Práctica 3: Protocolos criptográficos

> Ángel Gómez Martín
>
> agomezm@correo.ugr.es
>
> Seguridad y Protección de Sistemas Informáticos
>
> UGR 2018-19

Tareas

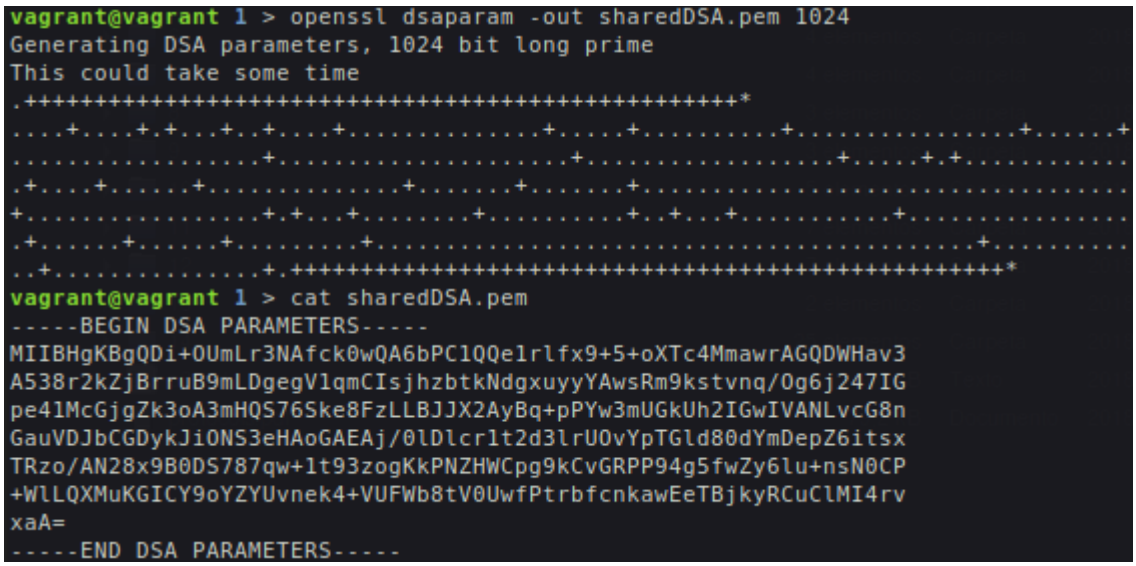
> Para determinar la sintaxis de las ordenes a utilizar me he basado en el manual de OpenSSL.

1

Para generar el archivo utilizo la siguiente orden:

```
openssl dsaparam -out sharedDSA.pem 1024
```

- *-out*: Fichero de salida.
- *1024*: Tamaño de la clave.

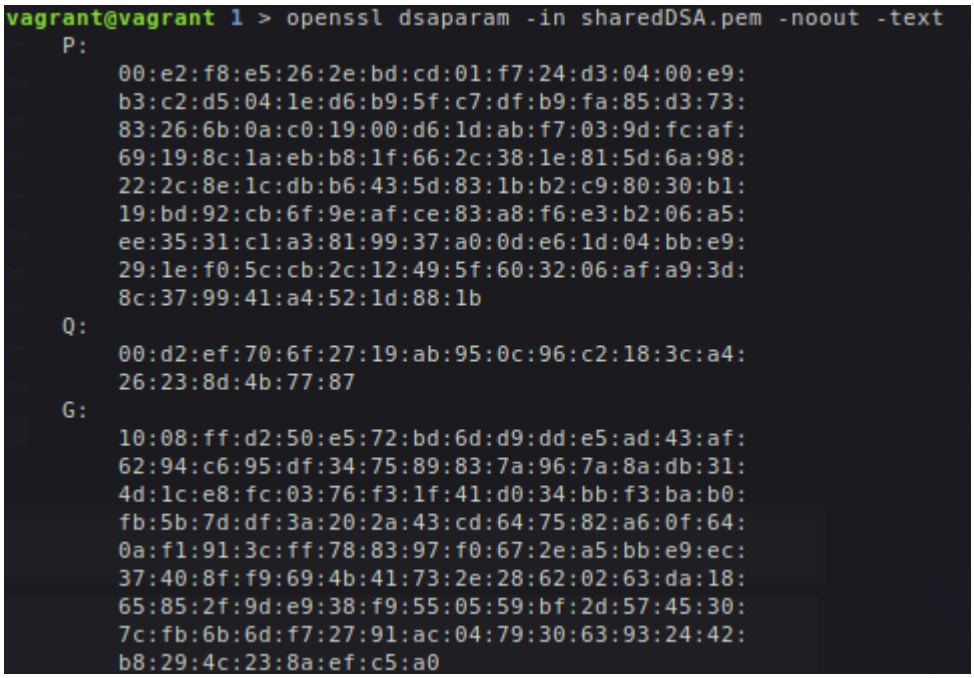


Por otro lado si queremos ver los valores generados (*P, Q y G*) uso la orden siguiente:

```
openssl dsaparam -in sharedDSA.pem -noout -text
```

- *-in*: Archivo de entrada.

- *-noout*: No devuelve el resultado en un archivo.
- *-text*: Devuelve el resultado por pantalla en hexadecimal.

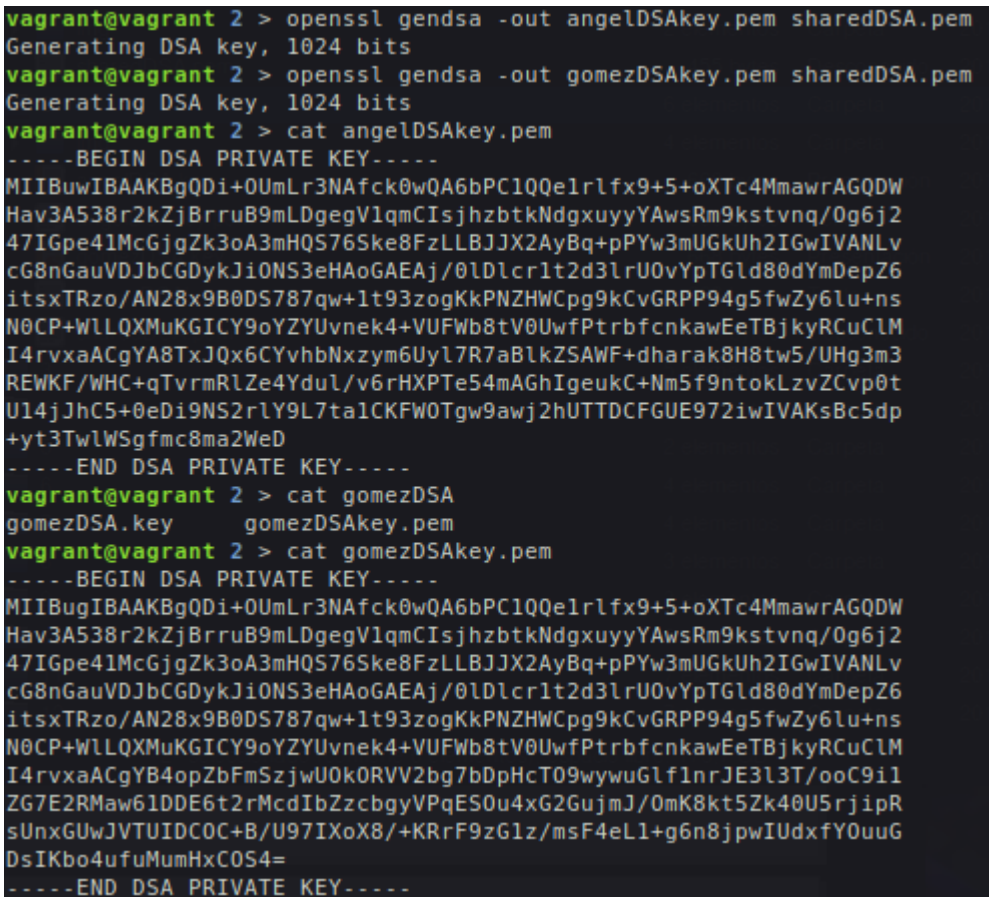


2

Para generar el par de claves uso las siguientes ordenes:

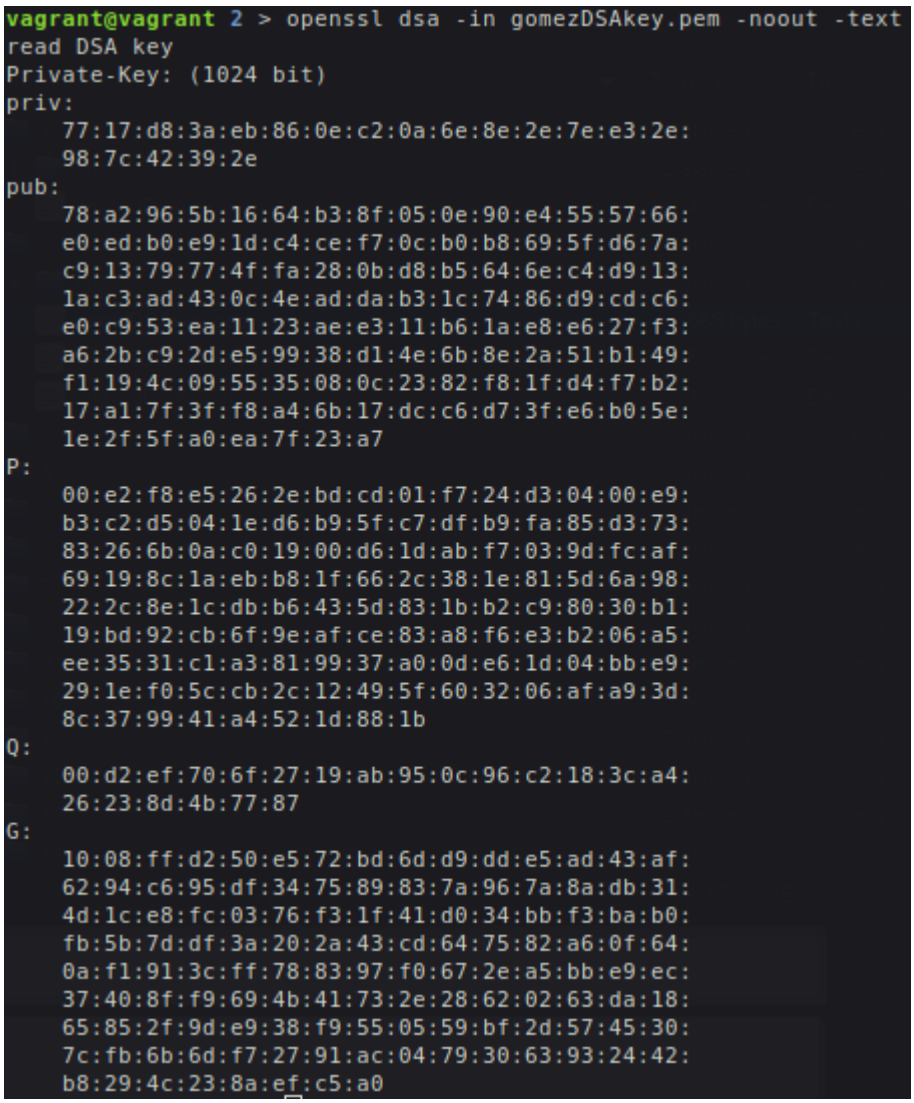
```
openssl gendsa -out angelDSAkey.pem sharedDSA.pem
openssl gendsa -out gomezDSAkey.pem sharedDSA.pem
```

- *-out*: Archivo de salida.
- *sharedDSA.pem*: Archivo del cual se toman los parámetros para generar las claves.



Y también sus parámetros:

```
vagrant@vagrant 2 > openssl dsa -in angelDSAkey.pem -noout -text
read DSA key
Private-Key: (1024 bit)
priv:
    00:ab:01:73:97:69:fb:2b:77:4f:09:56:4a:07:e6:
    73:c9:9a:d9:67:83
pub:
    3c:4f:12:50:c7:a0:98:be:16:cd:c7:3c:a6:e9:4c:
    a5:ed:1e:da:06:59:19:48:05:85:f9:d8:5a:ad:a9:
    3c:1f:cb:70:e7:f5:07:83:79:b7:44:45:8a:17:f5:
    87:0b:ea:93:be:b9:91:95:97:b8:61:db:a5:fe:fe:
    ab:1d:73:d3:7b:9e:26:00:68:48:81:eb:a4:0b:e3:
    66:e5:ff:67:b6:89:0b:ce:f6:42:be:9d:2d:53:5e:
    23:26:10:b9:fb:47:83:8b:d3:52:da:b9:58:f4:be:
    ed:6b:50:8a:15:63:93:83:0f:5a:c2:3d:a1:51:34:
    c3:08:51:94:13:de:f6:8b
P:
    00:e2:f8:e5:26:2e:bd:cd:01:f7:24:d3:04:00:e9:
    b3:c2:d5:04:1e:d6:b9:5f:c7:df:b9:fa:85:d3:73:
    83:26:6b:0a:c0:19:00:d6:1d:ab:f7:03:9d:fc:af:
    69:19:8c:1a:eb:b8:1f:66:2c:38:1e:81:5d:6a:98:
    22:2c:8e:1c:db:b6:43:5d:83:1b:b2:c9:80:30:b1:
    19:bd:92:cb:6f:9e:af:ce:83:a8:f6:e3:b2:06:a5:
    ee:35:31:c1:a3:81:99:37:a0:0d:e6:1d:04:bb:e9:
    29:1e:f0:5c:cb:2c:12:49:5f:60:32:06:af:a9:3d:
    8c:37:99:41:a4:52:1d:88:1b
Q:
    00:d2:ef:70:6f:27:19:ab:95:0c:96:c2:18:3c:a4:
    26:23:8d:4b:77:87
G:
    10:08:ff:d2:50:e5:72:bd:6d:d9:dd:e5:ad:43:af:
    62:94:c6:95:df:34:75:89:83:7a:96:7a:8a:db:31:
    4d:1c:e8:fc:03:76:f3:1f:41:d0:34:bb:f3:ba:b0:
    fb:5b:7d:df:3a:20:2a:43:cd:64:75:82:a6:0f:64:
    0a:f1:91:3c:ff:78:83:97:f0:67:2e:a5:bb:e9:ec:
    37:40:8f:f9:69:4b:41:73:2e:28:62:02:63:da:18:
    65:85:2f:9d:e9:38:f9:55:05:59:bf:2d:57:45:30:
    7c:fb:6b:6d:f7:27:91:ac:04:79:30:63:93:24:42:
    b8:29:4c:23:8a:ef:c5:a0
```



3

> La contraseña utilizada para el cifrado ha sido: 0123456789

Del mismo modo que se hacía en prácticas anteriores, la sintaxis del comando para extraer la clave privada es muy similar en el caso de DSA, en este caso es el siguiente:

```
openssl dsa -in angelDSAkey.pem -out angelDSApriv.pem -aes256
openssl dsa -in gomezDSAkey.pem -out gomezDSApriv.pem -aes256
```

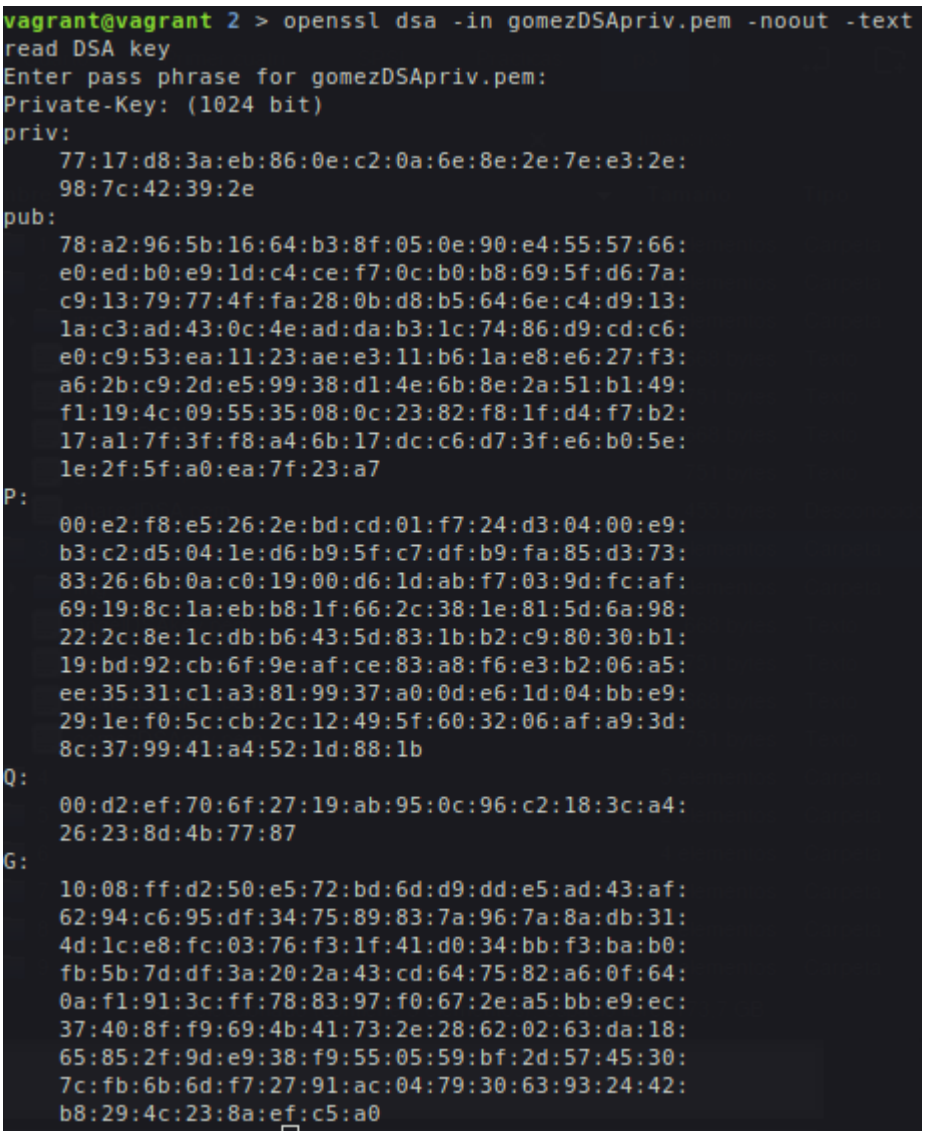
- *-in*: Archivo de entrada.
- *-out*: Archivo de salida.
- *-aes256*: Método de cifrado.

```
vagrant@vagrant 2 > openssl dsa -in angelDSAkey.pem -out angelDSApriv.pem -aes256
read DSA key
writing DSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
vagrant@vagrant 2 > openssl dsa -in gomezDSAkey.pem -out gomezDSApriv.pem -aes256
read DSA key
writing DSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
vagrant@vagrant 2 > cat angelDSApriv.pem
-----BEGIN DSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,2DA4F7118644B74233C7C1DD3C5553A7

PweqMvO8Ml77h37/UDDkFweabFto90BEqOY5SPY/uRP5NeK7Wuj0YBCOfejxldum
31WfyOi0dOvw++YUv+9+lP/8oOhch8n5GxkWeQTOWwUlNF1/zgLYpfID0EPCkzrN
Mne5Ze40pUnaA5jSW2ZJ30Dbu4zc5S6hOv8SUZ0TCVsaY3y15jHHQWYD0o6kD+SO
pAOekZqsZQh6kIP+5kK6BFg0z+SROBsUFQMzdsRvtSOXQZdVhymmJBHsgrNMO2Uu
S++2IWuP9spSfl7sfM4PnkOYhxEOPUEqtGgO4yF3Mfhz0FfhsFp35Q2PNusN8XnM
A1V/AUf+lP9kph770NuDfnkCcb+4lNmslmipiyB0Q21Acw+OHKvrLUMgL2sX58q4
7yQaIr/cyp+QBYpXSDu4mX13WBMTCkCP1W3PBZ9t3Rab0vyDRBmIk4IENsA5f3M0
SE4ilT5/eVpHSv/17LrMj/gTWhN8zYsQ7gNOEiRH0/R03kJHR2UdQpVr8lJI6Y7J
K5R+V+6X77O1gKKyfiDWWSfWG1Q9SyEakbMmBkyS2FHShyPsvzr9cBseOpr/wECj
mZWSNY+JRK8THERTyrwXEw==
-----END DSA PRIVATE KEY-----
vagrant@vagrant 2 > cat gomezDSApriv.pem
-----BEGIN DSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,E012B12D4018C84DB9724AF1FD067516

UTpeY3FKjvzlpvUFGiBmJ0+j7vsPOsmeU+dYqqd2bm+Qq16AhMTbTb5CE4F2DD8I
eLpvmTjmwvk10vbWNUsL5oABvyHLIW88mvbLquuxVbnosJ+0rT/zXo1iPJcUZHe5
L9RoQU31tG47dDA/iB+cVcbAIf7R/wBlMly8kbZIkG53zYFmZzHMopWsOPNK13YP
o5bdULh2rlQvIgoA1PGCe82BaiU4ovd42J0wuuP6nRxhmu3HjAm+wETIxplaifUd
gIY/11vMM1j3v3zVZ/Euce5OurdiOVcCokLiQbFPvhiYp7aEKyZ9wg4nAUDJS/4T
TqjCg8qKqns5aSnfM5qALMYhwQUnJFhFw6HOB9k0fTDSBcwDGKcoi4w1KeIorTJX
vmXkmigS1ZZsbW3OP28YIxoKmf65x0TAudfWcOmXASTIYwvnWREA0ixNNpuCsLOr
F8y0z0UAvY5QAojA1BQoni4HSH+f1Sl2sIScOh3ZFaqiYTBeAdmTnjuQLoM7Lm2D
WWZ+V5HDYarksBsmzj7I3p8u0d1VE3yyIW1eUzyHBgazziR7hNya+0WGUi7xeR4K
E5C+37aVsqdbddzWqY2qJQ==
-----END DSA PRIVATE KEY-----
```

```
vagrant@vagrant 2 > openssl dsa -in angelDSApriv.pem -noout -text
read DSA key
Enter pass phrase for angelDSApriv.pem:
Private-Key: (1024 bit)
priv:
    00:ab:01:73:97:69:fb:2b:77:4f:09:56:4a:07:e6:
    73:c9:9a:d9:67:83
pub:
    3c:4f:12:50:c7:a0:98:be:16:cd:c7:3c:a6:e9:4c:
    a5:ed:1e:da:06:59:19:48:05:85:f9:d8:5a:ad:a9:
    3c:1f:cb:70:e7:f5:07:83:79:b7:44:45:8a:17:f5:
    87:0b:ea:93:be:b9:91:95:97:b8:61:db:a5:fe:fe:
    ab:1d:73:d3:7b:9e:26:00:68:48:81:eb:a4:0b:e3:
    66:e5:ff:67:b6:89:0b:ce:f6:42:be:9d:2d:53:5e:
    23:26:10:b9:fb:47:83:8b:d3:52:da:b9:58:f4:be:
    ed:6b:50:8a:15:63:93:83:0f:5a:c2:3d:a1:51:34:
    c3:08:51:94:13:de:f6:8b
P:
    00:e2:f8:e5:26:2e:bd:cd:01:f7:24:d3:04:00:e9:
    b3:c2:d5:04:1e:d6:b9:5f:c7:df:b9:fa:85:d3:73:
    83:26:6b:0a:c0:19:00:d6:1d:ab:f7:03:9d:fc:af:
    69:19:8c:1a:eb:b8:1f:66:2c:38:1e:81:5d:6a:98:
    22:2c:8e:1c:db:b6:43:5d:83:1b:b2:c9:80:30:b1:
    19:bd:92:cb:6f:9e:af:ce:83:a8:f6:e3:b2:06:a5:
    ee:35:31:c1:a3:81:99:37:a0:0d:e6:1d:04:bb:e9:
    29:1e:f0:5c:cb:2c:12:49:5f:60:32:06:af:a9:3d:
    8c:37:99:41:a4:52:1d:88:1b
Q:
    00:d2:ef:70:6f:27:19:ab:95:0c:96:c2:18:3c:a4:
    26:23:8d:4b:77:87
G:
    10:08:ff:d2:50:e5:72:bd:6d:d9:dd:e5:ad:43:af:
    62:94:c6:95:df:34:75:89:83:7a:96:7a:8a:db:31:
    4d:1c:e8:fc:03:76:f3:1f:41:d0:34:bb:f3:ba:b0:
    fb:5b:7d:df:3a:20:2a:43:cd:64:75:82:a6:0f:64:
    0a:f1:91:3c:ff:78:83:97:f0:67:2e:a5:bb:e9:ec:
    37:40:8f:f9:69:4b:41:73:2e:28:62:02:63:da:18:
    65:85:2f:9d:e9:38:f9:55:05:59:bf:2d:57:45:30:
    7c:fb:6b:6d:f7:27:91:ac:04:79:30:63:93:24:42:
    b8:29:4c:23:8a:ef:c5:a0
```



4

La orden para extraer la clave pública tiene la misma forma que el usado en la práctica anterior, es el siguiente:

```
openssl dsa -in angelDSAkey.pem -out angelDSApub.pem -pubout
openssl dsa -in gomezDSAkey.pem -out gomezDSApub.pem -pubout
```

- *-in*: Archivo de entrada.
- *-out*: Archivo de salida.
- *-pubout*: Indica que se extrae la clave pública.

```
vagrant@vagrant 4 > openssl dsa -in angelDSAkey.pem -out angelDSApub.pem -pubout
read DSA key
writing DSA key
vagrant@vagrant 4 > openssl dsa -in gomezDSAkey.pem -out gomezDSApub.pem -pubout
read DSA key
writing DSA key
vagrant@vagrant 4 > cat angelDSAkey.pem
-----BEGIN DSA PRIVATE KEY-----
MIIBuwIBAAKBgQDi+OUmLr3NAfck0wQA6bPC1QQe1rlfx9+5+oXTc4MmawrAGQDW
Hav3A538r2kZjBrruB9mLDgegV1qmCIsjhzbtkNdgxuyyYAwsRm9kstvnq/Og6j2
47IGpe41McGjgZk3oA3mHQS76Ske8FzLLBJJX2AyBq+pPYw3mUGkUh2IGwIVANLv
cG8nGauVDJbCGDykJiONS3eHAoGAEAj/0lDlcr1t2d3lrUOvYpTGld80dYmDepZ6
itsxTRzo/AN28x9B0DS787qw+1t93zogKkPNZHWCpg9kCvGRPP94g5fwZy6lu+ns
N0CP+WlLQXMuKGICY9oYZYUvnek4+VUFWb8tV0UwfPtrbfcnkawEeTBjkyRCuClM
I4rvxaACgYA8TxJQx6CYvhbNxzym6Uyl7R7aBlkZSAWF+dharak8H8tw5/UHg3m3
REWKF/WHC+qTvrmRlZe4Ydul/v6rHXPTe54mAGhIgeukC+Nm5f9ntokLzvZCvp0t
U14jJhC5+0eDi9NS2rlY9L7ta1CKFWOTgw9awj2hUTTDCFGUE972iwIVAKsBc5dp
+yt3TwlWSgfmc8ma2WeD
-----END DSA PRIVATE KEY-----
vagrant@vagrant 4 > cat gomezDSAkey.pem
-----BEGIN DSA PRIVATE KEY-----
MIIBugIBAAKBgQDi+OUmLr3NAfck0wQA6bPC1QQe1rlfx9+5+oXTc4MmawrAGQDW
Hav3A538r2kZjBrruB9mLDgegV1qmCIsjhzbtkNdgxuyyYAwsRm9kstvnq/Og6j2
47IGpe41McGjgZk3oA3mHQS76Ske8FzLLBJJX2AyBq+pPYw3mUGkUh2IGwIVANLv
cG8nGauVDJbCGDykJiONS3eHAoGAEAj/0lDlcr1t2d3lrUOvYpTGld80dYmDepZ6
itsxTRzo/AN28x9B0DS787qw+1t93zogKkPNZHWCpg9kCvGRPP94g5fwZy6lu+ns
N0CP+WlLQXMuKGICY9oYZYUvnek4+VUFWb8tV0UwfPtrbfcnkawEeTBjkyRCuClM
I4rvxaACgYB4opZbFmSzjwUOkORVV2bg7bDpHcTO9wywuGlf1nrJE3l3T/ooC9i1
ZG7E2RMaw61DDE6t2rMcdIbZzcbgyVPqESOu4xG2GujmJ/OmK8kt5Zk40U5rjipR
sUnxGUwJVTUIDCOC+B/U97IXoX8/+KRrF9zG1z/msF4eL1+g6n8jpwIUdxfYOuuG
DsIKbo4ufuMumHxCOS4=
-----END DSA PRIVATE KEY-----
```

5

El archivo *message* que he creado se trata de un fichero de texto plano de 128 bytes con ceros en su interior.

```
vagrant@vagrant 5 > cat message
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
vagrant@vagrant 5 >
```

6

> Debido a un bug de la versión 1.1.0g de OpenSSL que daba un error al firmar ficheros desde este ejercicio en adelante utilizo una máquina virtual con LibreSSL v2.1.6.

Para firmar el mensaje uso la siguiente orden:

```
openssl pkeyutl -sign -inkey angelDSApriv.pem -in message -out message.sign
```

- *-sign*: Firmar el mensaje.
- *-inkey*: Clave privada usada para firmar.
- *-in*: Archivo de entrada.

- *-out*: Archivo de salida.

```
vagrant@vagrant 6 > openssl pkeyutl -sign -inkey angelDSApriv.pem -in message -out
 message.sign
Enter pass phrase for angelDSApriv.pem:
vagrant@vagrant 6 > xxd message.sign
00000000: 302d 0214 1069 68b7 5036 1771 3254 9466  0-...ih.P6.q2T.f
00000010: 6909 b520 7ff6 ba79 0215 00b8 721e 40ef  i.. ...y....r.@.
00000020: addf 4add 646b 819f 36a9 8c81 1b2e 43    ..J.dk..6.....C
```

7

El archivo que he usado se trata del mismo que el generado en el ejercicio 5, pero en lugar de contener todo ceros contiene cinco unos al final.

```
vagrant@vagrant 7 > cat message
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000011111
```

Y ahora compruebo que la verificación sea correcta:

```
openssl pkeyutl -verify -pubin -inkey angelDSApub.pem -in message -sigfile message.sign
```

- *-verify*: Verificar el archivo de entrada con la firma.
- *-pubin*: Clave pública de entrada.
- *-inkey*: Archivo con la clave pública.
- *-in*: Archivo de entrada.
- *-sigfile*: Archivo con la firma.

```
vagrant@vagrant 7 > openssl pkeyutl -verify -pubin -inkey angelDSApub.pem -in mess
age -sigfile message.sign
Signature Verified Successfully
```

La verificación es correcta porque aunque se añadan mas bytes al final del archivo la firma está realizada para los mismos primeros bytes. Al verificar se comprueban los bytes que se firmaron al crear la firma, al ser los mismos la verificación es correcta.

8

Genero el hash de la clave pública usando sha384 de la siguiente manera:

```
openssl dgst -sha384 -hex -c -out angelDSApub.sha384 angelDSApub.pem
```

- *-sha384*: Algoritmo usado para calcular hash.
- *-hex*: Salida en hexadecimal.
- *-c*: Salida por pares de dígitos separados por dos puntos.

- *-out*: Archivo de salida.
- *angelDSApub.pem*: Clave pública de entrada.

```
vagrant@vagrant 8 > openssl dgst -sha384 -hex -c -out angelDSApub.sha384 angelDSAp
ub.pem
vagrant@vagrant 8 > cat angelDSApub.sha384
SHA384(angelDSApub.pem)= 0c:a3:59:cc:7e:4a:3e:0a:9c:c3:47:04:db:c2:c2:c0:a4:02:a5:
f4:5e:a5:1b:e1:4e:a3:de:d6:8a:08:aa:1c:e7:15:36:ac:68:b4:5d:0f:1e:30:e9:30:48:8b:9
3:7f
```

9

> Aunque en el ejercicio 7 el archivo modificado se llama *message*, lo renombro como *message2* para realizar el resto de ejercicios y así utilizar la nomenclatura del guión de prácticas.

Calculo el valor hash de *message2* de una forma muy similar al ejercicio anterior:

```
openssl dgst -sha1 -binary -out message2.sha1 message2
```

- *-sha1*: Algoritmo usado para calcular hash. He elegido *sha1* porque devuelve una salida de 160 bits, aunque también podría haber usado *sha0*, que también devuelve una salida de 160 bits.
- *-binary*: Devuelve la salida en formato binario.
- *-out*: Archivo de salida.
- *message2*: Archivo de entrada.

```
vagrant@vagrant 9 > openssl dgst -sha1 -binary -out message2.sha1 message2
vagrant@vagrant 9 > xxd message2
message2       message2.sha1
vagrant@vagrant 9 > xxd message2.sha1
00000000: ee71 a039 e4c9 9ec7 56c9 8b8a 70ee a5a3  .q.9....V...p...
00000010: 5845 d24f                                XE.O
```

10

Firmo el archivo *message2* con la firma anterior (*message2.sha1*) con la siguiente orden:

```
openssl dgst -sign angelDSApriv.pem -out message2.sign message2.sha1 message2
```

- *-sign*: Firmar usando la clave privada.
- *-out*: Archivo de salida.
- *message2.sha1*: Entrada del hash.
- *message2*: Entrada del archivo a firmar.

```
vagrant@vagrant 10 > openssl dgst -sign angelDSApriv.pem -out message2.sign messag
e2.sha1 message2
Enter pass phrase for angelDSApriv.pem:
vagrant@vagrant 10 > xxd message2.sign
00000000: 302c 0214 3d0e adfb fd58 25ec 75a1 9eb2  0,..=....X%.u...
00000010: 9782 1881 3931 9957 0214 5b08 59cb e3c1  ....91.W..[.Y...
00000020: 1ff8 25ce 9c56 aa79 69b3 0e88 cf27 302c  ..%..V.yi....'0,
00000030: 0214 0bc1 c52f 696f 06ff 8a83 4f00 5040  ...../io....O.P@
00000040: b943 1f6e ae55 0214 7c27 6077 ba67 f3c2  .C.n.U..|'`w.g..
00000050: 654e c9f0 cafa ee3e 8581 9038            eN.....>...8
```

11

La orden para verificar es la siguiente:

```
openssl dgst -verify angelDSApub.pem -signature message2.sign message
openssl dgst -verify angelDSApub.pem -signature message2.sign message2
```

Obtengo los siguientes errores:

```
vagrant@vagrant 11 > openssl dgst -verify angelDSApub.pem -signature message2.sign
 message
Error Verifying Data
vagrant@vagrant 11 >
vagrant@vagrant 11 > openssl dgst -verify angelDSApub.pem -signature message2.sign
 message2
Error Verifying Data
vagrant@vagrant 11 >
```

En el primer caso se produce (creo) porque estoy intentando verificar el mensaje con una firma generada a partir de otro archivo, que aunque comparten la mayoria de bits, no son exactamente iguales.

Por otro lado no entiendo el segundo error, pues al tratarse de una firma obtenida a partir de *message2* debería verificarse correctamente.

12

```
openssl pkeyutl -verify -pubin -inkey angelDSApub.pem -in message2 -sigfile message2.sign
```

```
vagrant@vagrant 12 > openssl pkeyutl -verify -pubin -inkey angelDSApub.pem -in mes
sage2 -sigfile message2.sign
Public Key operation error
```

En este caso el error se produce (creo) porque estoy intentando verificar con una firma obtenida a partir del hash *sha1* del archivo, por lo que no usa en absoluto la clave pública o privada de *angel*.

13

El comando que he usado para generar el valor HMAC de *sharedDSA.pem* es el siguiente:

```
openssl dgst -hmac 12345 sharedDSA.pem
```

- *-hmac 12345*: Crear hashed MAC usando como clave *12345*.
- *sharedDSA.pem*: Archivo de entrada.

```
vagrant@vagrant 13 > openssl dgst -hmac 12345 sharedDSA.pem
HMAC-SHA256(sharedDSA.pem)= 5e7202c7b185c7839be9a037c1121bdcb102056e1cb5f19136fa8b9793f64da5
vagrant@vagrant 13 >
```

14

En primer lugar genero el par de claves pública y privada asociadas a una curva elíptica para ambos usuarios (pues ahora uso una máquina virtual y he obtenido fallos usando las claves generadas anteriormente). Lo hago del mismo modo que en la práctica anterior.

Elijo la curva B-163, que corresponde con *sect163k1* en OpenSSL.

```
openssl ecparam -name sect163k1 -out stdECparam.pem
```

```
vagrant@vagrant 14 > cat stdECparam.pem
-----BEGIN EC PARAMETERS-----
BgUrgQQAAQ==
-----END EC PARAMETERS-----
```

Genero la clave a partir de la curva...

```
openssl ecparam -in stdECparam.pem -out angelECkey.pem -genkey -noout
openssl ecparam -in stdECparam.pem -out gomezECkey.pem -genkey -noout
```

```
vagrant@vagrant 14 > cat angelECkey.pem
-----BEGIN EC PRIVATE KEY-----
MFMCAQEEFQPPHzwCZ+faM4GHpEsFFPdu8KCfUKAHBgUrgQQAAaEuAywABAUWfukj
Q9E0QBYhL2OBcbZ1bHIHYwKvxjlf2lt7kcfwv768zEiqZ/VGTw==
-----END EC PRIVATE KEY-----
vagrant@vagrant 14 > cat gomezECkey.pem
-----BEGIN EC PRIVATE KEY-----
MFMCAQEEFQPZNg3Pb9FpRPFrNHLfjNEZr3/XP6AHBgUrgQQAAaEuAywABAPNENhu
cl/I1O48WBxpp6LUFMud5QQhUh9EybcFAI7tj1sEAH8wkTHElg==
-----END EC PRIVATE KEY-----
```

... y extraigo las claves privadas y públicas:

```
# Claves privadas:
openssl ec -in angelECkey.pem -out angelECpriv.pem -aes128
openssl ec -in gomezECkey.pem -out gomezECpriv.pem -aes128

# Claves públicas:
openssl ec -in angelECkey.pem -out angelECpub.pem -pubout
openssl ec -in gomezECkey.pem -out gomezECpub.pem -pubout
```

> Pass usado: 0123456789

```
vagrant@vagrant 14 > openssl ec -in angelECkey.pem -out angelECpriv.pem -aes128
read EC key
writing EC key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
vagrant@vagrant 14 > openssl ec -in gomezECkey.pem -out gomezECpriv.pem -aes128
read EC key
writing EC key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
vagrant@vagrant 14 > openssl ec -in angelECkey.pem -out angelECpub.pem -pubout
read EC key
writing EC key
vagrant@vagrant 14 > openssl ec -in gomezECkey.pem -out gomezECpub.pem -pubout
read EC key
writing EC key
vagrant@vagrant 14 > cat angelECpriv.pem
-----BEGIN EC PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,D2D85A01360EB949A4BF2DE2642015B9

SrgJtmwrnhYbH+LAz168hafi/M47QFGUU7ClQsPBfZCHpYjAonrBANKKzaRJ9fQt
S5hXEOunNgX34EbCLll68/19u3Ihm5unFDCaPzKS6mcO837RoIg5Fsog7pekkPre
-----END EC PRIVATE KEY-----
vagrant@vagrant 14 > cat gomezECpriv.pem
-----BEGIN EC PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,4D7F620AF5B5ACE4AB1A3415EA57D831

eWedP+XBN0p1Z62W0E0HIYBNwt0wl+wrW3KseLGE1d5RN7RIj0hCdyqQ4gjSoT4U
CHFOREscC6khqxdmYv32Hz7a2PxprGUywA7ayaJlqh0pyU5dm6OncivEQJEmrPQf
-----END EC PRIVATE KEY-----
vagrant@vagrant 14 > cat angelECpub.pem
-----BEGIN PUBLIC KEY-----
MEAwEAYHKoZIzj0CAQYFK4EEAAEDLAAEBRZ+6SND0TRAFiEvY4FxtnVscgdjAq/G
OV/aW3uRx/C/vrzMSKpn9UZP
-----END PUBLIC KEY-----
vagrant@vagrant 14 > cat gomezECpub.pem
-----BEGIN PUBLIC KEY-----
MEAwEAYHKoZIzj0CAQYFK4EEAAEDLAAEA80Q2G5yX8jU7jxYHGmnotQUy53lBCFS
H0TJtwUAju2PWwQAfzCRMcSW
-----END PUBLIC KEY-----
```

Ahora cada usuario genera su clave generada:

```
openssl pkeyutl -inkey angelECpriv.pem -peerkey gomezECpub.pem -derive -out angelKEY.bin
openssl pkeyutl -inkey gomezECpriv.pem -peerkey angelECpub.pem -derive -out gomezKEY.bin
```

```
vagrant@vagrant 14 > openssl pkeyutl -derive -inkey angelECpriv.pem -peerkey gomezECpub.pem -
out angelKEY.bin
Enter pass phrase for angelECpriv.pem:
vagrant@vagrant 14 > openssl pkeyutl -derive -inkey gomezECpriv.pem -peerkey angelECpub.pem -
out gomezKEY.bin
Enter pass phrase for gomezECpriv.pem:
vagrant@vagrant 14 >
vagrant@vagrant 14 > cat angelKEY.bin
[illegible]vagrant@vagrant 14 >
vagrant@vagrant 14 >
vagrant@vagrant 14 > cat gomezKEY.bin
[illegible]vagrant@vagrant 14 >
vagrant@vagrant 14 >
vagrant@vagrant 14 > diff angelKEY.bin gomezKEY.bin
vagrant@vagrant 14 >
vagrant@vagrant 14 > cp angelKEY.bin key.bin
```

```
vagrant@vagrant 14 > xxd angelKEY.bin
00000000: 0330 af77 5793 c220 dc86 8167 bcf0 ff1a  .0.wW.. ...g....
00000010: 75bf 10c2 b6                             u....
vagrant@vagrant 14 >
vagrant@vagrant 14 > xxd gomezKEY.bin
00000000: 0330 af77 5793 c220 dc86 8167 bcf0 ff1a  .0.wW.. ...g....
00000010: 75bf 10c2 b6                             u....
```

Se observa que ambas KEYs son iguales, por lo que renombro una de ellas a *key.bin* para usar la nomenclatura del guión.

Procedimiento:

Primero, *angel* concatena la clave pública de *gomez* con la suya:

```
cat gomezECpub.pem angelECpub.pem > gomezangel.pub
```

```
vagrant@vagrant 14 > cat gomezECpub.pem angelECpub.pem > gomezangel.pub
vagrant@vagrant 14 > cat gomezangel.pub
-----BEGIN PUBLIC KEY-----
MEAwEAYHKoZIzj0CAQYFK4EEAAEDLAAEA80Q2G5yX8jU7jxYHGmnotQUy53lBCFS
H0TJtwUAju2PWwQAfzCRMcSW
-----END PUBLIC KEY-----
-----BEGIN PUBLIC KEY-----
MEAwEAYHKoZIzj0CAQYFK4EEAAEDLAAEBRZ+6SND0TRAFiEvY4FxtnVscgdjAq/G
OV/aW3uRx/C/vrzMSKpn9UZP
-----END PUBLIC KEY-----
```

Ahora *angel* firma con su clave privada DSA el archivo anterior.

```
openssl dgst -out angel.sign -sign angelDSApriv.pem gomezangel.key
```

```
vagrant@vagrant 14 > openssl dgst -out angel.sign -sign angelDSApriv.pem gomezangel.pub
Enter pass phrase for angelDSApriv.pem:
vagrant@vagrant 14 > cat angel.sign
0,[illegible]
[illegible]W!uvagrant@vagrant 14 >
vagrant@vagrant 14 >
```

```
vagrant@vagrant 14 > xxd angel.sign
00000000: 302c 0214 0a98 9898 ef7c 6947 7b1c fe12  0,.......|iG{...
00000010: e7b6 4ec4 939c d047 0214 1560 1d9e d305  ..N....G...`....
00000020: ab1f 1079 a78a 1e7b 04f4 1d57 2175       ...y...{...W!u
```

Tras esto *angel* encripta el archivo firmado con la clave derivada y se la envía a *gomez*:

```
openssl enc -aes-128-cfb8 -out angelSIGN.crypt -in angel.sign -kfile key.bin
```

```
vagrant@vagrant 14 > openssl enc -aes-128-cfb8 -out angelSIGN.crypt -in angel.sign -kfile key
.bin
vagrant@vagrant 14 > xxd angelSIGN.crypt
00000000: 5361 6c74 6564 5f5f e694 a652 fa9a d191  Salted__...R....
00000010: 2c82 6254 38d0 16de d4e3 4b47 92bb 52cc  ,.bT8.....KG..R.
00000020: 4f6a a31c b939 868c f4d3 1f44 5853 0778  Oj...9.....DXS.x
00000030: 16e6 1ab1 bda3 93f3 09f3 e585 3697       ............6.
```

gomez desencripta el archivo...

```
openssl aes-128-cfb8 -d -in angelSIGN.crypt -out angelSIGN.decrypt -kfile key.bin
```

```
vagrant@vagrant 14 > openssl aes-128-cfb8 -d -in angelSIGN.crypt -out angelSIGN.decrypt -kfile
 key.bin
vagrant@vagrant 14 >
vagrant@vagrant 14 > xxd angelSIGN.decrypt
00000000: 302c 0214 0a98 9898 ef7c 6947 7b1c fe12  0,.......|iG{...
00000010: e7b6 4ec4 939c d047 0214 1560 1d9e d305  ..N....G...`....
00000020: ab1f 1079 a78a 1e7b 04f4 1d57 2175       ...y...{...W!u
```

... y verifica:

```
openssl dgst -verify angelDSApub.pem -signature angelSIGN.decrypt gomezangel.pub
```

```
vagrant@vagrant 14 > openssl dgst -verify angelDSApub.pem -signature angelSIGN.decrypt gomezan
gel.pub
Verified OK
```

Por otro lado, el usuario gomez realiza el mismo proceso. En primer lugar concatena ambas claves (EC) públicas.

```
cat angelECpub.pem gomezECpub.pem > angelgomez.pub
```

```
vagrant@vagrant 14 > cat angelECpub.pem gomezECpub.pem > angelgomez.pub
vagrant@vagrant 14 > cat angelgomez.pub
-----BEGIN PUBLIC KEY-----
MEAwEAYHKoZIzj0CAQYFK4EEAAEDLAAEBRZ+6SND0TRAFiEvY4FxtnVscgdjAq/G
OV/aW3uRx/C/vrzMSKpn9UZP
-----END PUBLIC KEY-----
-----BEGIN PUBLIC KEY-----
MEAwEAYHKoZIzj0CAQYFK4EEAAEDLAAEA80Q2G5yX8jU7jxYHGmnotQUy53lBCFS
H0TJtwUAju2PWwQAfzCRMcSW
-----END PUBLIC KEY-----
```

Ahora firma el archivo anterior:

```
openssl dgst -out gomez.sign -sign gomezDSApriv.pem angelgomez.pub
```

```
vagrant@vagrant 14 > openssl dgst -out gomez.sign -sign gomezDSApriv.pem angelgomez.pub
Enter pass phrase for gomezDSApriv.pem:
vagrant@vagrant 14 >
vagrant@vagrant 14 > xxd gomez.sign
00000000: 302d 0215 00a1 c824 c7b7 222c 0f9f e9fa  0-.....$..",....
00000010: db9d 131a 68dd b60d c702 1458 4655 01c3  ....h......XFU..
00000020: 401f adb3 1897 68e2 dfd1 12da f409 14    @.....h........
vagrant@vagrant 14 >
```

Tras firmarlo, lo cifra y se lo envía a *angel*:

```
openssl enc -aes-128-cfb8 -out gomezSIGN.crypt -in gomez.sign -kfile key.bin
```

```
vagrant@vagrant 14 > openssl enc -aes-128-cfb8 -out gomezSIGN.crypt -in gomez.sign -kfile key.
bin
vagrant@vagrant 14 > xxd gomezSIGN.crypt
00000000: 5361 6c74 6564 5f5f 1c31 3f66 f66d e9af  Salted__.1?f.m..
00000010: c02f ee69 2de7 9c65 fb24 790f 02c8 e67d  ./.i-..e.$y....}
00000020: 7e51 06d8 8910 3174 b72b 4f57 c442 d13f  ~Q....1t.+OW.B.?
00000030: 83a6 84a5 24ce ee8a 93a5 c004 088a b4    ....$..........
vagrant@vagrant 14 >
```

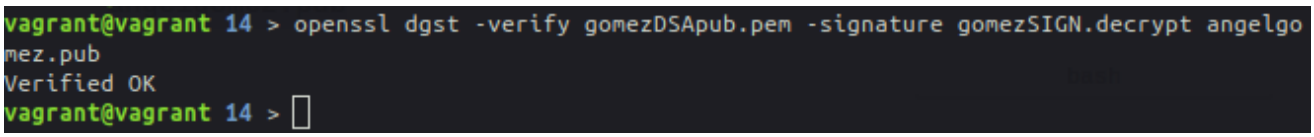
angel lo desencripta:

```
openssl aes-128-cfb8 -d -in gomezSIGN.crypt -out gomezSIGN.decrypt -kfile key.bin
```

```
vagrant@vagrant 14 > openssl aes-128-cfb8 -d -in gomezSIGN.crypt -out gomezSIGN.decrypt -kfile
 key.bin
vagrant@vagrant 14 > xxd gomezSIGN.decrypt
00000000: 302d 0215 00a1 c824 c7b7 222c 0f9f e9fa  0-.....$..",....
00000010: db9d 131a 68dd b60d c702 1458 4655 01c3  ....h......XFU..
00000020: 401f adb3 1897 68e2 dfd1 12da f409 14    @.....h........
```

Y finalmente lo verifica:

```
openssl dgst -verify gomezDSApub.pem -signature gomezSIGN.decrypt angelgomez.pub
```



Habiendo obtenido ambos usuarios una verificación correcta, ya saben que la clave derivada (*K*) sólo está compartida entre ellos dos.