

ACT0302 PRÁCTICA DE CIFRADO ASIMÉTRICO



Contenido

1. INTRODUCCIÓN.....	2
2. DESARROLLO DE LA PRÁCTICA.....	2
2.1 Ejercicio simple de cifrado asimétrico	2
2.2 Ejercicio de cifrado y descifrado.....	3
2.3 Ejercicio simple de firma electrónica	6
2.4 Jugando con gpg.....	7
Observaciones y conclusiones.....	12

1. INTRODUCCIÓN

En esta ocasión vamos a utilizar las dos herramientas vistas en la ACT0301 para trabajar con criptografía asimétrica. Para ello crearemos un conjunto de claves y las utilizaremos para las distintas funciones vistas en clase: cifrado y firma. Recuerda que los algoritmos de cifrado no siempre se utilizan para todas las funciones, por ejemplo, RSA se puede usar para todo: distribución de claves, firma o cifrado; mientras que DSA sólo se suele usar para firmar. Tenlo en cuenta.

2. DESARROLLO DE LA PRÁCTICA

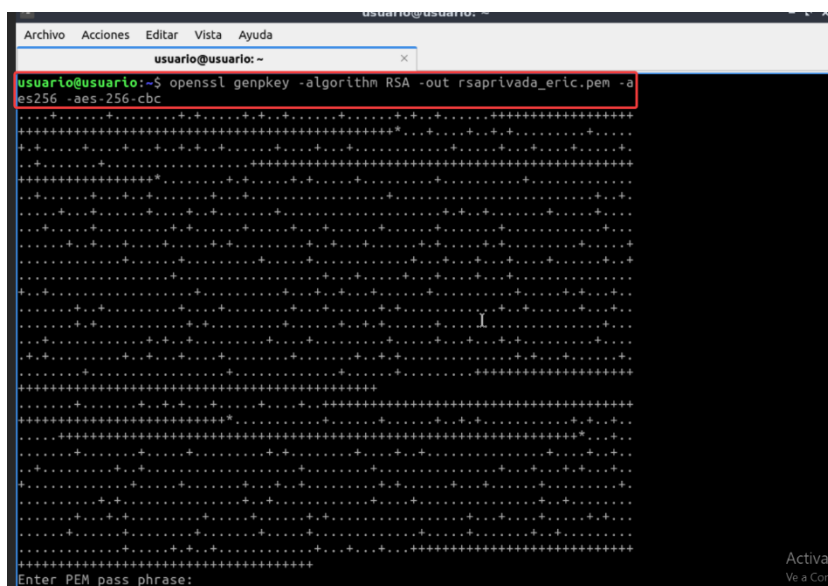
2.1 Ejercicio simple de cifrado asimétrico

Antes de nada, necesitamos generar nuestro par de claves. Para ello vamos a utilizar

OpenSSL. Genera una clave RSA de 4096 bits ¿Cuál es el tamaño de la clave por omisión? Serían 2048 bits.

¿Cuántos ficheros se han generado? Se generará solo un archivo, la clave privada.

Es recomendable explicitar en el archivo de salida que se trata de la clave privada, puedes llamar al fichero `rsa_privada`.



```
usuario@usuario:~$ openssl genpkey -algorithm RSA -out rsaprivada_eric.pem -aes256 -aes-256-cbc
```

The screenshot shows a terminal window with a dark background. The command `usuario@usuario:~$ openssl genpkey -algorithm RSA -out rsaprivada_eric.pem -aes256 -aes-256-cbc` is entered and highlighted with a red box. Below the command, there is a large block of asterisks representing the progress of key generation. At the bottom of the terminal, it says "Enter PEM pass phrase:" and "Activar" is visible on the right side.

Muy bien, ¿y dónde está la clave pública? ... Hay que generarla a partir de la clave privada creada, busca qué comando de openssl debes usar para generar la clave pública y génerala. Al igual que antes, es buena idea nombrarla de tal forma que sepamos qué es: `rsa_publica`.

```
usuario@usuario:~$ openssl rsa -pubout -in rsaprivada_eric.pem -out rsapublica_eric.pem
Enter pass phrase for rsaprivada_eric.pem:
writing RSA key
usuario@usuario:~$
```

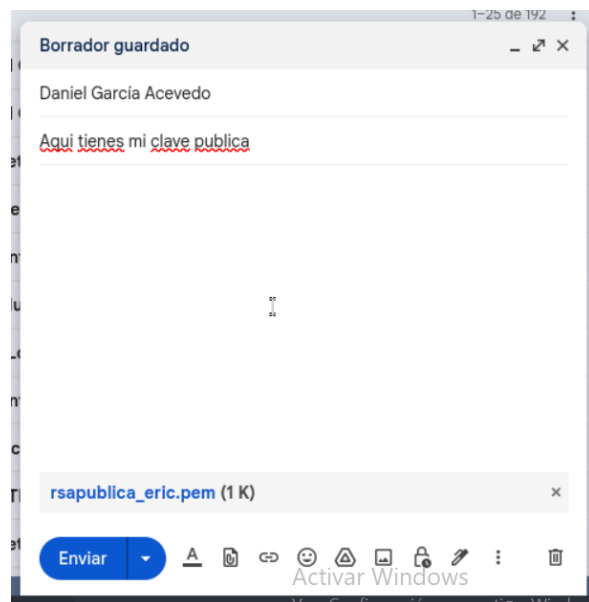
Comprueba que tienes los dos ficheros.

```
Verifying - Enter PEM pass phrase:
usuario@usuario:~$ openssl rsa -pubout -in rsaprivada_eric.pem -out rsapublica_eric.pem
Enter pass phrase for rsaprivada_eric.pem:
writing RSA key
usuario@usuario:~$ ls
Descargas  file      Plantillas  rsaprivada_eric.pem  Videos
Desktop    Imágenes Público      rsapublica_eric.pem  ZeroNet-linux-dist-linux64
Documentos Música    rekall      Snap                 ZeroNet-py3-linux64.tar.gz
usuario@usuario:~$
```

2.2 Ejercicio de cifrado y descifrado

Una vez que tenemos nuestras claves es hora de hacer uso de ellas. Para ello sigue los siguientes pasos:

a) Intercambia el fichero que corresponda con el compañero. ¿clave pública o privada? Clave pública.



b) Cifra un mensaje de texto para que el compañero pueda descifrarlo con su clave.

Documenta los pasos indicando en cada momento qué claves utilizas, privada o pública y por qué.

Creación de mensaje.

```
usuario@usuario:~$ echo "¿Cuál es el animal más antiguo? La cebra, porque está en blanco y negro."
> mensaje_eric.txt
usuario@usuario:~$
```

Cifrando mensaje con clave pública de Dani (procedemos después a enviarle el mensaje para que el lo descifre con su clave privada).

```
usuario@usuario:~$ openssl rsautl -encrypt -inkey rsa_publica.pem -pubin -in mensaje_eric.txt -out mensaje_para_dani_cifrado.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
usuario@usuario:~$ ls
Descargas  Imágenes  Plantillas  rsa_publica.pem
Desktop    mensaje_eric_cifrado.txt  Público     snap
Documentos mensaje_eric.txt          rekall      Videos
Escritorio mensaje_para_dani_cifrado.txt rsaprivada_eric.pem ZeroNet-linux-dist-linux64
file       Musica      rsapublica_eric.pem ZeroNet-py3-linux64.tar.gz
usuario@usuario:~$
```

Ahora para descifrar (el mensaje que nos pasa Dani) vamos a usar nuestra propia clave privada, ya que nuestro compañero ha encriptado con nuestra pública.

```
usuario@usuario:~$ openssl rsautl -decrypt -inkey rsaprivada_eric.pem -in MensajeparaEricCifrado.txt -out mensajededanidescifrado.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
Enter pass phrase for rsaprivada_eric.pem:
usuario@usuario:~$ cat mensajededanidescifrado.txt
Eric cómeme el cimbrel
```

d) Intercambia el fichero cifrado con el compañero y descifralo. Primero inténtalo con la clave del compañero que te envió y que tú has utilizado para cifrar el mensaje que le enviaste.

```
usuario@usuario:~$ openssl rsautl -decrypt -inkey rsa_publica.pem -in MensajeparaEricCifrado.txt -out mensajededanidescifrado2.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
Could not read private key from rsa_publica.pem
40D73ABD777F0000:error:1608010C:STORE routines:ossl_store_handle_load_result:unsupported:../crypto/store/store_result.c:151:
40D73ABD777F0000:error:1608010C:STORE routines:ossl_store_handle_load_result:unsupported:../crypto/store/store_result.c:151:
```

¿Qué ocurre? No se puede descifrar.

¿Por qué? El ha encriptado el mensaje con mi clave pública, y estamos intentandolo con su clave pública, para que funcionase, tendríamos que hacerlo con nuestra clave pública.

e) Ahora descifralo con la clave que no compartiste.

Ahora para descifrarlo vamos a usar nuestra propia clave privada, ya que nuestro compañero ha encriptado con nuestra pública.

```
usuario@usuario:~$ openssl rsautl -decrypt -inkey rsaprivada_eric.pem -in MensajeparaEricCifrado.txt -out mensajededanidescifrado.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
Enter pass phrase for rsaprivada_eric.pem:
usuario@usuario:~$ cat mensajededanidescifrado.txt
Eric cómeme el cimbrel
```

¿Qué pone el mensaje? “Eric cómeme el cimbrel”, ¡qué original!

2.3 Ejercicio simple de firma electrónica

Ahora vamos a firmar un documento, para ello ¿qué certificado debes utilizar?
¿Por qué?

Para verificar la firma sigue los pasos indicados en (<https://stackoverflow.com/questions/14327517/openssl-rsa-using-a-public-key-todecrypt>) como ayuda.

```
usuario@usuario:~$ openssl genrsa -out mykey
usuario@usuario:~$ ls
Descargas  mensajededanidescifrado.txt  mykey  rsa_publica.pem
Desktop    mensajeeric_cifrado.txt      Plantillas  snap
Documentos mensaje_eric.txt             Público     Videos
Escritorio mensaje_para_dani_cifrado.txt rekall      ZeroNet-linux-dist-linux64
file       MensajeparaEricCifrado.txt   rsaprivada_eric.pem ZeroNet-py3-linux64.tar.gz
Imágenes   Música                       rsapublica_eric.pem
```

```
usuario@usuario:~$ openssl rsa -in mykey -pubout -out mykey.pub
writing RSA key
usuario@usuario:~$ ls
Descargas  mensajededanidescifrado.txt  mykey  rsapublica_eric.pem
Desktop    mensajeeric_cifrado.txt      mykey.pub  rsa_publica.pem
Documentos mensaje_eric.txt             Plantillas  snap
Escritorio mensaje_para_dani_cifrado.txt Público     Videos
file       MensajeparaEricCifrado.txt   rekall      ZeroNet-linux-dist-linux64
Imágenes   Música                       rsaprivada_eric.pem ZeroNet-py3-linux64.tar.gz
```

```
usuario@usuario:~$ sudo nano myfile
[sudo] contraseña para usuario:
usuario@usuario:~$ md5sum myfile | openssl rsautl -inkey mykey -sign > checksum.singed
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
usuario@usuario:~$ ls
checksum.singed  mensajededanidescifrado.txt  mykey  rsa_publica.pem
Descargas       mensajeeric_cifrado.txt      mykey.pub  snap
Desktop         mensaje_eric.txt             Plantillas  Videos
Documentos      mensaje_para_dani_cifrado.txt Público     ZeroNet-linux-dist-linux64
Escritorio      MensajeparaEricCifrado.txt   rekall      ZeroNet-py3-linux64.tar.gz
file            Música                       rsaprivada_eric.pem
Imágenes        myfile                       rsapublica_eric.pem
usuario@usuario:~$ cat checksum.singed
@000(!M00#40}00b0B000X00Tw000It:02'0{)U0000|0I00000000F030000000N0
;000000e%X0::G00MW001'D%Jv
)00jM00c700100W00B!y?-00000000&U010&00M00rY0U00X00}00?0
M000G0Y000V=0000oe090
usuario@usuario:~$
```

```
usuario@usuario:~$ openssl rsautl -inkey mykey.pub -pubin -in checksum.singed
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
4d0699598b267981363c8c7e54370fca myfile
usuario@usuario:~$
```

2.4 Jugando con gpg.

GPG implementa un almacén de claves (una especie de PKI personal con muchas “” y *) que se ubica en `./gnupg/trustdb.gpg`. En este almacén se guardan todos los certificados de usuarios en los que se confía. Al igual que con openssl, vamos a generar nuestro par de claves con gpg:**

\$ gpg --gen-key

```
usuario@usuario:~$ gpg --gen-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Nota: Usa "gpg --full-generate-key" para el diálogo completo de generación de clave.

GnuPG debe construir un ID de usuario para identificar su clave.

Nombre y apellidos: Eric Serrano
Dirección de correo electrónico: cire78961@gmail.com
Ha seleccionado este ID de usuario:
    "Eric Serrano <cire78961@gmail.com>"

¿Cambia (N)ombre, (D)irección o (V)ale/(S)alir? V
Es necesario generar muchos bytes aleatorios. Es una buena idea realizar
alguna otra tarea (trabajar en otra ventana/consola, mover el ratón, usar
la red y los discos) durante la generación de números primos. Esto da al
generador de números aleatorios mayor oportunidad de recoger suficiente
entropía.
Es necesario generar muchos bytes aleatorios. Es una buena idea realizar
alguna otra tarea (trabajar en otra ventana/consola, mover el ratón, usar
la red y los discos) durante la generación de números primos. Esto da al
generador de números aleatorios mayor oportunidad de recoger suficiente
entropía.
gpg: /home/usuario/.gnupg/trustdb.gpg: se ha creado base de datos de confianza
gpg: clave 4EE1DCC7A14BAA4F marcada como de confianza absoluta
gpg: creado el directorio '/home/usuario/.gnupg/openpgp-revocs.d'
gpg: certificado de revocación guardado como '/home/usuario/.gnupg/openpgp-revocs.d/79C4061F9886B83
8F8D753D84EE1DCC7A14BAA4F.rev'
claves pública y secreta creadas y firmadas.
```

Activ
Ve a C

```
pub  rsa3072 2023-12-18 [SC] [caduca: 2025-12-17]
      79C4061F9886B838F8D753D84EE1DCC7A14BAA4F
uid                               Eric Serrano <cire78961@gmail.com>
sub  rsa3072 2023-12-18 [E] [caduca: 2025-12-17]

usuario@usuario:~$
```


En esta ocasión, el programa solicitará los datos necesarios para crear el par de claves: algoritmo, tamaño, etc. El ID de usuario se utiliza para asociar las claves correspondientes a cada usuario en el repositorio local. [Documenta]

Para verificar las claves que se han creado puedes usar:

\$ gpg --list-keys

```

usuario@usuario:~$ gpg --list-keys
gpg: comprobando base de datos de confianza
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: nivel: 0  validez: 1  firmada: 0  confianza: 0-, 0q, 0n, 0m, 0f, 1u
gpg: siguiente comprobación de base de datos de confianza el: 2025-12-17
/home/usuario/.gnupg/pubring.kbx
-----
pub  rsa3072 2023-12-18 [SC] [caduca: 2025-12-17]
    79C4061F9886B838F8D753D84EE1DCC7A14BAA4F
uid          [ absoluta ] Eric Serrano <cire78961@gmail.com>
sub  rsa3072 2023-12-18 [E] [caduca: 2025-12-17]

```

Activar

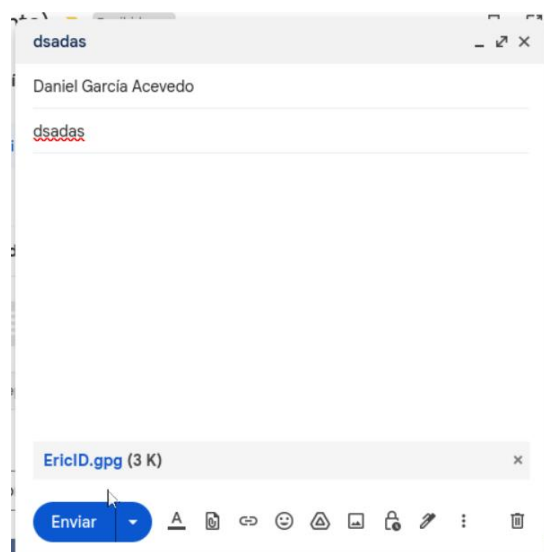
Exporta la clave pública para que la importe tu compañero.

\$ gpg --armor --output usuariID.gpg --export

```

usuario@usuario:~$ gpg --armor --output EricID.gpg --export
usuario@usuario:~$ ls
checksum.singed  Inágenes          myfile            rsapublica_eric.pem
Descargas        mensajededanidescifrado.txt  mykey            rsa_publica.pem
Desktop          mensajeeric_cifrado.txt      mykey.pub        snap
Documentos       mensaje_eric.txt            Plantillas       Videos
EricID.gpg       mensaje_para_dani_cifrado.txt  Público          ZeroNet-linux-dist-linux64
Escritorio       MensajeparaEricCifrado.txt    rekall           ZeroNet-py3-linux64.tar.gz
file             Música              rsaprivada_eric.pem

```



Una vez recibida la clave del compañero hay que importarla en nuestro almacén local:

\$gpg --import compañeroID.gpg

```
usuario@usuario:~$ ls
checksum_singed  file          Música      rsaprivada_eric.pem
DaniID.gpg       Imágenes     myfile      rsapublica_eric.pem
Descargas        mensajedanidescifrado.txt  mykey       rsa_publica.pem
Desktop          mensajeeric_cifrado.txt    mykey.pub   snap
Documentos       mensaje_eric.txt           Plantillas  Videos
EricID.gpg       mensaje_para_dani_cifrado.txt Público     ZeroNet-linux-dist-linux64
Escritorio       MensajeParaEricCifrado.txt rekall      ZeroNet-py3-linux64.tar.gz

usuario@usuario:~$ gpg --import DaniID.gpg
gpg: clave 681A5DD7068529FE: clave pública "Daniel García <dgarace2809@iesmartinezm.es>" importada
gpg: Cantidad total procesada: 1
gpg: importadas: 1
usuario@usuario:~$
```

Antes de continuar comprueba las claves de tu almacén. [Documenta]

```
usuario@usuario:~$ gpg --list-keys
/home/usuario/.gnupg/pubring.kbx
-----
pub   rsa3072 2023-12-18 [SC] [caduca: 2025-12-17]
      79C4061F9886B838F8D753D84EE1DCC7A14BAA4F
uid   [ absoluta ] Eric Serrano <cire78961@gmail.com>
sub   rsa3072 2023-12-18 [E] [caduca: 2025-12-17]

pub   rsa3072 2023-12-18 [SC] [caduca: 2025-12-17]
      58F69F3E9EBC33D0867209C9681A5DD7068529FE
uid   [desconocida] Daniel García <dgarace2809@iesmartinezm.es>
sub   rsa3072 2023-12-18 [E] [caduca: 2025-12-17]
```

Tenemos la clave del compañero en nuestro almacén de claves, pero no la hemos marcado como “de confianza”, para ello sigue los pasos indicados en

(<http://www.gnupg.org/gph/en/manual/x334.html>)

```
usuario@usuario:~$ gpg --edit-key dgarace2809@iesmartinezm.es
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

pub  rsa3072/681A5DD7068529FE
     creado: 2023-12-18  caduca: 2025-12-17  uso: SC
     confianza: desconocido  validez: desconocido
sub  rsa3072/E0567EB094C843D5
     creado: 2023-12-18  caduca: 2025-12-17  uso: E
[desconocida] (1). Daniel García <dgarace2809@iesmartinezm.es>

gpg> trust
pub  rsa3072/681A5DD7068529FE
     creado: 2023-12-18  caduca: 2025-12-17  uso: SC
     confianza: desconocido  validez: desconocido
sub  rsa3072/E0567EB094C843D5
     creado: 2023-12-18  caduca: 2025-12-17  uso: E
[desconocida] (1). Daniel García <dgarace2809@iesmartinezm.es>

Por favor, decida su nivel de confianza en que este usuario
verifique correctamente las claves de otros usuarios (mirando
pasaportes, comprobando huellas dactilares en diferentes fuentes...)
```

```
1 = No lo sé o prefiero no decirlo
2 = NO tengo confianza
3 = Confío un poco
4 = Confío totalmente
5 = confío absolutamente
m = volver al menú principal

¿Su decisión? 3

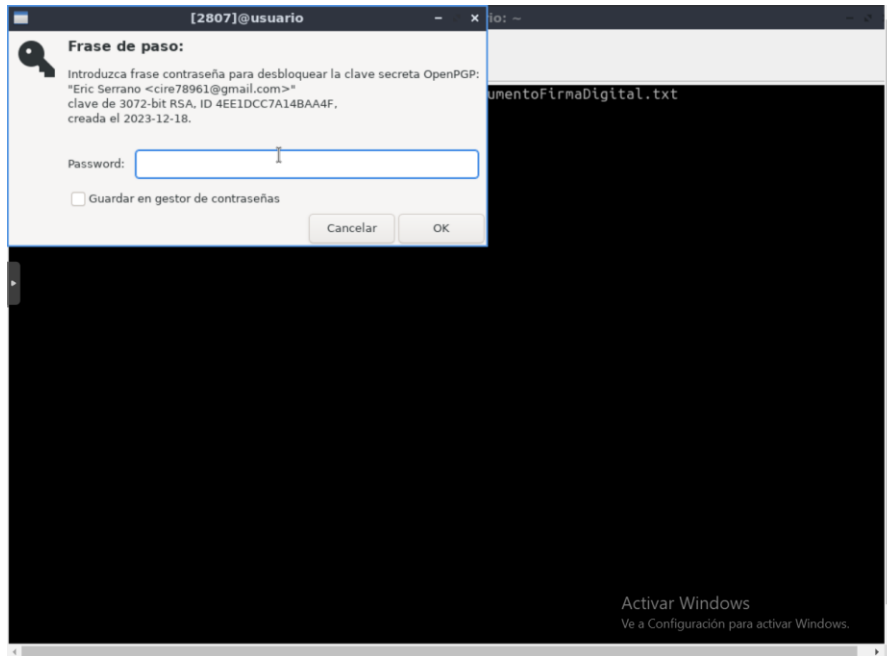
pub  rsa3072/681A5DD7068529FE
     creado: 2023-12-18  caduca: 2025-12-17  uso: SC
     confianza: dudosa  validez: desconocido
sub  rsa3072/E0567EB094C843D5
     creado: 2023-12-18  caduca: 2025-12-17  uso: E
[desconocida] (1). Daniel García <dgarace2809@iesmartinezm.es>
Ten en cuenta que la validez de clave mostrada no es necesariamente
correcta a menos de que reinicies el programa.

gpg> quit
```

Ahora genera la firma digital de un documento:

\$ gpg --armor --output [firma.sig] --sign documento.txt

```
usuario@usuario:~$ gpg --armor --output Eric --sign documentoFirmaDigital.txt
```



Envía el documento firmado a tu compañero y verifica con la opción **--verify** que el fichero es el original. Puedes usar el siguiente comando para descifrar el documento y recuperar el original:

\$ gpg --output [documento.txt] -decrypt [fiorma.sig]

```
usuario@usuario:~$ gpg --output danimensajefirmadigital.txt --decrypt firmaDani.sig
gpg: Firmado el lun 18 dic 2023 21:39:12 CET
gpg: usando RSA clave 58F69F3E9EBC33D0867209C9681A5DD7068529FE
gpg: comprobando base de datos de confianza
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: nivel: 0 validez: 1 firmada: 0 confianza: 0-, 0q, 0n, 0m, 0f, 1u
gpg: siguiente comprobación de base de datos de confianza el: 2025-12-17
gpg: Firma correcta de "Daniel García <dgarace2809@iesmartinezm.es>" [desconocido]
gpg: ATENCIÓN: ¡Esta clave no está certificada por una firma de confianza!
gpg: No hay indicios de que la firma pertenezca al propietario.
Huellas dactilares de la clave primaria: 58F6 9F3E 9EBC 33D0 8672 09C9 681A 5DD7 0685 29FE
usuario@usuario:~$ cat danimensajefirmadigital.txt
Para firmar
usuario@usuario:~$
```

Observaciones y conclusiones.

La práctica involucró la generación de claves RSA, el intercambio y uso de claves públicas y privadas para cifrado y descifrado de mensajes, la firma digital de documentos y la exploración de funcionalidades de GPG para compartir claves y verificar la autenticidad de documentos.

Se comprendió la importancia y el proceso de generación, intercambio y uso adecuado de claves en sistemas criptográficos asimétricos para mantener la confidencialidad, autenticidad e integridad de la información.

La práctica proporcionó una visión práctica y aplicada de la criptografía asimétrica, permitiendo comprender cómo se utilizan las claves pública y privada en diversas operaciones criptográficas para asegurar la comunicación segura y la autenticación de documentos.