

**CURSO 2016-2017**  
GRADO EN INGENIERÍA INFORMÁTICA  
UNIVERSIDAD DE GRANADA

---

# SPSI

## Práctica 2

---

Carlos Manuel Sequí Sánchez

26 de noviembre de 2017

## Índice

1	EJERCICIO 1. Generad, cada uno de vosotros, una clave RSA (que contiene el par de claves) de 768 bits. Para referirnos a ella supondré que se llama nombreRSAkey.pem. Esta clave no es necesario que esté protegida por contraseña.	4
2	EJERCICIO 2."Extraed" la clave privada contenida en el archivo nombreRSAkey.pem a otro archivo que tenga por nombre nombreRSApriv.pem. Este archivo deberá estar protegido por contraseña cifrándolo con AES-128. Mostrad sus valores.	4
3	EJERCICIO 3.Extraed en nombreRSApub.pem la clave pública contenida en el archivo nombreRSAkey.pem. Evidentemente nombreRSApub.pem no debe estar cifrado ni protegido. Mostrad sus valores	5
4	EJERCICIO 4.Intentad cifrar input.bin con vuestras claves públicas. Explicad el resultado.	6
5	EJERCICIO 5.Diseñad un cifrado híbrido, con RSA como criptosistema asimétrico.	6
6	EJERCICIO 6.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.	9
7	EJERCICIO 7.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.	12
8	EJERCICIO 8. 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.	12
9	EJERCICIO 9."Extraed" la clave privada contenida en el archivo nombreECkey.pem a otro archivo que tenga por nombre nombreECpriv.pem. Este archivo deberá estar protegido por contraseña cifrándolo con 3DES. Mostrad sus valores.	13
10	EJERCICIO 10.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.	13

## Índice de figuras

2.1. CarlosRSAPriv . . . . .	4
3.1. CarlosRSAPub . . . . .	5
5.1. archivo sessionkey . . . . .	6
5.2. sessionkey encriptado . . . . .	7
5.3. input.bin encriptado . . . . .	8
6.1. mensaje original . . . . .	9
6.2. archivo sessionkey . . . . .	9
6.3. mensaje cifrado . . . . .	10
6.4. sessionkey encriptado . . . . .	10
6.5. sessionkey descriptado . . . . .	11
6.6. mensaje descriptado por el receptor . . . . .	11
7.1. Parámetros curva elíptica . . . . .	12
8.1. Clave generada para los parámetros . . . . .	12
9.1. Clave privada cifrada con 3DES . . . . .	13
10.1. Clave pública . . . . .	13

## Índice de tablas

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

Generamos la clave RSA de 768 bits sin protección por contraseña:  
`openssl genrsa -out CarlosRSAkey.pem 768`

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

Extraemos la clave privada en CarlosRSApriv.pem y la protegemos mediante AES-128 introduciéndole la contraseña 0123456789:

`openssl rsa -in CarlosRSAkey.pem -out CarlosRSApriv.pem -outform PEM -aes128`

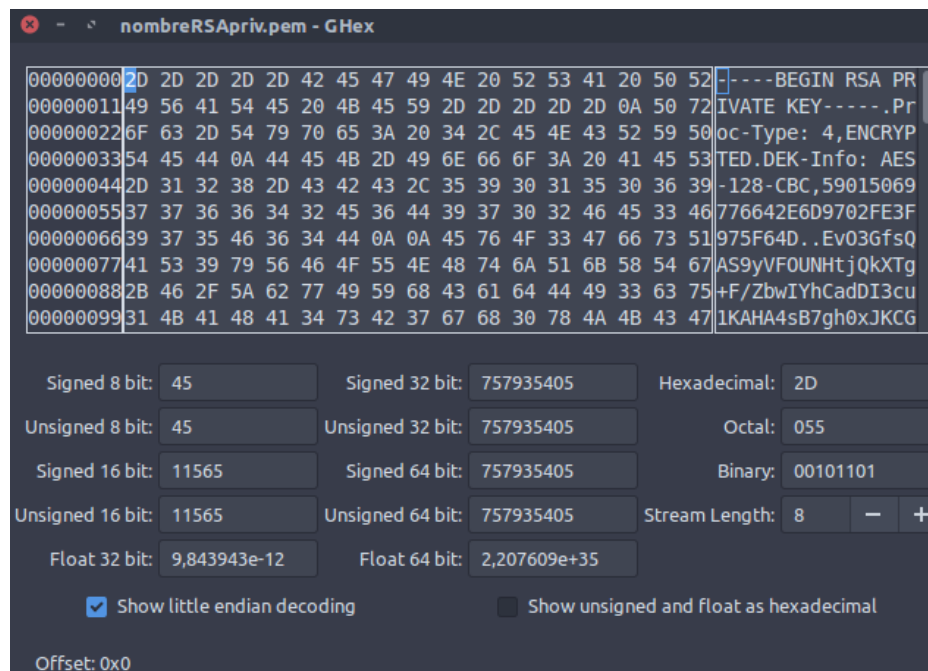


Figura 2.1: CarlosRSApriv

**3. EJERCICIO 3.Extraed en nombreRSApub.pem la clave pública contenida en el archivo nombreRSAkey.pem. Evidentemente nombreRSApub.pem no debe estar cifrado ni protegido. Mostrad sus valores**

Extraemos la clave publica de CarlosRSAkey.pem: `openssl rsa -in /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/CarlosRSAkey.pem -outform PEM -pubout -out /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/CarlosRSApub.pem`

```
1 |-----BEGIN PUBLIC KEY-----  
2 |MHwwDQYJKoZIhvcNAQEBBQADAwAwAJhALuCe2vRFkSTP/sbzPtbdDuxANNG/  
3 |KywK  
4 |awEHW5FW+8Sdxd3Z3qL4KM78uuxF+mII3QSlxrslnLPHcg0/  
5 |qzb37hRmw388W0dw  
6 |qoF08Zv4zz7yP8Rmtz3T/drFT0EBk+mmIQIDAQAB  
7 |-----END PUBLIC KEY-----
```

Figura 3.1: CarlosRSApub

#### 4. EJERCICIO 4.Intentad cifrar input.bin con vuestras claves públicas. Explicad el resultado.

Ciframos input.bin con clave pública

```
openssl rsautl -encrypt -pubin -inkey /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/CarlosRSAPub.pem -in /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/input.bin -out /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/inputEncrypted.bin
```

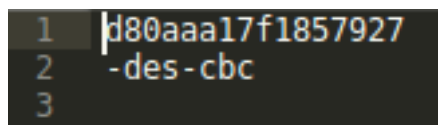
No está permitido cifrar input.bin ya que este tiene un tamaño de 1024 bits y la clave es de 768 bits, por tanto llegamos a la conclusión de que no es posible cifrar un fichero que tiene mayor tamaño que la clave que se quiere utilizar para cifrar. Esto es porque RSA es cifrado de flujo y no por bloques, por lo que necesita que la clave sea igual de grande que el mensaje.

#### 5. EJERCICIO 5.Diseñad un cifrado híbrido, con RSA como criptosistema asimétrico.

1. Elijo el sistema simétrico DES CBC.
2. Generamos sessionkey.txt
  - Para generar la cadena aleatoria utilizamos openssl rand, con un tamaño 8, ya que el tamaño de clave para DES es de 16 bytes:  

```
openssl rand -hex 8 >/home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/sessionkey.txt
```
  - Establecemos el sistema simétrico utilizado en la segunda línea del fichero  

```
echo des-cbc»>/home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/sessionkey.txt
```



```
1 p80aaa17f1857927
2 -des-cbc
3
```

Figura 5.1: archivo sessionkey

3. Ciframos sessionkey con la clave pública del receptor:  

```
openssl rsautl -encrypt -pubin -inkey /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/CarlosRSAPub.pem -in /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/sessionkey.txt -out sessionkeyEncrypted.txt
```

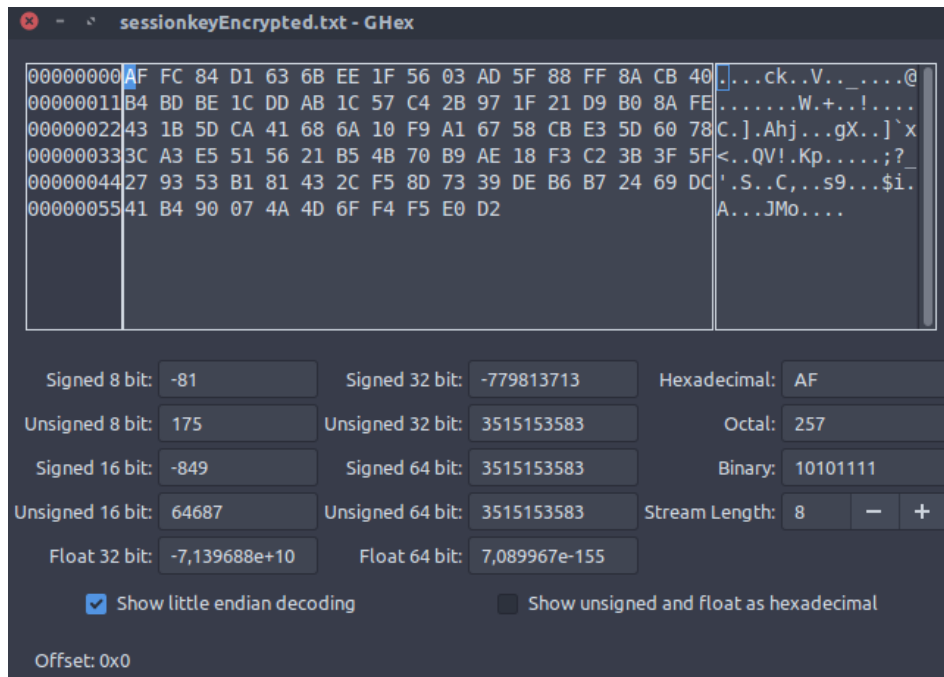


Figura 5.2: sessionkey encriptado

4. Ciframos el mensaje con DES CBC con clave semidébil:

**openssl enc -des-cbc -pass**

**file:/home/carlos/Escritorio/ETSIIT/SPSI/**

**Prácticas/P2/sessionkey.txt -iv 1231231231231231 -in /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/input.bin -out /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/inputEncriptado.bin**

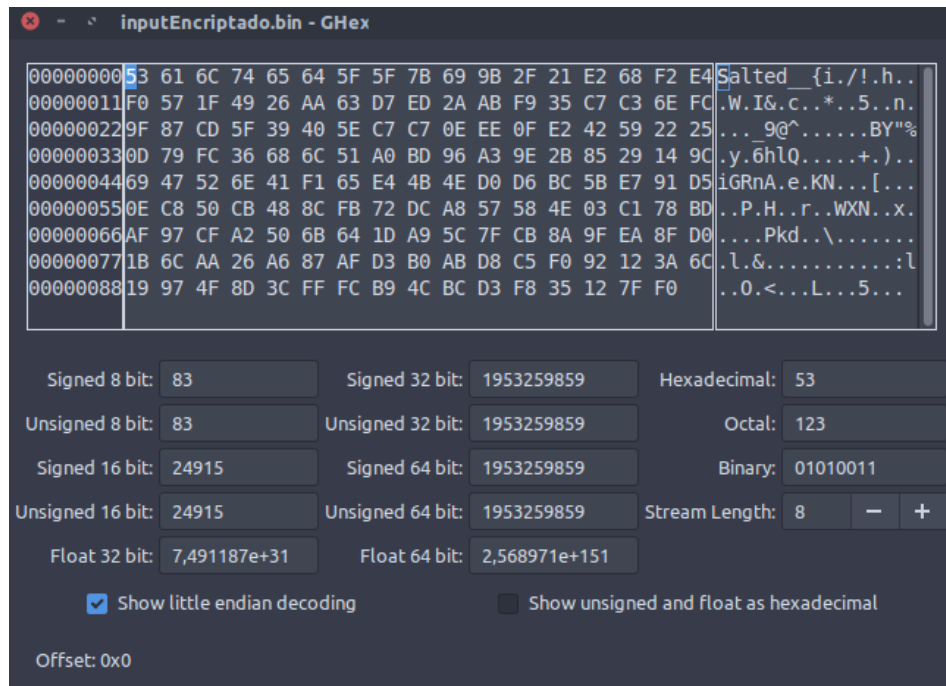
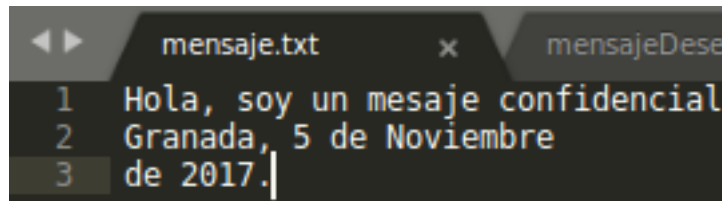


Figura 5.3: input.bin encriptado



**6. EJERCICIO 6. 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.**

Mensaje original que se desea enviar desde el emisor hasta el receptor.

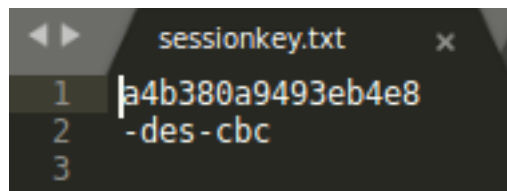


```
mensaje.txt x mensajeDeser
1 Hola, soy un mensaje confidencial.
2 Granada, 5 de Noviembre
3 de 2017.
```

Figura 6.1: mensaje original

1. Creamos el archivo sessionkey.

- -Para generar la cadena aleatoria utilizamos openssl rand, con un tamaño 8, ya que el tamaño de clave para DES es de 16 bytes:  
**openssl rand -hex 8 >/home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/ej7/sessionkey.txt**
- -Establecemos el sistema simétrico utilizado en la segunda línea del fichero **echo des-cbc» >/home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/ej7/sessionkey.txt**



```
sessionkey.txt x
1 a4b380a9493eb4e8
2 -des-cbc
3
```

Figura 6.2: archivo sessionkey

2. Ciframos el mensaje con el criptosistema diseñado (DES-CBC) y utilizando como clave el archivo sessionkey generado:

```
openssl enc -des-cbc -pass file:/home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/ej7/sessionkey.txt -iv 1231231231231231 -in /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/ej7/mensaje.txt -out /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/ej7/mensajeEncriptado.txt
```

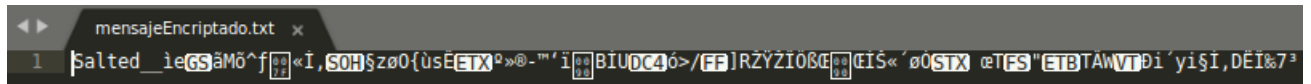


Figura 6.3: mensaje cifrado

3. Ciframos el sessionkey con la clave publica del receptor.

```
openssl rsautl -encrypt -pubin -inkey
/home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/
ej7/CarlosRSAPub.pem -in /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/
ej7/sessionkey.txt -out /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/
ej7/sessionkeyEncrypted.txt
```

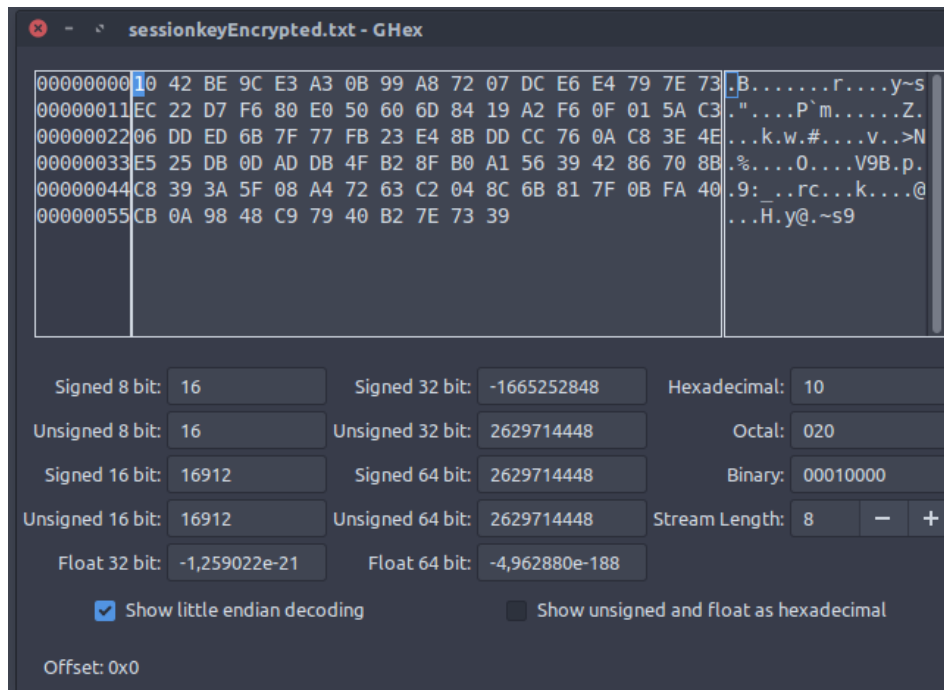
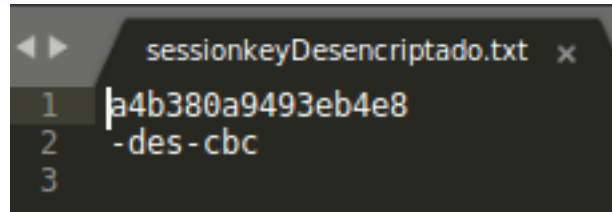


Figura 6.4: sessionkey encriptado

4. Enviamos al receptor esos dos archivos encriptados, tanto la sessionkey como el propio mensaje.
5. El receptor con su clave privada descrypta el sessionkey introduciendo la contraseña adecuada

```
openssl rsautl -decrypt -inkey
/home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/
ej7/CarlosRSAPriv.pem -in /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/
```

```
ej7/sessionkeyEncrypted.txt -out /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/  
P2/ej7/sessionkeyDesencriptado.txt
```

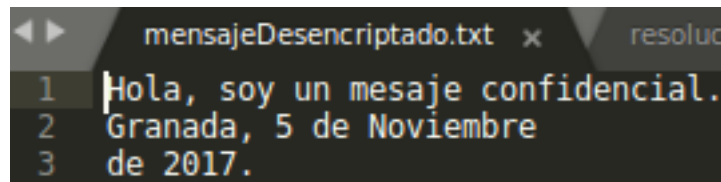


```
sessionkeyDesencriptado.txt x  
1 a4b380a9493eb4e8  
2 -des-cbc  
3
```

Figura 6.5: sessionkey desencriptado

6. Una vez ha desencriptado el sessionkey se genera un archivo sessionkeyDesencriptado que usaremos para desencriptar el mensaje original.

```
openssl des-cbc -d -pass file:/home/carlos/Escritorio/ETSIIT/SPSI/  
Prácticas/P2/ej7/sessionkeyDesencriptado.txt -iv 1231231231231231 -in  
/home/carlos/Escritorio/ETSIIT/SPSI/  
Prácticas/P2/ej7/mensajeEncriptado.txt -out /home/carlos/Escritorio/ETSIIT/SPSI/  
Prácticas/P2/ej7/mensajeDesencriptado.txt
```



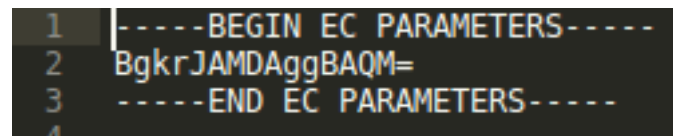
```
mensajeDesencriptado.txt x resoluc  
1 Hola, soy un mensaje confidencial.  
2 Granada, 5 de Noviembre  
3 de 2017.
```

Figura 6.6: mensaje desencriptado por el receptor

7. **EJERCICIO 7.** 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.

Generamos los parámetros de la curva:

```
openssl ecparam -name brainpoolP192r1 -out  
/home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/ej8/stdECparam.pem
```



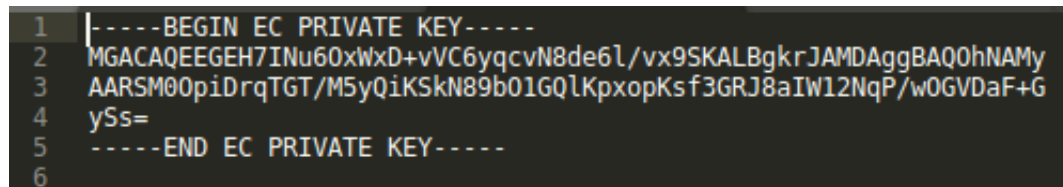
```
1 |-----BEGIN EC PARAMETERS-----  
2 |BgkrJAMDAggBAQM=  
3 |-----END EC PARAMETERS-----  
4
```

Figura 7.1: Parámetros curva elíptica

8. **EJERCICIO 8.** 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.

Generamos la clave para esos parámetros:

```
openssl ecparam -in /home/carlos/Escritorio/  
ETSIIT/SPSI/Prácticas/P2/ej8/stdECparam.pem -genkey -noout  
-out /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/  
ej8/CarlosECkey.pem
```



```
1 |-----BEGIN EC PRIVATE KEY-----  
2 |MGACAQEEGEH7INu60xWxD+vVC6yqcvN8de6l/vx9SKALBgkrJAMDAggBAQ0hNAMY  
3 |AARSM00piDrqTGT/M5yQiKSkN89b01GQlKpxopKsf3GRJ8aIW12NqP/w0GVDaF+G  
4 |ySs=  
5 |-----END EC PRIVATE KEY-----  
6
```

Figura 8.1: Clave generada para los parámetros

9. **EJERCICIO 9.** "Extraed" la clave privada contenida en el archivo nombreECkey.pem a otro archivo que tenga por nombre nombreECpriv.pem. Este archivo deberá estar protegido por contraseña cifrándolo con 3DES. Mostrad sus valores.

Sacamos la privada y la protegemos con la clave 0123456789:

```
openssl ec -in /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/ej8/CarlosECkey.pem -out /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/ej8/CarlosECpriv.pem -outform PEM -des3
```

```
1 -----BEGIN EC PRIVATE KEY-----
2 Proc-Type: 4, ENCRYPTED
3 DEK-Info: DES-EDE3-CBC,43A9B3CCD7DE2CA1
4
5 e6kRnbIxKcSwm5YL67t08E8BM845/3+LAZRWoJyYc0TltrylwASVmlxtlxc5V9hV
6 xYoKmbQh3ANDr4yyeXTP6nemGwL/BIHwixBrFL0qwYcbX59lSy78RTj1EylC1W3Z
7 lc0xqW1L8fc=
8 -----END EC PRIVATE KEY-----
9
```

Figura 9.1: Clave privada cifrada con 3DES

10. **EJERCICIO 10.** 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.

Sacamos la pública:

```
openssl ec -in /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/ej8/CarlosECkey.pem -pubout -out /home/carlos/Escritorio/ETSIIT/SPSI/Prácticas/P2/ej8/CarlosECpub.pem
```

```
1 -----BEGIN PUBLIC KEY-----
2 MEowFAYHKOZIzj0CAQYJKyQDAwIIAQEDAzIABFIzQ6mIOupMZP8znJCipKQ3z1s7
3 UZCUqnGikqx/cZENxohbXY2o//A4ZUNoX4bJKw==
4 -----END PUBLIC KEY-----
5
```

Figura 10.1: Clave pública