

SEGURIDAD Y PROTECCIÓN DE SISTEMAS INFORMÁTICOS (2017-2018)  
GRADO EN INGENIERÍA INFORMÁTICA  
UNIVERSIDAD DE GRANADA

---

# Protocolos Criptográficos

---

Juan Ramón Gómez Berzosa

27 de noviembre de 2017

## Índice

1	Generad un archivo sharedDSA.pem que contenga los parámetros. Mostrad los valores.	4
2	Generad dos parejas de claves para los parámetros anteriores. Las claves se almacenarán en los archivos nombreDSAkey.pem y apellidoDSAkey.pem. No es necesario protegerlas por contraseña.	5
3	"Extraed" la clave privada contenida en el archivo nombreDSAkey.pem a otro archivo que tenga por nombre nombreDSApriv.pem. Este archivo debe de estar protegido por contraseña. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem.	5
4	Extraed en nombreDSAPub.pem la clave pública contenida en el archivo nombreDSAkey.pem. De nuevo nombreDSAPub.pem no debe estar cifrado ni protegido. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem	7
5	Calculad el valor hash del archivo con la clave pública nombreDSAPub.pem usando sha384 con salida hexadecimal con bloques de dos caracteres separados por dos puntos. Mostrad los valores por salida estándar y guardadlo en nombreDSAPub.sha384	9
6	Calculad el valor hash del archivo con la clave pública apellidoDSAPub.pem usando una función hash de 160 bits con salida binaria. Guardad el hash en apellidoDSAPub.[algoritmo] y mostrad su contenido.	10
7	Generad el valor HMAC del archivo sharedDSA.pem con clave '12345' mostrándolo por pantalla	10
8	Simulad una ejecución completa del protocolo Estación-Estación. Para ello emplearemos como claves para firma/verificación las generadas en esta práctica, y para el protocolo DH emplearemos las claves asociadas a curvas elípticas de la práctica anterior junto con las de otro usuario simulado que deberéis generar nuevamente. Por ejemplo, si mi clave privada está en javierECpriv.pem y la clave pública del otro usuario está en lobilloECpub.pem, el comando para generar la clave derivada será:  # openssl pkeyutl -inkey javierECpriv.pem -peerkey lobilloECpub.pem -derive -out key.bin El algoritmo simétrico a utilizar en el protocolo estación a estación será AES-128 en modo CFB8.	11
8.1	Primera etapa . . . . .	14
8.2	Segunda etapa . . . . .	14
8.3	Tercera etapa . . . . .	16

8.4	Cuarta etapa . . . . .	19
8.5	Quinta etapa . . . . .	20

## Índice de figuras

1.1.	Parámetros DSA . . . . .	4
1.2.	Generación de parámetros DSA . . . . .	4
3.1.	Clave privada JuanRamonDSApriv.pem . . . . .	6
3.2.	Clave privada GomezBerzosaDSApriv.pem . . . . .	7
4.1.	Clave publica JuanRamonDSAPub.pem . . . . .	8
4.2.	Clave publica GomezBerzosaDSAPub.pem . . . . .	9
5.1.	Valor hash para la clave publica JuanRamonDSAPub.pem . . . . .	10
6.1.	Valor hash para la clave publica GomezBerzosaDSAPub.pem . . . . .	10
7.1.	Valor HMAC para el archivo sharedDSA.pem . . . . .	11
8.1.	Clave pública EC JuanRamonECpriv.pem . . . . .	12
8.2.	Clave pública EC JuanRamonECpub.pem . . . . .	12
8.3.	Clave privada EC GomezBerzosaECpriv.pem . . . . .	13
8.4.	Clave pública EC GomezBerzosaECpub.pem . . . . .	13
8.5.	Resumen del protocolo estación a estación. . . . .	14
8.6.	Clave derivada GomezBerzosa (Bernabé) . . . . .	15
8.7.	Concatenación $c  d$ . . . . .	15
8.8.	Firma $s$ . . . . .	16
8.9.	Firma $s$ encriptada. . . . .	16
8.10.	Clave derivada JuanRamon (Alicia) . . . . .	17
8.11.	Firma $s$ descriptada . . . . .	17
8.12.	Verificación de la firma $s$ . . . . .	18
8.13.	Concatenación $d  c$ . . . . .	18
8.14.	Firma $t$ . . . . .	18
8.15.	Cifrado de la firma $t$ . . . . .	19
8.16.	Firma $t$ descriptada . . . . .	19
8.17.	Verificación de $t$ . . . . .	20

## Índice de tablas

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

En primer lugar usaremos el comando **dsaparam**, el cual nos permite generar y manipular los parámetros asociados al algoritmo DSA: primos  $p$  y  $q$  y el generador  $g$ . También permite generar la pareja de claves pública y privada, sin embargo lo realizaremos con otra orden más adelante.

La orden sería la siguiente:

```
# openssl dsaparam -out sharedDSA.pem 768
```

El resultado será la generación de los parámetros necesarios para la posterior obtención de claves DSA usando dichos parámetros con un tamaño 768 bits. Estos son los valores obtenidos en hexadecimal ordenados en bloques de 2 caracteres separados por dos puntos:

```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl dsaparam -in share
DSA.pem -text -noout
P:
00:b9:c8:de:c2:aa:f0:93:4a:f9:ba:6e:26:14:9c:
b1:2f:aa:a2:a8:34:e7:3e:da:a1:7b:db:29:f4:f6:
20:74:6d:a0:13:2c:fc:94:e9:53:e5:77:ac:33:73:
4b:79:2f:3d:c0:ce:f8:58:08:89:5f:57:1b:97:2f:
3c:ad:11:ec:35:07:44:59:d9:37:9a:5c:26:5f:db:
47:31:80:d5:fd:f8:b7:a2:18:39:82:f7:29:16:3e:
0f:9e:bf:d8:d4:7c:31
Q:
00:e7:ea:b8:a2:66:c2:68:9c:b7:05:9a:16:06:96:
37:66:79:d1:74:83
G:
00:a6:b3:a0:58:6d:bc:04:c5:35:41:26:85:52:f0:
5e:1a:cf:d6:f5:22:3d:09:7d:08:5d:14:3e:ed:07:
9c:c4:f3:93:74:26:f1:d5:1d:53:7c:14:f1:3c:92:
d5:2f:45:b6:d3:28:ca:ad:85:4b:3e:f4:44:f4:7f:
e8:be:66:78:27:25:dc:fd:bc:1e:c2:a4:02:f4:9c:
29:4e:b5:d1:2d:02:f9:ee:f0:dc:7b:a2:1a:74:82:
fc:e8:9c:23:33:78:da
```

Figura 1.1: Parámetros DSA

Si observamos la consola cuando ejecutamos el comando obtendremos algo así:

```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl dsaparam -out sha256DSA.pem 768
Generating DSA parameters, 768 bit long prime
This could take some time
.....+.....+.+++++
+++++*
```

Figura 1.2: Generación de parámetros DSA

Si nos fijamos, aparecen una serie de "." y símbolos "+" . Esto es una indicación del programa para decirnos que está intentando encontrar un primo  $q$  y luego un primo  $p$  válidos, así como el generador  $g$ . Este proceso será más largo cuanto mayor tamaño establezcamos.

## 2. **Generad dos parejas de claves para los parámetros anteriores. Las claves se almacenarán en los archivos nombreDSAkey.pem y apellidoDSAkey.pem. No es necesario protegerlas por contraseña.**

Para generar las parejas de claves vamos a emplear el comando de openssl **gendsa**, el cual nos permite generar las parejas de claves DSA a partir de un archivo de parámetros existente, en nuestro caso el archivo de parámetros lo hemos generado en el apartado anterior (sharedDSA.pem).

La generación de claves se podría haber hecho junto a la generación de los parámetros DSA, sin embargo en nuestro caso lo hemos realizado en dos pasos distintos porque queremos que dos personas tengan sus parejas de claves asociadas a los mismos parámetros ya que serán empleadas para un protocolo de intercambio de claves.

En primer lugar generaremos el par para Juan Ramón:

```
# openssl gensa -out JuanRamonDSAkey.pem sharedDSA.pem
```

En primer lugar generaremos el par para GómezBerzosa:

```
# openssl gensa -out GomezBerzosaDSAkey.pem sharedDSA.pem
```

Una vez hecho esto ya tenemos generadas nuestras parejas de claves públicas y privadas para su manipulación.

## 3. **"Extraed" la clave privada contenida en el archivo nombreDSAkey.pem a otro archivo que tenga por nombre nombreDSApriv.pem. Este archivo debe de estar protegido por contraseña. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem.**

Al igual que en prácticas anteriores, vamos a "extraer" de las parejas de claves DSAkey la parte privada. Decimos "extraer" porque en realidad lo único que estamos haciendo es cifrar el fichero (la clave) con un algoritmo simétrico y usando una contraseña específica. Como cifrado simétrico utilizaremos aes en su modo de 128 bits y como contraseña: 0123456789.

Para realizar el proceso que hemos mencionado anteriormente, vamos a usar el comando **dsa** de OpenSSL, el cual permite la sustracción de claves públicas y privadas, conversiones de formato, adición o supresión de contraseña... etc, funciona de manera similar a los comandos rsa y ec.

Ahora vamos a proceder a extraer y cifrar la clave privada del archivo "JuanRamonDSAkey.pem":

```
# openssl dsa -in JuanRamonDSAkey.pem -out JuanRamonDSApriv.pem -aes128
```

Vamos a ver que valores se han generado:

```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl dsa -in JuanRamonDSApriv.pem -text -noout
read DSA key
Enter pass phrase for JuanRamonDSApriv.pem:
Private-Key: (768 bit)
priv:
    00:8a:3e:fa:b6:83:29:14:c4:27:48:43:c6:0d:15:
    3e:2b:0c:13:ca:66
pub:
    00:80:45:1d:54:a6:0c:0f:0c:1e:56:76:c7:df:5b:
    ea:0f:64:e2:b5:15:3e:1b:df:bb:22:6e:e4:39:85:
    2f:8f:e9:71:26:54:bd:09:75:52:7e:26:7e:81:bb:
    82:52:d1:f1:fd:9e:17:b5:55:bd:6e:74:dc:c8:72:
    b9:fc:8f:a1:e3:d8:d1:63:e8:73:65:e3:4d:54:3e:
    55:0d:03:d9:5a:e6:eb:c6:16:bc:7b:d7:1b:93:8c:
    19:e4:27:57:c8:84:7f
P:
    00:b9:c8:de:c2:aa:f0:93:4a:f9:ba:6e:26:14:9c:
    b1:2f:aa:a2:a8:34:e7:3e:da:a1:7b:db:29:f4:f6:
    20:74:6d:a0:13:2c:fc:94:e9:53:e5:77:ac:33:73:
    4b:79:2f:3d:c0:ce:f8:58:08:89:5f:57:1b:97:2f:
    3c:ad:11:ec:35:07:44:59:d9:37:9a:5c:26:5f:db:
    47:31:80:d5:fd:f8:b7:a2:18:39:82:f7:29:16:3e:
    0f:9e:bf:d8:d4:7c:31
Q:
    00:e7:ea:b8:a2:66:c2:68:9c:b7:05:9a:16:06:96:
    37:66:79:d1:74:83
G:
    00:a6:b3:a0:58:6d:bc:04:c5:35:41:26:85:52:f0:
    5e:1a:cf:d6:f5:22:3d:09:7d:08:5d:14:3e:ed:07:
    9c:c4:f3:93:74:26:f1:d5:1d:53:7c:14:f1:3c:92:
    d5:2f:45:b6:d3:28:ca:ad:85:4b:3e:f4:44:f4:7f:
    e8:be:66:78:27:25:dc:fd:bc:1e:c2:a4:02:f4:9c:
    29:4e:b5:d1:2d:02:f9:ee:f0:dc:7b:a2:1a:74:82:
    fc:e8:9c:23:33:78:da
```

Figura 3.1: Clave privada JuanRamonDSApriv.pem

noindent Ahora vamos a proceder a extraer y cifrar la clave privada del archivo "GomezBerzosaDSAkey.pem":

```
# openssl dsa -in GomezBerzosaDSAkey.pem -out GomezBerzosaDSApriv.pem -aes128
```

Vamos a ver que valores se han generado:

```

jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl dsa -in GomezBerzosaDSApriv.pem -text -noout
read DSA key
Enter pass phrase for GomezBerzosaDSApriv.pem:
Private-Key: (768 bit)
priv:
  00:e6:e1:b9:0f:31:e4:fc:66:82:62:c2:5e:65:d4:
  7d:a7:5b:cb:93:0b
pub:
  29:85:f3:26:52:f4:c9:48:a2:b4:4e:89:1a:96:b7:
  09:7f:2d:09:fe:25:f5:d1:78:a8:bd:f7:0b:9f:06:
  ae:61:82:98:00:4c:28:13:9a:56:d2:63:aa:9f:3d:
  30:b5:52:72:49:9b:58:3b:e6:ae:e3:f5:b8:10:94:
  46:f3:0a:3c:37:ac:41:2d:88:cf:85:b1:39:b6:3f:
  85:95:47:49:2b:4f:5f:8c:4c:36:de:87:6a:11:ac:
  f9:bc:82:54:24:84
P:
  00:b9:c8:de:c2:aa:f0:93:4a:f9:ba:6e:26:14:9c:
  b1:2f:aa:a2:a8:34:e7:3e:da:a1:7b:db:29:f4:f6:
  20:74:6d:a0:13:2c:fc:94:e9:53:e5:77:ac:33:73:
  4b:79:2f:3d:c0:ce:f8:58:08:89:5f:57:1b:97:2f:
  3c:ad:11:ec:35:07:44:59:d9:37:9a:5c:26:5f:db:
  47:31:80:d5:fd:f8:b7:a2:18:39:82:f7:29:16:3e:
  0f:9e:bf:d8:d4:7c:31
Q:
  00:e7:ea:b8:a2:66:c2:68:9c:b7:05:9a:16:06:96:
  37:66:79:d1:74:83
G:
  00:a6:b3:a0:58:6d:bc:04:c5:35:41:26:85:52:f0:
  5e:1a:cf:d6:f5:22:3d:09:7d:08:5d:14:3e:ed:07:
  9c:c4:f3:93:74:26:f1:d5:1d:53:7c:14:f1:3c:92:
  d5:2f:45:b6:d3:28:ca:ad:85:4b:3e:f4:44:f4:7f:
  e8:be:66:78:27:25:dc:fd:bc:1e:c2:a4:02:f4:9c:
  29:4e:b5:d1:2d:02:f9:ee:f0:dc:7b:a2:1a:74:82:
  fc:e8:9c:23:33:78:da

```

Figura 3.2: Clave privada GomezBerzosaDSApriv.pem

Como observamos en ambas impresiones de valores de las claves, se pueden observar como están también la parte pública así como los parámetros.

#### 4. Extraed en nombreDSAPub.pem la clave pública contenida en el archivo nombreDSAkey.pem. De nuevo nombreDSAPub.pem no debe estar cifrado ni protegido. Mostrad sus valores. Lo mismo para el archivo apellidoDSAkey.pem

En este apartado, al igual que en prácticas anteriores, vamos a extraer la clave pública. Como es lógico, esta parte es compartida y por tanto no tiene porque estar protegida por contraseña. Para realizar dicha extracción emplearemos el comando **dsa**.

Ahora vamos a proceder a extraer la clave publica del archivo "JuanRamonDSAkey.pem":

```
# openssl dsa -in JuanRamonDSAkey.pem -out JuanRamonDSAPub.pem -pubout
```

Vamos a ver que valores se han generado:

```

jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl dsa -pubin -in JuanRamonDSAPub.pem -text -noout
read DSA key
pub:
 00:80:45:1d:54:a6:0c:0f:0c:1e:56:76:c7:df:5b:
 ea:0f:64:e2:b5:15:3e:1b:df:bb:22:6e:e4:39:85:
 2f:8f:e9:71:26:54:bd:09:75:52:7e:26:7e:81:bb:
 82:52:d1:f1:fd:9e:17:b5:55:bd:6e:74:dc:c8:72:
 b9:fc:8f:a1:e3:d8:d1:63:e8:73:65:e3:4d:54:3e:
 55:0d:03:d9:5a:e6:eb:c6:16:bc:7b:d7:1b:93:8c:
 19:e4:27:57:c8:84:7f
P:
 00:b9:c8:de:c2:aa:f0:93:4a:f9:ba:6e:26:14:9c:
 b1:2f:aa:a2:a8:34:e7:3e:da:a1:7b:db:29:f4:f6:
 20:74:6d:a0:13:2c:fc:94:e9:53:e5:77:ac:33:73:
 4b:79:2f:3d:c0:ce:f8:58:08:89:5f:57:1b:97:2f:
 3c:ad:11:ec:35:07:44:59:d9:37:9a:5c:26:5f:db:
 47:31:80:d5:fd:f8:b7:a2:18:39:82:f7:29:16:3e:
 0f:9e:bf:d8:d4:7c:31
Q:
 00:e7:ea:b8:a2:66:c2:68:9c:b7:05:9a:16:06:96:
 37:66:79:d1:74:83
G:
 00:a6:b3:a0:58:6d:bc:04:c5:35:41:26:85:52:f0:
 5e:1a:cf:d6:f5:22:3d:09:7d:08:5d:14:3e:ed:07:
 9c:c4:f3:93:74:26:f1:d5:1d:53:7c:14:f1:3c:92:
 d5:2f:45:b6:d3:28:ca:ad:85:4b:3e:f4:44:f4:7f:
 e8:be:66:78:27:25:dc:fd:bc:1e:c2:a4:02:f4:9c:
 29:4e:b5:d1:2d:02:f9:ee:f0:dc:7b:a2:1a:74:82:
 fc:e8:9c:23:33:78:da

```

Figura 4.1: Clave publica JuanRamonDSAPub.pem

Ahora vamos a proceder a extraer la clave publica del archivo "GomezBerzosaDSA-key.pem":

```
# openssl dsa -in GomezBerzosaDSAkey.pem -out GomezBerzosaDSAPub.pem -pubout
```

Vamos a ver que valores se han generado:



```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl dsa -pubin -in G
omezBerzosaDSAPub.pem -text -noout
read DSA key
pub:
29:85:f3:26:52:f4:c9:48:a2:b4:4e:89:1a:96:b7:
09:7f:2d:09:fe:25:f5:d1:78:a8:bd:f7:0b:9f:06:
ae:61:82:98:00:4c:28:13:9a:56:d2:63:aa:9f:3d:
30:b5:52:72:49:9b:58:3b:e6:ae:e3:f5:b8:10:94:
46:f3:0a:3c:37:ac:41:2d:88:cf:85:b1:39:b6:3f:
85:95:47:49:2b:4f:5f:8c:4c:36:de:87:6a:11:ac:
f9:bc:82:54:24:84
P:
00:b9:c8:de:c2:aa:f0:93:4a:f9:ba:6e:26:14:9c:
b1:2f:aa:a2:a8:34:e7:3e:da:a1:7b:db:29:f4:f6:
20:74:6d:a0:13:2c:fc:94:e9:53:e5:77:ac:33:73:
4b:79:2f:3d:c0:ce:f8:58:08:89:5f:57:1b:97:2f:
3c:ad:11:ec:35:07:44:59:d9:37:9a:5c:26:5f:db:
47:31:80:d5:fd:f8:b7:a2:18:39:82:f7:29:16:3e:
0f:9e:bf:d8:d4:7c:31
Q:
00:e7:ea:b8:a2:66:c2:68:9c:b7:05:9a:16:06:96:
37:66:79:d1:74:83
G:
00:a6:b3:a0:58:6d:bc:04:c5:35:41:26:85:52:f0:
5e:1a:cf:d6:f5:22:3d:09:7d:08:5d:14:3e:ed:07:
9c:c4:f3:93:74:26:f1:d5:1d:53:7c:14:f1:3c:92:
d5:2f:45:b6:d3:28:ca:ad:85:4b:3e:f4:44:f4:7f:
e8:be:66:78:27:25:dc:fd:bc:1e:c2:a4:02:f4:9c:
29:4e:b5:d1:2d:02:f9:ee:f0:dc:7b:a2:1a:74:82:
fc:e8:9c:23:33:78:da
```

Figura 4.2: Clave publica GomezBerzosaDSAPub.pem

## 5. Calculad el valor hash del archivo con la clave pública nombreDSAPub.pem usando sha384 con salida hexadecimal con bloques de dos caracteres separados por dos puntos. Mostrad los valores por salida estándar y guardadlo en nombreDSAPub.sha384

Para generar el valor hash de un fichero vamos a emplear el comando **dgst**. Este comando permite generar el valor hash de un fichero tanto en formato hexadecimal como en binario, por defecto lo genera en hexadecimal. También sirve para generar el HMAC y para firmar y verificar un fichero.

Ahora vamos a generar el valor hash de la clave pública de JuanRamon:

```
# openssl dgst -sha384 -c -hex -out JuanRamonDSAPub.sha384 JuanRamonDSAPub.pem
```

Como podemos observar, hemos añadido la opción **-c** ya que esto obliga a que la salida generada se organicen los valores en bloques de dos caracteres separados por dos puntos. La función hash que hemos empleado es sha384, como nos han pedido en el enunciado. A continuación podemos el valor hash obtenido:

```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ cat JuanRamonDSAPub.sha384
SHA384(JuanRamonDSAPub.pem)= 12:b3:54:b9:29:02:7b:97:fc:82:d9:49:30:5c:07:a3:c8:
d9:d2:c9:09:b7:ec:fc:4d:a0:6c:86:1e:e9:87:85:35:d0:76:1c:19:83:63:75:50:91:4f:1d
:16:b6:37:31
```

Figura 5.1: Valor hash para la clave publica JuanRamonDSAPub.pem

Como podemos observar, los datos están organizados como nos pedían en bloques de dos caracteres separados por dos puntos.

## 6. Calculad el valor hash del archivo con la clave pública apellidoDSAPub.pem usando una función hash de 160 bits con salida binaria. Guardad el hash en apellidoDSAPub.[algoritmo] y mostrad su contenido.

En este caso tenemos que elegir una función hash que tenga como salida 160 bits, para realizar el ejercicio utilizaremos **sha1**.

Ahora vamos a generar el valor hash de la clave pública de GomezBerzosa con el algoritmo de salida 160 bits:

```
# openssl dgst -sha1 -binary -out GomezBerzosaDSAPub.sha1 GomezBerzosaDSAPub.pem
```

A continuación podemos ver el valor hash en binario:

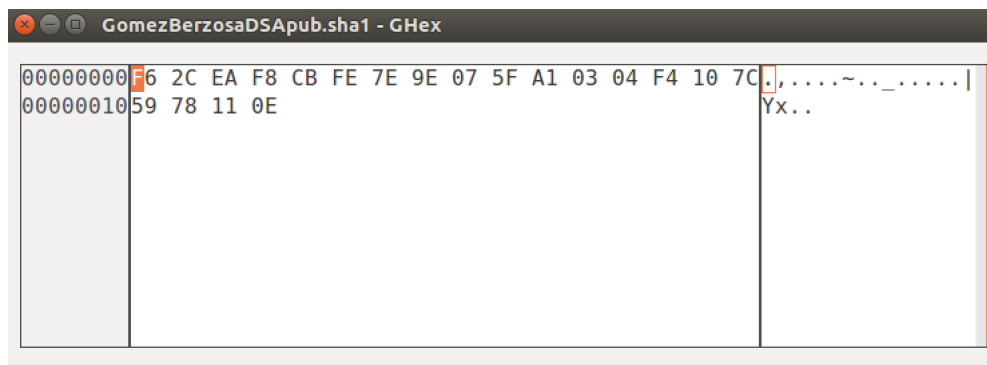


Figura 6.1: Valor hash para la clave publica GomezBerzosaDSAPub.pem

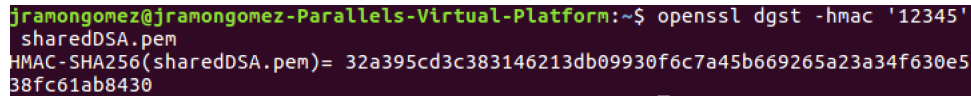
## 7. Generad el valor HMAC del archivo sharedDSA.pem con clave '12345' mostrándolo por pantalla

Para generar el código de autenticación de mensajes (HMAC) vamos a emplear el comando mencionado anteriormente de OpenSSL, **dgst**, con la opción **"-hmac"**. Como clave emplearemos la cadena **'12345'**.

La orden será la siguiente:

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

La salida de dicha orden y el correspondiente valor hmac del archivo sería la siguiente:



```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl dgst -hmac '12345'
sharedDSA.pem
HMAC-SHA256(sharedDSA.pem)= 32a395cd3c383146213db09930f6c7a45b669265a23a34f630e5
b8fc61ab8430
```

Figura 7.1: Valor HMAC para el archivo sharedDSA.pem

8. **Simulad una ejecución completa del protocolo Estación-Estación.** Para ello emplearemos como claves para firma/verificación las generadas en esta práctica, y para el protocolo DH emplearemos las claves asociadas a curvas elípticas de la práctica anterior junto con las de otro usuario simulado que deberéis generar nuevamente. Por ejemplo, si mi clave privada está en `javierECpriv.pem` y la clave pública del otro usuario está en `lobilloECpub.pem`, el comando para generar la clave derivada será:

```
# openssl pkeyutl -inkey javierECpriv.pem -peerkey lobilloECpub.pem -derive -out key.bin
```

**El algoritmo simétrico a utilizar en el protocolo estación a estación será AES-128 en modo CFB8.**

En primer lugar vamos a recopilar las claves que vamos a utilizar a lo largo del protocolo estación a estación. En nuestro caso nuestros participantes van a ser JuanRamon y GomezBerzosa (Alicia y Bernabé respectivamente si hacemos una analogía con el protocolo Diffie-Helman y el protocolo Estación a Estación).

Las claves que utilizaremos para la firma y verificación serán las parejas que hemos generado a lo largo de la práctica: `JuanRamonDSAPub.pem/JuanRamonDSAPriv.pem` y `GomezBerzosaDSAPub.pem/GomezBerzosaDSAPriv.pem`, las cuales se corresponderían con  $V_A$ ,  $S_A$  y  $V_B$ ,  $S_B$  respectivamente.

Las claves que utilizaremos para el protocolo Diffie-Helman serán las claves asociadas a curvas elípticas generadas en la práctica anterior. Sin embargo, tendremos que generar una nueva pareja de claves para GomezBerzosa a partir de los parámetros asociados a la curva elíptica que elegimos, "stdECparam.pem". Las claves públicas de ambos se corresponderán con  $c$  y  $d$ .

En primer lugar voy a mostrar la clave privada asociada a curvas elípticas de JuanRamon, cabe destacar que está cifrada con aes-128 y protegida con contraseña (0123456789):

```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl ec -in JuanRamon
ECpriv.pem -text -noout
read EC key
Enter PEM pass phrase:
Private-Key: (192 bit)
priv:
    00:ab:45:ef:87:73:51:f7:04:c0:e5:52:5f:1c:f4:
    9f:f8:e1:81:69:40:0c:2d:75:6f
pub:
    04:84:fa:f9:d4:0f:f4:9b:a5:0e:15:0b:5f:ee:2a:
    b9:10:17:17:9b:39:99:a0:d2:5a:94:45:95:05:43:
    8b:38:a6:3e:a9:61:0e:80:7c:be:47:99:24:d0:c3:
    6f:66:cc:2e
ASN1 OID: prime192v1
NIST CURVE: P-192
```

Figura 8.1: Clave pública EC JuanRamonECpriv.pem

Ahora mostraré la clave pública asociada a curvas elípticas de JuanRamon:

```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl ec -pubin -in Ju
anRamonECpub.pem -text -noout
read EC key
Private-Key: (192 bit)
pub:
    04:84:fa:f9:d4:0f:f4:9b:a5:0e:15:0b:5f:ee:2a:
    b9:10:17:17:9b:39:99:a0:d2:5a:94:45:95:05:43:
    8b:38:a6:3e:a9:61:0e:80:7c:be:47:99:24:d0:c3:
    6f:66:cc:2e
ASN1 OID: prime192v1
NIST CURVE: P-192
```

Figura 8.2: Clave pública EC JuanRamonECpub.pem

Ahora procederemos a generar la pareja de claves pública y privada de GomezBerzosa. Para ello en primer lugar extraeremos la pareja de claves publica/privada en el fichero GomezBerzosaECkey.pem asociadas a los parámetros de curvas elípticas almacenados en stdECparam.pem.

```
# openssl ecparam -in stdECparam.pem -out GomezBerzosaECkey.pem -genkey
```

Posteriormente, "extraeremos" la clave privada cifrándola con el algoritmo des3 y protegiéndola con la contraseña: 0123456789.

```
# openssl ec -in GomezBerzosaECkey.pem -des3 -out GomezBerzosaECpriv.pem
```

Los valores obtenidos son:

```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl ec -in GomezBerzosaECpriv.pem -text -noout
read EC key
Enter PEM pass phrase:
Private-Key: (192 bit)
priv:
    66:9d:d4:78:e7:da:c7:01:60:8a:ff:30:f3:0a:ff:
    87:e2:22:0b:8b:15:6b:80:5b
pub:
    04:28:49:73:11:81:4d:56:12:db:29:ca:24:23:85:
    fa:62:15:d5:02:9f:08:69:13:c5:f0:09:bc:2f:5d:
    e6:9f:48:ba:4c:0a:87:5e:b7:8e:5b:fc:83:41:51:
    73:81:27:2a
ASN1 OID: prime192v1
NIST CURVE: P-192
```

Figura 8.3: Clave privada EC GomezBerzosaECpriv.pem

Ahora extraeremos la parte pública de la clave:

```
# openssl ec -in GomezBerzosaECkey.pem -pubout -out GomezBerzosaECpub.pem
```

Los valores obtenidos son:

```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl ec -pubin -in GomezBerzosaECpub.pem -text -noout
read EC key
Private-Key: (192 bit)
pub:
    04:28:49:73:11:81:4d:56:12:db:29:ca:24:23:85:
    fa:62:15:d5:02:9f:08:69:13:c5:f0:09:bc:2f:5d:
    e6:9f:48:ba:4c:0a:87:5e:b7:8e:5b:fc:83:41:51:
    73:81:27:2a
ASN1 OID: prime192v1
NIST CURVE: P-192
```

Figura 8.4: Clave pública EC GomezBerzosaECpub.pem

Una vez hecho esto, ya tenemos todo listo para simular la ejecución del protocolo estación a estación. Este protocolo surge para mejorar las debilidades del protocolo de Diffie-Helman, ya que este no implicaba una autenticación en ambas partes y era vulnerable a ataques **man-in-the-middle**. Ambos son protocolos de intercambio de claves. El resumen del protocolo estación a estación es el siguiente:

### 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  $c = g^a \bmod p$  y lo envía a Bernabé.
- ② Bernabé escoge aleatoriamente  $1 \leq b \leq p-2$ , calcula  $d = g^b \bmod p$  y  $k = g^{ab} = (c)^b \bmod p$ . Calcula  $s = \text{sgn}_{s_B}(c || d)$ . Envía  $(d, e_k(s))$  a Alicia.
- ③ Alicia calcula  $k = g^{ab} = d^a \bmod p$  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.

Figura 8.5: Resumen del protocolo estación a estación.

Como hemos mencionado anteriormente nuestra analogía sería Alicia -> JuanRamon y Bernabé -> GomezBerzosa. Desglosaremos en 5 etapas los pasos necesarios para la simulación.

#### 8.1. Primera etapa

Alicia comparte su clave pública (c -> JuanRamonECpub.pem) con Bernabé.

#### 8.2. Segunda etapa

Bernabé genera su clave derivada usando el comando **pkeyutl** tal y como se dice en el enunciado del ejercicio, usando su clave privada y la clave pública de Alicia (claves asociadas a curvas elípticas):

```
# openssl pkeyutl -inkey GomezBerzosaECpriv.pem -peerkey JuanRamonECpub.pem -derive -out key2.bin
```

La clave resultante es la siguiente:

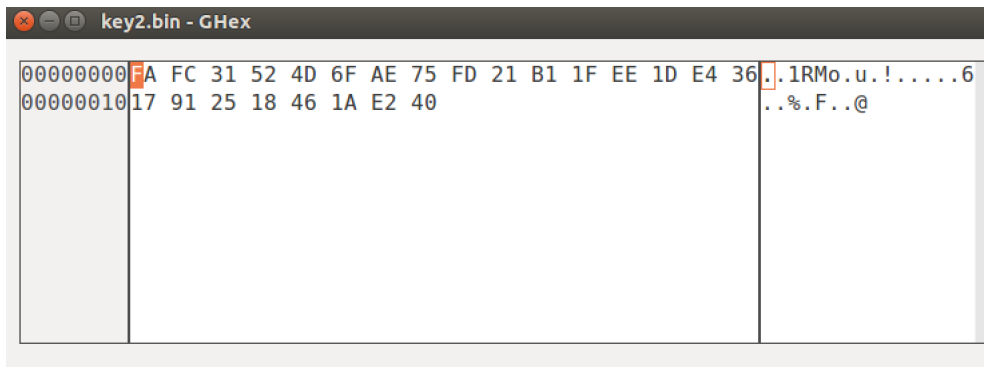


Figura 8.6: Clave derivada GomezBerzosa (Bernabé)

Una vez hecho esto, Bernabé hace la concatenación de la clave pública de Alicia (c -> JuanRamonECpub.pem) con su clave pública (d -> GomezBerzosaECpub.pem). Para ello usaremos el siguiente comando:

```
# cat JuanRamonECpub.pem GomezBerzosaECpub.pem >concatCD.txt
```

El resultado sería el siguiente:

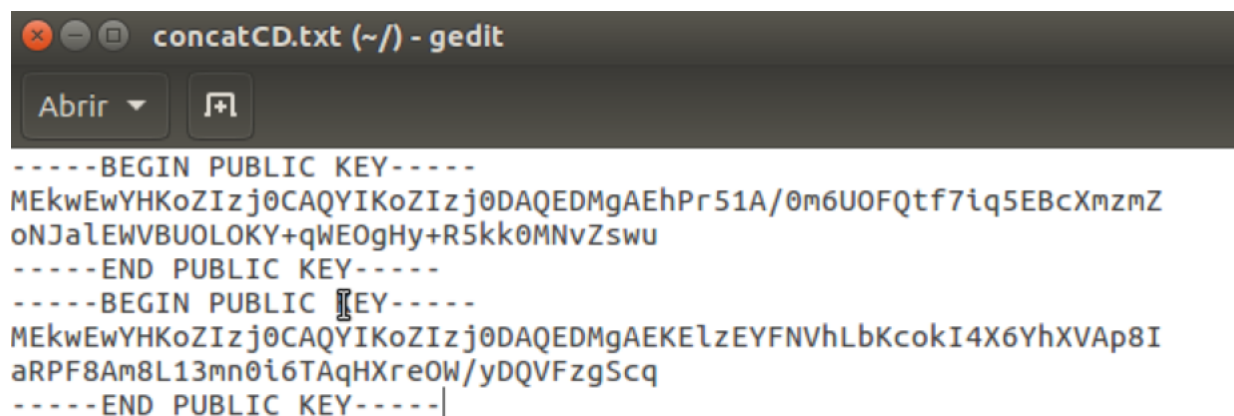


Figura 8.7: Concatenación c||d

Ahora Bernabé firmará el fichero concatCD, cifrando el valor hash del fichero con su clave privada DSA ( $S_B$  -> GomezBerzosaDSApriv.pem), el resultado lo guardaremos en firma\_s.sign (s). Para ello, usaremos el comando **dgst** con la opción "-sign":

```
# openssl dgst -sha256 -sign GomezBerzosaDSApriv.pem -out firma_s.sign concatCD.txt
```

La firma s tendría el siguiente aspecto:

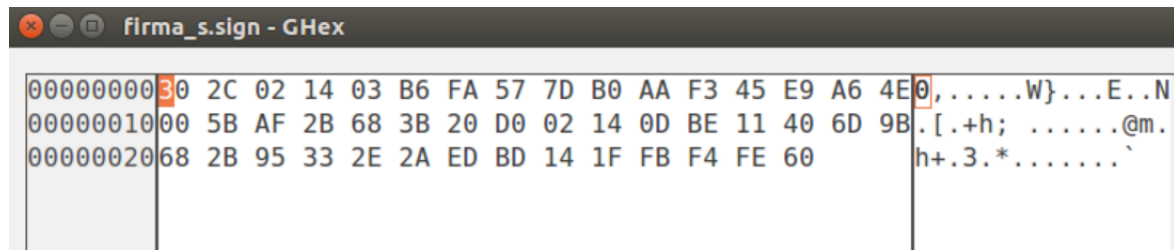


Figura 8.8: Firma s

Cabe destacar que como algoritmo hash hemos decidido utilizar -sha256. Ahora, Bernabé encriptará la firma para posteriormente compartirla con Alicia junto a su clave pública (d).

Para el cifrado del archivo utilizaremos un cifrado simétrico empleando el algoritmo criptográfico aes de 128 bits en su modo CFB8, utilizando como clave la clave derivada que hemos calculado anteriormente :

```
# openssl aes-128-cfb8 -pass file:key2.bin -in firma_s.sign -out firma_s_Encrypt.bin
```

El resultado del cifrado de la firma es:

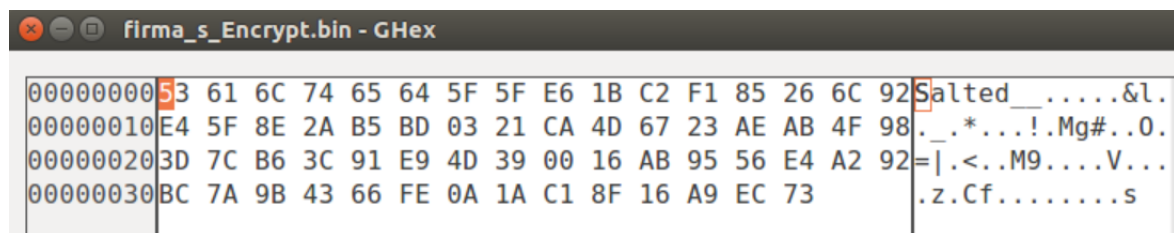


Figura 8.9: Firma s encriptada.

Ahora Bernabé le envía la firma s encriptada y su clave pública d a Alicia.

### 8.3. Tercera etapa

Alicia calcula en primer lugar su clave derivada al igual que hizo anteriormente Bernabé, aunque esta vez con la clave privada de Alicia y la clave pública de Bernabé.

```
# openssl pkeyutl -inkey JuanRamonECpriv.pem -peerkey GomezBerzosaECpub.pem -derive -out key1.bin
```

El resultado sería el siguiente:



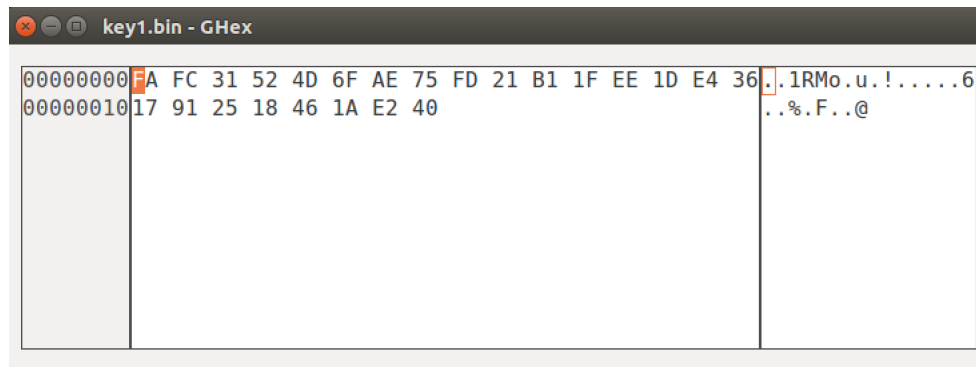


Figura 8.10: Clave derivada JuanRamon (Alicia)

Una vez hecho esto, Alicia procede a descifrar la firma s encriptada usando como clave la clave derivada que acabamos de crear:

```
# openssl aes-128-cfb8 -d -pass file:key1.bin -in firma_s_Encrypt.bin -out firma_s_Decrypt.bin
```

Si nos damos cuenta y como es lógico al ser un protocolo de intercambio de claves y estar intentando compartir la clave y que sólo ambos la conozcan, las claves derivadas (K1 y K2) son exactamente la misma. Como podemos comprobar la firma s descifrada coincide con la firma s original:

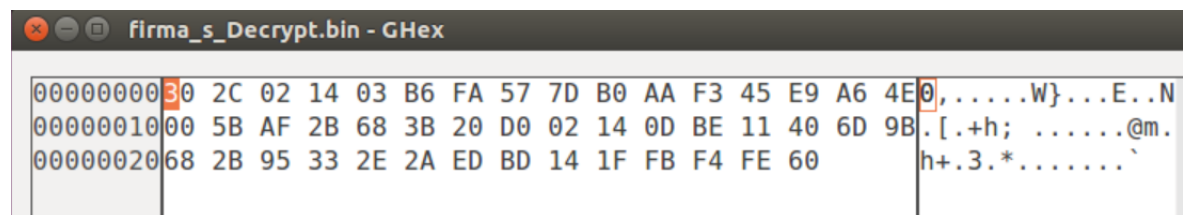


Figura 8.11: Firma s descifrada

Ahora Alicia volverá a concatenar las claves c y d (aunque ya no lo realizaremos porque lo hemos hecho antes) y verificará que el hash de dicho fichero y el de la firma descifrada con la clave pública de Bernabé (GomezBerzosaDSAPub.pem) son el mismo (Proceso de verificación "vfy").

Para ello emplearemos el siguiente comando:

```
# openssl dgst -sha256 -verify GomezBerzosaDSAPub.pem -signature firma_s_Decrypt.bin concatCD.txt
```

Como podemos comprobar, la verificación ha sido un éxito:

```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl dgst -sha256 -verify GomezBerzosaDSAPub.pem -signature firma_s_Decrypt.bin concatCD.txt
Verified OK
```

Figura 8.12: Verificación de la firma s

Cabe destacar, que las claves de verificación ( $V_A$  y  $V_B$ ) son públicas como es lógico, para comprobar las firmas.

Ahora Alicia procederá a concatenar la clave d con la clave c, es decir, la clave Gomez-BerzosaECPub.pem con la clave JuanRamonECPub.pem:

```
# cat GomezBerzosaECPub.pem JuanRamonECPub.pem >concatDC.txt
```

El resultado de la concatenación es el siguiente:



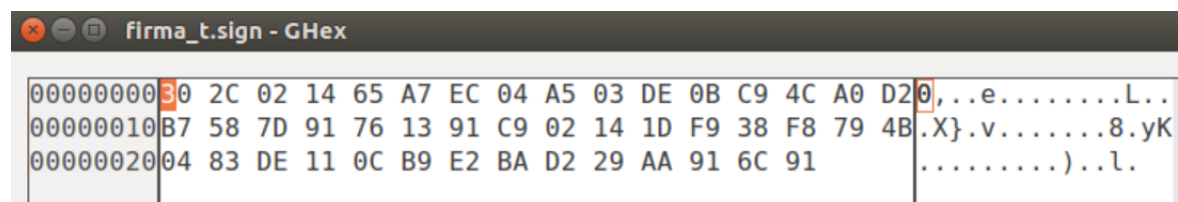
```
concatDC.txt (~/) - gedit
Abrir
-----BEGIN PUBLIC KEY-----
MEkwEwYHKoZiZj0CAQYIKoZiZj0DAQEDMgAEKElzEYFNVhLbKcokI4X6YhXVAp8I
aRPF8Am8L13mn0i6TAqHXreOW/yDQVFzgScq
-----END PUBLIC KEY-----
-----BEGIN PUBLIC KEY-----
MEkwEwYHKoZiZj0CAQYIKoZiZj0DAQEDMgAEhPr51A/0m6UOFQtf7iq5EBcXmzmZ
oNJa1EWVBUOLOKY+qWE0gHy+R5kk0MnvZswu
-----END PUBLIC KEY-----
```

Figura 8.13: Concatenación d||c

A continuación, Alicia procederá a firmar dicho fichero con su clave privada DSA (Juan-RamonDSAPriv.pem):

```
# openssl dgst -sha256 -sign JuanRamonDSAPriv.pem -out firma_t.sign concatDC.txt
```

El contenido de la firma t es el siguiente:



```
firma_t.sign - GHex
00000000 30 2C 02 14 65 A7 EC 04 A5 03 DE 0B C9 4C A0 D2 0, ..e.....L..
00000010 B7 58 7D 91 76 13 91 C9 02 14 1D F9 38 F8 79 4B .X}.v.....8.yK
00000020 04 83 DE 11 0C B9 E2 BA D2 29 AA 91 6C 91 .....).l.
```

Figura 8.14: Firma t

Con esto tendremos la firma t. Ahora se procederá a cifrarla al igual que antes con un cifrado simétrico aes-128 en su modo CFB8 con su clave derivada (key1.bin):

```
# openssl aes-128-cfb8 -pass file:key1.bin -in firma_t.sign -out firma_t_Encrypt.bin
```

El cifrado de la firma t tendrá el siguiente aspecto:

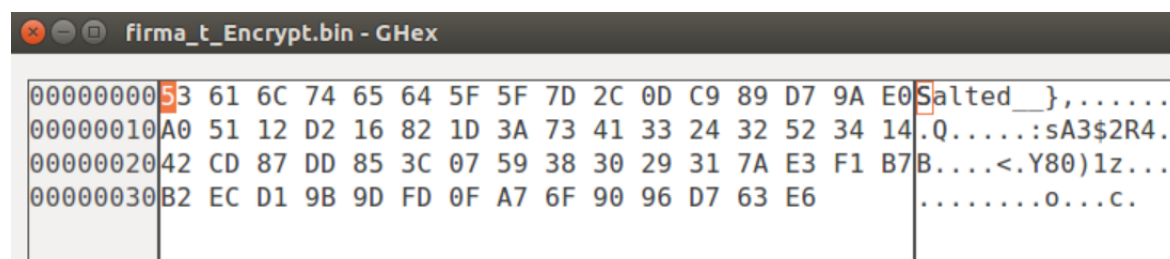


Figura 8.15: Cifrado de la firma t

Una vez cifrada la firma, Alicia la comparte con Bernabé.

## 8.4. Cuarta etapa

En primer lugar Bernabé procederá a descifrar la firma t recibida de Alicia usando su clave compartida (la clave de Bernabé, key2.bin).

```
# openssl aes-128-cfb8 -d -pass file:key2.bin -in firma_t_Encrypt.bin -out firma_t_Decrypt.bin
```

Como podemos comprobar que el descifrado de t se corresponde con la original:

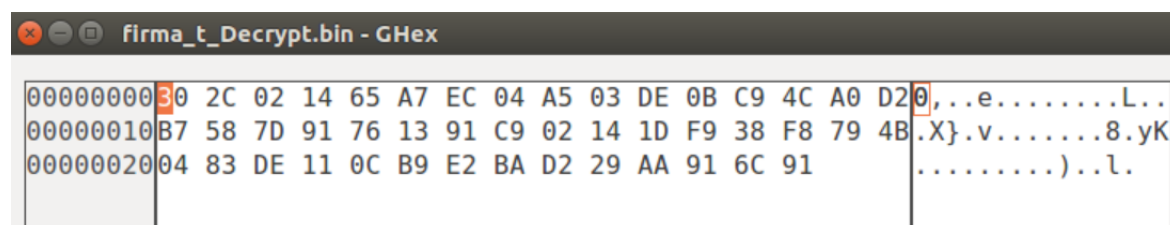


Figura 8.16: Firma t descifrada

Ahora procederemos, al igual que antes con la firma s, a la verificación de la firma de t usando la clave pública de Alicia (JuanRamonDSAPub.pem):

```
# openssl dgst -sha256 -verify JuanRamonDSAPub.pem -signature firma_t_Decrypt.bin concatDC.txt
```

Como podemos comprobar el proceso de verificación ha sido un éxito:

```
jramongomez@jramongomez-Parallels-Virtual-Platform:~$ openssl dgst -sha256 -verify JuanRamonDSAPub.pem -signature firma_t_Decrypt.bin concatDC.txt
Verified OK
```

Figura 8.17: Verificación de t

### 8.5. Quinta etapa

A partir de este momento ambos pueden estar seguros de que la clave secreta  $k$  está compartida sólo por ellos dos, por tanto podemos dar por terminada la simulación del protocolo estación a estación.

## Referencias