

# Práctica 3

## PROTOCOLOS CRIPTOGRÁFICOS



Manuel Ruiz Maldonado

## Índice

Ejercicio 1.....	3
Ejercicio 2.....	4
Ejercicio 3.....	4
Ejercicio 4.....	6
Ejercicio 5.....	8
Ejercicio 6.....	9
Ejercicio 7.....	10
Ejercicio 8.....	10
Conclusión.....	14
Índice de imágenes.....	15
Bibliografía.....	16

En esta práctica voy a seguir con el uso de OpenSSL, esta vez para aprender el funcionamiento del DSA (Digital Signature Algorithm) y las funciones Hash.

## Ejercicio 1

**Generad un archivo sharedDSA.pem que contenga los parámetros. Mostrad los valores.**

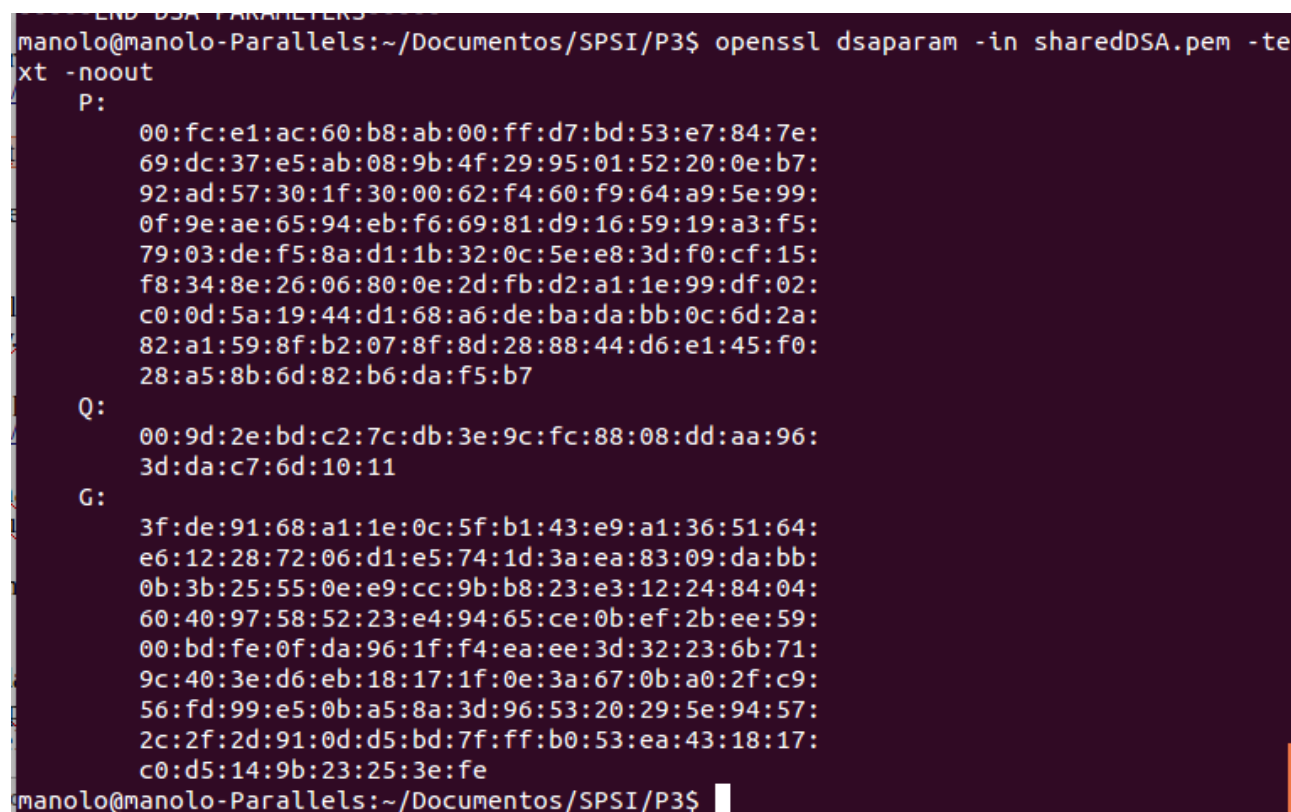
Lo primero que se pide es generar los parámetros necesarios para poder utilizar este estándar. Se van a generar 3 parámetros:

- Un primo  $q$ .
- Un primo  $p$  tal que  $p = 2kq + 1$ .
- Un valor  $g$  tal que  $g = h^{(p-1)/q} \bmod p$  y  $g \neq 1 \bmod p$  y  $h \in \mathbb{Z}_p^*$

Para generar los parámetros del protocolo DSA utilizo la sintaxis que aparece en [1]. La orden utilizada es la siguiente:

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

El comando `openssl dsaparam` permite generar los parámetros. Con la opción `-out` se indica el fichero en el que se va a guardar la salida. Al final se indica el tamaño en bits, que como esta vez no se especificaba he elegido que sean 1024 bits.



```
manolo@manolo-Parallels:~/Documentos/SPSI/P3$ openssl dsaparam -in sharedDSA.pem -text -noout
P:
00:fc:e1:ac:60:b8:ab:00:ff:d7:bd:53:e7:84:7e:
69:dc:37:e5:ab:08:9b:4f:29:95:01:52:20:0e:b7:
92:ad:57:30:1f:30:00:62:f4:60:f9:64:a9:5e:99:
0f:9e:ae:65:94:eb:f6:69:81:d9:16:59:19:a3:f5:
79:03:de:f5:8a:d1:1b:32:0c:5e:e8:3d:f0:cf:15:
f8:34:8e:26:06:80:0e:2d:fb:d2:a1:1e:99:df:02:
c0:0d:5a:19:44:d1:68:a6:de:ba:da:bb:0c:6d:2a:
82:a1:59:8f:b2:07:8f:8d:28:88:44:d6:e1:45:f0:
28:a5:8b:6d:82:b6:da:f5:b7

Q:
00:9d:2e:bd:c2:7c:db:3e:9c:fc:88:08:dd:aa:96:
3d:da:c7:6d:10:11

G:
3f:de:91:68:a1:1e:0c:5f:b1:43:e9:a1:36:51:64:
e6:12:28:72:06:d1:e5:74:1d:3a:ea:83:09:da:bb:
0b:3b:25:55:0e:e9:cc:9b:b8:23:e3:12:24:84:04:
60:40:97:58:52:23:e4:94:65:ce:0b:ef:2b:ee:59:
00:bd:fe:0f:da:96:1f:f4:ea:ee:3d:32:23:6b:71:
9c:40:3e:d6:eb:18:17:1f:0e:3a:67:0b:a0:2f:c9:
56:fd:99:e5:0b:a5:8a:3d:96:53:20:29:5e:94:57:
2c:2f:2d:91:0d:d5:bd:7f:ff:b0:53:ea:43:18:17:
c0:d5:14:9b:23:25:3e:fe
manolo@manolo-Parallels:~/Documentos/SPSI/P3$
```

*Imagen 1: Conjunto de parámetros generados para utilizar en DSA.*

Puedo ver el contenido de este fichero en la Imagen 1. Para ello se utiliza también una orden:

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

En este caso se indica como parámetro de entrada con la opción *-in* el fichero que contiene el conjunto de parámetros. Las opciones *-text* y *-noout* nos permiten ver los parámetros en un formato que es legible (en este caso hexadecimal) y sin mostrar la versión codificada de los mismos.

## 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.**

Para generar las claves a partir de los parámetros anteriores voy a utilizar la sintaxis de [2]. Así obtendremos un par de claves pública y privada para un determinado usuario para el protocolo DSA. Como pide generar dos parejas de claves voy a utilizar la misma orden dos veces cambiando únicamente el fichero de salida.

```
$$ openssl gendsa -out manuelDSAkey.pem sharedDSA.pem  
$$ openssl gendsa -out ruizDSAkey.pem sharedDSA.pem
```

El comando *openssl gendsa* permite generar la pareja de claves a partir del conjunto de parámetros que se indica al final de la orden, en *sharedDSA.pem*. En *-out* se indica el fichero de salida, que es el que cambia para simular que dos usuarios han generado sus claves a partir del mismo conjunto de parámetros.

## 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*.**

Para extraer ahora la clave privada voy a utilizar un nuevo comando. Puedo conocer su funcionamiento en [3]. A partir del fichero que contiene la clave vamos a extraer el fichero que almacenará la privada, aunque en realidad este fichero almacenará también la pública, al igual que el fichero original. De nuevo, se usa dos veces la misma orden, una vez para cada uno de los usuarios. La orden usada es la siguiente:

```
$$ openssl dsa -in manuelDSAkey.pem -out manuelDSApriv.pem -aes128  
$$ openssl dsa -in ruizDSAkey.pem -out ruizDSApriv.pem -aes128
```

El comando *openssl dsa* permite manipular claves de DSA. Con *-in* indicamos cuál es el fichero que contiene la clave. Con *-out* indicamos dónde queremos guardar la clave privada. En este caso añadimos al final *-aes128* porque se pide proteger con contraseña el fichero. La contraseña introducida ha sido “0123456789” en ambos casos.

De nuevo, este comando tiene las opciones *-text* y *-noout* para poder mostrar los valores de las claves.

```
$$ openssl dsa -in manuelDSApriv.pem -text -noout  
$$ openssl dsa -in ruizDSApriv.pem -text -noout
```

En las Imágenes 2 y 3 se pueden comprobar los resultados obtenidos tras utilizar las órdenes anteriores respectivamente. Cabe destacar que aunque el proceso utilizado por los dos usuarios ha

sido el mismo, las claves generadas para cada uno de ellos son distintas y los parámetros son los mismos.

```
read DSA key
Enter pass phrase for manuelDSApriv.pem:
Private-Key: (1024 bit)
priv:
    23:0a:fe:1d:a8:6b:d1:cf:83:6c:ab:94:94:f6:96:
    b5:0f:98:f0:c1
pub:
    43:66:a7:5b:d8:53:8a:4b:30:28:c7:68:91:d8:90:
    57:95:d3:18:c0:70:c0:6e:83:b9:5c:91:52:a4:ba:
    b6:69:52:7e:23:4e:56:dd:db:93:4d:37:00:56:3c:
    65:b6:38:b2:23:15:83:aa:42:ad:81:30:5f:5f:16:
    d2:84:2e:cc:11:29:a3:70:ee:46:25:b0:e4:f9:3c:
    40:f5:04:31:c1:a0:28:07:17:1c:18:bf:24:07:c4:
    24:0a:1e:6f:9e:94:44:29:b5:99:a0:dd:56:df:2b:
    80:d8:45:f8:74:47:d1:d7:02:55:df:23:63:e0:fc:
    cf:c8:5c:e3:bb:7c:a2:bf
P:
    00:fc:e1:ac:60:b8:ab:00:ff:d7:bd:53:e7:84:7e:
    69:dc:37:e5:ab:08:9b:4f:29:95:01:52:20:0e:b7:
    92:ad:57:30:1f:30:00:62:f4:60:f9:64:a9:5e:99:
    0f:9e:ae:65:94:eb:f6:69:81:d9:16:59:19:a3:f5:
    79:03:de:f5:8a:d1:1b:32:0c:5e:e8:3d:f0:cf:15:
    f8:34:8e:26:06:80:0e:2d:fb:d2:a1:1e:99:df:02:
    c0:0d:5a:19:44:d1:68:a6:de:ba:da:bb:0c:6d:2a:
    82:a1:59:8f:b2:07:8f:8d:28:88:44:d6:e1:45:f0:
    28:a5:8b:6d:82:b6:da:f5:b7
Q:
    00:9d:2e:bd:c2:7c:db:3e:9c:fc:88:08:dd:aa:96:
    3d:da:c7:6d:10:11
G:
    3f:de:91:68:a1:1e:0c:5f:b1:43:e9:a1:36:51:64:
    e6:12:28:72:06:d1:e5:74:1d:3a:ea:83:09:da:bb:
    0b:3b:25:55:0e:e9:cc:9b:b8:23:e3:12:24:84:04:
    60:40:97:58:52:23:e4:94:65:ce:0b:ef:2b:ee:59:
    00:bd:fe:0f:da:96:1f:f4:ea:ee:3d:32:23:6b:71:
    9c:40:3e:d6:eb:18:17:1f:0e:3a:67:0b:a0:2f:c9:
    56:fd:99:e5:0b:a5:8a:3d:96:53:20:29:5e:94:57:
    2c:2f:2d:91:0d:d5:bd:7f:ff:b0:53:ea:43:18:17:
    c0:d5:14:9b:23:25:3e:fe
manolo@manolo-Parallels:~/Documentos/SPSI/P3$
```

*Imagen 2: Fichero con la clave privada del usuario “manuel”.*

```

read DSA key
Enter pass phrase for ruizDSApriv.pem:
Private-Key: (1024 bit)
priv:
    48:0e:b1:39:53:47:8c:44:5b:80:b6:12:e6:43:c9:
    0c:a0:c1:fc:8a
pub:
    00:f4:9e:f4:aa:b1:79:82:0a:ba:89:ca:aa:e9:46:
    6c:c8:8e:6f:8a:2f:72:39:56:38:70:f7:c3:b6:5f:
    b4:f1:a0:9d:31:a7:14:87:96:27:8a:87:8d:db:21:
    dd:c1:1d:9a:f5:c2:9f:11:df:7a:c0:9a:0f:22:e5:
    fe:1f:67:be:a9:81:e8:fb:41:0b:92:b2:76:d4:ae:
    f4:30:98:e2:35:ef:de:65:cf:2d:b8:f2:72:00:5a:
    7d:12:b4:54:25:21:d3:cb:be:ae:88:6c:20:84:b4:
    2b:60:60:5f:d1:ac:ef:3c:a3:f5:6c:dc:81:ff:d1:
    0b:c2:f1:27:49:ac:43:c0:11
P:
    00:fc:e1:ac:60:b8:ab:00:ff:d7:bd:53:e7:84:7e:
    69:dc:37:e5:ab:08:9b:4f:29:95:01:52:20:0e:b7:
    92:ad:57:30:1f:30:00:62:f4:60:f9:64:a9:5e:99:
    0f:9e:ae:65:94:eb:f6:69:81:d9:16:59:19:a3:f5:
    79:03:de:f5:8a:d1:1b:32:0c:5e:e8:3d:f0:cf:15:
    f8:34:8e:26:06:80:0e:2d:fb:d2:a1:1e:99:df:02:
    c0:0d:5a:19:44:d1:68:a6:de:ba:da:bb:0c:6d:2a:
    82:a1:59:8f:b2:07:8f:8d:28:88:44:d6:e1:45:f0:
    28:a5:8b:6d:82:b6:da:f5:b7
Q:
    00:9d:2e:bd:c2:7c:db:3e:9c:fc:88:08:dd:aa:96:
    3d:da:c7:6d:10:11
G:
    3f:de:91:68:a1:1e:0c:5f:b1:43:e9:a1:36:51:64:
    e6:12:28:72:06:d1:e5:74:1d:3a:ea:83:09:da:bb:
    0b:3b:25:55:0e:e9:cc:9b:b8:23:e3:12:24:84:04:
    60:40:97:58:52:23:e4:94:65:ce:0b:ef:2b:ee:59:
    00:bd:fe:0f:da:96:1f:f4:ea:ee:3d:32:23:6b:71:
    9c:40:3e:d6:eb:18:17:1f:0e:3a:67:0b:a0:2f:c9:
    56:fd:99:e5:0b:a5:8a:3d:96:53:20:29:5e:94:57:
    2c:2f:2d:91:0d:d5:bd:7f:ff:b0:53:ea:43:18:17:
    c0:d5:14:9b:23:25:3e:fe
manolo@manolo-Parallels:~/Documentos/SPSI/P3$

```

Imagen 3: Fichero con la clave privada del usuario “ruiz”.

## 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.**

Con el mismo comando del ejercicio anterior, el cual permitía manipular claves de DSA, voy a extraer ahora únicamente la parte pública de la clave. La única diferencia que cabe destacar es que para que esto ocurra hay que indicar la opción *-pubout* que permite extraer solo la parte pública.

```

$$ openssl dsa -in manuelDSAkey.pem -out manuelDSAPub.pem -pubout
$$ openssl dsa -in ruizDSAkey.pem -out ruizDSAPub.pem -pubout

```

En este caso como no pide que este cifrado ni protegido no le indico ninguna opción de cifrado.



De nuevo, para mostrar el contenido utilizo las opciones `-text -noout` pero además le añado la opción `-pubin`, para que OpenSSL sepa que el contenido del fichero es una clave pública, ya que por defecto él suele mostrar siempre las privadas.

```
$$ openssl dsa -in manuelDSAPub.pem -text -pubin -noout
$$ openssl dsa -in ruizDSAPub.pem -text -pubin -noout
```

Los valores se pueden ver en las Imágenes 4 y 5.

```
manolo@manolo-Parallels:~/Documentos/SPSI/P3$ openssl dsa -in manuelDSAPub.pem -text -pub
in -noout
read DSA key
pub:
 43:66:a7:5b:d8:53:8a:4b:30:28:c7:68:91:d8:90:
 57:95:d3:18:c0:70:c0:6e:83:b9:5c:91:52:a4:ba:
 b6:69:52:7e:23:4e:56:dd:db:93:4d:37:00:56:3c:
 65:b6:38:b2:23:15:83:aa:42:ad:81:30:5f:5f:16:
 d2:84:2e:cc:11:29:a3:70:ee:46:25:b0:e4:f9:3c:
 40:f5:04:31:c1:a0:28:07:17:1c:18:bf:24:07:c4:
 24:0a:1e:6f:9e:94:44:29:b5:99:a0:dd:56:df:2b:
 80:d8:45:f8:74:47:d1:d7:02:55:df:23:63:e0:fc:
 cf:c8:5c:e3:bb:7c:a2:bf
P:
 00:fc:e1:ac:60:b8:ab:00:ff:d7:bd:53:e7:84:7e:
 69:dc:37:e5:ab:08:9b:4f:29:95:01:52:20:0e:b7:
 92:ad:57:30:1f:30:00:62:f4:60:f9:64:a9:5e:99:
 0f:9e:ae:65:94:eb:f6:69:81:d9:16:59:19:a3:f5:
 79:03:de:f5:8a:d1:1b:32:0c:5e:e8:3d:f0:cf:15:
 f8:34:8e:26:06:80:0e:2d:fb:d2:a1:1e:99:df:02:
 c0:0d:5a:19:44:d1:68:a6:de:ba:da:bb:0c:6d:2a:
 82:a1:59:8f:b2:07:8f:8d:28:88:44:d6:e1:45:f0:
 28:a5:8b:6d:82:b6:da:f5:b7
Q:
 00:9d:2e:bd:c2:7c:db:3e:9c:fc:88:08:dd:aa:96:
 3d:da:c7:6d:10:11
G:
 3f:de:91:68:a1:1e:0c:5f:b1:43:e9:a1:36:51:64:
 e6:12:28:72:06:d1:e5:74:1d:3a:ea:83:09:da:bb:
 0b:3b:25:55:0e:e9:cc:9b:b8:23:e3:12:24:84:04:
 60:40:97:58:52:23:e4:94:65:ce:0b:ef:2b:ee:59:
 00:bd:fe:0f:da:96:1f:f4:ea:ee:3d:32:23:6b:71:
 9c:40:3e:d6:eb:18:17:1f:0e:3a:67:0b:a0:2f:c9:
 56:fd:99:e5:0b:a5:8a:3d:96:53:20:29:5e:94:57:
 2c:2f:2d:91:0d:d5:bd:7f:ff:b0:53:ea:43:18:17:
 c0:d5:14:9b:23:25:3e:fe
manolo@manolo-Parallels:~/Documentos/SPSI/P3$
```

*Imagen 4: Fichero con la clave pública del usuario “manuel”.*

```

manolo@manolo-Parallels:~/Documentos/SPSI/P3$ openssl dsa -in ruizDSAPub.pem -text -pubin
-noout
read DSA key
pub:
 00:f4:9e:f4:aa:b1:79:82:0a:ba:89:ca:aa:e9:46:
 6c:c8:8e:6f:8a:2f:72:39:56:38:70:f7:c3:b6:5f:
 b4:f1:a0:9d:31:a7:14:87:96:27:8a:87:8d:db:21:
 dd:c1:1d:9a:f5:c2:9f:11:df:7a:c0:9a:0f:22:e5:
 fe:1f:67:be:a9:81:e8:fb:41:0b:92:b2:76:d4:ae:
 f4:30:98:e2:35:ef:de:65:cf:2d:b8:f2:72:00:5a:
 7d:12:b4:54:25:21:d3:cb:be:ae:88:6c:20:84:b4:
 2b:60:60:5f:d1:ac:ef:3c:a3:f5:6c:dc:81:ff:d1:
 0b:c2:f1:27:49:ac:43:c0:11
P:
 00:fc:e1:ac:60:b8:ab:00:ff:d7:bd:53:e7:84:7e:
 69:dc:37:e5:ab:08:9b:4f:29:95:01:52:20:0e:b7:
 92:ad:57:30:1f:30:00:62:f4:60:f9:64:a9:5e:99:
 0f:9e:ae:65:94:eb:f6:69:81:d9:16:59:19:a3:f5:
 79:03:de:f5:8a:d1:1b:32:0c:5e:e8:3d:f0:cf:15:
 f8:34:8e:26:06:80:0e:2d:fb:d2:a1:1e:99:df:02:
 c0:0d:5a:19:44:d1:68:a6:de:ba:da:bb:0c:6d:2a:
 82:a1:59:8f:b2:07:8f:8d:28:88:44:d6:e1:45:f0:
 28:a5:8b:6d:82:b6:da:f5:b7
Q:
 00:9d:2e:bd:c2:7c:db:3e:9c:fc:88:08:dd:aa:96:
 3d:da:c7:6d:10:11
G:
 3f:de:91:68:a1:1e:0c:5f:b1:43:e9:a1:36:51:64:
 e6:12:28:72:06:d1:e5:74:1d:3a:ea:83:09:da:bb:
 0b:3b:25:55:0e:e9:cc:9b:b8:23:e3:12:24:84:04:
 60:40:97:58:52:23:e4:94:65:ce:0b:ef:2b:ee:59:
 00:bd:fe:0f:da:96:1f:f4:ea:ee:3d:32:23:6b:71:
 9c:40:3e:d6:eb:18:17:1f:0e:3a:67:0b:a0:2f:c9:
 56:fd:99:e5:0b:a5:8a:3d:96:53:20:29:5e:94:57:
 2c:2f:2d:91:0d:d5:bd:7f:ff:b0:53:ea:43:18:17:
 c0:d5:14:9b:23:25:3e:fe
manolo@manolo-Parallels:~/Documentos/SPSI/P3$

```

Imagen 5: Fichero con la clave pública del usuario “ruiz”.

## 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`.

Las funciones hash permiten generar un valor único correspondiente a un mensaje, de forma que el mensaje sea indescifrable para quien conoce el valor pero sea fácil de calcular dicho valor a partir del propio mensaje. La función hash SHA-384, perteneciente a la familia SHA-2, tiene como salida un valor de 384 bits.

Para calcular el valor hash que se pide en el enunciado voy a utilizar la información recogida en [4]. La orden usada se muestra a continuación:

```
$$ openssl dgst -sha384 -hex -c manuelDSAPub.pem
```

El comando usado es `openssl dgst`, que hace referencia al nombre que se le da en inglés a las funciones hash, “message digest”. Con la opción `-sha384` indicamos que la función utilizada va a ser la correspondiente a la función SHA-384 con salida de 384 bits. Para establecer el formato de



salida como se pide en el enunciado se utiliza la opción `-c`. Finalmente se indica el fichero (o mensaje) al que se le quiere calcular su valor hash. No se ha indicado fichero de salida porque primero pide que se muestre por pantalla su valor.

El resultado obtenido por la salida estándar se puede ver en la Imagen 6.

```
manolo@manolo-Parallels:~/Documentos/SPSI/P3$ openssl dgst -sha384 -hex -c manuelDSAPub.pem
SHA384(manuelDSAPub.pem)= 40:ad:ba:9a:44:ca:77:09:4d:b7:79:7b:8b:e4:0c:bb:37:47:51:4a:f8:
79:e9:1e:19:66:66:74:d7:7c:4d:b4:1d:fe:ed:dc:9b:41:5a:c3:2a:6f:da:8d:42:98:f1:d8
manolo@manolo-Parallels:~/Documentos/SPSI/P3$
```

*Imagen 6: Salida estándar del valor hash del fichero manuelDSAPub.pem.*

Si ahora quiero que el valor se guarde en un fichero solo tengo que añadir a la orden anterior la opción `-out` e indicar el nombre del fichero.

```
$$ openssl dgst -sha384 -hex -out manuelDSAPub.sha384 -c manuelDSAPub.pem
```

Se puede comprobar en la Imagen 7 el valor hash almacenado en un fichero, y además, como debe ser, este valor hash es el mismo que el que se mostraba por pantalla hace un momento.

```
manolo@manolo-Parallels:~/Documentos/SPSI/P3$ cat manuelDSAPub.sha384
SHA384(manuelDSAPub.pem)= 40:ad:ba:9a:44:ca:77:09:4d:b7:79:7b:8b:e4:0c:bb:37:47:
51:4a:f8:79:e9:1e:19:66:66:74:d7:7c:4d:b4:1d:fe:ed:dc:9b:41:5a:c3:2a:6f:da:8d:42
:98:f1:d8
manolo@manolo-Parallels:~/Documentos/SPSI/P3$
```

*Imagen 7: Contenido del fichero que almacena el valor hash del fichero manuelDSAPub.pem.*

## 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.**

En este caso se pide realizar el mismo proceso que en el ejercicio anterior, pero esta vez para el usuario “ruiz”. En lugar de utilizar la función SHA-384 se pide utilizar una cuya salida sea de 160 bits. En teoría hemos visto un par de funciones que tienen esa característica, como son la función SHA-1 y la RIPEMD-160. La orden usada es la siguiente:

```
$$ openssl dgst -sha1 -binary -out ruizDSAPub.sha1 ruizDSAPub.pem
```

Las opciones que acompañan en este caso al comando `openssl dgst` son `-sha1`, que indica la función resumen con la que calcular el valor hash, `-binary` para que la salida sea con formato binario, `-out` para el fichero donde guardar el valor y finalmente el fichero al que calcular el valor hash.

Para mostrar el valor hash en este caso tengo que utilizar alguna forma de leer ficheros binarios, como puede ser por ejemplo el programa `ghex`. El valor hash calculado se puede ver en la Imagen 8.

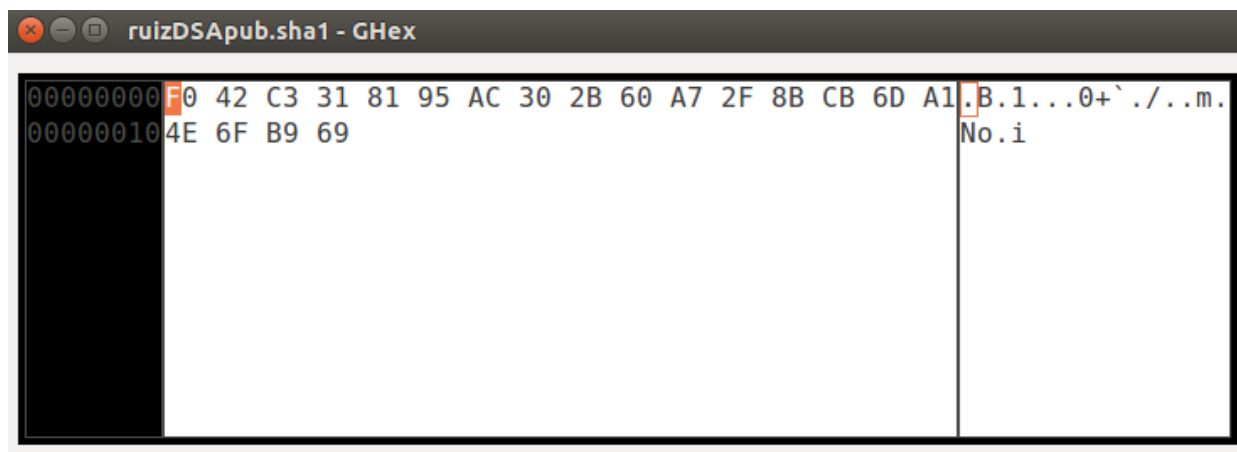


Imagen 8: Valor hash del fichero ruizDSAPub.pem.

## Ejercicio 7

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

El valor HMAC (Código de Autenticación de Mensajes basado en funciones Hash) de un archivo se utiliza para autenticar un mensaje, para conocer el origen del mismo y garantizar también la integridad.

Se puede generar el valor HMAC de un archivo con el comando `openssl dgst` también. Para ello solo hay que indicar que se quiere generar el valor HMAC con la opción `-hmac` acompañando la opción de la clave utilizada para el proceso de generación del valor. En este caso la clave usada ha sido “12345”. La orden usada es la siguiente:

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

El valor HMAC del archivo sharedDSA.pem se puede ver en la Imagen 9.

```
manolo@manolo-Parallels:~/Documentos/SPSI/P3$ openssl dgst -hmac 12345 sharedDSA
.pem
HMAC-SHA256(sharedDSA.pem)= 3c94c8dd501275ac350ea64964fa8e2fcc36bd64b0cc450e0ac1
c73b63059e5d
manolo@manolo-Parallels:~/Documentos/SPSI/P3$
```

Imagen 9: Valor HMAC del fichero sharedDSA.pem.

## 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 estación a estación será AES-128 en modo CFB8.

Inicialmente vamos a suponer que contamos con dos usuarios, Manuel y Ruiz, los cuales tienen cada uno un par de claves pública-privada para firma y verificación. Además, cada uno de ellos dispone de otro par de claves pública-privada de un criptosistema basado en curvas elípticas, generadas a partir de los mismos parámetros.

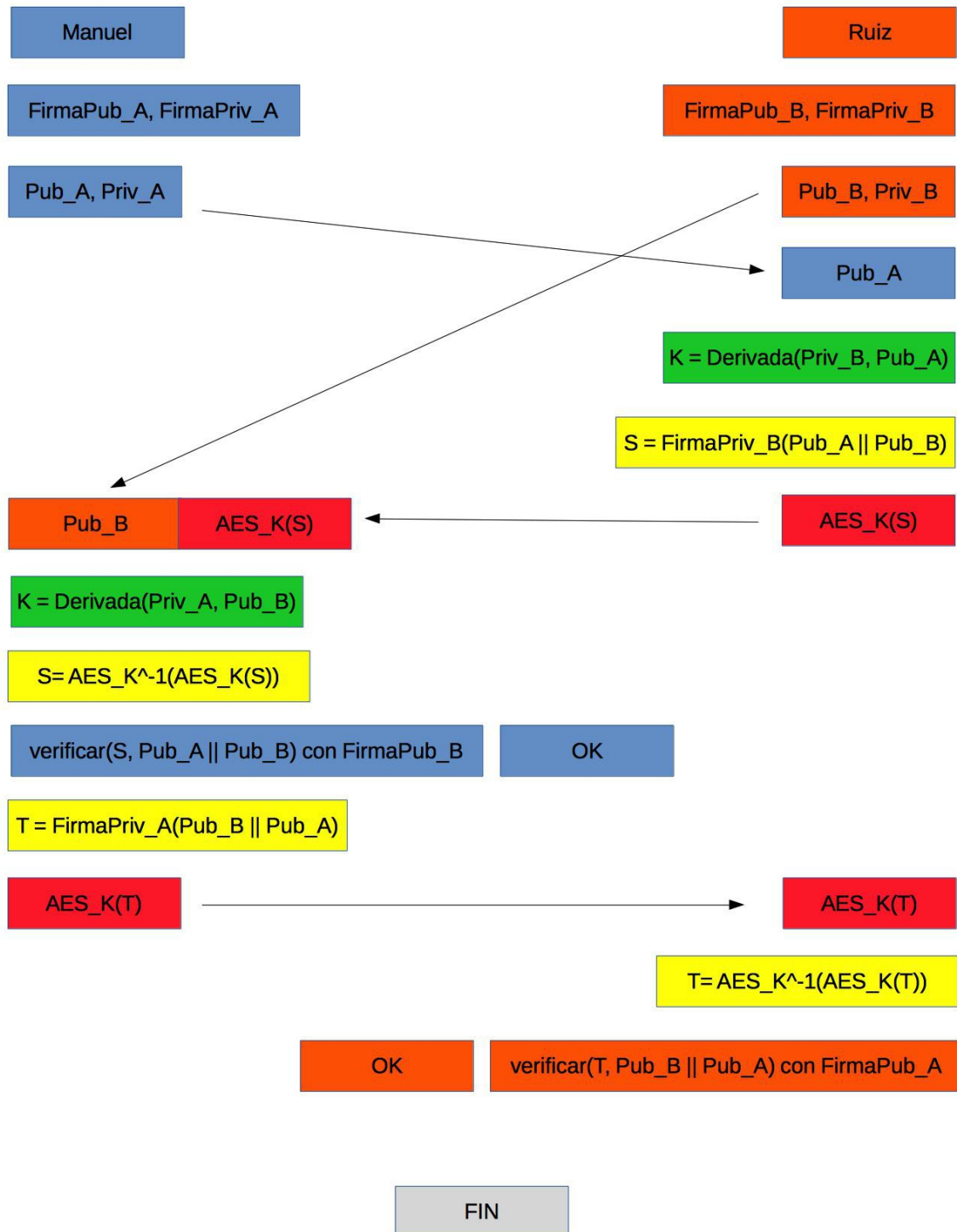


Imagen 10: Explicación gráfica del protocolo estación a estación.

El funcionamiento del protocolo estación a estación es sencillo (ver Imagen 10): Manuel da su clave pública del criptosistema de EC a Ruiz para que éste, junto con su clave privada de EC genere una clave  $k$  que deberá ser común a ambos. Como Ruiz dispone de las dos claves públicas de EC, concatena estos archivos (primero el de Manuel y luego el suyo) y los firma utilizando su clave privada. Una vez hecho esto, Ruiz encripta el fichero que contiene su firma, firmaRuiz, utilizando el algoritmo indicado (AES-128-CFB8) y le envía a Manuel tanto la firma encriptada como su clave pública de EC.

Ahora es el turno de Manuel. Dado que Manuel cuenta con la clave pública de EC de Ruiz, siguiendo el mismo proceso que siguió previamente Ruiz, puede generar la misma clave  $k$  que obtuvo antes Ruiz. Con esta clave y conociendo el método de cifrado que usó Ruiz, Manuel descifra el mensaje que contiene la firma y verifica con la clave pública usada para firmar de Ruiz. Si el proceso de verificación da como resultado OK entonces ha llegado el momento de realizar el proceso inverso.

En este caso es Manuel el encargado de firmar el fichero que contiene las claves públicas pero esta vez en orden inverso. Una vez firmado, lo encripta usando  $k$  y se lo envía a Ruiz. Ruiz, que ya conoce  $k$ , lo desencripta y verifica, con la clave pública para firmar de Manuel, que la firma que se ha obtenido da como resultado OK.

Si en ambos casos se obtiene OK en los procesos de verificación, pueden estar seguro de que la clave secreta  $k$  es conocida solo por ellos dos. A partir de ahí ya pueden enviarse mensaje utilizando esa clave.

A continuación voy a escribir los comandos utilizados para realizar todo lo explicado anteriormente.

Primero voy a generar valores del segundo usuario en EC ya que en la práctica anterior solo se simuló la existencia de un usuario.

```
$$ openssl ecparam -in stdECparam.pem -genkey -out ruizECkey.pem -noout  
$$ openssl ec -in ruizECkey.pem -out ruizECpriv.pem -des3 [redacted]  
$$ openssl ec -in ruizECkey.pem -out ruizECpub.pem -pubout
```

Ahora, aunque se altere un poco el proceso del protocolo voy a suponer que ambos usuarios han enviado al otro su clave pública y que cada uno ha generado por su parte la clave  $K$ .

```
$$ openssl pkeyutl -derive -inkey manuelECpriv.pem -peerkey ruizECpub.pem -out manuelKey.bin  
$$ openssl pkeyutl -derive -inkey ruizECpriv.pem -peerkey manuelECpub.pem -out ruizKey.bin [redacted]
```

El comando *openssl pkeyutl* se usa para realizar tareas de utilidad con claves públicas en algoritmos. Con *-derive* se obtiene un secreto compartido usando el archivo aportado por *-peerkey*. Con *-inkey* se pasa la clave privada del algoritmo y en *-out* se indica el nombre del fichero que contendrá la clave secreta compartida.

Dado que esto es una simulación en un solo ordenador, puedo comprobar que efectivamente las dos claves generadas son iguales, aunque esto no es necesario ya que ambos usuarios sabrían que eso es así después de completar el protocolo.

```
$$ diff manuelKey.bin ruizKey.bin
```

Como se puede ver en la Imagen 11, no aparece que exista ninguna diferencia entre esos dos archivos, por lo que lo voy a llamar key.bin tal y como dice en el enunciado.

```
$$ mv manuelKey.bin key.bin
```

```
manolo@manolo-Parallels:~/Documentos/SPSI/P3$ diff manuelKey.bin ruizKey.bin
manolo@manolo-Parallels:~/Documentos/SPSI/P3$
```

*Imagen 11: Las claves derivadas obtenidas por ambos usuarios son la misma.*

Ahora puedo seguir simulando el protocolo de nuevo tal y como se explicaba en la Imagen 10.

Ruiz concatena las claves públicas de Manuel y Ruiz respectivamente.

```
$$ cat manuelECpub.pem ruizECpub.pem > concatManuel-Ruiz
```

Ruiz firma con su clave privada para firmar con DSA el archivo anterior. La opción *-sign* se utiliza para leer la clave privada utilizada para firmar.

```
$$ openssl dgst -out firmaRuiz -sign ruizDSApriv.pem concatManuel-Ruiz
```

Ruiz encripta el archivo firmado con la clave secreta común K y envía la firma encriptada a Manuel (ya que habíamos supuesto anteriormente que ya se le había enviado a Manuel la clave pública de EC).

```
$$ openssl enc -aes-128-cfb8 -out encripFirmaRuiz -in firmaRuiz -kfile key.bin
```

Ahora es el turno de Manuel. Éste desencripta el archivo de la firma con la clave secreta común K.

```
$$ openssl aes-128-cfb8 -d -in encripFirmaRuiz -out decripFirmaRuiz -kfile key.bin
```

Concatena respectivamente las claves públicas suya y de Ruiz y verifica el valor de la firma de dicho archivo con el desencriptado correspondiente a la firma de Ruiz. El comando *openssl dgst* tiene la opción *-verify* a la cual se le pasa el valor público de la clave con la que se supone que se generó la firma y con *-signature* el valor con el que se quiere comprobar si corresponde la firma.

```
$$ cat manuelECpub.pem ruizECpub.pem > concatManuel-Ruiz-2
$$ openssl dgst -verify ruizDSAPub.pem -signature decripFirmaRuiz concatManuel-Ruiz-2
```

En la Imagen 12 se puede comprobar que el proceso de verificación ha dado como resultado el primer OK.

```
manolo@manolo-Parallels:~/Documentos/SPSI/P3$ openssl dgst -verify ruizDSAPub.pe
m -signature decripFirmaRuiz concatManuel-Ruiz-2
Verified OK
manolo@manolo-Parallels:~/Documentos/SPSI/P3$
```

*Imagen 12: Resultado tras realizar la verificación el primer usuario.*

Ahora toca entrar a la segunda parte del protocolo. Manuel concatena las claves públicas esta vez en orden inverso, primero la de Ruiz y luego la suya.

```
$$ cat ruizECpub.pem manuelECpub.pem > concatRuiz-Manuel
```

Lo firma con su clave de DSA privada.

```
$$ openssl dgst -out firmaManuel -sign manuelDSApriv.pem concatRuiz-Manuel
```

Envía a Ruiz el encriptado de la firma.

```
$$ openssl enc -aes-128-cfb8 -out encripFirmaManuel -in firmaManuel -kfile key.bin
```

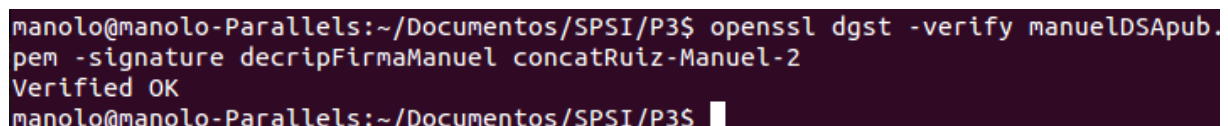
Ruiz descripta la firma.

```
$$ openssl aes-128-cfb8 -d -in encripFirmaManuel -out decripFirmaManuel -kfile key.bin
```

Ruiz verifica lo que ha descriptado con la firma de la concatenación de las claves suya y de Manuel en ese orden con la clave de DSA publica de Manuel.

```
$$ cat ruizECpub.pem manuelECpub.pem > concatRuiz-Manuel-2  
$ openssl dgst -verify manuelDSAPub.pem -signature decripFirmaManuel concatRuiz-Manuel-2
```

En la Imagen 13 se puede comprobar que el resultado de la verificación ha sido correcto también.



```
manolo@manolo-Parallels:~/Documentos/SPSI/P3$ openssl dgst -verify manuelDSAPub.  
pem -signature decripFirmaManuel concatRuiz-Manuel-2  
Verified OK  
manolo@manolo-Parallels:~/Documentos/SPSI/P3$
```

*Imagen 13: Resultado tras realizar la verificación el segundo usuario.*

Así, ambos usuarios saben que la clave secreta K está compartida sólo y exclusivamente por ellos.

## Conclusión

Esta práctica me ha servido para darme cuenta de la importancia tanto del DSA como de las funciones hash usadas para firmar. Estos métodos nos permiten dotar de propiedades como la autenticidad y el no repudio a criptosistemas que de otro modo carecen de ellas. Creo que son una buena forma de complementar los sistemas más potentes de cifrado que carecen de estas propiedades.



## Índice de imágenes

Imagen 1: Conjunto de parámetros generados para utilizar en DSA.....	3
Imagen 2: Fichero con la clave privada del usuario “manuel” .....	5
Imagen 3: Fichero con la clave privada del usuario “ruiz” .....	6
Imagen 4: Fichero con la clave pública del usuario “manuel” .....	7
Imagen 5: Fichero con la clave pública del usuario “ruiz” .....	8
Imagen 6: Salida estándar del valor hash del fichero manuelDSAPub.pem.....	9
Imagen 7: Contenido del fichero que almacena el valor hash del fichero manuelDSAPub.pem.....	9
Imagen 8: Valor hash del fichero ruizDSAPub.pem.....	10
Imagen 9: Valor HMAC del fichero sharedDSA.pem.....	10
Imagen 10: Explicación gráfica del protocolo estación a estación.....	11
Imagen 11: Las claves derivadas obtenidas por ambos usuarios son la misma.....	13
Imagen 12: Resultado tras realizar la verificación el primer usuario.....	13
Imagen 13: Resultado tras realizar la verificación el segundo usuario.....	14

## Bibliografía

- [1] Documentación en OpenSSL para generar parámetros del protocolo DSA.  
<https://www.openssl.org/docs/man1.0.2/apps/dsaparam.html>
- [2] Documentación en OpenSSL para generar claves para DSA a partir de los parámetros.  
<https://www.openssl.org/docs/man1.0.2/apps/gendsa.html>
- [3] Documentación en OpenSSL para procesamiento de claves en DSA.  
<https://www.openssl.org/docs/man1.0.2/apps/dsa.html>
- [4] Documentación en OpenSSL para el uso de funciones resumen de mensajes.  
<https://www.openssl.org/docs/man1.0.2/apps/dgst.html>