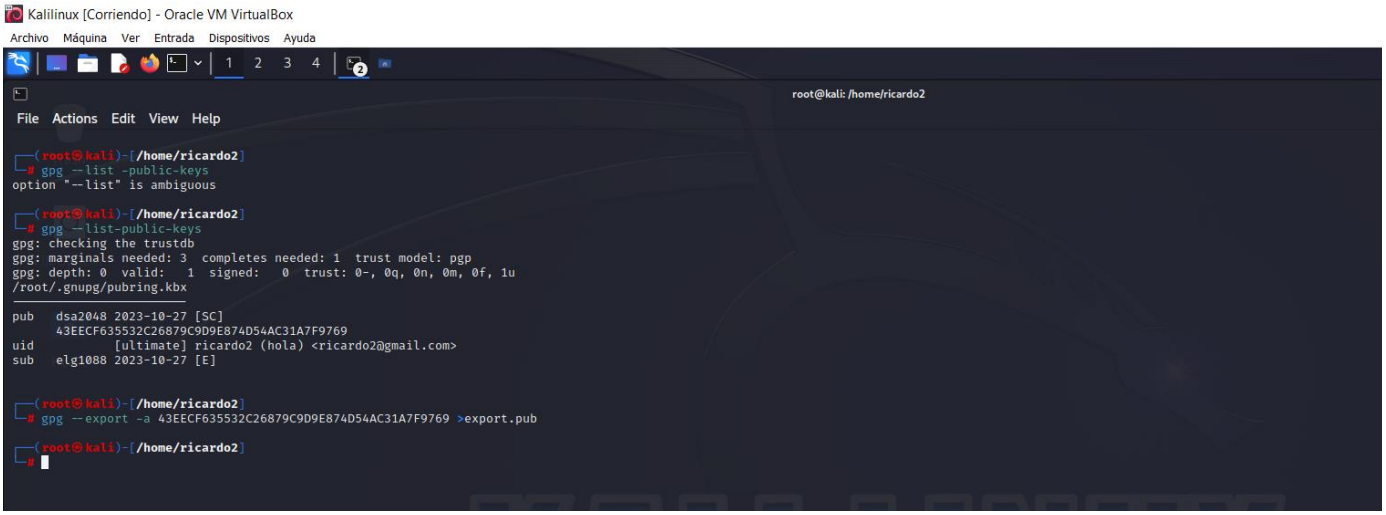
Ricardo2:

$ gpg -a --export -o clave_publica_ricardo2.asc



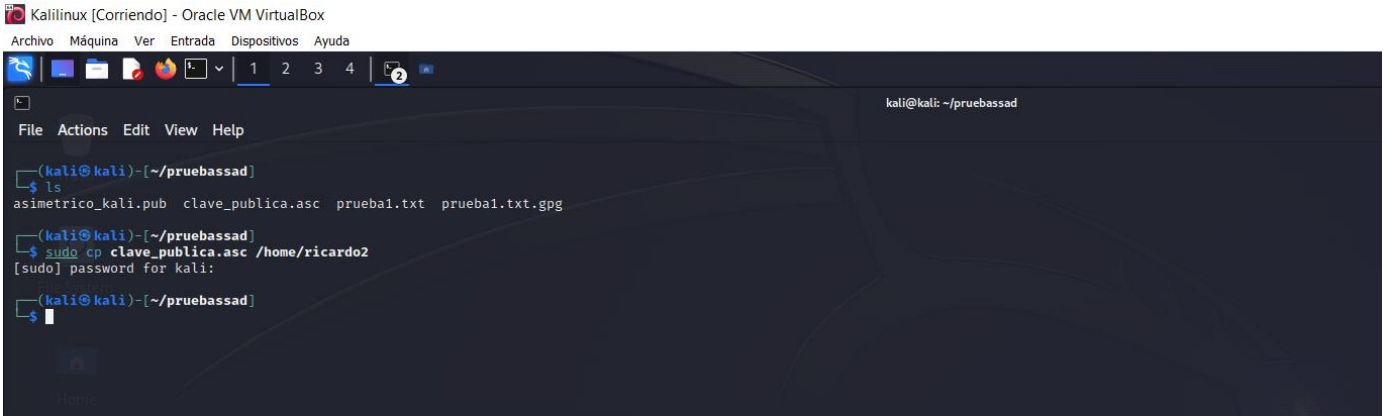
Recordamos que con el comando *gpg –list-public-keys* podemos ver las claves públicas que tenemos.

Podemos de esa forma exportar la clave publica que queramos.

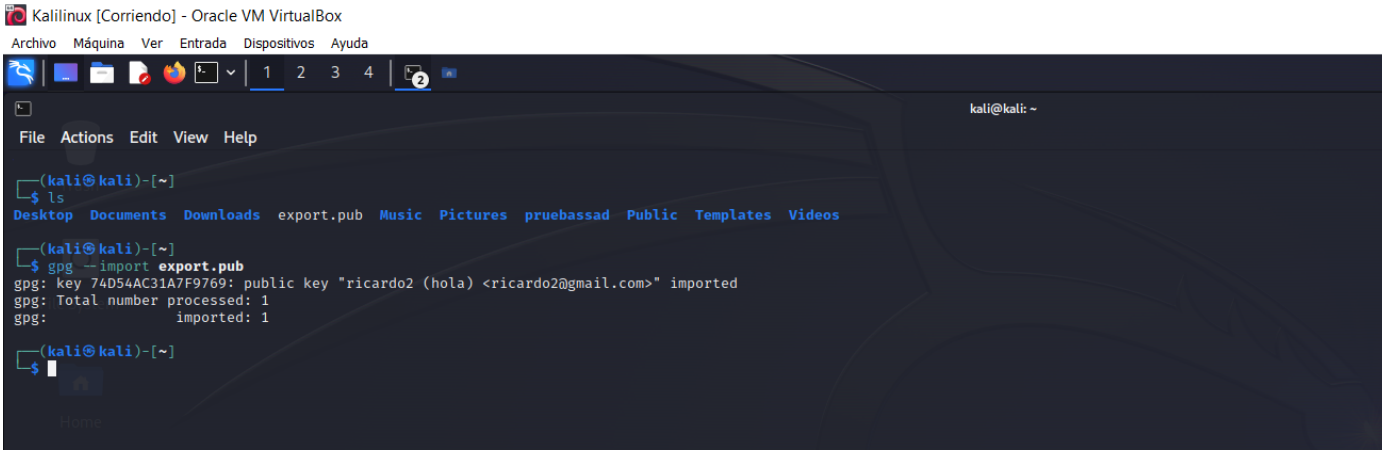


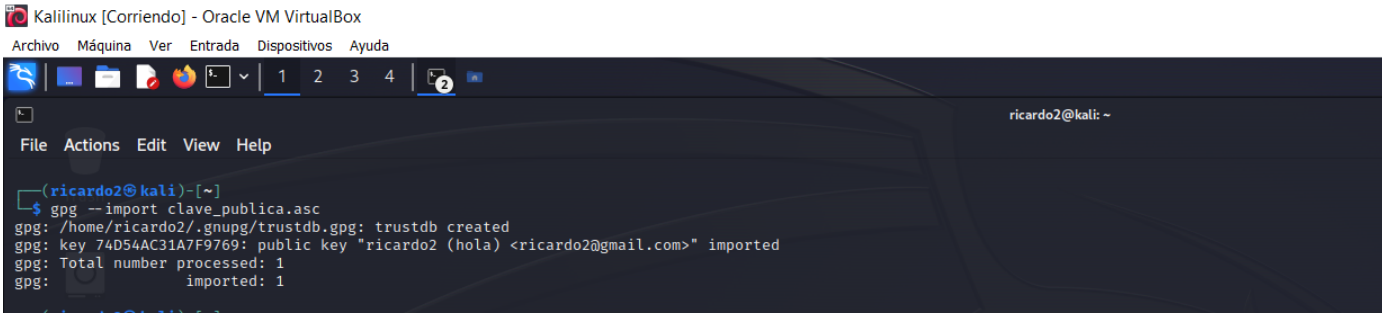
Ahora copiaremos las claves públicas de un usuario a otro. Clave publica de ricardo2 a kali.

Clave publica de kali a ricardo2.



Ahora importaremos las claves copiadas a cada usuario.

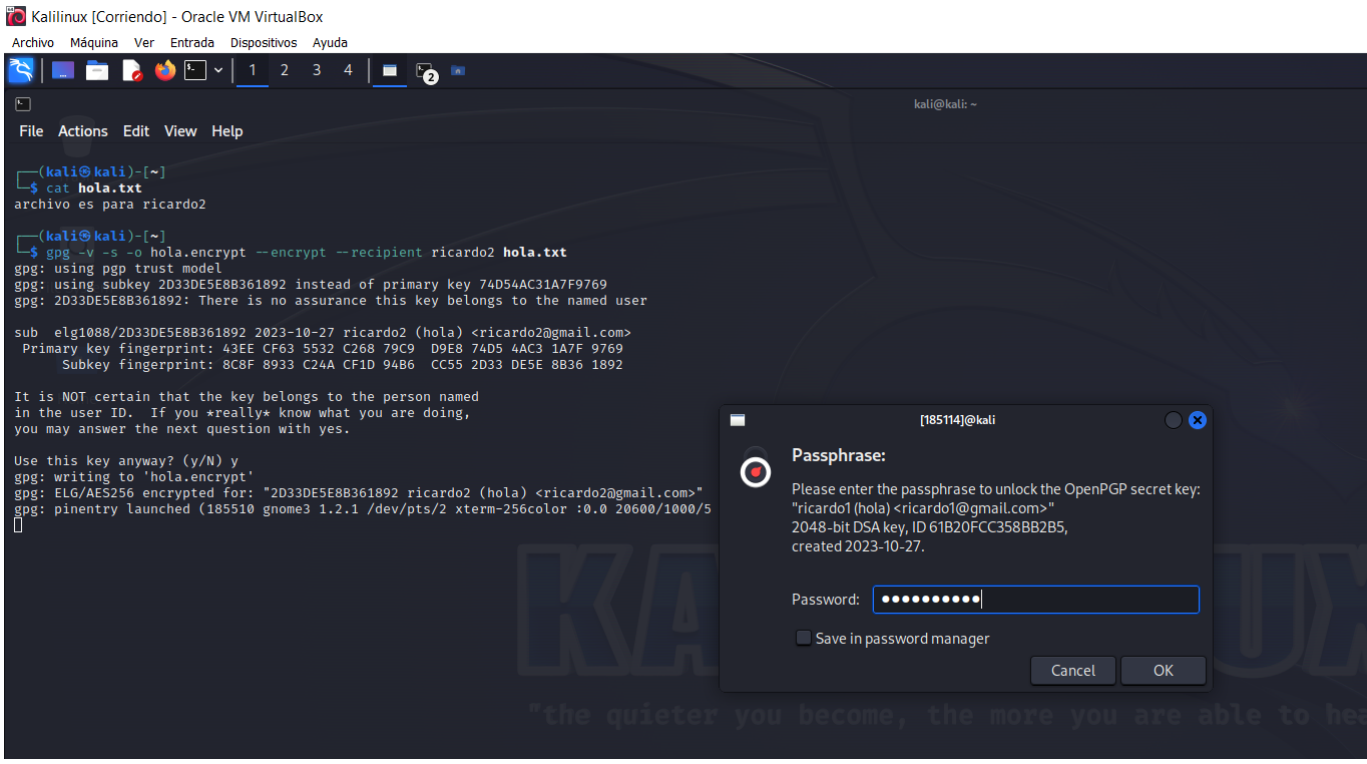




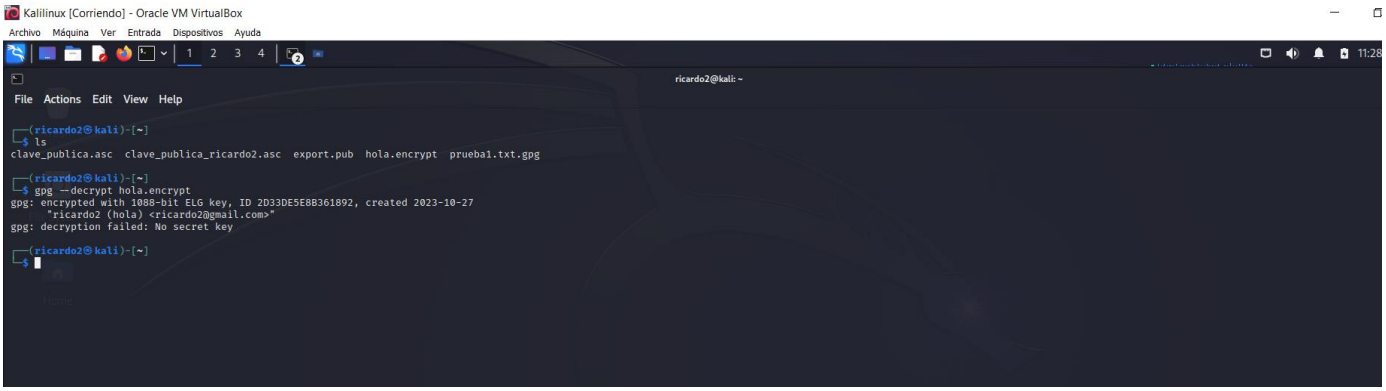
Ahora creamos un archivo nuevo en la cuenta kali y lo encriptaremos con la clavepublica del usuario ricardo2 que acabamos de importar.

Lo encriptamos con la clave publica de ricardo2.

$ gpg -v -s -o hola.encrypt --encrypt --recipient franky hola.txt



Pasamos el archivo a ricardo2 y lo desciframos con su clave privada (lo podemos realizar ya que elarchivo lo ciframos con la clave publica de ricardo2).

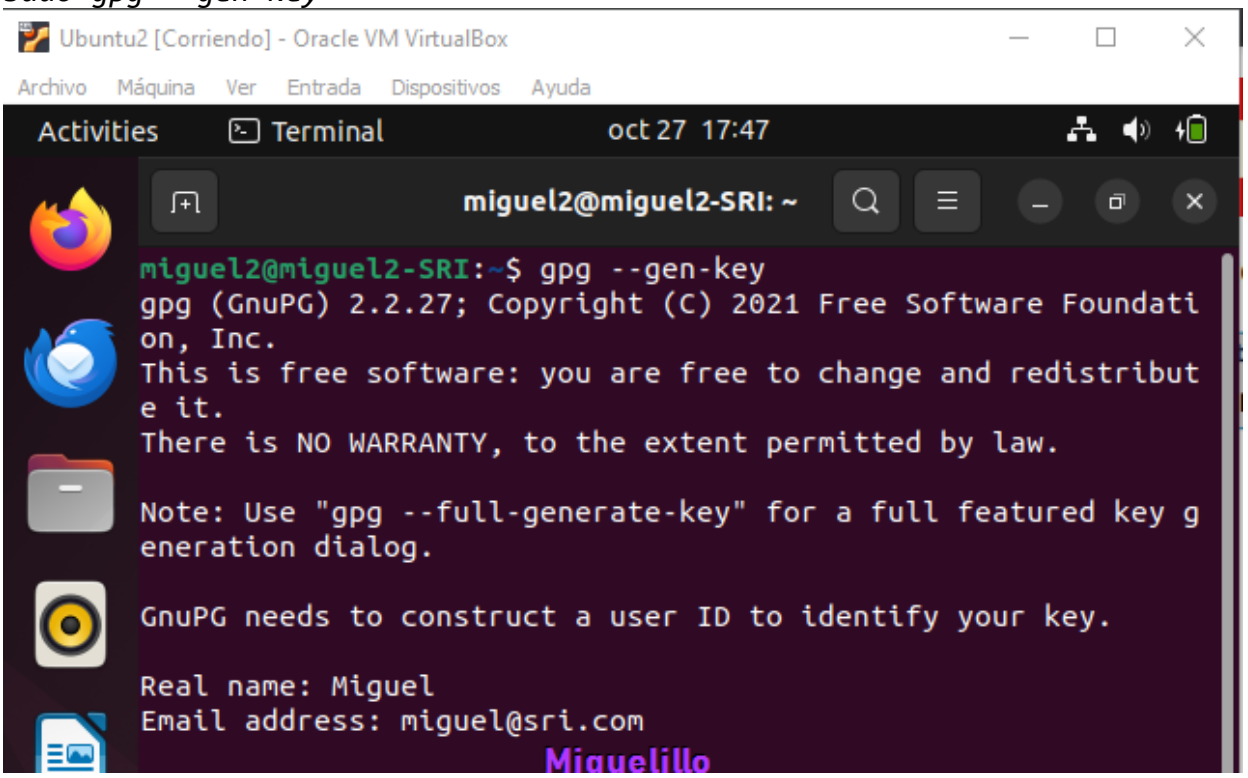


- 3. Utiliza un programa en Windows o GNU/Linux para simular la autenticidad mediante “cifrado asimétrico”.

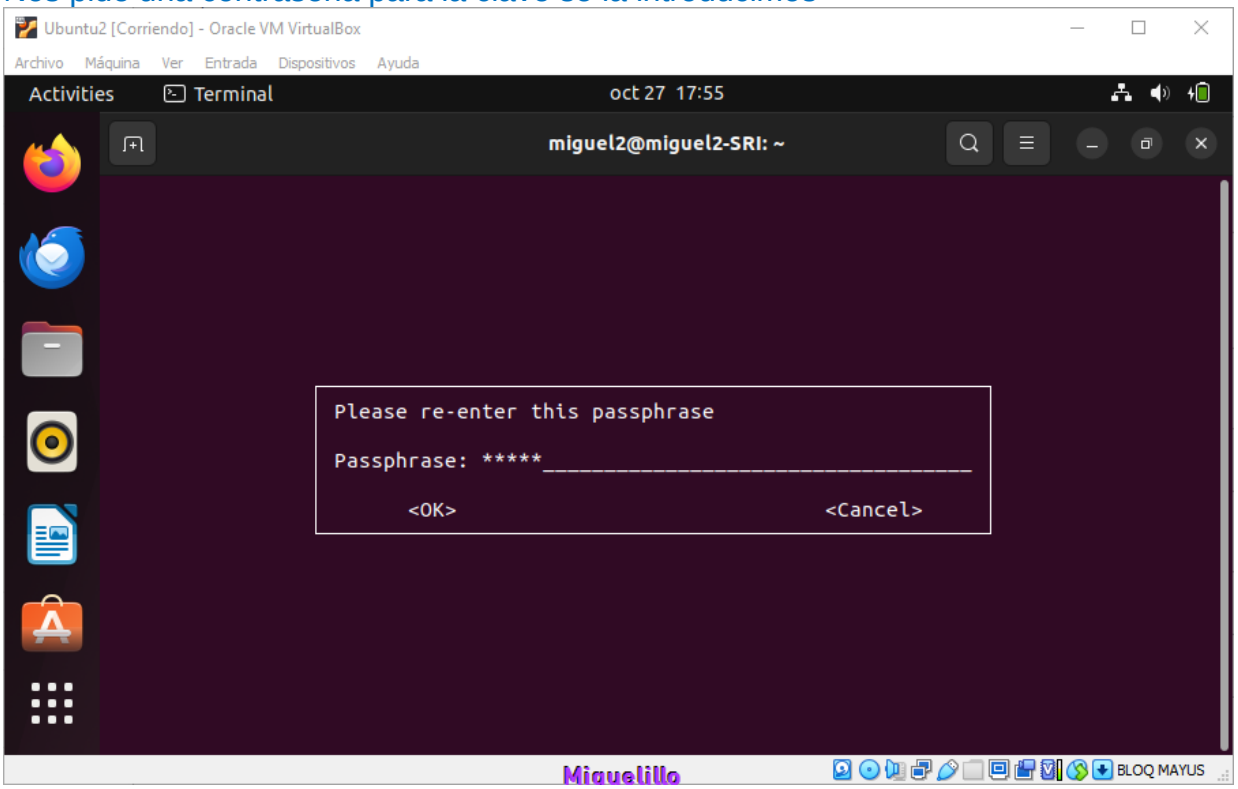
Para asegurar autenticidad usaré una una firma digital si el documento luego es modificado de cualquier forma, la verificación de la firma fallará

Para eso voy a utilizar gpg

Lo primero que haremos será generar un par de claves pública y privada lo haremos con *sudo gpg --gen-key*

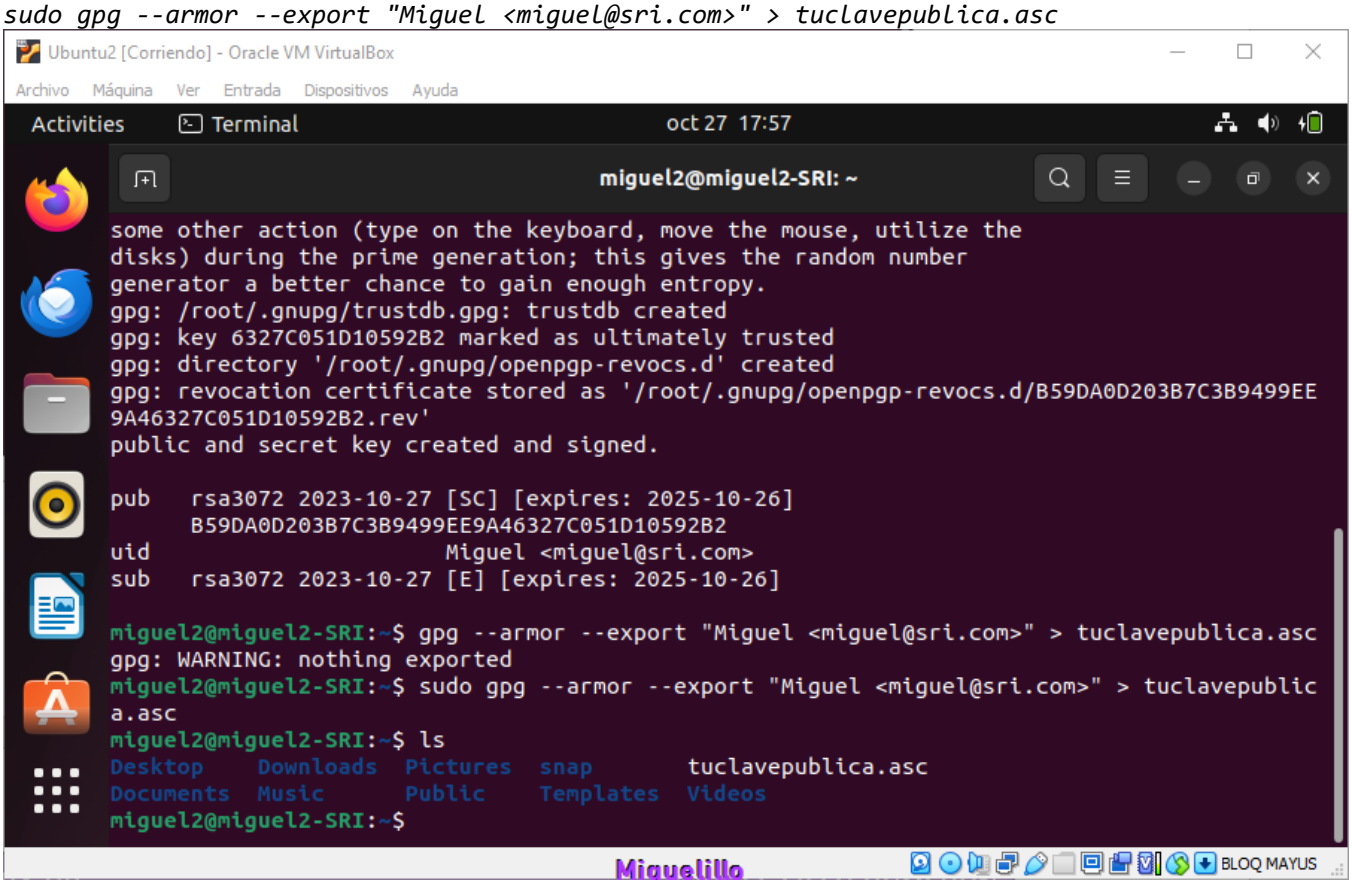


Nos pide una contraseña para la clave se la introducimos



Exportamos la clave pública con el comando

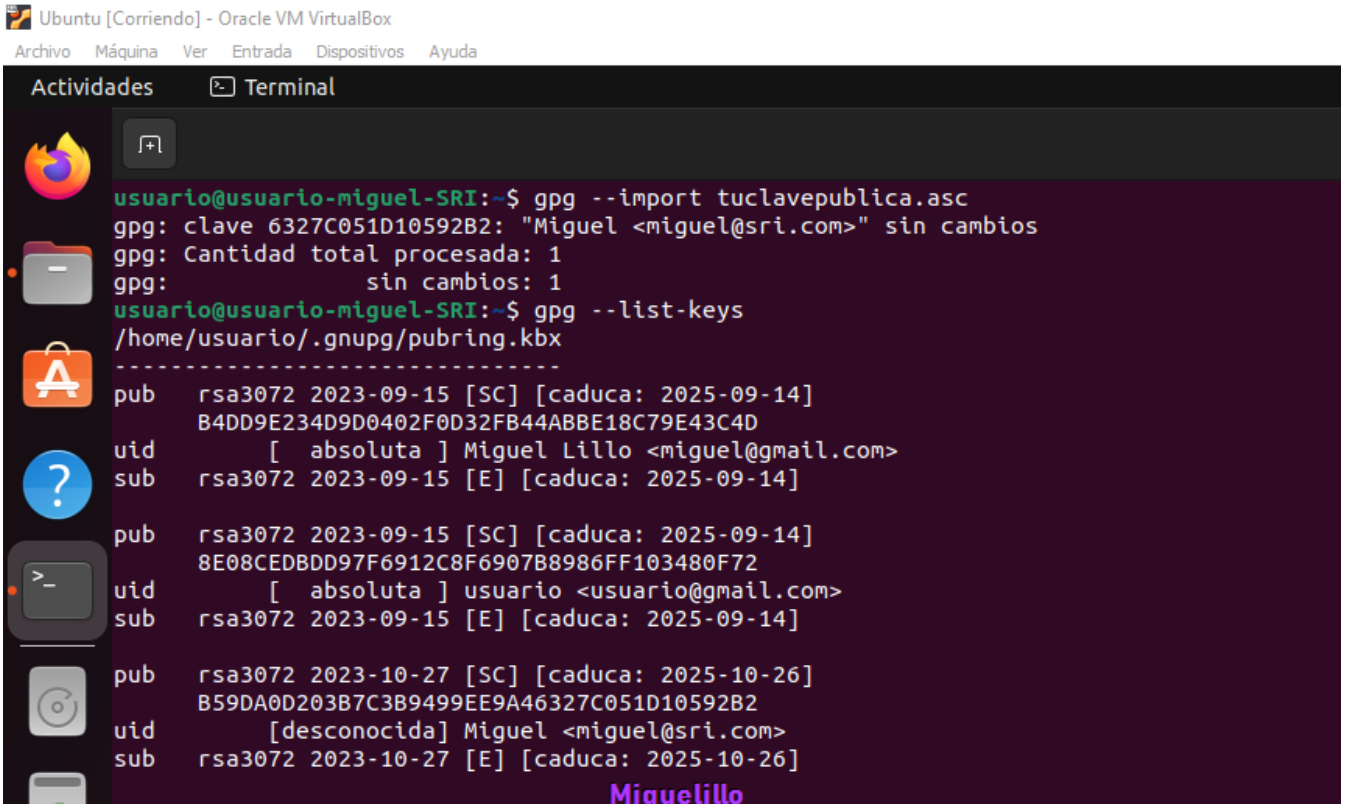
sudo gpg --armor --export "Miguel <miguel@sri.com>" > tuclavepublica.asc



Importamos la clave publica al otro equipo con el comando

gpg --import tuclavepublica.asc

gpg --list-keys



Para ver las claves públicas que tenesmos disponibles ponemos el siguiente comando

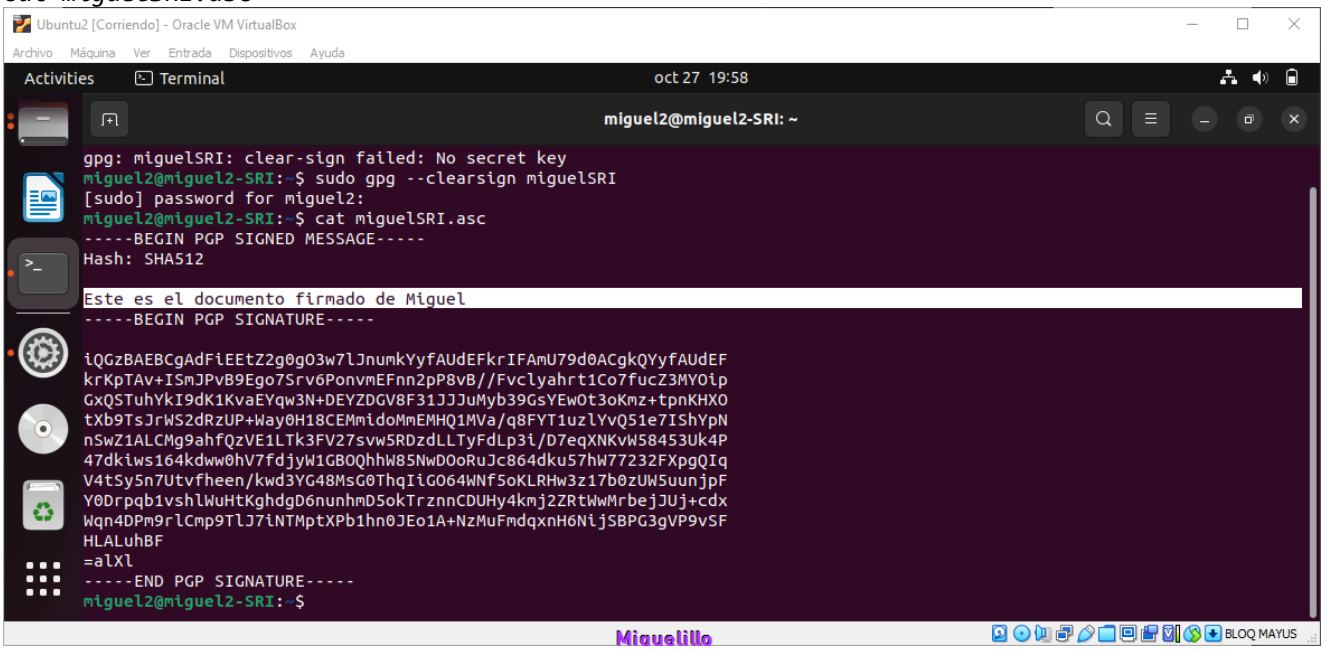
sudo gpg --list-secret-keys

Firmo el archivo indicando el nombre y correo, el nombre del archivo y del resultado con el siguiente comando

sudo gpg --clearsign miguelSRI

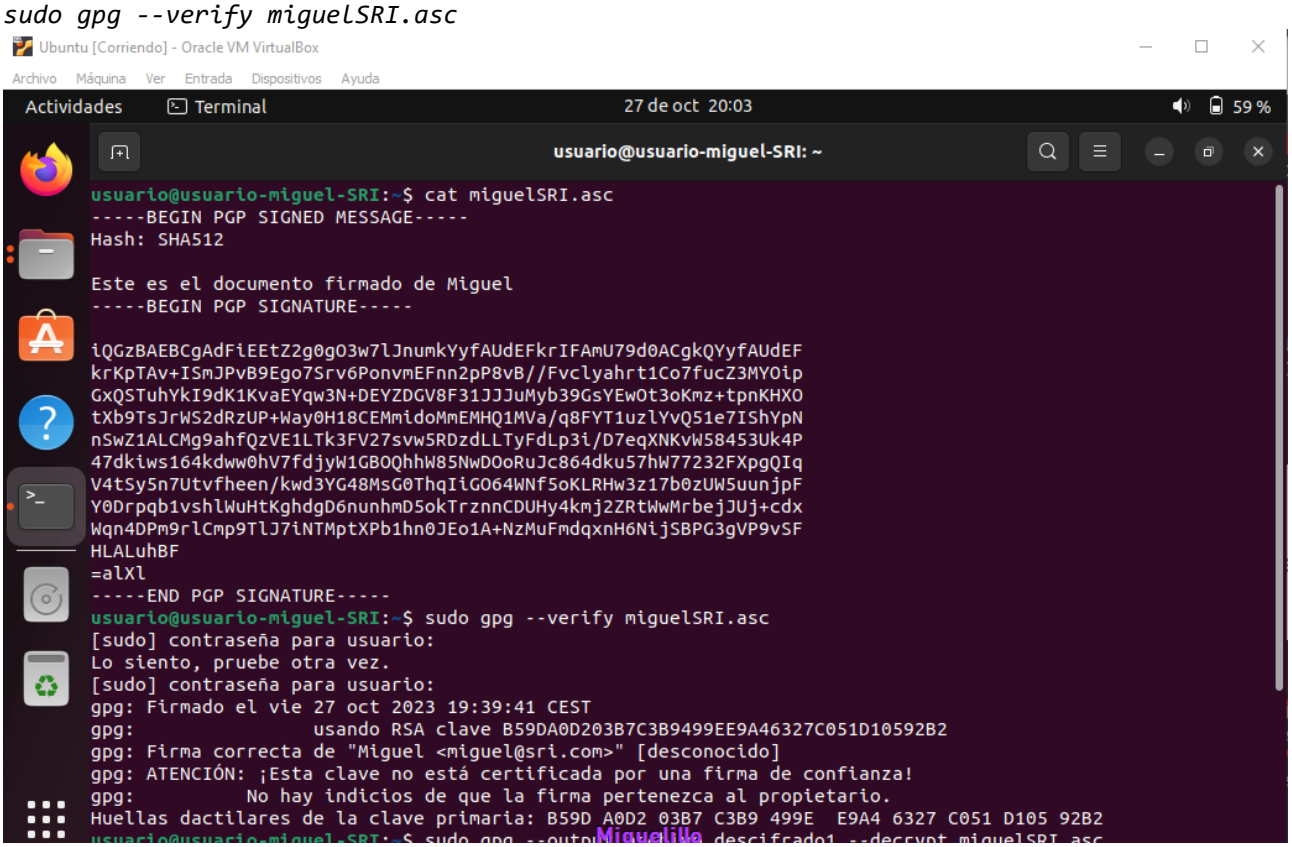
Se crea un archivo con la extensión .asc ara ver el contenido y que se ha firmado

cat miguelSRI.asc



Para ver la clave que lo ha firmado y el remitente usamos el siguiente comando:

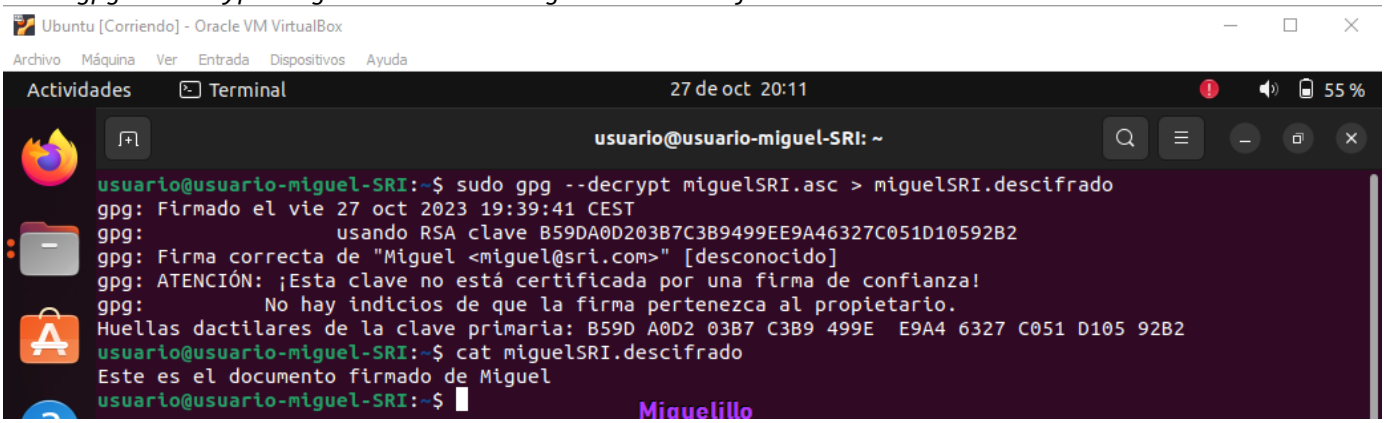
sudo gpg --verify miguelSRI.asc



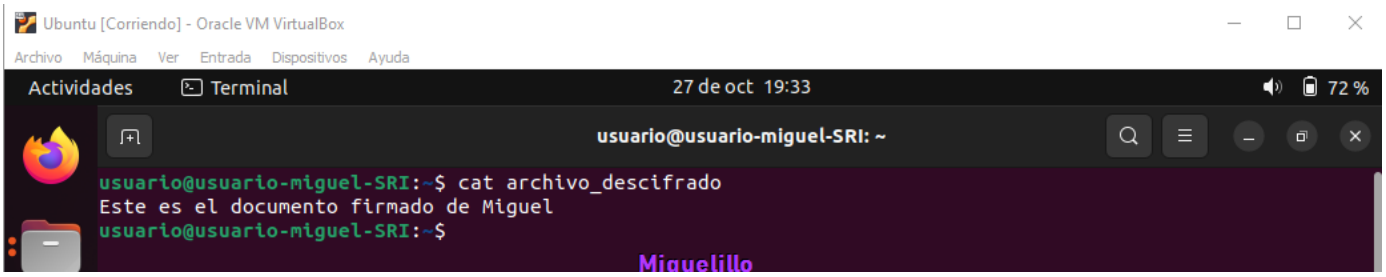
Aparece como firma correcta, la clave no es de confianza ya qu no es de un distribuidor de claves de confianza ya que la he creado yo

Para ver el contenido lo desencriptamos

sudo gpg --decrypt miguelSRI.asc > miguelSRI.descifrado

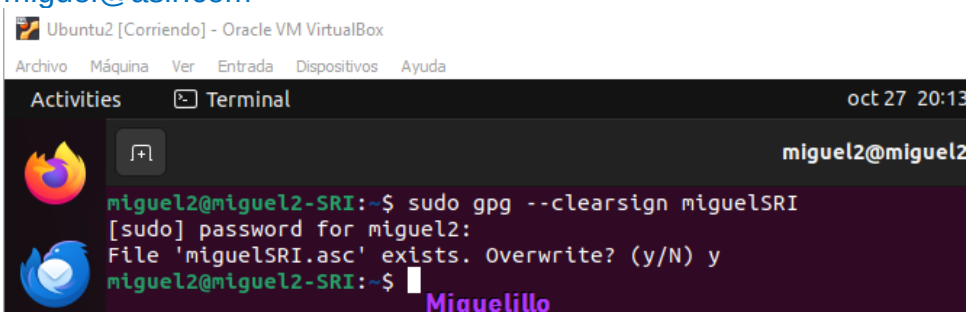


Con un Cat veo el contenido

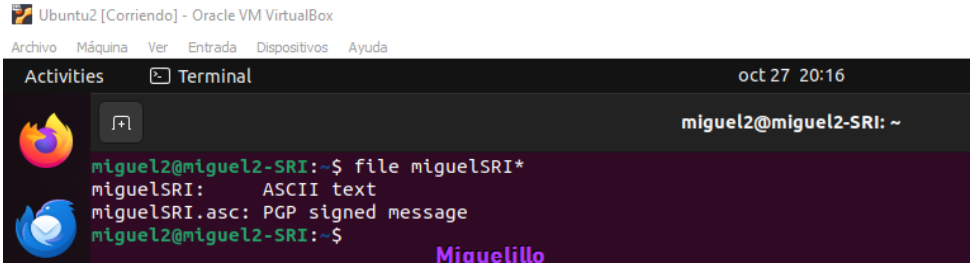


4. Utiliza un programa en Windows o GNU/Linux para simularla "autenticidad"+"confidencialidad" mediante "cifrado asimétrico".

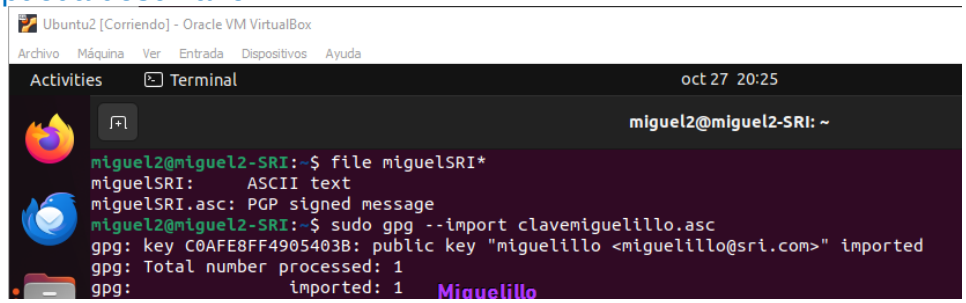
Para autenticidad + confidencialidad primero firmamos con la clave privada de Miguel miguel@asir.com



Verificamos que se ha creado bien

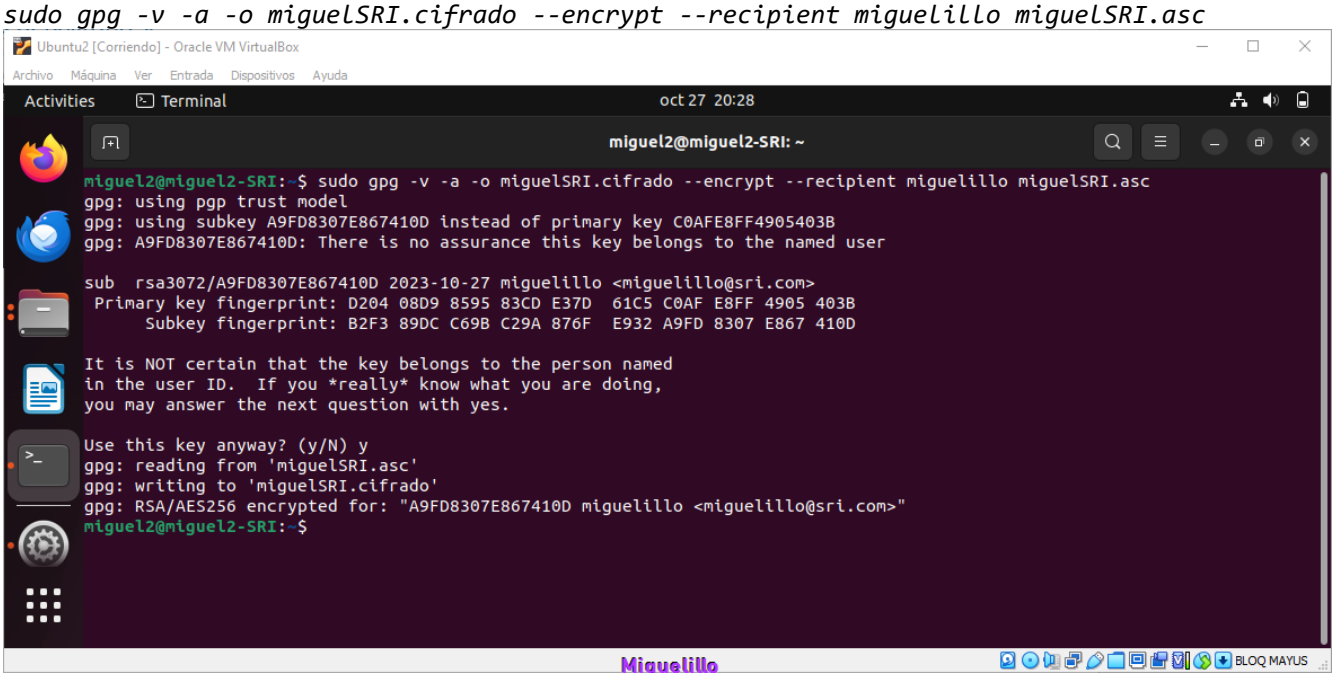


Ahora ciframos el documento firmado con la clave publica de la otra máquina para que este pueda descifrarlo.

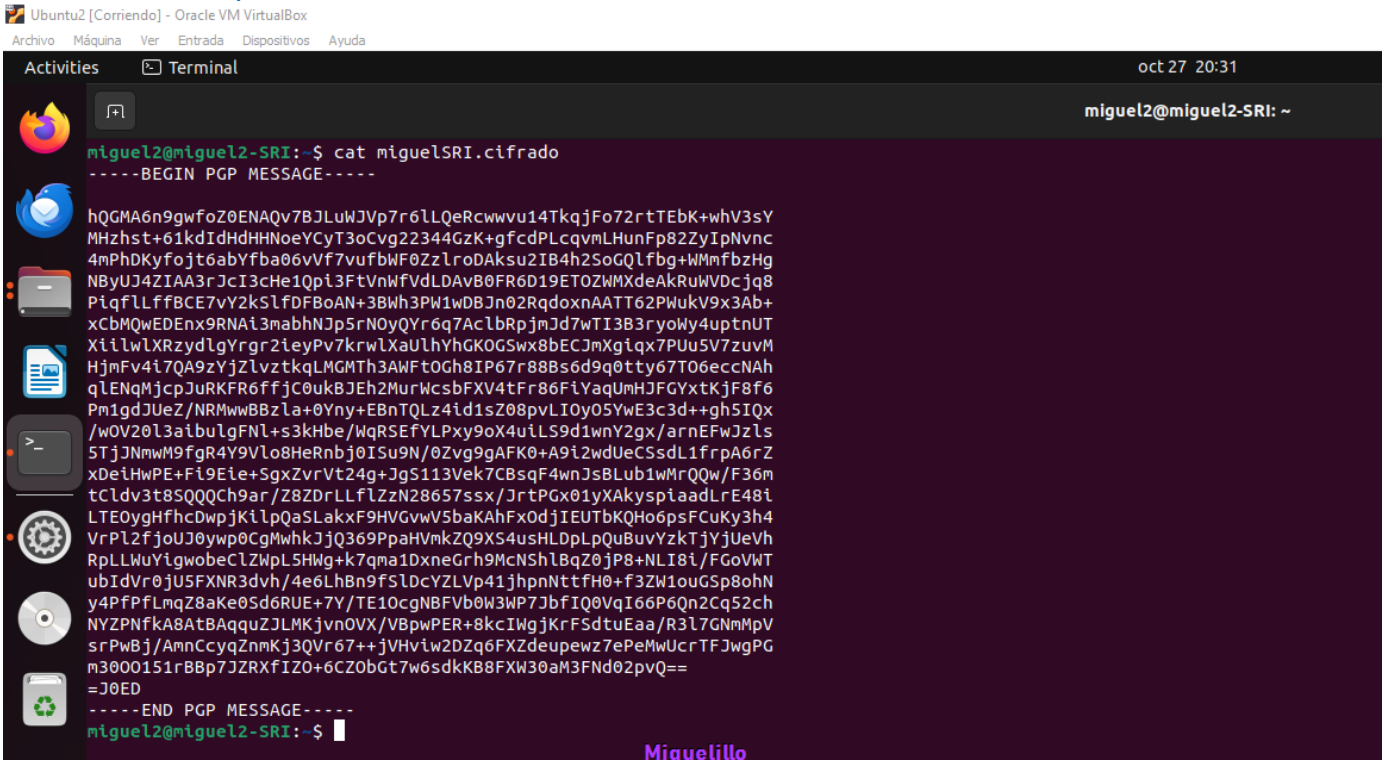


Lo ciframos con

sudo gpg -v -a -o miguelSRI.cifrado --encrypt --recipient miguelillo miguelSRI.asc

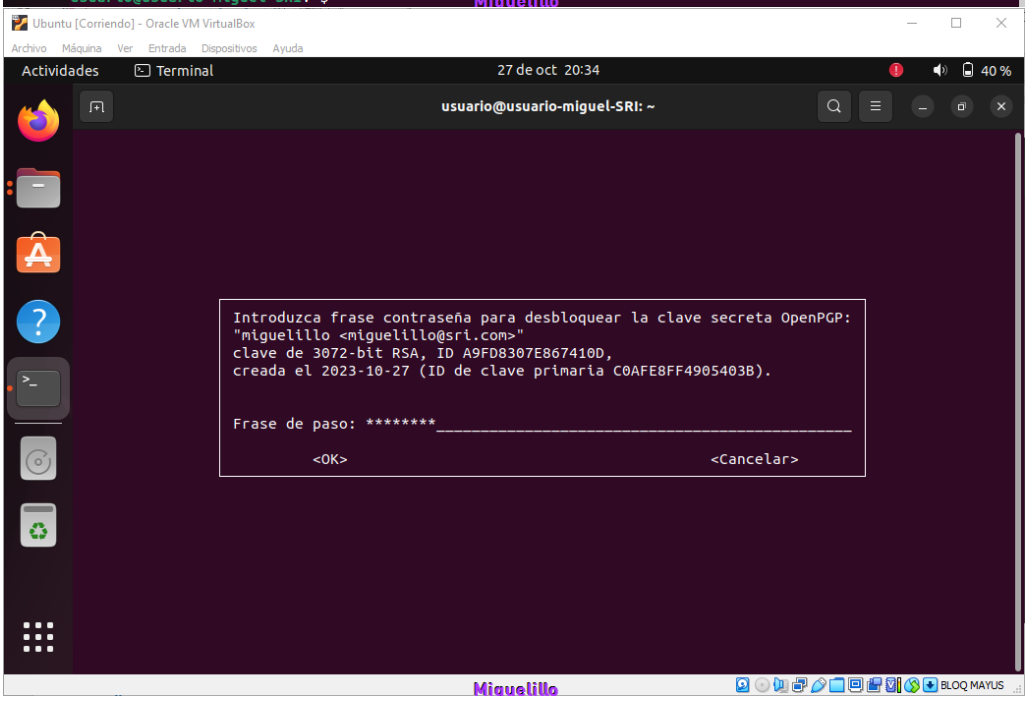


Vemos con cat que se ha cifrado

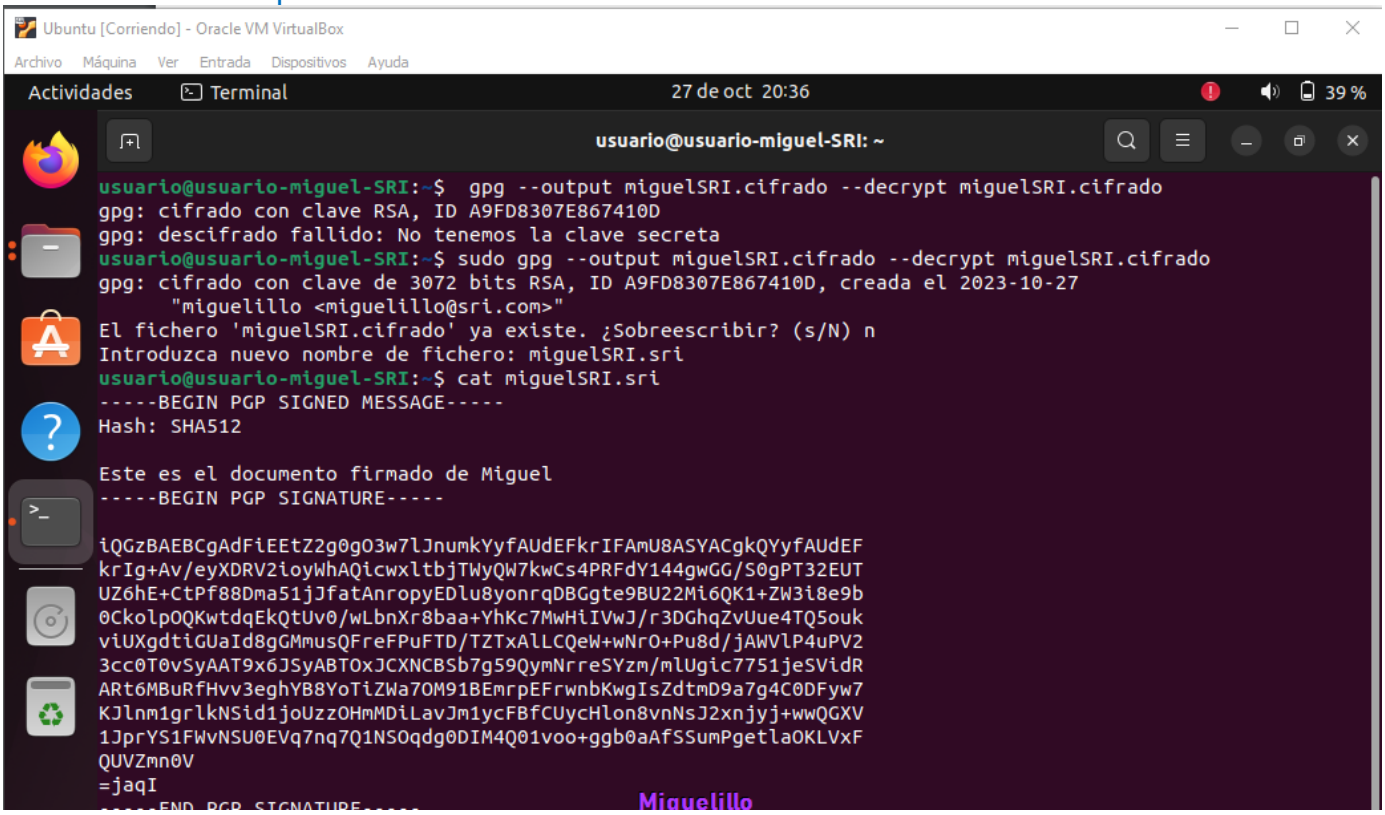


Pasamos el archivo al otro equipo y lo desencriptamos

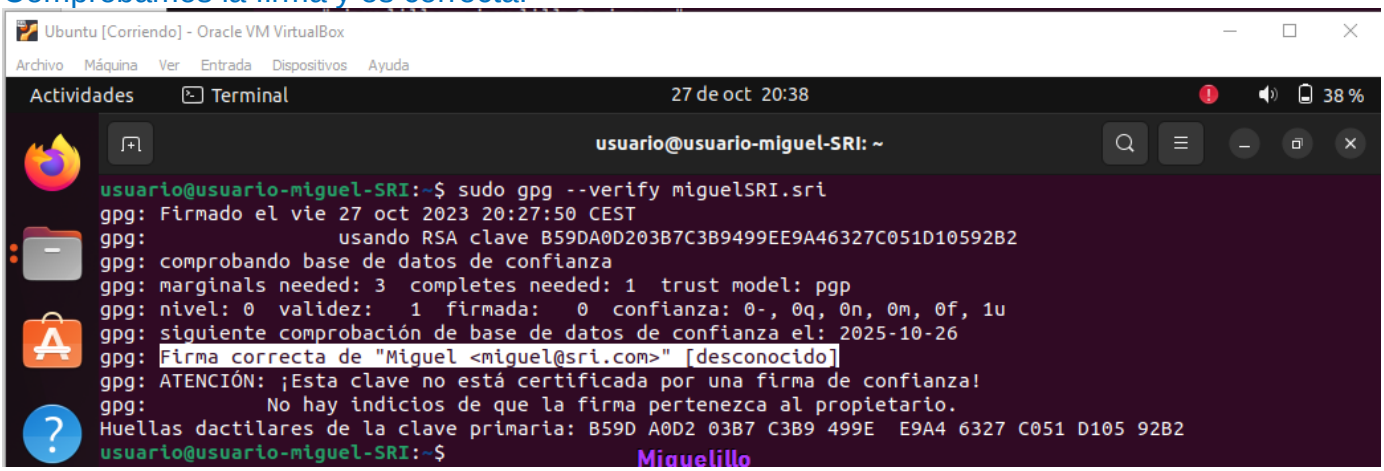
gpg --output autenticidad.desencriptado --decrypt autenticidad.encrypt



Si hacemos un cat podemos ver el contenido

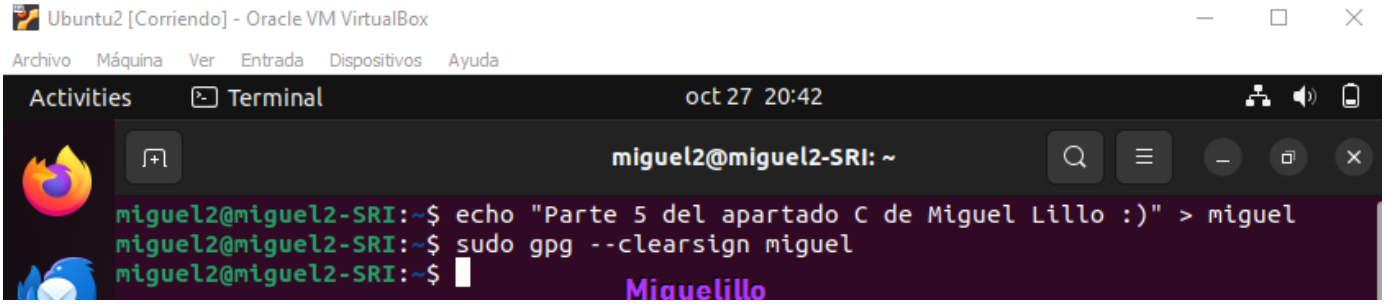


Comprobamos la firma y es correcta.

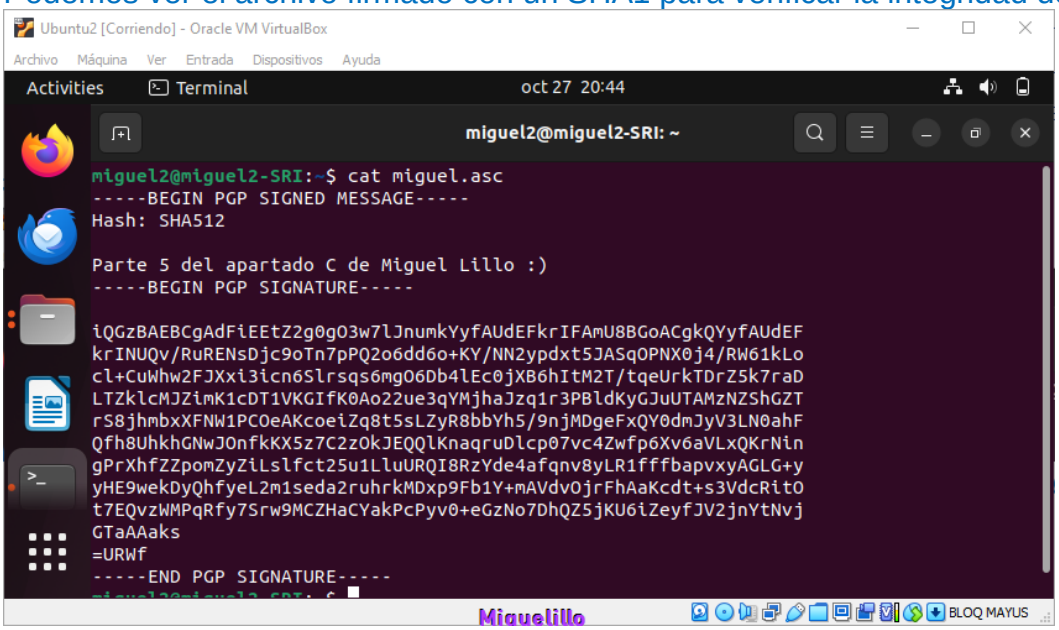


5. Utiliza un programa en Windows o GNU/Linux para simular una comunicación segura utilizando cifrados híbridos “autenticidad” + “confidencialidad”+”integridad”: asimétricos (clave pública) y simétricos (clave privada).

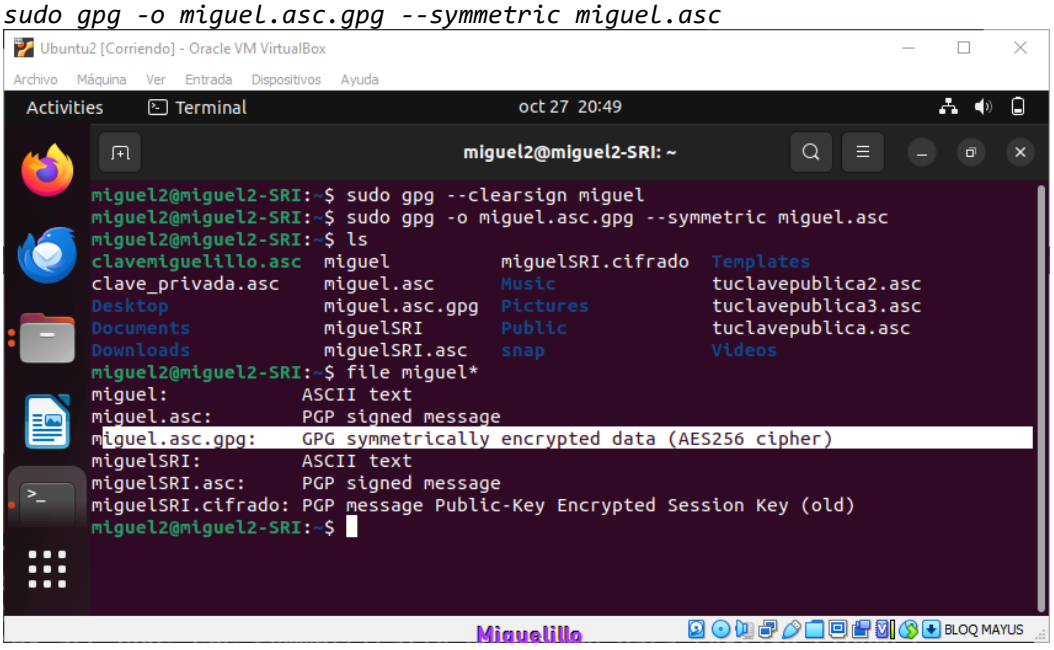
Creamos un archivo y lo firmamos



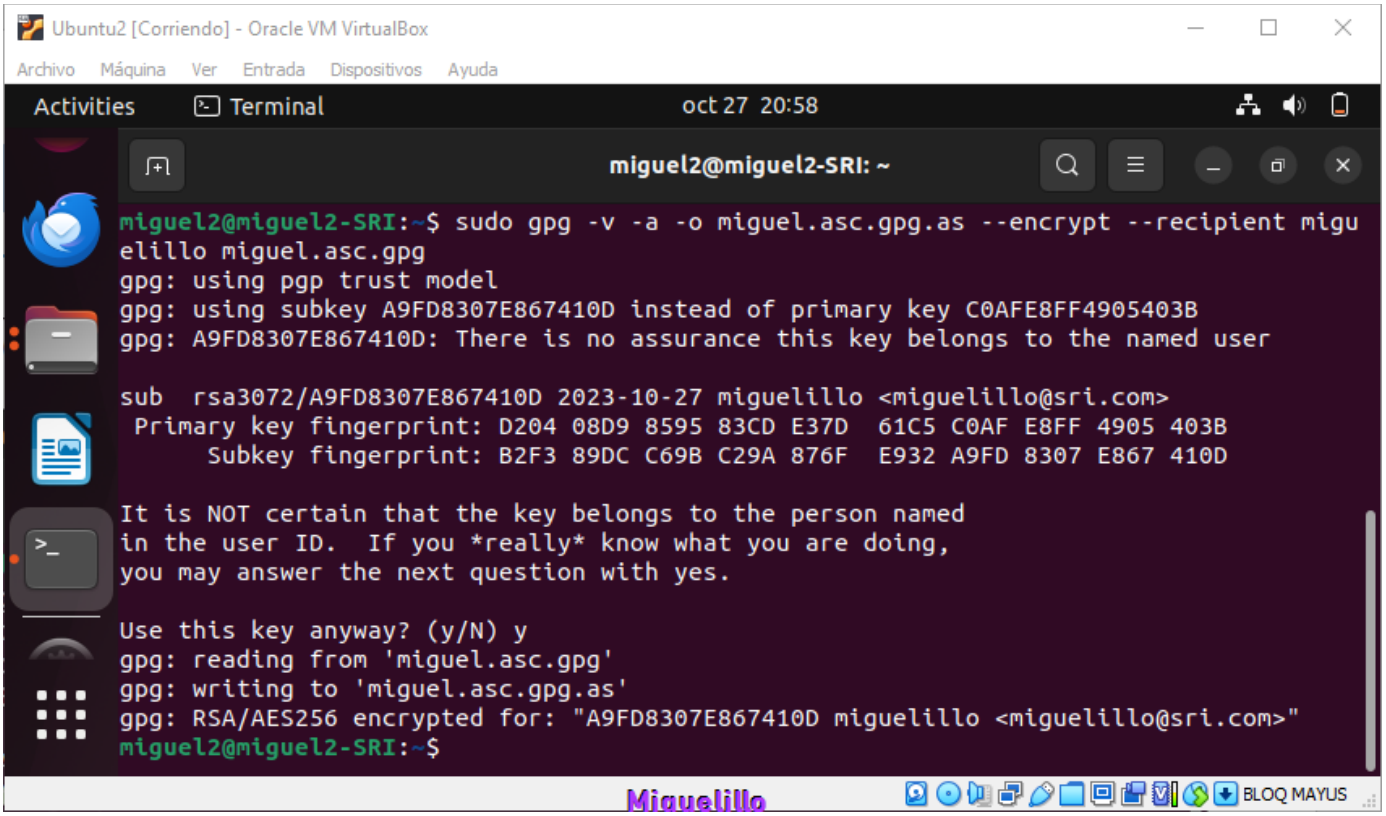
Podemos ver el archivo firmado con un SHA1 para verificar la integridad del contenido y la firma



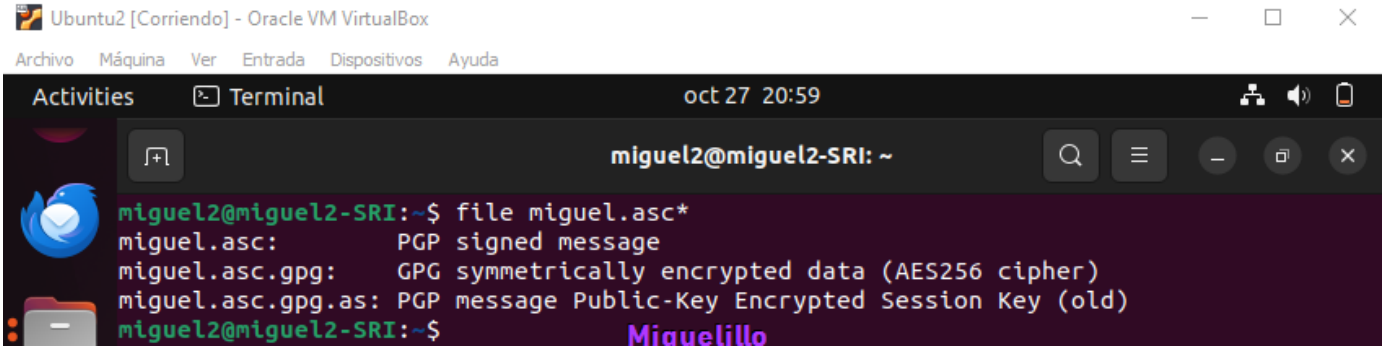
Lo siguiente es hacer un cifrado simétrico para que sea un cifrado híbrido. Con el comando:
sudo gpg -o miguel.asc.gpg --symmetric miguel.asc



Ahora con las claves haremos un cifrado asimétricon con la clase pública de migueillo con el comando:

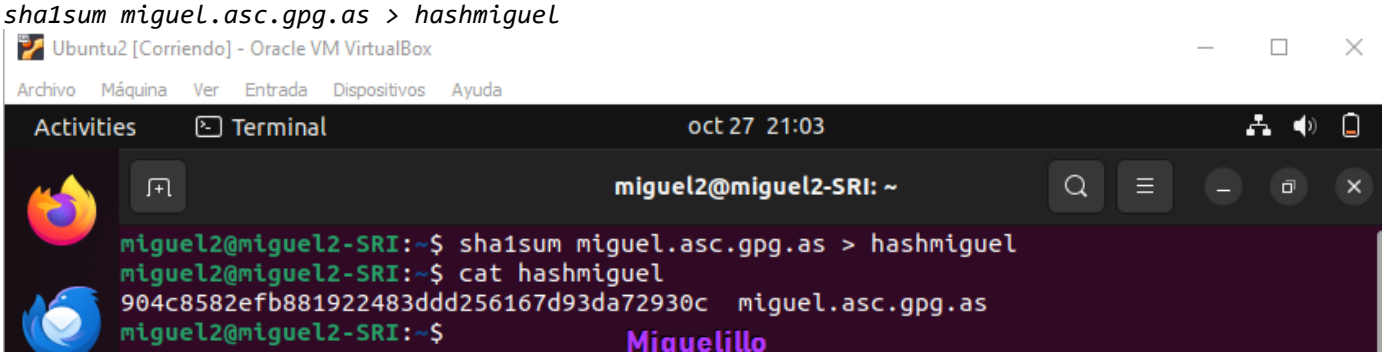


Revisamso el tipo de archivo



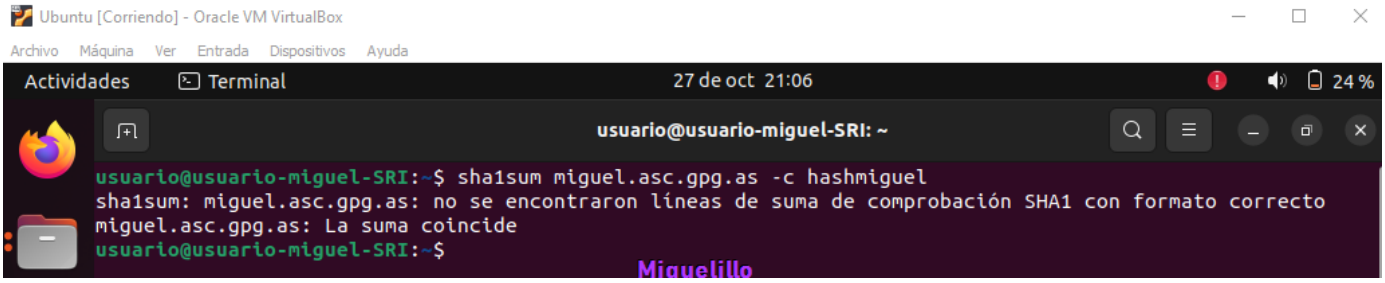
Generamos el código hash del archivo y pasamos los dos archivos al otro equipo

sha1sum miguel.asc.gpg.as > hashmiguel

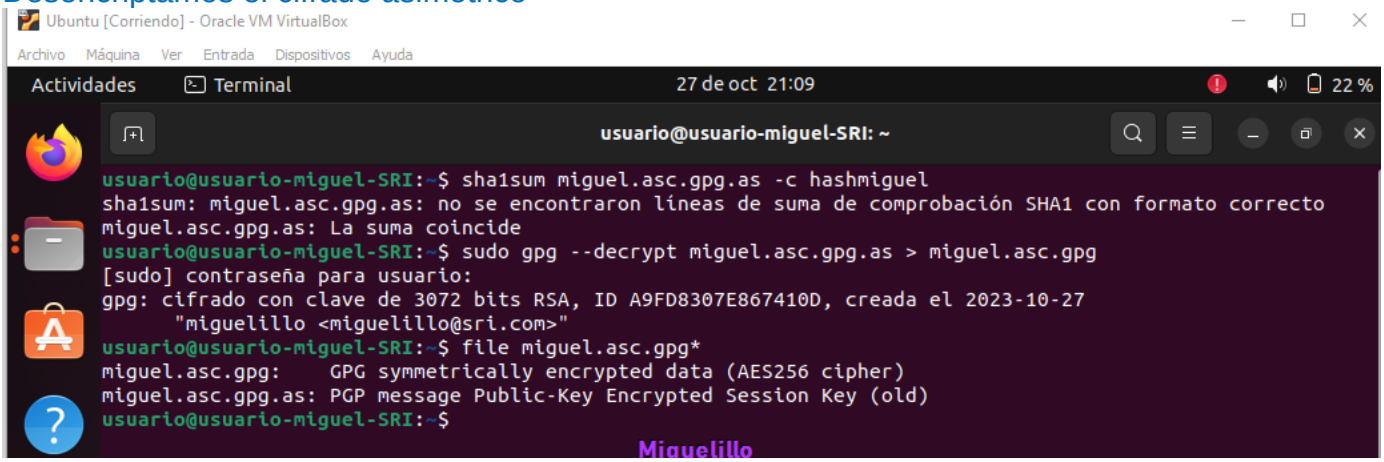


Una vez en este equipo comparamos el hash para comprobar que no ha sido alterado el archivo con el comando

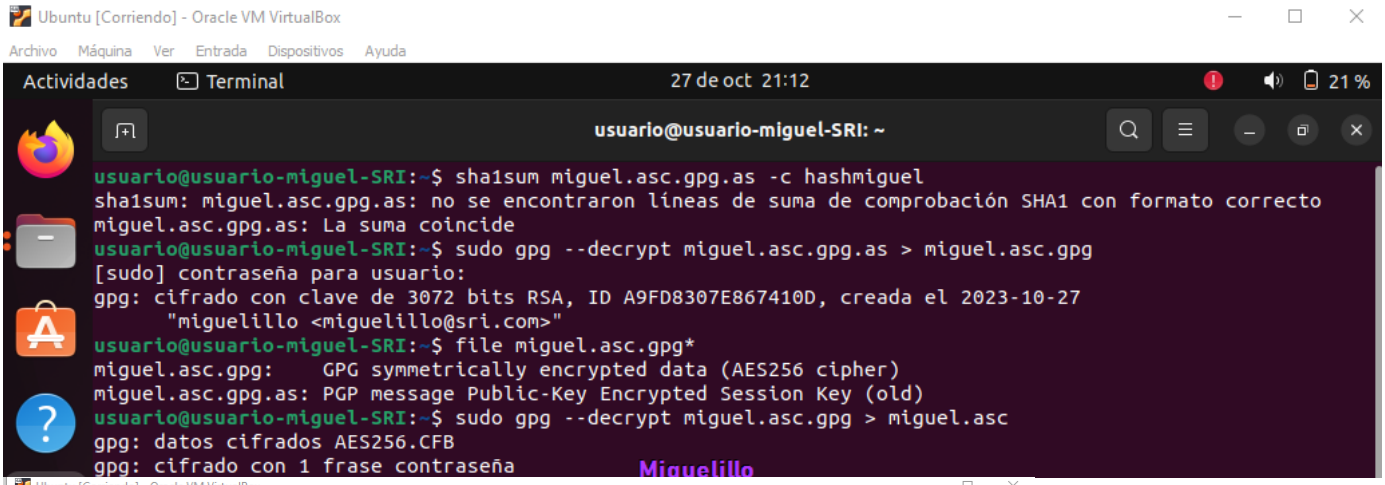
sha1sum miguel.asc.gpg.as -c hashmiguel

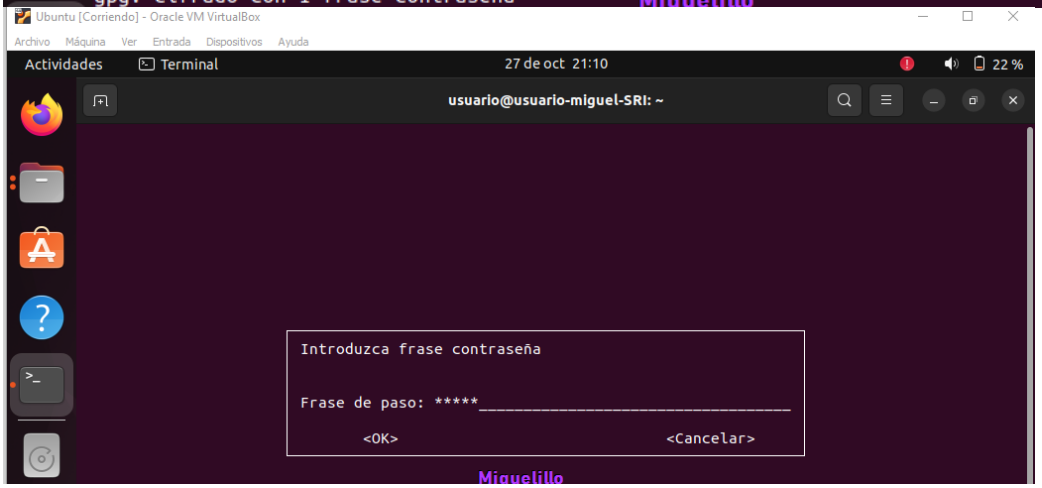


Desencriptamos el cifrado asimétrico

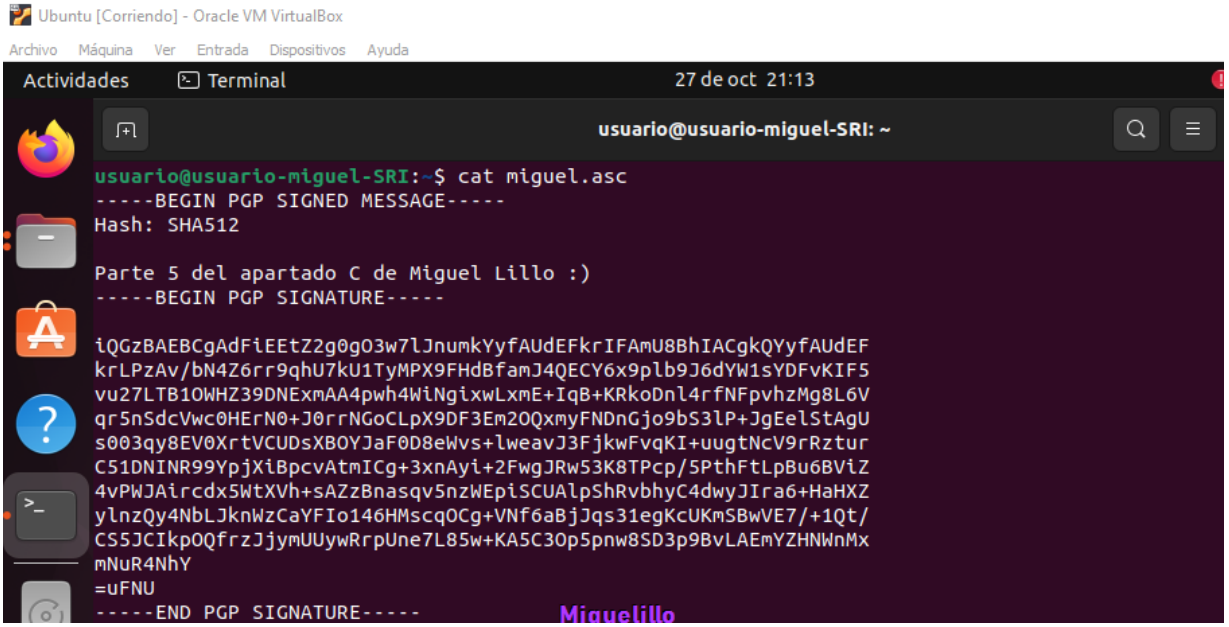


Desencriptamos de la misma manera anterior el cifrado simétrico

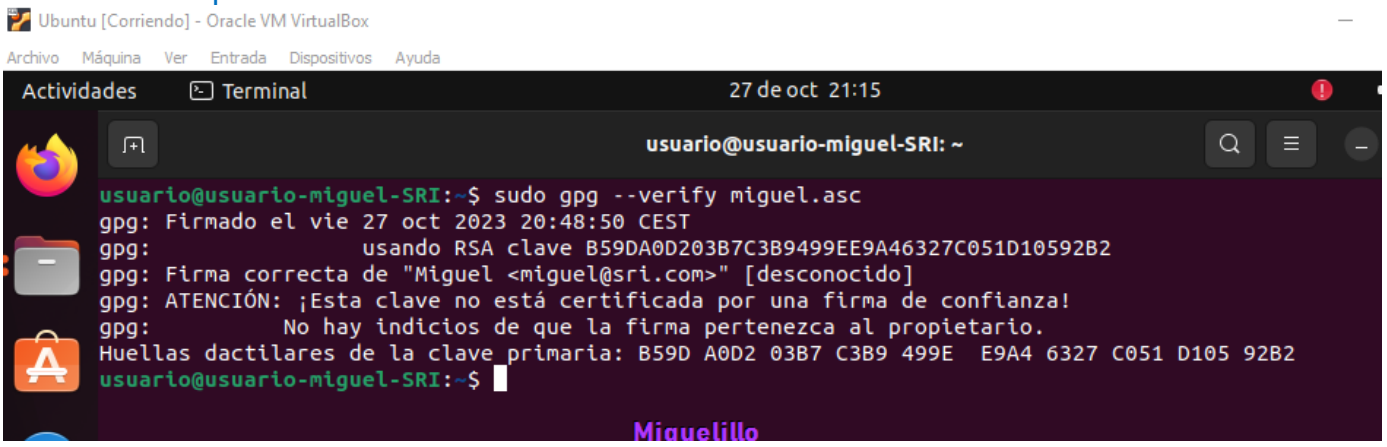




Podemos ver el SHA1 para compararlo con la firma y verificar la autenticidad de la firma



Podemos ver que la firma es válida



También podemos visualizar el mensaje