

Practica 3

1. Generar un archivo sharedDSA.pem que contenga los parámetros. Mostrar los valores:

Utilizaremos el comando "dsaparam" que nos permite generar y manipular parámetros asociados al algoritmo DSA: primos p y q y el generador g .

```
openssl dsaparam -out sharedDSA.pem 901
```

Así se generará el resultado con un tamaño de 901 bits.

La consola nos mostrará algo como esto:

```
~/MyGitHub/SPSI/practica3 on master! © 22:57:26  
$ openssl dsaparam -out sharedDSA.pem 901  
Generating DSA parameters, 901 bit long prime  
This could take some time  
.....++++++  
+..+.~.+...+.~.+...+.~.+...+.~.+...+.~.+...+.~.+...+.~.+...+++++  
+.....+
```

Los valores obtenidos en hexadecimal tras generar el archivo son estos:

```
~/MyGitHub/SPSI/practica3 on  $\angle$  master!  $\odot$  23:00:18
$ openssl dsaparam -in sharedDSA.pem -text -noout
P:
00:da:7f:d0:03:eb:e1:75:81:c7:5f:81:0b:97:78:
79:e8:fb:d2:04:27:7c:2a:86:36:9c:02:ba:26:ba:
9a:aa:36:c3:08:6f:c0:9a:e6:11:49:44:f2:c5:40:
78:97:51:2e:5f:4f:d5:6e:9d:d2:00:fe:27:bd:97:
4f:d4:ae:a2:93:7e:63:c2:7d:58:34:6b:bc:c6:68:
fc:38:a5:39:b2:23:07:7f:9a:c2:b7:81:dc:39:97:
d4:79:a8:bf:e3:44:5a:ab:06:b3:b2:74:df:3e:33:
2c:3c:39:c7:24:12:6b:7b:02:32:d8:e9:6e:c0:c7:
a7
Q:
00:9a:f9:74:f5:0e:83:df:90:93:08:a0:88:7d:4c:
bc:76:a5:90:ad:85
G:
38:08:1a:43:0c:cd:4d:65:2f:1d:b7:9c:98:f5:74:
26:12:e5:a9:77:77:b9:0a:33:11:a8:bd:e2:bb:3c:
5d:da:9c:89:48:c3:c2:35:97:c4:e3:85:fc:16:2a:
64:d9:f4:11:6c:cb:4f:25:2b:a3:87:d2:d5:04:5c:
c9:f0:b7:34:1f:f8:31:db:d9:24:fe:99:38:58:79:
90:14:47:78:ca:33:06:6f:ab:07:cb:2f:98:e9:77:
7d:4b:0f:30:12:a2:4e:ce:2c:e6:ef:8d:9f:73:72:
6b:17:f1:13:2e:17:64:3c:39:a8:7b:6c:f7:03:ab
```

2. Generar dos parejas de claves para los parámetros anteriores. Las claves se almacenarán en los archivos DSAkey.pem y DSAkey.pem. No es necesario protegerlas con contraseña.

Para generar la pareja de claves necesitamos del archivo anterior (sharedDSA.pem) el cual actuará como parámetro. Esta vez utilizaremos el comando de openssl **gendsa**.

En primer lugar generamos el par para Javier:

```
openssl gensa -out JavierDSAkey.pem sharedDSA.pem
```

Por último, las generamos para Galera:

```
openssl gensa -out GaleraDSAkey.pem sharedDSA.pem
```

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 23:17:31
$ openssl gendsa -out JavierDSAkey.pem sharedDSA.pem
Generating DSA key, 960 bits

~/MyGitHub/SPSI/practica3 on ◀ master! @ 23:17:47
$ openssl gendsa -out GaleraDSAkey.pem sharedDSA.pem
Generating DSA key, 960 bits
```

Ya tenemos generadas las dos parejas de claves.

3."Extraer" la clave privada contenida en el archivo DSAkey.pem a otro archivo que tenga por nombre DSApriv.pem. Este archivo deberá estar protegido por contraseña. Mostrad sus valores. Haced lo mismo para el archivo DSAkey.pem

Para realizarlo utilizaremos el comando de openssl **dsa**. El cual nos permitirá sustraer la clave public y/o privada. Además utilizaremos como cifrado simétrico AES-128 y como contraseña: 0123456789

De esta forma extraeremos y cifraremos la clave privada de Javier:

```
openssl dsa -in JavierDSAkey.pem -out JavierDSApriv.pem -aes128
```

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 23:34:53
$ openssl dsa -in JavierDSAkey.pem -out JavierDSApriv.pem -aes128
read DSA key
writing DSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Del mismo modo para Galera:

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 23:35:05
$ openssl dsa -in GaleraDSAkey.pem -out GaleraDSApriv.pem -aes128
read DSA key
writing DSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Por último mostraremos el resultado de dichas claves, a través del comando:

```
openssl dsa -in <nombre>DSApriv.pem -text -noout
```

De **Javier**:

```
~/MyGitHub/SPSI/practica3 on ✖ master! @ 23:35:31
$ openssl dsa -in JavierDSApriv.pem -text -noout
read DSA key
Enter pass phrase for JavierDSApriv.pem:
Private-Key: (960 bit)
priv:
  48:5e:ba:e9:42:ce:4d:55:0c:82:e8:22:87:ed:54:
  32:3c:e7:8c:6b
pub:
  39:cf:34:a3:e7:58:49:da:c8:6e:31:52:69:08:07:
  59:1c:1a:7e:7d:ce:df:35:1c:0a:8c:f1:6b:ba:05:
  ba:1f:1d:6e:c4:09:36:cc:7a:e4:f1:00:f2:d9:d9:
  4b:69:88:23:31:b7:82:f2:39:4a:23:47:b6:ec:60:
  82:ed:53:f1:d5:11:fd:40:98:ed:85:fd:75:ab:35:
  a8:19:05:f4:4a:dd:33:c5:57:87:4a:e8:15:d1:f5:
  1a:c4:7b:9c:11:db:cb:e9:db:97:9a:49:61:a6:74:
  19:1c:ff:eb:d8:bd:57:18:3e:90:ed:97:52:58:af
P:
  00:da:7f:d0:03:eb:e1:75:81:c7:5f:81:0b:97:78:
  79:e8:fb:d2:04:27:7c:2a:86:36:9c:02:ba:26:ba:
  9a:aa:36:c3:08:6f:c0:9a:e6:11:49:44:f2:c5:40:
  78:97:51:2e:5f:4f:d5:6e:9d:d2:00:fe:27:bd:97:
  4f:d4:ae:a2:93:7e:63:c2:7d:58:34:6b:bc:c6:68:
  fc:38:a5:39:b2:23:07:7f:9a:c2:b7:81:dc:39:97:
  d4:79:a8:bf:e3:44:5a:ab:06:b3:b2:74:df:3e:33:
```

De **Galera**:


```
~/MyGitHub/SPSI/practica3 on ◀ master! ⌚ 23:39:40
$ openssl dsa -in GaleraDSApriv.pem -text -noout
read DSA key
Enter pass phrase for GaleraDSApriv.pem:
Private-Key: (960 bit)
priv:
    00:8a:30:91:0d:51:61:83:37:61:c4:e9:7e:11:e0:
    91:46:a4:bb:51:ed
pub:
    00:ab:1e:cb:13:3c:31:ca:0e:56:8f:90:7c:89:b5:
    ea:a5:7e:15:56:21:9c:35:e3:ef:be:5e:97:c8:02:
    2a:fa:ed:73:14:ee:8f:3f:26:8c:5d:a4:fb:1b:e5:
    1e:d1:f9:97:c8:83:0f:01:01:02:cd:3d:c0:c6:a5:
    7e:d6:b5:f6:90:af:d5:77:3b:a1:35:05:e5:ef:31:
    e8:30:07:2a:c2:6c:77:57:6a:77:5d:10:27:90:de:
    d8:e5:27:66:58:16:6d:c9:8a:fc:be:85:fc:57:65:
    6f:ad:f4:ce:22:bc:10:b0:3a:77:51:90:61:e7:52:
    0d
P:
    00:da:7f:d0:03:eb:e1:75:81:c7:5f:81:0b:97:78:
    79:e8:fb:d2:04:27:7c:2a:86:36:9c:02:ba:26:ba:
    9a:aa:36:c3:08:6f:c0:9a:e6:11:49:44:f2:c5:40:
    78:97:51:2e:5f:4f:d5:6e:9d:d2:00:fe:27:bd:97:
    4f:d4:ae:a2:93:7e:63:c2:7d:58:34:6b:bc:c6:68:
    fc:38:a5:39:b2:23:07:7f:9a:c2:b7:81:dc:39:97:
```

4.Extraer en DSAPub.pem la clave pública contenida en el archivo DSAkey.pem. De nuevo DSAPub.pem no debe estar cifrado ni protegido. Mostrad sus valores. Lo mismo para el archivo DSAkey.pem

Al igual que en el apartado anterior, utilizaremos el comando de openssl **dsa**.

Para la extracción de la clave pública utilizaremos el comando:

```
openssl dsa -in <nombre>DSAkey.pem -out <nombre>DSAPub.pem -pubout
```

Tanto para Javier como para Galera:

```
~/MyGitHub/SPSI/practica3 on   master!   23:48:55
$ openssl dsa -in JavierDSAkey.pem -out JavierDSAPub.pem -pubout
read DSA key
writing DSA key

~/MyGitHub/SPSI/practica3 on   master!   23:49:19
$ openssl dsa -in GaleraDSAkey.pem -out GaleraDSAPub.pem -pubout
read DSA key
writing DSA key
```

Por  ltimo mostraremos los valores de las claves:

Para **Javier**:

```
~/MyGitHub/SPSI/practica3 on   master!   23:52:28
$ openssl dsa -pubin -in JavierDSAPub.pem -text -noout
read DSA key
pub:
 39:cf:34:a3:e7:58:49:da:c8:6e:31:52:69:08:07:
 59:1c:1a:7e:7d:ce:df:35:1c:0a:8c:f1:6b:ba:05:
 ba:1f:1d:6e:c4:09:36:cc:7a:e4:f1:00:f2:d9:d9:
 4b:69:88:23:31:b7:82:f2:39:4a:23:47:b6:ec:60:
 82:ed:53:f1:d5:11:fd:40:98:ed:85:fd:75:ab:35:
 a8:19:05:f4:4a:dd:33:c5:57:87:4a:e8:15:d1:f5:
 1a:c4:7b:9c:11:db:cb:e9:db:97:9a:49:61:a6:74:
 19:1c:ff:eb:d8:bd:57:18:3e:90:ed:97:52:58:af
P:
 00:da:7f:d0:03:eb:e1:75:81:c7:5f:81:0b:97:78:
 79:e8:fb:d2:04:27:7c:2a:86:36:9c:02:ba:26:ba:
 9a:aa:36:c3:08:6f:c0:9a:e6:11:49:44:f2:c5:40:
 78:97:51:2e:5f:4f:d5:6e:9d:d2:00:fe:27:bd:97:
 4f:d4:ae:a2:93:7e:63:c2:7d:58:34:6b:bc:c6:68:
 fc:38:a5:39:b2:23:07:7f:9a:c2:b7:81:dc:39:97:
 d4:79:a8:bf:e3:44:5a:ab:06:b3:b2:74:df:3e:33:
 2c:3c:39:c7:24:12:6b:7b:02:32:d8:e9:6e:c0:c7:
 a7
Q:
 00:9a:f9:74:f5:0e:83:df:90:93:08:a0:88:7d:4c:
 bc:76:a5:90:ad:85
```

Para **Galera**:

```
~/MyGitHub/SPSI/practica3 on ↵ master! @ 23:52:50
$ openssl dsa -pubin -in GaleraDSAPub.pem -text -noout
read DSA key
pub:
 00:ab:1e:cb:13:3c:31:ca:0e:56:8f:90:7c:89:b5:
 ea:a5:7e:15:56:21:9c:35:e3:ef:be:5e:97:c8:02:
 2a:fa:ed:73:14:ee:8f:3f:26:8c:5d:a4:fb:1b:e5:
 1e:d1:f9:97:c8:83:0f:01:01:02:cd:3d:c0:c6:a5:
 7e:d6:b5:f6:90:af:d5:77:3b:a1:35:05:e5:ef:31:
 e8:30:07:2a:c2:6c:77:57:6a:77:5d:10:27:90:de:
 d8:e5:27:66:58:16:6d:c9:8a:fc:be:85:fc:57:65:
 6f:ad:f4:ce:22:bc:10:b0:3a:77:51:90:61:e7:52:
 0d
P:
 00:da:7f:d0:03:eb:e1:75:81:c7:5f:81:0b:97:78:
 79:e8:fb:d2:04:27:7c:2a:86:36:9c:02:ba:26:ba:
 9a:aa:36:c3:08:6f:c0:9a:e6:11:49:44:f2:c5:40:
 78:97:51:2e:5f:4f:d5:6e:9d:d2:00:fe:27:bd:97:
 4f:d4:ae:a2:93:7e:63:c2:7d:58:34:6b:bc:c6:68:
 fc:38:a5:39:b2:23:07:7f:9a:c2:b7:81:dc:39:97:
 d4:79:a8:bf:e3:44:5a:ab:06:b3:b2:74:df:3e:33:
 2c:3c:39:c7:24:12:6b:7b:02:32:d8:e9:6e:c0:c7:
 a7
Q:
 00:9a:f9:74:f5:0e:83:df:90:93:08:a0:88:7d:4c:
```

5. Coger un archivo que actuará como entrada, de al menos 128 bytes. En adelante, dicho archivo será denominado como "message".

Para crear el archivo utilizaremos:

```
dd if=/dev/zero of=message count=1024 bs=1
```

El parámetro count recibe el número de bits del archivo que se creará.

Hemos introducido 1024, pues $128 \text{ bytes} * 8 \text{ bits/byte} = 1024 \text{ bits}$.

El archivo será una retahíla de ceros, puesto que solo hemos creado el archivo, no lo hemos modificado nada.

6. Firmar directamente el archivo message

empleando el comando openssl pkeyutl sin calcular los valores hash, la firma deberá almacenarse en un archivo llamado message.sign. Mostrad el archivo con la firma.

Para firmar el archivo utilizaremos la clave privada de Javier con el comando:

```
openssl pkeyutl -sign -in message -inkey <nombre>DSApriv.pem -out message.bin
```

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 11:07:56
$ openssl pkeyutl -sign -in message -inkey JavierDSApriv.pem -out message.sign
Enter pass phrase for JavierDSApriv.pem:
```

Ahora mostraremos el original y el resultado:

#####Original

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 11:11:13
$ xxd message
00000000: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000010: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000040: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000050: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000060: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000070: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000080: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000090: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000000a0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000000b0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000000d0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000000e0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000000f0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000100: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000110: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000120: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000130: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000140: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000150: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000160: 0000 0000 0000 0000 0000 0000 0000 0000 .....
```

#####Firmado


```
~/MyGitHub/SPSI/practica3 on ◀ master! © 11:11:19
$ xxd message.sign
00000000: 302c 0214 5269 89f3 4921 baa4 b2c3 d64e  0,...Ri..I!.....N
00000010: 6464 150e 34d8 815e 0214 7c3a 9d09 5af8  dd..4..^..|:...Z.
00000020: 475f 74fe 8b44 dd41 ae6a f4d4 c719      G_t..D.A.j....
```

7.Construir un archivo message2 diferente de message tal que la verificación de la firma message.sign sea correcta con respecto al archivo message2.

Para ello crearemos un archivo message2, en este caso será del doble de tamaño que message(2048 bits):

```
~/MyGitHub/SPSI/practica3 on ◀ master! © 18:02:00
$ dd if=/dev/zero of=message2 count=2048 bs=1
2048+0 registros leídos
2048+0 registros escritos
2048 bytes (2,0 kB, 2,0 KiB) copied, 0,00811345 s, 252 kB/s
```

Ahora introduciremos al final, en los últimos bits, un cambio para comprobar que aún así verificará la firma:

```
~/MyGitHub/SPSI/practica3 on master! 18:03:32
$ bless message2
Gtk-Message: 18:03:36.291: Failed to load module "canberra-gtk-module"
Unexpected end of file has occurred. The following elements are not closed: pref, preference
Directory '/home/jota/.config/bless/plugins' not found.
Directory '/home/jota/.config/bless/plugins' not found.
Directory '/home/jota/.config/bless/plugins' not found.
Could not find file "/home/jota/.config/bless/export_patterns".
Could not find file "/home/jota/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/jota/.config/bless/preferences.xml
Sharing violation on path /home/jota/.config/bless/preferences.xml
Sharing violation on path /home/jota/.config/bless/preferences.xml
```

Ahora utilizando este comando comprobaremos la verificación:

```
openssl pkeyutl -verify -in message2 -sigfile message.sign -pubin -inkey <nombre>DS
```

```
~/MyGitHub/SPSI/practica3 on master! 18:14:28
$ openssl pkeyutl -verify -in message2 -sigfile message.sign -pubin -inkey JavierDSApub.pem
Signature Verified Successfully
```

Esto tiene una explicación "simple", resulta que **pkeyutl** tiene un **pequeño BUG**. Si firmamos un archivo con "message" de 128 bits(a 0 todos los bits o bien a un contenido "X") y creamos un archivo "message2" de 129 bits o más, donde los primermos 128 bits coinciden con los de "message", resulta que verifica la firma. Esto es debido a que trunca (algo así como "recortar") el archivo "message2" a partir de los 128 primeros bits.

8. Calcular el valor hash del archivo con la clave pública DSApub.pem usando sha384 con salida hexadecimal con bloques de dos caracteres separados por dos puntos. Mostad los valores por salida estándar y guardarlos en DSApub.sha384.

Para ello utilizaremos el siguiente comando:

```
openssl dgst -c -sha384 -hex <nombre>DSApub.pem > <nombre>DSApub.sha384
```

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 18:50:25
$ openssl dgst -c -sha384 -hex JavierDSApub.pem > JavierDSApub.sha384
~/MyGitHub/SPSI/practica3 on ◀ master! @ 18:50:50
$ cat JavierDSApub.sha384
SHA384(JavierDSApub.pem)= 6e:0f:33:7c:f6:3c:6a:c1:20:7e:d1:42:17:e1:ec:21:d6:cf:47:d5:5e:5e:43:40:ca:78:d0:f7:3a:1c:c2:94:62:c0:d5:85:c8:97:dd:6c:c3:cf:41:c8:81:94:8a:cc
```

Además mostraremos también el resultado:

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 18:51:28
$ xxd JavierDSApub.sha384
00000000: 5348 4133 3834 284a 6176 6965 7244 5341  SHA384(JavierDSA
00000010: 7075 622e 7065 6d29 3d20 3665 3a30 663a  pub.pem)= 6e:0f:
00000020: 3333 3a37 633a 6636 3a33 633a 3661 3a63  33:7c:f6:3c:6a:c
00000030: 313a 3230 3a37 653a 6431 3a34 323a 3137  1:20:7e:d1:42:17
00000040: 3a65 313a 6563 3a32 313a 6436 3a63 663a  :e1:ec:21:d6:cf:
00000050: 3437 3a64 353a 3565 3a35 653a 3433 3a34  47:d5:5e:5e:43:4
00000060: 303a 6361 3a37 383a 6430 3a66 373a 3361  0:ca:78:d0:f7:3a
00000070: 3a31 633a 6332 3a39 343a 3632 3a63 303a  :1c:c2:94:62:c0:
00000080: 6435 3a38 353a 6338 3a39 373a 6464 3a36  d5:85:c8:97:dd:6
00000090: 633a 6333 3a63 663a 3431 3a63 383a 3831  c:c3:cf:41:c8:81
000000a0: 3a39 343a 3861 3a63 630a  :94:8a:cc.
```

Aquí podemos comprobar que se ha aplicado **sha384**.

Además se puede mostrar por pantalla el resultado directamente con este comando:

```
openssl dgst -c -sha384 -hex <nombre>DSApub.pem && xxd <nombre>DSApub.pem
```

9. Calcular el valor hash de message2 usando una función hash de 160 bits con salida binaria. Guardad el hash en message2. y mostrar su

contenido.

En este caso utilizaremos SHA1, con ello el comando utilizado será:

```
openssl dgst -binary -sha1 message2 > message2.ripemd160
```

Aquí podemos observar el contenido del archivo y la creación de este:

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 22:23:40
$ openssl dgst -binary -sha1 message2 > message2.sha1

~/MyGitHub/SPSI/practica3 on ◀ master! @ 22:24:13
$ bless message2.sha1
Gtk-Message: 22:24:33.685: Failed to load module "canberra-gtk-module"
Root element is missing.
Directory '/home/jota/.config/bless/plugins' not found.
Directory '/home/jota/.config/bless/plugins' not found.
Directory '/home/jota/.config/bless/plugins' not found.
Could not find file "/home/jota/.config/bless/export_patterns".
Could not find file "/home/jota/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/jota/.config/bless/preferences.xml
Sharing violation on path /home/jota/.config/bless/preferences.xml
Sharing violation on path /home/jota/.config/bless/preferences.xml
```

Consola

practica3 : bless - KDE Terminal Emulator

/home/jota/MyGitHub/SPSI/practica3/message2.sha1 - Bless

File Edit View Search Tools Help

message2.sha1 ✕

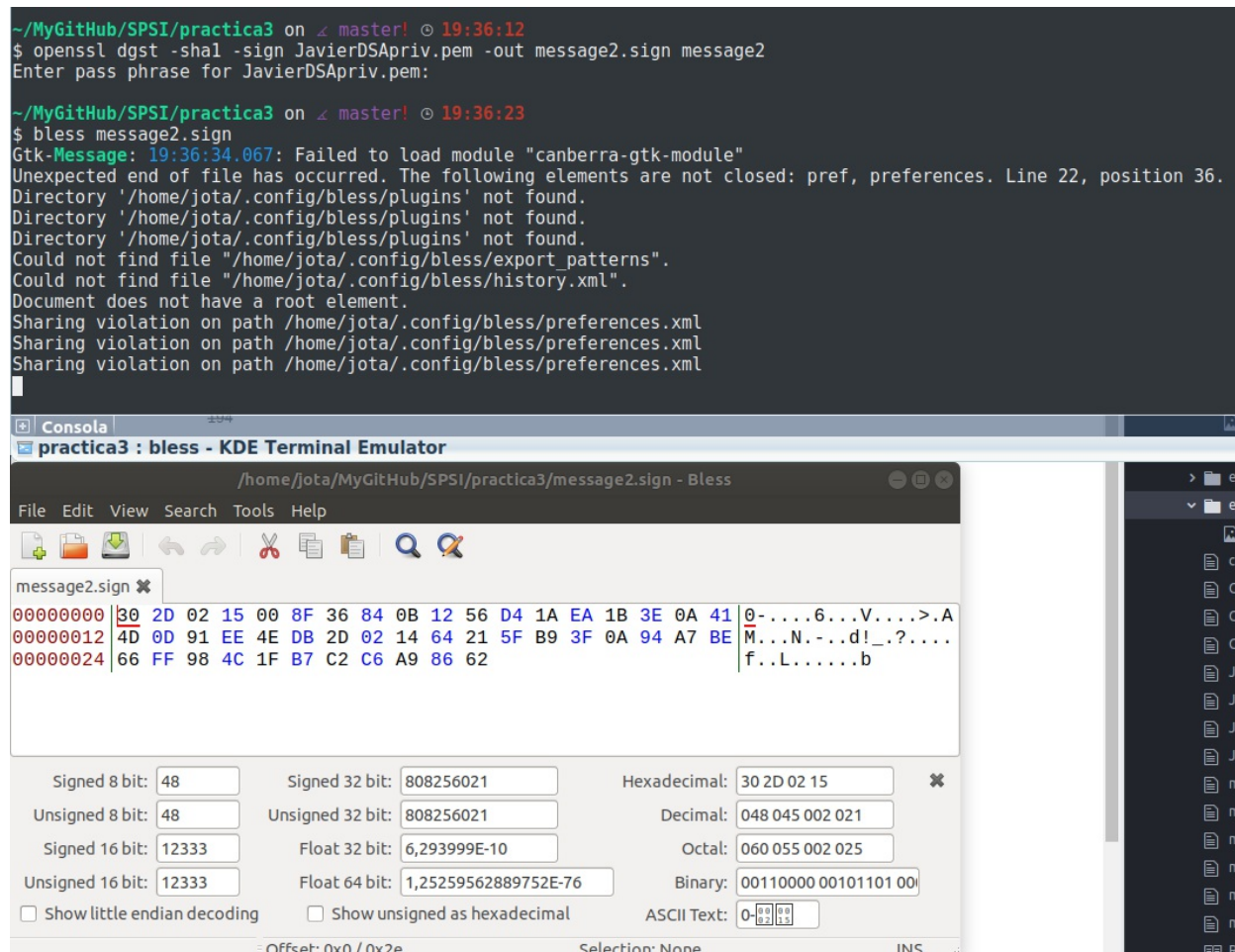
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|----|---|-----|---|---|---|---|---|---|--|
| 00000000 | 1D | D4 | 7D | EF | 5C | C3 | 5B | 8D | BE | EB | 61 | DC | 50 | 4A | 1E | 21 | 1C | 43 | ..} | .. | [| ... | a | P | J | ! | ! | C | |
| 00000012 | EB | FD | | | | | | | | | | | | | | | | | | .. | | | | | | | | | |

| | | | | | | |
|--|------|---|-----------------------|--------------|----------------------|---|
| Signed 8 bit: | 29 | Signed 32 bit: | 500465135 | Hexadecimal: | 1D D4 7D EF | ✕ |
| Unsigned 8 bit: | 29 | Unsigned 32 bit: | 500465135 | Decimal: | 029 212 125 239 | |
| Signed 16 bit: | 7636 | Float 32 bit: | 5,624615E-21 | Octal: | 035 324 175 357 | |
| Unsigned 16 bit: | 7636 | Float 64 bit: | 5,56013485953123E-165 | Binary: | 00011101 11010100 01 | |
| <input type="checkbox"/> Show little endian decoding | | <input type="checkbox"/> Show unsigned as hexadecimal | | ASCII Text: | ??? | |
| Offset: 0x0 / 0x13 | | | Selection: None | | INS | |

10. Firmad el archivo message2 mediante el comando openssl dgst y la funcion hash del punto anterior. La firma deberá almacenarse en un archivo llamando message2.sign

Firmaremos el archivo utilizando el comando:

```
openssl dgst -sha1 -sign <nombre>DSApriv.pem -out message2.sign message2
```



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

Podemos verificar la firma con dgst de Openssl mediante:

```
openssl dgst -sha1 -verify JavierDSAPub.pem -signature message2.sign message2
```

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 19:37:18  
$ openssl dgst -sha1 -verify JavierDSAPub.pem -signature message2.sign message2  
Verified OK
```

12. Verificad que message2.sign es una firma correcta para message2 pero empleando el comando openssl pkeyutl

Utilizando el siguiente comando podremos comprobar que se verifica:

```
openssl pkeyutl -verify -in message2.sha1 -sigfile message2.sign -pubin  
-inkey <nombre>DSAPub.key
```

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 22:26:09  
$ openssl pkeyutl -verify -in message2.sha1 -sigfile message2.sign -pubin -inkey JavierDSAPub.pem  
Signature Verified Successfully
```

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

Para ello emplearemos el comando utilizado anteriormente **dgst** y como dice el enunciado le añadiremos la clave '12345', tal que:

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

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 0:58:23  
$ openssl dgst -hmac '12345' sharedDSA.pem  
HMAC-SHA1(sharedDSA.pem)= faled61c00b464b51ff00a20630bf93afee6410d
```

14. Simular una ejecución completa del protocolo Estación a Estación. Para ello emplearemos como

claves para firmar/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 se deben generar nuevamente. El algoritmo simétrico a utilizar en el protocolo estación a estación será AES-128 en modo CFB8.

1. Recopilamos las claves que vamos a utilizar a lo largo del ejercicio. Los participantes serán: Javier y Galera.

Por tanto las claves que utilizaremos para firmar y verificar serán:

- **Javier:** JavierDSAPub.pem/JavierDSAPriv.pem

- **Galera:** GaleraDSAPub.pem/GaleraDSAPriv.pem

Para el protocolo DH utilizaré las claves de la práctica anterior. Añadiendo una nueva pareja de claves para Galera.

2. Mostramos la clave privada y pública asociada a las curvas elípticas de Javier:

Nota: la pass de las claves privadas es: 0123456789

Por parte de Javier tenemos:

PRIVADA

```
~/MyGitHub/SPSI/practica3 on ◀ master! ☹ 1:23:24
$ openssl ec -in JavierECpriv.pem -text -noout
read EC key
Enter PEM pass phrase:
Private-Key: (256 bit)
priv:
    00:9f:22:21:5f:fe:a7:48:c9:23:ba:0b:55:09:50:
    9e:f6:29:c4:9b:ee:b9:cc:c3:3f:51:89:95:92:81:
    f3:a4:71
pub:
    04:c8:b7:6c:c7:5c:78:1e:f9:de:9e:97:d2:c7:f4:
    5e:e8:21:f8:c7:7b:02:c3:2f:c4:03:6b:7f:fc:ff:
    e7:08:29:55:55:4e:66:c1:61:b0:f8:fa:28:d5:2d:
    4c:1e:5b:f2:97:21:6b:f5:91:4c:aa:4c:be:19:60:
    16:d6:49:db:2f
ASN1 OID: prime256v1
```

PUBLICA

```
~/MyGitHub/SPSI/practica3 on ◀ master! ☹ 1:27:56
$ openssl ec -pubin -in JavierECpub.pem -text -noout
read EC key
Private-Key: (256 bit)
pub:
    04:c8:b7:6c:c7:5c:78:1e:f9:de:9e:97:d2:c7:f4:
    5e:e8:21:f8:c7:7b:02:c3:2f:c4:03:6b:7f:fc:ff:
    e7:08:29:55:55:4e:66:c1:61:b0:f8:fa:28:d5:2d:
    4c:1e:5b:f2:97:21:6b:f5:91:4c:aa:4c:be:19:60:
    16:d6:49:db:2f
ASN1 OID: prime256v1
```

Por parte de Galera tenemos:

PRIVADA

```
~/MyGitHub/SPSI/practica3 on ◀ master! ☹ 1:26:25
$ openssl ec -in GaleraECpriv.pem -text -noout
read EC key
Enter PEM pass phrase:
Private-Key: (256 bit)
priv:
    5d:04:f6:1a:30:84:bf:0d:44:18:00:0b:48:eb:7b:
    8a:f9:d6:44:33:3c:e1:53:d6:ad:df:23:11:f7:80:
    a4:aa
pub:
    04:0c:9c:5e:42:b1:cf:4f:4a:a3:45:f2:ac:be:3d:
    a0:33:66:17:c4:85:af:98:b5:b5:13:b0:9c:a1:d8:
    8e:89:db:a0:3a:90:37:0d:c5:c4:36:bf:6d:b5:a4:
    de:8b:9a:50:02:dc:9b:bf:47:bf:1c:16:20:6a:08:
    c0:97:5a:5e:50
ASN1 OID: prime256v1
```

PUBLICA

```
~/MyGitHub/SPSI/practica3 on ◀ master! ☹ 1:27:20
$ openssl ec -pubin -in GaleraECpub.pem -text -noout
read EC key
Private-Key: (256 bit)
pub:
    04:0c:9c:5e:42:b1:cf:4f:4a:a3:45:f2:ac:be:3d:
    a0:33:66:17:c4:85:af:98:b5:b5:13:b0:9c:a1:d8:
    8e:89:db:a0:3a:90:37:0d:c5:c4:36:bf:6d:b5:a4:
    de:8b:9a:50:02:dc:9b:bf:47:bf:1c:16:20:6a:08:
    c0:97:5a:5e:50
ASN1 OID: prime256v1
```

Para generar la pareja de claves públicas y privadas se ha seguido el mismo proceso que en la práctica anterior.

Ya tenemos todo listo para empezar la simulación.

3. Comenzaremos utilizando una diapositiva del profesor:

Estación a estación

Mejora del protocolo de Diffie-Hellman buscando resolver sus debilidades. Fijamos $g \in \mathbb{Z}_p^*$ como en el protocolo de Diffie-Hellman. Asumimos que tenemos disponible un esquema de firma digital. Cada usuario tiene una pareja de claves (s, v) privada-pública para firma y verificación. Las partes públicas son auténticas y accesibles.

- 1 Alicia escoge aleatoriamente $1 \leq a \leq p - 2$, calcula $c = g^a \text{ mód } p$ y lo envía a Bernabé.
- 2 Bernabé escoge aleatoriamente $1 \leq b \leq p - 2$, calcula $d = g^b \text{ mód } p$ y $k = g^{ab} = (c)^b \text{ mód } p$. Calcula $s = \text{sgn}_{s_b}(c||d)$. Envía $(d, e_k(s))$ a Alicia.
- 3 Alicia calcula $k = g^{ab} = d^a \text{ mód } p$ y $s = d_k(e_k(s))$, verifica $\text{vfy}_{v_b}(c||d, s)$. Firma $t = \text{sgn}_{s_a}(d||c)$ y envía a Bernabé $e_k(t)$.
- 4 Bernabé calcula $t = d_k(e_k(t))$ y verifica $\text{vfy}_{v_a}(d||c, t)$.
- 5 A partir de este momento ambos pueden estar seguros de que la clave secreta k está compartida sólo por ellos dos.

4. Javier comparte su clave pública con Galera.

5. Galera genera su clave derivada usando el comando **pkeyutl**, a partir de su clave privada y la pública de Javier tal que:

```
openssl pkeyutl -derive -inkey GaleraECpriv.pem -peerkey JavierECpub.pem -out keyGa.
```

El resultado de esta operación es:

```
~/MyGitHub/SPSI/practica3 on ◀ master! ⌚ 2:11:23
$ openssl pkeyutl -derive -inkey GaleraECpriv.pem -peerkey JavierECPub.pem -out keyGalera.b
Enter pass phrase for GaleraECpriv.pem:
```

```
~/MyGitHub/SPSI/practica3 on ◀ master! ⌚ 2:11:31
$ bless keyGalera.bin
Gtk-Message: 02:11:39.019: Failed to load module "canberra-gtk-module"
Unexpected end of file has occurred. The following elements are not closed: pref, preferenc
Directory '/home/jota/.config/bless/plugins' not found.
Directory '/home/jota/.config/bless/plugins' not found.
Directory '/home/jota/.config/bless/plugins' not found.
Could not find file "/home/jota/.config/bless/export_patterns".
Could not find file "/home/jota/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/jota/.config/bless/preferences.xml
Sharing violation on path /home/jota/.config/bless/preferences.xml
Sharing violation on path /home/jota/.config/bless/preferences.xml
```

Consola

practica3 : bless - KDE Terminal Emulator

/home/jota/MyGitHub/SPSI/practica3/keyGalera.bin - Bless

File Edit View Search Tools Help

keyGalera.bin ✕

| | | |
|----------|---|-------------------|
| 00000000 | 7E 8D C7 3B 7C 6A A9 9D 6F 5B 72 63 E0 59 7B 7A 89 68 | ~.; j..o[rc.Y{z.h |
| 00000012 | 29 FC F7 10 E5 63 EA 6A 96 56 A3 E2 5C 14 |)....c.j.V..\.. |

| | | | | | |
|--|-------|---|-----------------------|-----------------|----------------------|
| Signed 8 bit: | 126 | Signed 32 bit: | 2123220795 | Hexadecimal: | 7E 8D C7 3B |
| Unsigned 8 bit: | 126 | Unsigned 32 bit: | 2123220795 | Decimal: | 126 141 199 059 |
| Signed 16 bit: | 32397 | Float 32 bit: | 9,422781E+37 | Octal: | 176 215 307 073 |
| Unsigned 16 bit: | 32397 | Float 64 bit: | 3,98845658985986E+301 | Binary: | 01111110 10001101 11 |
| <input type="checkbox"/> Show little endian decoding | | <input type="checkbox"/> Show unsigned as hexadecimal | | ASCII Text: ~?; | |
| Offset: 0x0 / 0x1f | | | | Selection: None | |
| | | | | INS | |

Una vez realizado esto, Galera concatena la clave pública de Javier con su clave pública, para ello utiliza:

```
cat GaleraECPub.pem JavierECPub.pem > concatenacion.txt
```

```
~/MyGitHub/SPSI/practica3 on  master! 2:12:37
$ cat GaleraECpub.pem JavierECpub.pem > concatenacion.txt

~/MyGitHub/SPSI/practica3 on  master! 2:47:02
$ cat concatenacion
cat: concatenacion: No existe el archivo o el directorio

~/MyGitHub/SPSI/practica3 on  master! 2:47:08
$ cat concatenacion.txt
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEJxeQrHPT0qjRfKsvj2gM2YXxIWv
mLW1E7Ccodi0idug0pA3DcXENr9ttaTei5pQAtybv0e/HBYgagjAllpeUA==
-----END PUBLIC KEY-----
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEyLdsx1x4HvnenpfSx/Re6CH4x3sC
wy/EA2t//P/nCClVVU5mwWGw+PoolS1MHlvlyFr9ZFMqky+GWAW1knbLw==
-----END PUBLIC KEY-----
```

Galera ahora firmará este fichero(concatenacion.txt), cifrando el valor hash con su clave privada. El resultado será almacenado en un archivo llamado firma_galera.sign y utilizará el comando **dgst** tal que:

```
openssl dgst -sha256 -sign GaleraDSPriv.pem -out firma_galera.sign concatenacion.txt
```

```
~/MyGitHub/SPSI/practica3 on  master! 2:48:21
$ cat firma_galera.sign
0-x?]008U0+0{7/0zz0000yM0500_r\00%
```

Para cifrar el archivo se utilizará el cifrado simétrico indicado en el enunciado, AES-128 en modo CFB8:

```
openssl aes-128-cfb8 -pass file:keyGalera.bin -in firma_galera.sign -out firma_gale
```


Como resultado obtenemos:

```
~/MyGitHub/SPSI/practica3 on ◀ master! @ 2:54:20
$ openssl aes-128-cfb8 -pass file:keyGalera.bin -in firma_galera.sign -out firma_galera_E.sign

~/MyGitHub/SPSI/practica3 on ◀ master! @ 2:54:57
$ bless firma_galera_E.sign
Gtk-Message: 02:55:09.393: Failed to load module "canberra-gtk-module"
Root element is missing.
Directory '/home/jota/.config/bless/plugins' not found.
Directory '/home/jota/.config/bless/plugins' not found.
Directory '/home/jota/.config/bless/plugins' not found.
Could not find file "/home/jota/.config/bless/export_patterns".
Could not find file "/home/jota/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/jota/.config/bless/preferences.xml
Sharing violation on path /home/jota/.config/bless/preferences.xml
Sharing violation on path /home/jota/.config/bless/preferences.xml
```

| Hex | ASCII |
|--|-------------------|
| 00000000 53 61 6C 74 65 64 5F 5F CE B6 83 4A 40 86 FD EC 97 3C | Salted____J@....< |
| 00000012 7C 3E 1B 49 AA 93 AD EB 5E 0C 09 0F 96 10 7B 26 78 57 | >.I....^.....{&xW |
| 00000024 3C E1 9A C2 50 A1 51 10 74 C2 53 3F A4 FA DD 9B A4 1A | <...P.Q.t.S?..... |
| 00000036 3C AD 20 87 C8 46 4D A0 3C | <. ..FM.< |

| | | | | | |
|--|-------|---|----------------------|-----------------|----------------------|
| Signed 8 bit: | 83 | Signed 32 bit: | 1398893684 | Hexadecimal: | 53 61 6C 74 |
| Unsigned 8 bit: | 83 | Unsigned 32 bit: | 1398893684 | Decimal: | 083 097 108 116 |
| Signed 16 bit: | 21345 | Float 32 bit: | 9,681872E+11 | Octal: | 123 141 154 164 |
| Unsigned 16 bit: | 21345 | Float 64 bit: | 4,54305331895921E+93 | Binary: | 01010011 01100001 01 |
| <input type="checkbox"/> Show little endian decoding | | <input type="checkbox"/> Show unsigned as hexadecimal | | ASCII Text: | Salt |
| Offset: 0x0 / 0x3e | | | | Selection: None | INS |

Galera envía la firma encriptada y su clave pública a Javier.

6. Javier calculará su clave derivada como hizo Galera, simplemente ahora utilizará su clave privada y la pública de Galera:

```
openssl pkeyutl -derive -inkey JavierECpriv.pem -peerkey GaleraECpub.pem -out keyJa
```