Práctica 2
Cifrados Asimétricos

1. **Generad, cada uno de vosotros, una clave RSA (que contiene el par de claves) de 768 bits. Para referirnos a ella supondre que se llama nombreRSAkey.pem. Esta clave no es necesario que esté protegida por contraseña.**

openssl genrsa 768 -out rafaRSAKey.pem

Esto nos genera el archivo con la clave rafaRSAKey.pem

```
Generating RSA private key, 768 bit long modulus
.................++++++++
.........++++++++
e is 65537 (0x10001)
-----BEGIN RSA PRIVATE KEY-----
MIIBygIBAAJhALxAqtCOXRI8uqYSSb2GVzB1tqYye5PC5m+7ljzt8R2Ec5xUZ7yT
crr0Gu5bt52Zfw7nJt/tIJBP5HycfkoBbgRlr7Nx8xUcJkj7Dh6s4e+gduFNyoil
R0yFOxS23szIsQIDAQABAmBugmfD1k68jfnj04iGQX6da/y0w53Xt+oeXThsvCF2
+nIDq2jaQpRvOy00D5NSaPAzgx3U277N/waUCOWU44xwf1uhVKOOYw+o0qBW3+c9
bmzjec4T/1RGc34hvasDfKECMQDl5ukl5ToBgWyoKf2wxYl592SmaK16JH//Z5Ml
ripM18FhWfP/tfrNIPwW6yTvF4sCMQDRn2aA9DuqqX5bc3MlLWeFghGN3AtLZHjM
1bHz76ovfulxba3hDbF/Et/vSn8kSDMCMFNSRN1/Hm69QWwRC+itLw4NI2VMOtMx
t1IRb7sQTDRdXbCysUPT6Hm/op2Ly2770QIwG2Obbg/7V9Irm2FbUyk6lHwpNQLi
SuiGzCTDyp6u+uO3NsXg6T3zyim6LrbqI+B3AjEAzgITatzkvvgCJQuDVGt6b+eJ
xJeySCDIwnzXlq3v2DPg2zqgIaikOuIYIRNoKQ2A
-----END RSA PRIVATE KEY-----
```

2. **Extraed" la clave privada contenida en el archivo nombreRSAkey.pem a otro archivo que tenga por nombre nombreRSApriv.pem. Este archivo debera estar protegido por contraseña cifrandolo con AES-128. Mostrad sus valores**

openssl rsa -in rafaRSAKey.pem -aes-128-cbc -out rafaPRIV.pem

Nos pedirá la contraseña, y obtenemos el archivo cifrado privado, rafaPRIV.pem, cifrado con esa contraseña:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,1FB925F866EDA38D7761694DBEE94559
```

yVCI7OiSHJBLgWPsDe1nfGS3066HeXGz41ro5Etid9DAEEiRc4l+RMZ6uHGvC8TE
3z7VBs/EfPpCc3XZHSAvOPU1d9XVP1EYly0mEr75qEwebOd2vWjCFcCLADzgM2Hh
JbTXymbTFrIEKW6Px4ApTwa8NPoc3bim1W0I4VgCUGwPhaYrF6N1i2i63o0cgM+A
qUjKF5Tf1KgZk7+jnZw3ToJWHwo6/2qkRPPdj9/kvnFPU6k4I+U3hp2PAAsjOUhZ
hisL9r2hJyqzqaKvlxE+TMXVSGDJZq1Zx7slxInWtLsNm8pbRma6kNOmKSd/Kk1p
VIHeZTMKg271b1tJgaskCRCDsQjHYRdv2pvjEdtnuqACrMw+0Og1vOj+GsURnSTe
doTTfBa53ZNlZzw8fPObhsYx1nQmm7BqNwgkrqZZB+uPlAPZG9zZZvcKQ7xIYOx/
zO2rx/n8Zr2s11EVkEHs0fX4/3Xm35/hTNnTDm3Mdp06iU/0mYsQXZH1soa1Jk5G
pCCQHcIXqzi7CyiUoO2sdwxYDKhty1iVGS0jWe2qHR2msvFci1isqs1dmQaB1G+C
7pGFQ/BMqAVjmOuwKczPqZKAWTB7SNIonuxQH9f8xmwZogH/QFjeTIi7vu6n67ia
-----END RSA PRIVATE KEY-----

CONTRASEÑA: 1234

3. Extraed en nombreRSApub.pem la clave pu blica con-tenida en el archivo nombreRSAkey.pem. Evidentemente nombreRSApub.pem no debe estar cifrado ni protegido. Mostrad sus valores.

openssl rsa -in rafaRSAKey.pem -pubout -out rafaRSApub.pem

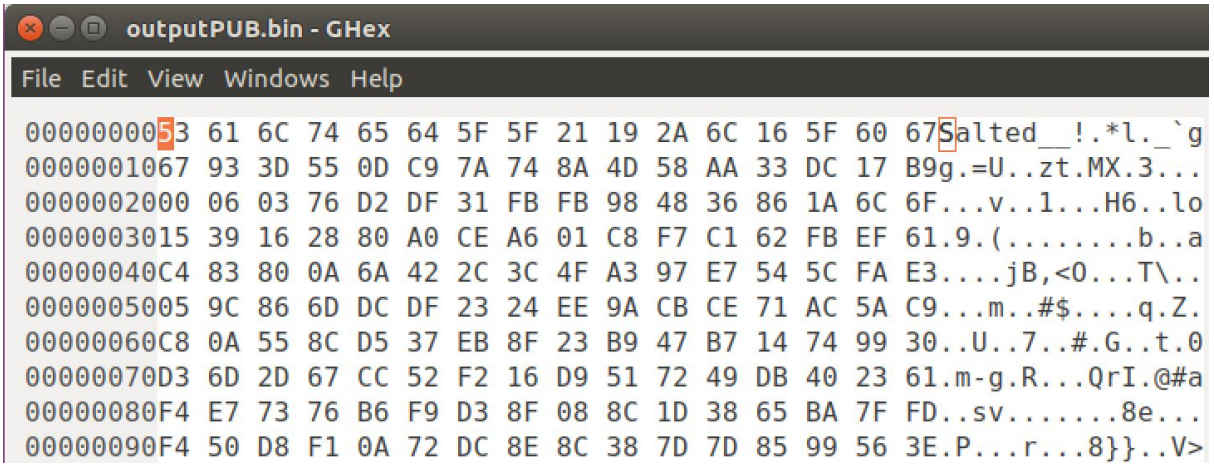
```
rafa@rafa-virtual-machine:~/Escritorio/spsi/p2$ cat rafaRSApub.pem
-----BEGIN PUBLIC KEY-----
MHwwDQYJKoZIhvcNAQEBBQADawAwaAJhAORJ5E4qM7t/5fhR1mNrJD/Uy23dnPtE
ZKbdKfbo3s/LX0RfQzMfVmICdVPNj4vBggriztMd5OR7EFop0zunvLvyvKDOGpWK
e6I3rFl6VtVOsXmK/LF07FbjSDGmWjxB0QIDAQAB
-----END PUBLIC KEY-----
```

4. Reutilizaremos el archivo binario input.bin de 1024 bits, todos ellos con valor 0, de la práctica anterior

```
rafa@rafa-virtual-machine:~/Escritorio/spsi/p2$ xxd input.bin
00000000: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000010: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000020: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000030: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000040: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000050: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000060: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000070: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000080: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000090: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000000a0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000000b0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000000c0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000000d0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000000e0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000000f0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000100: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000110: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000120: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000130: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000140: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000150: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000160: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000170: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
```

5. Intentad cifrar input.bin con vuestras claves pública. Explicad el resultado.

openssl aes-128-cbc -in input.bin -pass file:rafaRSApub.pem -out outputPUB.bin



Aquí hacemos el cifrado del archivo input.bin con el método de cifrado AES-128 en modo CBC, que recibirá como llave de cifrado la clave pública RSA que generamos previamente.

6. Diseñad un cifrado híbrido, con RSA como criptosistema asimétrico. El modo de proceder ser a el siguiente:

6.a. El emisor debe seleccionar un sistema simétrico con su correspondiente modo de operación.

Usaremos **AES-128-CBC.**

6.b. El emisor generará un archivo de texto, llamado por ejemplo sessionkey con dos líneas. La primera línea contendrá una cadena aleatoria hexadecimal cuya longitud sea la requerida por la clave. OpenSSL permite generar cadenas aleatorias con el comando openssl rand. La segunda línea contendrá la información del criptosistema simétrico seleccionado. Por ejemplo, si hemos decidido emplear el algoritmo Blow sh en modo ECB, la segunda línea deberá contener -bf-ecb.

openssl rand -hex 32 -out sessionkey.txt

echo -aes-128-cbc >> sessionkey.txt

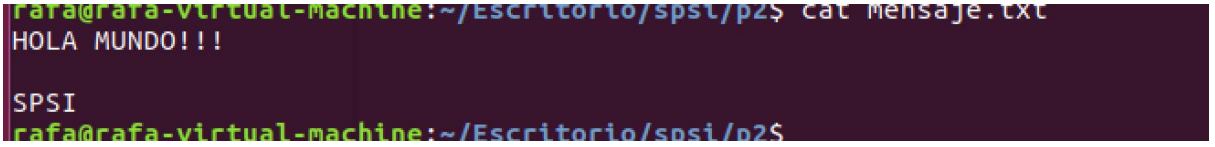
En nuestro caso, hemos usado aes-128-cbc, por lo que generamos una clave de longitud variable, hemos optado por 32 bits.

6.c. El archivo sessionkey se cifrará con la clave pública del receptor.

openssl rsautl -encrypt -inkey rafaRSApub.pem -pubin -in sessionkey.txt -out sessionkey.bin

6.d. El mensaje se cifrará utilizando el criptosistema simétrico, la clave se generará partir del archivo anterior mediante la opción -pass file:sessionkey

Como ejemplo, he creado un archivo de prueba sin cifrar en texto plano. Mensaje.txt:

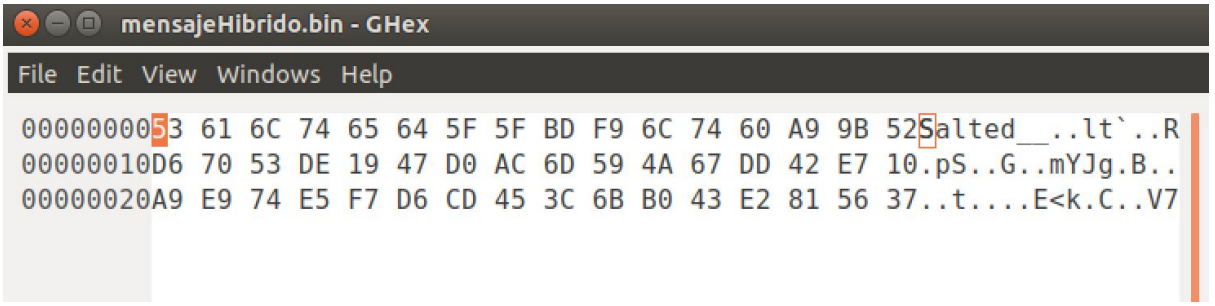


Ciframos como antes:

Cifrar con la sessionkey.txt

openssl enc -aes-128-cbc -in mensaje.txt -pass file:sessionkey.txt -out mensajeHibrido.bin

Mostramos con GHEX el resultado del mensaje cifrado:



Probamos el desencriptado:

Desencriptamos la clave
openssl rsautl -decrypt -inkey rafaPRIV.pem -in sessionkey.bin -out sessionkey2.txt

Desencriptamos el mensaje
openssl enc -d -aes-128-cbc -in mensajeHibrido.bin -out mensajeHibrido.txt -pass file:sessionkey2.txt

7. Utilizando el criptosistema híbrido diseñado, cada uno debe cifrar el archivo input.bin con su clave pública para, a continuación, descifrarlo con la clave privada. comparad el resultado con el archivo original.

Por simplicidad, hemos creado un script que automatiza todo el proceso anterior:

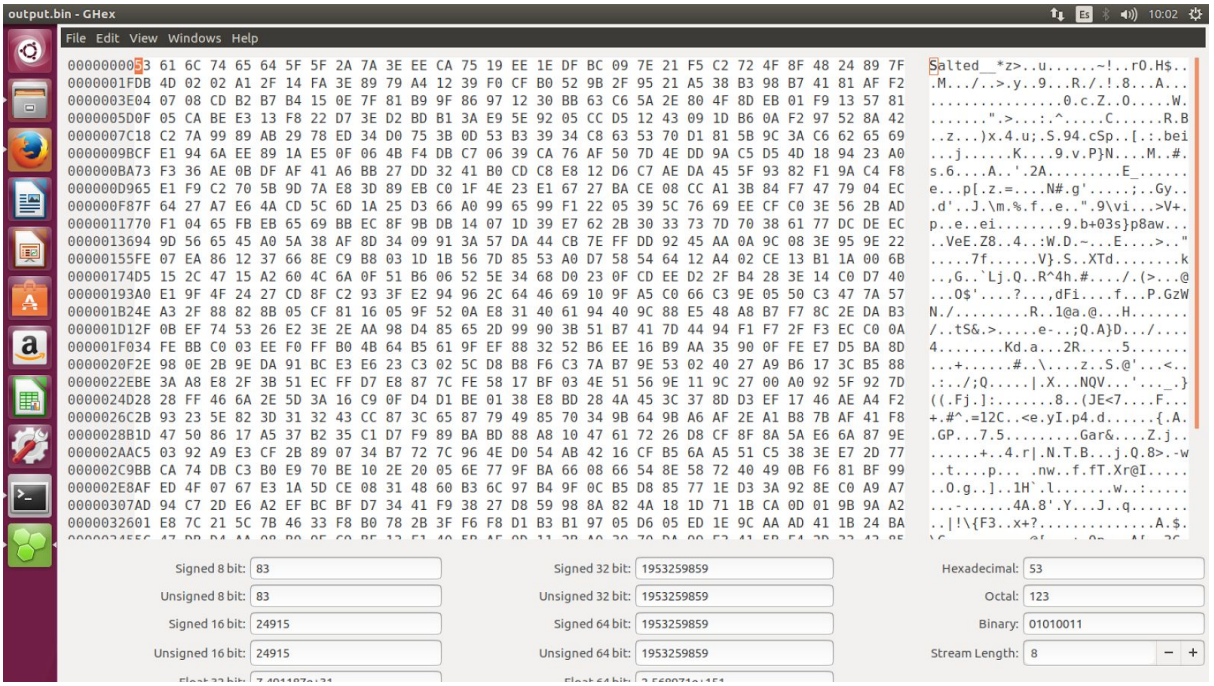
#generar clave aleatoria
openssl rand -hex 32 -out randomkey.txt

echo -aes-128-cbc >> randomkey.txt

#Cifrar clave, con la clave publica
openssl rsautl -encrypt -inkey rafaRSApub.pem -pubin -in randomkey.txt -out randomkey.enc

#Finalmente cifremos el mensaje
openssl enc -aes-128-cbc -in input.bin -out output.bin -pass file:randomkey.txt

Resultado del archivo cifrado output.bin:



Desencriptado

Primero, desencriptamos la llave, y después mediante la llave, desencriptamos el mensaje:

Desencriptamos la llave cifrada, con la clave privada. Nos pide la contraseña, en este caso 1234.

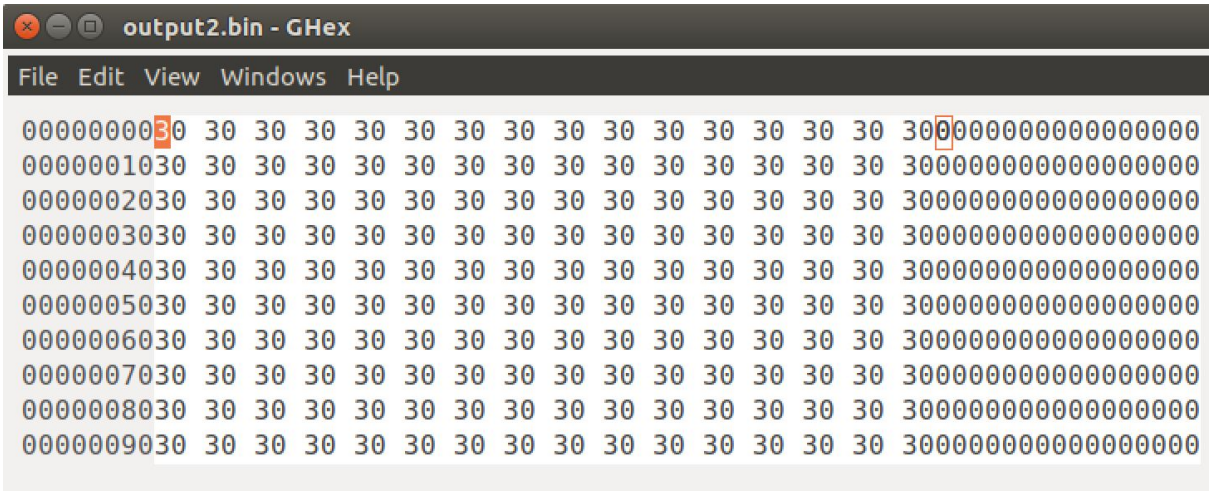
openssl rsautl -decrypt -inkey rafaPRIV.pem -in randomkey.enc -out randomkey2.txt

Con la clave ya descifrada, podemos descifrar el archivo cifrado, output.bin:

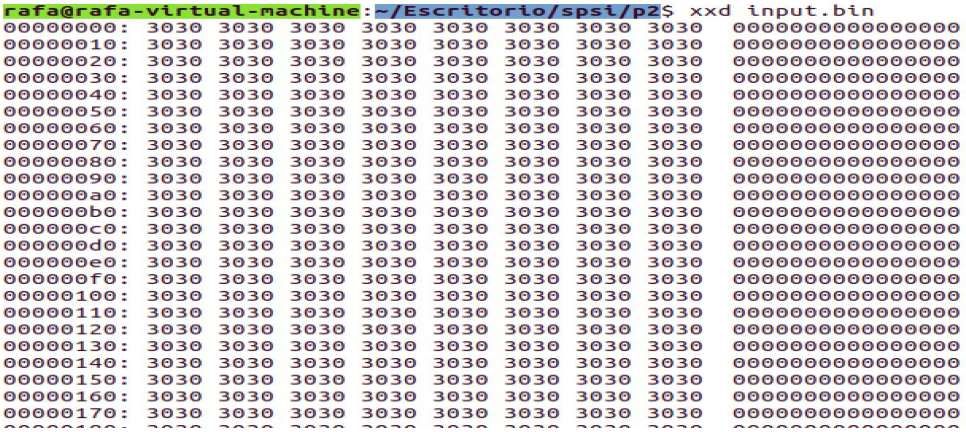
openssl enc -d -aes-128-cbc -in output.bin -out output2.bin -pass file:randomkey2.txt

```
rafa@rafa-virtual-machine:~/Escritorio/spsi/p2$ openssl rsautl -decrypt -inkey r
afaPRIV.pem -in randomkey.enc -out randomkey2.txt
Enter pass phrase for rafaPRIV.pem:
rafa@rafa-virtual-machine:~/Escritorio/spsi/p2$ openssl enc -d -aes-128-cbc -in
output.bin -out output2.bin -pass file:randomkey2.txt
```

Archivo output2.bin



Comparamos y vemos que es el mismo resultado que input.bin



8. Generad un archivo stdECparam.pem que contenga los parámetros públicos de una de las curvas elípticas contenidas en las transparencias de teoría. Si no lográis localizarlas haced el resto de la práctica con una curva cualquiera a vuestra elección de las disponibles en OpenSSL. Mostrad los valores.

Usaremos la curva de 128 bits, **sec128r1**

openssl ecparam -name secp128r1 -out stdECparam.pem

```
rafa@rafa-virtual-machine:~/Escritorio/spsi/p2$ openssl ecparam -name secp128r1
-out stdECparam.pem
rafa@rafa-virtual-machine:~/Escritorio/spsi/p2$ cat stdECparam.pem
-----BEGIN EC PARAMETERS-----
BgUrgQQAHA==
-----END EC PARAMETERS-----
```

9. Generad cada uno de vosotros una clave para los parámetros anteriores. La clave se almacenará en nombreECkey.pem y no es necesario protegerla por contraseña.

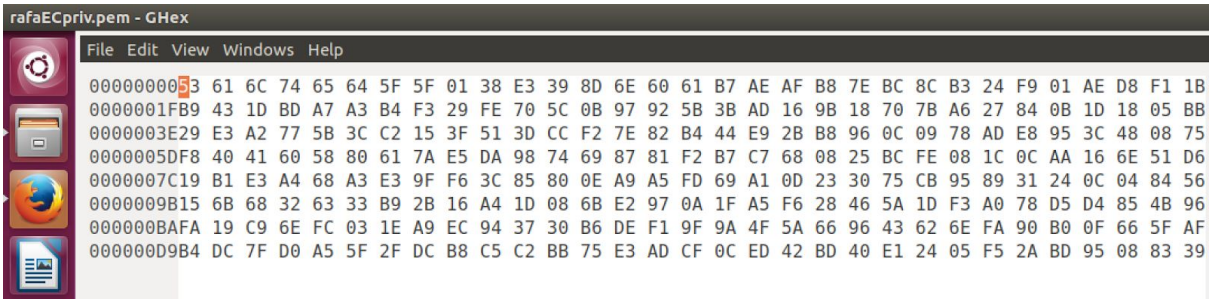
openssl ecparam -name secp128r1 -genkey -out rafaECkey.pem

```
rafa@rafa-virtual-machine:~/Escritorio/spsi/p2$ openssl ecparam -name secp128r1
-genkey -out rafaECkey.pem
rafa@rafa-virtual-machine:~/Escritorio/spsi/p2$ cat rafaECkey.pem
-----BEGIN EC PARAMETERS-----
BgUrgQQAHA==
-----END EC PARAMETERS-----
-----BEGIN EC PRIVATE KEY-----
MEQCAQEEEDz700kSvSUdI3CCUiDrCOGgBwYFK4EEABYhJAMiAASHxP8aj/O5JLER
L5yJmQpcXc45Eg6L98HIIs16JMReCw==
-----END EC PRIVATE KEY-----
```

10. Extraed" la clave privada contenida en el archivo nombreECkey.pem a otro archivo que tenga por nombre nombreECpriv.pem. Este archivo deber a estar protegido por contrasenña cifrándolo con 3DES. Mostrad sus valores.

openssl des3 -in rafaECkey.pem -out rafaECpriv.pem

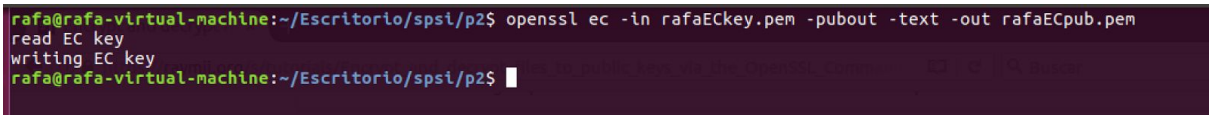
```
rafa@rafa-virtual-machine:~/Escritorio/spsi/p2$ openssl des3 -in rafaECkey.pem -
out rafaECpriv.pem
enter des-ede3-cbc encryption password:
Verifying - enter des-ede3-cbc encryption password:
```



11. Extraed en nombreECpub.pem la clave pública contenida en el archivo nombreECkey.pem. Como antes nombreECpub.pem no debe estar cifrado ni protegido. Mostrad sus valores.

openssl ec -in rafaECkey.pem -pubout -text -out rafaECpub.pem

Con esta orden obtenemos la clave pública.



Mostramos los valores de rafaECpub.pem

```
rafa@rafa-virtual-machine:~/Escritorio/spsi/p2$ cat rafaECpub.pem
Private-Key: (128 bit)
priv:
    24:a2:7a:dc:2b:f9:da:87:e7:9c:a6:3a:9a:1e:c3:
    74
pub:
    04:2d:69:f0:96:c3:b2:d0:47:b4:c9:b1:21:7f:3b:
    a6:a3:44:d3:68:6c:b8:9c:de:87:2d:28:e1:32:73:
    53:78:e9
ASN1 OID: secp128r1
-----BEGIN PUBLIC KEY-----
MDYwEAYHKoZIzj0CAQYFK4EEABwDIgAELWnwlsOy0Ee0ybEhfzumo0TTaGy4nN6H
LSjhMnNTeOk=
-----END PUBLIC KEY-----
```

BIBIOGRAFIA

- https://raymii.org/s/tutorials/Encrypt_and_decrypt_files_to_public_keys_via_the_OpenSSL_Command_Line.html

- https://wiki.openssl.org/index.php/Command_Line_Utilities