

ACT0301 PRÁCTICA DE CIFRADO SIMÉTRICO



Contenido

1. INTRODUCCIÓN.....	2
1.1 Creación de directorio llamado 'cifrado' en el directorio home:	2
2. DESARROLLO DE LA PRÁCTICA.....	2
2.1 Familiarización con las herramientas:	2
2.2 Ejercicio de cifrado simétrico simple:	4

1. INTRODUCCIÓN

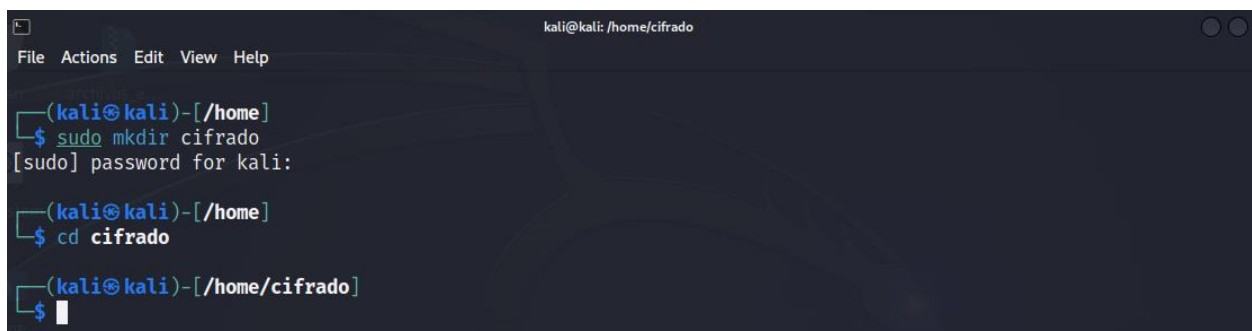
Como hemos visto en clase, existen varios algoritmos de cifrados en uso en la actualidad.

En esa práctica vamos a instalar las herramientas necesarias para practicar poder compartir mensajes cifrados.

Vamos a utilizar dos herramientas que implementan los algoritmos vistos en clase, OpenSSL y Pretty Good Privacy gpg. Ambas permiten usar cifrado tanto simétrico como asimétrico, en esta primera práctica veremos el primero de los usos.

Para empezar, vamos a crear un directorio llamado **cifrado** en el directorio **home** de cualquier máquina Linux que utilicéis habitualmente (Kali, Ubuntu, WSL, ...)

1.1 Creación de directorio llamado 'cifrado' en el directorio home:



```
kali@kali: /home/cifrado
File Actions Edit View Help

(kali@kali)-[/home]
$ sudo mkdir cifrado
[sudo] password for kali:

(kali@kali)-[/home]
$ cd cifrado

(kali@kali)-[/home/cifrado]
$
```

2. DESARROLLO DE LA PRÁCTICA

2.1 Familiarización con las herramientas:

En esta práctica vamos a hacer uso de las herramientas **gp1g** y **openssl2**. Asegúrate de que las tienes instaladas, en caso contrario instálalas y, una vez instaladas, dedica 10 minutos a leer los manuales de las dos aplicaciones.

OpenPGP es una implementación de la **RFC4880**, que es un estándar de comunicaciones seguras para el servicio de correo electrónico.

OpenSSL es una implementación de la librería usada por los protocolos **SSH** y **TLS**.

man gpg

```

GPG(1)                                GNU Privacy Guard 2.2                                GPG(1)
NAME
    gpg - OpenPGP encryption and signing tool

SYNOPSIS
    gpg [--homedir dir] [--options file] [options] command [args]

DESCRIPTION
    gpg is the OpenPGP part of the GNU Privacy Guard (GnuPG). It is a tool to provide digital encryption
    and signing services using the OpenPGP standard. gpg features complete key management and all the
    bells and whistles you would expect from a full OpenPGP implementation.

    There are two main versions of GnuPG: GnuPG 1.x and GnuPG 2.x. GnuPG 2.x supports modern encryption
    algorithms and thus should be preferred over GnuPG 1.x. You only need to use GnuPG 1.x if your plat-
    form doesn't support GnuPG 2.x, or you need support for some features that GnuPG 2.x has deprecated,
    e.g., decrypting data created with PGP-2 keys.

    If you are looking for version 1 of GnuPG, you may find that version installed under the name gpg1.

RETURN VALUE
    The program returns 0 if there are no severe errors, 1 if at least a signature was bad, and other er-
    ror codes for fatal errors.

    Note that signature verification requires exact knowledge of what has been signed and by whom it has
    been signed. Using only the return code is thus not an appropriate way to verify a signature by a
    script. Either make proper use of the status codes or use the gpgv tool which has been designed to
    make signature verification easy for scripts.

WARNINGS
    Use a good password for your user account and make sure that all security issues are always fixed on
    your machine. Also employ diligent physical protection to your machine. Consider to use a good
    passphrase as a last resort protection to your secret key in the case your machine gets stolen. It
    is important that your secret key is never leaked. Using an easy to carry around token or smartcard
    with the secret key is often a advisable.

    If you are going to verify detached signatures, make sure that the program knows about it; either
    give both filenames on the command line or use '-' to specify STDIN.

    For scripted or other unattended use of gpg make sure to use the machine-parseable interface and not
    Manual page gpg(1) line 1 (press h for help or q to quit)

```

man openssl

```

OPENSSL(1SSL)                        OpenSSL                        OPENSSL(1SSL)
NAME
    openssl - OpenSSL command line program

SYNOPSIS
    openssl command [ options ... ] [ parameters ... ]

    openssl no-XXX [ options ]

DESCRIPTION
    OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1)
    network protocols and related cryptography standards required by them.

    The openssl program is a command line program for using the various cryptography functions of OpenSSL's crypto library from the
    shell. It can be used for

    o Creation and management of private keys, public keys and parameters
    o Public key cryptographic operations
    o Creation of X.509 certificates, CSRs and CRLs
    o Calculation of Message Digests and Message Authentication Codes
    o Encryption and Decryption with Ciphers
    o SSL/TLS Client and Server Tests
    o Handling of S/MIME signed or encrypted mail
    o Timestamp requests, generation and verification

COMMAND SUMMARY
    The openssl program provides a rich variety of commands (command in the "SYNOPSIS" above). Each command can have many options
    and argument parameters, shown above as options and parameters.

    Detailed documentation and use cases for most standard subcommands are available (e.g., openssl-x509(1)). The subcommand
    openssl-list(1) may be used to list subcommands.

    The command no-XXX tests whether a command of the specified name is available. If no command named XXX exists, it returns 0
    (success) and prints no-XXX; otherwise it returns 1 and prints XXX. In both cases, the output goes to stdout and nothing is
    printed to stderr. Additional command line arguments are always ignored. Since for each cipher there is a command of the same
    name, this provides an easy way for shell scripts to test for the availability of ciphers in the openssl program. (no-XXX is
    not able to detect pseudo-commands such as quit, list, or no-XXX itself.)

Configuration Option
    Manual page openssl(1ssl) line 1 (press h for help or q to quit)

```

2.2 Ejercicio de cifrado simétrico simple:

Lo primero que haremos será crear un fichero de texto plano llamado M.txt en el que se habrá que escribir un mensaje en texto plano (elige el que quieras), este será nuestro mensaje en claro M.

```
kali@kali: /home/cifrado
File Actions Edit View Help

(kali@kali)-[/home/cifrado]
$ sudo touch M.txt

(kali@kali)-[/home/cifrado]
$ sudo nano M.txt

(kali@kali)-[/home/cifrado]
$ cat M.txt
Eric Serrano Marín
```

Comprueba su tamaño y su contenido, para el contenido vamos a ver su contenido de texto (cat o more) y su contenido en hexadecimal (od o hexdump)

```
kali@kali: /home/cifrado
File Actions Edit View Help

(kali@kali)-[/home/cifrado]
$ more M.txt
Eric Serrano Marín
Contento con more

(kali@kali)-[/home/cifrado]
$ ls -lh
total 4.0K
-rw-r--r-- 1 root root 20 Dec  5 06:41 M.txt
Tamaño con ls -lh

(kali@kali)-[/home/cifrado]
$ od M.txt
0000000 071105 061551 051440 071145 060562 067556 046440 071141
0000020 126703 005156
0000024
Contentos exadecimales con od y hexdump

(kali@kali)-[/home/cifrado]
$ hexdump M.txt
0000000 7245 6369 5320 7265 6172 6f6e 4d20 7261
0000010 adc3 0a6e
0000014

(kali@kali)-[/home/cifrado]
$
```

Vamos a empezar usando gpg, para ello comprueba la versión que estás usando

```
kali@kali: /home/cifrado
File Actions Edit View Help
(kali@kali)-[/home/cifrado]
$ gpg --version
gpg (GnuPG) 2.2.40
libgcrypt 1.10.2
Copyright (C) 2022 g10 Code GmbH
License GNU GPL-3.0-or-later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /home/kali/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
```

Vamos allá con nuestro primer cifrado. Antes de intentarlo piensa, ¿qué nos hará falta definir para poder cifrar el mensaje?

[Invierte 1 minuto en esto antes de seguir leyendo]

¡Correcto! Necesitamos el mensaje, la clave de cifrado y el algoritmo que vamos a utilizar para cifrarlo. Pues vamos al lío, cifra M con el algoritmo 3DES.

```
(kali@kali)-[/home/cifrado]
$ sudo gpg --cipher-algo 3DES -c M.txt
gpg: directory '/root/.gnupg' created
gpg: keybox '/root/.gnupg/pubring.kbx' created
```

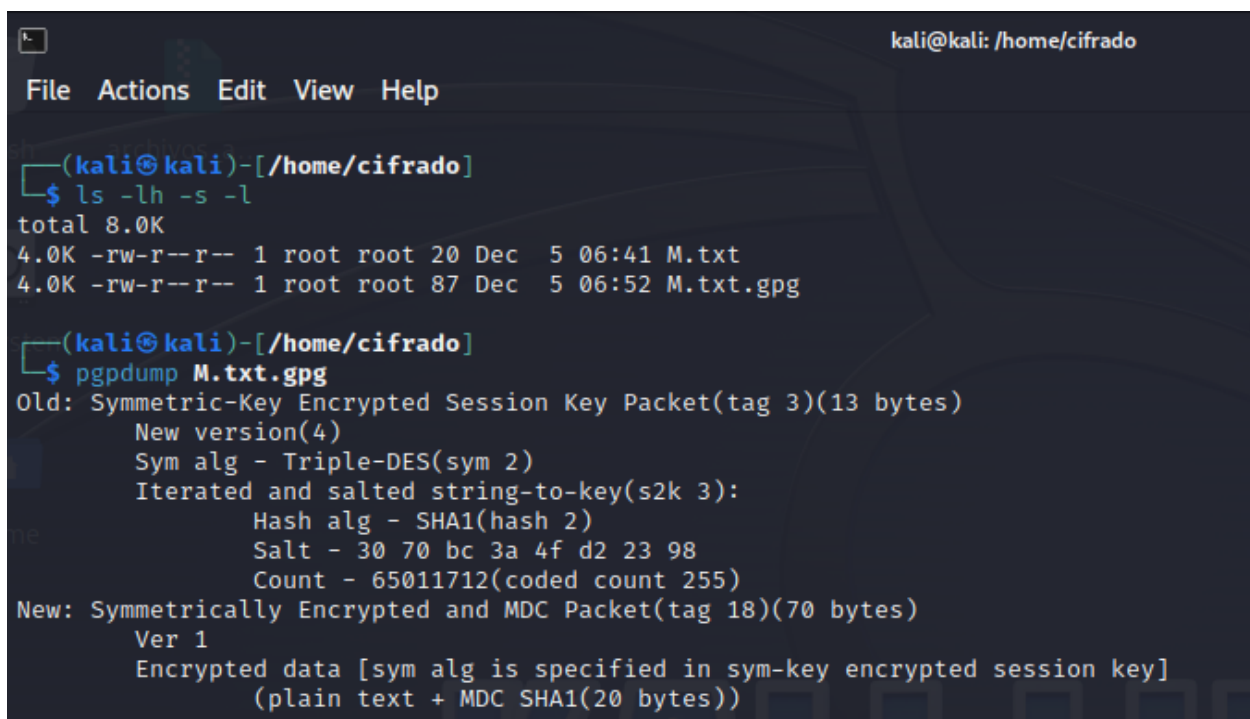
Si haces un ls verás que se ha creado un nuevo archivo ¿qué extensión tiene?

Tiene la extensión. gpg.

¿Tiene el mismo tamaño que M? Sí, tienen el mismo tamaño.

Por un lado, pgp comprime el fichero y por otro ¿Sabrías decir por qué? [Pista: la rfc4880 tiene la solución]: Comprime archivos al cifrarlos para mejorar la eficiencia y velocidad de transmisión. La compresión reduce el tamaño del mensaje cifrado. La información adicional, como la compresión, se gestiona mediante paquetes PGP. La especificación RFC4880 proporciona detalles específicos sobre cómo se aplica la compresión y qué datos se añaden al mensaje cifrado.

Una vez encuentres qué datos se añaden al mensaje cifrado, utiliza pgpdump para mostrarlos: Paquetes que contienen información adicional junto con los datos cifrados. PGP usa varios tipos de paquetes, y uno de ellos es el paquete de compresión.



```
kali@kali: /home/cifrado
File Actions Edit View Help

(kali@kali)-[/home/cifrado]
$ ls -lh -s -l
total 8.0K
4.0K -rw-r--r-- 1 root root 20 Dec  5 06:41 M.txt
4.0K -rw-r--r-- 1 root root 87 Dec  5 06:52 M.txt.gpg

(kali@kali)-[/home/cifrado]
$ pgpdump M.txt.gpg
Old: Symmetric-Key Encrypted Session Key Packet(tag 3)(13 bytes)
  New version(4)
  Sym alg - Triple-DES(sym 2)
  Iterated and salted string-to-key(s2k 3):
    Hash alg - SHA1(hash 2)
    Salt - 30 70 bc 3a 4f d2 23 98
    Count - 65011712(coded count 255)
New: Symmetrically Encrypted and MDC Packet(tag 18)(70 bytes)
  Ver 1
  Encrypted data [sym alg is specified in sym-key encrypted session key]
    (plain text + MDC SHA1(20 bytes))
```


Ahora prueba a cifrar el mensaje haciendo uso de AES y comprueba lo mismo que para el caso 3DES.

```
kali@kali: /home/cifrado
File Actions Edit View Help

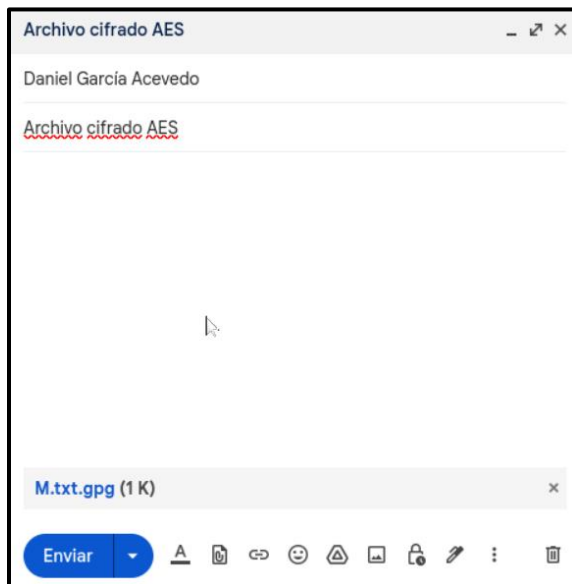
(kali@kali)-[/home/cifrado]
$ sudo gpg --cipher-algo AES -c M.txt
File 'M.txt.gpg' exists. Overwrite? (y/N) y

(kali@kali)-[/home/cifrado]
$ ls -lh -s -l
total 8.0K
4.0K -rw-r--r-- 1 root root 20 Dec  5 06:41 M.txt
4.0K -rw-r--r-- 1 root root 95 Dec  5 06:58 M.txt.gpg

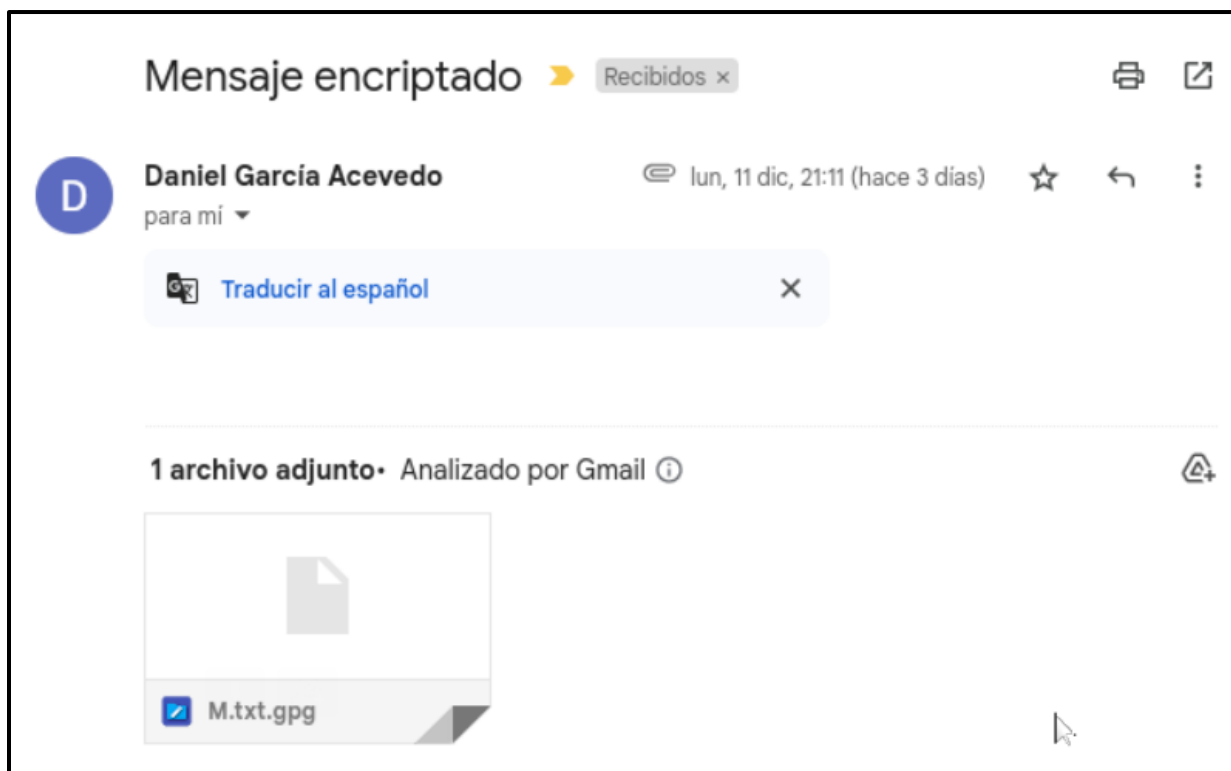
(kali@kali)-[/home/cifrado]
$ pgpdump M.txt.gpg
Old: Symmetric-Key Encrypted Session Key Packet(tag 3)(13 bytes)
  New version(4)
  Sym alg - AES with 128-bit key(sym 7)
  Iterated and salted string-to-key(s2k 3):
    Hash alg - SHA1(hash 2)
    Salt - 78 d0 8f 49 0b 75 18 53
    Count - 65011712(coded count 255)
New: Symmetrically Encrypted and MDC Packet(tag 18)(78 bytes)
  Ver 1
  Encrypted data [sym alg is specified in sym-key encrypted session key]
    (plain text + MDC SHA1(20 bytes))
```


Ahora que ya sabes cómo cifrar un mensaje, intercambia el mensaje cifrado con AES con tu compañero e intenta descifrar el que te ha pasado el compañero.

Enviando archivo a Daniel.

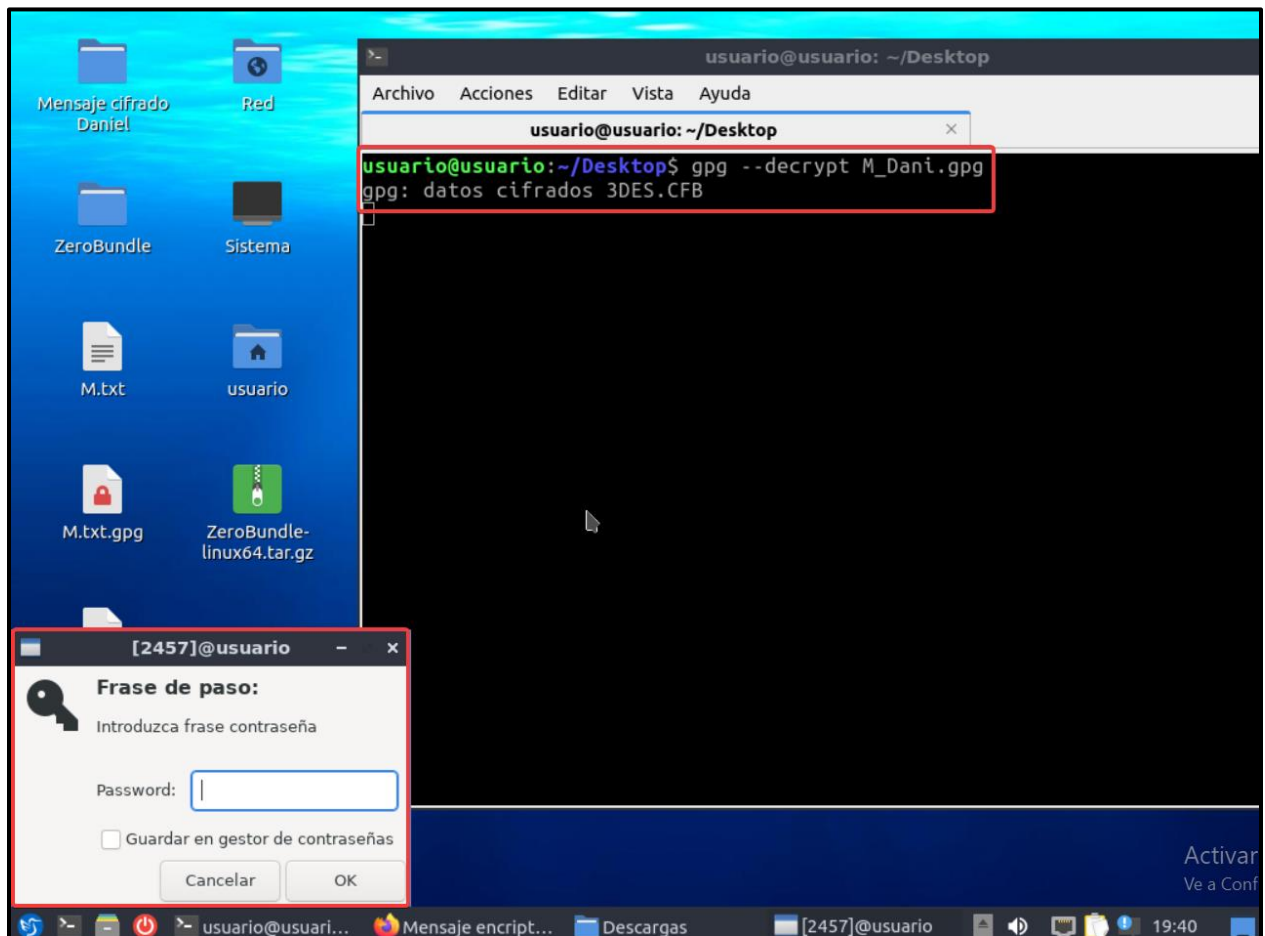


Archivo que Daniel me ha enviado.



Descifrando el archivo de Dani.

gpg --decrypt M_Dani.gpg y después ponemos la contraseña que nos ha dado el compañero.



Resultado



¿Qué necesitas para descifrar el mensaje? ¿Y tú compañero? Proponed algún sistema que os permita compartir la información necesaria en un entorno no fiable.

Hemos necesitado la contraseña que hemos usado en los cifrados y la hemos compartido la contraseña en persona.

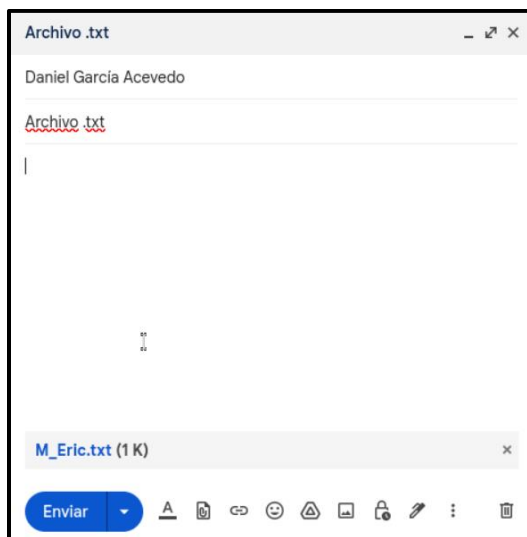
El fichero generado en el proceso de cifrado es un fichero binario, ¿qué opciones debes usar para poder enviar el mensaje como texto de un correo electrónico?

Podemos pasarlo a base 64 y meterlo en un .txt.

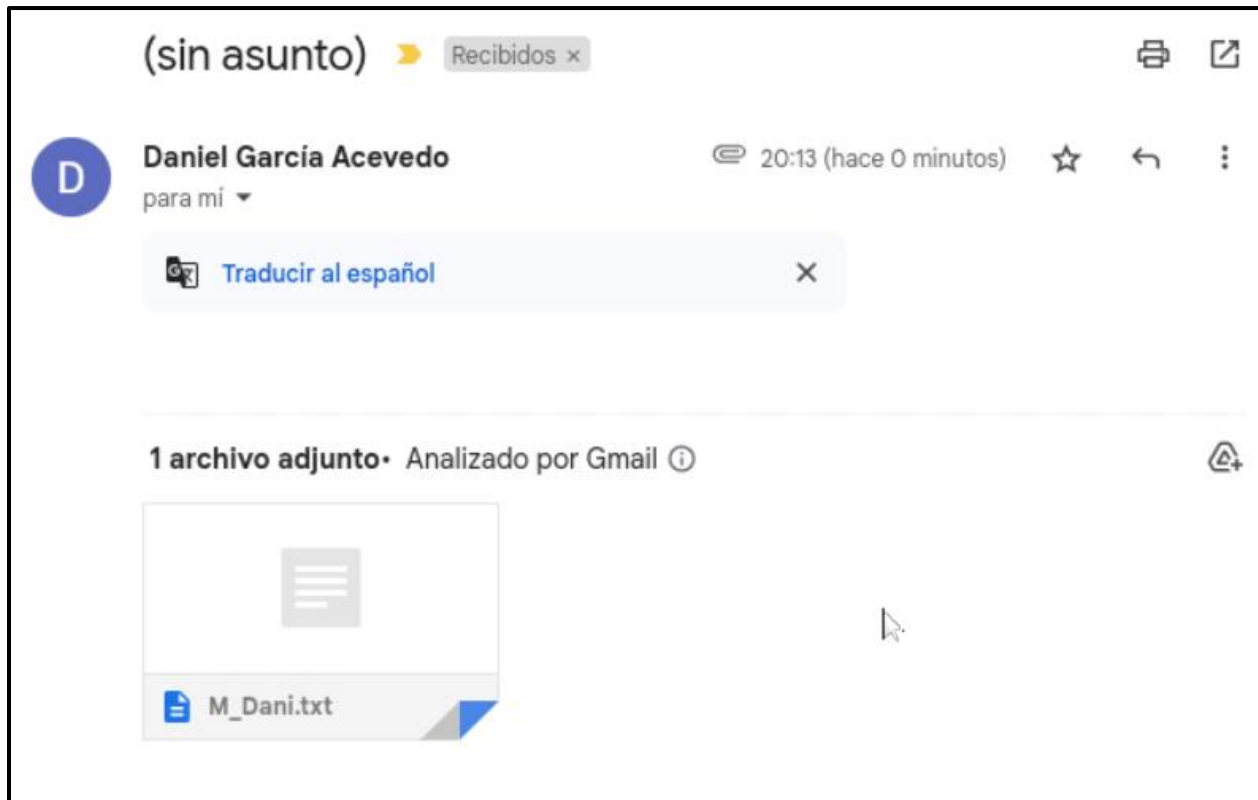
```
usuario@usuario:~/Desktop$ base64 M.txt.gpg > M_Eric.txt
usuario@usuario:~/Desktop$ cat M_Eric.txt
jA0EBwMCV+Mocx0eRNb/0k4Btt6NmPLq0K0iue5Yic9qDTE2bI+bvls3qB2InMPauyeL9MgPoGbi
ytcVC1XvxM0j5gPH21EhpGXvSIZHoqZW42TDRecx+Ip09JQhQSA=
usuario@usuario:~/Desktop$
```

Realiza el proceso y comparte con tu compañero un correo con el texto del mensaje. Descifradlo a partir del correo.

Pasamos el archivo de texto al compañero.



Este es el correo que me ha pasado el compañero.



Ahora vamos a proceder a decodificarlo y descifrarlo. Hemos hecho la práctica distinto día, por ello mi compañero ha tenido que crear otro archivo el día en el que hemos acabado la práctica, de ahí que su texto del mensaje ha cambiado.

```
M.txt.gpg          tor-browser-linux64-12.5.6_ALL.tar.xz
usuario@usuario:~/Descargas$ base64 -d M_Dani.txt > mensajeDani_recibido.gpg
usuario@usuario:~/Descargas$ ls
M_Dani.gpg          rsa_publica.pem
M_Dani.txt          tor-browser
mensaje_cifrado.txt 'tor-browser-linux64-12.5.6_ALL.tar(1).xz'
mensajeDani_recibido.gpg tor-browser-linux64-12.5.6_ALL.tar.xz
M.txt.gpg
usuario@usuario:~/Descargas$ gpg --decrypt mensajeDani_recibido.gpg
gpg: datos cifrados AES.CFB
gpg: cifrado con 1 frase contraseña
Ahora es por SSL
usuario@usuario:~/Descargas$
```

Ahora pasemos a comprobar el rendimiento de los distintos algoritmos de cifrado utilizando la propia funcionalidad de openssl, el comando speed de openssl permite realizar una comparación de los distintos cifradores usando cada uno durante 3 segundos.

Para medir las prestaciones de uno de los cifradores basta con escribir detrás del comando speed el cifrador que nos interese. Haz una prueba con los siguientes: AES(128), AES(256), DES, 3DES de dos claves. Utiliza para todo el mismo modo de operación: CBC.

```

usuario@usuario:~$ openssl speed aes-128-cbc aes-256-cbc des des-ede3
Doing des-ede3 for 3s on 16 size blocks: 6825099 des-ede3's in 2.99s
Doing des-ede3 for 3s on 64 size blocks: 1719244 des-ede3's in 3.00s
Doing des-ede3 for 3s on 256 size blocks: 431823 des-ede3's in 3.00s
Doing des-ede3 for 3s on 1024 size blocks: 110444 des-ede3's in 3.00s
Doing des-ede3 for 3s on 8192 size blocks: 13947 des-ede3's in 3.00s
Doing des-ede3 for 3s on 16384 size blocks: 7032 des-ede3's in 3.00s
Doing aes-128-cbc for 3s on 16 size blocks: 34736853 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 64 size blocks: 9826163 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 256 size blocks: 2498982 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 1024 size blocks: 1209624 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 8192 size blocks: 159658 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 16384 size blocks: 79392 aes-128-cbc's in 3.00s
Doing aes-256-cbc for 3s on 16 size blocks: 25919348 aes-256-cbc's in 3.00s
Doing aes-256-cbc for 3s on 64 size blocks: 6938655 aes-256-cbc's in 3.00s
Doing aes-256-cbc for 3s on 256 size blocks: 1757425 aes-256-cbc's in 3.00s
Doing aes-256-cbc for 3s on 1024 size blocks: 887853 aes-256-cbc's in 3.00s
Doing aes-256-cbc for 3s on 8192 size blocks: 112386 aes-256-cbc's in 3.00s
Doing aes-256-cbc for 3s on 16384 size blocks: 56568 aes-256-cbc's in 3.00s
version: 3.0.2
built on: Wed May 24 17:12:55 2023 UTC
options: bn(64,64)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -Wa,--noexecstack -g -O2 -ffile-prefix-map=/build/openssl-Z1YLMC/openssl-3.0.2=. -flto=auto -ffat-lto-objects -flto=auto -ffat-lto-objects -fstack-protector-strong -Wformat -Werror=format-security -DOPENSSL_TLS_SECURITY_LEVEL=2 -DOPENSSL_USE_NODELETE -DL_ENDIAN -DOPENSSL_PIC -DOPENSSL_BUILDING_OPENSSL -DNDEBUG -Wdate-time -D_FORTIFY_SOURCE=2
CPUINFO: OPENSSL_ia32cap=0x80202001078bffff:0x0
The 'numbers' are in 1000s of bytes per second processed.
type           16 bytes      64 bytes      256 bytes    1024 bytes    8192 bytes
16384 bytes
des-cbc          0.00          0.00          0.00          0.00          0.00
0.00
des-ede3        36522.27k       36677.21k       36848.90k       37698.22k       38084.61k
38404.10k
aes-128-cbc     185263.22k      209624.81k      213246.46k      412884.99k      435972.78k
433586.18k
aes-256-cbc     138236.52k      148024.64k      149966.93k      303053.82k      306888.70k
308936.70k
4017380FD37F0000:error:0308010C:digital envelope routines:inner_evp_generic_fetch:unsupported:../crypto/evp/evp_fetch.c:349:Global default library context

```

¿Podría el ordenador en el que has ejecutado la prueba generar un mensaje cifrado de 1MB en menos de un segundo?

➤ DES-ede3 (Triple DES):

Velocidad máxima: 38.404,10 KB por segundo (para bloques de 16.384 bytes)

Incapaz de cifrar un mensaje de 1MB en menos de un segundo debido a su menor velocidad máxima.

➤ AES-128-CBC:

Velocidad máxima: 435.972,78 KB por segundo (para bloques de 8.192 bytes)

Capaz de cifrar un mensaje de 1MB en menos de un segundo debido a su alta velocidad.

➤ AES-256-CBC:

Velocidad máxima: 308.936,70 KB por segundo (para bloques de 8.192 bytes)

Capaz de cifrar un mensaje de 1MB en menos de un segundo debido a su alta velocidad, aunque ligeramente más lento que AES-128-CBC.

Además del comando speed podemos recurrir a medir manualmente la velocidad del algoritmo. Para eso vamos a crear un fichero de tamaño controlado y vamos a doblando su tamaño en cada iteración. Sigue los siguientes pasos:

1. Baja de github el fichero alice.txt basta con buscarlo en Google.

2. Para ir doblando el tamaño del fichero iremos ejecutando:

```
$ echo `cat alice.txt` >> alice.txt
```

3. Cada vez que el fichero engorde significativamente mide el tiempo de cifrado haciendo uso del comando time.

Sin doblar el archivo pesa 4,0k.

```
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ l
s -lh
total 4,0K
-rw-rw-r-- 1 usuario usuario 1,3K dic 11 2011 alice.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$
```

Primer encriptado con archivo sin engordar (4,0K)

```
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ time openssl enc -aes-128-cbc -e -in alice.txt -out alice_encrypted.txt
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

real    0m5,873s
user    0m0,004s
sys     0m0,000s
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$
```

Engordamos el archivo por primera vez. (8,0K)

```
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ ls -lh
total 8,0K
-rw-rw-r-- 1 usuario usuario 5,4K dic 15 11:35 alice.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ time openssl enc -aes-128-cbc -e -in alice.txt -out alice_encrypted.txt
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

real    0m4,239s
user    0m0,000s
sys     0m0,004s
```

Engordamos por segunda vez. (12,0K)

```
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ cat alice.txt >> alice_tmp.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ cat alice_tmp.txt >> alice.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ rm alice_tmp.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ ls -lh
ls: no se puede acceder a 'lh': No existe el archivo o el directorio
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ ls -lh
total 20K
-rw-rw-r-- 1 usuario usuario 5,4K dic 15 11:37 alice_encrypted.txt
-rw-rw-r-- 1 usuario usuario 11K dic 15 11:38 alice.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ rm alice_encrypted.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ ls -lh
total 12K
-rw-rw-r-- 1 usuario usuario 11K dic 15 11:38 alice.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ time openssl enc -aes-128-cbc -e -in alice.txt -out alice_encrypted.txt
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

real    0m4,639s
user    0m0,004s
sys     0m0,000s
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$
```


Engordamos por tercera vez (24,0K)

```

usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ cat alice.txt >> alice_
tmp.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ cat alice_tmp.txt >> al
ice.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ rm alice_tmp.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ ls -lh
total 36K
-rw-rw-r-- 1 usuario usuario 11K dic 15 11:39 alice_encrypted.txt
-rw-rw-r-- 1 usuario usuario 22K dic 15 11:40 alice.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ rm alice_encrypted.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ ls -lh
total 24K
-rw-rw-r-- 1 usuario usuario 22K Dec 15 11:40 alice.txt
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$ time openssl enc -aes-1
28-cbc -e -in alice.txt -out alice_encrypted.txt
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

real    0m3.788s
user    0m0.000s
sys     0m0.004s
usuario@usuario:~/Desktop/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd$

```

Repita el proceso hasta que el fichero tenga aproximadamente 20MB

Documenta el proceso y responde a la siguiente pregunta ¿es coherente el resultado obtenido con el del uso del comando speed?

No, no son coherentes, ya que, aunque ambas mediciones están relacionadas con el cifrado, son tipos diferentes de mediciones y no se pueden comparar directamente sin una metodología específica para relacionar las velocidades de cifrado con los tiempos de ejecución.

Realiza el documento correspondiente a la práctica y no olvides incluir las observaciones y conclusiones que estimes oportunas.

Durante la práctica, hemos explorado herramientas de cifrado como gpg y openssl, comprendiendo su utilidad en la seguridad de la comunicación mediante algoritmos de cifrado simétrico. Aprendimos a realizar operaciones de cifrado y descifrado utilizando diferentes algoritmos como 3DES y AES, observando diferencias en la extensión y contenido de los archivos cifrados. Además, exploramos el intercambio seguro de información cifrada entre pares y evaluamos el rendimiento de los algoritmos mediante la herramienta OpenSSL, así como la medición manual de la velocidad del cifrado.