

Práctica 3. Protocolos criptográficos

Patricia Maldonado Mancilla

Índice

1. (0,5 puntos) Generad un archivo sharedDSA.pem que contenga los parámetros. Mostrad los valores. 6
2. (0,5 puntos) Generad dos parejas de claves para los parámetros anteriores. Las claves se almacenarán en los archivos <nombre>DSAkey.pem y <apellido>DSAkey.pem. No es necesario protegerlas por contraseña. 7
3. (0,5 puntos) 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. Haced lo mismo para el archivo apellidoDSAkey.pem. 8
4. Extraed en <nombre>DSAPub.pem la clave pública contenida en el archivo <nombre>DSAkey.pem. De nuevo <nombre>DSAPub.pem no debe estar cifrado ni protegido. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem 12
5. Coged un archivo cualquiera cualquiera, que actuará como entrada, con al menos 128 bytes. En adelante me referiré a él como message, pero podéis llamarlo como os parezca. 14
6. (0,5 puntos) Firmad directamente el archivo message empleando el comando openssl pkeyutl sin calcular valores hash, la firma deberá almacenarse en un archivo llamado, por ejemplo, message.sign. Mostrad el archivo con la firma. 15
7. (1 punto) Construid un archivo message2 diferente de message tal que la verificación de la firma message.sign sea correcta con respecto al archivo message2. 16
8. (0,5 puntos) 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 17
9. (0,5 puntos) Calculad el valor hash de message2 usando una función hash de 160 bits con salida binaria. Guardad el hash en message2 algoritmo y mostrad su contenido. 17
10. (0,5 puntos) Firmad el archivo message2 mediante el comando openssl dgst y la función hash del punto anterior. La firma deberá almacenarse en un archivo llamado, por ejemplo, message2.sign. 18

- 11.(1 punto) Verificad la firma message2.sign con los archivos message y message2 empleando el comando openssl dgst. 19
- 12.(0,5 puntos) Verificad que message2.sign es una firma correcta para message2 pero empleando el comando openssl pkeyutl 19
- 13.(0,5 puntos) Generad el valor HMAC del archivo sharedDSA.pem con clave '12345' mostrándolo por pantalla. 20
- 14.(3 puntos) 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 -outkey.bin El algoritmo simétrico a utilizar en el protocolo estación a estación será AES-128 en modo CFB8 21

Índice de figuras

1.1. Comando para generar parámetros DSA	6
1.2. Contenido parámetros DSA	6
1.3. Contenido parámetros DSA	7
2.1. Comando para generar las parejas de claves en archivo patriciaDSAkey.pem y maldonadoDSAkey.pem	7
2.2. Contenido patriciaDSAkey.pem	8
2.3. Contenido maldonadoDSAkey.pem	8
3.1. Comando para extraer la clave privada del archivo patriciaDSAkey.pem .	9
3.2. Comando para extraer la clave privada del archivo maldonadoDSAkey.pem	9
3.3. Contenido clave privada DSA en patriciaDSApriv.pem	10
3.4. Contenido clave privada DSA en maldonadoDSApriv.pem	11
4.1. Comandos para extraer clave pública DSA	12
4.2. Contenido clave pública DSA en patriciaDSAPub.pem	13
4.3. Contenido clave pública DSA en maldonadoDSAPub.pem	14
5.1. Contenido message.bin	15
6.1. Comando para firmar el archivo message	15
6.2. Contenido de message.sign firmado	16
7.1. Contenido message2	16
7.2. firma	16
8.1. Comando para calcular el valor hash usando sha384 mostrando su contenido	17
9.1. Comando para calcular el valor hash usando sha1	17
9.2. Contenido de message2.sha1	18
10.1. Comando dgst para firmar message2.bin con sha1	18
10.2. Contenido message2 firmado	18
11.1. Comando dgst para verificar firma message2.sign con el archivo messa- ge.bin	19
11.2. Comando dgst para verificar firma message2 con el archivo message2.bin .	19
12.1. Comando pkeyutl para verificar firma message2	19
13.1. Comando dgst para generar hmac de sharedDSA.pem	20
14.1. Comando para generar la clave EC maldonadoECkey	21
14.2. Comandos para la extracción de la parte pública y privada de maldona- doECkey.pem	21
14.3. Comando para generar patriciakey.bin	22
14.4. Comando para generar maldonadokey.bin	22
14.5. Contenido de patriciakey.bin y maldonadokey.bin	22
14.6. Comando para concatenar las claves públicas	22
14.7. Contenido de las claves públicas concatenadas	23
14.8. Firma patricia de patricia-maldonado-concat	23
14.9. Encriptación de la firma patricia	23
14.10Firma patricia descifrada	23
14.11Concatenación patriciaECpub.pem y maldonadoEcpub.pem	24
14.12Comando de verificación	24

14.13Concatenación inversa	24
14.14Firma maldonado	24
14.15Encriptación de la firma maldonado	24
14.16Concatenación maldonadoEcpub.pem y patriciaEcpub.pem	25
14.17Descifrado del archivo signMaldonado	25
14.18Comando para la verificación	25

1. (0,5 puntos) Generad un archivo sharedDSA.pem que contenga los parámetros. Mostrad los valores.

Vamos a generar un archivo con los parámetros DSA con el siguiente comando:

```
patri@patri:~/Escritorio$ openssl dsaparam -out sharedDSA.pem 901
Generating DSA parameters, 901 bit long prime
This could take some time
.....+.....*
.....+. ....+. ...+. ..+. .....+. ....+. .+. ....+. ..+++++
+++++
```

Figura 1.1: Comando para generar parámetros DSA

Con **openssl dsparam** podemos generar y manipular los parámetros asociados (primos p, q y el generador g) y parejas de claves privadas/públicas. Se guardan los parámetros mediante el argumento `-out` en un archivo `sharedDSA.pem` y con un tamaño de 901 bits.

```
1 |-----BEGIN DSA PARAMETERS-----
2 |MIIBDAJSAImjxItI1fLQBTNk2ALYysnwIA+j2FPCNHNxmyfEHL27COgUa9KHU9jD
3 |guUjLSKHAC97JOqaJRYNcMlk95/TrYh96NdLZTfvcBzGskOPNEN/wAQ4g8v3T2+M
4 |Gohhkh3qh/fZ/NcAQvcUGmFgk4C8INajGLWQABC7XwIVAP98y8EGbwlg/4n5ENkV
5 |kZUhHS/zAnglyBo1mrpzF5LAcc9F4c9ANZo37gR/MSstI00q+0ImGrmx5/37boZY
6 |ljJlVYjcGGeZFhL0qpW SacBXDLawNAHJGX0yULW/iFc0HBMaCGjjpbGUD8eIwbvx
7 |qMmGri9uX2KxyUZmrhDQ7k+/TOAa+zNe2k4zbYBag18=
8 |-----END DSA PARAMETERS-----
```

Figura 1.2: Contenido parámetros DSA

Para verlo de forma más legible utilizamos el comando **openssl dsparam** indicándole el archivo de entrada `sharedDSA.pem` y con las opciones `-text` y `-noout`. `-noout` que no produce salida del archivo y `-text` imprime los parámetros DSA en forma legible por humanos.

```
patri@patri:~/Escritorio$ openssl dsaparam -in sharedDSA.pem -text -noout
P:
 00:89:a3:c4:8b:48:d4:59:50:05:33:64:d8:02:d8:
 ca:c9:f0:88:0f:a3:d8:53:c2:34:73:71:9b:27:c4:
 1c:bd:bb:08:e8:14:6b:d2:87:53:d8:c3:82:e5:23:
 2d:22:87:01:cf:7b:24:ea:9a:25:16:0d:70:c9:64:
 f7:9f:d3:ad:88:7d:e8:d7:4b:65:37:ef:70:1c:c6:
 b0:a3:8f:34:43:7f:c0:04:38:83:cb:f7:4f:6f:8c:
 1a:88:61:92:1d:ea:87:f7:d9:fc:d7:00:42:f7:14:
 1a:61:60:93:80:bc:20:d6:a3:18:b5:90:00:17:3b:
 5f
Q:
 00:ff:7c:cb:c1:06:6d:69:60:ff:89:f9:10:d9:15:
 91:95:21:1d:2f:f3
G:
 25:60:1a:35:9a:ba:73:17:92:c0:71:cf:45:e1:cf:
 40:35:9a:37:ee:04:7f:31:2b:2d:23:43:aa:fb:42:
 26:1a:b9:b1:e7:fd:fb:6e:86:58:96:38:e5:55:88:
 dc:18:67:99:16:12:ce:aa:9c:12:69:c0:57:0e:56:
 96:34:01:c9:19:73:b2:50:b5:bf:88:57:34:1c:13:
 1a:08:68:e3:a5:b1:94:0f:c7:88:59:bb:f1:a8:c9:
 86:ae:2f:6e:5f:62:b1:c9:46:66:ae:10:d0:ee:4f:
 bf:4c:e0:1a:fb:33:5e:da:4e:33:6d:80:5a:83:5f
```

Figura 1.3: Contenido parámetros DSA

2. (0,5 puntos) Generad dos parejas de claves para los parámetros anteriores. La claves se almacenarán en los archivos <nombre>DSAkey.pem y <apellido>DSAkey.pem. No es necesario protegerlas por contraseña.

Para generar dos parejas de claves con los parámetros anteriores usamos los siguientes comandos:

```
patri@patri:~/Escritorio$ openssl gendsa -out patriciaDSAkey.pem sharedDSA.pem
Generating DSA key, 960 bits
patri@patri:~/Escritorio$ openssl gendsa -out maldonadoDSAkey.pem sharedDSA.pem
Generating DSA key, 960 bits
```

Figura 2.1: Comando para generar las parejas de claves en archivo patriciaDSAkey.pem y maldonadoDSAkey.pem

El comando **gendsa** genera una clave privada DSA a partir de un archivo de parámetros

DSA. En este caso hemos generado un par de claves en `patriciaDSAkey.pem` y `maldonadoDSAkey.pem` a partir de archivo creado anteriormente con los parámetros llamado `sharedDSA.pem`.

```
*patriciaDSAkey.pem x
1 |-----BEGIN DSA PRIVATE KEY-----
2 MIIBoAIBAAJSAImjXItI1FlQBTnK2ALYysnwiA+j2FPCNHnxmyFEHL27C0gUa9KH
3 U9jDguUjLSKHAc97J0qaJRYNcMlk95/TrYh96NdLZTfvcBzGsKOPNEN/wAQ4g8v3
4 T2+MGohhkh3qh/fZ/NcAQvcUGmFgk4C8INajGLWQABc7XwIVAP98y8EGbWlg/4n5
5 ENkVkJZUHS/zAngLYBo1mrpzF5LAcc9F4c9ANZo37gR/MSstI00q+0ImGrmx5/37
6 boZYLjJlVYjcGGeZFhL0qpW5acBXDLawNAHJGXOyULW/iFc0HBMAcGjjpbGUD8eI
7 WbvXqMmGri9uX2KxyUZmrhDQ7k+/TOAa+zNe2k4zbYBag18CeEz107AkqEUaeHMj
8 4l9TJ6YmWxh1Ihw8zUn3005Ebk5Jyz1yp2eXXH85TBjwu1V5iIceSR8+0EgBo6t/
9 MzGievNG8z2af8tigrWbo1VFG1n+P4nugvUCzN0uEFnrY1cZL30DtS332qcPDLkI
10 UdHhjr8u8cxzpdCZXAIVALSGefgfggqhIz2wEgGG+Iugbu5h
11 -----END DSA PRIVATE KEY-----,
```

Figura 2.2: Contenido `patriciaDSAkey.pem`

```
maldonadoDSAkey.pem x
1 |-----BEGIN DSA PRIVATE KEY-----
2 MIIBnwIBAAJSAImjXItI1FlQBTnK2ALYysnwiA+j2FPCNHnxmyFEHL27C0gUa9KH
3 U9jDguUjLSKHAc97J0qaJRYNcMlk95/TrYh96NdLZTfvcBzGsKOPNEN/wAQ4g8v3
4 T2+MGohhkh3qh/fZ/NcAQvcUGmFgk4C8INajGLWQABc7XwIVAP98y8EGbWlg/4n5
5 ENkVkJZUHS/zAngLYBo1mrpzF5LAcc9F4c9ANZo37gR/MSstI00q+0ImGrmx5/37
6 boZYLjJlVYjcGGeZFhL0qpW5acBXDLawNAHJGXOyULW/iFc0HBMAcGjjpbGUD8eI
7 WbvXqMmGri9uX2KxyUZmrhDQ7k+/TOAa+zNe2k4zbYBag18CeByXdvawP9D1t08H
8 LGAAAtqmRbcBQvqVVDwhoG6/uQo1pZaIMrQzh18ab7TzyXg8cGz6gFh1ZqQD50n8
9 Iz0jWzX/gZPV7JcZEYMDqGeaHWUSejPLLY9g/Q0//De09Q4jYngk6BhfUyvl70YH
10 PA1pz+YYQ+/B9hDa/gIufHaorWipxMP0+8DEm9j41EHNRFs=
11 -----END DSA PRIVATE KEY-----
```

Figura 2.3: Contenido `maldonadoDSAkey.pem`

3. (0,5 puntos) Extraed la clave privada contenida en el archivo `nombreDSAkey.pem` a otro archivo que tenga por nombre `nombreDSApriv.pem`. Este archivo deber a estar protegido por contraseña. Mostrad sus valores. Haced lo mismo para el archivo `apellidoDSAkey.pem`.

Para extraer la clave privada de los archivos `patriciaDSAkey.pem` y `maldonadoDSAkey.pem` vamos a usar el comando `openssl dsa` que sirve para manipular claves generadas. Además de extraer la parte privada también con este comando se pueden realizar conversiones de formato, extraer la parte pública, adición o sustracción de contraseñas, etc.


```
patri@patri:~/Escritorio$ openssl dsa -in patriciaDSAkey.pem -out patriciaDSAkeypriv.pem -aes128
read DSA key
writing DSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Figura 3.1: Comando para extraer la clave privada del archivo patriciaDSAkey.pem

```
patri@patri:~/Escritorio$ openssl dsa -in maldonadoDSAkey.pem -out maldonadoDSAkeypriv.pem -aes128
read DSA key
writing DSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Figura 3.2: Comando para extraer la clave privada del archivo maldonadoDSAkey.pem

Mediante la opción **-in** le especificamos el archivo de entrada (patriciaDSAkey.pem y apellidoDSAkey.pem en cada caso) para leer la clave privada. Y cada uno con la opción **-out** genera un archivo de salida con la clave privada llamados patricisaDSAkeypriv.pem y maldonadoDSAkeypriv.pem.

Además protegemos cada fichero cifrando la clave privada con **aes128**, la contraseña que se ha introducido ha sido la usada en la práctica anterior: 0123456789.

A continuación se muestra el contenido extraído de la clave privada. Para cada fichero se ha visualizado mediante gedit y para visualizarlo en una forma más legible he utilizado el comando usado anteriormente.

```
patriciaDSAkeypriv.pem x
1 |-----BEGIN DSA PRIVATE KEY-----
2 Proc-Type: 4,ENCRYPTED
3 DEK-Info: AES-128-CBC,2A4BCE290E151460B06DD231E229AA81
4
5 k+1bMbvuXfb/fX8h0WwLXes7DTMVEPQ/Q9V/jw2NYtWEQ44nUC0XAWgd5dJdsWze
6 CZLvCWB9V4zx0V84F2odwYSF7VuoWzQHm2atQzcQLAAx2xpXfDCB5po+YgypTErK
7 3lCsw1PQGF/8ZVE9yVnbVI2iw+cAn/4ZhALgWuvc/vGN6kbXBUEjiQ7yVWA9q7Wd
8 YMBwvEIIULrERL2SRP+X9sy52GtWew2UQGdTnJaiosSQ7Cs209pPz0ihrSVw5IBX
9 Ivc6KXZ9swIjEnwwV3Y46u5/hLOWoNW2LasyyC2pVbNrWywiYrMSolIspQ6GQGLx
10 svj+aY3r8DaFPuxZ1JbeT62ivZjjKAMPaRBGytdZaWI035dtTVyt+PiIvuh3M1vt
11 RvzlEmePuBxVpqrJAXFDeno7VPXvV0gR6KKEoSr/jtWFUsTK8vqALx8LHrwrC/00
12 Y3xR4zH1XT4bfS0xfF0UtirwhCuKuqLPG+XLJZE02vx42d6ns2cbCUJq17Pww1C0
13 0NBFjg6d6Eva0V2pPhqD/bGU2pDmN8/Rs7c20eV6k6t/Ko5tZU0eq+Q2vJ1mu1QCc
14 -----END DSA PRIVATE KEY-----

patri@patri:~/Escritorio$ openssl dsa -in patriciaDSAkey.pem -noout -text
read DSA key
Private-Key: (960 bit)
priv:
  00:b4:86:79:f8:1f:82:0a:a1:23:3d:b0:12:01:86:
  f8:8b:a0:6e:ee:61
pub:
  4c:f5:d3:b0:24:a8:45:1a:78:73:23:e2:5f:53:27:
  a6:26:c1:78:75:22:1c:3c:cd:49:f7:d3:4e:44:6e:
  4e:49:cb:3d:72:a7:67:97:5c:7f:39:4c:18:f0:bb:
  55:79:88:87:1e:49:1f:3e:d0:48:01:a3:ab:7f:33:
  31:a2:7a:f3:46:f3:3d:9a:7f:cb:62:82:b5:9b:a3:
  55:45:1b:59:fe:3f:89:ee:82:f5:02:cc:d3:ae:10:
  59:eb:63:57:19:2f:73:83:b5:2d:f7:da:a0:8f:0e:
  59:08:51:d1:e1:8e:bf:2e:f1:cc:73:a5:d0:99:5c
P:
  00:89:a3:c4:8b:48:d4:59:50:05:33:64:d8:02:d8:
  ca:c9:f0:88:0f:a3:d8:53:c2:34:73:71:9b:27:c4:
  1c:bd:bb:08:e8:14:6b:d2:87:53:d8:c3:82:e5:23:
  2d:22:87:01:cf:7b:24:ea:9a:25:16:0d:70:c9:64:
  f7:9f:d3:ad:88:7d:e8:d7:4b:65:37:ef:70:1c:c6:
  b0:a3:8f:34:43:7f:c0:04:38:83:cb:f7:4f:6f:8c:
  1a:88:61:92:1d:ea:87:f7:d9:fc:d7:00:42:f7:14:
  1a:61:60:93:80:bc:20:d6:a3:18:b5:90:00:17:3b:
  5f
Q:
  00:ff:7c:cb:c1:06:6d:69:60:ff:89:f9:10:d9:15:
  91:95:21:1d:2f:f3
G:
  25:60:1a:35:9a:ba:73:17:92:c0:71:cf:45:e1:cf:
  40:35:9a:37:ee:04:7f:31:2b:2d:23:43:aa:fb:42:
  26:1a:b9:b1:e7:fd:fb:6e:86:58:96:38:e5:55:88:
  dc:18:67:99:16:12:ce:aa:9c:12:69:c0:57:0e:56:
  96:34:01:c9:19:73:b2:50:b5:bf:88:57:34:1c:13:
  1a:08:68:e3:a5:b1:94:0f:c7:88:59:bb:f1:a8:c9:
  86:ae:2f:6e:5f:62:b1:c9:46:66:ae:10:d0:ee:4f:
  bf:4c:e0:1a:fb:33:5e:da:4e:33:6d:80:5a:83:5f
```

Figura 3.3: Contenido clave privada DSA en patriciaDSApriv.pem

```
maldonadoDSAkeypriv.pem x
1 |-----BEGIN DSA PRIVATE KEY-----
2 Proc-Type: 4, ENCRYPTED
3 DEK-Info: AES-128-CBC,A0D08A5850130578C8D8A692DF635B78
4
5 35hNf0/LuC8/09kP8sD0pJgxADBjZiHtunCZTEeI1PKUEVg8RMKqE47STLS9qoHg
6 ILJY3NsIO0wD75Xvt5LqyUCwTP4pNOFWXAV8ZLFPe0GIVg9tCuQ4ayQtujTaEFM1
7 CWFQEub1P4YBZxzFa1+ZF/Qsf/cQl3wKz1hgNS3kKYM2sETTWRyA/jZ3zJAp3lvZ
8 XZrxevSHzBpRrHS09hVQCFH64ajv33fNuc0FpTqeYqgbWNSjXbsz1dq6jy0WQPJK
9 FqBdoEdfYjw4FXnGX/C5xYbkaobLDyQNe/aGfS/arnNvpdyGtHKQ442DhL0DpRD
10 T06kzv04QwFB2AZsUp9RL1ZfhUaYF0JP5Whm4/JKa7XBQGABLA5JSEFWMDi9jXkL
11 5bGNQgLDLoQH+ciORWyCvFwgekQ07NP710QFHkPI7ZIKyfcLE3W1e7HGvOXFHNb
12 bgGrCb1Eq9g8c98yDE0BB5XkfBjFgSSYSpZFP3Xsx0FZ76oHGduUttLZcN/KVlfo
13 rtd89vBnMv10yTS1thVpa2PeGge+yGY6XNHAz3Abu4zTHJCjmyz6m7RqE2vH1YCh
14 -----END DSA PRIVATE KEY-----

patri@patri:~/Escritorio$ openssl dsa -in maldonadoDSAkey.pem -noout -text
read DSA key
Private-Key: (960 bit)
priv:
    7c:76:a8:ad:68:a9:c4:c3:f4:fb:c0:c4:9b:d8:f8:
    d4:41:cd:44:5b
pub:
    1c:97:76:f6:b0:3f:d0:f5:b4:ef:07:2c:60:1a:02:
    da:a6:45:b7:1b:42:fa:95:54:3c:21:a0:6e:bf:b9:
    0a:35:a5:96:88:32:b4:33:86:5f:1a:6f:b4:f3:c9:
    78:3c:70:6c:fa:80:58:75:66:a4:03:48:e9:fc:23:
    3d:23:5b:35:ff:81:93:d5:ec:97:19:11:83:1d:a8:
    67:9a:1d:65:12:7a:33:cb:2d:8f:60:fd:0d:3f:fc:
    37:8e:f5:0e:23:62:78:24:07:a8:5f:53:2b:cb:ec:
    e6:07:3c:0d:69:cf:e6:18:43:ef:c1:f6:10:da:fe
P:
    00:89:a3:c4:8b:48:d4:59:50:05:33:64:d8:02:d8:
    ca:c9:f0:88:0f:a3:d8:53:c2:34:73:71:9b:27:c4:
    1c:bd:bb:08:e8:14:6b:d2:87:53:d8:c3:82:e5:23:
    2d:22:87:01:cf:7b:24:ea:9a:25:16:0d:70:c9:64:
    f7:9f:d3:ad:88:7d:e8:d7:4b:65:37:ef:70:1c:c6:
    b0:a3:8f:34:43:7f:c0:04:38:83:cb:f7:4f:6f:8c:
    1a:88:61:92:1d:ea:87:f7:d9:fc:d7:00:42:f7:14:
    1a:61:60:93:80:bc:20:d6:a3:18:b5:90:00:17:3b:
    5f
Q:
    00:ff:7c:cb:c1:06:6d:69:60:ff:89:f9:10:d9:15:
    91:95:21:1d:2f:f3
G:
    25:60:1a:35:9a:ba:73:17:92:c0:71:cf:45:e1:cf:
    40:35:9a:37:ee:04:7f:31:2b:2d:23:43:aa:fb:42:
    26:1a:b9:b1:e7:fd:fb:6e:86:58:96:38:e5:55:88:
    dc:18:67:99:16:12:ce:aa:9c:12:69:c0:57:0e:56:
    96:34:01:c9:19:73:b2:50:b5:bf:88:57:34:1c:13:
    1a:08:68:e3:a5:b1:94:0f:c7:88:59:bb:f1:a8:c9:
    86:ae:2f:6e:5f:62:b1:c9:46:66:ae:10:d0:ee:4f:
    bf:4c:e0:1a:fb:33:5e:da:4e:33:6d:80:5a:83:5f
```

Figura 3.4: Contenido clave privada DSA en maldonadoDSApriv.pem

4. Extraed en `<nombre>DSAPub.pem` la clave pública contenida en el archivo `<nombre>DSAkey.pem`. De nuevo `<nombre>DSAPub.pem` no debe estar cifrado ni protegido. Mostrad sus valores. Lo mismo para el archivo `apellidoDSAkey.pem`

Para extraer la clave pública del archivo `patriciaDSAkey.pem` y de `maldonadoDSAkey.pem` utilizamos el comando **openssl dsa**. Pero en este caso añadimos la opción **-pubout** para indicar que queremos extraer la parte pública. Le pasamos mediante **-in** el archivo del que queremos extraer la parte pública y lo guardamos mediante el argumento **-out**.

```
patri@patri:~/Escritorio$ openssl dsa -in patriciaDSAkey.pem -out patriciaDSAPub
.pem -pubout
read DSA key
writing DSA key
patri@patri:~/Escritorio$ openssl dsa -in maldonadoDSAkey.pem -out maldonadoDSAP
ub.pem -pubout
read DSA key
writing DSA key
```

Figura 4.1: Comandos para extraer clave pública DSA

A continuación podemos visualizar el contenido de la extracción de la clave pública en los archivos de salida `patriciaDSAPub.pem` y `maldonadoDSAPub.pem`.

```

patriciaDSAPub.pem x
1 |-----BEGIN PUBLIC KEY-----
2 MIIBmJCCARkGBYqGSM44BAEwggEMAnkAiaPEi0jUWVAFM2TYAtjKyfCID6PYU8I0
3 c3GbJ8QcvbsI6BRr0odT2MOC5SMtIocBz3sk6polFg1wyWT3n90tiH3o10tLN+9w
4 HMawo480Q3/ABDiDy/dPb4waiGGSHeqH99n81wBC9xQaYwCTgLwg1qMYtZAAFztf
5 AhUA/3zLwQZtaWd/IfkQ2RWRLSEdL/MCeCVgGjWaunMXksBxz0Xhz0A1mjfuBH8x
6 Ky0jQ6r7QiYaubHn/ftuhliW00VViNwYZ5kWEs6qnBJpwFc0VpY0AckZc7JQtB+I
7 VzQcExoIa00lsZQPX4hZu/GoyYauL25fYrHJRmauENDuT79M4Br7M17aTjNtgFqD
8 XwN7AAJ4TPXTsCSorRp4cyPiX1MnpibBeHUiHDzNSffTTkRuTknLPXKnZ5dcfzLM
9 GPC7VxmIhx5JHz7QSAGjq38zMaJ680bzPZp/y2KctZujVUUbWf4/ie6C9QLM064Q
10 WetjVxkvc401LffaoI80WQhR0eG0vy7xzH0l0Jlc
11 -----END PUBLIC KEY-----

patri@patri:~/Escritorio$ openssl dsa -pubin -in patriciaDSAPub.pem -text -noout

read DSA key
pub:
  4c:f5:d3:b0:24:a8:45:1a:78:73:23:e2:5f:53:27:
  a6:26:c1:78:75:22:1c:3c:cd:49:f7:d3:4e:44:6e:
  4e:49:cb:3d:72:a7:67:97:5c:7f:39:4c:18:f0:bb:
  55:79:88:87:1e:49:1f:3e:d0:48:01:a3:ab:7f:33:
  31:a2:7a:f3:46:f3:3d:9a:7f:cb:62:82:b5:9b:a3:
  55:45:1b:59:fe:3f:89:ee:82:f5:02:cc:d3:ae:10:
  59:eb:63:57:19:2f:73:83:b5:2d:f7:da:a0:8f:0e:
  59:08:51:d1:e1:8e:bf:2e:f1:cc:73:a5:d0:99:5c

P:
  00:89:a3:c4:8b:48:d4:59:50:05:33:64:d8:02:d8:
  ca:c9:f0:88:0f:a3:d8:53:c2:34:73:71:9b:27:c4:
  1c:bd:bb:08:e8:14:6b:d2:87:53:d8:c3:82:e5:23:
  2d:22:87:01:cf:7b:24:ea:9a:25:16:0d:70:c9:64:
  f7:9f:d3:ad:88:7d:e8:d7:4b:65:37:ef:70:1c:c6:
  b0:a3:8f:34:43:7f:c0:04:38:83:cb:f7:4f:6f:8c:
  1a:88:61:92:1d:ea:87:f7:d9:fc:d7:00:42:f7:14:
  1a:61:60:93:80:bc:20:d6:a3:18:b5:90:00:17:3b:
  5f

Q:
  00:ff:7c:cb:c1:06:6d:69:60:ff:89:f9:10:d9:15:
  91:95:21:1d:2f:f3

G:
  25:60:1a:35:9a:ba:73:17:92:c0:71:cf:45:e1:cf:
  40:35:9a:37:ee:04:7f:31:2b:2d:23:43:aa:fb:42:
  26:1a:b9:b1:e7:fd:fb:6e:86:58:96:38:e5:55:88:
  dc:18:67:99:16:12:ce:aa:9c:12:69:c0:57:0e:56:
  96:34:01:c9:19:73:b2:50:b5:bf:88:57:34:1c:13:
  1a:08:68:e3:a5:b1:94:0f:c7:88:59:bb:f1:a8:c9:
  86:ae:2f:6e:5f:62:b1:c9:46:66:ae:10:d0:ee:4f:
  bf:4c:e0:1a:fb:33:5e:da:4e:33:6d:80:5a:83:5f

```

Figura 4.2: Contenido clave pública DSA en patriciaDSAPub.pem

```
maldonadoDSAPub.pem x
1 |-----BEGIN PUBLIC KEY-----
2 MIIBmjCCARKGBYqGSM44BAEwggEMAnkAiaPEi0jUWVAFM2TYAtjKyfCID6PYU8I0
3 c3GbJ8QcvbsI6BRr0odT2MOC5SMtIocBz3sk6polFg1wyWT3n90tiH3o10tLN+9w
4 HMawo480Q3/ABDiDy/dPb4waiGGSheqH99n81wBC9xQaYwCTgLwg1qMYtZAAFztf
5 AhUA/3zLwQZtaWD/ifkQ2RWRLSEdL/MCeCVgGjWaunMXksBxz0Xhz0A1mjfuBH8x
6 Ky0jQ6r7QiYaubHn/ftuhliW00VViNwYZ5kWEs6qnBJpwFcoVPY0AckZc7JQtB+I
7 VzQcExoIa00lsZQPX4hZu/GoyYauL25fYrHJRmauENDuT79M4Br7M17aTjNtgFqD
8 Xwn7AAJ4HJd29rA/0PW07wcsYBoC2qZFtxtc+pVUPCGgbr+5CjWllogytD0GXxpV
9 tPPJeDxwbPqAWHVmpANI6fwjPSNbNf+Bk9XslxkRgx2oZ5odZRJ6M8stj2D9DT/8
10 N471DiNieCQHqF9TK8vs5gc8DwnP5hhD78H2ENr+
11 -----END PUBLIC KEY-----

patri@patri:~/Escritorio$ openssl dsa -pubin -in maldonadoDSAPub.pem -text -noout
read DSA key
pub:
1c:97:76:f6:b0:3f:d0:f5:b4:ef:07:2c:60:1a:02:
da:a6:45:b7:1b:42:fa:95:54:3c:21:a0:6e:bf:b9:
0a:35:a5:96:88:32:b4:33:86:5f:1a:6f:b4:f3:c9:
78:3c:70:6c:fa:80:58:75:66:a4:03:48:e9:fc:23:
3d:23:5b:35:ff:81:93:d5:ec:97:19:11:83:1d:a8:
67:9a:1d:65:12:7a:33:cb:2d:8f:60:fd:0d:3f:fc:
37:8e:f5:0e:23:62:78:24:07:a8:5f:53:2b:cb:ec:
e6:07:3c:0d:69:cf:e6:18:43:ef:c1:f6:10:da:fe

P:
00:89:a3:c4:8b:48:d4:59:50:05:33:64:d8:02:d8:
ca:c9:f0:88:0f:a3:d8:53:c2:34:73:71:9b:27:c4:
1c:bd:bb:08:e8:14:6b:d2:87:53:d8:c3:82:e5:23:
2d:22:87:01:cf:7b:24:ea:9a:25:16:0d:70:c9:64:
f7:9f:d3:ad:88:7d:e8:d7:4b:65:37:ef:70:1c:c6:
b0:a3:8f:34:43:7f:c0:04:38:83:cb:f7:4f:6f:8c:
1a:88:61:92:1d:ea:87:f7:d9:fc:d7:00:42:f7:14:
1a:61:60:93:80:bc:20:d6:a3:18:b5:90:00:17:3b:
5f

Q:
00:ff:7c:cb:c1:06:6d:69:60:ff:89:f9:10:d9:15:
91:95:21:1d:2f:f3

G:
25:60:1a:35:9a:ba:73:17:92:c0:71:cf:45:e1:cf:
40:35:9a:37:ee:04:7f:31:2b:2d:23:43:aa:fb:42:
26:1a:b9:b1:e7:fd:fb:6e:86:58:96:38:e5:55:88:
dc:18:67:99:16:12:ce:aa:9c:12:69:c0:57:0e:56:
96:34:01:c9:19:73:b2:50:b5:bf:88:57:34:1c:13:
1a:08:68:e3:a5:b1:94:0f:c7:88:59:bb:f1:a8:c9:
86:ae:2f:6e:5f:62:b1:c9:46:66:ae:10:d0:ee:4f:
bf:4c:e0:1a:fb:33:5e:da:4e:33:6d:80:5a:83:5f
```

Figura 4.3: Contenido clave pública DSA en maldonadoDSAPub.pem

5. Coged un archivo cualquiera cualquiera, que actuar a como entrada, con al menos 128 bytes. En adelante me referir e a el como message, pero podéis llamarlo como os parezca.

He utilizado el archivo input.bin relleno de 0 de 128 bytes(1024 bits) y lo he renombrado como message.bin

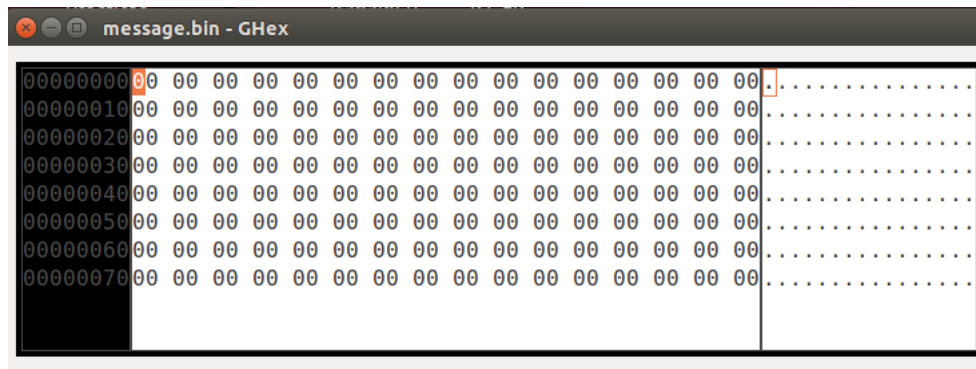


Figura 5.1: Contenido message.bin

6. (0,5 puntos) Firmad directamente el archivo message empleando el comando openssl pkeyutl sin calcular valores hash, la firma deber a almacenarse en un archivo llamado, por ejemplo, message.sign. Mostrad el archivo con la firma.

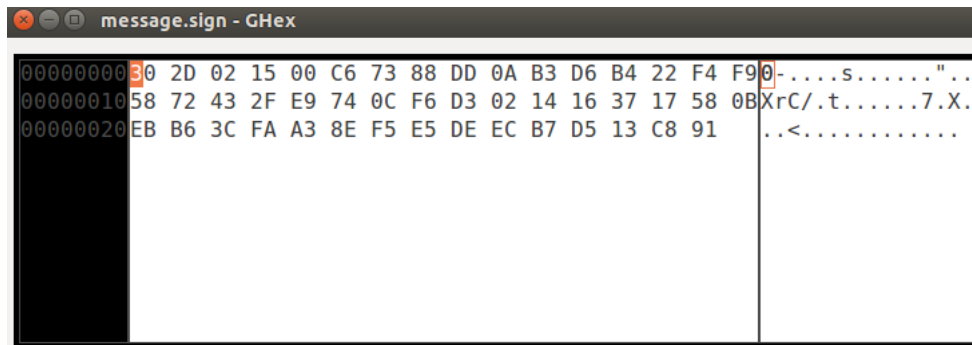
Con el comando **openssl pkeyutl** podemos realizar tareas de firma y verificación En este caso vamos a firmar con el siguiente comando:

```
patri@patri:~/Escritorio$ openssl pkeyutl -sign -in message.bin -inkey patriciaD
SAkeypriv.pem -out message.sign
Enter pass phrase for patriciaDSAkeypriv.pem:
```

Figura 6.1: Comando para firmar el archivo message

Los argumentos que se han utilizado son los siguientes:

- **-sign:** para firmar los datos de entrada y enviar el resultado firmado. Esto requiere una clave privada.
- **-in:** archivo de que le pasamos para firmar.
- **-inkey:** archivo de clave de entrada, que debe de ser una clave privada.
- **-out:** archivo de salida firmado.



7. (1 punto) Construid un archivo `message2` diferente de `message` tal que la verificación de la firma `message.sign` sea correcta con respecto al archivo `message2`.

Para el archivo message2 me he basado en editar message.bin añadiendo una serie de bits más, de modo que este archivo ahora contiene 160 bytes (1312 bits).

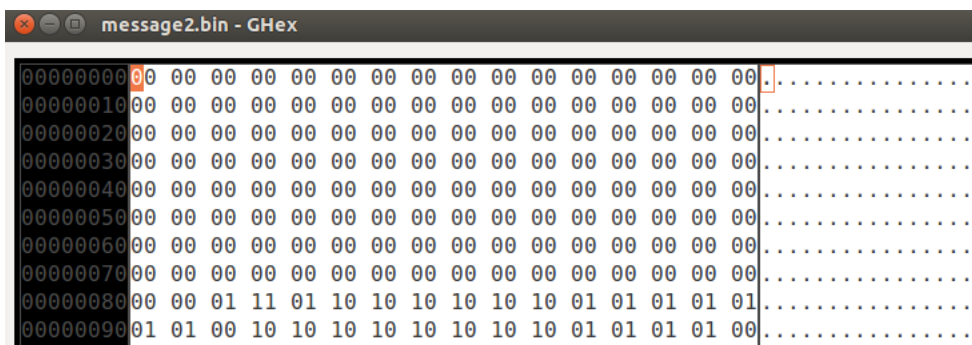


Figura 7.1: Contenido message2

Para verificar la firma usamos el siguiente comando:

```
patri@patri:~/Escritorio$ openssl pkeyutl -verify -in message2.bin -sigfile message.sign -pubin -inkey patriciaDSAPub.pem
Signature Verified Successfully
```

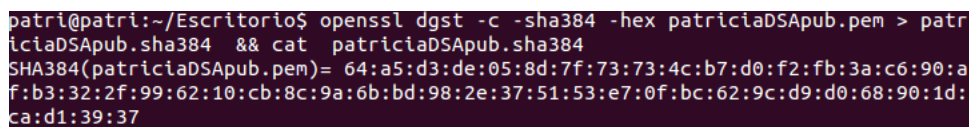
Figura 7.2: firma

Para verificar la firma tenemos que añadir el argumento **-pubin** para indicar que la verificación se realice con la clave pública que le pasamos mediante el argumento **-inkey**. Además usamos el argumento **-verify** para indicar que queremos verificar la firma y muestra si se realizó correctamente y con **-sigfile** le pasamos el archivo con la firma.

Podemos comprobar que la firma se ha verificado correctamente. Lo normal sería que una firma creada para un archivo no fuese válida para otro diferente. Con pkeyutil cuando recibe un archivo más grande lo trunca al tamaño del hash calculado. Por lo tanto al recibir para verificar el archivo message2 trunca al tamaño del hash e interpreta que la firma message.sign es válida para message2.

8. . (0,5 puntos) 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 **openssl dgst** permite generar valores hash en hexadecimal y binario, códigos de autenticación de mensajes (HMAC) y firmar digitalmente un valor hash. En este caso vamos a calcular el valor hash usando la función sha384.



```
patri@patri:~/Escritorio$ openssl dgst -c -sha384 -hex patriciaDSAPub.pem > patriciaDSAPub.sha384 && cat patriciaDSAPub.sha384
SHA384(patriciaDSAPub.pem)= 64:a5:d3:de:05:8d:7f:73:73:4c:b7:d0:f2:fb:3a:c6:90:a
f:b3:32:2f:99:62:10:cb:8c:9a:6b:bd:98:2e:37:51:53:e7:0f:bc:62:9c:d9:d0:68:90:1d:
ca:d1:39:37
```

Figura 8.1: Comando para calcular el valor hash usando sha384 mostrando su contenido

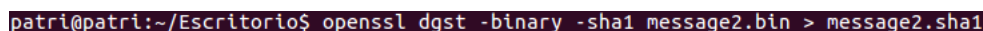
Usamos el comando **openssl dgst** indicando las opciones:

- **-c:** imprime en bloques de dos caracteres separados por dos puntos, solo es relevante si se utiliza la salida en formato hexadecimal.
- **-sha384:** función hash de 384 bits.
- **-hex:** para mostrar la salida en formato hexadecimal

Además para guardar la salida utilizamos **>** y lo mostramos a la vez con **cat**.

9. (0,5 puntos) Calculad el valor hash de message2 usando una función hash de 160 bits con salida binaria. Guardad el hash en message2 algoritmo y mostrad su contenido.

Para calcular una función hash con 160 bits utilizamos sha1.



```
patri@patri:~/Escritorio$ openssl dgst -binary -sha1 message2.bin > message2.sha1
```

Figura 9.1: Comando para calcular el valor hash usando sha1

Los argumentos usados para este comando **openssl dgst** son los siguientes:

- **-sha1**: función hash con 160 bits.
- **-binary**: para que la salida se muestre en formato binario.

La salida se almacenará en message2.sha1

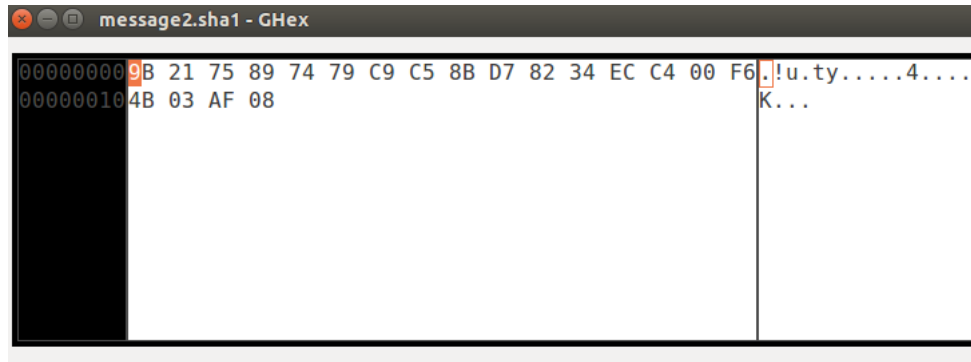


Figura 9.2: Contenido de message2.sha1

10. (0,5 puntos) Firmad el archivo message2 mediante el comando **openssl dgst** y la función hash del punto anterior. La firma deberá almacenarse en un archivo llamado, por ejemplo, message2.sign.

```
patri@patri:~/Escritorio$ openssl dgst -sha1 -sign patriciaDSAkeypriv.pem -out message2.sign message2.bin
```

Figura 10.1: Comando dgst para firmar message2.bin con sha1

A continuación se muestra message2.bin firmado guardado en message2.sign.

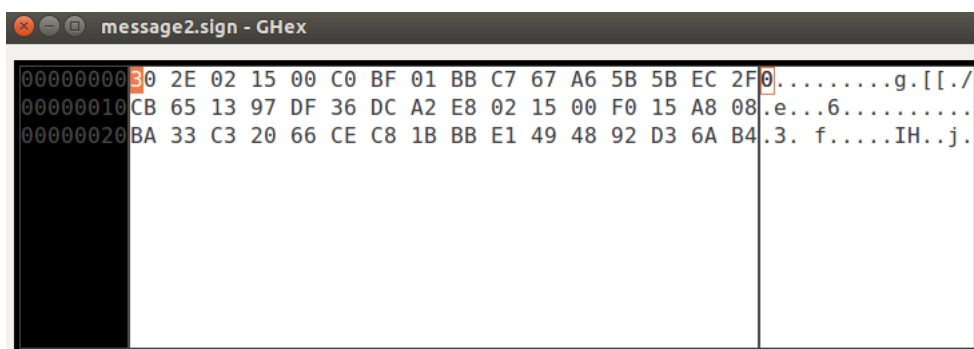


Figura 10.2: Contenido message2 firmado

11. (1 punto) Verificad la firma message2.sign con los archivos message y message2 empleando el comando openssl dgst.

La firma message2.sign es correcta para el archivo message2.bin

```
patri@patri:~/Escritorio$ openssl dgst -sha1 -verify patriciaDSAPub.pem -signature message2.sign message2.bin
Verified OK
```

Figura 11.1: Comando dgst para verificar firma message2.sign con el archivo message2.bin

Los argumentos que necesitamos para verificar la firma con dgst son los siguientes:

- **-sha1**: función hash de 160 bits. Ya que nuestra firma se ha realizado mediante esta función.
- **-verify**: para verificar la firma, e indicará si la firma es correcta o si ha habido un error en la verificación.
- **-signature**: para pasar la firma.

```
patri@patri:~/Escritorio$ openssl dgst -sha1 -verify patriciaDSAPub.pem -signature message2.sign message2.bin
Verification Failure
```

Figura 11.2: Comando dgst para verificar firma message2 con el archivo message2.bin

Podemos comprobar que en este caso la verificación no se ha realizado correctamente ya que dgst calcula el hash del archivo y lo comprueba para un archivo que no es el correspondiente a la firma con ese hash.

12. (0,5 puntos) Verificad que message2.sign es una firma correcta para message2 pero empleando el comando openssl pkeyutl

Para verificar con el comando **openssl pkeyutl** debemos pasarle el archivo al que le hemos calculado el hash llamado message2.sha1 para que la verificación de la firma se realice correctamente.

```
patri@patri:~/Escritorio$ openssl pkeyutl -verify -in message2.sha1 -sigfile message2.sign -pubin -inkey patriciaDSAPub.pem
Signature Verified Successfully
```

Figura 12.1: Comando pkeyutl para verificar firma message2

Los argumentos son los siguientes:

- **-in:** archivo de entrada que en este caso es el que contiene el hash message2.sha1.
- **-sigfile:** para pasarle el archivo con la firma.
- **-pubin:** indicamos que queremos usar la clave pública
- **-inkey:** le pasamos la clave pública para verificar la firma
- **-verify:** para verificar los datos de entrada contra el archivo de firma e indicar si la firma se ha realizado correctamente.

Comprobamos que efectivamente la firma es correcta.

13. (0,5 puntos) Generad el valor HMAC del archivo sharedDSA.pem con clave '12345' mostrándolo por pantalla.

Un código de autenticación de mensajes en clave-hash (HMAC) es una construcción específica para calcular un código de autenticación de mensaje (MAC) que implica una función hash criptográfica en combinación con una llave criptográfica secreta. Usamos el comando dgst con la opción **-hmac** al que le pasamos la clave "1234".

```
patri@patri:~/Escritorio$ openssl dgst -hmac '12345' sharedDSA.pem
HMAC-SHA256(sharedDSA.pem)= f9c47495945c22dea6970192abcc12ccd2875c6d43d5491debd3
7be35d296ce6
```

Figura 13.1: Comando dgst para generar hmac de sharedDSA.pem

14. (3 puntos) 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 sería `openssl pkeyutl -inkey javierECpriv.pem -peerkey lobilloECpub.pem -derive -outkey.bin`. El algoritmo simétrico a utilizar en el protocolo estación a estación será `AES-128` en modo `CFB8`.

Para simular el protocolo estación a estación necesitamos dos usuarios, `patricia` y `maldonado`. Cada uno tiene sus pares de claves pública y privada para la firma y verificación. Además necesitaremos ficheros creados en la práctica anterior.

Primero necesitamos generar una pareja de claves EC (curvas elípticas) del usuario que nos falta que se llamará `maldonado`, donde almacenamos los parámetros creados en `stdECparam.pem`.

```
patri@patri:~/Escritorio$ openssl ecparam -genkey -in stdECparam.pem -out maldonadoECkey.pem
patri@patri:~/Escritorio$
```

Figura 14.1: Comando para generar la clave EC `maldonadoECkey`

A continuación extraemos la pareja de claves privada y pública de `maldonadoECkey.pem` para el usuario `maldonado`.

```
patri@patri:~/Escritorio$ openssl ec -in maldonadoECkey.pem -out maldonadoECpriv
.pem -aes128
read EC key
writing EC key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
patri@patri:~/Escritorio$ openssl ec -pubout -in maldonadoECkey.pem -out maldonadoECpub.pem
read EC key
writing EC key
```

Figura 14.2: Comandos para la extracción de la parte pública y privada de `maldonadoECkey.pem`

Ahora que tenemos los archivos necesarios, realizaremos los siguientes pasos:

1. El usuario patricia calcula la clave común en el fichero patriciakey.bin y el usuario maldonado realiza la misma acción y se guarda en el fichero maldonadokey.bin. Usamos el comando **openssl pkeyutl** con los argumentos: **-derive** para derivar un secreto compartido usando el archivo que le pasamos con el argumento **-peerkey**, con este argumento le pasamos el archivo con la clave pública y con **-inkey** la clave privada. Mediante **-out** indicamos el fichero de salida donde se almacenará la clave secreta compartida. De esta forma conseguimos que cada uno haya enviado al otro su clave pública y además que han generado su clave K.

```
patri@patri:~/Escritorio$ openssl pkeyutl -derive -inkey patriciaECpriv.pem -peerkey maldonadoECPub.pem -out patriciakey.bin
```

Figura 14.3: Comando para generar patriciakey.bin

```
patri@patri:~/Escritorio$ openssl pkeyutl -derive -inkey maldonadoECpriv.pem -peerkey patriciaECPub.pem -out maldonadokey.bin
Enter pass phrase for maldonadoECpriv.pem:
```

Figura 14.4: Comando para generar maldonadokey.bin

Podemos comprobar que las claves generadas patriciakey.bin y maldonadokey.bin son idénticas.

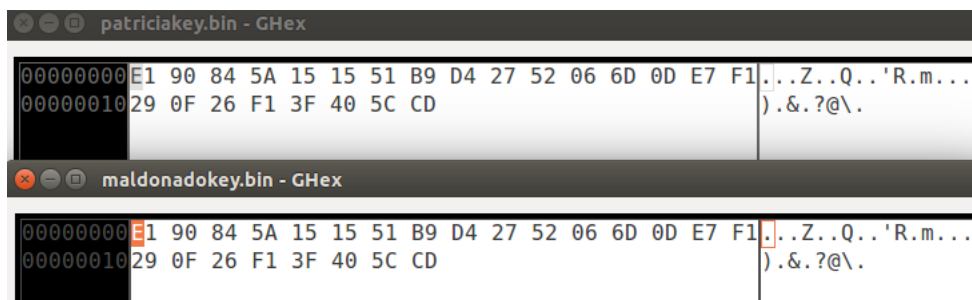


Figura 14.5: Contenido de patriciakey.bin y maldonadokey.bin

2. El usuario patricia concatenará su clave pública generada a partir de la curva elíptica común con la clave pública del usuario maldonado en un archivo llamado patricia-maldonado-concat.pem.

```
patri@patri:~/Escritorio$ cat patriciaECPub.pem maldonadoECPub.pem > patricia-maldonado-concat.pem
```

Figura 14.6: Comando para concatenar las claves públicas

```
patri@patri:~/Escritorio$ cat patricia-maldonado-concat.pem
-----BEGIN PUBLIC KEY-----
MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEyH1F0977mo/laY3kn6uNewrXHPTd
HqX6fI36xaBkLCMZtmRLDZg+JEixMJXjPLYG
-----END PUBLIC KEY-----
-----BEGIN PUBLIC KEY-----
MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAE+FxieWhvyAXex8mw8viTCBtNIAoF
rkadLwjFgpavCx51E1BtDxIgTn0kHKJ4e4QY
-----END PUBLIC KEY-----
```

Figura 14.7: Contenido de las claves públicas concatenadas

3. El usuario patricia firmará con la función hash `-sha1` con su clave privada, el fichero que hemos concatenado anteriormente `patricia-maldonado-concat.pem`, este archivo se cifrará en el algoritmo `aes-128` modo `cfb8`. Para ello usamos el comando **openssl dgst**, usando la opción **-sign** para firmar, firmaremos con la clave privada de `patriciaDSAkeypriv.pem` y lo guardaremos mediante el argumento **-out** en el archivo `signPatricia.sha1`

```
patri@patri:~/Escritorio$ openssl dgst -sha1 -sign patriciaDSAkeypriv.pem -out signPatricia.sha1 patricia-maldonado-concat.pem
Enter pass phrase for patriciaDSAkeypriv.pem:
```

Figura 14.8: Firma patricia de `patricia-maldonado-concat`

4. El usuario patricia también cifra el archivo que contiene la firma con el algoritmo `aes-128` en modo `CFB8` y envía al usuario maldonado la firma encriptada y la clave pública EC. Usamos el siguiente comando:

```
patri@patri:~/Escritorio$ openssl enc -aes-128-cfb8 -pass file:patriciakey.bin -in signPatricia.sha1 -out signPatricia_enc.bin
```

Figura 14.9: Encriptación de la firma patricia

5. Cuando el usuario maldonado ha recibido el fichero con la firma encriptada `signPatricia_enc.bin`, verificará que el archivo que ha recibido procede realmente del usuario patricia. Para realizar la verificación primero tendrá que descifrar el archivo, utilizando la clave común que hemos creado anteriormente.

```
patri@patri:~/Escritorio$ openssl aes-128-cfb8 -d -in signPatricia_enc.bin -out signPatricia_dec.sha1 -pass file:maldonadokey.bin
```

Figura 14.10: Firma patricia descifrada

6. El usuario maldonado procederá a realizar la verificación, pero para ello concatenará los archivos que contienen la clave pública de ambos en el mismo sentido en el que lo realizó el usuario patricia.

```
patri@patri:~/Escritorio$ cat patriciaECPub.pem maldonadoECPub.pem > patricia-maldonado-concat2.pem
```

Figura 14.11: Concatenación patriciaECPub.pem y maldonadoECPub.pem

Para realizar la verificación usaremos el siguiente comando:

```
patri@patri:~/Escritorio$ openssl dgst -sha1 -verify patriciaDSAPub.pem -signature signPatricia_dec.sha1 patricia-maldonado-concat2.pem
Verified OK
```

Figura 14.12: Comando de verificación

La verificación es correcta, por lo tanto el usuario maldonado tiene garantizado que el mensaje enviado efectivamente ha sido por parte del usuario patricia.

- Ahora el usuario maldonado procederá a realizar el mismo proceso realizado por maldonado. Concatenaremos los archivos que contienen las curvas elípticas de ambos pero en este caso, en sentido contrario.

```
patri@patri:~/Escritorio$ cat maldonadoECPub.pem patriciaECPub.pem > maldonado-patricia-concat.pem
```

Figura 14.13: Concatenación inversa

- El usuario maldonado firmará con la función hash -sha1 con su clave privada DSA, el fichero que hemos concatenado anteriormente maldonado-patricia-concat.pem, este archivo se cifrará en el algoritmo aes-128 modo cfb8. Para ello usamos el comando openssl dgst, usando la opción **-sign** para firmar, firmaremos con la clave privada de maldonadoDSAkeypriv.pem y lo guardaremos mediante el argumento **-out** en el archivo signMaldonado.sha1

```
patri@patri:~/Escritorio$ openssl dgst -sha1 -sign maldonadoDSAkeypriv.pem -out signMaldonado.sha1 maldonado-patricia-concat.pem
Enter pass phrase for maldonadoDSAkeypriv.pem:
```

Figura 14.14: Firma maldonado

- Ahora procede a cifrar el archivo con la firma al igual que hizo el usuario patricia, con el mismo algoritmo y modo (aes-128-cfb8).

```
patri@patri:~/Escritorio$ openssl enc -aes-128-cfb8 -pass file:maldonadokey.bin -in signMaldonado.sha1 -out signMaldonado_enc.bin
```

Figura 14.15: Encriptación de la firma maldonado

- Ahora es el turno de que el usuario patricia verifique que le ha enviado el mensaje el usuario maldonado. Para ello volveremos a concatenar las claves públicas. Además de descifrar la firma y por último verificar.


```
patri@patri:~/Escritorio$ cat maldonadoEcpub.pem patriciaEcpub.pem > maldonado-patricia-concat2.pem
```

Figura 14.16: Concatenación maldonadoEcpub.pem y patriciaEcpub.pem

11. Descifra el archivo firmado signMaldonado.

```
patri@patri:~/Escritorio$ openssl aes-128-cfb8 -d -in signMaldonado_enc.bin -out signMaldonado_enc.sha1 -pass file:patriciakey.bin
```

Figura 14.17: Descifrado del archivo signMaldonado

12. Y por último procede a verificar la firma.

```
patri@patri:~/Escritorio$ openssl dgst -sha1 -verify maldonadoDSAPub.pem -signature signMaldonado.sha1 maldonado-patricia-concat2.pem
Verified OK
```

Figura 14.18: Comando para la verificación

Comprobamos que se ha realizado la verificación correctamente, como en el caso anterior. Por lo tanto el usuario patricia y maldonado están seguros de que la comunicación se ha realizado correctamente. En los dos casos hemos obtenido que la verificación ha sido correcta, ya están seguros de que la clave secreta solo es conocida por ellos dos.