PROTOCOLOS CRIPTOGRÁFICOS

Práctica 3

Índice

1. Generación de parámetros

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dsaparam -out sharedDSA.pem 2048
Generating DSA parameters, 2048 bit long prime
This could take some time
.....+.....+.+.+..+.+.....+.....+.....+.....+.....+.....+..
+.....+.....+.....+...+.....+.....+.....+.....+..+.....+..
.....+.....+.....+...+.....+.....+.....+.....+.....+...+..
....+...+.....+.....+.....+.....+.....+.....+.....
.....+.....+.....+.....+.....++++
++++++++++++++++++++++++++++++++++++++++++++++++++*
+.+.+.....+.....+.....+.....+.....+.....+.....
...+.....+.+.....+.....+.....+.....+.....+.
.....+.....+.....+.....+.....+.....+.....+..+...
.....+.....+.....+..+.....+....+..+.....+.....+.....+
.....+...+.....+.....+.....+.....+..+.....+.....+.....+.....
+.....+.....+.....+.....+.....+.....+.+.+...+....+.....+.
.....+.....+.....+..+.....+.+.+..+.+.....+..+.....+.....+.....
.....+..+.....+.....+...+....+.....+.....+..++++++++++
++++++++++++++++++++++++++++++++++++++++*
```

Dsaparam: indica que se van a crear parámetros de generación
-out: nombre del archivo de salida
2048: para que sea de 2048 bits

Parámetros

```
1  -----BEGIN DSA PARAMETERS-----
2  MIICLAKCAQEA4LkE881EKvj6OOYlHKsxLlpfXnyKnBMCiQxG5Hl9Ebljb9t5rF/m
3  4ZS9+CWWBHV6Ozn++6VNkt12lkM1P/uAoes01guZ/0fj1vW81MMH1u3OCfQJJ0hp
4  2u6ifx30+9Pbn/B9ASigL7e20tuThCrA3EpEx/yaSqd1sTLVJC2HZsvXdqkE2AXr
5  Wm5Mksx+VmJPRS2bV8d3eEe5vjCV/0Qeut78gCpz2V0bqsf3/2BZNeaNqfz657l1
6  cV8uRJIfWJXxCKsI3dC76uoY50h4Klk5XK6KwPZAnPlFZWVuERpJF7Gtbyo2PVPN
7  gKydZGj5ShRdMH7MYMGWmIXEZJcWeEf5GQIhAN0NMwQneHjRxO1RtNE+zPZeknoP
8  YEBbQb9GTIlUNsS/AoIBAEblAHEFUf5mUj4ETJmhW6ozmWWNb6kmX87nQguGGZC4
9  heD2gEVOBO7r2ZglNvbA14R7NJzaiTTtjLN+w40/iYBv8iMN1WD/2h4qTlRsz5Wt
10 U5bKi8I5HeCMybEYRNuVPpFu9E2LwQIa87aRNgwIzapgB2wpB+OQVB54oMvnskhR
11 o9KueH40lBayOGODJW9B1eiSSietCwMLTtZCNTN87GIZNv6fX35TKxiPI9Z2gC6O
12 6XOu+sVdYoDPfcAD+UJT6QiG4vUGaoweXHNLBxzf419TTU/3CPT4UuTGHcfjSE4F
13 gakfWzbLTj5RfH1myuGO1obSDk48bilhlWzrwQZwxfY=
14 -----END DSA PARAMETERS-----
```

2. Generación de claves

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl gendsa -out pedroDSAkey.pem sharedDSA.pem
Generating DSA key, 2048 bits
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl gendsa -out fuertesDSAkey.pem sharedDSA.pem
Generating DSA key, 2048 bits
```

Gendsa: indica que se va a crear una clave DSA
-in: indica el fichero donde están los parámetros, (Se han generado con el comando anterior)
-out: el nombre del archivo con la firma
Al final del todo se pone el archivo de donde obtener los parámetros, en este caso sharedDSA.pem

3. Extracción de la clave privada

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dsa -in pedroDSAkey.pem -out pedroDSApriv.pem
-des
read DSA key
writing DSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dsa -in fuertesDSAkey.pem -out fuertesDSApriv.
pem -des
read DSA key
writing DSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Contraseña: “0123456789”
Dsa: para extraer la clave privada
-des: para cifrarlo

Clave privada

pedroDSApriv.pem

```
-----BEGIN DSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,95855DE92D9C3281

b8OYvzimVLBeeNQzIrW5UubM+Q1XpwGk8ZUJkQ15M7ztk3Lhj4SfFUGEA8PoS3J2
wNlUhgqifeSTsO6E4Hq4giCH7wxxYEH0UYVqa8fCK57OI3Tryk48PJjpDgoffMpm
LeJ0jlX8RlQR4uuSn0WqHsMoG1lqTs1Yxe1tCNwP5qUOE0wtjrLNx69S7tKTdMJY
5SVAjfmY4teHBK9dgEE02JRmt1yQw17et/9Ns1Rur/m4ghJRZ9T4+FNLMBariigG
SxLxef/cVUISmXWeNnoksSj0OU7dxL3HyM+l9a4cYu3+15PMbJ7jtANeW0SRYNfm
Hdy3iVm/Z2bGNEB2OmWcmRoHYfXRQLIWbrHqwZoqioNgl3whelHDbcndaSpBxBII
+u6YQe0leoaTD+c89RHlbBp7RW5v1FyBNJX+SKFLwaBk/8XxBnsqqXa1EGQdMU9F
d2pYvzUFMOywoBxpWL+BJJvQpBZZpo85+KCa18Vq/aohMGP6rRnarn9ydyMkNoUm
58PGiX5YcLfbHGdKlzUIxV7urS3t3xp2SIjYe6SqwFdsBaNtNZxMirIAxrvsKHMJ
U513f7fZnawUICsdlznzs4B3zsGjff63sB88d8q36w30MPU4g++pY3JkpMnqYhsc
FbWiW8jnfK60npaqppVl4t8wUitYJMqUCqWsgmRsNMQXVO3XHgSoXuOPcCwMX5WC
Zk4kR3DwIwE59PWjdrC/l8EG5hJHsObNZY6N6228MJMWO5SPA2nqpgJefugSGwG6
v1/FKhHhqvwlZYPhK3oiJWZGURNwhEzYrlcf1RHbghw+l9vAbLq/zRaEWIMORra3
U5wD7LU1lglhdYDv1+WUmsdLus/56rR3/eTDkLySeaHNFR7NxZC7ZPjsZ3MVqYZq
VWH4L4Hj9DFVuSQBPZwZZ55cj5Ckf53e05uShRkaLmPSsN4UpTMxFMr6s9fPJ9qj
wjQESokjoGsijN3RdMS+d8RKsxjK12dMxPbYWdwX/Qyd6n3GPETman5upJkQm3AZ
Avq5WJ1oeybAuOPoIRB0IBzCmUH1n8RFA2vxG6hGUJqo19BdQhKEoxl7a2SKms6f
9YTnatdMr9htyHR81ga8yXWkWGxLG8f1CP5s8uJNAfQrcPIF5C18KBLR0UFT71wQ
-----END DSA PRIVATE KEY-----
```

fuertesDSApriv.pem

```
-----BEGIN DSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,71123DFAB6847A83

SGWUWPAtjaZqUW8lzVAprFzVf9LXwmAFsr41/zId363j9Lq80GFoi+T0yido5m8S
q6TAW5Pm9xnYkslLFRCtrFsRD4bQ4N/R2u8a6pp/Wrhwi46uhTf46ypbOatXD9VO
JhE/8xDVNcPEIlmC4gJyeJAe0cbuCgGBt6gjiXd6I/gEcMUYS042L6hFEhlpjeDZ
wcuwpl5BUnApKF1a0l6ksMvBIR2J732ks9mzT965eVHu0xvx0mjGhYoaANOHZpEl
k7oWtQPJ6irL/tczAiJpK5BeIvjzxHw7Y7El2ZviZfp/oQyqv0dSxRsvAMEjk3Vv
k/FRXIFgBWcUSjP7ffZsTW3oXqRJ+PSAQXxZIiygarxUkOJjy0Eq0HBr+1ezh6eP
YSuK7bcEzsDC0cUTjkK/Ij8c2dcso+etXaK9MnSwZnLBJetnbWWql3B0NspJGf1E
gIao0pEZ4Zj6pWOPuudZumLPkzzc/yVX3qVhwgZBzfj8pN7oLBI2OYEbP2kwmmSI
yGIQCAEn1dZ7xOneVu/UW23TpdPSOpU9dHPdUJR+kfJ6N9xqWKgLbKuexcxuNgHk
d6GwuQvOWe5Z7T6GtHSZjwRc5lgOVMjpv6q1R1W3Okqr2TH60amewPvgDcw7aXAK
rT0SgjYBEm8j6E/hxIghkynFStCbOMLNQTpjuK2jsA0xOfIzYFTu14Bldix7OJkm
RgTK1wU2htskLKekK4gLab+x562qqiwzquRbkL8scMnvb5rUythvkVKDzPEgTmiT
Cv+NMQSHl9+EvldkCTnnItx+jT9G3QKiLA6hOrc7+qFXpa9ZPlOZclrfn8tsok9j
4OTxOiuOR5IiTsNFKNdud+3zIbJsK4qrVEh+TTZoLje0m/Q9Rs4oHpWclFgJULlP
SL2ellHEcce7z3vLNa6PJP4tWzkvRDXeM+pw8ok5SVZ0k6n56kgAp6vapTdGCT8+
4u5kKHWjlBM14CchUUEcwUw58zNLRCYuATfeMf2Ix+yYgnvLM5VhS/8bt6nwUFs8
ExYfYBBPl7EbDdHNCUwNn2U8txfv0vc0vwUbH+dBS0nPSTN06mBxZskHIwSR52Xb
WnzWtPVZt3+Ml1oRH+RZkpIKNKkPL20+v99Y0bie8qiIv2Jil3LAFQsD35ASjdsf
-----END DSA PRIVATE KEY-----
```

Conclusiones

Se puede ver que los archivos resultantes son diferentes, esto puede ser debido al salt del cifrado, que mete número aleatorios, por lo que pese a tener ambos archivos la misma contraseña, el archivo resultante es distinto.

4. Extracción de la clave pública

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dsa -in pedroDSAkey.pem -out pedroDSApub.pem -
pubout
read DSA key
writing DSA key
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dsa -in fuertesDSAkey.pem -out fuertesDSApub.p
em -pubout
read DSA key
writing DSA key
```

-pubout: para indicar que queremos la clave pública

Clave pública

pedroDSApub.pem

```
-----BEGIN PUBLIC KEY-----
MIIDRjCCAjkGByqGSM44BAEwggIsAoIBAQDguQTzzUQq+Po45iUcqzEuWl9efIqc
EwKJDEbkeX0RuWNv23msX+bhlL34JZYEdXo7Of77pU2S3XaWQzU/+4Ch6zTWC5n/
R+PW9bzUwwfW7c4J9AknSGna7qJ/HfT709uf8H0BKKAvt7bS25OEKsDcSkTH/JpK
p3WxMtUkLYdmy9d2qQTYBetabkySzH5WYk9FLZtXx3d4R7m+MJX/RB663vyAKnPZ
XRuqx/f/YFk15o2p/PrnuXVxXy5Ekh9YlfEIqwjd0Lvq6hjnSHgqWTlcrorA9kCc
+UVlZW4RGkkXsa1vKjY9U82ArJ1kaPlKFF0wfsxgwZaYhcRklxZ4R/kZAiEA3Q0z
BCd4eNHE7VG00T7M9l6Seg9gQFtBv0ZMiVQ2xL8CggEARuUAcQVR/mZSPgRMmaFb
qjOZZY1vqSZfzudCC4YZkLiF4PaARU4E7uvZmCU29sDXhHs0nNqJNO2Ms37DjT+J
gG/yIw3VYP/aHipOVGzPla1TlsqLwjkd4IzJsRhE25U+kW70TYvBAhrztpE2DAjN
qmAHbCkH45BUHnigy+eySFGj0q54fjSUFrI4Y4Mlb0HV6JJKJ60LAwtO1kI1M3zs
Yhk2/p9fflMrGI8j1naALo7pc676xV1igM99wAP5QlPpCIbi9QZqjB5cc0sHHN/j
X1NNT/cI9PhS5MYdx+NITgWBqR9bNstOPlF8fWbK4Y7WhtIOTjxuKWGVbOvBBnDF
9gOCAQUAAoIBACmGs+NQKx1VplWzGq2blNiiFiWuPCi9gwMhzSX9U/dmo1uCAtHJ
bOK/4ZiwOZae7OlGwISFwC86y183OLpWzbXSPha72tavVN0HLkUBY+GHKVdTMJBk
C0eFRjt4v6uJHPzPY/Bf9zpoL7XgsJ5Pf9JTT5fsvpkkdBKfvr/QPTnO6kIQBIYt
XISIiKLsKwyblhhgp8z7ATFNaL6ht7lBfzXsxCSQh9yfmsIGvZRezm//2IKCphsz
DaELpTJGewC7QVHyUCNcL6ZLGDa7jnyHisweQyP2cHgkUGi72FeLYF3+9IE9/ZFm
guA9gYYgWIBw3qJZHa2komOHX+JL7IDuNug=
-----END PUBLIC KEY-----
```

fuertesDSApub.pem

```
-----BEGIN PUBLIC KEY-----
MIIDRzCCAjkGByqGSM44BAEwggIsAoIBAQDguQTzzUQq+Po45iUcqzEuWl9efIqc
EwKJDEbkeX0RuWNv23msX+bhlL34JZYEdXo7Of77pU2S3XaWQzU/+4Ch6zTWC5n/
R+PW9bzUwwfW7c4J9AknSGna7qJ/HfT709uf8H0BKKAvt7bS25OEKsDcSkTH/JpK
p3WxMtUkLYdmy9d2qQTYBetabkySzH5WYk9FLZtXx3d4R7m+MJX/RB663vyAKnPZ
XRuqx/f/YFk15o2p/PrnuXVxXy5Ekh9YlfEIqwjd0Lvq6hjnSHgqWTlcrorA9kCc
+UVlZW4RGkkXsa1vKjY9U82ArJ1kaPlKFF0wfsxgwZaYhcRklxZ4R/kZAiEA3Q0z
BCd4eNHE7VG00T7M9l6Seg9gQFtBv0ZMiVQ2xL8CggEARuUAcQVR/mZSPgRMmaFb
qjOZZY1vqSZfzudCC4YZkLiF4PaARU4E7uvZmCU29sDXhHs0nNqJNO2Ms37DjT+J
gG/yIw3VYP/aHipOVGzPla1TlsqLwjkd4IzJsRhE25U+kW70TYvBAhrztpE2DAjN
qmAHbCkH45BUHnigy+eySFGj0q54fjSUFrI4Y4Mlb0HV6JJKJ60LAwtO1kI1M3zs
Yhk2/p9fflMrGI8j1naALo7pc676xV1igM99wAP5QlPpCIbi9QZqjB5cc0sHHN/j
X1NNT/cI9PhS5MYdx+NITgWBqR9bNstOPlF8fWbK4Y7WhtIOTjxuKWGVbOvBBnDF
9gOCAQYAAoIBAQCCDxt9Chwlw4a7BB+aXZEvhDxSE0ErKqnNSv0Y8wDb7Eb29eEZ
/PTCeIo2lW9d8xVyP7+IbiRhe0jpuvmwiowVVWiV9rLrR3kKgibCDdCsq4ml4rR2
nHjHRqshs3dB+ucm8fiS4/3qT+cXdqw1zwJcIXN17xj73lWsMfClB2/ge1S3plbd
gbZ7MUazPSoDFx3Px9fO/k2wDSEsQooQVcKf3rDtdhUjZJL6sR1at913dM34XbCR
H9UREqB5gIMcxD6T5uaaUbiCnU/H/Fpaut8p8o+jgcywJFmJ5OVbF7XWvFqUGEmw
Lp+CDHwGO5n3+Swp/YBOvlSS/pmPROKendiB
-----END PUBLIC KEY-----
```

5. Creamos un archivo de al menos 128 bytes

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ ls -la mensaje.in
-rwxr-xr-x 1 pedro pedro 128 Oct 28 03:53 mensaje.in
```

6. Firmamos el mensaje

Llegado a este punto, se encuentra un error al usar la clave privada generada "pedroDSApriv.pem", podríamos generar una clave RSA para solventar el problema, sin embargo, para evitar problemas futuros, se decide instalar Libre SSl.

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl version
OpenSSL 1.1.0g  2 Nov 2017
```

```
 96  cd /tmp/
 97  wget http://ftp.openbsd.org/pub/OpenBSD/LibreSSL/libressl-2.8.2.tar.gz
 98  tar -xzvf libressl-2.8.2.tar.gz
 99  cd libressl-2.8.2/
100  ./configure
101  make
102  sudo make install
103  sudo ldconfig
```

```
pedro@ubuntu:/tmp/libressl-2.8.2$ openssl version
LibreSSL 2.8.2
```

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl pkeyutl -sign -in mensaje.in -out mensaje.sign
 -inkey pedroDSApriv.pem
Enter pass phrase for pedroDSApriv.pem:
Public Key operation error
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ chmod 600 pedroDSA*
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ ls -l
total 44
-rw------- 1 pedro pedro 1224 Oct 29 01:47 fuertesDSAkey.pem
-rw------- 1 pedro pedro 1311 Oct 29 01:48 fuertesDSApriv.pem
-rw-r--r-- 1 pedro pedro 1194 Oct 29 01:49 fuertesDSApub.pem
-rw------- 1 pedro pedro 1675 Oct 28 05:09 keyRSA.pem
-rw------- 1 pedro pedro  128 Oct 28 03:53 mensaje.in
-rw-r--r-- 1 pedro pedro    0 Oct 29 02:04 mensaje.sign
-rw------- 1 pedro pedro 1228 Oct 29 01:47 pedroDSAkey.pem
-rw------- 1 pedro pedro 1311 Oct 29 01:48 pedroDSApriv.pem
-rw------- 1 pedro pedro 1194 Oct 29 01:48 pedroDSApub.pem
-rw------- 1 pedro pedro  169 Oct 28 04:23 pedroDSApub.sha384
-rw-r--r-- 1 pedro pedro  169 Oct 28 04:20 p.txt
-rw-r--r-- 1 pedro pedro  820 Oct 29 01:46 sharedDSA.pem
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ rm mensaje.sign
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl pkeyutl -sign -in mensaje.in -out mensaje.sign
 -inkey pedroDSApriv.pem
Enter pass phrase for pedroDSApriv.pem:
Public Key operation error
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl pkeyutl -sign -in mensaje.in -out mensaje.sign
 -inkey keyRSA.pem
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ ls -l
total 48
-rw------- 1 pedro pedro 1224 Oct 29 01:47 fuertesDSAkey.pem
-rw------- 1 pedro pedro 1311 Oct 29 01:48 fuertesDSApriv.pem
-rw-r--r-- 1 pedro pedro 1194 Oct 29 01:49 fuertesDSApub.pem
-rw------- 1 pedro pedro 1675 Oct 28 05:09 keyRSA.pem
-rw------- 1 pedro pedro  128 Oct 28 03:53 mensaje.in
-rw-r--r-- 1 pedro pedro  256 Oct 29 02:08 mensaje.sign
-rw------- 1 pedro pedro 1228 Oct 29 01:47 pedroDSAkey.pem
-rw------- 1 pedro pedro 1311 Oct 29 01:48 pedroDSApriv.pem
-rw------- 1 pedro pedro 1194 Oct 29 01:48 pedroDSApub.pem
-rw------- 1 pedro pedro  169 Oct 28 04:23 pedroDSApub.sha384
-rw-r--r-- 1 pedro pedro  169 Oct 28 04:20 p.txt
-rw-r--r-- 1 pedro pedro  820 Oct 29 01:46 sharedDSA.pem
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$
```

Pkeyutl: indica que se van a usar las claves
-sign: indica que se va a firmar

-in: archivo a firmar
-out: nombre del archivo que tendrá la firma
-inkey: la privada del usuario que quiere firmar

Contenido

mensaje.in

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

mensaje.sign

```
3046 0221 0082 a4b0 5cd8 7805 25b6 8a4a
a57c 630a 3607 1c99 0876 d51a 8c02 6bb1
5531 3fa1 e402 2100 d4f5 f132 e59c c544
0a4f bf5b 8833 1b93 c352 e7ce 92bb aeb1
b948 fc38 9904 281f
```

7. Dar un falso positivo

A continuación intentaremos verificar dos mensajes distintos y que openssl nos diga que son iguales.

Para esto, se ha creado un nuevo archivo llamado mensaje2.in.

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0973 6f0e b07c d3f6 2ce3 bd95 7d31 fee9
757f 5512 5c86 c0c9 7436 fdbc 054c 297e
b163 3bbe 316d 10c6 14e4 1aba d4fd 6c54
f1c8 bc73 fdfe 46e2 93a7 7e4f db36 cdab
44e1 8798 3578 4fbc e048 9d4d 2f4c f836
8bbd e42e 04bf a782 ec94 27ca de7e 5394
2131 8595 6f94 cf37 26ea a5fa e521 a65a
e44f dc2c 60cc 9705 4449 2675 9823 4d1b
a515 33ba 7b88 ffb1 d069 36fb 54b4 904d
9861 5d19 bb94 d0b3 7fa5 4354 386c 961c
4540 d89c 167e 63e1 f896 3922 e72d 57da
d9f6 7f3e d53f bef1 bde8 d8d5 50b0 10bc
cbec 86d0 d06d d3f9 657e 908f fd0a 08d2
dcf3 1022 45e0 be9f 2640 fb15 eff7 f979
453b 6710 78c2 3030 3758 9594 64aa 8a84
c96e 398c 1219 4a2f 9284 d3e5 73e9 007b
2d7f 07f3 9d87 eb85 ad84 661e 4bbd ff79
251d 4418 8caf e361 09f7 11ec 526c ac13
c557 229f 6db5 ea7e 53be 6828 840e d12c
a570 8ea6 93db 8661 1b14 58d8 13e8 04ad
c2eb 496b 1418 9948 6dfb 613e f4fb 24e1
96d0 b2a7 7159 c975 e8d2 5802 7a85 815c
aeb6 a0f5 76a1 b014 dd45 e78e 8c88 5476
ec66 fbee d8f9 5935 82e2 afd2 e064 86d6
4c46 b30e 3ce2 d1ea 2ca1
```

Como puede verse, los primeros bits del archivo son iguales a los de mensaje.in.

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl pkeyutl -verify -pubin -inkey pedroDSApub.pem
 -in mensaje2.in -sigfile mensaje.sign
Signature Verified Successfully
```

-verify: indica que se quiere verificar una firma
-pubin: porque se va a usar la pública del autor (Normal, se quiere verificar que el autor es el autor)
-inkey: para indicarle la clave a uar
-in: mensaje a verificar
-sigfile: firma del mensaje

Conclusiones

Se puede ver como al verificar el mensaje “mensaje2.sig” con la firma de “mensaje.in”, openssl nos dice que la firma es correcta.

Esto es debido a que al usar la orden **pkeyutl**, openssl **espera que se trabaje con el Hash** de los archivos y no con los archivos en sí, debido a esto, openssl trunca la entrada y como mensaje.in y mensaje2.in tienen los primeros bits iguales, el resultado es que openssl verifica una firma para un archivo diferente (¡Que, por ser, es hasta de mayor tamaño!)

Si se quiere trabajar con archivos directamente, se debe usar la orden **dgst**.

8. Generar hash

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl sha384 -hex -c -out pedroDSApub.sha384 pedroDS
Apub.pem; cat pedroDSApub.sha384
SHA384(pedroDSApub.pem)= c9:f1:66:04:98:ca:c2:29:aa:9a:9f:b5:9d:4a:5d:ea:33:97:eb:b5:43:ca:58:10:3a:4
f:7a:4d:71:2f:bc:e8:6a:03:d8:8f:49:63:4d:a1:33:0d:58:76:2e:8e:70:a7
```

Sha384: Es el tipo de hash a usar
-hex: porque se quiere en hexadecimal
-c: Para que el texto esté separado por dos puntos (:)
-out: nombre del archivo con el hash
Finalmente, el nombre del archivo al que se le quiere calcular el hash.

Archivo pedroDSApub.sha384

```
1  SHA384(pedroDSApub.pem)= c9:f1:66:04:98:ca:c2:29:aa:9a:9f:b5:9d:4a:5d:ea:33:97:eb:b5:43:ca:58:10
   :3a:4f:7a:4d:71:2f:bc:e8:6a:03:d8:8f:49:63:4d:a1:33:0d:58:76:2e:8e:70:a7
```

9. Generar hash de 160 bits

Para generar un hash de 160 bits se va a usar el algoritmo hash sha1, el cual da un tamaño de 160 bits

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl sha1 -binary -out mensaje2.sha1 mensaje2.in
```

Sha1: para obtener una salida fijada a 128 bits
-binary: porque se quiere vinario

Salida

```
1  É@|yø<0x8d>µœ¥´<0x11>õ,@<0x02>ñäLÝg
```

10. Firmar mensaje 2 usando dgst

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dgst -sha1 -sign pedroDSApriv.pem -out mensaje2.sign mensaje2.in
Enter pass phrase for pedroDSApriv.pem:
```

Dgst: indica que voy a firmar o verificar firmas de **archivos**, por lo que no es necesario hacer un Hash previo.
-sha1: para usar la función hash sha1
-sign pedroDSApriv.pem: para indicar que vamos a firmar con la clave pedroDSApriv.pem
-out: nombre del archivo de salida
Y finalmente el archivo a firmar.

Contenido

```
1  3045 0220 20f9 1ec2 81c8 ccee 7470 ffc8
2  639c 9c46 02ae 98fa 7738 d63f daa9 4f7e
3  9dc4 c750 0221 00ca 5718 b350 07a3 90da
4  5a9a de14 643b 995a 37c1 2001 733f 5b8a
5  138b 6d32 7a58 00
```

11. Verificar los mensajes “mensaje” y “mensaje2” con “mensaje2.sign”

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dgst -sha1 -verify pedroDSApub.pem -signature mensaje2.sign mensaje2.in
Verified OK
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dgst -sha1 -verify pedroDSApub.pem -signature mensaje2.sign mensaje.in
Verification Failure
```

-sha1: función hash a utilizar

-verify: clave pública con la que se quiere verificar

-signature: firma del mensaje a verificar

Finalmente se pasa el mensaje original .

Explicación

Como se puede ver, ahora **el resultado es correcto**, verificando la firma únicamente el mensaje2.

Esto es debido a que el comando **dgst está pensado para trabajar con archivos** y no con funciones hash, por eso tal y como se puede ver a la hora de introducir el comando, se hace un hash previo al archivo, de modo que la firma se hace al hash del archivo y no a los primeros bits de este como ocurría con **pkeyutl,** ya que dicho comando **está pensado para trabajar con funciones hash** directamente.

12. Verificación de "mensaje2" con "mensaje2.sign" usando la orden pkeyutl en vez de dgst

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl pkeyutl -verify -pubin -inkey pedroDSApub.pem -in mensaje2.in -sigfile mensaje
2.sign
Signature Verification Failure
```

Explicación

Nuevamente nos da error de verificación, esto es debido a que como ya hemos dicho pkeyutl esperas hash y se le ha pasado un archivo.

Si se quiere verificar, había que hacerle el mismo hash usado en el comando dgst a mensaje2 y hacer la verificación.

Como se dispone de dicho archivo, se prueba a verificar con el hash.

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl pkeyutl -verify -pubin -inkey pedroDSApub.pem -in mensaje2.sha1 -sigfile mensa
je2.sign
Signature Verified Successfully
```

Y como se puede ver, ahora el resultado es correcto.

13. Valor HMAC de shared.pem

Comando

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dgst -sha1 -hmac 12345 sharedDSA.pem
HMAC-SHA1(sharedDSA.pem)= 5a5536f5ad1cb04b3ef61042ec98be3342a44ce7
```

-hmac: para indicar que queremos calcular un hmac y la clave a usar

14. Protocolo estación a estación

Suponemos que tanto Alice como Bob comparten los parámetros de la curva elíptica:

Archivos de partida:

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ ls alice/
fuertesDSApub.pem  pedroDSAkey.pem  pedroDSApriv.pem  pedroDSApub.pem  stdECparam.pem
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ la bob/
fuertesDSAkey.pem  fuertesDSApriv.pem  fuertesDSApub.pem  pedroDSApub.pem  stdECparam.pem
```

1. Alicia calcula una clave pública y privada en función de los parámetros compartidos de la curva elíptica y envía la clave pública a Bob

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl ecparam -name prime192v1 -genkey -out alice/pe
droECkey.pem
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl ec -pubout -in alice/pedroECkey.pem -out alice
/pedroECpub.pem
read EC key
writing EC key
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cp alice/pedroECpub.pem bob/
```

2. Bob calcula una clave pública y privada

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl ecparam -name prime192v1 -genkey -out bob/fuer
tesECkey.pem
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl ec -pubout -in bob/fuertesECkey.pem -out bob/f
uertesECpub.pem
read EC key
writing EC key
```

3. Bob genera la clave derivada con la pública recibida por Alice y la privada generada por él

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl pkeyutl -derive -inkey bob/fuertesECkey.pem -p
eerkey bob/pedroECpub.pem -out key.bin
```

4. Bob concatena la pública con la pública de Alice, las firma con su privada (No la generada), y las cifra con la clave derivada generada. Finalmente envía la firma cifrada y la pública generada a Alice

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cat bob/fuertesECpub.pem >> bob/bobKeys.txt
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cat bob/pedroECpub.pem >> bob/bobKeys.txt
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cat bob/bobKeys.txt
```

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dgst -sha1 -sign bob/fuertesDSApriv.pem -out b
ob/bobKeys.sing bob/bobKeys.txt
Enter pass phrase for bob/fuertesDSApriv.pem:
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl aes-128-cfb8 -pass file:bob/key.bin -in bob/bo
bKeys.sing  -out bob/bobKeys.sing.enc
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cp bob/bobKeys.sing.enc alice/
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cp bob/fuertesECpub.pem alice/
```

5. Alice genera la clave derivada

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl pkeyutl -derive -inkey alice/pedroECkey.pem -p
eerkey alice/fuertesECpub.pem -out alice/key.bin
```

6. Alice descifra las firmas y las verifica con la clave pública de Bob

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl aes-128-cfb8 -d -pass file:alice/key.bin -in a
lice/bobKeys.sing.enc  -out alice/bobKeys.sing
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dgst -sha1 -verify alice/fuertesDSApub.pem -si
gnature alice/bobKeys.sing alice/bobKeys.txt
Verified OK
```

7. Alice concatena la pública generada por ella y la de bob, las firma con su privada (no la generada), lo cifra con la clave derivada y se lo envía a Bob

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cat alice/pedroECpub.pem >> alice/aliceKeys.txt
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cat alice/fuertesECpub.pem >> alice/aliceKeys.txt
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cat alice/aliceKeys.txt
-----BEGIN PUBLIC KEY-----
MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEozF2lvsIC5+XvEp8g9FRrslJhOq8
5c8ZpCnQ+ZYuM7Ij3Bv/cbTqX2xpq2bHZLPO
-----END PUBLIC KEY-----
-----BEGIN PUBLIC KEY-----
MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEsWBdr+HMEj0bnME7QAyG4mMnQfEu
f6sFKV5Dl2DdC4MadxuUBJVkSg8uJktqgoQF
-----END PUBLIC KEY-----
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dgst -sha1 -sign alice/pedroDSApriv.pem -out a
lice/aliceKeys.sing  alice/aliceKeys.txt
Enter pass phrase for alice/pedroDSApriv.pem:
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl aes-128-cfb8  -pass file:alice/key.bin -in ali
ce/aliceKeys.sing  -out alice/aliceKeys.sing.enc
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cp alice/aliceKeys.sing.enc bob/
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$
```

8. Bob desencripta con la clave derivada y verifica la firma con la pública de Alice

```
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl aes-128-cfb8 -d -pass file:bob/key.bin -in bob
/aliceKeys.sing.enc  -out bob/aliceKeys.sing
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cat bob/pedroECpub.pem >> bob/aliceKeys.txt
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ cat bob/fuertesECpub.pem >> bob/aliceKeys.txt
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$ openssl dgst -sha1 -verify bob/pedroDSApub.pem -signat
ure bob/aliceKeys.sing bob/aliceKeys.txt
Verified OK
pedro@ubuntu:~/Documents/git/inform-4-SPSI/p3$
```

9. Conclusiones

Este sistema permite, por un lado, verificar que Alice y Bob, son quienes dicen ser, ya que en el paso 6 Alice comprueba la firma de Bob y en el paso 8 Bob verifica la de Alice.

Por otro lado, permite generar una clave de sesión que se puede usar para usar un cifrado simétrico, mucho más rápido y eficiente que el asimétrico, e intercambiarla en un canal no seguro.

Esto se consigue gracias a la generación de una clave derivada, la cual únicamente necesita la clave pública generada por el otro, de esta manera, aunque haya alguien escuchando en el canal, al no disponer de las privadas, no podrá obtener la clave derivada y por tanto no podrá descifrar la comunicación.