



---

# CRIPTOSISTEMAS ASIMÉTRICOS

---

Práctica 2



## Índice

1. Generación y manipulación de claves con RSA.....	3
a. Generación clave privada.....	3
Comando .....	3
Parámetros .....	3
Texto plano.....	4
b. Cifrando la clave privada.....	4
Comando .....	4
Parámetros .....	5
Texto plano.....	6
c. Generar la pública .....	6
Comando .....	6
Texto plano.....	6
d. Cifrando con la clave pública.....	6
Comando .....	6
Error.....	6
2. Cifrado híbrido (Simétrico y asimétrico) .....	7
a. Generando el Archivo sessionkey .....	7
Comando .....	7
Texto plano.....	7
b. Cifrando del archivo a enviar con la clave simétrica almacenada en el sessionkey .....	7
Comando .....	7
Texto plano.....	7
c. Cifrando el archivo sesiónkey con la clave pública del receptor.....	7
Comando .....	7
Texto plano.....	8
d. Descifrando el sessinkey con la clave privada del receptor .....	8
Comando .....	8
Texto Plano.....	8
e. Descifrando el archivo recibido con la clave simétrica almacenada en el sessionkey.....	8
Comando .....	8
Texto Plano.....	8
f. Conclusiones.....	8
3. Curvas elípticas.....	9
a. Encontrar una de las curvas del profesor y sacar los parámetros de esta.....	9
Comando .....	9

Texto plano.....	10
b.    Generar la clave.....	10
Comandos.....	10
Texto plano.....	10
c.    Extraer la clave privada cifrada .....	10
Comandos.....	10
Texto plano.....	11
d.    Extraer la pública.....	11
Comandos.....	11
Texto plano.....	11

## 1. Generación y manipulación de claves con RSA

### a. Generación clave privada

Comando

```
pedro@ubuntu:~/Desktop/spsi/p2$ openssl genrsa -out pedroLuisFuentesRSAkey.pem 901
Generating RSA private key, 901 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
```

Genrsa: Indica que se va a generar una clave

-out: Es el nombre de salida del archivo

901: Es el número de bits de la clave que se va a generar

Parámetros

```
pedro@ubuntu:~/Desktop/spsi/p2$ openssl rsa -in pedroLuisFuentesRSAkey.pem -text
Private-Key: (901 bit)
modulus:
    18:9a:32:2d:1f:e6:76:ed:c1:35:f9:c4:80:6d:eb:
    d9:ca:de:fd:4d:21:ea:16:99:3a:2f:07:78:2e:db:
    d8:29:8d:2f:80:ae:dd:54:ba:ad:ac:8e:a2:1e:19:
    db:af:ee:5a:30:21:68:4c:99:19:95:c3:2f:f4:d7:
    eb:b2:a0:3a:20:eb:39:24:c9:1e:38:b0:d7:46:35:
    d9:e7:b6:b4:49:ec:84:e3:db:9e:69:ad:2e:4c:04:
    2a:2d:f4:d2:ad:84:1c:5b:60:04:25:db:84:c0:1b:
    3b:7d:c2:5b:af:6f:0c:4d
publicExponent: 65537 (0x10001)
privateExponent:
    06:78:98:76:12:0b:f6:80:36:ef:d1:90:84:0f:65:
    97:d5:aa:ad:89:9a:40:0e:4d:a3:66:37:5a:bf:48:
    88:24:f5:c3:e6:df:17:cc:6f:85:ba:fb:91:5c:c5:
    84:69:54:12:58:d2:90:b2:85:1a:9b:ac:e8:6b:3b:
    98:c8:ec:b0:e2:7a:d1:47:2b:fb:3f:43:42:1e:03:
    69:66:42:d1:2a:0c:1f:85:56:f2:5d:24:ed:d8:45:
    8d:a0:e0:63:ae:a5:a7:c9:bc:83:26:76:bf:1c:f0:
    10:cb:e3:fb:d6:35:6a:a9
prime1:
    07:c9:17:c3:09:c0:a5:b8:c4:7f:fa:a6:10:34:d2:
    fe:a1:90:90:4f:5b:ea:43:9f:6d:e2:e5:a3:be:f1:
    a7:de:47:0e:06:23:47:6e:0c:9d:5f:2a:9b:1b:b5:
    d1:bc:38:2b:eb:dd:f1:6b:bd:64:7d:d7
prime2:
    03:28:f6:81:8f:b1:71:bb:33:e8:f9:bc:46:86:15:
    e8:d7:35:8c:b8:1d:11:26:21:9c:0b:43:a7:77:92:
    0c:cf:fc:98:32:06:5e:cb:34:06:37:6c:2b:19:48:
    5b:ad:72:97:65:c8:51:5f:3c:3f:5a:7b
exponent1:
    04:c6:18:4c:de:59:03:04:c5:70:57:c7:a5:9e:4e:
    56:15:c9:27:c3:91:6f:96:a1:56:66:cf:3a:55:65:
    a5:f2:14:8d:93:d3:e4:03:a5:b6:85:59:01:9c:bf:
    22:e8:8f:64:d7:0e:d2:ee:a6:45:cd:27
exponent2:
    02:b8:34:3e:18:be:6b:d4:e2:bd:f4:7f:d6:69:fb:
    5d:22:29:d1:eb:7a:08:92:3a:86:a3:23:1c:73:49:
    5c:81:7b:30:38:8d:46:f3:b6:d8:4a:c6:6a:e9:1d:
    5e:d9:0d:2f:c2:c6:fd:50:ea:f6:d3:15
coefficient:
    12:9b:1e:fa:aa:85:cb:05:a0:97:ba:6c:4f:1b:b2:
    69:06:56:bd:4b:9a:2c:7c:d5:3c:0b:2d:f2:76:b2:
    bc:8c:38:b0:a4:fe:c2:9d:1a:d6:ae:1d:f9:83:69:
    ff:10:2a:be:32:82:fa:dd:da:96:27
```

```
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICFAIBAAJxGJoyLR/mdu3BNfnEgG3r2cre/U0h6haZ0i8HeC7b2CmNL4Cu3VS6
ray0oh4Z26/uWjAhaEyZGZXDL/TX67Kg0iDrOSTJHjiw10Y12ee2tEnsh0Pbnmmt
LkwEKi300q2EHFtgBCXbhmAb033CW69vDE0CAwEAAQJxBniYdhIL9oA279GQhA9L
l9WqrYmaQA5No2Y3Wr9IiCT1w+bff8xvhbr7kVzFhGLUEljSkLKFGpus6Gs7mMjs
s0J60Ucr+z9DQh4DaWZC0SoMH4VW8l0k7dhFjaDgY66lp8m8gyZ2vxzwEMvj+9Y1
aqkCOQfJF8MJwKW4xH/6phA00v6hkJBPW+pDn23i5a0+8afeRw4GI0duDJ1fKpsb
tdG80Cvr3fFrvWR91wI5Ayy2gY+xcbsz6Pm8RoYV6Nc1jLgdESYhnAtDp3eSDM/8
mDIGXss0BjdsKxliW61yl2XIUV88P1p7AjkExhhM3lkDBMVwV8elnk5WFcknw5Fv
lqFWZs86VWwL8hSNk9PKA6W2hVkBnL8i6I9k1w7S7qZFzScCOQK4ND4YvmvU4r30
f9Zp+10iKdHregiS0oajIxxzSVyBezA4jUbztthKxmrvPHV7ZDS/Cxv1Q6vbTFQI4
Epse+qqFywWgl7psTxuyaQZWvUuaLHzVPast8nayvIw4sKT+wp0a1q4d+YNp/xAq
vjKC+t3alic=
-----END RSA PRIVATE KEY-----
```

Texto plano

```
1 -----BEGIN RSA PRIVATE KEY-----
2 MIICFAIBAAJxGJoyLR/mdu3BNfnEgG3r2cre/U0h6haZ0i8HeC7b2CmNL4Cu3VS6
3 ray0oh4Z26/uWjAhaEyZGZXDL/TX67Kg0iDrOSTJHjiw10Y12ee2tEnsh0Pbnmmt
4 LkwEKi300q2EHFtgBCXbhmAb033CW69vDE0CAwEAAQJxBniYdhIL9oA279GQhA9L
5 l9WqrYmaQA5No2Y3Wr9IiCT1w+bff8xvhbr7kVzFhGLUEljSkLKFGpus6Gs7mMjs
6 s0J60Ucr+z9DQh4DaWZC0SoMH4VW8l0k7dhFjaDgY66lp8m8gyZ2vxzwEMvj+9Y1
7 aqkCOQfJF8MJwKW4xH/6phA00v6hkJBPW+pDn23i5a0+8afeRw4GI0duDJ1fKpsb
8 tdG80Cvr3fFrvWR91wI5Ayy2gY+xcbsz6Pm8RoYV6Nc1jLgdESYhnAtDp3eSDM/8
9 mDIGXss0BjdsKxliW61yl2XIUV88P1p7AjkExhhM3lkDBMVwV8elnk5WFcknw5Fv
10 lqFWZs86VWwL8hSNk9PKA6W2hVkBnL8i6I9k1w7S7qZFzScCOQK4ND4YvmvU4r30
11 f9Zp+10iKdHregiS0oajIxxzSVyBezA4jUbztthKxmrvPHV7ZDS/Cxv1Q6vbTFQI4
12 Epse+qqFywWgl7psTxuyaQZWvUuaLHzVPast8nayvIw4sKT+wp0a1q4d+YNp/xAq
13 vjKC+t3alic=
14 -----END RSA PRIVATE KEY-----
15
```

## b. Cifrando la clave privada

Comando

```
pedro@ubuntu:~/Desktop/spsi/p2$ openssl rsa -in pedroLuisFuentesRSAkey.pem -aes128 -out pedroLuisFuentesRSApriv.pem
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

-in: El archivo que se va a cifrar

-aes128: El cifrado que se va a usar

-out: el nombre de salida del archivo ya cifrado

## Parámetros

```
pedro@ubuntu:~/Desktop/spst/p2$ openssl rsa -in pedroLuisFuertesRSAPriv.pem -text
Enter pass phrase for pedroLuisFuertesRSAPriv.pem:
Private-Key: (901 bit)
modulus:
    18:9a:32:2d:1f:e6:76:ed:c1:35:f9:c4:80:6d:eb:
    d9:ca:de:fd:4d:21:ea:16:99:3a:2f:07:78:2e:db:
    d8:29:8d:2f:80:ae:dd:54:ba:ad:ac:8e:a2:1e:19:
    db:af:ee:5a:30:21:68:4c:99:19:95:c3:2f:f4:d7:
    eb:b2:a0:3a:20:eb:39:24:c9:1e:38:b0:d7:46:35:
    d9:e7:b6:b4:49:ec:84:e3:db:9e:69:ad:2e:4c:04:
    2a:2d:f4:d2:ad:84:1c:5b:60:04:25:db:84:c0:1b:
    3b:7d:c2:5b:af:6f:0c:4d
publicExponent: 65537 (0x10001)
privateExponent:
    06:78:98:76:12:0b:f6:80:36:ef:d1:90:84:0f:65:
    97:d5:aa:ad:89:9a:40:0e:4d:a3:66:37:5a:bf:48:
    88:24:f5:c3:e6:df:17:cc:6f:85:ba:fb:91:5c:c5:
    84:69:54:12:58:d2:90:b2:85:1a:9b:ac:e8:6b:3b:
    98:c8:ec:b0:e2:7a:d1:47:2b:fb:3f:43:42:1e:03:
    69:66:42:d1:2a:0c:1f:85:56:f2:5d:24:ed:d8:45:
    8d:a0:e0:63:ae:a5:a7:c9:bc:83:26:76:bf:1c:f0:
    10:cb:e3:fb:d6:35:6a:a9
prime1:
    07:c9:17:c3:09:c0:a5:b8:c4:7f:fa:a6:10:34:d2:
    fe:a1:90:90:4f:5b:ea:43:9f:6d:e2:e5:a3:be:f1:
    a7:de:47:0e:06:23:47:6e:0c:9d:5f:2a:9b:1b:b5:
    d1:bc:38:2b:eb:dd:f1:6b:bd:64:7d:d7
prime2:
    03:28:f6:81:8f:b1:71:bb:33:e8:f9:bc:46:86:15:
    e8:d7:35:8c:b8:1d:11:26:21:9c:0b:43:a7:77:92:
    0c:cf:fc:98:32:06:5e:cb:34:06:37:6c:2b:19:48:
    5b:ad:72:97:65:c8:51:5f:3c:3f:5a:7b
exponent1:
    04:c6:18:4c:de:59:03:04:c5:70:57:c7:a5:9e:4e:
    56:15:c9:27:c3:91:6f:96:a1:56:66:cf:3a:55:65:
    a5:f2:14:8d:93:d3:e4:03:a5:b6:85:59:01:9c:bf:
    22:e8:8f:64:d7:0e:d2:ee:a6:45:cd:27
exponent2:
    02:b8:34:3e:18:be:6b:d4:e2:bd:f4:7f:d6:69:fb:
    5d:22:29:d1:eb:7a:08:92:3a:86:a3:23:1c:73:49:
    5c:81:7b:30:38:8d:46:f3:b6:d8:4a:c6:6a:e9:1d:
    5e:d9:0d:2f:c2:c6:fd:50:ea:f6:d3:15
coefficient:
    12:9b:1e:fa:aa:85:cb:05:a0:97:ba:6c:4f:1b:b2:
    69:06:56:bd:4b:9a:2c:7c:d5:3c:0b:2d:f2:76:b2:
    bc:8c:38:b0:a4:fe:c2:9d:1a:d6:ae:1d:f9:83:69:
    ff:10:2a:be:32:82:fa:dd:da:96:27

writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICFAIBAAJxGJoyLR/mdu3BNfnEgG3r2cre/U0h6haZ0i8HeC7b2CmNL4Cu3VS6
ray0oh4Z26/uWjAhaEyZGZXDL/TX67Kg0iDrOSTJHjiw10Y12ee2tEnsh0Pbnmmt
LkwEKi300q2EHFtgBCXbhMAb033CW69vDE0CAwEAAQJxBniYdhIL9oA279GQhA9l
l9WqrYmaQASNo2Y3Wr9IiCT1w+bFF8xvhbr7kVzFhGLUEljSkLKFGpus6Gs7mMjs
sOJ60Ucr+z9DQh4DaWZC0SoMH4VW8l0k7dhFjaDgY66lp8m8gyZ2vxzwEMvj+9Y1
aqkCOQfJF8MJwKW4xH/6pHA00v6hkJPW+pDn23i5a0+8afeRw4GI0duDJ1fKpsb
tdG80Cvr3fFrVWR91wI5Ayy2gY+xcbsz6Pm8RoYV6Nc1jLgdESYhnAtDp3eSDM/8
mDIGXss0BjdsKxliW61yl2XIUV88P1p7AjkExhHM3lkDBMVwV8elnk5WFcknw5Fv
lqFWZs86VWwL8hSNk9PkA6w2hVkBnL8i6I9k1w7S7qZFzScCOQK4ND4YvmvU4r30
f9Zp+10iKdHregiS0oajIxxzSVyBezA4jUbztthKxmrvHV7ZDS/Cxv1Q6vbTFQI4
Epse+qqFyWg17psTxuyaQZWUuaLHzVPast8nayvIw4sKT+wp0a1q4d+YNp/xAq
vjKC+t3alic=
-----END RSA PRIVATE KEY-----
```

### Texto plano

```

1  -----BEGIN RSA PRIVATE KEY-----
2  Proc-Type: 4,ENCRYPTED
3  DEK-Info: AES-128-CBC,0D5383E11E3AB1F84F2306822DBFE4BD
4
5  9dEUmzocMD2A46vdf0CIPXK3xLkq2zpYbQlZGDo2nz0X2iP8f7ahGas0AroT62yV
6  pkQgAxVfyuCptBCUM8Jk/lqovI+pP0FTSFgcefeh2XuPZYtR+PE1U25DlxalidF2
7  RyFdTekRgQshrhrvplre7aV8R1qLvKWoI3igPIaJgGj7aStIpYtb0ZmCagrqAMULC
8  LADiJlHLg8luAuaDlfsJ7EmyUMqUU3JGNnuDhquCZrvUPRPxKKEIaCkQ6KqfCeKI
9  MC4Gqd9LeK7w04dsZYsiFa6YwWxkox28HotxP0aqLzNkSkZ9Ki4KICSD1chT/GS
10 pItTAR8R9Jkg58GyuWSYVSzykqToJg3Ug9YKtgTfcGpCfDX/4dfFTU9Ipg5eG3qX
11 GPhf/rf3Pfa9y9dWAX/SolTEvhHaw6VPtPLvqJ+LQZKsIB7ZY8gZ6onL078LuoCW
12 MSLzN+E75p8Tin8bKMIBgFckr6KcK5WrHNjo0vaRZ0eB8eLviPt9CS2vKvAt0M0u
13 EglDlLJxgcte0ovNEWfYscP69l0zQ2HcKyXJbHSin9DglEkygkSo20oYhVyls
14 o6DTnBwxIN4rRTjnh1a/OfCztYNbQotuI9AzUimxUmVDhvnwnw6QIKiIc5oJ6Da
15 jUTzKDuIN0cxnHgDI2Alpz/X+g3EzAE1N1oPmTBKxnhBvocnSENAYKDKYtqVJqUE
16 KNK89MIGunKqM+h2Abp2Yg==
17  -----END RSA PRIVATE KEY-----

```

### c. Generar la pública

#### Comando

```

pedro@ubuntu:~/Desktop/spsi/p2$ openssl rsa -in pedroLuisFuentesRSAkey.pem -pubout -out pedroLuisFuentesRSApub.pem
writing RSA key

```

-pubout: Indica que se quiere extraer la clave pública del archivo de entrada

#### Texto plano

```

1  |-----BEGIN PUBLIC KEY-----
2  MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQK3v1NIeow
3  mTovB3gu29gpjS+Art1Uuq2sjqIeGduv7lowIWhMmRmVwy/0l+uyoDog6zkkYR44
4  sNdGNdntrRJ7ITj255prS5MBCot9NKthBxbYAYL24TAGzt9wluvbwxNagMBAAE=
5  -----END PUBLIC KEY-----
6

```

### d. Cifrando con la clave pública

#### Comando

```

pedro@ubuntu:~/Desktop/spsi/p2$ openssl rsautl -encrypt -in input.bin -pubin -inkey pedroLuisFuentesRSApub.pem -out output.bin
RSA operation error
13978986623552:error:0406D06E:rsa routines:RSA_padding_add_PKCS1_type_2:data too large for key size:
rsa_pk1.c:151:

```

rsautl: Indica que se va a llevar acciones con una clave.

-encrypt: Indica que se quiere cifrar

-in: El archivo que se quiere cifrar

-pubin: Indica que se va a hacer con clave pública

-inkey: Se le pasa dónde está la clave que se va a usar

-out: El nombre del fichero una vez cifrado.

#### Error

Como se puede ver el archivo que se intenta cifrar es demasiado largo.

Esto es debido a que la clave pública está pensada para cifrar archivos pequeños, como claves de sesión, o claves pequeñas de unos 128 o 256 bits, mientras que el archivo que se intenta cifrar es de 1024 bits.

Además, el cifrado asimétrico es realmente ineficiente, por lo que intentar cifrar y descifrar grandes cantidades de datos tomaría demasiado tiempo por lo que no es viable.

Lo que se hace es usar un cifrado simétrico, mucho más eficiente, y cifrar de manera asimétrica la contraseña del cifrado simétrico.



## 2. Cifrado híbrido (Simétrico y asimétrico)

a. Generando el Archivo sessionkey

## Comando

```

pedro@ubuntu:~/Desktop/spsi/p2$ openssl rand -hex 8 -out sessionkey
pedro@ubuntu:~/Desktop/spsi/p2$ ls -l
total 100
-rwxrwx-rw- 1 pedro pedro 78135 oct  8 00:54 asymmetric.pdf
-rwxrwxrwx 1 pedro pedro  128 sep 17 01:48 input.bin
-rw-rw-r-- 1 pedro pedro    0 oct  8 01:38 output.bin
-rw-rw-r-- 1 pedro pedro  790 oct  8 00:58 pedroLuisFuertesRSAkey.pem
-rw-rw-r-- 1 pedro pedro  881 oct  8 01:15 pedroLuisFuertesRSAPriv.pem
-rw-rw-r-- 1 pedro pedro  247 oct  8 01:19 pedroLuisFuertesRSAPub.pem
-rw-rw-r-- 1 pedro pedro   17 oct  8 02:14 sessionkey
-rw-rw-r-- 1 pedro pedro    0 oct  8 02:07 sessionkey.ssh

```

## Openssl rand: Para generar un archivo aleatorio

-hex: para que el formato sea hexadecimal

8: para decir que sea de 128 bits (8 caracteres x 2 bytes/carácter hexadecimal x 8bit/byte = 128 bits)

-out: indica el nombre del fichero de salida

## Texto plano

```
1 e2f8bc18027ca4a0
2 aes-128-ecb ← Modo de cifrado
```

\*El método de cifrado se ha añadido a mano

b. Cifrado del archivo a enviar con la clave simétrica almacenada en el sessionkey

## Comando

```
pedro@ubuntu:~/Desktop/spsi/p2$ openssl enc -aes-128-ecb -pass file:sessionkey -in input.bin -out out
put.bin
```

Openssl enc -aes-128-ecb: El tipo de cifrado a usar

-pass file:sessionkey: indica que se va a cifrar con contraseña, y que esta se encuentra en un archivo y el nombre del archivo donde se encuentra

## Texto plano

1 Salted \$t\$0S0e'UxDC2# Q-R0€ENO'UüxC2#Q-R0€ENO'UüxC2#Q-R0€ENO'UüxC2#  
Q-R0€ENO'UüxC2#Q-R0€ENO'UüxC2#Q-R0€ENO'UüxC2#Q-R0€ENO'UüxC2#  
Q-R0€ENO'U8ââ@ÄESQ2Q-d!,E%

Como se puede ver el output no tiene nada que ver con el input, además, se usa `salted` ya que se usa una contraseña.

c. Cifrando el archivo sesiónkey con la clave pública del receptor

## Comando

```
pedro@ubuntu:~/Desktop/spst/p2$ openssl rsautl -encrypt -in sessionkey -pubin -inkey pedroLuisFuentes
RSApub.pem -out sessionkey.ssh
```

Openssl rsautl: Para decir que se va a usar una de las opciones de RSA

-encrypt: Para decir que se va a encriptar

-in: el archivo de entrada que se quiere cifrar, sessinkey en este caso.

-pubin: indica que se va a cifrar con una clave pública



- inkey: archivo donde se encuentra la clave pública.
- out: nombre del fichero de salida

Texto plano

```
1  ETB+VTDC1rHa,âN&P}ñ-Æ•ŠòÖSYNNæS1Ly$QõI04SYNP[88]IÄ6ACK
   ÈSUBISTX$a{½fp"0¾U€+ÜYñeaw$-qCAN±0E0STXACK;~h%æM6½dnü#9$SUB"úÛè
2  EMÉi&1<#~R¥ÁfZ$DC2ôÄh
```

Como se puede ver, el sessionkey queda cifrado

#### d. Descifrando el sessinkey con la clave privada del receptor

Comando

```
pedro@ubuntu:~/Desktop/spst/p2$ openssl rsautl -decrypt -in sessionkey.ssh -inkey pedroLuisFuentesRSA
priv.pem -out sessionkey_recib
Enter pass phrase for pedroLuisFuentesRSApriv.pem:
```

- decrypt: Para indicar que se va a descifrar.
- inkey: es la clave privada del receptor
- in: seria el archivo sessionkey cifrado
- out: daría el archivo descifrado
- \*Notar que ya no se usa la opción – pubin, ya que se usa la clave privada.

Texto Plano

```
1  e2f8bc18027ca4a0
2  aes-128-ecb
```

Como se puede ver, se descifra correctamente

#### e. Descifrando el archivo recibido con la clave simétrica almacenada en el sessionkey

Comando

```
pedro@ubuntu:~/Desktop/spst/p2$ openssl enc -aes-128-ecb -pass file:sessionkey -d -in output.bin -out
input_recib.bin
```

- d: indica que se quiere descifrar
- in: seria el archivo cifrado
- out: daría el archivo descifrado, input\_ecib.bin

Texto Plano

```
1  0000 0000 0000 0000 0000 0000 0000 0000
2  0000 0000 0000 0000 0000 0000 0000 0000
3  0000 0000 0000 0000 0000 0000 0000 0000
4  0000 0000 0000 0000 0000 0000 0000 0000
5  0000 0000 0000 0000 0000 0000 0000 0000
6  0000 0000 0000 0000 0000 0000 0000 0000
7  0000 0000 0000 0000 0000 0000 0000 0000
8  0000 0000 0000 0000 0000 0000 0000 0000
```

#### f. Conclusiones

Como se puede ver, esto permite transmitir datos de manera segura por redes no seguras.

Además, evitamos la limitación que imponen los sistemas asimétricos para archivos de gran tamaño, ya que estos se cifran con sistemas simétricos, con una contraseña

de sesión, cifrando únicamente de manera asimétrica la contraseña y el tipo de cifrado que se ha usado.

Como se cifra con la pública del receptor, el único capaz de descifrar la contraseña de sesión será el propio receptor y por tanto el único capaz de descifrar todos los archivos.

### 3. Curvas elípticas

a. Encontrar una de las curvas del profesor y sacar los parámetros de esta

**Curva P-192**

- La curva se define en  $\mathbb{F}_p$  donde
 
$$p = 2^{192} - 2^{64} - 1$$

$$= 6277101735386680763835789423207666416083908700390324961279.$$
- Tiene por ecuación
 
$$y^2 = x^3 - 3x + \beta,$$
 donde
 
$$\beta = 0x\ 64210519\ e59c80e7\ 0fa7e9ab\ 72243049\ feb8deec\ c146b9b1.$$
- La curva tiene orden
 
$$6277101735386680763835789423176059013767194773182842284081.$$

La curva de las diapositivas P-192 cuyos parámetros son:

- $P = 6277101735386680763835789423207666416083908700390324961279$
- $B = 0x\ 64210519\ e59c80e7\ 0fa7e9ab\ 72243049\ feb8deec\ c146b9b1$

Se corresponde con la curva prime192v1:

```

prime192v1:
  F(p) with
  p = 6277101735386680763835789423207666416083908700390324961279,
  a = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF,
  b = 0x64210519E59C80E70FA7E9AB72243049FEB8DEECC146B9B1,
  G = 0x3188DA80EB03090F67CBF20EB43A18800F4FF0AFD82FF1012,
  n = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF99DEF836146BC9B1B4D22831,
  h = 1.
    
```

Comando

```
pedro@ubuntu:~/Desktop/spsi/p2$ openssl ecparam -name prime192v1 -C -out stdECparam.pem
```

OpenSSL ecparam: Indica que se van a usar parámetros de curvas elípticas

-name: especifica el nombre de la curva

-C: genera los parámetros de la curva

-out: fichero donde se van a almacenar los parámetros

### Texto plano

```

1 static unsigned char ec_p_192[] = {
2     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
3     0xFF,0xFF,0xFF,0xFE,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
4 };
5
6 static unsigned char ec_a_192[] = {
7     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
8     0xFF,0xFF,0xFF,0xFE,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFC
9 };
10
11 static unsigned char ec_b_192[] = {
12     0x64,0x21,0x05,0x19,0xE5,0x9C,0x80,0xE7,0x0F,0xA7,0xE9,0xAB,
13     0x72,0x24,0x30,0x49,0xFE,0xB8,0xDE,0xEC,0xC1,0x46,0xB9,0xB1
14 };
15
16 static unsigned char ec_gen_192[] = {
17     0x04,0x18,0x8D,0xA8,0x0E,0xB0,0x30,0x90,0xF6,0x7C,0xBF,0x20,
18     0xEB,0x43,0xA1,0x88,0x00,0xF4,0xFF,0x0A,0xFD,0x82,0xFF,0x10,
19     0x12,0x07,0x19,0x2B,0x95,0xFF,0xC8,0xDA,0x78,0x63,0x10,0x11,
20     0xED,0x6B,0x24,0xCD,0xD5,0x73,0xF9,0x77,0xA1,0x1E,0x79,0x48,
21     0x11
22 };
23
24 static unsigned char ec_order_192[] = {
25     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
26     0x99,0xDE,0xF8,0x36,0x14,0x6B,0xC9,0xB1,0xB4,0xD2,0x28,0x31
27 };
28
29 static unsigned char ec_cofactor_192[] = {
30     0x01
31 };
32

```

### b. Generar la clave

#### Comandos

```

pedro@ubuntu:~/Desktop/spsl/p2$ openssl ecparam -name prime192v1 -genkey -out pedroLuisFuertesEKey.pem

```

-genkey: genera la clave

### Texto plano

```

1 -----BEGIN EC PARAMETERS-----
2 BggqhkJOPQMBAQ==
3 -----END EC PARAMETERS-----
4 -----BEGIN EC PRIVATE KEY-----
5 MF8CAQEEGPiLTGfX36fXWhRvjQ0k4u1chQnlmQzpJKAKBggqhkJOPQMBAaE0AzIA
6 BNLPNz982v4BeIJKXhGaHKKHtPEgatst0YqJHvMc1ljQS6oaFmWy+zbVxIravxyC
7 eQ==
8 -----END EC PRIVATE KEY-----

```

### c. Extraer la clave privada cifrada

#### Comandos

```

pedro@ubuntu:~/Desktop/spsl/p2$ openssl ec -aes-128-ecb -in pedroLuisFuertesEKey.pem -out pedroLuisFuertesECpriv.pem
read EC key
writing EC key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

```

Openssl ec: indica que se va a trabajar con curvas elípticas

-aes-128-ecb: tipo de cifrado

- in: archivo donde está la clave
- out: el archivo con la clave privada cifrada

Texto plano

```
1 -----BEGIN EC PRIVATE KEY-----
2 Proc-Type: 4, ENCRYPTED
3 DEK-Info: AES-128-ECB,
4
5 QKaMF9M4YI9nZCcEOQea2DEv4GTHf4e/lTAvXwjUwky0kq8aMXMTfh0Ql10tD7uf
6 MTwBJ5mDX5GpQis3MCXfAP+xBwk8Ty20D0vipWfv2q+p4HlwZIx6b+U3/Z30/BZN
7 X0XMaAoh0+0p4a61NZ0plg==
8 -----END EC PRIVATE KEY-----
```

#### d. Extraer la pública

Comandos

```
pedro@ubuntu:~/Desktop/spsi/p2$ openssl ec -pubout -in pedroLuisFuentesEKey.pem -out pedroLuisFuentes
ECpub.pem
read EC key
writing EC key
```

Openssl ec: indica que se va a trabajar con curvas elípticas

- pubout: para indicar que quieres obtener la clave pública
- in: archivo donde está la clave
- out: el archivo con la clave pública

Texto plano

```
1 -----BEGIN PUBLIC KEY-----
2 MEkwEwYHKOZIzj0CAQYIKoZIzj0DAQEDMgAE0s83P3za/gF4gkpeEZocooe08SBq
3 2y3Rioke8xyXWNBLqhowZbL7NtXEitq/HIJ5
4 -----END PUBLIC KEY-----
```