

PRACTICA 2 SPSI

Criptosistemas Asimétricos

Antonio Manuel Rodríguez Martos

Generad, cada uno de vosotros, una clave RSA (que contiene el par de claves) de 901 bits. Para referirnos a ella supondré que se llama <nombre>RSAkey.pem. Esta clave no es necesario que esté protegida por contraseña.

Utilizaremos openssl genrsa para generar una clave para RSA, pasándole el nombre del archivo destino con -out . Indicaremos el número de bits del tamaño de la clave al final.

Openssl genrsa -out nombreRSAkey.pem 901

```
antonio@antonio-DELL:~/Escritorio$ openssl genrsa -out nombreRSAkey.pem 901
Generating RSA private key, 901 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
antonio@antonio-DELL:~/Escritorio$ cat nombreRSAkey.pem
-----BEGIN RSA PRIVATE KEY-----
MIICFQIBAAJxFiHbzMTFRs6jinWatoijJt60cFL/EizFv4wdN1eJoLU9/RDe4Jze
7bpeAgnksAobSHMLWTD3HxeDVNBAMLW2qAxaUediFsebh6gSRbWtN/XKfp1W3Xt
gWKMCPeWU6QPbgBrR2Nwhan1FkuKnPucf7MCAwEAAQJxBB4IZe+1Q5bt1bSGXW7N
60078rbh/N+XaisGxGTklaEiBxI8ZEzKrc8fjgGJKba3nKLPL+iMEJytTsCIjGxt
iJvPS3E9Zo9xDcavMeXvInieL3Hz0mrX8Fv/LTee9Si/R2EM0anETSZ5twM2CAI6
IQEQOqb40+x9fhz/BUD0FxXL1B6ATgnWasNa1yUnP7AmpUCIGIvx36osi+qU7NOL
6PxcVIQb09JGC7JNHwI5AyyPMskrohcGmSQztEFMDB2xxvtNLeuK5ka1AL9FNCEv
eC0LTh2GLLglUqLT3H/I3eJD2wssV6btAjkEDpCKoS9cQgzVkfNI7JVetjwyv0Da
siZU6TTHFIW+OjNGY5VJmahE9Smlo1kMV9GaGZrCrZ2qv/MCOQF44vazu1KvDxZS
3c/JQLWVx+8aeQ7E4GQsfuhbzIKCqXs4mLefpdyvELaix9LJSKTu3L1njycPeQI5
BK8602MX45a+/eIr9yISXXBeABspRdQDY5kIZTixRJWDC/XXYqSsUFXMTPhvB3/S
Q1ex1dohwiXV
-----END RSA PRIVATE KEY-----
antonio@antonio-DELL:~/Escritorio$
```

nombreRSAkey.pem
Clave privada RSA



1. Generacion de clave 901bits

“Extraed” la clave privada contenida en el archivo <nombre>RSAkey.pem a otro archivo que tenga por nombre <nombre>RSAPriv.pem. Este archivo deberá estar protegido por contraseña cifrándolo con AES-128. Mostrad sus valores.

Para el tratamiento de las claves utilizaremos openssl rsa, pasándole un archivo de entrada y otro de salida, cifrando la clave con aes128 en este caso. Por defecto se extrae la clave privada.

Openssl rsa -in nombreRSAkey.pem -out nombreRSAPriv.pem -aes128

```
antonio@antonio-DELL:~/Escritorio$ openssl rsa -in nombreRSAkey.pem -out nombreRSAPriv.pem -aes128
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

2. Comando para extraer clave privada cifrada rsa

Para mostrar los valores utilizaremos la opción -text de rsa la cual nos imprimirá los diferentes componentes de la clave pública o privada codificada y en texto plano.

openssl rsa -in nombreRSApriv.pem -text

```
antonio@antonio-DELL:~/Escritorio$ openssl rsa -in nombreRSApriv.pem -text
Enter pass phrase for nombreRSApriv.pem:
Private-Key: (901 bit)
modulus:
 16:21:db:cc:c4:c5:46:ce:a3:8a:75:80:b6:88:a3:
 26:de:8e:71:f2:ff:12:2c:c5:bf:8c:1d:37:57:89:
 a0:b5:3d:fd:10:de:e0:9c:de:ed:ba:5e:02:09:e4:
 b0:0a:1b:48:73:25:59:30:f7:1f:17:83:54:d0:40:
 30:b5:b6:a8:0c:5a:95:47:9d:88:5b:1e:6e:1e:a0:
 49:16:d6:b4:df:d7:29:fa:75:5b:75:ed:81:62:8c:
 0a:91:30:53:a4:0f:6e:00:6b:47:63:70:85:a9:f5:
 16:4b:8a:9c:fb:9c:7f:b3
publicExponent: 65537 (0x10001)
privateExponent:
 04:1e:08:65:ef:b5:43:96:ed:95:b4:86:5d:6e:cd:
 eb:4d:3b:f2:b6:e1:fc:df:97:6a:2b:06:c4:64:e4:
 95:a1:22:07:12:3c:64:4c:ca:ac:2f:1f:8e:01:89:
 29:b6:b7:9c:a2:cf:2f:e8:8c:10:9c:ad:4e:c0:88:
 8c:65:ed:88:9b:cf:4b:71:3d:66:8f:71:0c:26:af:
 31:e5:d5:88:d8:9e:2f:71:f3:3a:6a:d7:f0:5b:ff:
 2d:37:9e:f5:28:bf:47:61:0c:39:a9:c4:4d:26:79:
 b7:03:36:08:02:3a:21:01
prime1:
 06:f8:d3:ec:7d:7e:1c:ff:05:40:f4:17:15:cb:d4:
 1e:80:4e:09:d6:6a:c3:5a:d7:25:27:3f:b0:26:a5:
 40:88:18:8b:f1:df:aa:2c:8b:ea:94:ec:d3:8b:e8:
 fc:5c:54:84:1b:3b:d2:46:0b:b2:4d:1f
prime2:
 03:2c:a9:31:29:2b:a2:17:06:99:24:33:b4:41:4c:
 0c:1d:b1:c6:fb:4d:2d:eb:8a:e6:40:35:00:bf:45:
 34:21:2f:78:2d:0b:4e:1d:86:94:b8:25:52:a2:d3:
 dc:7f:c8:dd:e2:43:db:0b:2c:57:a6:ed
exponent1:
 04:0e:90:8a:a1:2f:5c:42:0c:d5:91:f3:48:ec:95:
 5e:b6:3c:32:bf:40:da:b2:26:54:e9:34:c7:14:85:
 be:3a:33:46:63:95:49:99:a8:44:f5:29:a5:a3:59:
 0c:57:d1:9a:19:9a:c2:ad:9d:aa:bf:f3
exponent2:
 01:78:e2:f6:b3:bb:52:af:0f:16:52:dd:cf:c9:40:
 b5:95:c7:ef:1a:79:0e:c4:e0:64:2c:7e:e8:5b:cc:
 82:82:a9:7b:38:98:b7:9f:a5:dc:af:12:50:22:c7:
 d2:c9:48:a4:ee:dc:bd:67:8f:27:0f:79
coefficient:
 04:af:3a:d3:63:17:e3:96:be:fd:e2:2b:f7:22:12:
 5d:70:5e:00:1b:29:45:d4:03:63:99:08:65:38:b1:
 44:90:d6:0b:f5:d7:62:a4:ac:50:55:cc:4c:f1:ef:
 07:7f:d2:43:57:b1:d5:da:21:c2:25:d5
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICFQIBAAJxFiHbzMTFRs6jinWAtOijJt60cfl/EizFv4wdN1eJoLU9/RDe4Jze
7bpeAgnksAobSHMLWTD3HxeDVNBAMLW2qAxalUediFsebh6gSRbWtN/XKfp1W3Xt
gWKMCPeWU6QPbgBrR2Nwhan1FkuKnPucf7MCAwEAAQJxBB4IZE+1Q5bt1bSGXW7N
60078rbh/N+XaisGxGTklaEiBxI8ZEzKrc8fjgGJKba3nKLPL+iMEJytTsCIjGXt
iJvPS3E9Zo9xDCavMeXViNieL3Hz0mrX8Fv/LTee9Si/R2EMOanETSZ5twM2CAI6
IQECOqb40+x9fhz/BUD0FxxL1B6ATgnWasNa1yUnP7AmpUCIGIvx36osi+qU7NOL
6PxcVIQb09JGC7JNHwI5AypmSkrohcGmSQztEFMD82xxvtNLeuK5ka1AL9FNCEv
eC0LTh2GLlglUqLT3H/I3eJD2wssV6btAjkEDpCKoS9cQgzVkfNI7JVetjwyv0Da
siZU6TTHFIW+OjNGY5VJmahE9Sml01kMV9GaGZrCrZ2qv/MCOQF44vazu1KvDxZS
3c/JQLWVx+8aeQ7E4GQsFuhbzIKCqXs4mLefpdyvELaix9LJSKTu3L1njycPeQIS
BK8602MX45a+/eIr9yISXXBeABspRdQDY5kIZTiXRJDWC/XXYqSsUFXTMPHvB3/S
Q1ex1dohwiXV
-----END RSA PRIVATE KEY-----
```

3. Archivo con la clave privada rsa

Extraed en <nombre>RSAPub.pem la clave publica contenida en el archivo <nombre>RSAkey.pem. Evidentemente <nombre>RSAkey.pem. no debe estar cifrado ni protegido. Mostrad sus valores.

Utilizaremos prácticamente lo de la pregunta anterior salvo que para extraer la clave publica debemos utilizar la opción -pubout y como no hace falta que esté cifrada omitimos la opción de aes128.

Openssl rsa -in nombreRSAkey.pem -out nombreRSAPub.pem -pubout

```
antonio@antonio-DELL:~/Escritorio$ openssl rsa -in nombreRSAkey.pem -out nombreRSAPub.pem -pubout
writing RSA key
```

4. Comando para extraer clave publica rsa

Para leer la clave publica utilizaremos la opción de rsa -pubin ya que por defecto se leería del archivo de entrada la clave privada.

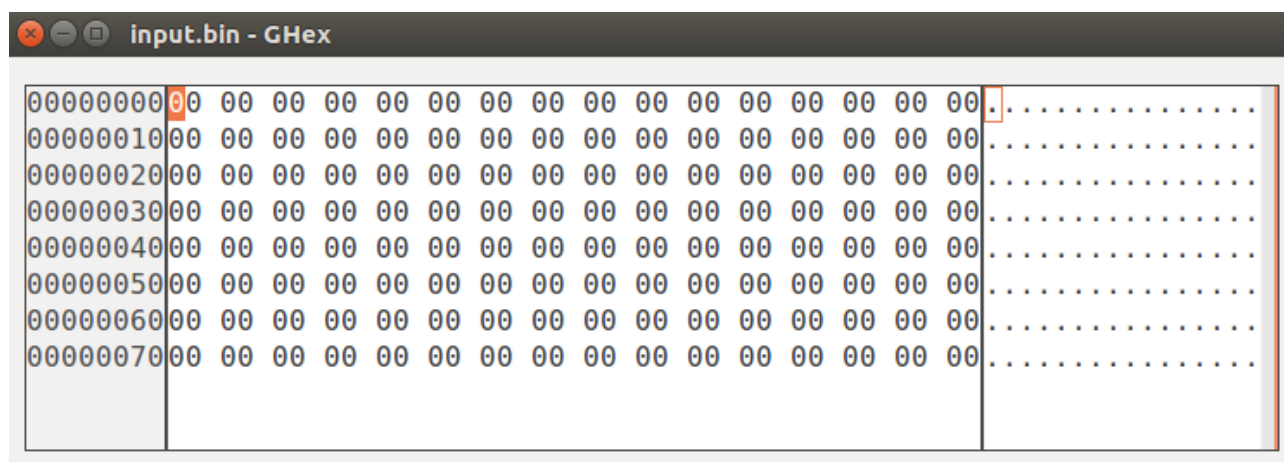
openssl rsa -in nombreRSAPub.pem -text -pubin

```
antonio@antonio-DELL:~/Escritorio$ openssl rsa -in nombreRSAPub.pem -text -pubin
Public-Key: (901 bit)
Modulus:
 16:21:db:cc:c4:c5:46:ce:a3:8a:75:80:b6:88:a3:
 26:de:8e:71:f2:ff:12:2c:c5:bf:8c:1d:37:57:89:
 a0:b5:3d:fd:10:de:e0:9c:de:ed:ba:5e:02:09:e4:
 b0:0a:1b:48:73:25:59:30:f7:1f:17:83:54:d0:40:
 30:b5:b6:a8:0c:5a:95:47:9d:88:5b:1e:6e:1e:a0:
 49:16:d6:b4:df:d7:29:fa:75:5b:75:ed:81:62:8c:
 0a:91:30:53:a4:0f:6e:00:6b:47:63:70:85:a9:f5:
 16:4b:8a:9c:fb:9c:7f:b3
Exponent: 65537 (0x10001)
writing RSA key
-----BEGIN PUBLIC KEY-----
MIGMMA0GCsQGSIB3DQEBAQUAA3sAMHgCcRYh28zExUb0o4p1gLaIoybejnHy/xIs
xb+MHTdXiaC1Pf0Q3uCc3u26XgIJ5LAKG0hzJVkw9x8Xg1TQQDC1tqgMWpVHnYhb
Hm4eoEkW1rTf1yn6dVt17YFijAqRMFOkD24Aa0djCIWp9RZLipz7nH+zAgMBAAE=
-----END PUBLIC KEY-----
```

5. Archivo con la clave publica rsa

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

```
-rw-rw-r-- 1 antonio antonio 128 sep 17 10:22 input.bin
```



6. Archivo input.bin

Intentad cifrar input.bin con vuestras claves pública. Explicad el mensaje de error obtenido.

Utilizaremos el comando de openssl rsautl el cual se puede utilizar para cifrar y descifrar datos usando el algoritmo RSA. Indicándole en este caso que queremos encriptar con - encrypt

Con la opcion -inkey le indicaremos el archivo de donde cogeremos la clave para cifrar y leemos como antes la clave publica con la opción -pubin.

Openssl rsautl -encrypt in input.bin -out output.bin -inkey nombreRSAPub.pem -pubin

```
antonio@antonio-DELL:~/Escritorio$ openssl rsautl -encrypt -in input.bin -out output.bin -inkey
nombreRSAPub.pem -pubin
RSA operation error
140373707220632:error:0406D06E:rsa routines:RSA_padding_add_PKCS1_type_2:data too large for key
size:rsa_pk1.c:153:
```

7. Comando para cifrar input con clave publica

El error nos dice que el tamaño del archivo es muy grande para el tamaño de la clave, ya que tenemos 1024b para el input y 901b para la clave.

Por defecto la clave deberá ser como mínimo 11 bytes mayor que el archivo.

Como nuestra clave es de aproximadamente 112 bytes, solo podríamos encriptar mensajes como mínimo de 123 bytes y en nuestro mensaje son 128 bytes.

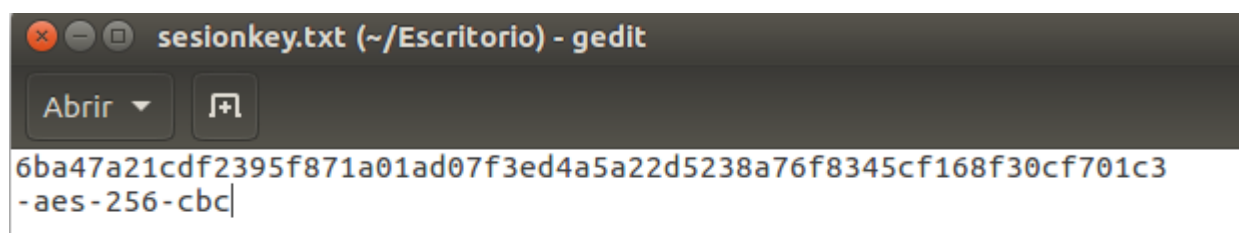
Diseñad un cifrado híbrido, con RSA como criptosistema asimétrico. El modo de proceder sera el siguiente:

1. El emisor debe seleccionar un sistema simétrico con su correspondiente modo de operación.
2. El emisor generara un archivo de texto, llamado por ejemplo sessionkey con dos líneas. La primera línea contendrá una cadena aleatoria hexadecimal cuya longitud sea la requerida para la clave del criptosistema simétrico. 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 Blowsh en modo ECB, la segunda línea deberá contener `-bf-ecb`.
3. El archivo sessionkey se cifrará con la clave pública del receptor.
4. El mensaje se cifrara utilizando el criptosistema simétrico, la clave se generara a partir del archivo anterior mediante la opción `-pass file:sesionkey`.

Utilizaremos el sistema simétrico AES 256 CBC por lo que la clave será de 32 bytes.

```
antonio@antonio-DELL:~/Escritorio$ openssl rand -hex 32 -out sesionkey.txt
```

8. Creacion de contraseña de 32B



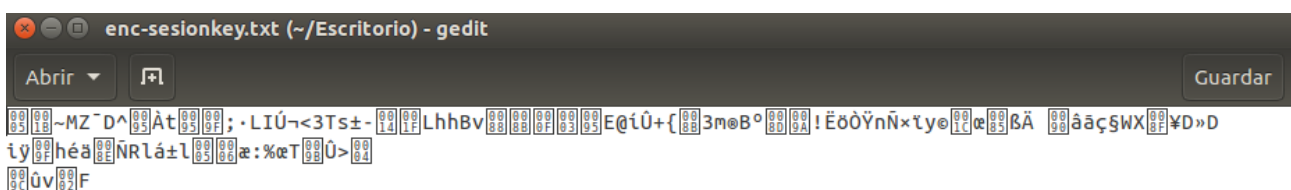
9. Archivo sesionkey.txt

Ciframos el archivo sesionkey.txt en el archivo enc-sesionkey.txt con la clave pública del receptor con los mismos comandos utilizados en el ejercicio anterior.

`openssl rsautl -encrypt -in sesionkey.txt -out enc-sesionkey.txt -inkey nombreRSAPub.pem -pubin`

```
antonio@antonio-DELL:~/Escritorio$ openssl rsautl -encrypt -in sesionkey.txt -out enc-sesionkey.txt -inkey nombreRSAPub.pem -pubin
```

10. Comando encriptar sesionkey.txt



11. Archivo enc-sesionkey.txt

Ciframos el archivo input.bin con aes 256 en modo cbc en el archivo input-cbc-256.bin con la contraseña contenida en sesionkey.txt

openssl en -aes-256-cbc -in input.bin -out input-cbc-256.bin -pass file:sesionkey.txt.

```
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-256-cbc -in input.bin -out input-cbc-256.bin -pass file:sesionkey.txt
```

12. Comando para cifrar input con sesionkey

input-cbc-256.bin - GHex

| | | | | | | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------------------|
| 00000000 | 53 | 61 | 6C | 74 | 65 | 64 | 5F | 5F | 0F | DC | 93 | 23 | D3 | D3 | E0 | CB | Salted__...#.... |
| 00000010 | 16 | 73 | 80 | AC | 4E | 39 | 7E | A8 | 2A | 43 | AA | 6F | 87 | D0 | A3 | 64 | .s..N9~.*C.o...d |
| 00000020 | F4 | 3A | 93 | F7 | 26 | B2 | 64 | 7D | 10 | 4E | CD | 01 | C6 | DA | F6 | 88 | ...&.d}.N..... |
| 00000030 | 26 | 81 | 19 | DD | 8D | 3D | 6D | 73 | 2F | 23 | 00 | 0E | FF | 4C | FF | FA | &....=ms/#...L.. |
| 00000040 | 7A | 4A | 2E | 5D | 37 | BA | 6C | 84 | CD | FE | 75 | 1E | 89 | 31 | 8F | 0C | zJ.]7.l...u..1.. |
| 00000050 | 66 | 93 | 81 | A1 | E3 | A5 | 96 | 59 | 0D | E2 | B8 | D6 | 70 | 53 | E6 | 8C | f.....Y....pS.. |
| 00000060 | 29 | 3A | 28 | 1D | 0C | DE | 8A | 33 | 91 | 5B | 33 | 7B | 75 | E0 | 97 | 3B |):(...3.[3{u..; |
| 00000070 | 2A | E6 | 80 | AC | A8 | 06 | B5 | 3C | 87 | 91 | 48 | 85 | CF | C1 | 41 | 55 | *.....<..H...AU |
| 00000080 | E1 | 25 | 56 | 68 | 8A | 96 | 9B | 07 | EF | DB | 32 | 1B | 30 | 0F | 1F | 40 | ..%Vh.....2.0..@ |
| 00000090 | 25 | 39 | EB | B3 | 32 | 5C | D9 | 91 | ED | 0A | 94 | A2 | 22 | EE | D1 | 4B | %9..2\....."..K |

Signed 8 bit: 83 Signed 32 bit: 1953259859 Hexadecimal: 53

Unsigned 8 bit: 83 Unsigned 32 bit: 1953259859 Octal: 123

Signed 16 bit: 24915 Signed 64 bit: 1953259859 Binary: 01010011

Unsigned 16 bit: 24915 Unsigned 64 bit: 1953259859 Stream Length: 8 - +

Float 32 bit: 7,491187e+31 Float 64 bit: 2,568971e+151

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x0

13. Archivo input encriptado

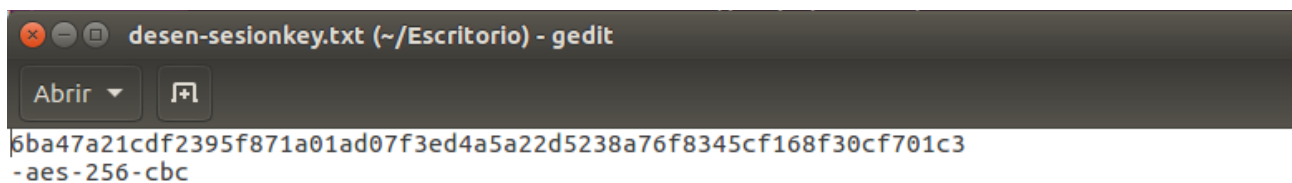
Utilizando el criptosistema hibrido diseñado, cada uno debe cifrar el archivo input.bin con su clave publica para, a continuación, descifrarlo con la clave privada. comparad el resultado con el archivo original.

Primero utilizaremos la clave privada contenida en nombreRSApriv.pem para descifrar el contenido de enc-sessionkey.txt

Openssl rsautl -decrypt -in enc-sesionkey.txt -out desen-sesionkey.txt -inkey nombreRSApriv.pem

```
antonio@antonio-DELL:~/Escritorio$ openssl rsautl -decrypt -in enc-sesionkey.txt -out desen-sesionkey.txt -inkey nombreRSApriv.pem
```

14. Comando para descifrar el enc-sesionkey



The screenshot shows a terminal window titled "desen-sesionkey.txt (~/Escritorio) - gedit". The command entered is: `openssl rsautl -decrypt -in enc-sesionkey.txt -out desen-sesionkey.txt -inkey nombreRSApriv.pem`. The output is a long hexadecimal string: `6ba47a21cdf2395f871a01ad07f3ed4a5a22d5238a76f8345cf168f30cf701c3` followed by `-aes-256-cbc`.

15. Archivo descifrado de enc-sesionkey

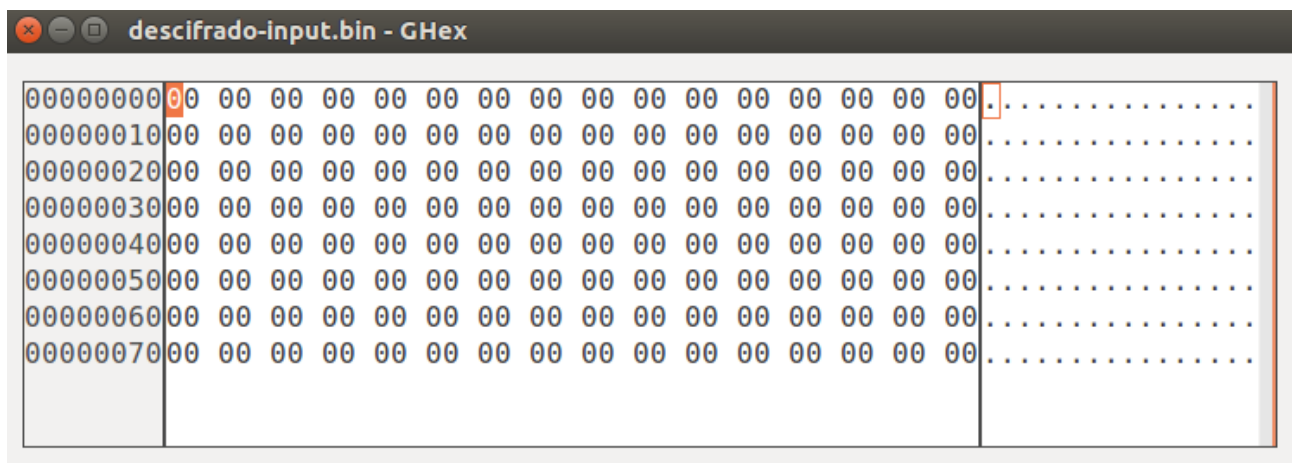
Desciframos el contenido del input que encriptamos en el ejercicio anterior pasándole como clave el archivo que desciframos en el apartado anterior.

openssl enc -aes-256-cbc -d -in input-cbc-256.bin -out descifrado-input.bin -pass file:desen-sesionkey.txt

```
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-256-cbc -d -in input-cbc-256.bin -out descifrado-input.bin -pass file:desen-sesionkey.txt
```

16. Comando para descifrar el input encriptado

Vemos que el resultado obtenido es correcto ya que es igual al archivo original input.bin



The screenshot shows a hex editor window titled "descifrado-input.bin - GHex". The main area displays a grid of hexadecimal values. The first column shows addresses from 00000000 to 00000070. The second column shows the hex values, which are mostly 00. The third column shows the corresponding ASCII values, which are mostly dots. This indicates that the decrypted file is a binary file with many zero bytes.

17. Archivo descifrado input

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 practica con una curva cualquiera a vuestra elección de las disponibles en OpenSSL. Mostrad los valores.

Para manipular o generar curvas elípticas utilizaremos el comando de openssl ecparam.

Con la opción -list_curves mostraremos todas las curvas elípticas que podemos usar.

```
antonio@antonio-DELL:~/Escritorio$ openssl ecparam -list_curves
secp112r1 : SECG/WTLS curve over a 112 bit prime field
secp112r2 : SECG curve over a 112 bit prime field
secp128r1 : SECG curve over a 128 bit prime field
secp128r2 : SECG curve over a 128 bit prime field
secp160k1 : SECG curve over a 160 bit prime field
secp160r1 : SECG curve over a 160 bit prime field
secp160r2 : SECG/WTLS curve over a 160 bit prime field
secp192k1 : SECG curve over a 192 bit prime field
secp224k1 : SECG curve over a 224 bit prime field
secp224r1 : NIST/SECG curve over a 224 bit prime field
secp256k1 : SECG curve over a 256 bit prime field
secp384r1 : NIST/SECG curve over a 384 bit prime field
secp521r1 : NIST/SECG curve over a 521 bit prime field
prime192v1: NIST/X9.62/SECG curve over a 192 bit prime field
prime192v2: X9.62 curve over a 192 bit prime field
```

18. Lista de curvas elípticas

Probamos los nombres y comprobamos que los valores concuerden con los de los de las curvas de las transparencias del tema.

Curva P-192

Tiene por ecuación $y^2 = x^3 - 3x + B$

Donde $B = 0x\ 64210519\ e59c80e7\ 0fa7e9ab\ 72243049\ feb8deec\ c146b9b1$.

Ejecutamos la posible opción, en este caso la curva perteneciente a prime192v1.

Con -name pondremos alguno de los nombres contenidos en la lista anterior y con la opción -param-enc especificaremos como se codificarán los datos, nosotros escogeremos explicit para darlos explícitamente y tener más información a la hora de mostrar resultados.

```
openssl ecparam -name prime192v1 -param_enc explicit -out stdECparam.pem
```

19. Comando para crear curva eliptica p-192

Mostramos los resultados.

Esta vez con la opción -text nos mostrará los parámetros de forma legible.

```
antonio@antonio-DELL:~/Escritorio$ openssl ecparam -in stdECparam.pem -text
```

20. Comando para mostrar curva eliptica

Y como vemos coincide con B.

```
Field Type: prime-field
Prime:
 00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
 ff:fe:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
A:
 00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
 ff:fe:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:fc
B:
 64:21:05:19:e5:9c:80:e7:0f:a7:e9:ab:72:24:30:
 49:fe:b8:de:ec:c1:46:b9:b1
Generator (uncompressed):
 04:18:8d:a8:0e:b0:30:90:f6:7c:bf:20:eb:43:a1:
 88:00:f4:ff:0a:fd:82:ff:10:12:07:19:2b:95:ff:
 c8:da:78:63:10:11:ed:6b:24:cd:d5:73:f9:77:a1:
 1e:79:48:11
Order:
 00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:99:de:
 f8:36:14:6b:c9:b1:b4:d2:28:31
Cofactor: 1 (0x1)
Seed:
 30:45:ae:6f:c8:42:2f:64:ed:57:95:28:d3:81:20:
 ea:e1:21:96:d5
-----BEGIN EC PARAMETERS-----
MIHHAqEBMCQGBYqGSM49AQECGQD/////////+//////////8wSwQY
//////////v//////////8BBhkIQUZ5ZyA5w+n6atyJDBJ/rje7MFG
ubEDFQAwRa5vyEivZO1XLSjTgSDq4SGW1QQxBBiNqA6wMJD2fL8g600hiAD0/wr9
gv8QEgcZK5X/yNp4YxAR7WskzdVz+XehHnliEQIZAP//////////5ne+DYU
a8mxtNIoMQIBAQ==
-----END EC PARAMETERS-----
```

21. Archivo con c.e. prime192v1

Curva B-163

Tiene por ecuación $y^2 + xy = x^3 - x^2 + B$

Donde $B = 0x\ 00000002\ 0a601907\ b8c953ca\ 1481eb10\ 512f\ 7874\ 4a3205fd$.

Ejecutamos la posible opción, en este caso la curva perteneciente a sect163r2.

Con -name pondremos alguno de los nombres contenidos en la lista anterior y con la opción -param-enc especificaremos como se codificarán los datos, nosotros escogeremos explicit para darlos explícitamente y tener más información a la hora de mostrar resultados.

```
openssl ecparam -name sect163r2 -param_enc explicit -out stdECparam.pem
```

22. Comando para crear curva eliptica b-163

Mostramos los resultados.

Esta vez con la opción -text nos mostrará los parámetros de forma legible.

```
antonio@antonio-DELL:~/Escritorio$ openssl ecparam -in stdECparam.pem -text
```

23. Comando para mostrar curva eliptica

Y como vemos coincide con B.

```
Field Type: characteristic-two-field
Basis Type: ppBasis
Polynomial:
    08:00:00:00:00:00:00:00:00:00:00:00:00:00:
    00:00:00:00:00:00:c9
A:      1 (0x1)
B:
    02:0a:60:19:07:b8:c9:53:ca:14:81:eb:10:51:2f:
    78:74:4a:32:05:fd
Generator (uncompressed):
    04:03:f0:eb:a1:62:86:a2:d5:7e:a0:99:11:68:d4:
    99:46:37:e8:34:3e:36:00:d5:1f:bc:6c:71:a0:09:
    4f:a2:cd:d5:45:b1:1c:5c:0c:79:73:24:f1
Order:
    04:00:00:00:00:00:00:00:00:00:02:92:fe:77:e7:
    0c:12:a4:23:4c:33
Cofactor:  2 (0x2)
-----BEGIN EC PARAMETERS-----
MIGNAgEBMCUGByqGSM49AQIwGgICAKMGCSqGSM49AQIDAzAJAgEDAgEGAgEHMB0E
AQEEFQIKYBkHuMLTyhSB6xBRL3h0SjIF/QQrBAPw66FihqLVfqCZEWjUmUY36DQ+
NgDVH7xscaAJT6LN1UWxHFWMeXMk8QIVBAAAAAAAAAAAAAKS/nfnDBKkI0wzAgEC
-----END EC PARAMETERS-----
```

24. Archivo con c.e. sect163r2

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

Escogemos la primera curva P-192.

Generamos la clave privada con la opción -genkey.

```
openssl ecparam -in stdECparam.pem -genkey -out nombreECkey.pem
```

25. Comando para generar clave con curva p-192

Mostramos los resultados.

```
antonio@antonio-DELL:~/Escritorio$ cat nombreECkey.pem
-----BEGIN EC PARAMETERS-----
MIHHAqEBMCQGBYqGSM49AQECGQD//////////+//////////8wSwQY
//////////v//////////8BBhkIQUZ5ZyA5w+n6atyJDBJ/rje7MFG
ubEDFQAwRa5vyEivZO1XlSjTgSDq4SGW1QQxBBiNqA6wMJD2fL8g600hiAD0/wr9
gv8QEgcZK5X/yNp4YxAR7WskzdVz+XehHnliEQIZAP//////////5ne+DYU
a8mxtNIoMQIBAQ==
-----END EC PARAMETERS-----
-----BEGIN EC PRIVATE KEY-----
MIIBIAIBAQQY8hVLzVVXuciGy4vICl+uVb9tyeVLp0boIHKMIHHAqEBMCQGBYqG
SM49AQECGQD//////////+//////////8wSwQY//////////
//////////v//////////8BBhkIQUZ5ZyA5w+n6atyJDBJ/rje7MFGubEDFQAwRa5vyEiv
ZO1XlSjTgSDq4SGW1QQxBBiNqA6wMJD2fL8g600hiAD0/wr9gv8QEgcZK5X/yNp4
YxAR7WskzdVz+XehHnliEQIZAP//////////5ne+DYUa8mxtNIoMQIBAAE0
AzIABD1k05kF4BFfW/vjKUenLP3ltChu84Pn4B0IJqvC7yANBo5EzAyC45TKbhcc
DWXPDQ==
-----END EC PRIVATE KEY-----
```

26. Archivo con clave de c.e.

“Extraed” la clave privada contenida en el archivo nombre>ECkey.pem a otro archivo que tenga por nombre <nombre>ECpriv.pem. Este archivo deberá estar protegido por contraseña. Mostrad sus valores.

Para el tratamiento de las claves utilizaremos openssl ec, pasándole un archivo de entrada y otro de salida, cifrando la clave privada. Por defecto se extrae la clave privada.

El manual especifica -des, -des3, -idea o cualquier otro sistema. Nosotros seguiremos con aes que ya hemos utilizado. Cifraremos la clave privada con aes 256 en modo cbc, con clave 0123456789.

openssl ec -in nombreECkey.pem -out nombreECpriv.pem -aes-256-cbc

```
antonio@antonio-DELL:~/Escritorio$ openssl ec -in nombreECkey.pem -out nombreECpriv.pem -aes-256-cbc
read EC key
writing EC key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

27. Comando para extraer clave privada y cifrarla c.e.

Mostramos los resultados.

```
antonio@antonio-DELL:~/Escritorio$ cat nombreECpriv.pem
-----BEGIN EC PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,5AC3E668170B803FC1409CC65326497C

DtIgEhRkgcnroTtGASc4H8QjH3eASpb7F85jCIHQ5L91nAecw6RzWEKVEWgNwhdO
OUX1acMCAaHanOKTryd/Df7WYwGON021tLOim1GGABbhmWNY+BFfuk/uVLD0X6as
gP8BT8EkZogdND7BL/GJmtP7Epn/G0lQj50MSJkXUBuxnZ+sfYi2qdo+rO+ZWnn2
6P4ErKRLjxKhtUnSbIs2Ffl3aiappixwXsILjF0inGfKH5xX/pksiHh1hzdqUqCl
l+cGCEqPZuMX6/DfWHKKszQw1Ya3CaIL3c0/fdjAfigXDdmTy+Cg253vSyIiUUSg
DGl3t3cL2hWSrd9QHLg6x2LJz3h27ldf0md4Puopd/eY/u3+Jlu6JKN/77fXZwFpo
SpvLcFRnHrgazrzUIInYiBw==
-----END EC PRIVATE KEY-----
```

28. Archivo clave privada cifrada c.e.

Si accedemos al archivo introduciendo la contraseña nos saldrán las claves.

Para mostrar los valores utilizaremos la opción -text de ec la cual nos imprimirá los diferentes componentes y parámetros de la clave pública y privada.

```
openssl ec -in nombreECpriv.pem -text
```

```

antonio@antonio-DELL:~/Escritorio$ openssl ec -in nombreECpriv.pem -text
read EC key
Enter PEM pass phrase:
Private-Key: (192 bit)
priv:
    00:f2:15:4b:cd:55:57:b9:c8:aa:1b:2e:2f:20:29:
    7e:b9:56:fd:b7:27:95:2e:9d:1b
pub:
    04:3d:64:d3:99:05:e0:11:5f:5b:fb:e3:29:47:a7:
    2c:fd:e5:b4:28:6e:f3:83:e7:e0:1d:08:26:ab:c2:
    ef:20:0d:06:8e:44:cc:0c:82:e3:94:ca:6e:17:1c:
    0d:65:cf:0d
Field Type: prime-field
Prime:
    00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
    ff:fe:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
A:
    00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
    ff:fe:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
B:
    64:21:05:19:e5:9c:80:e7:0f:a7:e9:ab:72:24:30:
    49:fe:b8:de:ec:c1:46:b9:b1
Generator (uncompressed):
    04:18:8d:a8:0e:b0:30:90:f6:7c:bf:20:eb:43:a1:
    88:00:f4:ff:0a:fd:82:ff:10:12:07:19:2b:95:ff:
    c8:da:78:63:10:11:ed:6b:24:cd:d5:73:f9:77:a1:
    1e:79:48:11
Order:
    00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
    f8:36:14:6b:c9:b1:b4:d2:28:31
Cofactor: 1 (0x1)
Seed:
    30:45:ae:6f:c8:42:2f:64:ed:57:95:28:d3:81:20:
    ea:e1:21:96:d5
writing EC key
-----BEGIN EC PRIVATE KEY-----
MIIBIAIBAQQY8hVLzVZXuciGy4vICl+uVb9tyeVLP0boIHKMIHHAgEBMCQGBYqG
SM49AQECGQD//////////+//////////8wSwQY//////////
//////////8BBhkIQUZ5ZyA5w+n6atyJDBJ/rje7MFGubEDFQAwRa5vyEIV
Z01XlSjTgSDq4SGW1QXBBiNqA6wMJD2fL8g600hiAD0/wr9gv8QEgcZK5X/yNp4
YxAR7WskzdVz+XehHnLIEQIZAP//////////5ne+DYUa8mxtNIoMQIBAAE0
AzIABD1k05kF4BFfW/vjKUenLP3ltChu84Pn4B0IJqvc7yANBo5EzAyC45TKbhcc
DWXPDQ==
-----END EC PRIVATE KEY-----

```

29. Archivo claves descifrado c.e.

Extraed en <nombre>ECpub.pem la clave publica contenida en el archivo <nombre>ECkey.pem. Como antes <nombre>ECpub.pem no debe estar cifrado ni protegido. Mostrad sus valores.

Extraemos la clave publica con la opción -pubout como ya sabemos.

`openssl ec -in nombreECkey.pem -out nombreECpub.pem -pubout`

```

antonio@antonio-DELL:~/Escritorio$ openssl ec -in nombreECkey.pem -out nombreECpub.pem -pubout
read EC key
writing EC key

```

30. Comando para extraer clave publica c.e.

Para mostrarlo en este caso especificaremos la publica con -pubin.
Como vemos parece correcto el valor de la clave publica viendo el archivo anterior.

openssl ec -in nombreECpub.pem -pubin -text

```
antonio@antonio-DELL:~/Escritorio$ openssl ec -in nombreECpub.pem -pubin -text
read EC key
Private-Key: (192 bit)
pub:
    04:3d:64:d3:99:05:e0:11:5f:5b:fb:e3:29:47:a7:
    2c:fd:e5:b4:28:6e:f3:83:e7:e0:1d:08:26:ab:c2:
    ef:20:0d:06:8e:44:cc:0c:82:e3:94:ca:6e:17:1c:
    0d:65:cf:0d
Field Type: prime-field
Prime:
    00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
    ff:fe:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
A:
    00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
    ff:fe:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:fc
B:
    64:21:05:19:e5:9c:80:e7:0f:a7:e9:ab:72:24:30:
    49:fe:b8:de:ec:c1:46:b9:b1
Generator (uncompressed):
    04:18:8d:a8:0e:b0:30:90:f6:7c:bf:20:eb:43:a1:
    88:00:f4:ff:0a:fd:82:ff:10:12:07:19:2b:95:ff:
    c8:da:78:63:10:11:ed:6b:24:cd:d5:73:f9:77:a1:
    1e:79:48:11
Order:
    00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:99:de:
    f8:36:14:6b:c9:b1:b4:d2:28:31
Cofactor: 1 (0x1)
Seed:
    30:45:ae:6f:c8:42:2f:64:ed:57:95:28:d3:81:20:
    ea:e1:21:96:d5
writing EC key
-----BEGIN PUBLIC KEY-----
MIIBCjCB0wYHKoZIzj0CATCBxwIBATAkBgqhkJOPQEBAhKA//////////
////v/////////MEsEGP/////////7/////////AQYZCEFGewc
g0cPp+mrciQwSf643uzBRrmxAxUAMEWub8hCL2TtV5Uo04Eg6uEhltUEMQQYjag0
sDCQ9ny/I0tDoYgA9P8K/YL/EBIHGSuV/8jaeGMQEe1rJM3Vc/l3oR55SBECGQD/
//////////+Z3vg2FGvJsbTSKDECAQEDMgAEPWTTmQXgEV9b++MpR6cs/eW0
KG7zg+fgHQgmq8LvIA0GjkTMDILjLMpuFwxNZc8N
-----END PUBLIC KEY-----
```

31. Archivo clave publica c.e.

Biografía

[1.] Manual de OpenSSL.

<https://www.openssl.org/docs/man1.0.2/apps/>