

UNIVERSIDAD DE GRANADA
E.T.S.I INFORMÁTICA Y TELECOMUNICACIONES
GRADO EN INGENIERÍA INFORMÁTICA (GII)

Memoria Práctica 3: Protocolos Criptográficos

Seguridad y Protección de Sistemas Informáticos
CURSO 2017-2018

Arthur Rodríguez Nesterenko - DNI Y1680851W

26 de noviembre de 2017

Índice

1. Tarea 1: Generad un archivo sharedDSA.pem que contenga los parámetros. Mostrad los valores. 3
2. Tarea 2: Generad dos parejas de claves para los parámetros anteriores. La claves se almacenarán en los archivos nombreDSAkey.pem y apellidoDSAkey.pem. 4
3. Tarea 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. Tarea 4: Extraed en nombreDSAPub.pem la clave pública contenida en el archivo nombreDSAkey.pem; este no debe estar cifrado ni protegido. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem. 7
5. Tarea 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. Tarea 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. Tarea 7: Generad el valor HMAC del archivo sharedDSA.pem con clave '12345' mostrándolo por pantalla. 10
8. Tarea 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. 11

Para esta práctica, deberéis simular la presencia de dos usuarios, por lo que la generación de claves será doble, incluyendo las reutilizadas de la práctica anterior.

1. Tarea 1: Generad un archivo sharedDSA.pem que contenga los parámetros. Mostrad los valores.

Para generar un fichero que contenga los parámetros de **DSA** o **Digital Signature Algorithm** haremos uso del comando **openssl dsaparam**, en el que utilizaremos una clave de 1024 bits. Esto nos generará un fichero con los primos **p** y **q** a partir de los cuales derivará la pareja de claves pública/privada.

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

El resultado se muestra a continuación:

```
[cvi075215:ficheros Arthur18$ openssl dsaparam -in sharedDSA.pem -noout -text
P:
 00:e4:6b:96:b1:9b:66:8c:53:39:80:76:37:e3:c6:
 25:86:3d:19:50:3d:d4:78:62:34:b6:39:b3:7b:e3:
 be:bb:6b:c9:af:4b:08:cf:5a:cc:31:ec:a3:a7:3b:
 a2:b8:06:a4:6e:da:73:ef:9f:a1:96:71:8e:58:18:
 96:15:fe:83:95:4b:b8:fe:a1:4d:07:a9:ef:c5:7e:
 1c:70:c5:61:9c:8a:35:b0:01:2f:c7:48:c8:b7:da:
 13:11:98:21:36:de:ec:e7:01:30:c6:e5:a6:dd:e3:
 1c:47:b0:9e:dd:ae:d9:ac:6d:ca:61:e9:f0:f6:ff:
 4d:9a:cf:e3:50:34:0e:42:87
Q:
 00:c0:a8:ff:ff:1c:83:87:94:3f:1b:b1:0c:fc:0c:
 55:63:9e:0a:8e:95
G:
 39:41:cc:0d:04:e8:80:43:c4:27:d8:ed:73:aa:9a:
 5c:30:c5:df:6f:83:22:db:21:f9:f8:8e:69:8a:f3:
 23:cd:6a:d8:fb:f2:d6:6b:cf:f2:c9:16:30:aa:da:
 7b:ad:ca:19:c0:b0:6a:25:2f:88:b3:e6:b9:d9:85:
 1d:2e:d6:f1:08:d3:7f:9d:f4:74:4a:38:6c:ff:56:
 04:f4:bd:a8:a7:6f:10:c9:b1:47:2f:ef:65:4f:e3:
 53:67:23:53:b5:bc:e1:b2:54:4f:41:ab:2c:a5:32:
 ba:05:48:ce:55:3a:a3:b0:ff:e6:d4:88:a8:27:49:
 35:cb:ee:18:a2:7e:60:b2
```

Figura 1.1: Parámetros de DSA generados

2. Tarea 2: Generad dos parejas de claves para los parámetros anteriores. La claves se almacenarán en los archivos nombreDSAkey.pem y apellidoDSAkey.pem.

La generación de parejas de claves se hará de manera análoga a como se hacía con las EC, mediante el comando **openssl gendsa**. Este comando recibe como entrada un fichero de parámetros a partir de los cuales se generarán ambas claves como se puede ver en la figura 3.1. Este procedimiento se realiza también para el fichero **RodriguezDSA.key**. Mostramos la salida de solo uno de ellos.

```
openssl gendsa ArthurDSAkey.pem sharedDSA.pem
```

```
read DSA key
Private-Key: (1024 bit)
priv:
    00:9f:ce:72:f9:05:9a:4e:01:1c:57:c5:52:04:b5:
    d3:95:3b:6b:95:ec
pub:
    00:89:9e:85:eb:ae:8f:92:90:50:33:3f:8d:a4:a5:
    12:c3:05:63:0b:ce:2b:ed:6f:2f:d6:3e:f0:c7:a4:
    a7:86:3d:9f:d3:8b:b5:97:4d:6c:46:9c:8e:93:70:
    f3:12:ba:82:28:4c:2a:1e:0e:1a:05:42:29:ae:e3:
    cf:45:3e:4b:f9:6f:57:2a:b3:c1:dc:0e:7c:6b:3c:
    5f:94:8b:29:da:44:ca:a0:84:44:5b:9d:35:9e:e7:
    8a:21:fa:ce:da:fc:33:84:1b:30:09:76:26:4f:ab:
    4b:1c:76:9c:ad:05:bf:08:78:19:d9:f8:44:65:91:
    f7:ab:c5:9a:9a:ce:41:31:d8
```

Figura 2.1: Fichero **ArthurDSAkey** que contiene la pareja de claves del sujeto 1

```
P:
    00:e4:6b:96:b1:9b:66:8c:53:39:80:76:37:e3:c6:
    25:86:3d:19:50:3d:d4:78:62:34:b6:39:b3:7b:e3:
    be:bb:6b:c9:af:4b:08:cf:5a:cc:31:ec:a3:a7:3b:
    a2:b8:06:a4:6e:da:73:ef:9f:a1:96:71:8e:58:18:
    96:15:fe:83:95:4b:b8:fe:a1:4d:07:a9:ef:c5:7e:
    1c:70:c5:61:9c:8a:35:b0:01:2f:c7:48:c8:b7:da:
    13:11:98:21:36:de:ec:e7:01:30:c6:e5:a6:dd:e3:
    1c:47:b0:9e:dd:ae:d9:ac:6d:ca:61:e9:f0:f6:ff:
    4d:9a:cf:e3:50:34:0e:42:87
Q:
    00:c0:a8:ff:ff:1c:83:87:94:3f:1b:b1:0c:fc:0c:
    55:63:9e:0a:8e:95
G:
    39:41:cc:0d:04:e8:80:43:c4:27:d8:ed:73:aa:9a:
    5c:30:c5:df:6f:83:22:db:21:f9:f8:8e:69:8a:f3:
    23:cd:6a:d8:fb:f2:d6:6b:cf:f2:c9:16:30:aa:da:
    7b:ad:ca:19:c0:b0:6a:25:2f:88:b3:e6:b9:d9:85:
    1d:2e:d6:f1:08:d3:7f:9d:f4:74:4a:38:6c:ff:56:
    04:f4:bd:a8:a7:6f:10:c9:b1:47:2f:ef:65:4f:e3:
    53:67:23:53:b5:bc:e1:b2:54:4f:41:ab:2c:a5:32:
    ba:05:48:ce:55:3a:a3:b0:ff:e6:d4:88:a8:27:49:
    35:cb:ee:18:a2:7e:60:b2
```

Figura 2.2: Fichero **ArthurDSAkey** que contiene la pareja de claves del sujeto 1

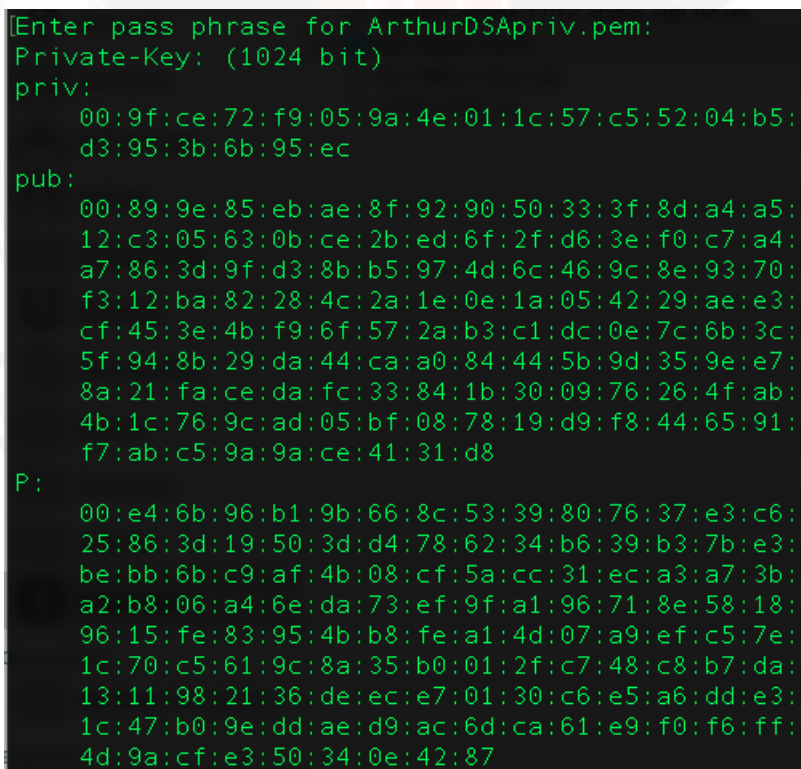
3. Tarea 3: “Extraed”la clave privada contenida en el archivo nombreDSAkey.pem a otro archivo que tenga por nombre nombreDSApriv.pem. Este archivo deberá estar protegido por contraseña. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem.

Para extraer la clave privada de un fichero de claves generado utilizamos el comando **openssl dsa**, con la misma clave que para la práctica anterior: **0123456789**. El cifrado simétrico a utilizar será AES128.

```
openssl dsa -in ArthurDSAkey.pem -out ArthurDSApriv.pem -aes128
```

Para mostrar el resultado de la extracción de la clave privada utilizamos el comando:

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



```
[Enter pass phrase for ArthurDSApriv.pem:
Private-Key: (1024 bit)
priv:
    00:9f:ce:72:f9:05:9a:4e:01:1c:57:c5:52:04:b5:
    d3:95:3b:6b:95:ec
pub:
    00:89:9e:85:eb:ae:8f:92:90:50:33:3f:8d:a4:a5:
    12:c3:05:63:0b:ce:2b:ed:6f:2f:d6:3e:f0:c7:a4:
    a7:86:3d:9f:d3:8b:b5:97:4d:6c:46:9c:8e:93:70:
    f3:12:ba:82:28:4c:2a:1e:0e:1a:05:42:29:ae:e3:
    cf:45:3e:4b:f9:6f:57:2a:b3:c1:dc:0e:7c:6b:3c:
    5f:94:8b:29:da:44:ca:a0:84:44:5b:9d:35:9e:e7:
    8a:21:fa:ce:da:fc:33:84:1b:30:09:76:26:4f:ab:
    4b:1c:76:9c:ad:05:bf:08:78:19:d9:f8:44:65:91:
    f7:ab:c5:9a:9a:ce:41:31:d8
P:
    00:e4:6b:96:b1:9b:66:8c:53:39:80:76:37:e3:c6:
    25:86:3d:19:50:3d:d4:78:62:34:b6:39:b3:7b:e3:
    be:bb:6b:c9:af:4b:08:cf:5a:cc:31:ec:a3:a7:3b:
    a2:b8:06:a4:6e:da:73:ef:9f:a1:96:71:8e:58:18:
    96:15:fe:83:95:4b:b8:fe:a1:4d:07:a9:ef:c5:7e:
    1c:70:c5:61:9c:8a:35:b0:01:2f:c7:48:c8:b7:da:
    13:11:98:21:36:de:ec:e7:01:30:c6:e5:a6:dd:e3:
    1c:47:b0:9e:dd:ae:d9:ac:6d:ca:61:e9:f0:f6:ff:
    4d:9a:cf:e3:50:34:0e:42:87
```

Figura 3.1: Clave privada DSA del sujeto **Arthur**

```

Q:
  00:c0:a8:ff:ff:1c:83:87:94:3f:1b:b1:0c:fc:0c:
  55:63:9e:0a:8e:95
G:
  39:41:cc:0d:04:e8:80:43:c4:27:d8:ed:73:aa:9a:
  5c:30:c5:df:6f:83:22:db:21:f9:f8:8e:69:8a:f3:
  23:cd:6a:d8:fb:f2:d6:6b:cf:f2:c9:16:30:aa:da:
  7b:ad:ca:19:c0:b0:6a:25:2f:88:b3:e6:b9:d9:85:
  1d:2e:d6:f1:08:d3:7f:9d:f4:74:4a:38:6c:ff:56:
  04:f4:bd:a8:a7:6f:10:c9:b1:47:2f:ef:65:4f:e3:
  53:67:23:53:b5:bc:e1:b2:54:4f:41:ab:2c:a5:32:
  ba:05:48:ce:55:3a:a3:b0:ff:e6:d4:88:a8:27:49:
  35:cb:ee:18:a2:7e:60:b2

```

Figura 3.2: Clave privada DSA del sujeto **Arthur**

Realizamos la misma operación para el segundo fichero **RodriguezDSAkey.pem**

```
openssl dsa -in RodriguezDSAkey.pem -out RodriguezDSApriv.pem -aes128
```

```

[Enter pass phrase for RodriguezDSApriv.pem:
Private-Key: (1024 bit)
priv:
  41:11:09:16:5c:60:c7:f6:20:7c:46:af:6b:9c:bc:
  03:8b:ee:52:f4
pub:
  2c:9e:0b:f6:c4:b7:b8:29:a0:a4:09:b4:02:f3:39:
  7b:e0:10:64:d3:25:0e:48:31:8b:b5:41:65:ea:ca:
  2e:b7:6e:b4:8d:03:f0:d8:26:0d:3e:d4:83:d7:8f:
  6e:fc:d9:fa:f2:b1:a7:d9:27:d5:c0:10:55:7c:8b:
  a2:1b:9d:56:a7:47:76:73:4f:65:b2:8c:14:33:7b:
  5d:de:3d:1f:1c:f4:87:6f:00:bd:25:aa:27:df:e6:
  b5:0a:5c:b5:07:b4:2a:c1:3d:b9:3a:8e:9e:b9:a6:
  cc:c0:f5:ab:ce:4d:94:2c:dc:94:2f:ec:2e:fe:f5:
  64:6d:47:a6:14:e0:f2:d0
P:
  00:e4:6b:96:b1:9b:66:8c:53:39:80:76:37:e3:c6:
  25:86:3d:19:50:3d:d4:78:62:34:b6:39:b3:7b:e3:
  be:bb:6b:c9:af:4b:08:cf:5a:cc:31:ec:a3:a7:3b:
  a2:b8:06:a4:6e:da:73:ef:9f:a1:96:71:8e:58:18:
  96:15:fe:83:95:4b:b8:fe:a1:4d:07:a9:ef:c5:7e:
  1c:70:c5:61:9c:8a:35:b0:01:2f:c7:48:c8:b7:da:
  13:11:98:21:36:de:ec:e7:01:30:c6:e5:a6:dd:e3:
  1c:47:b0:9e:dd:ae:d9:ac:6d:ca:61:e9:f0:f6:ff:
  4d:9a:cf:e3:50:34:0e:42:87

```

Figura 3.3: Clave privada DSA del sujeto **Rodriguez**

```

Q:
  00:c0:a8:ff:ff:1c:83:87:94:3f:1b:b1:0c:fc:0c:
  55:63:9e:0a:8e:95
G:
  39:41:cc:0d:04:e8:80:43:c4:27:d8:ed:73:aa:9a:
  5c:30:c5:df:6f:83:22:db:21:f9:f8:8e:69:8a:f3:
  23:cd:6a:d8:fb:f2:d6:6b:cf:f2:c9:16:30:aa:da:
  7b:ad:ca:19:c0:b0:6a:25:2f:88:b3:e6:b9:d9:85:
  1d:2e:d6:f1:08:d3:7f:9d:f4:74:4a:38:6c:ff:56:
  04:f4:bd:a8:a7:6f:10:c9:b1:47:2f:ef:65:4f:e3:
  53:67:23:53:b5:bc:e1:b2:54:4f:41:ab:2c:a5:32:
  ba:05:48:ce:55:3a:a3:b0:ff:e6:d4:88:a8:27:49:
  35:cb:ee:18:a2:7e:60:b2

```

Figura 3.4: Clave privada DSA del sujeto **Rodriguez**

4. Tarea 4: Extraed en `nombreDSAPub.pem` la clave pública contenida en el archivo `nombreDSAkey.pem`; este no debe estar cifrado ni protegido. Mostrad sus valores. Lo mismo para el archivo `apellidoDSAkey.pem`.

Realizamos un procedimiento análogo al anterior pero esta vez con las claves públicas. Para el fichero **ArthurDSAkey**:

```
openssl dsa -in ArthurDSAkey.pem -out ArthurDSAPub.pem -pubout
```

```

read DSA key
pub:
  00:89:9e:85:eb:ae:8f:92:90:50:33:3f:8d:a4:a5:
  12:c3:05:63:0b:ce:2b:ed:6f:2f:d6:3e:f0:c7:a4:
  a7:86:3d:9f:d3:8b:b5:97:4d:6c:46:9c:8e:93:70:
  f3:12:ba:82:28:4c:2a:1e:0e:1a:05:42:29:ae:e3:
  cf:45:3e:4b:f9:6f:57:2a:b3:c1:dc:0e:7c:6b:3c:
  5f:94:8b:29:da:44:ca:a0:84:44:5b:9d:35:9e:e7:
  8a:21:fa:ce:da:fc:33:84:1b:30:09:76:26:4f:ab:
  4b:1c:76:9c:ad:05:bf:08:78:19:d9:f8:44:65:91:
  f7:ab:c5:9a:9a:ce:41:31:d8
P:
  00:e4:6b:96:b1:9b:66:8c:53:39:80:76:37:e3:c6:
  25:86:3d:19:50:3d:d4:78:62:34:b6:39:b3:7b:e3:
  be:bb:6b:c9:af:4b:08:cf:5a:cc:31:ec:a3:a7:3b:
  a2:b8:06:a4:6e:da:73:ef:9f:a1:96:71:8e:58:18:
  96:15:fe:83:95:4b:b8:fe:a1:4d:07:a9:ef:c5:7e:
  1c:70:c5:61:9c:8a:35:b0:01:2f:c7:48:c8:b7:da:
  13:11:98:21:36:de:ec:e7:01:30:c6:e5:a6:dd:e3:
  1c:47:b0:9e:dd:ae:d9:ac:6d:ca:61:e9:f0:f6:ff:
  4d:9a:cf:e3:50:34:0e:42:87

```

Figura 4.1: Clave pública DSA del sujeto **Arthur**


```

Q:
 00:c0:a8:ff:ff:1c:83:87:94:3f:1b:b1:0c:fc:0c:
55:63:9e:0a:8e:95
G:
39:41:cc:0d:04:e8:80:43:c4:27:d8:ed:73:aa:9a:
5c:30:c5:df:6f:83:22:db:21:f9:f8:8e:69:8a:f3:
23:cd:6a:d8:fb:f2:d6:6b:cf:f2:c9:16:30:aa:da:
7b:ad:ca:19:c0:b0:6a:25:2f:88:b3:e6:b9:d9:85:
1d:2e:d6:f1:08:d3:7f:9d:f4:74:4a:38:6c:ff:56:
04:f4:bd:a8:a7:6f:10:c9:b1:47:2f:ef:65:4f:e3:
53:67:23:53:b5:bc:e1:b2:54:4f:41:ab:2c:a5:32:
ba:05:48:ce:55:3a:a3:b0:ff:e6:d4:88:a8:27:49:
35:cb:ee:18:a2:7e:60:b2

```

Figura 4.2: Clave pública DSA del sujeto **Arthur**

Para el fichero **RodríguezDSAkey**:

```
openssl dsa -in RodriguezDSAkey.pem -out RodriguezDSAPub.pem -pubout
```

```

read DSA key
pub:
 2c:9e:0b:f6:c4:b7:b8:29:a0:a4:09:b4:02:f3:39:
7b:e0:10:64:d3:25:0e:48:31:8b:b5:41:65:ea:ca:
2e:b7:6e:b4:8d:03:f0:d8:26:0d:3e:d4:83:d7:8f:
6e:fc:d9:fa:f2:b1:a7:d9:27:d5:c0:10:55:7c:8b:
a2:1b:9d:56:a7:47:76:73:4f:65:b2:8c:14:33:7b:
5d:de:3d:1f:1c:f4:87:6f:00:bd:25:aa:27:df:e6:
b5:0a:5c:b5:07:b4:2a:c1:3d:b9:3a:8e:9e:b9:a6:
cc:c0:f5:ab:ce:4d:94:2c:dc:94:2f:ec:2e:fe:f5:
64:6d:47:a6:14:e0:f2:d0
P:
 00:e4:6b:96:b1:9b:66:8c:53:39:80:76:37:e3:c6:
25:86:3d:19:50:3d:d4:78:62:34:b6:39:b3:7b:e3:
be:bb:6b:c9:af:4b:08:cf:5a:cc:31:ec:a3:a7:3b:
a2:b8:06:a4:6e:da:73:ef:9f:a1:96:71:8e:58:18:
96:15:fe:83:95:4b:b8:fe:a1:4d:07:a9:ef:c5:7e:
1c:70:c5:61:9c:8a:35:b0:01:2f:c7:48:c8:b7:da:
13:11:98:21:36:de:ec:e7:01:30:c6:e5:a6:dd:e3:
1c:47:b0:9e:dd:ae:d9:ac:6d:ca:61:e9:f0:f6:ff:
4d:9a:cf:e3:50:34:0e:42:87

```

Figura 4.3: Clave pública DSA del sujeto **Rodríguez**


```

Q:
  00:c0:a8:ff:ff:1c:83:87:94:3f:1b:b1:0c:fc:0c:
  55:63:9e:0a:8e:95
G:
  39:41:cc:0d:04:e8:80:43:c4:27:d8:ed:73:aa:9a:
  5c:30:c5:df:6f:83:22:db:21:f9:f8:8e:69:8a:f3:
  23:cd:6a:d8:fb:f2:d6:6b:cf:f2:c9:16:30:aa:da:
  7b:ad:ca:19:c0:b0:6a:25:2f:88:b3:e6:b9:d9:85:
  1d:2e:d6:f1:08:d3:7f:9d:f4:74:4a:38:6c:ff:56:
  04:f4:bd:a8:a7:6f:10:c9:b1:47:2f:ef:65:4f:e3:
  53:67:23:53:b5:bc:e1:b2:54:4f:41:ab:2c:a5:32:
  ba:05:48:ce:55:3a:a3:b0:ff:e6:d4:88:a8:27:49:
  35:cb:ee:18:a2:7e:60:b2

```

Figura 4.4: Clave pública DSA del sujeto **Rodriguez**

5. Tarea 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.

El comando a utilizar es **openssl dgst** con una determinada configuración. Vamos a ver como configurar este comando. La configuración del digest será **-sha384** con la salida **-hex**.

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

La salida estandar se muestra a continuación:

```

[cvi075215:ficheros Arthur18$ cat ArthurDSAPub.sha384
SHA384(ArthurDSAPub.pem)= 61:c3:81:c5:50:22:de:23:63:2a:8b
:b2:a2:2d:94:f2:3b:b6:9e:b2:32:dc:0e:c7:fe:fa:80:e2:10:20:
99:3f:21:37:3d:98:11:df:5e:74:83:a0:ce:a4:df:19:30:c1

```

Figura 5.1: DSA priv1

6. Tarea 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.

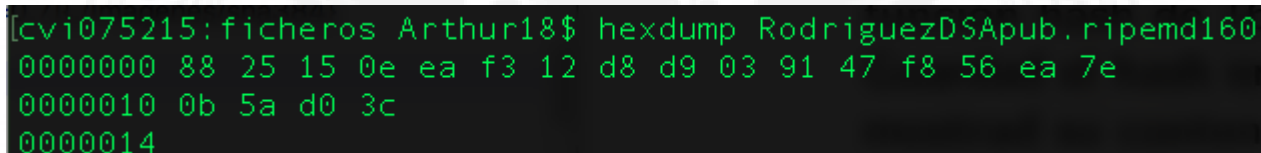
Para esta tarea basta con conocer cuales son los **digest** que ofrece openssl. Esto se puede conocer fácilmente con el comando:

```
openssl list-message-digest-commands
```

Nos indica que existe un digest llamado **ripemd160**, sobre el cual efectuamos una búsqueda rápida para comprobar que efectivamente es una función hash de 160 bits. Por tanto el comando que calcula el hash con este digest es el siguiente:

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

Salida:



```
[cvi075215:archivos Arthur18$ hexdump RodriguezDSAPub.ripemd160
00000000 88 25 15 0e ea f3 12 d8 d9 03 91 47 f8 56 ea 7e
00000010 0b 5a d0 3c
00000014
```

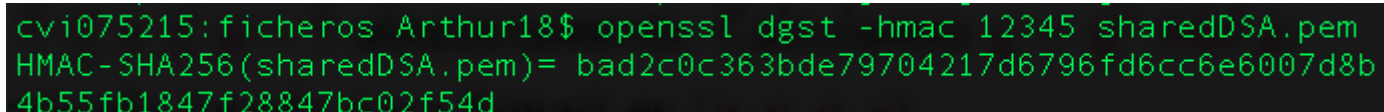
Figura 6.1: DSA priv1

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

Para generar el HMAC con una clave utilizamos el siguiente comando:

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

Resultado:



```
[cvi075215:archivos Arthur18$ openssl dgst -hmac 12345 sharedDSA.pem
HMAC-SHA256(sharedDSA.pem)= bad2c0c363bde79704217d6796fd6cc6e6007d8b
4b55fb1847f28847bc02f54d
```

Figura 7.1: DSA priv1

8. Tarea 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.

Para simular una ejecución completa del protocolo **Estación a Estación** tenemos que seguir una serie de pasos que iremos describiendo a continuación, teniendo en cuenta que reutilizaremos ficheros de claves de la práctica anterior. Para cada uno de estos pasos mostraremos capturas de pantalla correspondientes en caso de ser necesario.

1. Recuperar de la práctica anterior los ficheros correspondientes a curvas elípticas. Estos son los ficheros: **stdECparam** (correspondiente a los parámetros de la curva elíptica) y los dos ficheros de claves correspondientes a nuestro primer sujeto, en este caso **Arthur**, serán **ArthurECpriv** y **ArthurECpub**
2. Generamos la pareja de claves asociadas a curvas elípticas del segundo sujeto, en este caso **Rodríguez**, de la misma forma que para la práctica anterior, con el comando:

```
openssl ecparam -in stdECparam.pem -genkey -out RodriguezECkey.pem -noout
```

3. Ahora solo queda extraer la pareja pública/privada para este segundo sujeto. Los comandos son:

```
openssl ec -in RodriguezECkey.pem -out RodriguezECpriv.pem -outform PEM -des3
```

Para cifrar con DES3 utilizamos la misma clave que en la práctica anterior: **012345679**.
Extraemos la clave pública del sujeto 2.

```
openssl ec -in RodriguezECkey.pem -pubout -out RodriguezECpub.pem
```

A continuación se muestran todos los ficheros de claves generados a partir de curvas elípticas correspondientes a ambos sujetos.

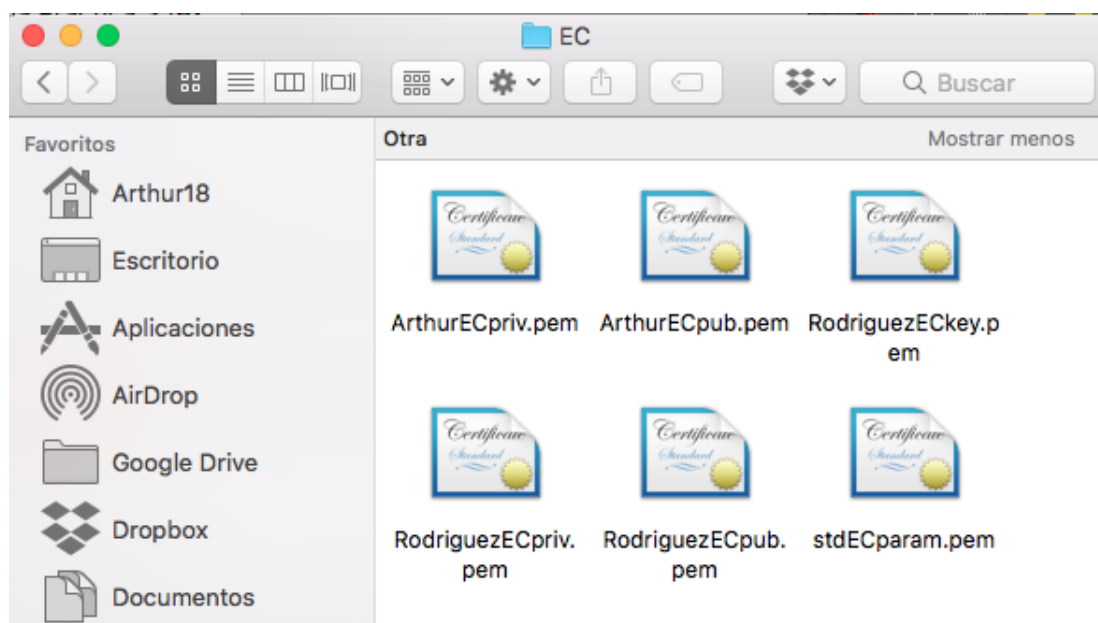


Figura 8.1: Ficheros de claves generados a partir de ECs de ambos sujetos.

4. Una vez preparadas todas las claves de ambos sujetos, el siguiente paso consiste en que cada uno de ellos calcule una clave común **key.bin** a partir de la clave privada de la curva elíptica de cada uno y la clave pública del otro sujeto. El comando a utilizar es el siguiente, y se hace el mismo procedimiento en ambos sentidos:

El sujeto 1 (**Arthur**) calcula la clave común en el fichero **Arthurkey.bin**

```
openssl pkeyutl -inkey ArthurECpriv.pem -peerkey RodriguezECpub.pem -derive -out Arthurkey.bin
```

El sujeto 2 (**Rodríguez**) hace lo propio (en este caso lo hacemos de forma paralela, dado que estamos simulando a ambos sujetos).

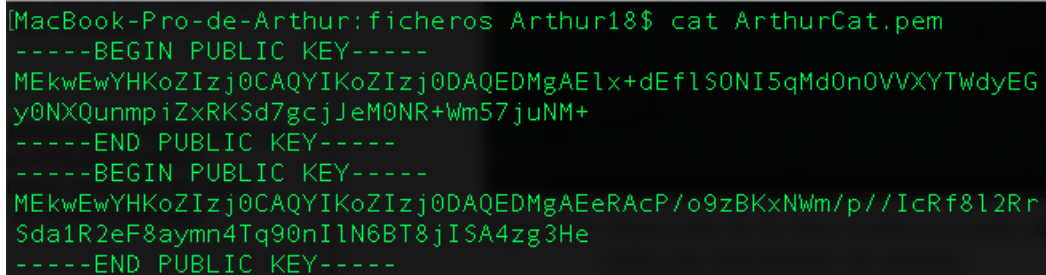
```
openssl pkeyutl -inkey RodriguezECpriv.pem -peerkey ArthurECpub.pem -derive -out Rodriguezkey.bin
```

```
[MacBook-Pro-de-Arthur:archivos Arthur18$ openssl pkeyutl -inkey ArthurECpriv.pem -peerkey RodriguezECpub.pem -derive -out Arthurkey.bin
[Enter pass phrase for ArthurECpriv.pem:
[MacBook-Pro-de-Arthur:archivos Arthur18$ openssl pkeyutl -inkey RodriguezECpriv.pem -peerkey ArthurECpub.pem -derive -out Rodriguezkey.bin
[Enter pass phrase for RodriguezECpriv.pem:
[MacBook-Pro-de-Arthur:archivos Arthur18$ diff Arthurkey.bin Rodriguezkey.bin
```

Figura 8.2: Generación de la clave común por parte de ambos sujetos. Las claves generadas son exactamente las mismas.

5. El protocolo Estación a Estación comienza a continuación. El sujeto **Arthur** concatenará su **clave pública generada a partir de la curva elíptica** común junto con la clave pública del sujeto **Rodríguez** en un fichero llamado **ArthurCat.pem**.

```
cat ArthurECpub.pem RodriguezECpub.pem > ArthurCat.pem
```



```
MacBook-Pro-de-Arthur:archivos Arthur18$ cat ArthurCat.pem
-----BEGIN PUBLIC KEY-----
MEkwEwYHkoZIZj0CAQYIKoZIZj0DAQEDMgAE1x+dEf1S0NI5qMd0n0VVXYTWdyEG
y0NXQuunmpiZxRKsd7gcjJeM0NR+Wm57juNM+
-----END PUBLIC KEY-----
-----BEGIN PUBLIC KEY-----
MEkwEwYHkoZIZj0CAQYIKoZIZj0DAQEDMgAEeRACp/o9zBKxNWm/p//IcRf812Rr
Sda1R2eF8aymn4Tq90nI1N6BT8jISA4zg3He
-----END PUBLIC KEY-----
```

Figura 8.3: Concatenación de los dos ficheros de claves públicas

6. Ahora el sujeto **Arthur** procederá a firmar con su **clave privada de DSA** el fichero concatenado y posteriormente lo cifrará con el algoritmo simétrico **AES-128 en modo CFB8**, utilizando la clave común generada en el paso 2. Firmamos el fichero de claves concatenadas con la clave privada de **Arthur**, indicándole como digest el **-sha256**. Este comando lo que hace internamente es **calcular el hash del fichero concatenado** y añadirlo al mismo, de forma que posteriormente se pueda verificar que ese hash es único para el fichero en cuestión.

```
openssl dgst -sha256 -sign ArthurDSPriv.pem -out SgnArthurCat.sha256
ArthurCat.pem
```

Posteriormente ciframos el **fichero firmado con AES-128 CFB8 y la clave común generada anteriormente**, para finalmente simular un envío a nuestro sujeto 2.

```
openssl enc -aes-128-cfb8 -pass file:Arthurkey.bin -in SgnArthurCat.sha256
-out EncSgnArthurCat.bin
```

7. Una vez el sujeto 2 ha recibido el fichero **EncSgnArthurCat.bin** procederá a **verificar** que el fichero que ha recibido procede **realmente** de quien lo ha enviado. Para verificar primero ha de descifrar el fichero utilizando **la clave común generada en el paso 4**. Esto se realiza con el siguiente comando.

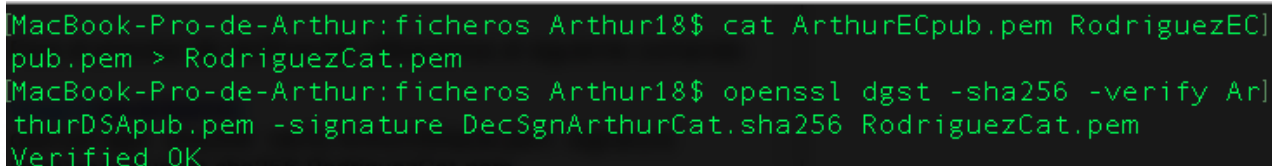
```
openssl aes-128-cfb8 -d -in EncSgnArthurCat.bin -out DecSgnArthurCat.sha256
-pass file:Rodriguezkey.bin
```

Para el proceso de verificación hace falta que el segundo sujeto (**Rodríguez**) concatene los dos ficheros de clave pública en el mismo sentido que lo hizo el sujeto 1 **Arthur**, para posteriormente realizar la verificación. Concatenamos ambos ficheros con:

```
cat ArthurECpub.pem RodriguezECpub.pem > RodriguezCat.pem
```

Para el proceso de verificación utilizaremos el siguiente comando

```
openssl dgst -sha256 -verify ArthurDSAPub.pem  
-signature DecSgnArthurCat.sha256 RodriguezCat.pem
```



```
MacBook-Pro-de-Arthur:archivos Arthur18$ cat ArthurECpub.pem RodriguezECpub.pem > RodriguezCat.pem  
MacBook-Pro-de-Arthur:archivos Arthur18$ openssl dgst -sha256 -verify ArthurDSAPub.pem -signature DecSgnArthurCat.sha256 RodriguezCat.pem  
Verified OK
```

Figura 8.4: Verificación realizada por parte del sujeto 2 **Rodriguez**

que nos muestra que la verificación ha sido realizada con éxito y por tanto **el sujeto Rodriguez** puede estar seguro de que **ha sido Arthur** quien le ha enviado ese mensaje, garantizándose la autenticidad y el no repudio del mensaje.

8. Una vez el sujeto **Rodriguez** ha verificado con éxito quien es el emisor del mensaje, procederá a realizar el mismo proceso de firma que el que hizo **Arthur**. Esta vez, concatenará los ficheros en sentido contrario:

```
cat RodriguezECpub.pem ArthurECpub.pem > RodriguezCat2.pem
```

Firmará el fichero con su clave privada de DSA:

```
openssl dgst -sha256 -sign RodriguezDSApriv.pem -out SgnRodriguezCat.sha256  
RodriguezCat2.pem
```

Y cifrará el fichero firmado con el **mismo algoritmo, modo y clave de cifrado** que en la anterior.

```
openssl enc -aes-128-cfb8 -pass file:Rodriguezkey.bin  
-in SgnRodriguezCat.sha256 -out EncSgnRodriguezCat.bin
```

9. El último paso consistirá en que el sujeto 1 **Arthur** verifique que es **Rodriguez** quien le ha enviado el mensaje. Para esto volveremos a concatenar las claves públicas de ambos en el sentido adecuado:

```
cat RodriguezECpub.pem ArthurECpub.pem > ArthurCat2.pem
```

Posteriormente solo queda descifrar el fichero firmado

```
openssl aes-128-cfb8 -d -in EncSgnRodriguezCat.bin -out  
DecSgnRodriguezCat.sha256 -pass file:Arthurkey.bin
```

y verificar la firma de la misma forma que antes:


```
openssl dgst -sha256 -verify RodriguezDSAPub.pem  
-signature DecSgnRodriguezCat.sha256 ArthurCat2.pem
```

```
[MacBook-Pro-de-Arthur:ficheros Arthur18$ openssl dgst -sha256 -verify RodriguezDSAPub.pem -signature DecSgnRodriguezCat.sha256 ArthurCat2.pem  
Verified OK
```

Figura 8.5: Verificación realizada por parte del sujeto 1 **Arthur**

A partir de este momento, ambos usuarios están seguros de que aquella persona con la que establecen una comunicación es realmente quien dice ser y por tanto se puede garantizar un **proceso de intercambio de mensajes seguro** utilizando, por ejemplo, la clave común generada a partir de los parámetros de las EC y además auténtico y garantizando el no repudio.

