# UT01: Adopción de pautas de seguridad informática – 3) – Técnicas De Cifrado.

Nombre: Francisco Jesús García – Uceda Díaz

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.

Link de cómo realizarlo (Cisco).

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.

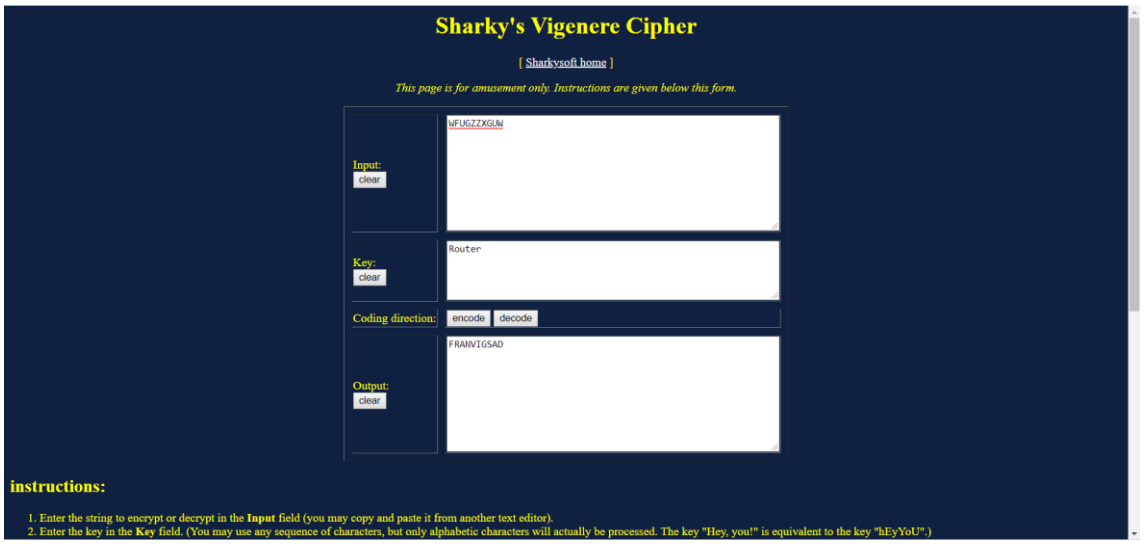
Encriptación

| 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

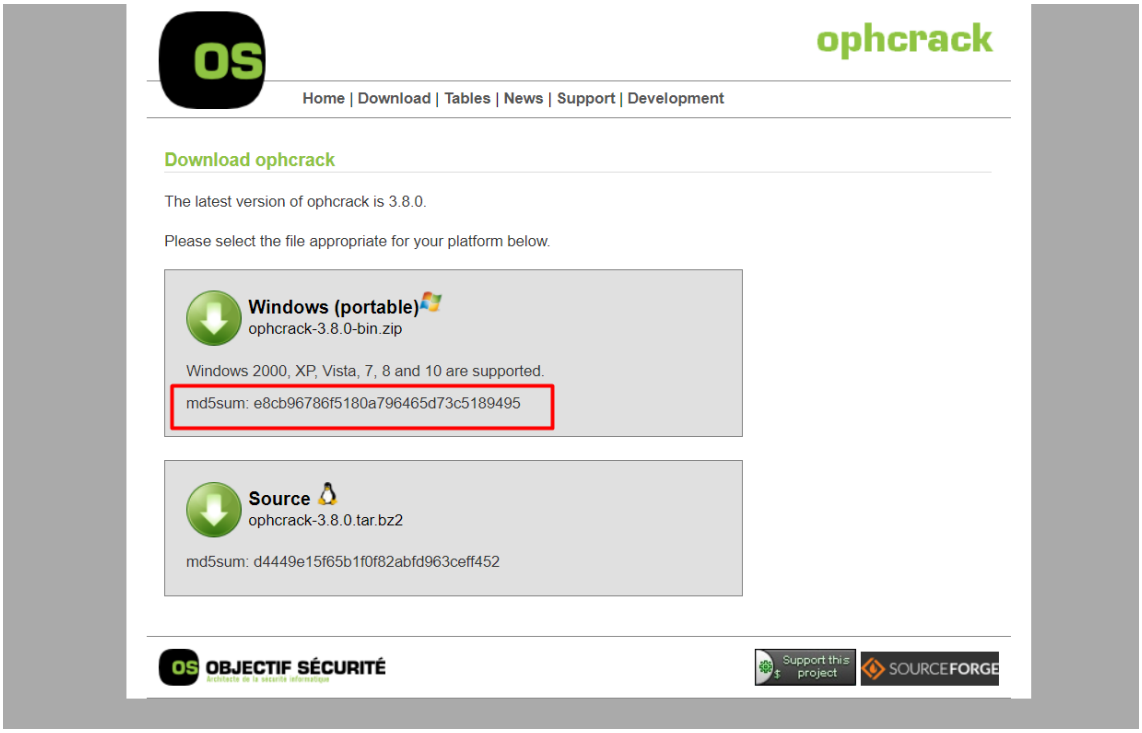

## b) CIFRADO: Funciones HASH--> Integridad y Autenticidad.

- 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.

Lo realizaré con PowerShell porque siempre viene instalado en los sistemas operativos Windows 7 en adelante la función que tenemos y no siempre podemos instalar un programa para comprobar la integridad de otro programa en MD5 o SHA1.

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:

Get-FileHash C:\ruta\al\archivo.iso -Algorithm MD5

También lo podremos hacer con multitud de algoritmos:

Get-FileHash C:\ruta\al\archivo.iso -Algorithm SHA1

Get-FileHash C:\ruta\al\archivo.iso -Algorithm SHA256

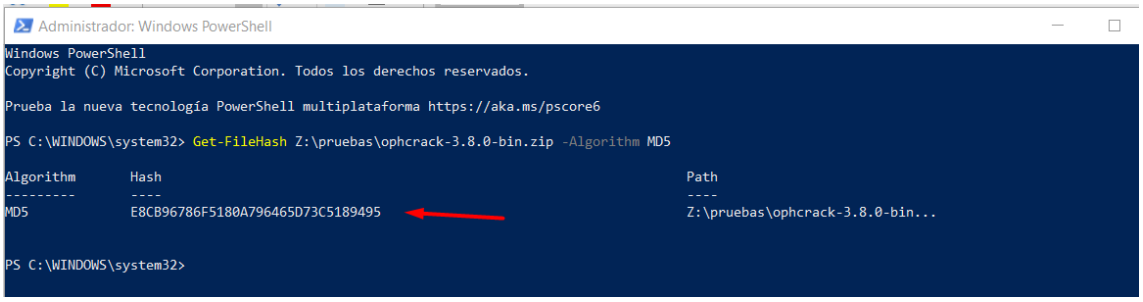
Get-FileHash C:\ruta\al\archivo.iso -Algorithm SHA384

Get-FileHash C:\ruta\al\archivo.iso -Algorithm SHA512

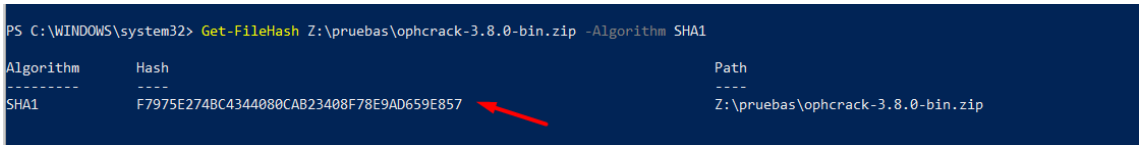
Get-FileHash C:\ruta\al\archivo.iso -Algorithm MACTripleDES

Get-FileHash C:\ruta\al\archivo.iso -Algorithm RIPEMD160

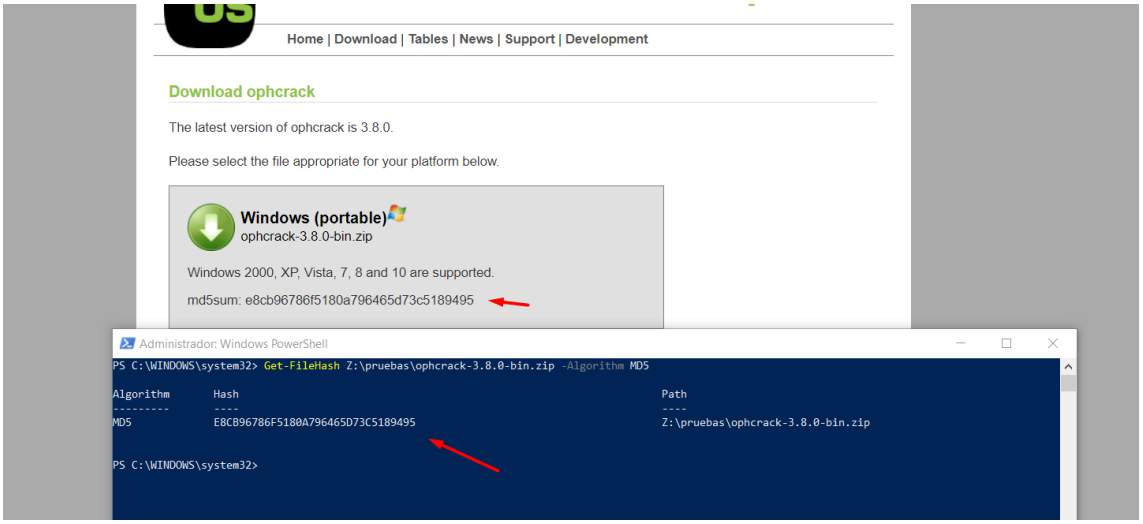
Vemos como nos genera un hash.




Podremos hacer lo mismo en SHA1.

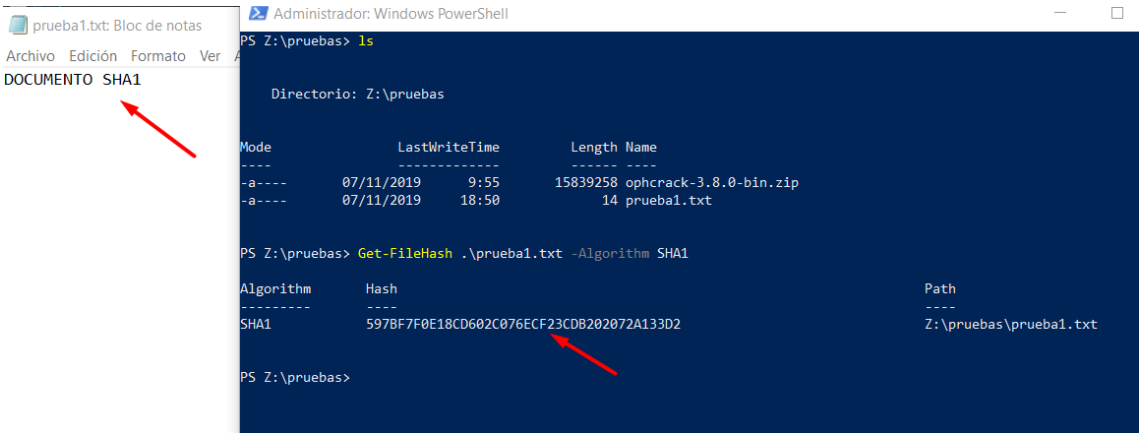



Ahora simplemente tenemos que comprar los dos hashes (el de PowerShell con el que ofrece la página Ophcrack. Vemos que son iguales por lo cual la integridad es correcta.

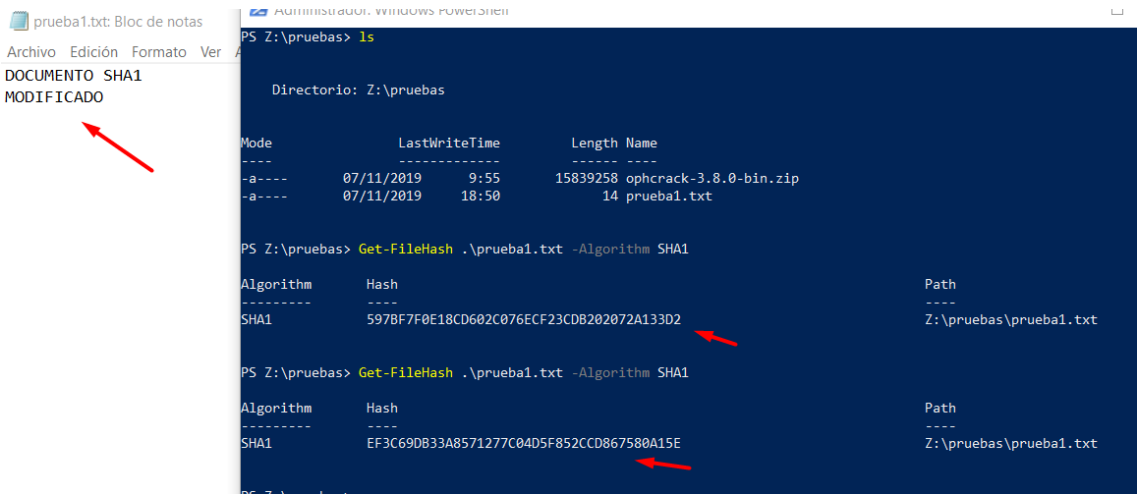



Para probar SHA1 al no encontrar programas que brinden en su página web el SHA1 para comprobar la integridad lo que haré es crear un archivo txt con algo escrito, generar su SHA1 y después modificarlo para que se vea cómo cambia.

Crearemos un documento de texto y generamos su SHA1.



Si editamos el contenido vemos cómo cambia el SHA1.

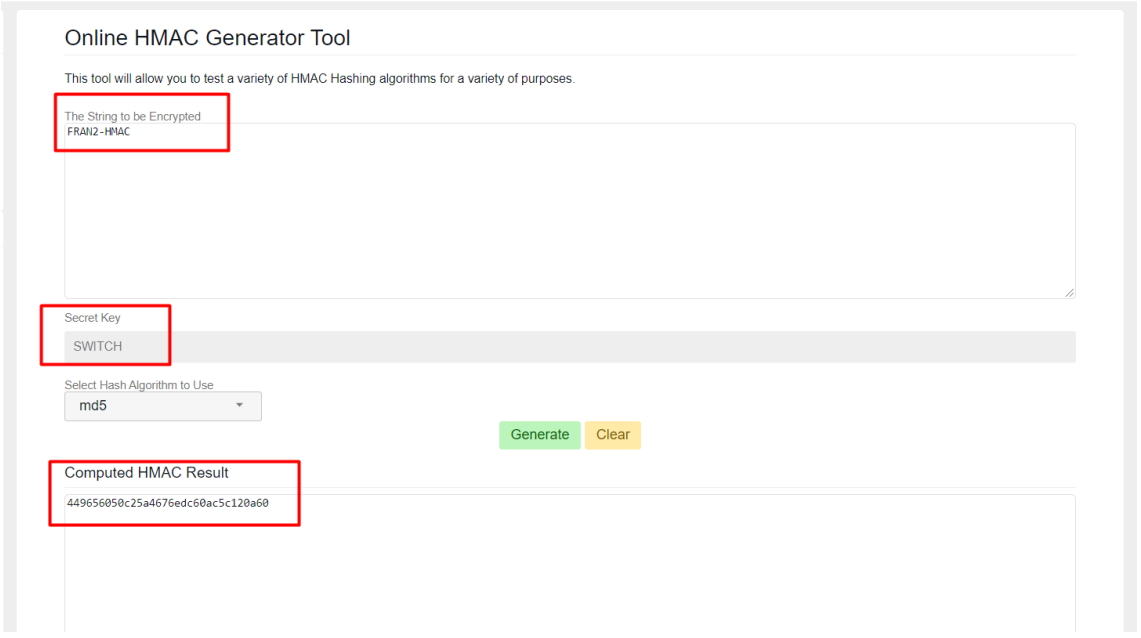



## 2. Utiliza un programa en Windows o GNU/Linux para simular la autenticidad utilizando HMAC-MD5, HMAC-SHA1.

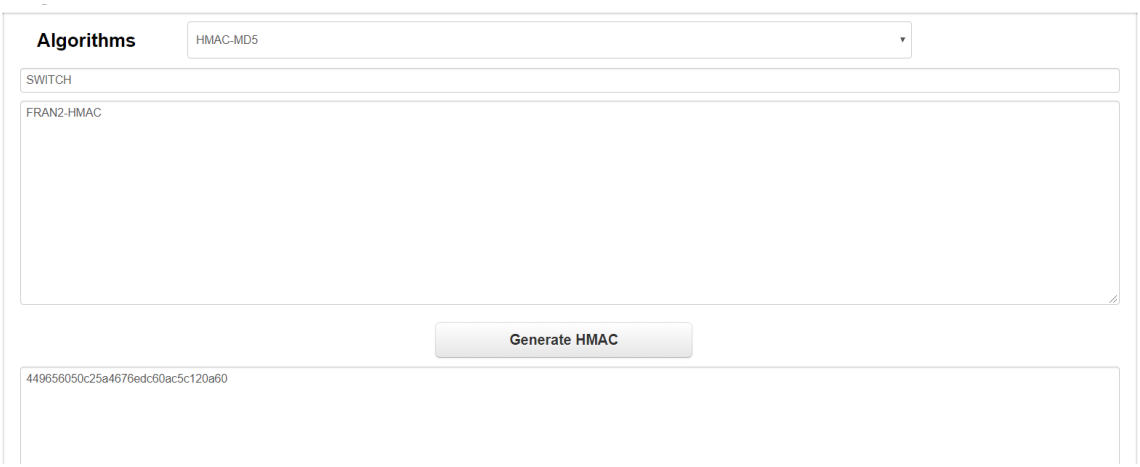
En términos criptográficos, es un tipo de código para autentificar mensajes en clave-hash (HMAC), es decir, es un diseño específico para generar un código de autentificación de mensaje (MAC) lo cual implica una función hash criptográfica junto con una llave criptográfica secreta.

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.

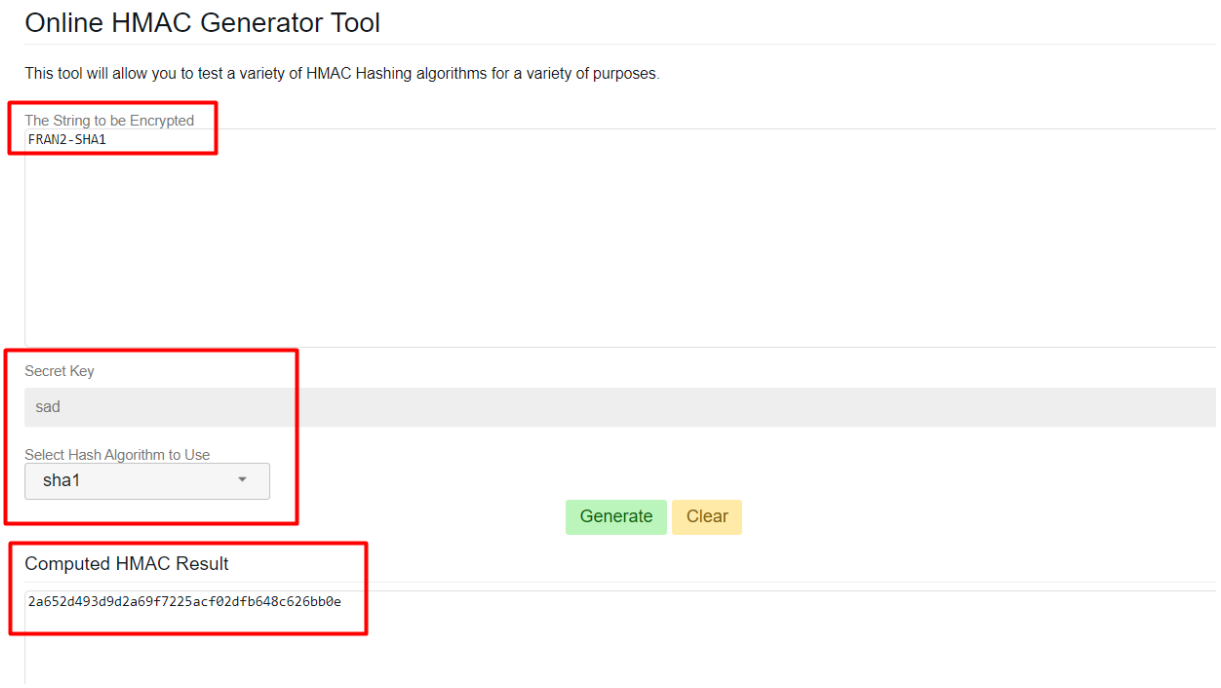
Utilizaré dos páginas web conocidas para realizar esta tarea. La primera es la siguiente (Link). Introduciremos un mensaje con su clave y generaremos el HMAC-MD5.



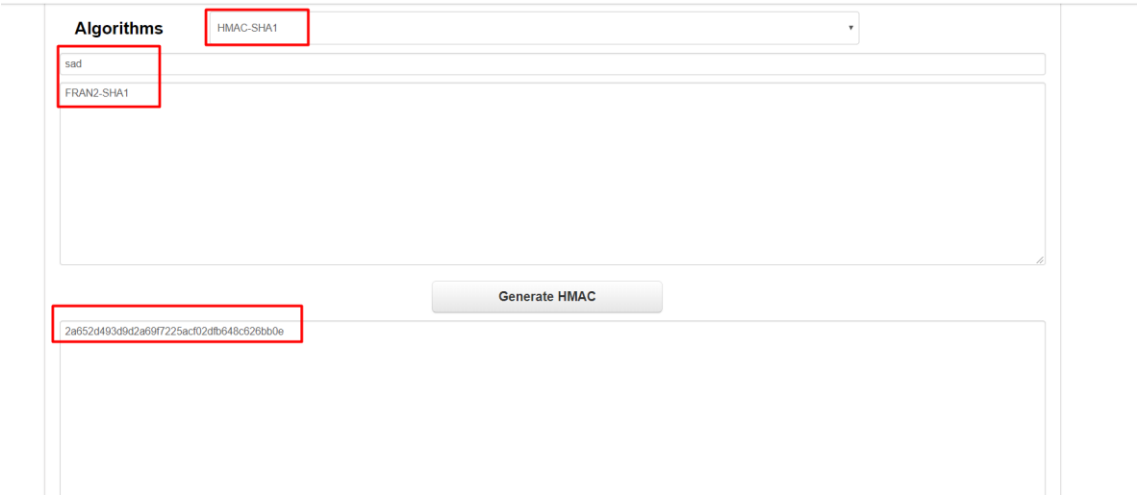
Comprobaremos con otra página que el resultado es realmente correcto. (Link)




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.



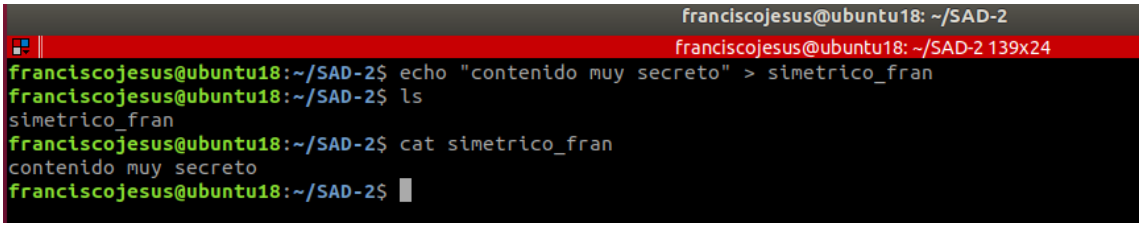

## c) CIFRADO: Simétrico y Asimétrico--> Confidencialidad, Integridad y Autenticidad.

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

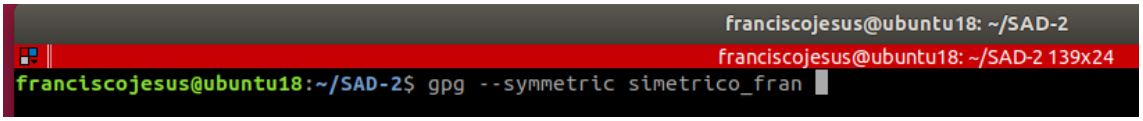
La criptografía de clave simétrica, también llamada criptografía de clave secreta o criptografía de una clave, es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y el receptor.

Utilizaremos para llevar acabo esta práctica **Linux con Ubuntu 18.** El programa que usare es **pgp.**

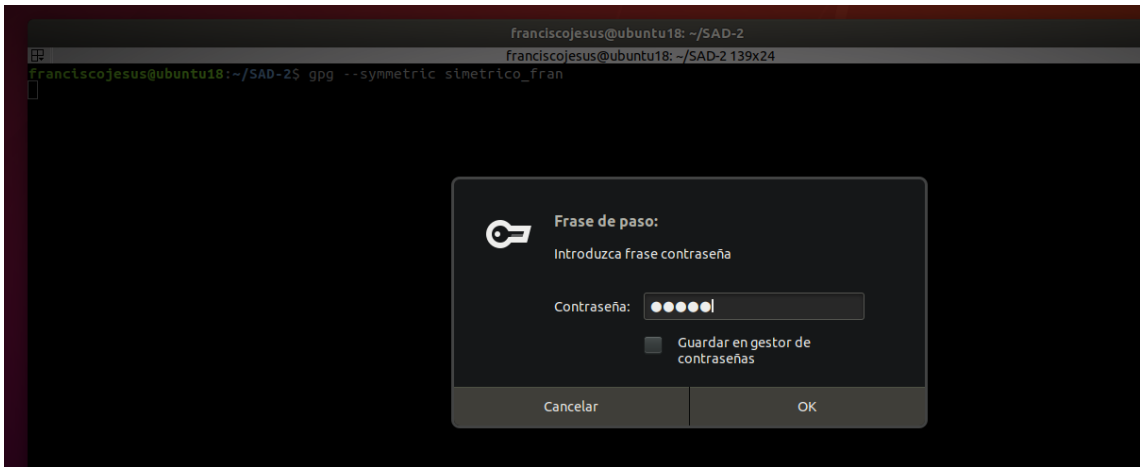
Primero crearemos un documento y lo cifraremos con una clave simétrica.



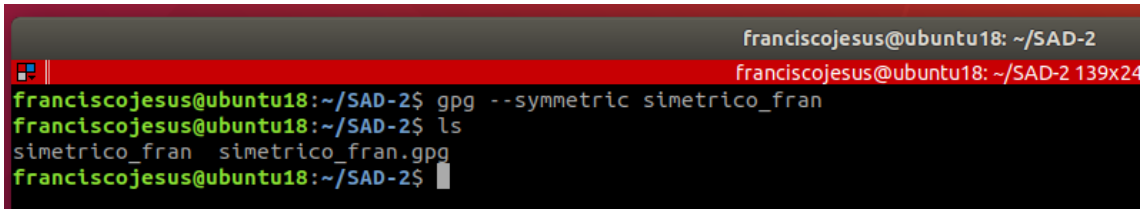

Ahora lo cifraremos de manera simétrica con gpg.



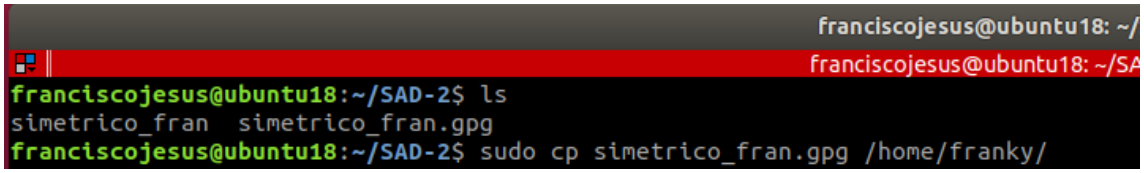

Ponemos una clave.




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



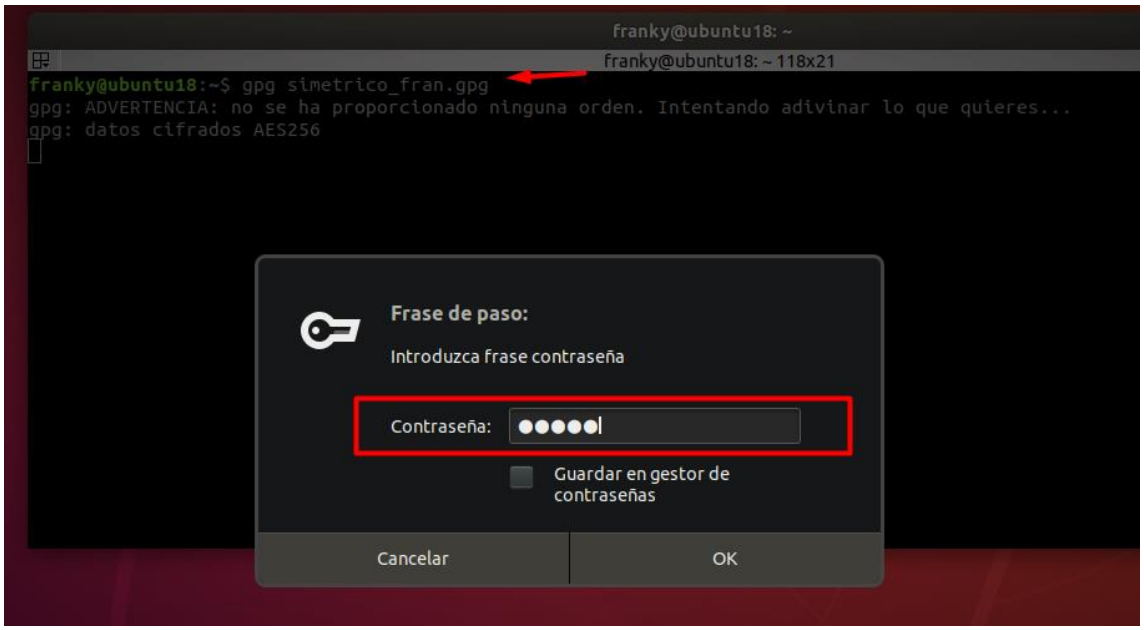

Ahora simplemente enviaremos el archivo cifrado con clave simétrica al /home del segundo usuario que tenemos.



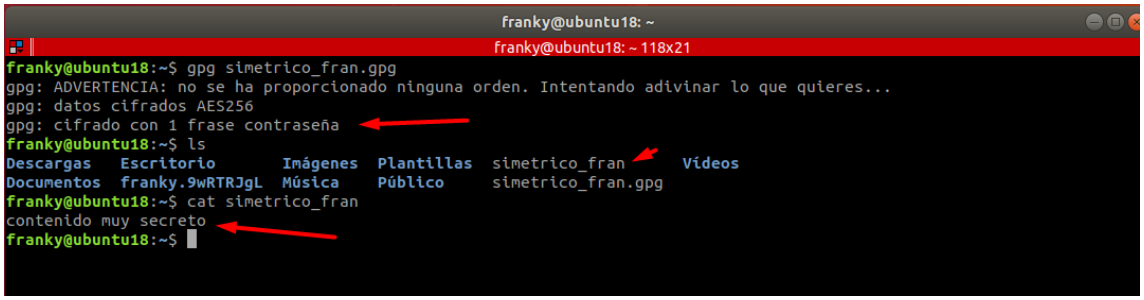

Iniciaremos sesión con el segundo usuario e intentaremos ver el contenido con un simple cat. Vemos como lógicamente no deja al estar cifrado.


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

*$ gpg simetrico_fran.gpg*




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

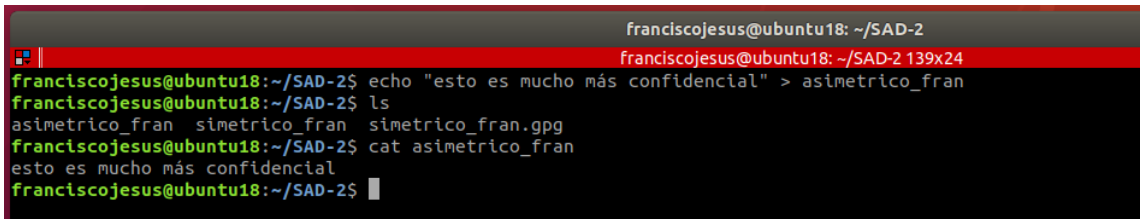



- 2. Utiliza un programa en Windows o GNU/Linux para simular la confidencialidad mediante “cifrado asimétrico”.

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 Ubuntu 18.

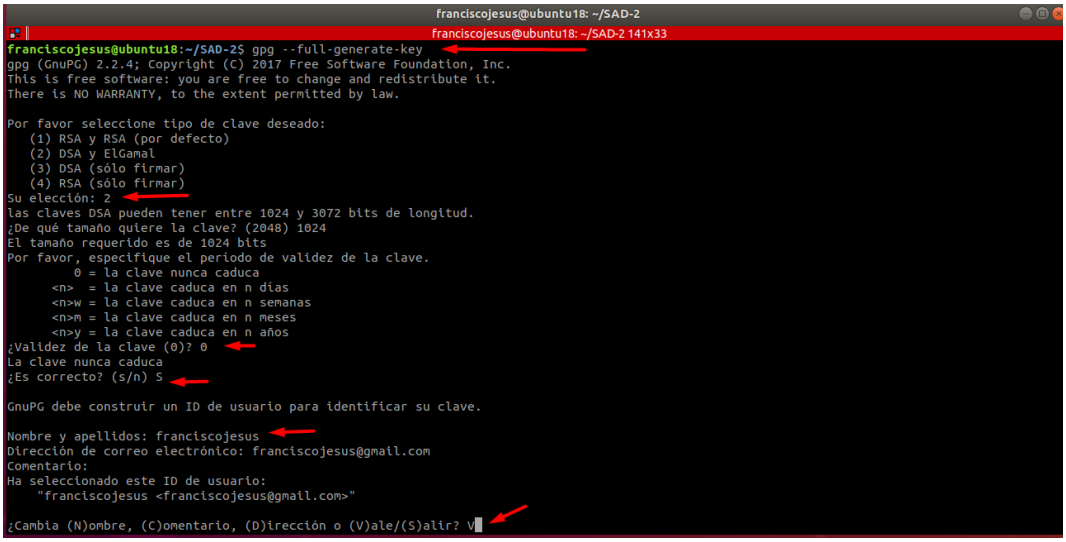
Crearemos para ello un nuevo archivo con la cuenta franciscojesus.



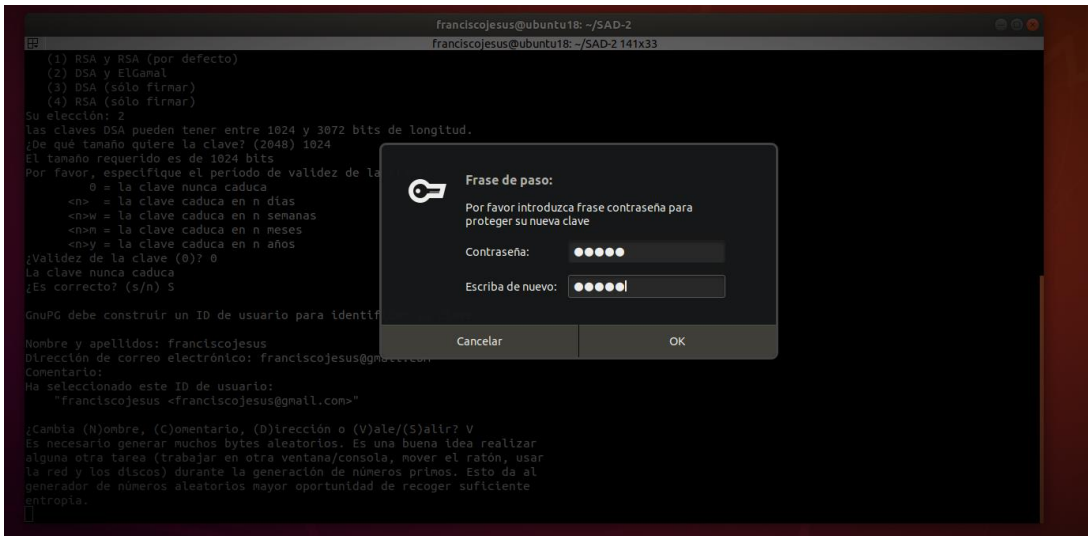
Ahora generaremos las claves necesarias (dos, una publica y otra privada) para cada usuario.

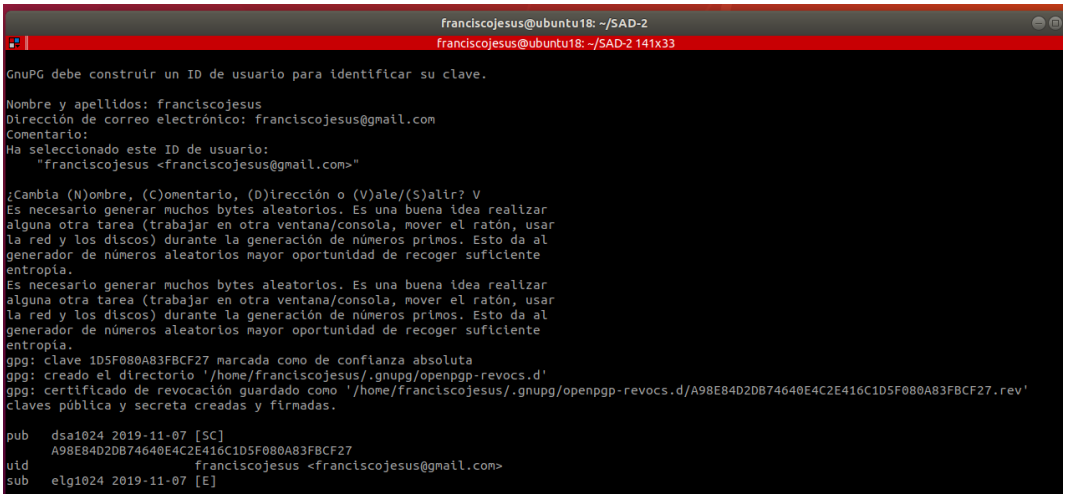
Primero crearemos las claves del primer usuario.

*$ gpg –full-generate-key*

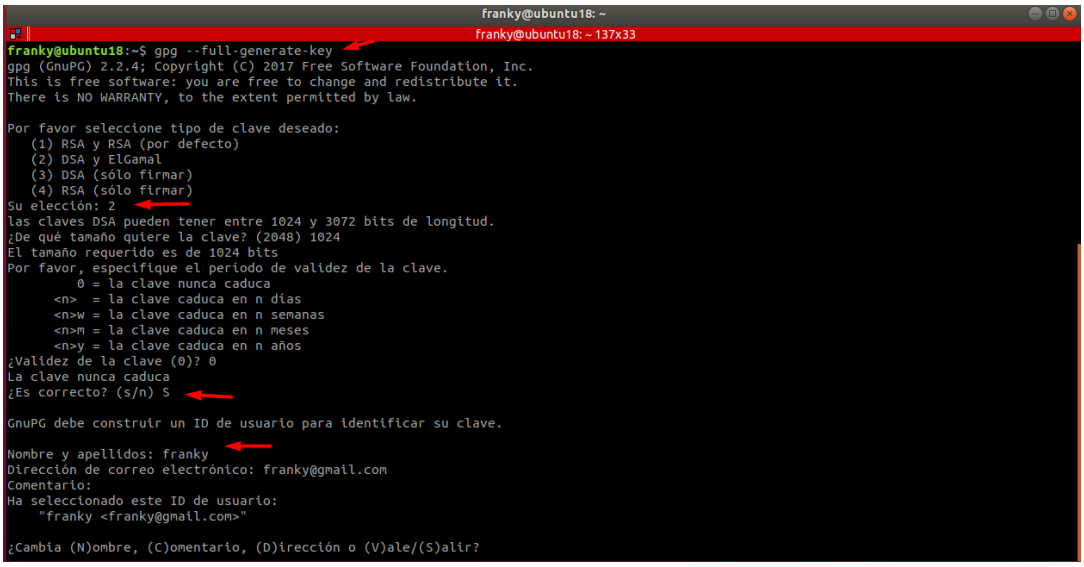



Le ponemos una contraseña.

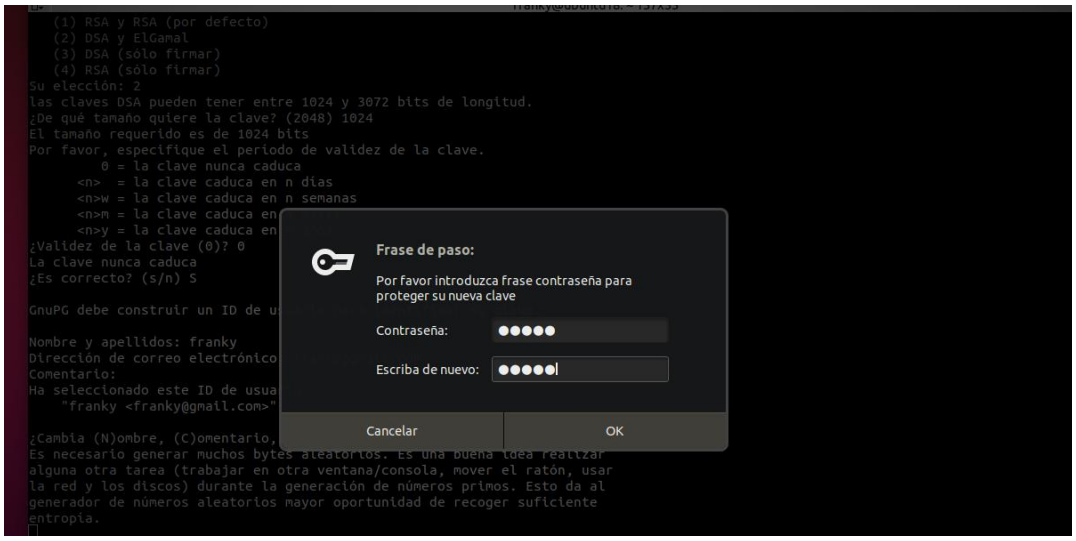


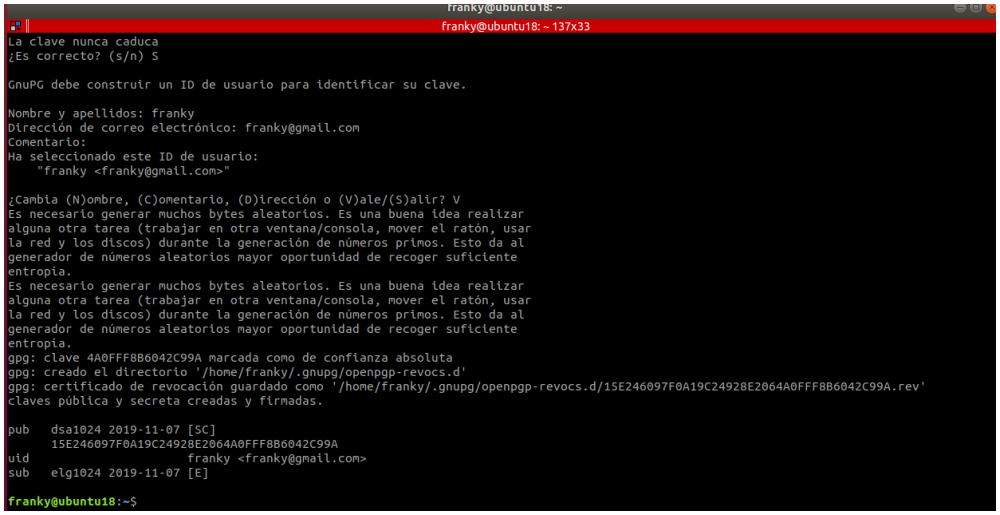
Ahora, **vamos al otro usuario (Franky)** y generaremos su clave igual que hicimos con franciscojesus.




Le ponemos una contraseña.



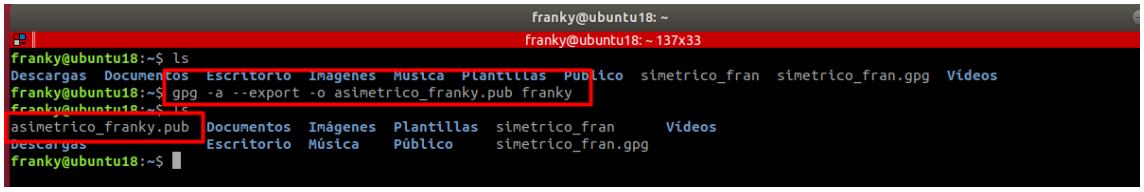

Ya la tendremos generada.



Ahora simplemente exportaremos las claves publicas de los dos usuarios.

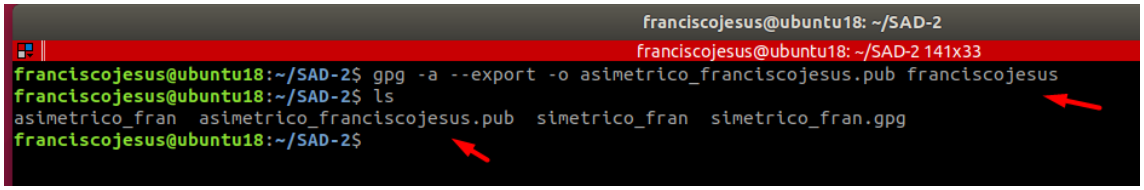
Franky:

*$ gpg -a --export -o asimetrico_franky.pub Franky*

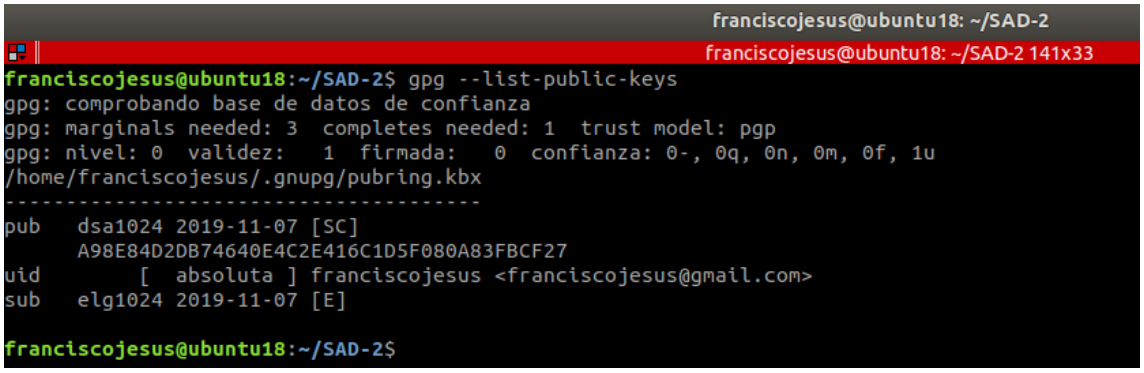



franciscojesus:

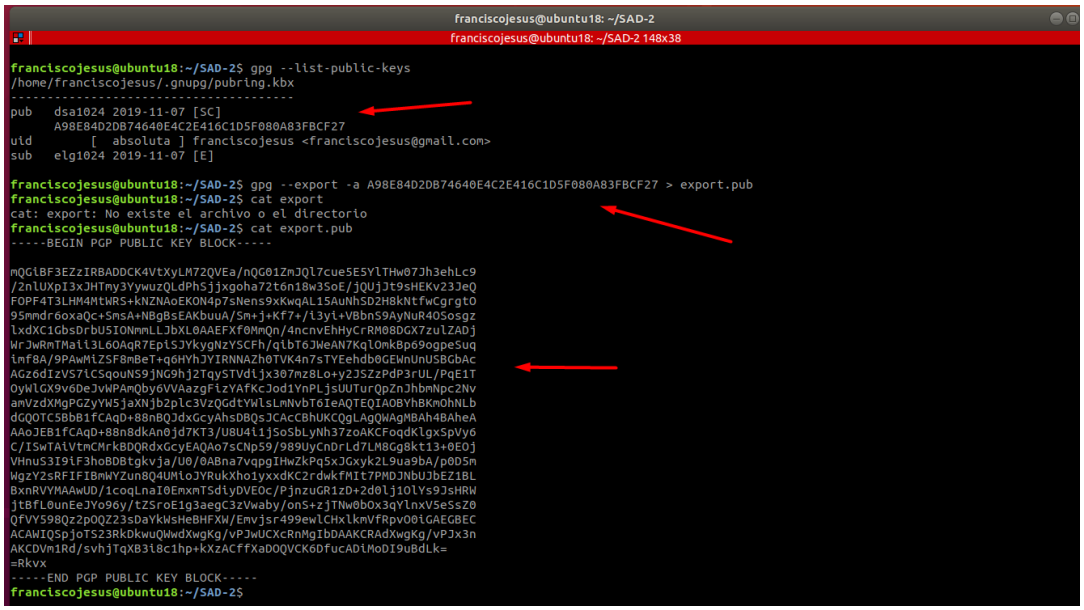
*$ gpg -a --export -o asimetrico_franciscojesus.pub franciscojesus*



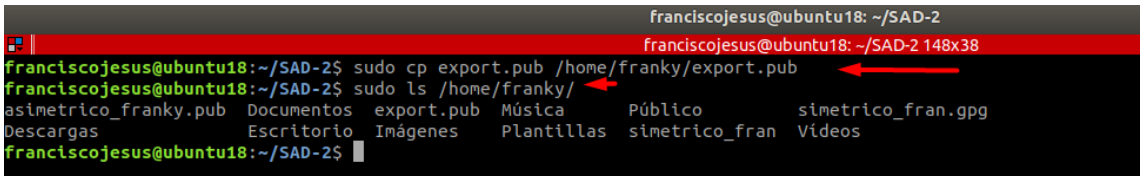

Recordamos que con el comando *gpg –list-public-keys* podemos ver las claves públicas que tenemos.



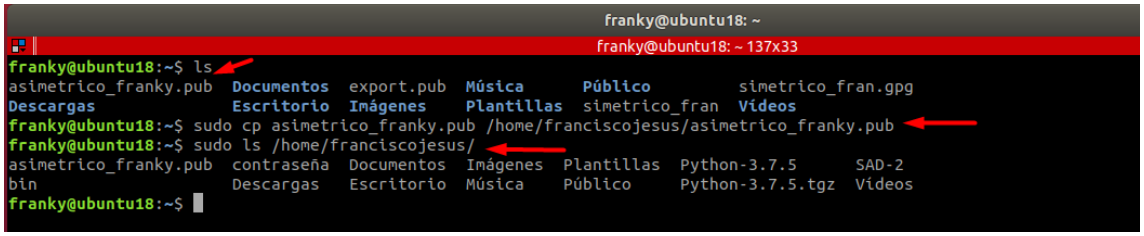

Podemos de esa forma exportar la clave publica que queramos.



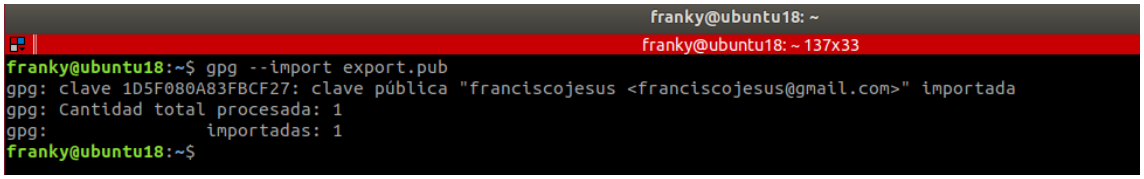
Ahora copiaremos las claves públicas de un usuario a otro. Clave publica de franciscojesus a Franky.

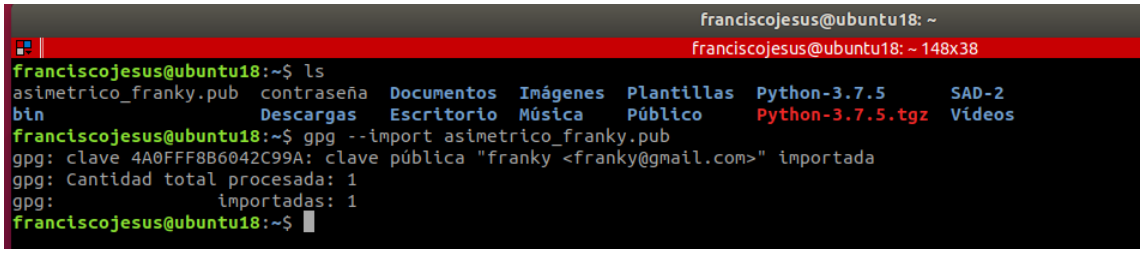



Clave publica de Franky a franciscojesus.

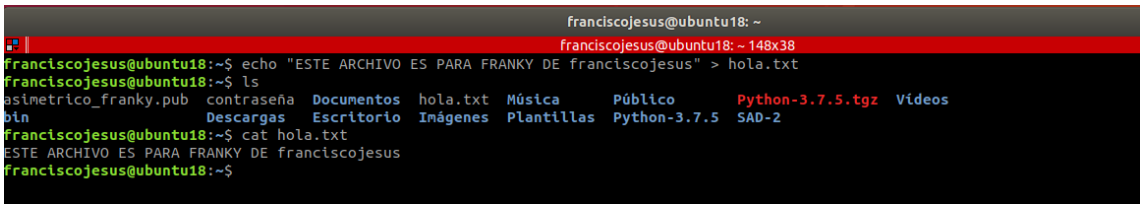



Ahora importaremos las claves copiadas a cada usuario.



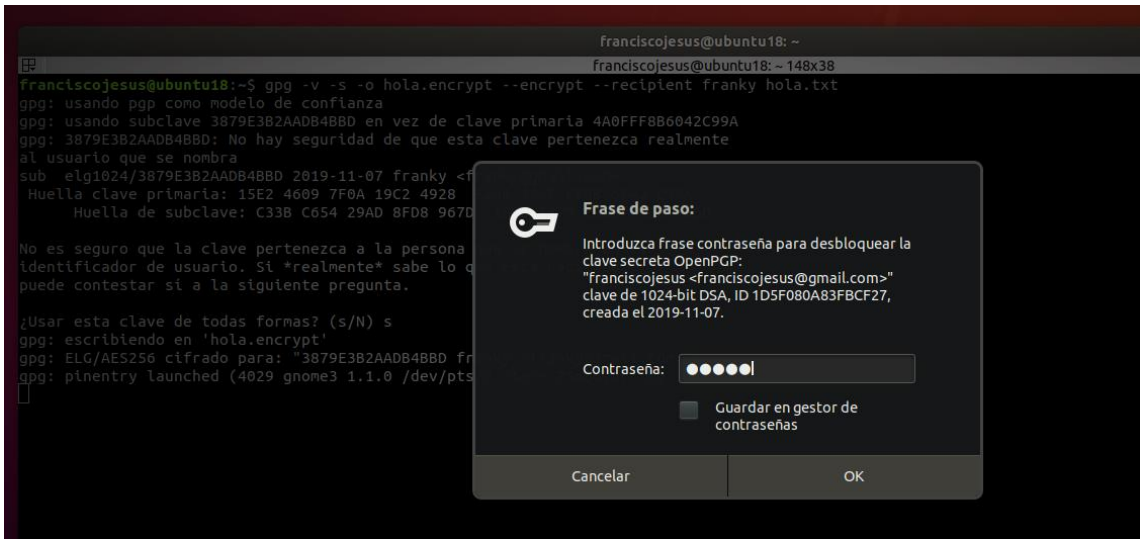



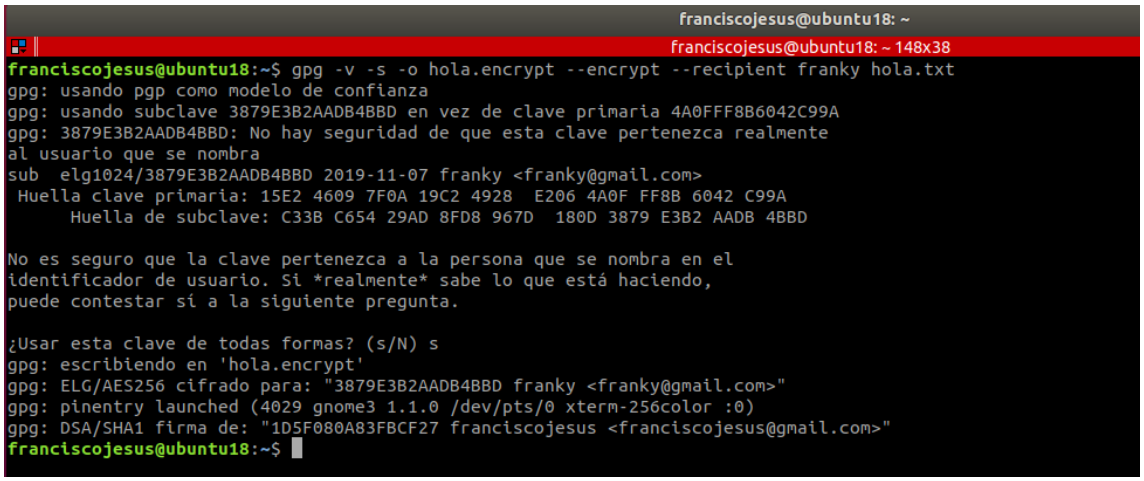

Ahora creamos un archivo nuevo en la cuenta franciscojesus y lo encriptaremos con la clave publica del usuario Franky que acabamos de importar.



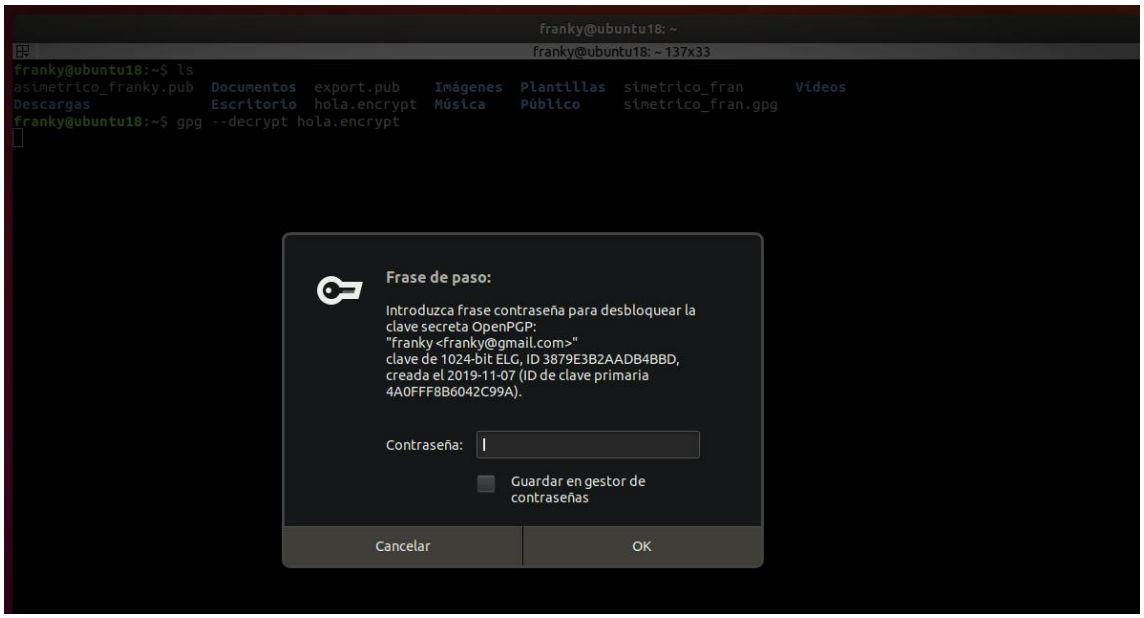

Lo encriptamos con la clave publica de Franky.

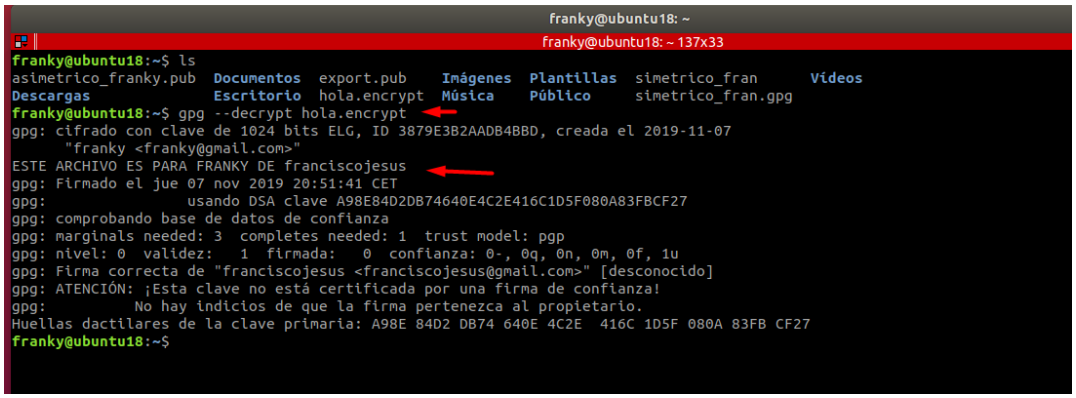
*$ gpg -v -s -o hola.encrypt --encrypt --recipient franky hola.txt*



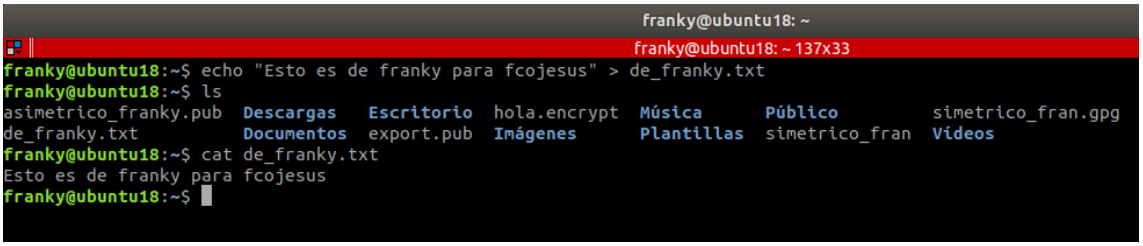



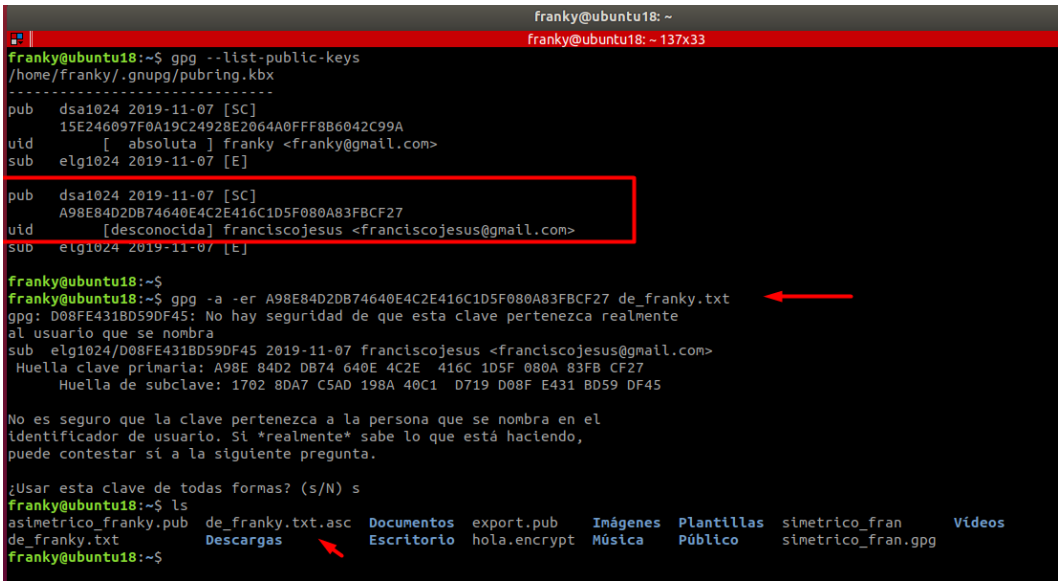

Pasamos el archivo a Franky y lo desciframos con su clave privada (lo podemos realizar ya que el archivo lo ciframos con la clave publica de Franky).



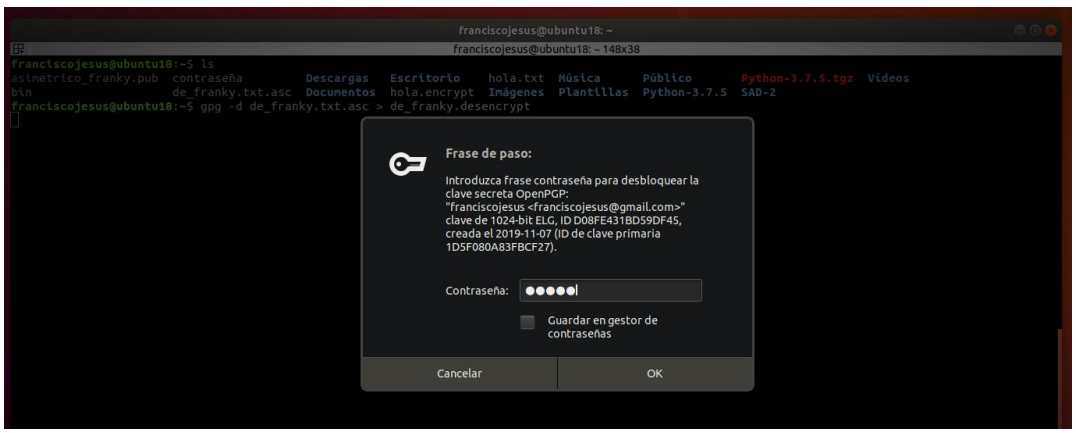


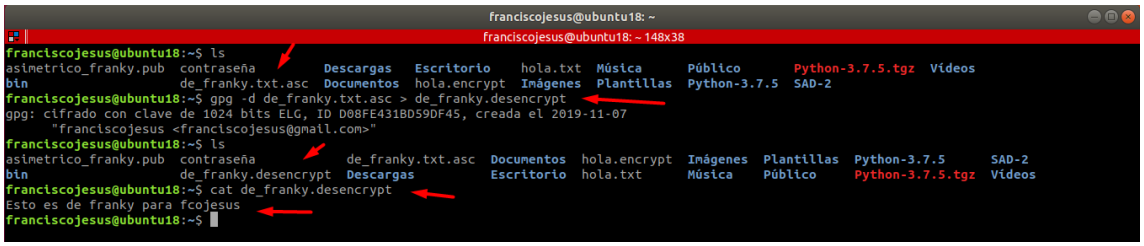

Podemos hacer lo mismo a la inversa (esta vez usaremos otro comando para cifrar con la clave publica, esta vez especificaremos con la clave).




Lo movemos al /home de franciscojesus y lo desciframos.



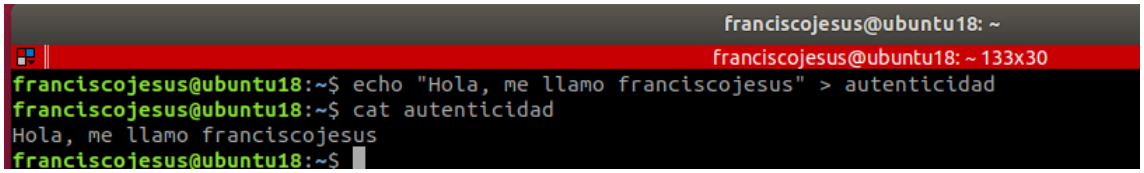



Hemos podido aprender a importar de dos modos y a desencriptar de dos modos, como vemos es sencillo, se tarda más en comprender y leer que en hacer.

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

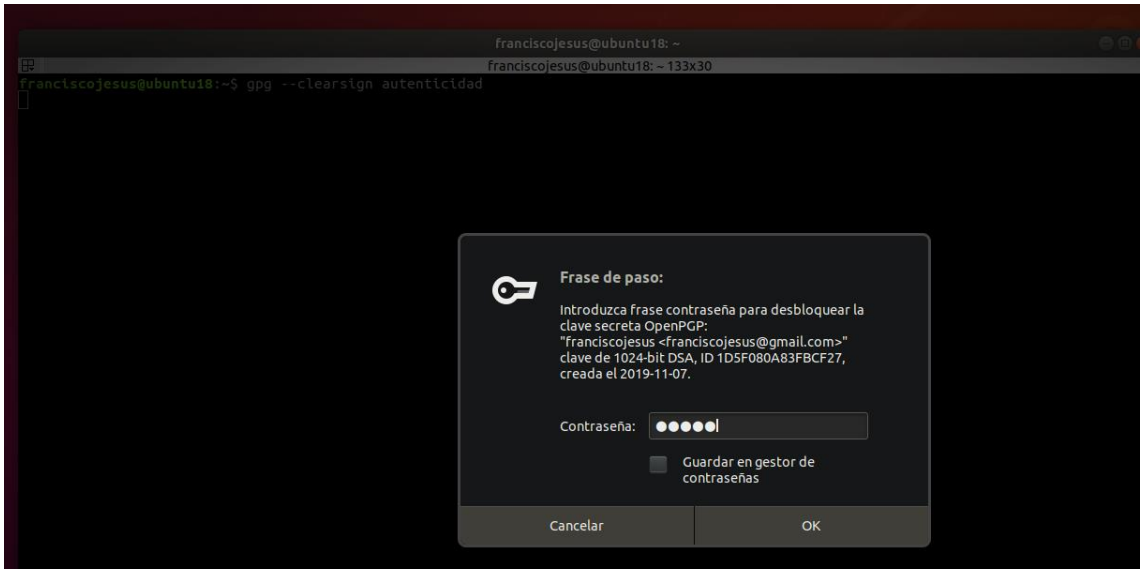
Una firma digital certifica el contenido, fecha y hora de un documento. Si el documento luego es modificado de cualquier forma, la verificación de la firma fallará. Una firma digital funciona entonces como una firma escrita a mano en papel, con la ventaja adicional de resistir cualquier tipo de falsificación o alteración.

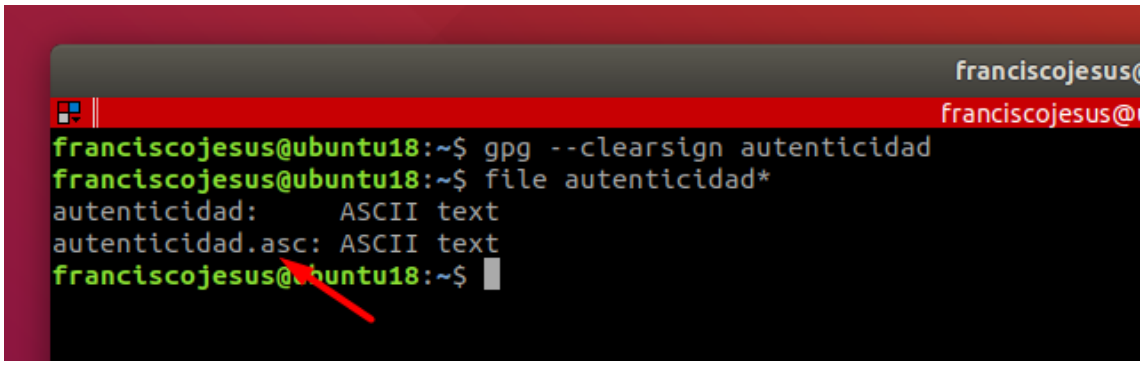
Crearemos primero un archivo con algo escrito en el.



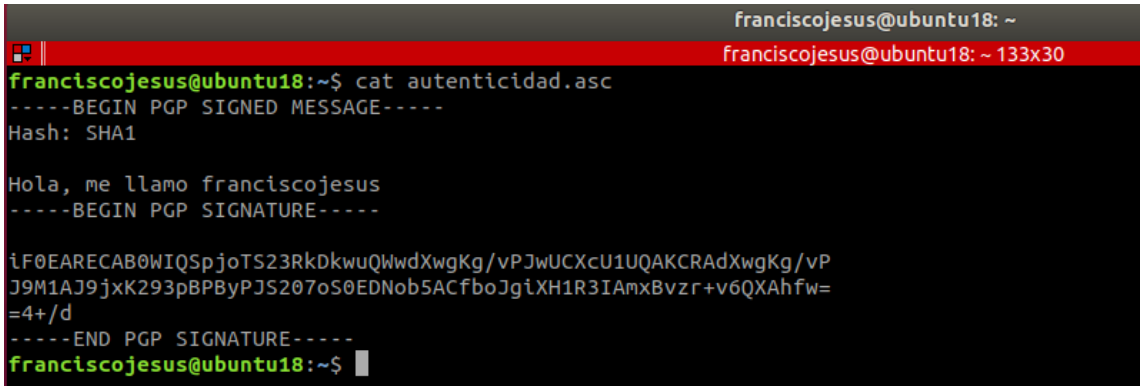

Lo siguiente que haremos es firmar el documento con la clave privada.

*$ gpg --clearsign autenticidad*

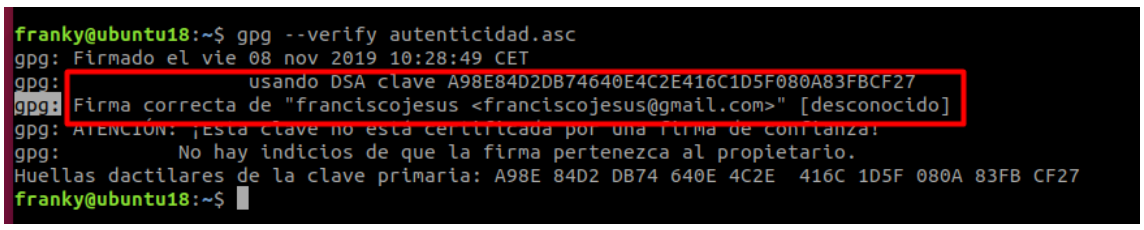




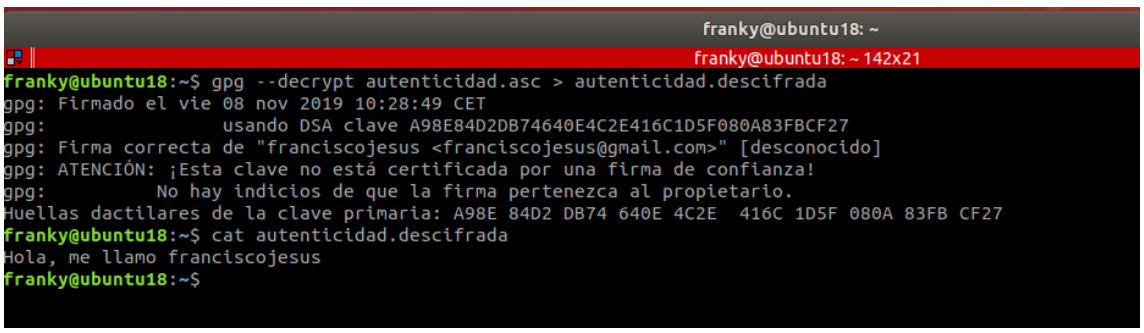

Podemos ver como se ha creado el mismo archivo con extensión .asc Si hacemos un cat podemos ver el contenido con la clave SHA1.




Ahora este mismo archivo se lo pasamos al usuario franky. Verificamos el archivo que este correctamente y no se haya alterado.

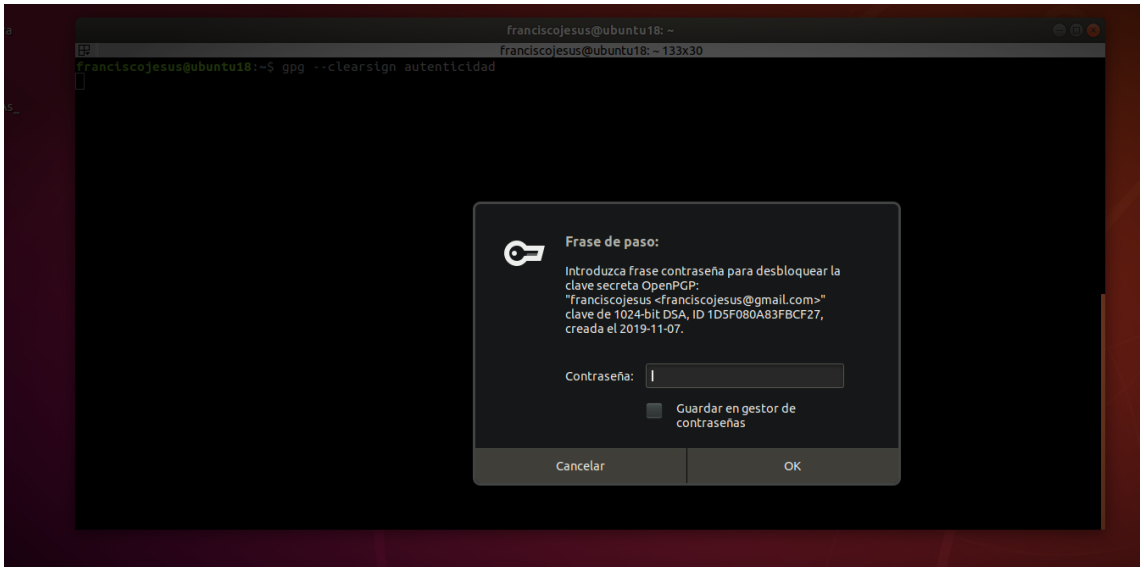


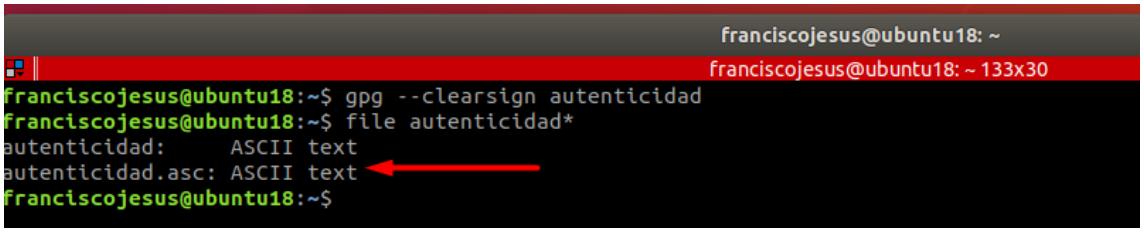

Lo desciframos con la clave publica de franciscojesus. Vemos tanto el mensaje como la firma.



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

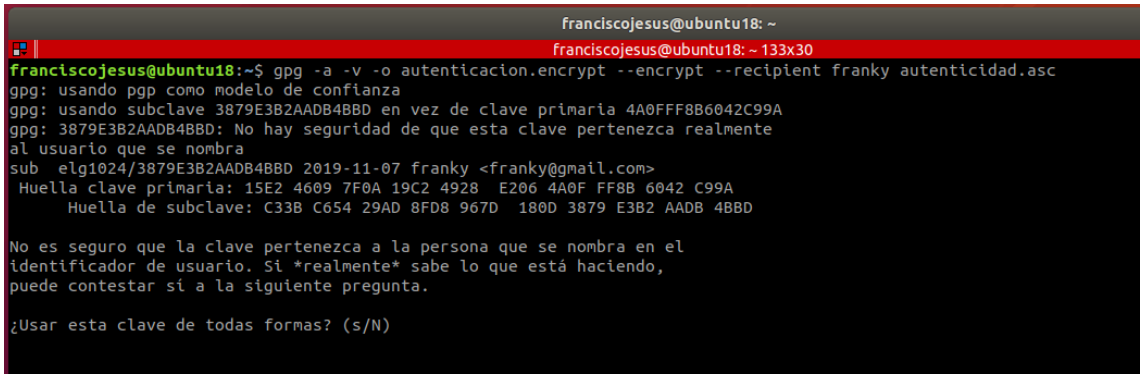
Primero firmaremos el documento *autenticidad* otra vez con la clave privada del usuario franciscojesus.

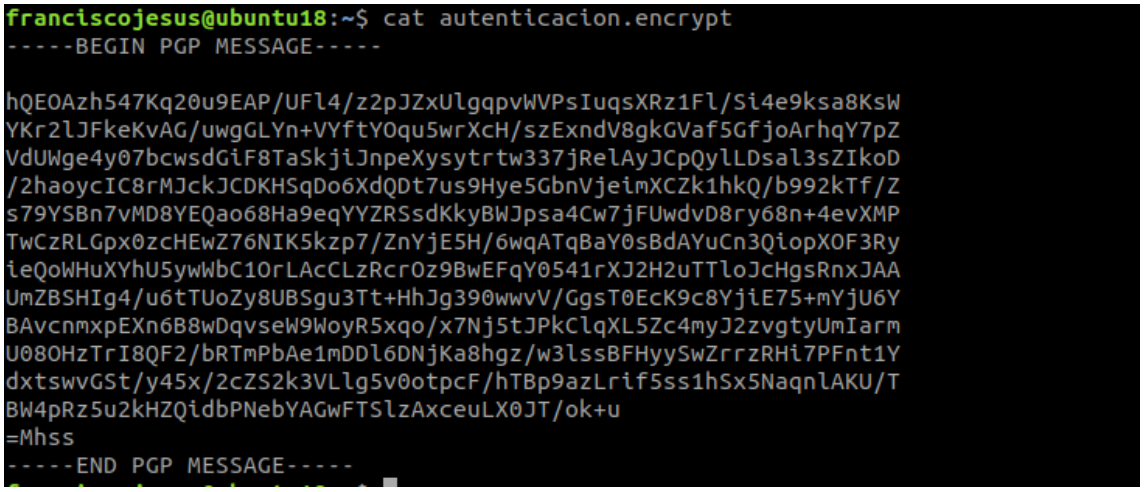



Ahora ciframos el documento firmado con la clave privada de franky para que este pueda descifrarlo.

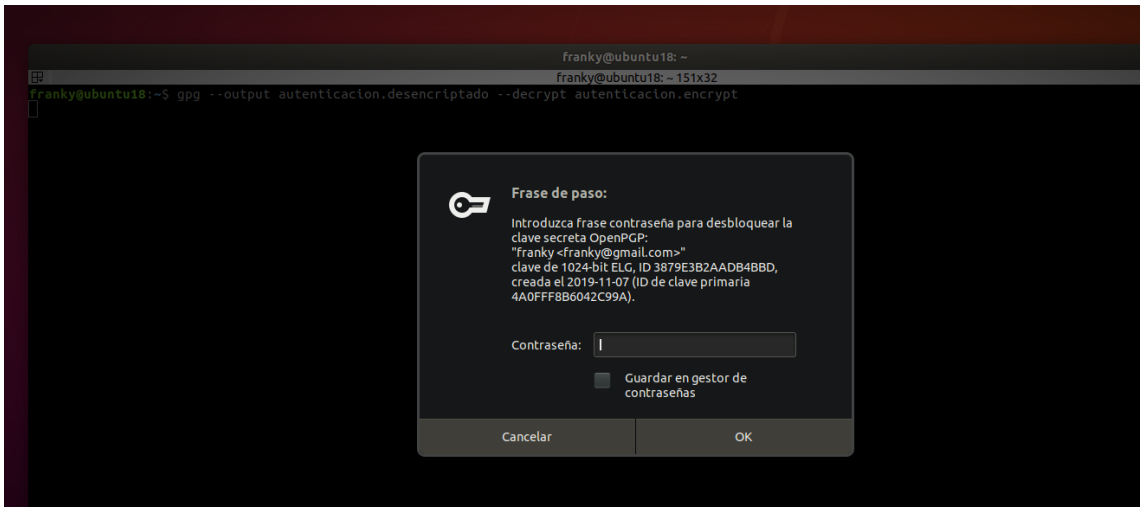
*$ gpg -v -a -o autenticacion.encriptado --encrypt --recipient franky autenticidad.asc*



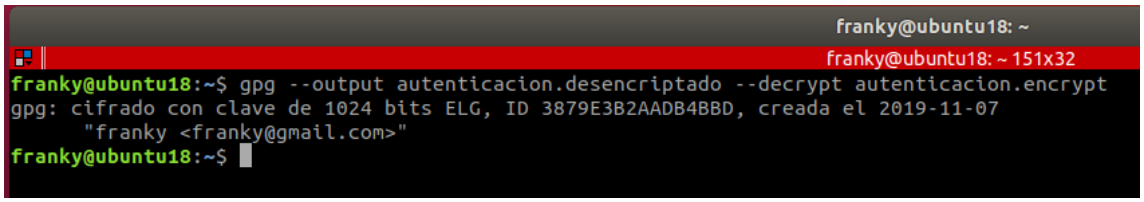




Le mandamos el documento a franky y lo desciframos con este.

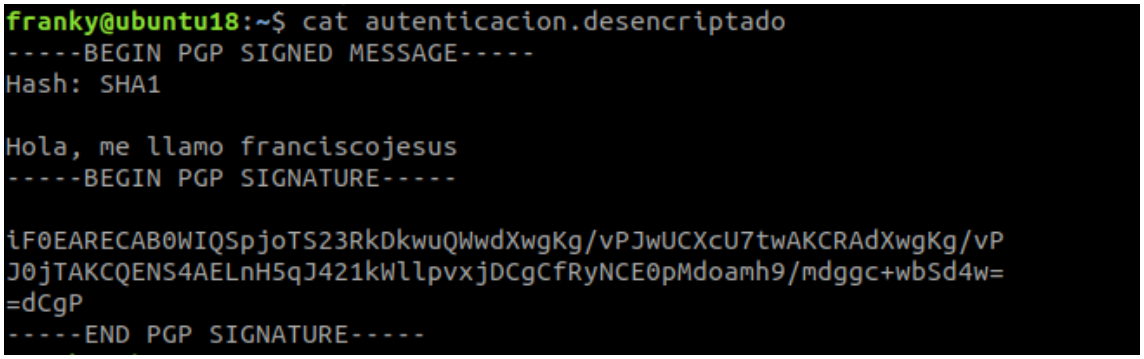
*$ gpg --output autenticidad.desencriptado --decrypt autenticidad.encrypt*



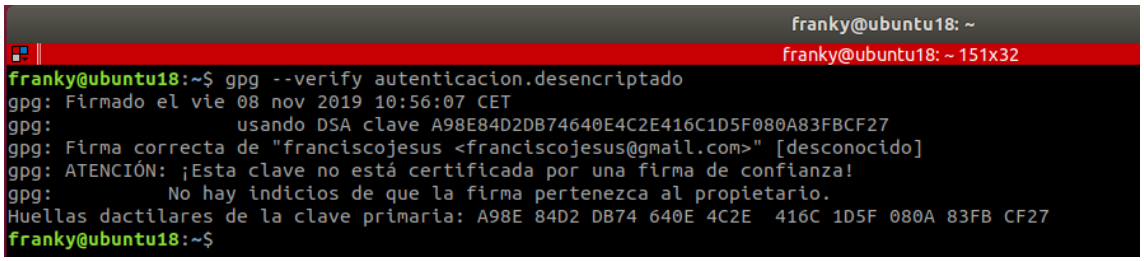

*$ gpg --decrypt autenticidad.desencriptado*




Podemos ver si hacemos un cat autenticidad.desencriptado como sigue aún la firma y el documento de desencripto.

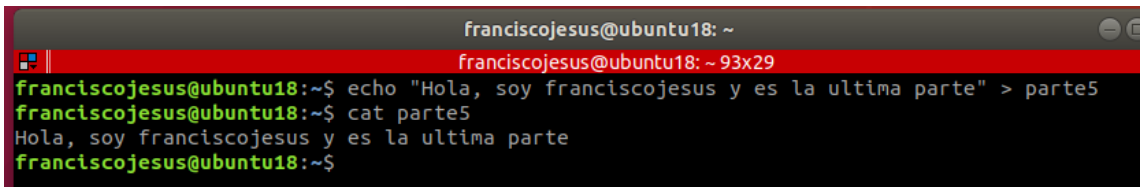



Si hacemos un *gpg –verify* podremos ver la firma la firma del documento y como está firmado por franciscojesus.



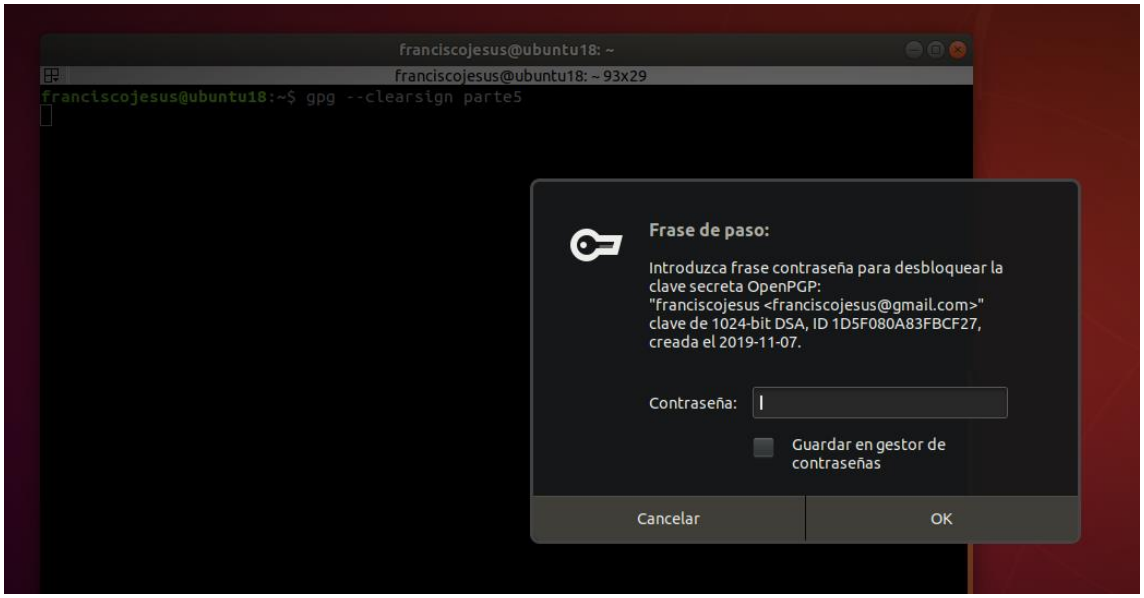

## 5. Utiliza un programa en Windows o GNU/Linux para simular una comunicación segura utilizando cifrados híbridos “autenticidad” + “confidencialidad”+”integridad”: asimétricos (clave pública) y simétricos (clave privada).

Lo primero es crear un archivo, en mi caso se llamará parte5

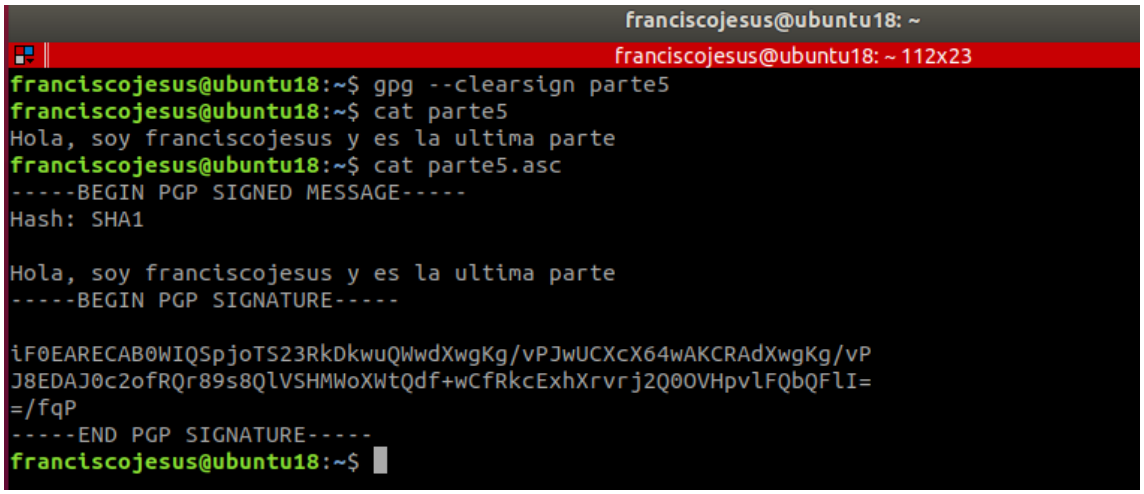


Lo firmaremos con el usuario *franciscojesus* :

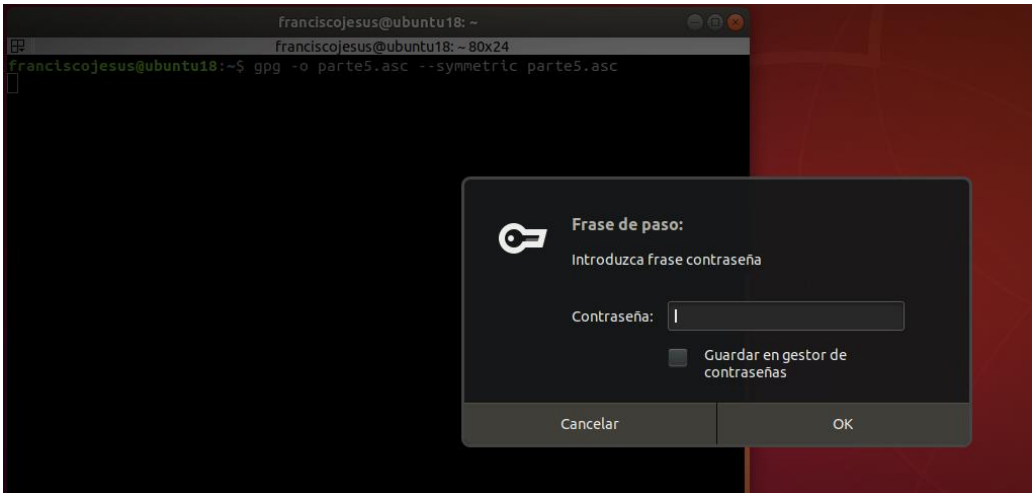
$ gpg –clearsign parte5

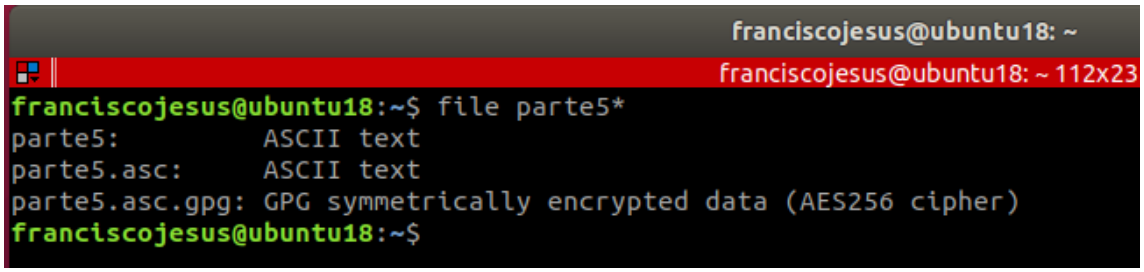



Podemos ver el archivo firmado con un SHA1 para verificar la integridad del contenido y la firma.



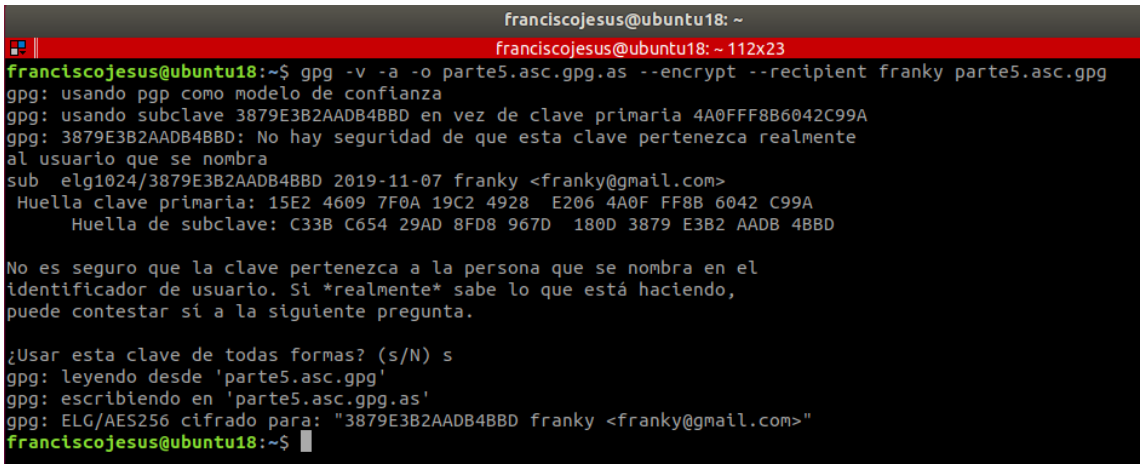

Lo siguiente que realizaremos es un cifrado simétrico. Podemos ver como coge la extensión gpg.



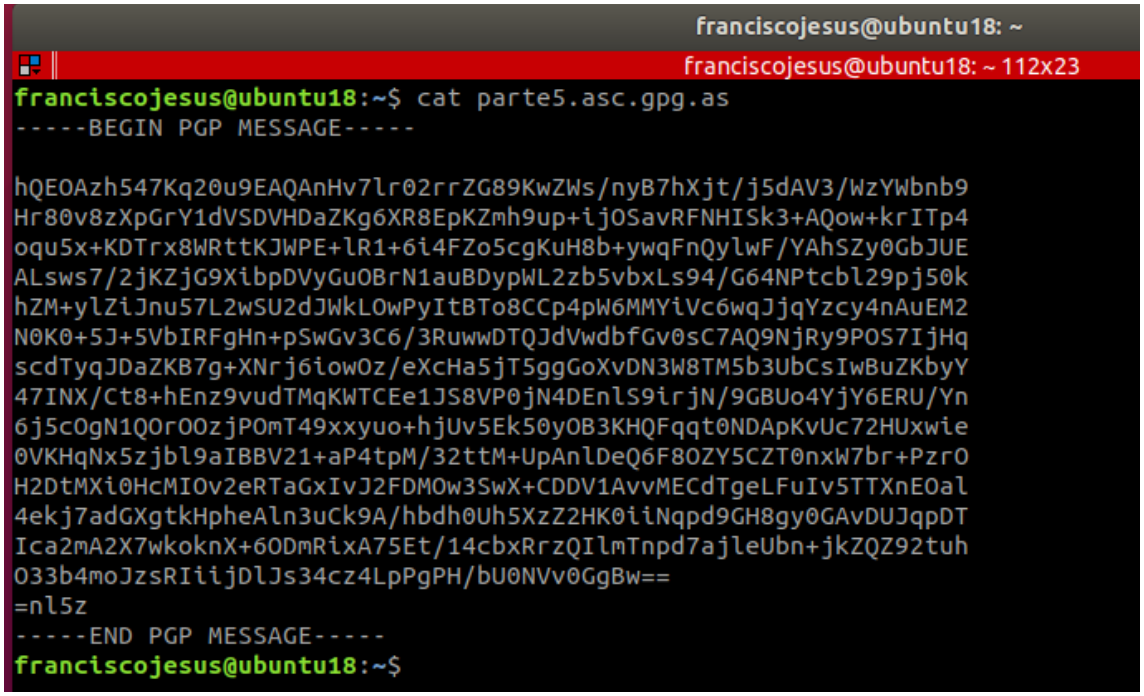



Ahora realizaremos el cifrado asimétrico, utilizaremos la clave publica de franky para que así este pueda descifrarlo con su clave privada cuando se lo pasemos. El nuevo archivo generado se llamara parte5.asc.gpg.as (lo redirigimos con -o).

gpg -v -a -o parte5.asc.gpg.as --encrypt --recipient franky parte5.asc.gpg

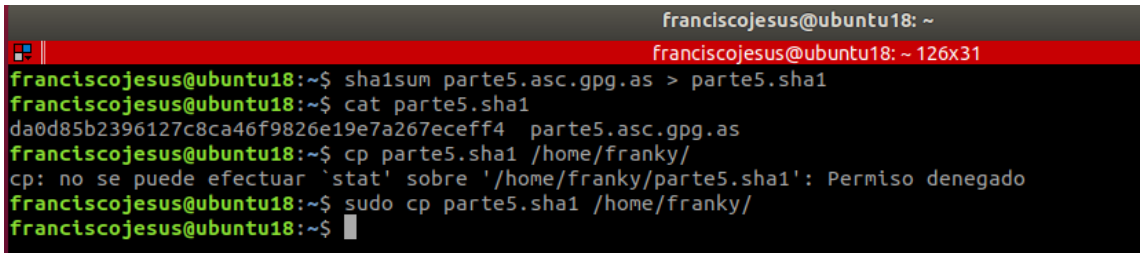



Podemos ver cómo está totalmente cifrado.

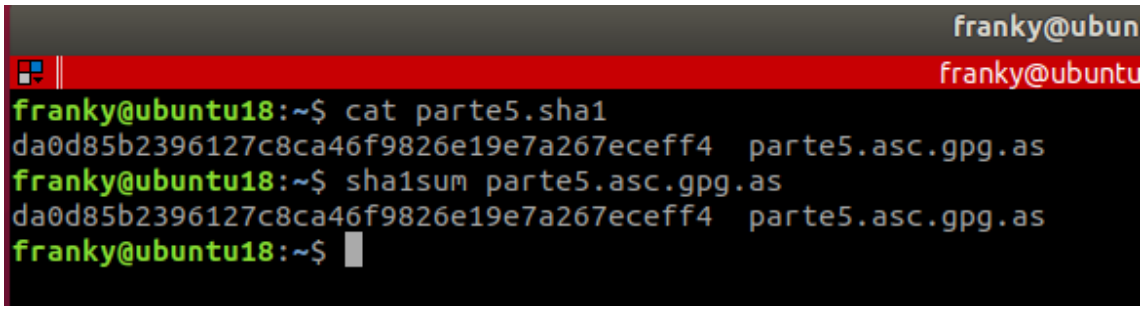


Lo siguiente que haremos es crear el SHA1 del archivo anterior cifrado (parte5.asc.gpg.as). Lo que hare es redirigir el sha1 al archivo *parte5.sha1* para después compararlo y verificar la integridad. Copiare tanto el archivo cifrado *parte5.asc.gpg.as* como el archivo *parte5.sha1* al usuario franky.

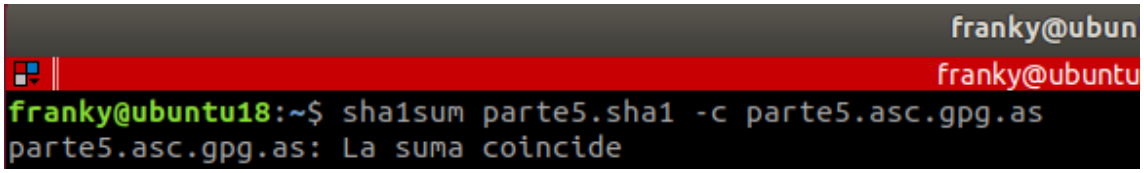
$ sha1sum parte5.asc.gpg.as > parte5.sha1



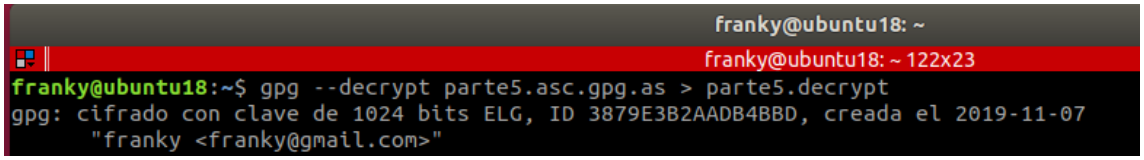

Ya en el usuario franky podemos hacer un cat a parte5.sha1 para ver el SHA1 generado, después, con sha1sum generamos el sha1 del archivo parte5.asc.gpg.as y compararemos que coinciden los SHA1 para verificar la integridad del archivo. Vemos que son iguales por lo cual no fue alterado.



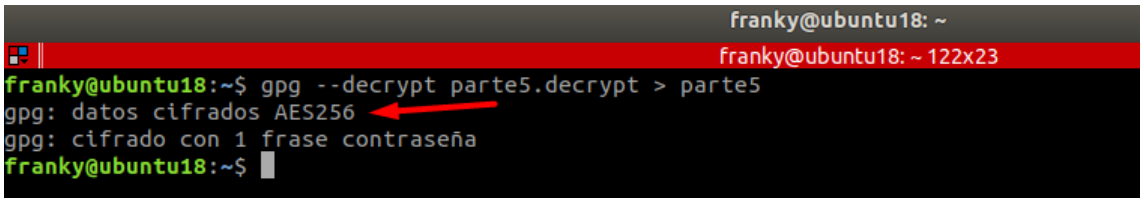

Si no queremos compararlo 1 a 1 podemos ejecutar el comando sha1sum con -c para comprar el sha1 que le hemos pasado al usuario franky con el archivo cifrado que ha recibido. Vemos que la suma coincide por lo cual la integridad es correcta.



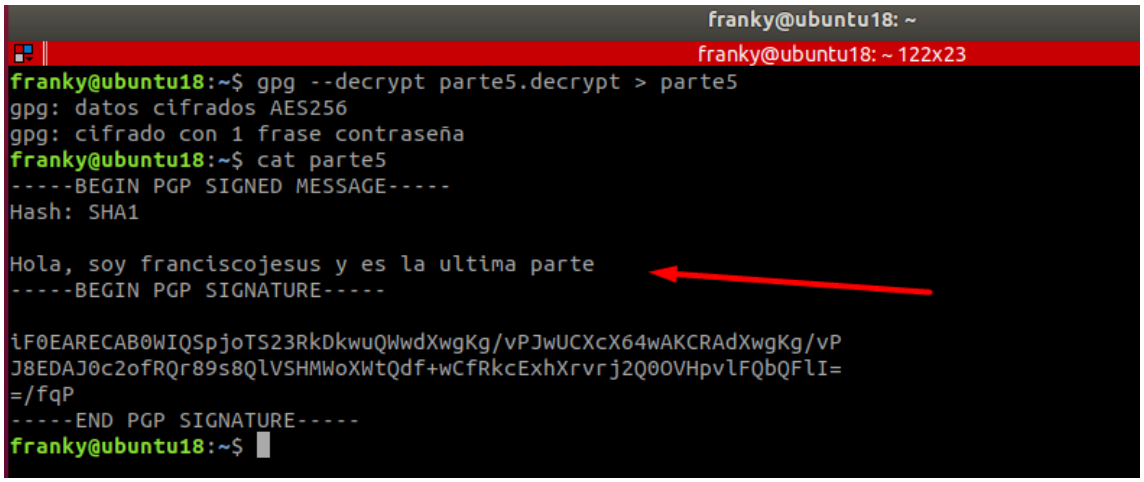

Ahora empezaremos a desencriptarlo. Primero desencriptaremos el cifrado asimétrico con la clave privada del usuario franky. Lo redirigimos al archivo parte5.decrypt



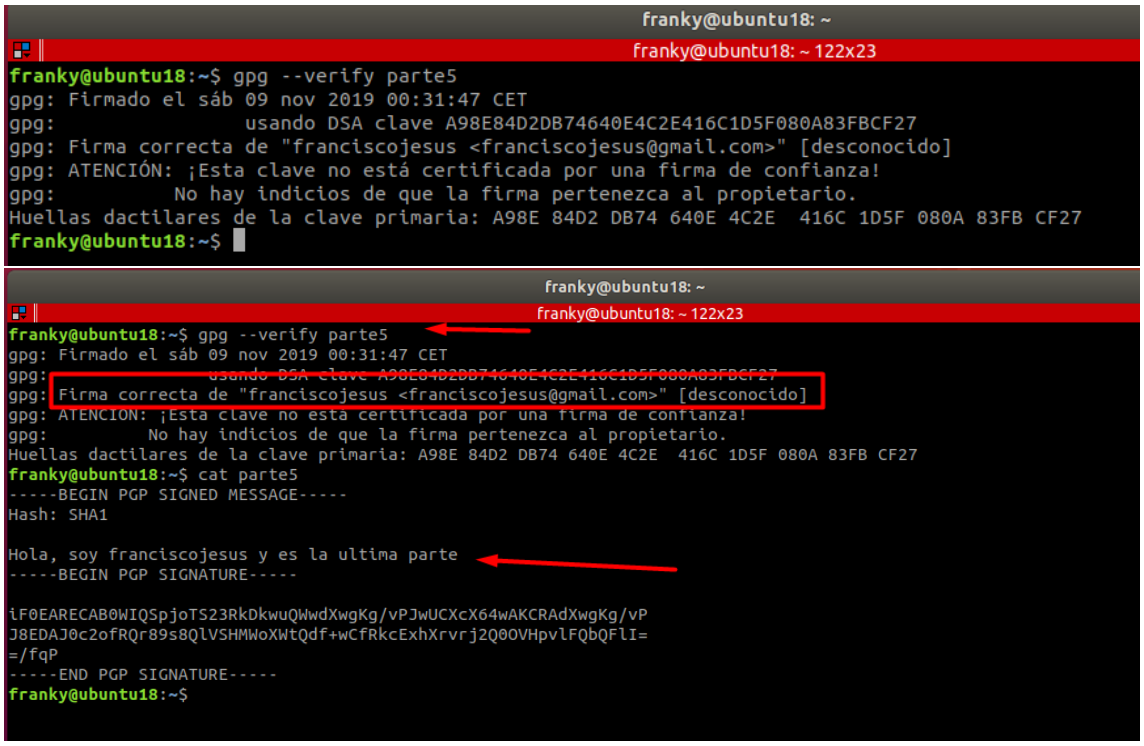

Después desencriptamos el cifrado simétrico con la clave secreta (la contraseña que le pusimos al archivo cuando hicimos el cifrado simétrico). Lo redirigimos al archivo parte5.



Podemos ver como se desencripto y tenemos el mensaje visible con el SHA1 (este SHA1 es para verificar la integridad de la firma y mensaje cuando lo firmamos).




Si ejecutamos el comando gpg –verify parte5 podemos ver que está firmado correctamente por *franciscojesus* y no se ha alterado en ningún momento. Ya tenemos el mensaje descifrado correctamente y visible.




## Conclusión

De las mejores prácticas que podemos hacer. Me ha divertido mucho aprender sobre los cifrados, las últimas partes me han costado entenderlas ya que no conseguía saber como juntar todo lo realizado en un solo archivo, pero con la explicación de algunos compañeros y comprendiendo lo que pedía el ejercicio me conseguí aclarar y pude comprender cada cosa perfectamente.