ETSIT, URJC

Práctica de Seguridad en Redes de Ordenadores: SRO Signature

Enrique Soriano-Salvador

February 17, 2017

1 Introducción

Implemente un programa en C para GNU/Linux llamado `sign` para crear y verificar firmas digitales usando como algoritmo de cifrado de clave asimétrica RSA con una clave de 4096 bits, y SHA-512 como algoritmo de resumen.
El programa necesita al menos dos argumentos:

```
sign [-v signfile] datafile keyfile
```

El fichero `datafile` es el fichero con los datos que se quiere firmar. El fichero `keyfile` contiene la clave RSA generada por el comando `openssl`, aplanada en base64 (formato PEM). Cuando se ejecuta únicamente con esos dos argumentos, el programa debe calcular la firma digital del fichero de entrada y escribirla por su salida estándar. Por ejemplo, para firmar un fichero:

```
sign myfile.txt privkey.pem > signature.pem
```

Cuando se usa el modificador `-v`, el programa verifica la firma. En este caso, el primer fichero pasado como argumento tiene que ser el fichero con la firma digital. El segundo fichero es el fichero de los datos firmados y el tercer fichero es el que contiene la clave pública correspondiente (nótese que no es un certificado, es la clave pública en formato PEM).
En caso de que la firma no sea correcta, debe imprimir un error por la salida de errores y acabar con un estatus de error. En caso de que la firma sea correcta, no debe imprimir nada y debe acabar con estatus correcto. Si desea imprimir errores de diagnóstico, puede añadir un

modificador extra `-d` para activar la depuración. Bajo ningún concepto se puede imprimir mensajes de diagnóstico si no se provee dicho modificador. Por ejemplo, para verificar el fichero del ejemplo anterior:

```
sign -v signature.pem myfile.txt pubkey.pem
```

Puede generar las claves con el comando `openssl` como sigue:

```
openssl genrsa -out privkey.pem 4096
openssl rsa -in privkey.pem -out  pubkey.pem -outform PEM -pubout
```

2 Formato de la firma

El formato de la firma es similar al estándar RSASSA-PKCS1 V1.5. Los algoritmos para generar la firma tienen que ser RSA (con clave de 4096 bits) y SHA-512.
La firma se tiene que generar de tal forma que autentique tanto el contenido del fichero firmado como el nombre del fichero. Para ello, se generará la firma del siguiente modo, siendo D los datos del fichero y N el nombre del fichero (únicamente se debe tener encuenta el nombre del fichero, no la ruta absoluta):

$E_{k_{priv}}(PADDING||HASH)$

siendo HASH

$HASH = SHA512(D||N)$

El padding para generar la firma con RSA debe seguir el estándar EMSA-PKCS1 V1.5 para una longitud de clave de 4096 y SHA-512:

- La hash del mensaje y se concatena con el ID del tipo de hash usada. El resultado es T.

  $T = ID||HASH$

  El ID de tipo para SHA-512 es:

  ```
  unsigned char EMSASHA512ID[] = {0x30, 0x51, 0x30, 0x0d,
                                  0x06, 0x09, 0x60, 0x86,
                                  0x48, 0x01, 0x65, 0x03,
                                  0x04, 0x02, 0x03, 0x05,
                                  0x00, 0x04, 0x40};
  ```

- Una cadena de bytes con valor 0xFF de longitud en bytes de $(4096/8) - len(T) - 3$. Esa cadena es PS.

- El mensaje a firmar resulta:

  $0x00||0x01||PS||0x00||T$

A la hora de verificar una firma, es necesario comprobar que el padding es correcto.
La firma digital generada debe estar codificada en base64, comenzar con una línea de cabecera, y terminar con una línea final exactamente como en este ejemplo:

```
---BEGIN SRO SIGNATURE---
AV1jWwBZog1zcoqA4uYNRe2/pguYOuywCL2lMigJVvanUSDz3amAp+9Yx3fD8Ku1
puZWLObiMWhKKc3hvq7MxV5+dfcvNVKQVR6ScvKuIUcEvyf5nptNaroq3NYjwztz
g2iJfHMGiSG38XG/XVM6wJ9xSOjJVNykoknE2wJwf2/SJfOdBmAiZ9WD7GMIuWoJ
4z1GhsVV9IjFU6+sZOlQrDkp3inDSB2OjgU5HyYWmGT9a4MgOPNYcNiiNPJiAKnt
llJZhnVLXxpt4f/Arm5EQPJll6LaVgMxtgFvxP5pLmkOXt7pFRVRAhYbiyuEYvK6
YXz0tbZOZQav42NFLovUi5iRH8DhKswfiljDdd1Q/aofagYuSaRyIZoR24Jfu6WO
paSHBSH68DGHNbTN/jklzc6GBQiDJXAHVbPA5UYU/xzxBgEY8Gmh+w2HPudYfrq/
KnePQEZSJgegiETAcCyLCJuF+1X8a1xEpOwWG2wyt8h/53fPt1GSCvgDvxiNVZVN
MMka5kHuCL7/11m1wCIqRqtim+6S/EjZiw7ilX96Q2XPufrK61Jr0GjBvslH9Zec
jTGKIhrLUgVPvjkKHtwW3hvDd12agPxTQDxbmumAE80TPFxShFCmWiRkOP6N5kQW
h2cDNkzVWqso8vWDSDB20NXs/rGERP6JSf1MrNFZ7aQ=
---END SRO SIGNATURE---
```

3 Implementación

Para implementar el programa se debe usar la biblioteca `Crypto` de openssl. Entre otras, se deben usar las siguientes funciones de esa biblioteca:

- `PEM_read_RSAPrivateKey`
- `PEM_read_RSA_PUBKEY`
- `RSA_private_encrypt`
- `RSA_public_decrypt`
- `SHA512_Init, SHA512_Update, SHA512_Final`
- `BIO_read, BIO_write`

Para compilar el programa:

```
gcc -Wall -g sign.c  -lssl  -lcrypto
```

Puede encontrar documentación en las páginas de manual de GNU/Linux y en las siguientes URLs:

- https://www.openssl.org/docs/
- http://wiki.openssl.org/index.php/Main_Page

4 Ejemplo

```
$ cat privkey.pem
-----BEGIN RSA PRIVATE KEY-----
MIIJKQIBAAKCAgEAyR0g1tMVaqIcA374yEsMaaSqfcYNjP9cuCtdBh//2iR5IeMw
UACwqWtMxB74bPA3ihaLj5R9HWM+irIERtlYU/nB+ZJbOnfWJtrVjJmpzczZYHa+
xXWYnsI+3S8Rqm8U51kEvDOlQkGMq3psPxpuYlOuZ/ic8PlnVJYrznxiv9KjVkXC
WaItwhh0yFj0ijBfg3YMwnv3sJGXwCvGoL/G8kUOqpDjea5sgTzvPnrg8g1D7MAR
rU89058OIE/f9RtzuZxKQl9SDQtrkYU9lOI8zPSJBWZydxRB5maNpg4NPU9LZx0g
IQP2ePJhd5lF2AyHnKVbNcI3T84fZJLgW7JPTiWeSIJhfZZ9QIrOgAcnxebxbmo/
ukQ/NVNi9uBjL8ugsRQjjBgTmYohNqrG85I+2tuRiX53uhnYeonWjvlx+ov3FoW6
8gaNlAmaNVLwE93fDJopFXtA+O4okdHHKqmQVVs2LYTf6MTQUGSgZbnF6ALVXa1W
7fvVhoswlTPty+Q4puBXqR8vPOZUMcEnmSGs8OYE41LXwBkeHMhgU5hOTZqp3eKX
FI+9WCinH3yWkDhW7a2nY7Ehsvyyp3+o7XhVl87JQzWbaiziypv+UL2VfxqzmCDP
ojVVf9uHijrWUGSxGYpb7bsS/EDYRGLuiWAisbhQvOXeNRtC5PkBt8A8YuMCAwEA
AQKCAgACeGYf3WXk8mrPrC6YHzvezFP/yX//HF/iLz4sRhZZcps+TFEamneRDS1b
N1or1GOKQa6jK9rBkqeBAqDE0gSgu2+jhiWyuSgbQBLhcD3CtmJxKeQ7/q7KPG6T
PvHDmyuxj8lcGpArmSyGKrHLsKJseMSqqEYdO6MGSfXtyl9YJdk1xROXEEPpn21H
zLfsPp3duoR7mxQ2ygMILEF7Vf+2mByPAcqZgwf4Kmxx4waCUqFj9hQBgfircce1
o+WHWDf6rq3G1O7oFBzVI8LW4lSG2/YW5+Q+DDSnBNl2kbOd7ixp7tSnMypC4A9M
bciK19SOmwVcyq7tPwXpsVgqKC2LLZnt1QhhHzPDzcRzbxiZEX2F4BBJ3U4PVYIS
WJFSSfi4BugYKapCEBQHJlGnUjPU7B9Cy6hJ7dPPIp2CkXmNSGnaFR4QLODTZ1gN
ihNkoopwMXOhkfleOeEBFg2NSW8CgTzfynzNE1sROOlk7i4Bn1+tTl6HV2xoK1Hx
+u+nFbN1WgOeEbgrgXAtZi0kkJ/FVizpgsL18QRfthMRC1TKLSUW4EvzWw4i/Q0t
0Cz5loxE1N+TPmF52WVgDFULk/nVGXc3la9mkcUxi5MfyxO2RmMQbJyMVZ6xL8ps
Blxlzr0HJmBs6zIXeTSnu1F8DkjqJF1JWkkmqzK6GM0nYxc+4QKCAQEA9PAKvPBX
oJBpPsexUtb2y87oQQmuIJrnshL1pQLsIVt8zW2MGDW3o4YV5/AedoZhTKn6TTnX
LecBwBkvvhOTjv6RQpIlmUKkmcey18MihoXgBE1DMSM8bQuNeHfy132aMr5tdPZC
lyFlAJWBG7oeMqE+0rTwalYVkEbF+RW5naN2deTCIm4QUDiOcQkA8w02UmYaOWCO
zgGUJlhmlwDk0cE1QZlXLjrC3Fm8ljj6Aurn0wZMfQxwM8M8qo0DQl7bLC0TTJO9
GJLBqlRkC1/eHRWgQF9lWfiXjqT01m9O/xkQyHWWpaMErBGOCw1ijgS2uvPvzN58
6evD2beUauoWUwKCAQEA0jJliPPNXkTttOPDFPs2HSxEJVfHk+ybXM+S/w83fMrF
F0najO4wBmDb9cii6HjVKRUPsgKMtA5sQCc2No90CaX/Y4zhHcoM9T4I9aG+zFqH
49PdOjwzrjKnJy23+/TbhEC5lj5hHrD1dGFSyP9y4AJLXG8zU+Inmye59fO7WIln
6u+JYnFID18buinOoujkuGQNk26zXheILB53iDFDR7YWidTTo9FvtncWyDjFvofj
JTubgrilaNL5E7pQSTtnUCTqnVkBncT2kq7lUV9+nSbGDUl1F3V0tVFPf5330C4r
7Zo9w2MIqMUkYk4cZlH/q9ClzNVwNfwVgXnHqvvPMQKCAQEAh5H9P4p/1d1Yg2kg
GsvkmfYR0z26ZU2YBJY95HFzpRrwPvvtWNESra3fnhrnoY7LeBV09x2Wnk+IRn0q
UbigKbt5RzGBIg0i8gL4WDgnefHLhlYFZMMuBOUqDo3FmcRpfsCr8NsFDIVtVB9r
8J7ZbAiXryR7FUBEezDRDwcZT8lUHfjaAxiMavqCzMnA/sZHVOAyj6OEJz30dCzl
y5qxC/A2u/JVsL7RcAkzOqqaptbCLakE2QnzaJMdlwCp1yiNgywHzJDRTzKbgt1m
6mzLkamQo1Cp1lyj9k4TPkUpokSLZ4i+MzvBsEOfLTrhW938DgpKpkhXN1pJcs4L
lgmvBQKCAQEAguV3bW3F+mqaTQd5ONunu0sRtN+RHYE+zvFE7SkguMndKr+HJjQ+
G/q9f7XOHU8CD29aMtR7orVInDrO+/Mho9CH1gqpNc/Zee+DzNNI6iGGdk49ekJ7
```

```
PIO2rCNAa9rzyMw1xmZaPK6ebDcfQqQxeWZ0X7+wCxDO8UQv/gYmKOCIojlBKNi6
szfbIHdggvrdVCaafbF2aaXl2vOuJFXpPAMczgCHO4D1PH+05ELWgexFe64/DYzH
FRwsmChyTuh7UeFraUlARGuf0YCwtZfuVRcMRsHz9QPkBfX4t9Q7upzGJlTjGYXW
oqMCIWXbMazmtqxcU86m7jdpfRPFT6k4QQKCAQBYOwocijnmz3KpIFBTwQVxQbPv
gfMJJU3aDaO6uKzVjIdc2u4PnIJbP1661loN7chEZ0IjUaryoQ69GV2m+L6G2GqR
JmGOloPABuam8c04QFhb2HnoBX+uNo3rLnYMQEMRTYoX/qmCMFzSDIXw8GEzCgsW
p8XHK4HtwNcF91tbuJXZO8FluYmHyZ70AdEkrGEXtCaXgwZ/qXYFeP9dRCg6u1dc
DWka7B2OEZsNuTuvqb5O807XsRQU78WIthocahPYgQVPtoXSc4+y2ZyPTma5Jmn1
CZAq2tqzxLYMNX7xwo4Mo6uM9D6pkBqJk/o0jQfD6AlHAIP3XlXzHTlz7p1g
-----END RSA PRIVATE KEY-----
$ cat pubkey.pem
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAyR0g1tMVaqIcA374yEsM
aaSqfcYNjP9cuCtdBh//2iR5IeMwUACwqWtMxB74bPA3ihaLj5R9HWM+irIERtlY
U/nB+ZJbOnfWJtrVjJmpzczZYHa+xXWYnsI+3S8Rqm8U51kEvD0lQkGMq3psPxpu
YlOuZ/ic8PlnVJYrznxiv9KjVkXCWaItwhh0yFj0ijBfg3YMwnv3sJGXwCvGoL/G
8kUOqpDjea5sgTzvPnrg8g1D7MARrU89058OIE/f9RtzuZxKQl9SDQtrkYU9lOI8
zPSJBWZydxRB5maNpg4NPU9LZx0gIQP2ePJhd5lF2AyHnKVbNcI3T84fZJLgW7JP
TiWeSIJhfZZ9QIrOgAcnxebxbmo/ukQ/NVNi9uBjL8ugsRQjjBgTmYohNqrG85I+
2tuRiX53uhnYeonWjvlx+ov3FoW68gaNlAmaNVLwE93fDJopFXtA+O4okdHHKqmQ
VVs2LYTf6MTQUGSgZbnF6ALVXa1W7fvVhoswlTPty+Q4puBXqR8vPOZUMcEnmSGs
80YE41LXwBkeHMhgU5hOTZqp3eKXFI+9WCinH3yWkDhW7a2nY7Ehsvyyp3+o7XhV
l87JQzWbaiziypv+UL2VfxqzmCDPojVVf9uHijrWUGSxGYpb7bsS/EDYRGLuiWAi
sbhQvOXeNRtC5PkBt8A8YuMCAwEAAQ==
-----END PUBLIC KEY-----
$ echo hola > myfile.txt
$ sign myfile.txt privkey.pem > signature.pem
$ cat signature.pem
---BEGIN SRO SIGNATURE---
AV1jWwBZog1zcoqA4uYNRe2/pguYOuywCL2lMigJVvanUSDz3amAp+9Yx3fD8Ku1
puZWLObiMWhKKc3hvq7MxV5+dfcvNVKQVR6ScvKuIUcEvyf5nptNaroq3NYjwztz
g2iJfHMGiSG38XG/XVM6wJ9xSOjJVNykoknE2wJwf2/SJfOdBmAiZ9WD7GMIuWoJ
4z1GhsVV9IjFU6+sZOlQrDkp3inDSB2OjgU5HyYWmGT9a4MgOPNYcNiiNPJiAKnt
llJZhnVLXxpt4f/Arm5EQPJll6LaVgMxtgFvxP5pLmk0Xt7pFRVRAhYbiyuEYvK6
YXz0tbZOZQav42NFLovUi5iRH8DhKswfiljDdd1Q/aofagYuSaRyIZoR24Jfu6WO
paSHBSH68DGHNbTN/jklzc6GBQiDJXAHVbPA5UYU/xzxBgEY8Gmh+w2HPudYfrq/
KnePQEZSJgegiETAcCyLCJuF+1X8a1xEpOwWG2wyt8h/53fPt1GSCvgDvxiNVZVN
MMka5kHuCL7/11m1wCIqRqtim+6S/EjZiw7ilX96Q2XPufrK61Jr0GjBvslH9Zec
jTGKIhrLUgVPvjkKHtwW3hvDd12agPxTQDxbmumAE80TPFxShFCmWiRkOP6N5kQW
h2cDNkzVWqso8vWDSDB20NXs/rGERP6JSf1MrNFZ7aQ=
---END SRO SIGNATURE---
$ sign -v signature.pem myfile.txt pubkey.pem
$
```