SEGURIDAD Y PROTECCIÓN DE SISTEMAS INFORMÁTICOS (2018-2019)

GRADO EN INGENIERÍA INFORMÁTICA

UNIVERSIDAD DE GRANADA

PRACTICA 2: CRIPTOSISTEMAS ASIMÉTRICOS

JUAN ALBERTO RIVERA PEÑA

TAREAS

1.-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.

Generamos la clave RSA de 901 bits con el comando :
openssl genrsa -out <nombre>RSAkey.pem 901

```
juan@juan-X541UAK:~/Documentos/Spsi/PRACTICAS/PRACTICA2$ openssl genrsa -out JuanAlbertoRSAkey.pem 901
Generating RSA private key, 901 bit long modulus
......++++++
..............++++++
e is 65537 (0x10001)
```

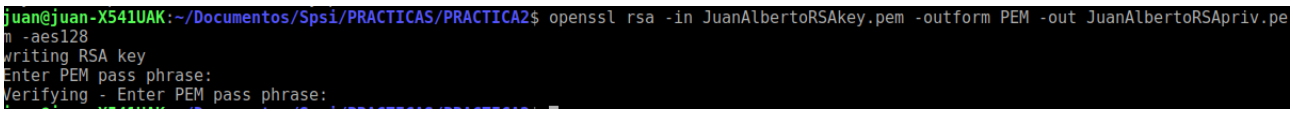
JuanAlbertoRSAkey.pem:

```
-----BEGIN RSA PRIVATE KEY-----
MIICFQIBAAJxFy7zkxUdZkgi0B2mFp/yj0V7ymjd7gWb9XBeQwH8AnSYdTRprmv2
6VJ54Lr91DmurvHP/59WB82c2erGaYMSzc+PVYTemeSb9uyHAZEiIO5JWOWTRQK9
EtpVT+cdoKFoJrbdXkmCBfJGdxa1CHCC8JMCAwEAAQJxA1rcDNDlbEPMJenkeymy
ZGFdxLDgTO2q4B3exjunSIOu/GcVVBRzqtgwowDscqXWTaLYwtlADekF7LMPmSsW
PAvcibudJCeJgHq+bV3rS8T9ji01gdUMhU1isW8skLGScAv9EGABw9Isnr1hHs+Z
eEECOQa9cn7WxYEPyhKGK0s+6XZJiHpeQpSI+FdjqMlu5erWqQTnseiaowO+EhCg
pdeLLdHP6sM7dWkP6wI5A3CNirmlSrh1VH5eCGcONqDcCPbCY3BvfC8JFCWCrsHP
Rri0PELU9GYYTJGqhLG5+HYZRO747x/5AjkC6nJd7QwEiZAh82lHZDAbvDQOvDiR
ia/mHncmrZNJxRPZy2Jr/noHZHLsPpOAa/WmjwxA6bghe4ECOQL7pYTdo1+9wONa
BCq9VTJDJxAPEF40sDq+f90c+sycACqRHJ32R8D8ma4t1ffmOiYkJZ4GsZSP4QI5
AxGkyr6ZZOSaprVRYPy214h/Aelp2Yi1oDQ7NKp2+SRlRVcnjmb7mzhjip5fRvrw
T5zICDngBdCd
-----END RSA PRIVATE KEY-----
```

2.-"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.

Extraemos la clave privada y la ciframos con:
openssl rsa -in <nombre>RSAkey.pem -outform PEM -out <nombre>RSApriv.pem -aes128

En mi caso:



Donde la frase secreta que he utilizado para el PEM es: 0123456789

JuanAlbertoRSApriv.pem:

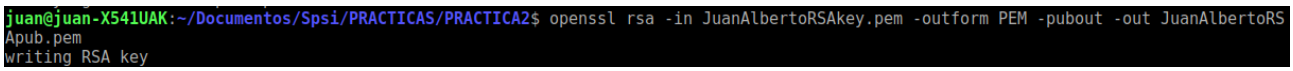
```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,05BB87093B53A519092C6F90F18C9E8F

NSSxKhSCtmPwwAAka894g7yvvYDuaiu2Be6g8hu6ezPtj4htQIvuleYf+tKxyWr9
cIV0nqsijQ1hIepUBQNCN2z8JoVlOToM+LOUUysgxLcmf//ADcsgDkhW6MOMuhBW
NGc+Q4eMjkwHso5E/kMRCxQposF2PU/8UCtp+/VFaSUSrhZbhrKPINElW6X5MD51
YYPAjpL6a39O3P4+24OOOEqEI5fLIFqIA6haQR9+m47Nu4XcOb75RQ3XjiFwNslq
sfvw9JOn7XaFT2UnLdwg6ZSts+Kha/P5j/aAQUJQMUbzakfu5kv9cjeAwPIC15l5
GnjbEqTm7C+8/t98yXfHtQdpT/WIPbwaxWXSw+BNwTB11muke4Xey665LdBU/u0q
AULlEDUqqyEDaETVrGTD9gnxanUn/RNXzF5gPNDRN/tfprz95vEBi26+im+BCiKM
VFg85vHW9Jg0ZtiEVc0fKxfKNNa4Xh1G2hEl7nwo34k34QwYePrxKvsjiTZTLkK7
2E8Yquk5ueBTBcmeLeQSdeMN02ur+vil2s8zmCpbgQt78JfnfCVH7B4/xD4GAX1v
ldcP2PVRyYoWJKr476eBleYvXD2ChavxmFdjyS675arl/IZ1swNbc4LmPfAE8GUe
3WzFLfikPOxKlgV3gRQKdw0M1XaIZ0VglZ/b/F/P7fturPithpgNM/hqwsuWdWkw
U2kah4VKtaz7KCOGZNLpUw==
-----END RSA PRIVATE KEY-----
```

3.-Extraed en <nombre>RSApub.pem la clave pública contenida en el archivo <nombre>RSAkey.pem . Evidentemente <nombre>RSApub.pem no debe estar cifrado ni protegido. Mostrad sus valores.

Extraemos la clave publica sin proteccion con:

openssl rsa -in <nombre>RSAkey.pem -outform PEM -pubout -out <nombre>RSApub.pem



JuanAlbertoRSApub.pem:

```
-----BEGIN PUBLIC KEY-----

MIGMMA0GCSqGSIb3DQEBAQUAA3sAMHgCcRcu85MVHWZIItAdphaf8o9Fe8po3e4F
m/VwXkMB/AJ0mHU0aa5r9ulSeeC6/dQ5rq7xz/+fVgfNnNnqxmmDEs3Pj1WE3pnk
m/bshwGRIiDuSVjlk0UCvRLaVU/nHaChaCa23V5JggXyRncWtQhwgvCTAgMBAAE=
-----END PUBLIC KEY-----
```

4.-Reutilizaremos el archivo binario input.bin de 1024 bits, todos ellos con valor 0 , de la práctica anterior. Intentad cifrar input.bin con vuestras claves pública. Explicad el mensaje de error obtenido.

Intentamos cifrar el archivo input.bin con la clave publica extraida en el ejercicio anterior:

```
juan@juan-X541UAK:~/Documentos/Spsi/PRACTICAS/PRACTICA2$ openssl rsautl -encrypt -pubin -inkey JuanAlbertoRSApub.pem -in input.bin -ou
t salida.enc
```

Y nos reporta el siguiente error:

```
RSA operation error
140014924958464:error:0406D06E:rsa routines:RSA_padding_add_PKCS1_type_2:data too large for key size:rsa_pk1.c:153:
```

El error lo da debido a que la criptografía de clave pública no es para cifrar archivos largos a diferencia de los cifrados que actúan sobre bloques. Por esto se utilizan los cifrados hibridos.

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

1. **El emisor debe seleccionar un sistema simétrico con su correspondiente modo de operación.**
2. **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 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 empleará el algoritmo Blow sh en modo ECB, la segunda línea debería contener -bf-ecb .**
3. **El archivo sessionkey se cifrará con la clave pública del receptor.**
4. **El mensaje se cifrará utilizando el criptosistema simétrico, la clave se generará a partir del archivo anterior mediante la opción**

-pass file:sessionkey .

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.

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.

Mostramos la lista de curvas disponibles:

```
juan@juan-X541UAK:~/Documentos/Spsi/PRACTICAS/PRACTICA2$ 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
```

Ahora seleccionamos una, por ejemplo voy a utilizar ‘secp192k1’ y procedemos a generar el archivo pem:

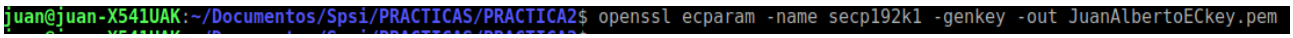
```
juan@juan-X541UAK:~/Documentos/Spsi/PRACTICAS/PRACTICA2$ openssl ecparam -name secp192k1 -out stdECparam.pem
```

Y procedemos a comprobar los parámetros de la curva:

```
juan@juan-X541UAK:~/Documentos/Spsi/PRACTICAS/PRACTICA2$ openssl ecparam -in stdECparam.pem -check
checking elliptic curve parameters: ok
-----BEGIN EC PARAMETERS-----
BgUrgQQAHw==
-----END EC PARAMETERS-----
```

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

Generamos la clave privada para los parámetros anteriores sin contraseña:
openssl ecparam -name secp160k1 -genkey -out <nombre>ECkey.pem

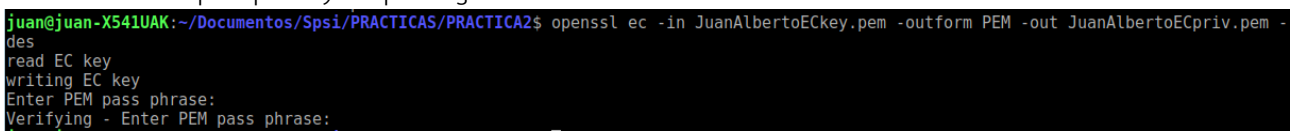


JuanAlbertoECkey.pem:

```
-----BEGIN EC PARAMETERS-----
BgUrgQQAHw==
-----END EC PARAMETERS-----
-----BEGIN EC PRIVATE KEY-----
MFwCAQEEGH1MMbNzm3pryYDQqGtMPLzWN+dmQigkkqAHBgUrgQQAH6E0AzIABI9T
092No4t0SDimif24IJwMbqP8dxl1BVFVf9vu5aIEhr0j2L7mcVe8CRe2R46uMA==
-----END EC PRIVATE KEY-----
```

9.-"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.

Extraemos la clave privada contenida en el archivo JuanAlbertoECkey.pem a el archivo JuanAlbertoECpriv.pem y lo protegemos con contraseña:



Donde la contraseña es: 0123456789

JuanAlbertoECpriv.pem:

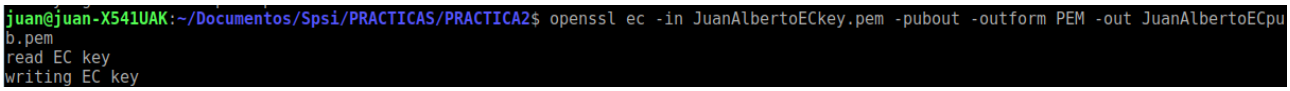
```
-----BEGIN EC PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,C976C47F3B3543DA

f5SUH/w9Kl1Dwe1dH09FyiBSaPDZiw19IIDS0WC1XDvLHn+R9f2+gyek2TR2wDr4
ycxve5ojSydl6RMFtkS2JCF4VAWXqlI/MpbRUvIpK45Brrqw2loZFmvW4cervEMw
-----END EC PRIVATE KEY-----
```

10.-Extraed en <nombre>ECpub.pem la clave pública 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:

openssl ec -in <nombre>ECkey.pem -pubout -outform PEM -out <nombre>ECpub.pem



JuanAlbertoECpub.pem:

```
-----BEGIN PUBLIC KEY-----
MEYwEAYHKoZIzj0CAQYFK4EEAB8DMgAEj1PT3Y2ji3RIOKaJ/bggnAxuo/x3GXUF
UVV/2+7logSGvSPYvuZxV7wJF7ZHjq4w
-----END PUBLIC KEY-----
```