

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

---

SPSI  
Práctica 2: Protocolos criptográficos.

---

Carlos Manuel Sequí Sánchez

26 de noviembre de 2017

## Índice

1	EJERCICIO 1. Generad un archivo sharedDSA.pem que contenga los parámetros. Mostrad los valores.	4
2	EJERCICIO 2. Generad dos parejas de claves para los parámetros anteriores. La claves se almacenarán en los archivos nombreDSAkey.pem y apellidoDSAkey.pem. No es necesario protegerlas por contraseña.	5
3	EJERCICIO 3. Extraed la clave privada contenida en el archivo nombreDSAkey.pem a otro archivo que tenga por nombre nombreDSApriv.pem. Este archivo deberá estar protegido por contraseña. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem.	5
4	EJERCICIO 4. Extraed en nombreDSAPub.pem la clave pública contenida en el archivo nombreDSAkey.pem. De nuevo nombreDSAPub.pem no debe estar cifrado ni protegido. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem.	7
5	EJERCICIO 5. Calculad el valor hash del archivo con la clave pública nombreDSAPub.pem usando sha384 con salida hexadecimal con bloques de dos caracteres separados por dos puntos. Mostrad los valores por salida estándar y guardadlo en nombreDSAPub.sha384.	9
6	EJERCICIO 6. Calculad el valor hash del archivo con la clave pública apellidoDSAPub.pem usando una función hash de 160 bits con salida binaria. Guardad el hash en apellidoDSAPub.[algoritmo] y mostrad su contenido.	10
7	EJERCICIO 7. Generad el valor HMAC del archivo sharedDSA.pem con clave '12345' mostrándolo por pantalla.	10
8	EJERCICIO 8. Simulad una ejecución completa del protocolo Estación a Estación. Para ello emplearemos como claves para firma/verificación las generadas en esta práctica, y para el protocolo DH emplearemos las claves asociadas a curvas elípticas de la práctica anterior junto con las de otro usuario simulado que deberéis generar nuevamente. Por ejemplo, si mi clave privada está en javierECpriv.pem y la clave pública del otro usuario está en lobilloECpub.pem, el comando para generar la clave derivada será: \$>openssl pkeyutl -inkey javierECpriv.pem -peerkey lobilloECpub.pem -derive -out key.bin El algoritmo simétrico a utilizar en el protocolo	11

## Índice de figuras

1.1.	sharedDSA.	4
3.1.	CarlosDSApriv.	6

3.2.	SequiDSApriv. . . . .	7
4.1.	CarlosDSAPub. . . . .	8
4.2.	SequiDSAPub. . . . .	9
5.1.	CarlosDSAPub.sha384. . . . .	9
6.1.	SequiDSAPub.ripemd160. . . . .	10
7.1.	Valor HMAC de sharedDSA.pem. . . . .	10
8.1.	Muestra de la correcta verificación de ambos archivos . . . . .	14

## 1. EJERCICIO 1. Generad un archivo sharedDSA.pem que contenga los parámetros. Mostrad los valores.

-Le especificamos el archivo de salida "sharedDSA.pem" el numbits del archivo, ya que no le indicamos ningún archivo de entrada:

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

-Mostramos los valores en modo texto(hexadecimal), ya que en modo privado no se ve nada. para ello hacemos uso del parámetro -text metiéndole como entrada el archivo de parámetros generado previamente y con la opción -noout para evitar que salga por pantalla el archivo sharedDSA.pem.

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

```
sequi96@eil40090:~/Escritorio$ openssl dsaparam -in sharedDSA.pem -text -noout
P:
00:9e:c5:ff:a8:77:de:17:0b:8e:63:c8:34:e0:e9:
08:a7:ef:2e:19:5f:57:bd:8e:62:ee:66:84:6e:ae:
9f:bd:02:2c:c3:a7:9c:d3:d0:a2:4f:c4:42:4b:28:
15:ff:de:b0:82:31:5e:5d:af:0c:02:bd:30:0a:e0:
33:58:43:e8:86:fb:64:95:1e:d6:e7:38:bc:86:b0:
0e:b8:e9:38:ce:57:3b:18:05:b2:b8:3a:55:61:d4:
ef:32:3f:3d:4c:76:08:69:18:4b:08:39:7c:6f:5d:
50:5d:5e:c6:3c:ca:11:37:ee:eb:c7:75:d4:0c:b7:
67:27:ab:22:51:f1:57:ae:d5
Q:
00:d7:e5:b3:49:17:f5:ab:a5:e7:68:59:57:9c:0b:
1f:10:d7:df:8e:49
G:
3a:88:81:b7:80:8e:24:06:95:f3:42:14:5c:53:35:
ec:76:8b:d8:7d:7f:93:d6:01:31:42:af:99:54:b2:
b6:28:3b:55:2e:d1:e3:98:80:b3:16:10:34:9a:64:
96:8f:8a:6c:3d:45:df:95:15:d5:cd:9b:41:7b:a4:
7f:7f:f8:26:1d:65:c5:55:44:38:31:ed:10:e6:c6:
07:68:34:08:81:e5:53:0f:02:24:68:60:1d:fe:ff:
1c:73:e4:b6:a3:c7:f7:28:9c:6c:10:d5:bc:a8:f6:
03:70:25:7f:69:34:d8:a4:f6:34:93:3b:5e:3c:f3:
93:40:29:6b:3b:35:9d:1d
```

Figura 1.1: sharedDSA.

**2. EJERCICIO 2. Generad dos parejas de claves para los parámetros anteriores. Las claves se almacenarán en los archivos nombreDSAkey.pem y apellidoDSAkey.pem. No es necesario protegerlas por contraseña.**

-Generamos las dos parejas de claves a partir de los parámetros creados anteriormente:

```
openssl genssa sharedDSA.pem -out CarlosDSAkey.pem  
openssl genssa sharedDSA.pem -out SequiDSAkey.pem
```

**3. EJERCICIO 3. Extraed la clave privada contenida en el archivo nombreDSAkey.pem a otro archivo que tenga por nombre nombreDSApriv.pem. Este archivo deberá estar protegido por contraseña. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem.**

-Extraemos las claves privadas con el comando `openssl dsa` utilizando los pares de claves generados en el ejercicio anterior y, además, los ciframos con `aes128` introduciéndoles como contraseña: 0123456789.

```
openssl dsa -in CarlosDSAkey.pem -aes128 -out CarlosDSApriv.pem
```

-Mostramos la clave:

```
openssl dsa -in CarlosDSApriv.pem -text -noout
```

```

sequi96@eil140090:~/Escritorio$ openssl dsa -in CarlosDSApriv.pem -text -noout
read DSA key
Enter pass phrase for CarlosDSApriv.pem:
Private-Key: (1024 bit)
priv:
    5a:ee:ee:1f:23:6b:2f:e2:80:76:c4:b9:85:9c:c1:
    ae:a9:52:9b:c6
pub:
    6d:cd:13:71:c4:c4:bb:9c:fe:48:53:2f:b0:42:
    cf:7a:ae:d6:e9:c6:d0:3e:16:da:84:5b:00:71:1d:
    b8:d9:57:70:01:fa:ac:04:4d:4d:b2:d8:98:b3:4d:
    66:30:2c:89:6a:4b:47:70:9c:4d:df:15:90:9d:2f:
    11:39:dd:60:8c:b9:db:20:ab:83:43:69:fa:87:c6:
    16:11:97:b1:84:e3:a9:bd:5b:aa:a3:8e:f3:ff:7c:
    0b:4f:d5:6f:c5:71:31:0a:90:b2:10:56:ee:df:e7:
    16:ae:f2:2a:b5:6c:74:9f:02:d3:60:bc:e2:12:73:
    de:3c:6e:ec:d1:f3:a4:2b
P:
    00:9e:c5:ff:a8:77:de:17:0b:8e:63:c8:34:e0:e9:
    08:a7:ef:2e:19:5f:57:bd:8e:62:ee:66:84:6e:ae:
    9f:bd:02:2c:c3:a7:9c:d3:d0:a2:4f:c4:42:4b:28:
    15:ff:de:b0:82:31:5e:5d:af:0c:02:bd:30:0a:e0:
    33:58:43:e8:86:fb:64:95:1e:d6:e7:38:bc:86:b0:
    0e:b8:e9:38:ce:57:3b:18:05:b2:b8:3a:55:61:d4:
    ef:32:3f:3d:4c:76:08:69:18:4b:08:39:7c:6f:5d:
    50:5d:5e:c6:3c:ca:11:37:ee:eb:c7:75:d4:0c:b7:
    67:27:ab:22:51:f1:57:ae:d5
Q:
    00:d7:e5:b3:49:17:f5:ab:a5:e7:68:59:57:9c:0b:
    1f:10:d7:df:8e:49
G:
    3a:88:81:b7:80:8e:24:06:95:f3:42:14:5c:53:35:
    ec:76:8b:d8:7d:7f:93:d6:01:31:42:af:99:54:b2:
    b6:28:3b:55:2e:d1:e3:98:80:b3:16:10:34:9a:64:
    96:8f:8a:6c:3d:45:df:95:15:d5:cd:9b:41:7b:a4:
    7f:7f:f8:26:1d:65:c5:55:44:38:31:ed:10:e6:c6:
    07:68:34:08:81:e5:53:0f:02:24:68:60:1d:fe:ff:
    1c:73:e4:b6:a3:c7:f7:28:9c:6c:10:d5:bc:a8:f6:
    03:70:25:7f:69:34:d8:a4:f6:34:93:3b:5e:3c:f3:
    93:40:29:6b:3b:35:9d:1d

```

Figura 3.1: CarlosDSApriv.

openssl dsa -in SequiDSAkey.pem -aes128 -out SequiDSApriv.pem

-Mostramos la clave:

openssl dsa -in SequiDSApriv.pem -text -noout

```

sequi96@eil40090:~/Escritorio$ openssl dsa -in SequiDSApriv.pem -text -noout
read DSA key
Enter pass phrase for SequiDSApriv.pem:
Private-Key: (1024 bit)
priv:
  00:bb:7f:c9:f2:38:b9:42:51:b9:00:e9:55:83:fc:
  97:f6:11:e6:f2:cf
pub:
  67:d9:22:e3:6e:88:5b:0c:43:34:8d:e8:bd:82:17:
  1f:89:ca:fa:eb:15:b2:9b:e6:34:4a:2e:02:f1:bc:
  5f:04:21:c1:cc:ab:d5:92:99:6f:b4:9d:87:51:b6:
  d4:54:ea:03:3e:81:8c:cb:e7:a4:9f:bc:e7:aa:08:
  fa:bd:88:f2:c7:9f:3b:4a:35:d5:d7:e2:27:f5:80:
  6d:ea:1e:34:12:38:60:ab:88:cb:6e:c4:7e:a4:61:
  dd:4f:73:d2:b1:94:b6:4e:8d:bd:dc:7a:05:8d:d9:
  6f:02:1f:1d:37:c6:7d:bb:f0:4b:26:a4:6b:87:6e:
  fa:4e:7e:68:2a:f9:e2:da
P:
  00:9e:c5:ff:a8:77:de:17:0b:8e:63:c8:34:e0:e9:
  08:a7:ef:2e:19:5f:57:bd:8e:62:ee:66:84:6e:ae:
  9f:bd:02:2c:c3:a7:9c:d3:d0:a2:4f:c4:42:4b:28:
  15:ff:de:b0:82:31:5e:5d:af:0c:02:bd:30:0a:e0:
  33:58:43:e8:86:fb:64:95:1e:d6:e7:38:bc:86:b0:
  0e:b8:e9:38:ce:57:3b:18:05:b2:b8:3a:55:61:d4:
  ef:32:3f:3d:4c:76:08:69:18:4b:08:39:7c:6f:5d:
  50:5d:5e:c6:3c:ca:11:37:ee:eb:c7:75:d4:0c:b7:
  67:27:ab:22:51:f1:57:ae:d5
Q:
  00:d7:e5:b3:49:17:f5:ab:a5:e7:68:59:57:9c:0b:
  1f:10:d7:df:8e:49
G:
  3a:88:81:b7:80:8e:24:06:95:f3:42:14:5c:53:35:
  ec:76:8b:d8:7d:7f:93:d6:01:31:42:af:99:54:b2:
  b6:28:3b:55:2e:d1:e3:98:80:b3:16:10:34:9a:64:
  96:8f:8a:6c:3d:45:df:95:15:d5:cd:9b:41:7b:a4:
  7f:7f:f8:26:1d:65:c5:55:44:38:31:ed:10:e6:c6:
  07:68:34:08:81:e5:53:0f:02:24:68:60:1d:fe:ff:
  1c:73:e4:b6:a3:c7:f7:28:9c:6c:10:d5:bc:a8:f6:
  03:70:25:7f:69:34:d8:a4:f6:34:93:3b:5e:3c:f3:
  93:40:29:6b:3b:35:9d:1d

```

Figura 3.2: SequiDSApriv.

4. **EJERCICIO 4.** Extraed en nombreDSAPub.pem la clave pública contenida en el archivo nombreDSAkey.pem. De nuevo nombreDSAPub.pem no debe estar cifrado ni protegido. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem.

-Extraemos las claves públicas con el comando `openssl dsa` utilizando los pares de claves generados en el ejercicio anterior y, además, los ciframos con `aes128` introduciéndoles como contraseña: 0123456789.

openssl dsa -in CarlosDSAkey.pem -out CarlosDSAPub.pem -pubout

-Mostramos la clave:

openssl dsa -in CarlosDSAPub.pem -text -noout -pubin

```
sequi96@eil40090:~/Escritorio$ openssl dsa -in CarlosDSAPub.pem -text -noout -pubin
read DSA key
pub:
 6d:cd:13:71:c4:c4:c4:bb:9c:fe:48:53:2f:b0:42:
 cf:7a:ae:d6:e9:c6:d0:3e:16:da:84:5b:00:71:1d:
 b8:d9:57:70:01:fa:ac:04:4d:4d:b2:d8:98:b3:4d:
 66:30:2c:89:6a:4b:47:70:9c:4d:df:15:90:9d:2f:
 11:39:dd:60:8c:b9:db:20:ab:83:43:69:fa:87:c6:
 16:11:97:b1:84:e3:a9:bd:5b:aa:a3:8e:f3:ff:7c:
 0b:4f:d5:6f:c5:71:31:0a:90:b2:10:56:ee:df:e7:
 16:ae:f2:2a:b5:6c:74:9f:02:d3:60:bc:e2:12:73:
 de:3c:6e:ec:d1:f3:a4:2b
P:
 00:9e:c5:ff:a8:77:de:17:0b:8e:63:c8:34:e0:e9:
 08:a7:ef:2e:19:5f:57:bd:8e:62:ee:66:84:6e:ae:
 9f:bd:02:2c:c3:a7:9c:d3:d0:a2:4f:c4:42:4b:28:
 15:ff:de:b0:82:31:5e:5d:af:0c:02:bd:30:0a:e0:
 33:58:43:e8:86:fb:64:95:1e:d6:e7:38:bc:86:b0:
 0e:b8:e9:38:ce:57:3b:18:05:b2:b8:3a:55:61:d4:
 ef:32:3f:3d:4c:76:08:69:18:4b:08:39:7c:6f:5d:
 50:5d:5e:c6:3c:ca:11:37:ee:eb:c7:75:d4:0c:b7:
 67:27:ab:22:51:f1:57:ae:d5
Q:
 00:d7:e5:b3:49:17:f5:ab:a5:e7:68:59:57:9c:0b:
 1f:10:d7:df:8e:49
G:
 3a:88:81:b7:80:8e:24:06:95:f3:42:14:5c:53:35:
 ec:76:8b:d8:7d:7f:93:d6:01:31:42:af:99:54:b2:
 b6:28:3b:55:2e:d1:e3:98:80:b3:16:10:34:9a:64:
 96:8f:8a:6c:3d:45:df:95:15:d5:cd:9b:41:7b:a4:
 7f:7f:f8:26:1d:65:c5:55:44:38:31:ed:10:e6:c6:
 07:68:34:08:81:e5:53:0f:02:24:68:60:1d:fe:ff:
 1c:73:e4:b6:a3:c7:f7:28:9c:6c:10:d5:bc:a8:f6:
 03:70:25:7f:69:34:d8:a4:f6:34:93:3b:5e:3c:f3:
 93:40:29:6b:3b:35:9d:1d
```

Figura 4.1: CarlosDSAPub.

openssl dsa -in SequiDSAkey.pem -out SequiDSAPub.pem -pubout

-Mostramos la clave:

openssl dsa -in SequiDSAPub.pem -text -noout -pubin



```

sequi96@ei140090:~/Escritorio$ openssl dsa -in SequiDSAPub.pem -text -noout -pubin
read DSA key
pub:
 67:d9:22:e3:6e:88:5b:0c:43:34:8d:e8:bd:82:17:
 1f:89:ca:fa:eb:15:b2:9b:e6:34:4a:2e:02:f1:bc:
 5f:04:21:c1:cc:ab:d5:92:99:6f:b4:9d:87:51:b6:
 d4:54:ea:03:3e:81:8c:cb:e7:a4:9f:bc:e7:aa:08:
 fa:bd:88:f2:c7:9f:3b:4a:35:d5:d7:e2:27:f5:80:
 6d:ea:1e:34:12:38:60:ab:88:cb:6e:c4:7e:a4:61:
 dd:4f:73:d2:b1:94:b6:4e:8d:bd:dc:7a:05:8d:d9:
 6f:02:1f:1d:37:c6:7d:bb:f0:4b:26:a4:6b:87:6e:
 fa:4e:7e:68:2a:f9:e2:da
P:
 00:9e:c5:ff:a8:77:de:17:0b:8e:63:c8:34:e0:e9:
 08:a7:ef:2e:19:5f:57:bd:8e:62:ee:66:84:6e:ae:
 9f:bd:02:2c:c3:a7:9c:d3:d0:a2:4f:c4:42:4b:28:
 15:ff:de:b0:82:31:5e:5d:af:0c:02:bd:30:0a:e0:
 33:58:43:e8:86:fb:64:95:1e:d6:e7:38:bc:86:b0:
 0e:b8:e9:38:ce:57:3b:18:05:b2:b8:3a:55:61:d4:
 ef:32:3f:3d:4c:76:08:69:18:4b:08:39:7c:6f:5d:
 50:5d:5e:c6:3c:ca:11:37:ee:eb:c7:75:d4:0c:b7:
 67:27:ab:22:51:f1:57:ae:d5
Q:
 00:d7:e5:b3:49:17:f5:ab:a5:e7:68:59:57:9c:0b:
 1f:10:d7:df:8e:49
G:
 3a:88:81:b7:80:8e:24:06:95:f3:42:14:5c:53:35:
 ec:76:8b:d8:7d:7f:93:d6:01:31:42:af:99:54:b2:
 b6:28:3b:55:2e:d1:e3:98:80:b3:16:10:34:9a:64:
 96:8f:8a:6c:3d:45:df:95:15:d5:cd:9b:41:7b:a4:
 7f:7f:f8:26:1d:65:c5:55:44:38:31:ed:10:e6:c6:
 07:68:34:08:81:e5:53:0f:02:24:68:60:1d:fe:ff:
 1c:73:e4:b6:a3:c7:f7:28:9c:6c:10:d5:bc:a8:f6:
 03:70:25:7f:69:34:d8:a4:f6:34:93:3b:5e:3c:f3:
 93:40:29:6b:3b:35:9d:1d

```

Figura 4.2: SequiDSAPub.

## 5. EJERCICIO 5. Calculad el valor hash del archivo con la clave pública nombreDSAPub.pem usando sha384 con salida hexadecimal con bloques de dos caracteres separados por dos puntos. Mostrad los valores por salida estándar y guardadlo en nombreDSAPub.sha384.

-Calculamos el valor hash usando sha384 con la salida hexadecimal (-hex) con bloques de dos caracteres separados por puntos (-c):

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

-Mostramos dichos valores:

```

sequi96@ei141071:~/Escritorio/Home/SPSI/SPSI$ cat CarlosDSAPub.sha384
SHA384(CarlosDSAPub.pem)= 0c:1a:8f:6f:cd:f3:68:61:af:51:21:a5:f5:f2:8f:3a:5c:85:40:0d:7a:59:d9:de:a7:27:22:dd:a6:a5:1d:49:a3:77:e3:90:b2:db:65:12:ee:cd:88:47:af:7c:f0:50

```

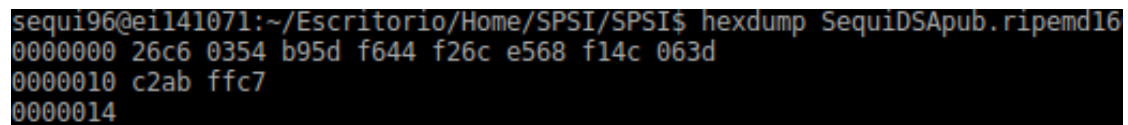
Figura 5.1: CarlosDSAPub.sha384.

**6. EJERCICIO 6. Calculad el valor hash del archivo con la clave pública apellidoDSAPub.pem usando una función hash de 160 bits con salida binaria. Guardad el hash en apellidoDSAPub.[algoritmo] y mostrad su contenido.**

-Usamos esta vez la salida binaria (-binary) con la función hash de 160 bits para el resumen del mensaje (-ripemd160):

```
openssl dgst -ripemd160 -binary -out SequiDSAPub.ripemd160 SequiDSAPub.pem
```

-Mostramos el contenido con la herramienta hexdump:



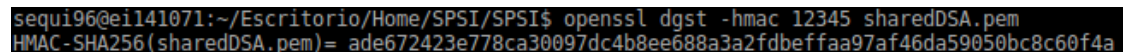
```
sequi96@ei141071:~/Escritorio/Home/SPSI/SPSI$ hexdump SequiDSAPub.ripemd16
00000000 26c6 0354 b95d f644 f26c e568 f14c 063d
00000010 c2ab ffc7
00000014
```

Figura 6.1: SequiDSAPub.ripemd160.

**7. EJERCICIO 7. Generad el valor HMAC del archivo sharedDSA.pem con clave '12345' mostrándolo por pantalla.**

-Generamos el valor HMAC de sharedDSA.pem con la clave '12345'

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



```
sequi96@ei141071:~/Escritorio/Home/SPSI/SPSI$ openssl dgst -hmac 12345 sharedDSA.pem
HMAC-SHA256(sharedDSA.pem)= ade672423e778ca30097dc4b8ee688a3a2fdbeffaa97af46da59050bc8c60f4a
```

Figura 7.1: Valor HMAC de sharedDSA.pem.

**8. EJERCICIO 8. Simulad una ejecución completa del protocolo Estación a Estación. Para ello emplearemos como claves para firma/verificación las generadas en esta práctica, y para el protocolo DH emplearemos las claves asociadas a curvas elípticas de la práctica anterior junto con las de otro usuario simulado que deberéis generar nuevamente. Por ejemplo, si mi clave privada está en javierECpriv.pem y la clave pública del otro usuario está en lobilloECpub.pem, el comando para generar la clave derivada será: `$>openssl pkeyutl -inkey javierECpriv.pem -peerkey lobilloECpub.pem -derive -out key.bin` El algoritmo simétrico a utilizar en el protocolo**

-Una vez tenemos las claves firma/verificación generadas en esta práctica, procedemos a generar los pares de claves de curvas elípticas para ambos usuarios a partir del mismo fichero de parámetros:

- Para el usuario Carlos:

```
openssl ecparam -in ./stdECparam.pem -genkey -noout -out ./CarlosECkey.pem
```

- Para el usuario Sequi:

```
openssl ecparam -in ./stdECparam.pem -genkey -noout -out ./SequiECkey.pem
```

-Obtenemos para cada par de claves de curva elíptica de cada usuario, la privada y la pública:

- SACAMOS LA PRIVADA:

- Para el usuario Carlos:

```
openssl ec -in ./CarlosECkey.pem -out ./CarlosECpriv.pem -outform PEM -des3
```

- Para el usuario Sequi:

```
openssl ec -in ./SequiECkey.pem -out ./SequiECpriv.pem -outform PEM -des3
```

- SACAMOS LA PÚBLICA:

- Para el usuario Carlos:

```
openssl ec -in ./CarlosECkey.pem -pubout -out ./CarlosECpub.pem
```

- Para el usuario Sequi:

```
openssl ec -in ./SequiECkey.pem -pubout -out ./SequiECpub.pem
```

-Derivamos las claves comunes a partir de las dos parejas privadaUsuario1/públicaUsuario2 (y viceversa) compartiendo parámetros. Al compartir el mismo fichero de parámetros las claves de curvas elípticas, las claves que vamos a derivar ahora mismo serán exactamente iguales para Carlos y para Sequi.

- La clave del usuario Carlos:

```
openssl pkeyutl -inkey CarlosECpriv.pem -peerkey SequiECpub.pem -derive -out CarlosKey.bin
```

- La clave del usuario Sequi:

```
openssl pkeyutl -inkey SequiECpriv.pem -peerkey CarlosECpub.pem -derive -out SequiKey.bin
```

-Cada usuario concatena su clave pública de la curva elíptica con la clave pública de la curva elíptica que ha recibido por parte del otro usuario y, utiliza una concatenación para firmar y otra para verificar a identidad del otro usuario:

- Fichero concatenado del usuario Carlos:

- Para firmar: `cat CarlosECpub.pem SequiECpub.pem >ficheroConcatenado-FirmarCarlos`
- Para verificar: `cat SequiECpub.pem CarlosECpub.pem >ficheroConcatenado-VerificarCarlos`

- Fichero concatenado del usuario Sequi:

- Para verificar: `cat CarlosECpub.pem SequiECpub.pem >ficheroConcatenado-VerificarSequi`
- Para firmar: `cat SequiECpub.pem CarlosECpub.pem >ficheroConcatenado-FirmarSequi`

-Cada usuario firma el fichero concatenado obtenido

- Firma por parte del usuario Carlos:

```
openssl dgst -sha256 -sign ./CarlosDSApriv.pem -out ./ficheroFirmadoCarlos.sha256
./ficheroConcatenadoFirmarCarlos
```

- Firma por parte del usuario Sequi:

```
openssl dgst -sha256 -sign ./SequiDSApriv.pem -out ./ficheroFirmadoSequi.sha256
./ficheroConcatenadoFirmarSequi
```

-Cada uno de los usuarios procede a encriptar el fichero firmado con AES-128 en modo CFB8 usando la clave derivada:

- Encriptación en AES-128 modo CFB8 por parte del usuario Carlos:

```
openssl enc -aes-128-cfb8 -pass file:./CarlosKey.bin -in ficheroFirmadoCarlos.sha256
-out ficheroFirmadoEncriptadoCarlos.bin
```

- Encriptación en AES-128 modo CFB8 por parte del usuario Sequi:

```
openssl enc -aes-128-cfb8 -pass file:./SequiKey.bin -in ficheroFirmadoSequi.sha256
-out ficheroFirmadoEncriptadoSequi.bin
```

-Los usuarios reciben los correspondientes ficheros firmados y después encriptados por parte del otro usuario y, proceden a desencriptarlo primeramente usando la clave derivada que, como hemos dicho, son iguales para ambos usuarios:

- Desencriptación en AES-128 modo CFB8 por parte del usuario Carlos:

```
openssl aes-128-cfb8 -d -pass file:./CarlosKey.bin -in ficheroFirmadoEncriptadoSequi.bin -out ./ficheroFirmadoDesencriptadoPorCarlos.sha256
```

- Desencriptación en AES-128 modo CFB8 por parte del usuario Sequi:

```
openssl aes-128-cfb8 -d -pass file:./SequiKey.bin -in ficheroFirmadoEncriptadoCarlos.bin -out ./ficheroFirmadoDesencriptadoPorSequi.sha256
```

-Ahora proceden a la verificación para asegurarse de la comunicación es entre los usuarios que han de ser. El usuario1, para verificar el fichero, utiliza la clave pública que le manda el usuario2 y verifica usando el fichero que ha desencriptado (que previamente e habia mandado el usuario2) y su propio fichero concatenado, el que había creado él mismo antes.

- Verificación por parte del usuario Carlos:

```
openssl dgst -verify SequiDSAPub.pem -signature ./ficheroFirmadoDesencriptado-  
PorCarlos.sha256 ficheroConcatenadoVerificarCarlos
```

- Verificación por parte del usuario Sequi:

```
openssl dgst -verify CarlosDSAPub.pem -signature ./ficheroFirmadoDesencripta-  
doPorSequi.sha256 ficheroConcatenadoVerificarCarlos
```

-Obtenemos un "Verified OK" por terminal en caso de que la verificación haya sido correcta (en caso contrario un "Verification Failure").

```
carlos@carlos-Aspire-E5-551G:~/Escritorio/ETSIIT/SPSI/Prácticas/P3/SPSI$ openssl dgst -verify CarlosDSAPub.pem -signature ./ficheroFirmadoDesencriptadoPorSequi.sha256  
heroConcatenadoVerificarSequi  
Verified OK  
carlos@carlos-Aspire-E5-551G:~/Escritorio/ETSIIT/SPSI/Prácticas/P3/SPSI$ openssl dgst -verify SequiDSAPub.pem -signature ./ficheroFirmadoDesencriptadoPorCarlos.sha256  
heroConcatenadoVerificarCarlos  
Verified OK
```

Figura 8.1: Muestra de la correcta verificación de ambos archivos

## Referencias