Mossio Giacomo

Fundamentos de Seguridad – Análisis de la seguridad en los Sistemas de Información – CURSO 2019-2020

Práctica 1: Taller de OpenSSL. Cifrado (simétrico y asimétrico), resúmenes, certificados X.509, correo S/MIME, correo PGP y servidor seguro SHTTP.

PARTE 1 - Utilización de OpenSSL (cifrado simétrico, resúmenes, claves asimétricas, firma y cifrado asimétrico)

1.1 Cifrado Simétrico de documentos

- Estudiar en la documentación de openssl cómo se utiliza el comando enc para cifrar y descifrar. Explicar sus opciones más importantes, con especial atención a los métodos de cifrado (ecb, cbc, etc.).

  enc command:

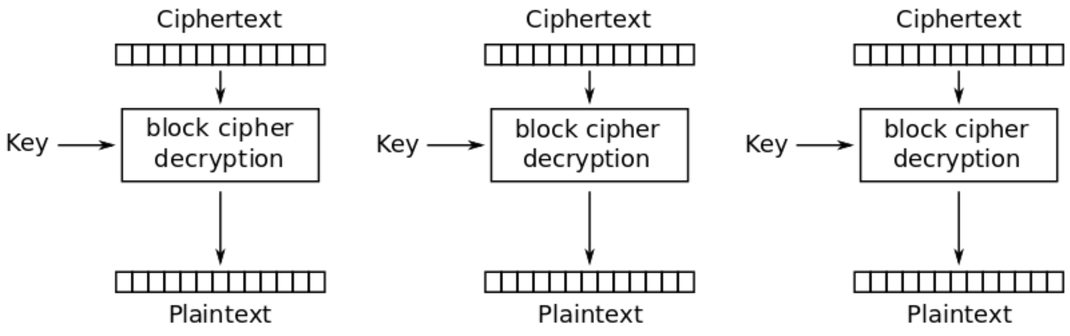
  `openssl enc -cipher [-help] [-list] [-ciphers] [-in filename] [-out filename] [-pass arg] [-e] [-d] [-a] [-base64] [-A] [-k password] [-kfile filename] [-K key] [-iv IV] [-S salt] [-salt] [-nosalt] [-z] [-md digest] [-iter count] [-pbkdf2] [-p] [-P] [-bufsize number] [-nopad] [-debug] [-none] [-rand file...] [-writerand file] [-engine id]`

  This is the enc command with all the possible parameters.
  The most important are:
  *`[-list]` and `[-ciphers]` are the same command and are used for display all the possible ciphers that can be used for encryption. If we try to execute the command we'll see a list of the ciphers combinations in format -[ciphers]-[bits number]-[mode]; there are a few common modes for encryption and their purpose is always to mask patterns which exist in encrypted data, the most common are: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Propagating CBC (PCBC), Cipher Feedback (CFB), Output Feedback (OFB) and Counter (CTR)

***[-in filename] [-out filename]** which allow to select the file in input and output (standard input/output by default)
***[-e] [-d]** commands will set the program in encryption and decryption mode
***[-pass arg]** is used for select the password from which will be derived the key and is generic, the format is operation:parameter and can contain:
1.pass:password receives password from command line (has the same drawbacks of the -k option);
2. env:var uses an environment variable as a password for key derivation;
file:path reads the password from a file (exactly like the -kfile option);
3.fd:number reads the password from a file descriptor (used in network and pipe lines);
4.stdin prompts the password from standard input.
***[-nosalt]** command is used for disable the salt insertion for the key generation
***[-nopad]** command is used for disable the padding insertion for the key generation

The common cipher modes of operation are the following:

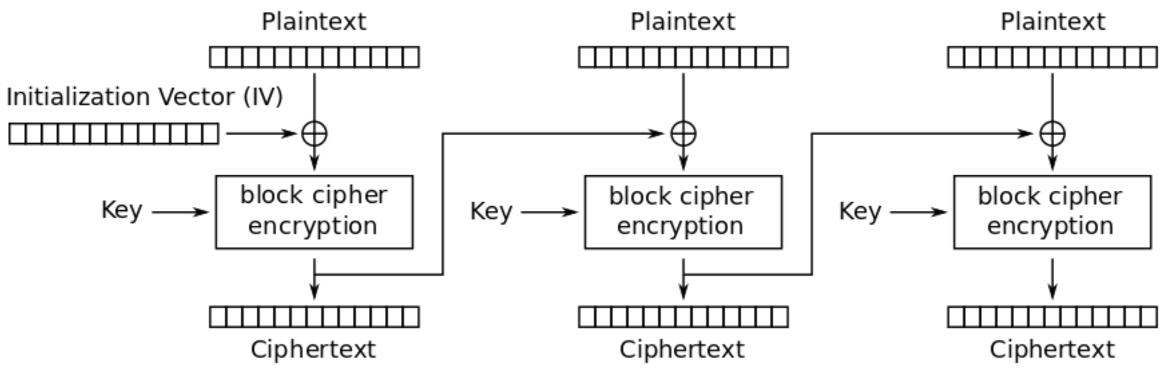
1. **Electronic codebook (ECB)**



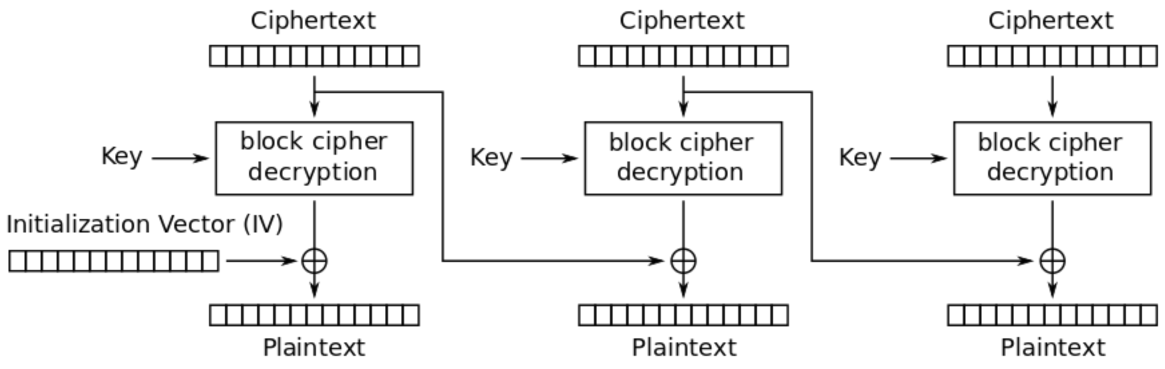
Electronic Codebook (ECB) mode decryption

This is the most simple mode of encryption, where the file is divided in block and the encrypted by blocks. It isn’t safe because the algorithm doesn’t hide the patterns and encrypt identical plaintext blocks into identical ciphertext blocks; for this reason it is not recommended for any encryption even if the encryption/decryption with it is parallelizable and each block is independent so we can read it randomly.

2. Cipher block chaining (CBC)



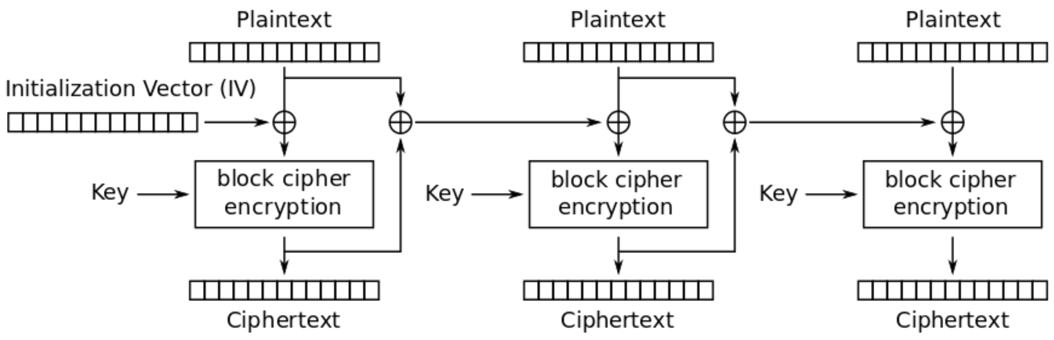
Cipher Block Chaining (CBC) mode encryption



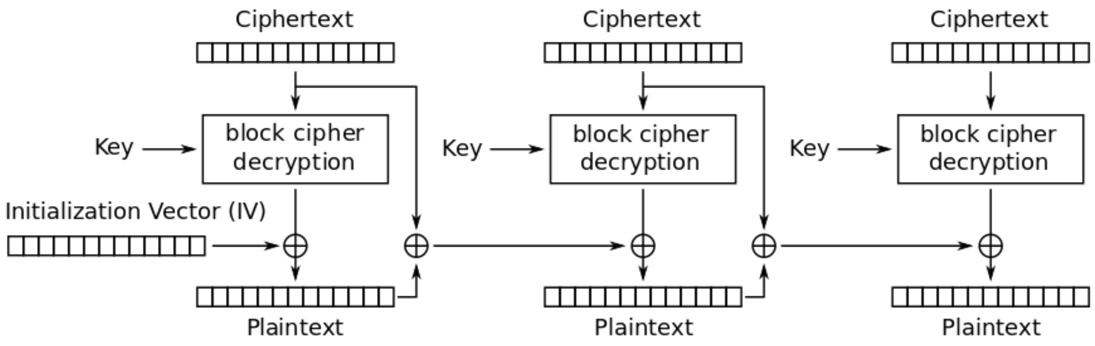
Cipher Block Chaining (CBC) mode decryption

In this mode the each block of plaintext is XORed with the previous ciphertext block before being encrypted, with this technique every ciphertext block depends on all the previous blocks and so it's not possible the parallel encryption. It also need a initialization from which start.
This is a safe and widely used mode because even a little change, for example in the initializing vector will affect all the successive blocks, and also because it is decryptable parallelized because each block only needs the previous ciphertext block for the decryption.

3. Propagating cipher block chaining (PCBC)



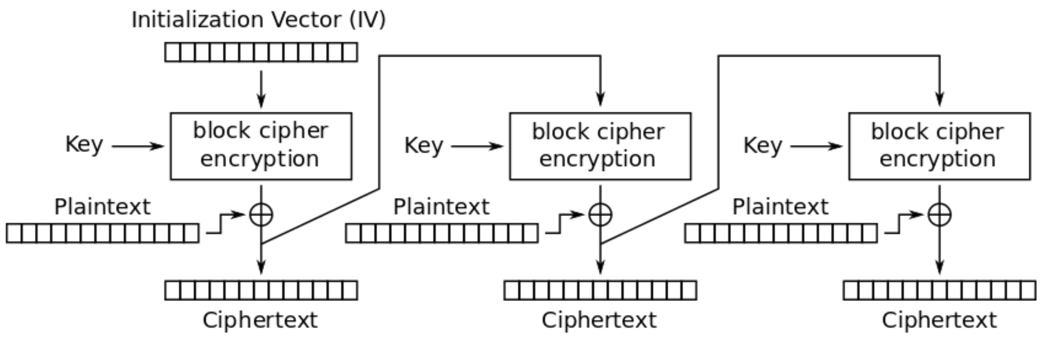
Propagating Cipher Block Chaining (PCBC) mode encryption



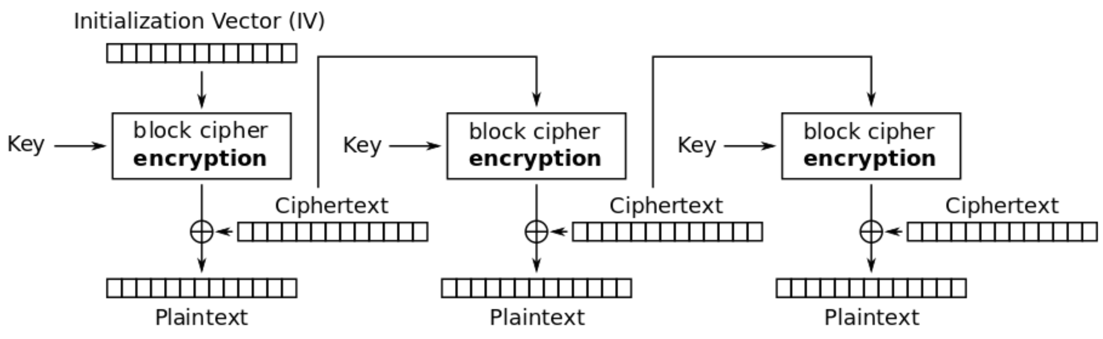
Propagating Cipher Block Chaining (PCBC) mode decryption

For creating a butterfly effect (small changes propagate indefinitely) even in the decryption, in this mode each block of plaintext is XORed with both the previous plaintext block and the previous ciphertext block, making this it will not be possible to parallel encrypt or decrypt the file. In this mode too it's needed an initialization vector.

4. Cipher feedback (CFB)



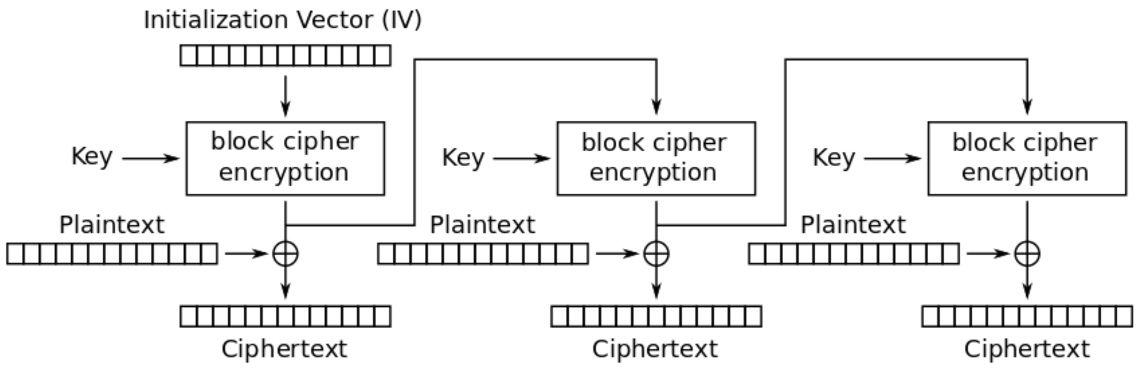
Cipher Feedback (CFB) mode encryption



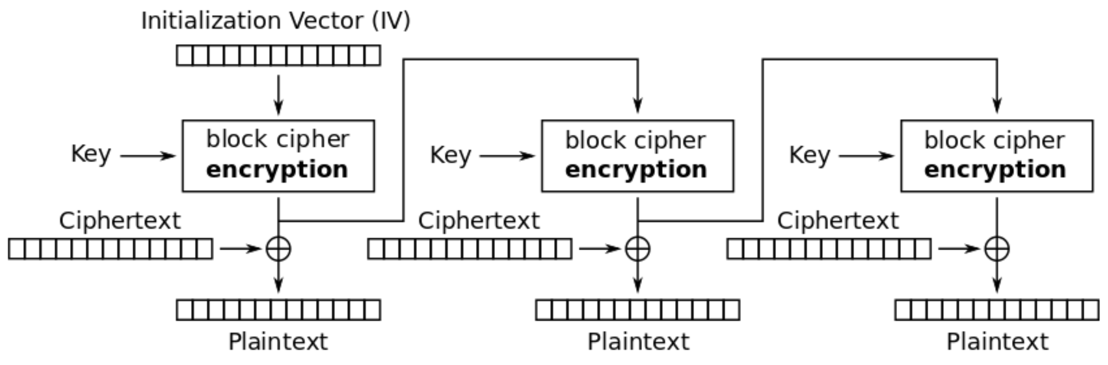
Cipher Feedback (CFB) mode decryption

This mode is very similar to the CBC mode, the different is that CFB makes a block cipher into a self-synchronizing stream cipher.
By definition of self-synchronising cipher, if part of the ciphertext is lost (e.g. due to transmission errors), then the receiver will lose only some part of the original message thanks to the use of a shift register.
Encryption can not be parallelized but a little change will cause a butterfly effect. The decryption, like in CBC is instead parallelizable but a change in the ciphertext affects two plaintext blocks.

5. **Output feedback (OFB)**



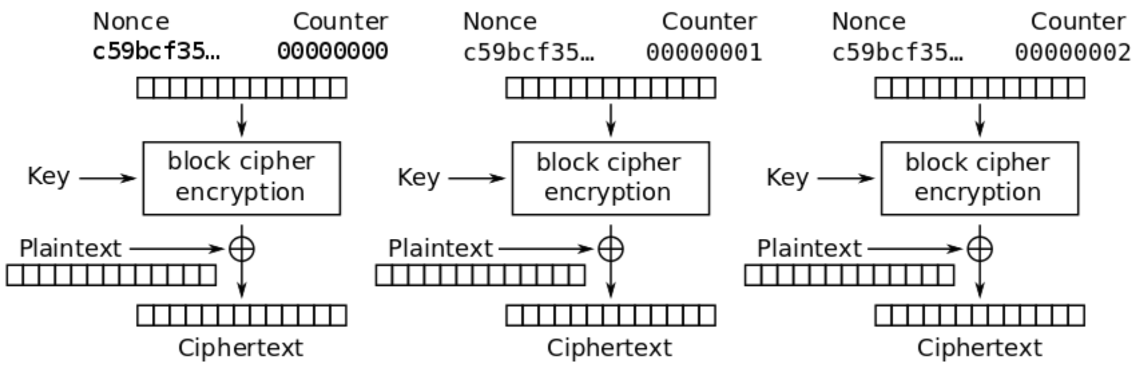
Output Feedback (OFB) mode encryption



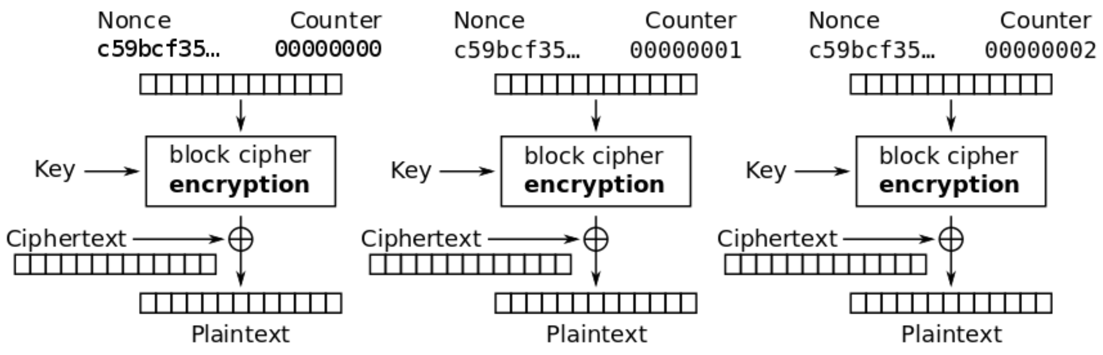
Output Feedback (OFB) mode decryption

As we can see, encryption and decryption are exactly the same, that means that now even the decryption is no longer parallelizable but it has the butterfly effect.
It is possible to obtain an OFB mode keystream by using CBC mode with a constant string of zeros as input.

6. Counter (CTR)



Counter (CTR) mode encryption



Counter (CTR) mode decryption

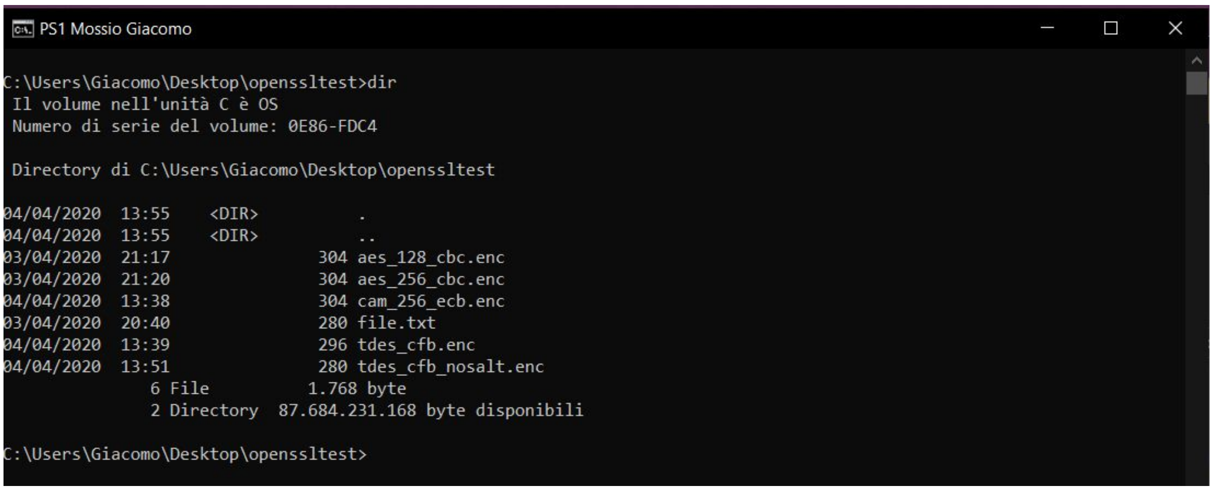
Its name is due to the fact that it generates the next keystream block by encrypting successive values of a "counter".
CTR mode has similar characteristics to OFB, but also allows a random access property during decryption and it does not suffer from the short-cycle problem that can affect OFB.

- Crear un archivo de texto legible (pequeño tamaño – entre 260 y 300 caracteres)

  The file created is of 280 characters and is the following:
  Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat mas

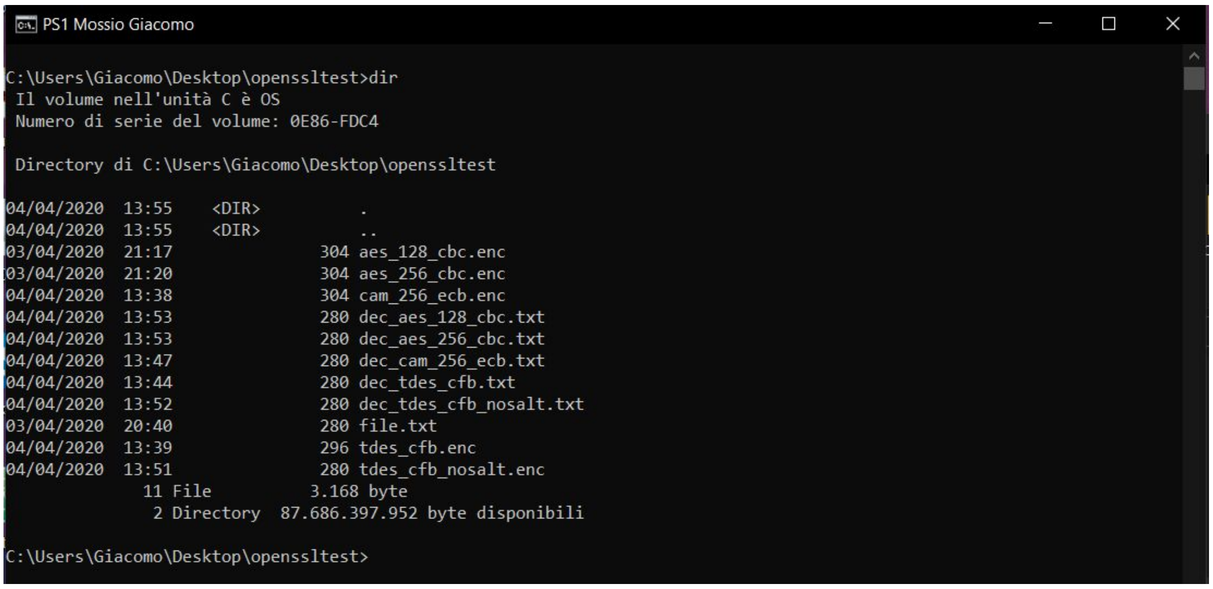
- Cifrarlo con CINCO algoritmos simétricos (AES y TDES obligatorios y un cifrador de flujo como mínimo).

  Executing the command:
  `enc -aes-128-cbc -in file.txt -out aes_128_cbc.enc`
  and changing the parameters (with the addition of the parameter **`-nosalt`** too) we’ll obtain the following encrypted files:



```
C:\Users\Giacomo\Desktop\openssltest>dir
 Il volume nell'unità C è OS
 Numero di serie del volume: 0E86-FDC4

 Directory di C:\Users\Giacomo\Desktop\openssltest

04/04/2020  13:55    <DIR>          .
04/04/2020  13:55    <DIR>          ..
03/04/2020  21:17               304 aes_128_cbc.enc
03/04/2020  21:20               304 aes_256_cbc.enc
04/04/2020  13:38               304 cam_256_ecb.enc
03/04/2020  20:40               280 file.txt
04/04/2020  13:39               296 tdes_cfb.enc
04/04/2020  13:51               280 tdes_cfb_nosalt.enc
               6 File          1.768 byte
               2 Directory  87.684.231.168 byte disponibili

C:\Users\Giacomo\Desktop\openssltest>
```

- Descifrarlos y comprobar el resultado.



PS1 Mossio Giacomo

```
C:\Users\Giacomo\Desktop\openssltest>dir
 Il volume nell'unità C è OS
 Numero di serie del volume: 0E86-FDC4

 Directory di C:\Users\Giacomo\Desktop\openssltest

04/04/2020  13:55    <DIR>          .
04/04/2020  13:55    <DIR>          ..
03/04/2020  21:17               304 aes_128_cbc.enc
03/04/2020  21:20               304 aes_256_cbc.enc
04/04/2020  13:38               304 cam_256_ecb.enc
04/04/2020  13:53               280 dec_aes_128_cbc.txt
04/04/2020  13:53               280 dec_aes_256_cbc.txt
04/04/2020  13:47               280 dec_cam_256_ecb.txt
04/04/2020  13:44               280 dec_tdes_cfb.txt
04/04/2020  13:52               280 dec_tdes_cfb_nosalt.txt
03/04/2020  20:40               280 file.txt
04/04/2020  13:39               296 tdes_cfb.enc
04/04/2020  13:51               280 tdes_cfb_nosalt.enc
              11 File          3.168 byte
               2 Directory  87.686.397.952 byte disponibili

C:\Users\Giacomo\Desktop\openssltest>
```

all the decrypted files are an exact copy of the original file.txt

- Explicar el tamaño de los diferentes ficheros cifrados en virtud del tamaño de bloque del cifrador (o no, si es un cifrador de flujo), y sabiendo que el empleo de sal añade 16 bits de más al inicio del fichero cifrado –Salted__XXXXXXXX-.

As we can see the encrypted files have different dimensions based on the different mode used for the encryption, the files encrypted by using CBC and ECB algorithms are 24 bytes bigger than the original. Those bytes are due to the padding and the salt:
-the padding because the file need to be "filled up" with some bytes (280 isn't a multiple of 16 (128 bits)) and so 8 bytes are gonna be added (288 is a multiple of 16)
-the salt is instead always gonna be of the dimension of 16 bytes
If we sum 280+8(padding)+16(salt) we obtain 304.
The option **`-nosalt`** will force the program not to put the salt (which is used for creating a stronger key) and the resulting is a file 16 bytes lighter than the one with the salt.
The stream algorithm TDES with cfb mode and no salt will generate a file of the same dimensions of the original file.txt because the message does not need to be padded to a multiple of the cipher block size.

- Explicar la gestión de contraseñas detallada en el estándar PKCS #5 (PBKDF1 y PBKDF2) y su aplicación a las claves de cifrado simétrico, vectores de inicialización y sal (derivación de claves e “iv” a partir de contraseñas). Documentar las diferentes alternativas, empleando diferentes algoritmos de cifrado. Demostrar que un fichero puede ser cifrado con contraseña y descifrado con su conjunto equivalente de clave (key), vector de inicialización (iv) y sal (salt).

Passwords are something fundamental in the cryptography and so is required a particular attention at it, the standard PKCS#5 handles the processing of a password. Because of the user security depends a lot on it’s password, is important to process it, for example combining a password with a salt to produce a key; the salt is important because humans can generate a relatively small set of password and that vulnerability can be exploited by a so called “dictionary attack” which consists in exhaustively trying all the set of passwords that we were talking before.
Another technique is to iterate the password with an algorithm an amount of times, like 1000 so that the cost of exhaustive search is gonna be very expensive.
PBKDF1 and PBKDF2(which is an evolution of the previous) have the following formula:
DerivedKey = PBKDF2(PRF, Password, Salt, c, dkLen)
where:
-PRF is a pseudorandom function of two parameters with output length hLen
-Password is the original password from which a derived key is generated
-Salt is a sequence of bits, which allows to produce a large set of keys starting from the original password
-c is the number of iterations desired
-dkLen is the desired bit-length of the derived key

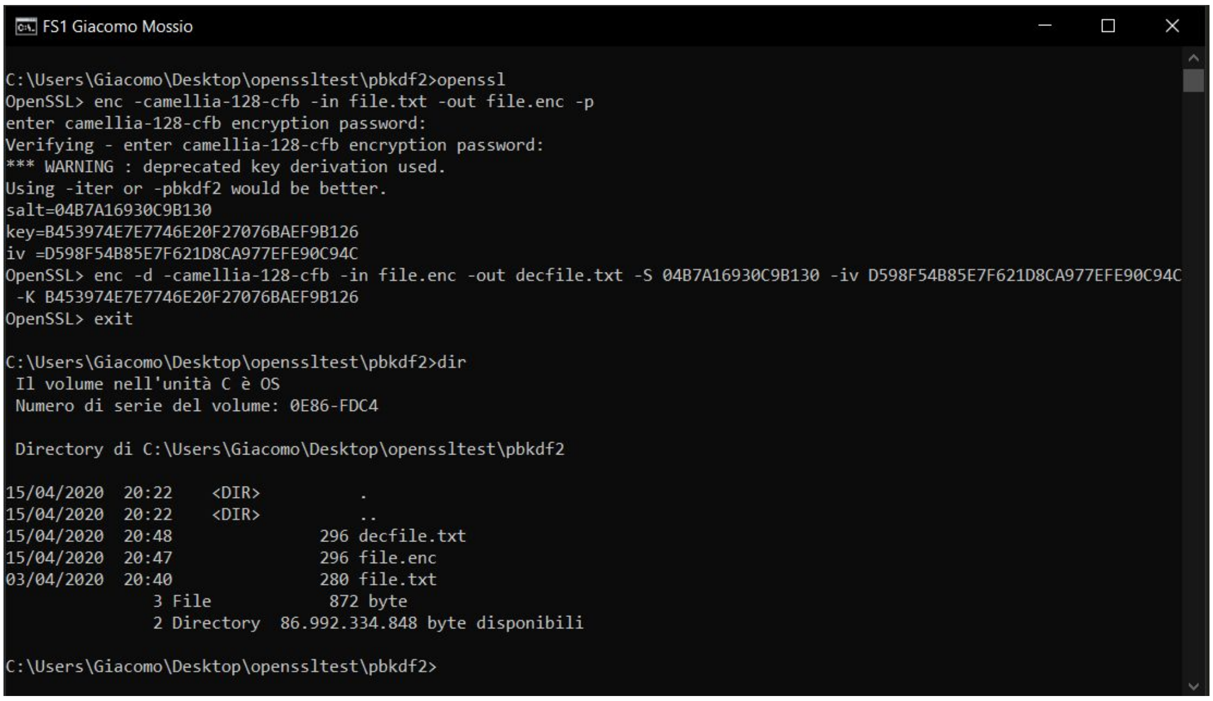
DK = T1 + T2 + ... + Tdklen/hlen
where each Ti = F(Password, Salt, c, i)
The function F is the xor (^) of c iterations of chained PRFs

For example, WPA2 uses:
DK = PBKDF2(HMAC−SHA1, passphrase, ssid, 4096, 256)

For demonstrating that we can decrypt a file by knowing the key, the salt and the initializing vector we can execute the following command:

`enc -camellia-128-cfb -in file.txt -out file.enc -p`
the printed results are the following:
salt=04B7A16930C9B130
key=B453974E7E7746E20F27076BAEF9B126
iv =D598F54B85E7F621D8CA977EFE90C94C
and then executing:
`enc -d -camellia-128-cfb -in file.enc -out decfile.txt -S 04B7A16930C9B130 -iv D598F54B85E7F621D8CA977EFE90C94C -K B453974E7E7746E20F27076BAEF9B126`
we'll obtain a file containing (the file.txt is the one that we already saw):
Sžº8¿é¸¢T
ëgi.»ÁÛŠ(û^"åÛEýÁ,'4r sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat mas
(actually with the first 16 bytes not correct due to salt problems)

- Documentar el trabajo realizado, con ejemplos de los resultados obtenidos (valores binarios en hexadecimal, Base64 o en formato PEM) y profusión de volcados de pantalla.

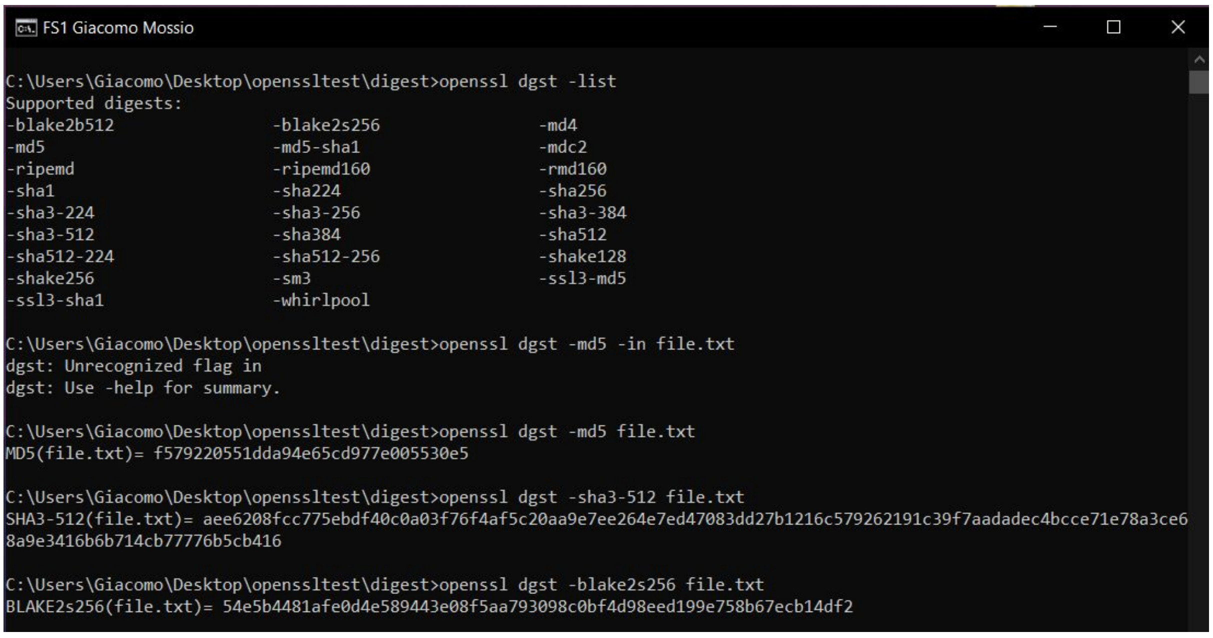

FS1 Giacomo Mossio

```
C:\Users\Giacomo\Desktop\openssltest\pbkdf2>openssl
OpenSSL> enc -camellia-128-cfb -in file.txt -out file.enc -p
enter camellia-128-cfb encryption password:
Verifying - enter camellia-128-cfb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=04B7A16930C9B130
key=B453974E7E7746E20F27076BAEF9B126
iv =D598F54B85E7F621D8CA977EFE90C94C
OpenSSL> enc -d -camellia-128-cfb -in file.enc -out decfile.txt -S 04B7A16930C9B130 -iv D598F54B85E7F621D8CA977EFE90C94C
 -K B453974E7E7746E20F27076BAEF9B126
OpenSSL> exit

C:\Users\Giacomo\Desktop\openssltest\pbkdf2>dir
 Il volume nell'unità C è OS
 Numero di serie del volume: 0E86-FDC4

 Directory di C:\Users\Giacomo\Desktop\openssltest\pbkdf2

15/04/2020  20:22    <DIR>          .
15/04/2020  20:22    <DIR>          ..
15/04/2020  20:48               296 decfile.txt
15/04/2020  20:47               296 file.enc
03/04/2020  20:40               280 file.txt
               3 File            872 byte
               2 Directory  86.992.334.848 byte disponibili

C:\Users\Giacomo\Desktop\openssltest\pbkdf2>
```

1.2 Generación y comprobación de Resúmenes. Generación de claves asimétricas (pública-privada) y firmado de resúmenes.

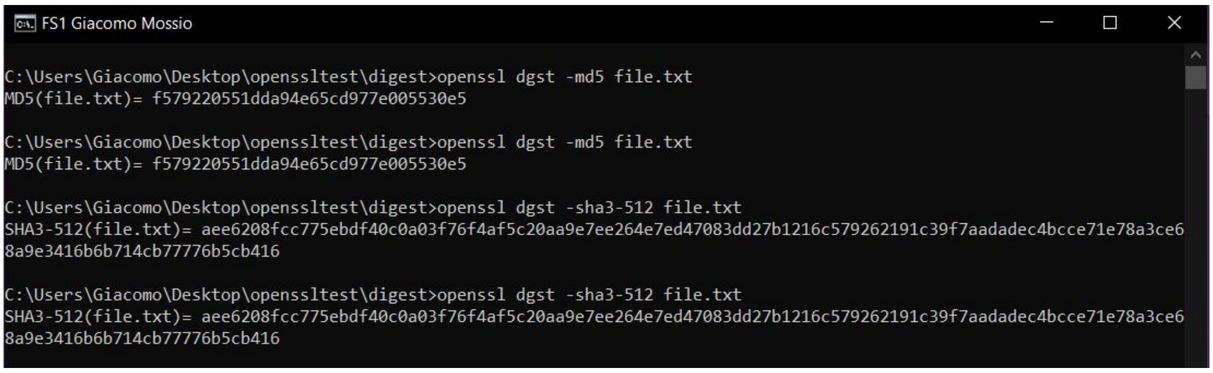
- Utilizando la bibliografía acerca de OpenSSL de la página de la asignatura, utilizar diferentes algoritmos de resumen (TRES de los más modernos) sobre un archivo de texto y comprobar dichos resúmenes ante mínimas modificaciones del fichero.

  For creating a digest we use the command:
  `openssl dgst [-digest] [-help] [-c] [-d] [-list] [-hex] [-binary] [-r] [-out filename] [-sign filename] [-keyform arg] [-passin arg] [-verify filename] [-prverify filename] [-signature filename] [-sigopt nm:v] [-hmac key] [-fips-fingerprint] [-rand file...] [-engine id] [-engine_impl] [file...]`



```
C:\Users\Giacomo\Desktop\openssltest\digest>openssl dgst -list
Supported digests:
-blake2b512                -blake2s256                -md4
-md5                       -md5-sha1                  -mdc2
-ripemd                    -ripemd160                 -rmd160
-sha1                      -sha224                    -sha256
-sha3-224                  -sha3-256                  -sha3-384
-sha3-512                  -sha384                    -sha512
-sha512-224                -sha512-256                -shake128
-shake256                  -sm3                       -ssl3-md5
-ssl3-sha1                 -whirlpool

C:\Users\Giacomo\Desktop\openssltest\digest>openssl dgst -md5 -in file.txt
dgst: Unrecognized flag in
dgst: Use -help for summary.

C:\Users\Giacomo\Desktop\openssltest\digest>openssl dgst -md5 file.txt
MD5(file.txt)= f579220551dda94e65cd977e005530e5

C:\Users\Giacomo\Desktop\openssltest\digest>openssl dgst -sha3-512 file.txt
SHA3-512(file.txt)= aee6208fcc775ebdf40c0a03f76f4af5c20aa9e7ee264e7ed47083dd27b1216c579262191c39f7aadadec4bcce71e78a3ce6
8a9e3416b6b714cb77776b5cb416

C:\Users\Giacomo\Desktop\openssltest\digest>openssl dgst -blake2s256 file.txt
BLAKE2s256(file.txt)= 54e5b4481afe0d4e589443e08f5aa793098c0bf4d98eed199e758b67ecb14df2
```

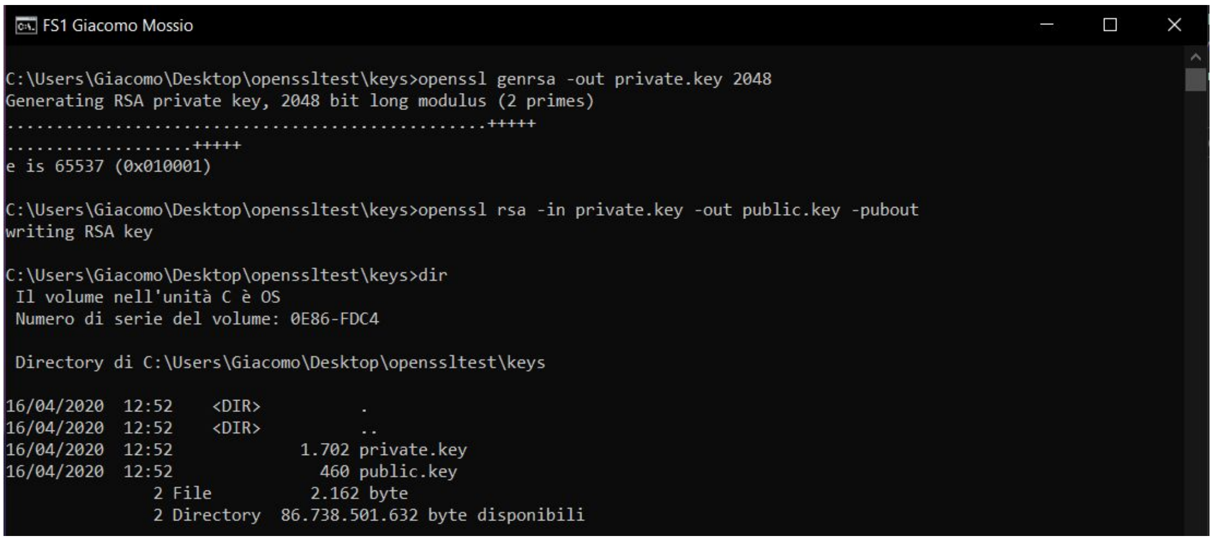
If we don't modify anything, the digests will remain the same as shown below


FS1 Giacomo Mossio

```
C:\Users\Giacomo\Desktop\openssltest\digest>openssl dgst -md5 file.txt
MD5(file.txt)= f579220551dda94e65cd977e005530e5

C:\Users\Giacomo\Desktop\openssltest\digest>openssl dgst -md5 file.txt
MD5(file.txt)= f579220551dda94e65cd977e005530e5

C:\Users\Giacomo\Desktop\openssltest\digest>openssl dgst -sha3-512 file.txt
SHA3-512(file.txt)= aee6208fcc775ebdf40c0a03f76f4af5c20aa9e7ee264e7ed47083dd27b1216c579262191c39f7aadadec4bcce71e78a3ce6
8a9e3416b6b714cb77776b5cb416

C:\Users\Giacomo\Desktop\openssltest\digest>openssl dgst -sha3-512 file.txt
SHA3-512(file.txt)= aee6208fcc775ebdf40c0a03f76f4af5c20aa9e7ee264e7ed47083dd27b1216c579262191c39f7aadadec4bcce71e78a3ce6
8a9e3416b6b714cb77776b5cb416
```

- Generar un par de claves asimétricas RSA de 2048 bits, de acuerdo con las indicaciones del apéndice A del manual básico (para RSA).

  Using the commands **`openssl genrsa -out private.key 2048`** and **`openssl rsa -in private.key -out public.key -pubout`** we obtain our keys.


FS1 Giacomo Mossio

```
C:\Users\Giacomo\Desktop\openssltest\keys>openssl genrsa -out private.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.................................................+++++
...................+++++
e is 65537 (0x010001)

C:\Users\Giacomo\Desktop\openssltest\keys>openssl rsa -in private.key -out public.key -pubout
writing RSA key

C:\Users\Giacomo\Desktop\openssltest\keys>dir
 Il volume nell'unità C è OS
 Numero di serie del volume: 0E86-FDC4

 Directory di C:\Users\Giacomo\Desktop\openssltest\keys

16/04/2020  12:52    <DIR>          .
16/04/2020  12:52    <DIR>          ..
16/04/2020  12:52             1.702 private.key
16/04/2020  12:52               460 public.key
               2 File          2.162 byte
               2 Directory  86.738.501.632 byte disponibili
```

the file private.key is the following:

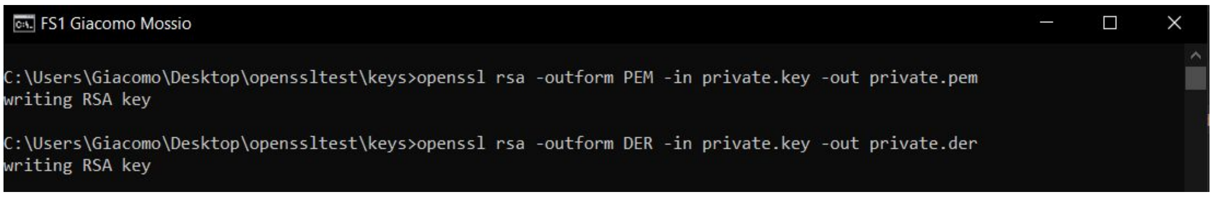
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAt7PM+oVVTYnAqyaMkytYwLpyoOFOxk46+GGO2LaH+jkav0YT
gacSlNnlzEHJOzzr/vHqolGDQdUWFy+5dO+pnfOGrhKPrxg9a9kZkx3KzfqYXyXO
2j0g+IfSuiyuaSidGP6Gpjj8eHGvHuYCT/gBNc1Tj1XtEgnPyvt35xgnZ4E+Vnpp
+9PckeZwOKK13ZsPXbN2sLUv9+6dd7ZVJQX+aPis845/vaWE7nlE8GhfHNBOeQQ7
M/Y/IvGp544yb5yyreXu+bLbcZff+O8r63ooADJTmkgOYVjofzQvKmd/ZVkBcdOK
9pB3P57HErhHQHebYWm0VKuet8D74ghmSo0tiQIDAQABAoIBAFPwiocmibXmbe1G
rhFJS0K1b09n5FDkyjlYiqcnsLZ7hWdFJACoUOtHLaPJo6O30LYr4GTPDkT0kyWB
HyL5p+2Oq0m7HYCKePZguzjn8hVwnlNLuVsrd3dYyMddR0yxzCbxlqE5hNOcTsK1
ovTbg+ILslTalHsblKFRipZECS1XsK43rzFQhB5kVj9oAI2ahlFtPsIbD003YFvv

vbB4qZZInz2jxyQMun3GYBg6Qzt/tCPpGRDWyCYUkAoygphg8N68Heq/pfCj1OnC
3Iru0aqTbA/Gp2WwHPz/zccr4Hr6g08pxtuGezXLVNZCHn9+xJnLbqdgIxRxQ8U0
7EJinyECgYEA5CWJ1TnC34+bqhMOTbHTKPsFsTHplr6oCYLCUNP/MhXJVomThHV8
sH+wyIeaV7HjvmBOQQHesWxSZSKVH2hB1u1VZft+Axx/DgcPdSIH7eKdbN2UjaNP
az8uAOVbdgEoltUfihyIY7QZceeTSGt0iEFqgMzdeP6OPscjycTzVzUCgYEAziE0
/jTg0ASP6Rz+ojhfWYmCxRLK1Y7pNmNvnhJuCDEtItQp9Ws+m3oTwAHtKwQ+9WKU
DDMnQfbeRYymNp41qSnkvAV+YBEIs3ar8LtdBWRoNAHfDqGOkelQ9VOvdP7PVbAH
85ttJm9xNB9oUZZFStLaWjtw+sCqtanj3RrWQ4UCgYEAp7hc+q7q7xUQ19gIdN+c
n4zio2BRG+vvdoZ1OZyQCA7yI7ciBoekcdB9cv4VXhC9n3AHN00bnI1IiWcB/8PS
WyHA3SckfT8OvI4+M9sfvycQmj7O/c0oDmy08h2qvjad0zy8YfkV305AQnd+1AeC
9MQ6epOg4Vkp00bsrWEnj10CgYBr83QfVW4PQvRjLy4josY8zAlSUF8trKRyOW0x
KZG6dJag0OAQiVPGdQ521gtxJc4bEL9cDEYZA8KwRdKedAjCSRBjb7UZdpJJ+nJd
pqI2urjf08MPlIhn3frqICOZcAEBbSkc6V8ma+KrW290mvA7GWYFL3Ahwlkkvo55
Mx7XSQKBgDcVqwZ18YBTaXqT19WwD0igOrzsy2SAFwxHWZvnMTOE/u0zk2jGrzuP
fcUD0aHRZiEvcoAbKCjMBDSr0JidflmEwdANr1vJacd5zJVcz26sJn/lIlxbu3oY
kDu8B6isY2+I8+qD24t4gT/A5oS+ZhrpC1BTst6TuN/RJv/WJKrC
-----END RSA PRIVATE KEY-----

And the file public.key is the following:

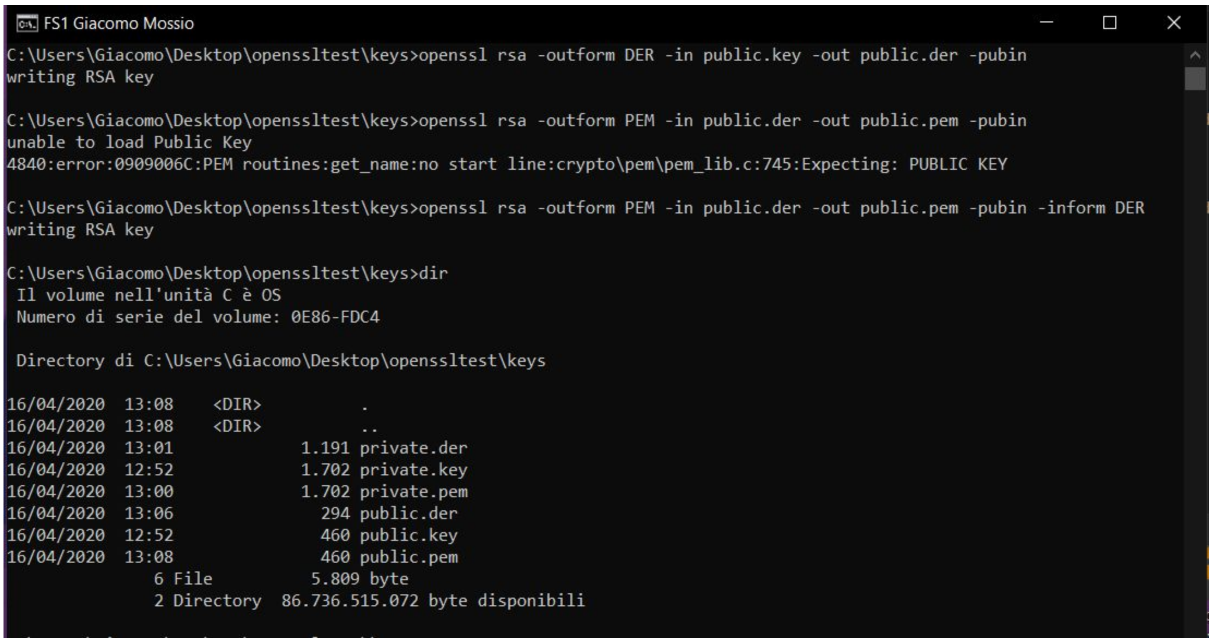
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt7PM+oVVTYnAqyaMkytY
wLpyoOFOxk46+GGO2LaH+jkav0YTgacSlNnlzEHJOzzr/vHqolGDQdUWFy+5dO+p
nfOGrhKPrxg9a9kZkx3KzfqYXyXO2j0g+IfSuiyuaSidGP6Gpjj8eHGvHuYCT/gB
Nc1Tj1XtEgnPyvt35xgnZ4E+Vnpp+9PckeZwOKK13ZsPXbN2sLUv9+6dd7ZVJQX+
aPis845/vaWE7nlE8GhfHNBOeQQ7M/Y/IvGp544yb5yyreXu+bLbcZff+O8r63oo
ADJTmkgOYVjofzQvKmd/ZVkBcdOK9pB3P57HErhHQHebYWm0VKuet8D74ghmSo0t
iQIDAQAB
-----END PUBLIC KEY-----

- Exportar dicho par de claves (pública y privada) en formato PEM (textual) y DER (binario). Utilizar los comandos de conversión de PEM a DER y viceversa.



FS1 Giacomo Mossio

```
C:\Users\Giacomo\Desktop\openssltest\keys>openssl rsa -outform PEM -in private.key -out private.pem
writing RSA key

C:\Users\Giacomo\Desktop\openssltest\keys>openssl rsa -outform DER -in private.key -out private.der
writing RSA key
```

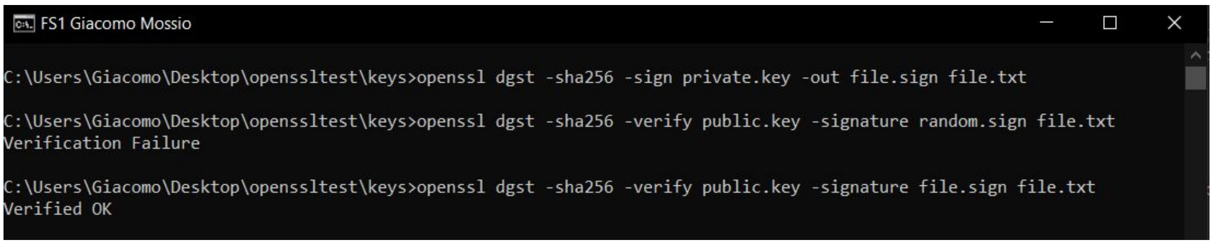


```
C:\Users\Giacomo\Desktop\openssltest\keys>openssl rsa -outform DER -in public.key -out public.der -pubin
writing RSA key

C:\Users\Giacomo\Desktop\openssltest\keys>openssl rsa -outform PEM -in public.der -out public.pem -pubin
unable to load Public Key
4840:error:0909006C:PEM routines:get_name:no start line:crypto\pem\pem_lib.c:745:Expecting: PUBLIC KEY

C:\Users\Giacomo\Desktop\openssltest\keys>openssl rsa -outform PEM -in public.der -out public.pem -pubin -inform DER
writing RSA key

C:\Users\Giacomo\Desktop\openssltest\keys>dir
 Il volume nell'unità C è OS
 Numero di serie del volume: 0E86-FDC4

 Directory di C:\Users\Giacomo\Desktop\openssltest\keys

16/04/2020  13:08    <DIR>          .
16/04/2020  13:08    <DIR>          ..
16/04/2020  13:01             1.191 private.der
16/04/2020  12:52             1.702 private.key
16/04/2020  13:00             1.702 private.pem
16/04/2020  13:06               294 public.der
16/04/2020  12:52               460 public.key
16/04/2020  13:08               460 public.pem
               6 File          5.809 byte
               2 Directory  86.736.515.072 byte disponibili
```

The file public.der is:

```
3082 0122 300d 0609 2a86 4886 f70d 0101
0105 0003 8201 0f00 3082 010a 0282 0101
00b7 b3cc fa85 554d 89c0 ab26 8c93 2b58
c0ba 72a0 e14e c64e 3af8 618e d8b6 87fa
391a bf46 1381 a712 94d9 e5cc 41c9 3b3c
ebfe f1ea a251 8341 d516 172f b974 efa9
9df3 86ae 128f af18 3d6b d919 931d cacd
fa98 5f25 ceda 3d20 f887 d2ba 2cae 6928
9d18 fe86 a638 fc78 71af 1ee6 024f f801
35cd 538f 55ed 1209 cfca fb77 e718 2767
813e 567a 69fb d3dc 91e6 7038 a2b5 dd9b
0f5d b376 b0b5 2ff7 ee9d 77b6 5525 05fe
68f8 acf3 8e7f bda5 84ee 7944 f068 5f1c
d04e 7904 3b33 f63f 22f1 a9e7 8e32 6f9c
b2ad e5ee f9b2 db71 97df f8ef 2beb 7a28
0032 539a 480e 6158 e87f 342f 2a67 7f65
5901 71d3 8af6 9077 3f9e c712 b847 4077
9b61 69b4 54ab 9eb7 c0fb e208 664a 8d2d
8902 0301 0001
```

- Con los dos pares de claves asimétricas creadas, firmar y comprobar la firma del resumen (con SHA-256) de un texto cualquiera.



```
C:\Users\Giacomo\Desktop\openssltest\keys>openssl dgst -sha256 -sign private.key -out file.sign file.txt

C:\Users\Giacomo\Desktop\openssltest\keys>openssl dgst -sha256 -verify public.key -signature random.sign file.txt
Verification Failure

C:\Users\Giacomo\Desktop\openssltest\keys>openssl dgst -sha256 -verify public.key -signature file.sign file.txt
Verified OK
```

The file random.sign is an exact copy of the file.sign with a bit changed, the verification fails.

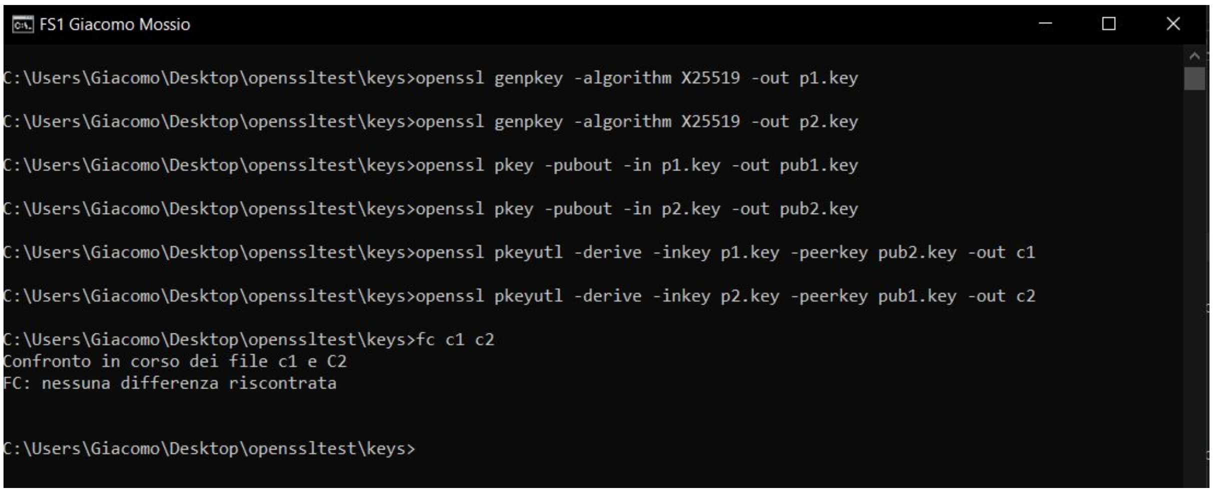
- Por último. Generar dos claves DH (preferiblemente con curva elíptica X25519) y demostrar que la combinación pública1-privada2 genera el mismo secreto que la combinación privada1-pública2.

For generate DH keys we need the followings commands:

openssl genpkey [-help] [-out filename] [-outform PEM|DER] [-pass arg] [-cipher] [-engine id] [-paramfile file] [-algorithm alg] [-pkeyopt opt:value] [-genparam] [-text]
is used for generate the private keys.

openssl pkey [-help] [-inform PEM|DER] [-outform PEM|DER] [-in filename] [-passin arg] [-out filename] [-passout arg] [-traditional] [-cipher] [-text] [-text_pub] [-noout] [-pubin] [-pubout] [-engine id] [-check] [-pubcheck]
is used for generate the public keys

openssl pkeyutl [-help] [-in file] [-out file] [-sigfile file] [-inkey file] [-keyform PEM|DER|ENGINE] [-passin arg] [-peerkey file] [-peerform PEM|DER|ENGINE] [-pubin] [-certin] [-rev] [-sign] [-verify] [-verifyrecover] [-encrypt] [-decrypt] [-derive] [-kdf algorithm] [-kdflen length] [-pkeyopt opt:value] [-hexdump] [-asn1parse] [-rand file...] [-writerand file] [-engine id] [-engine_impl]
is used for combine the keys



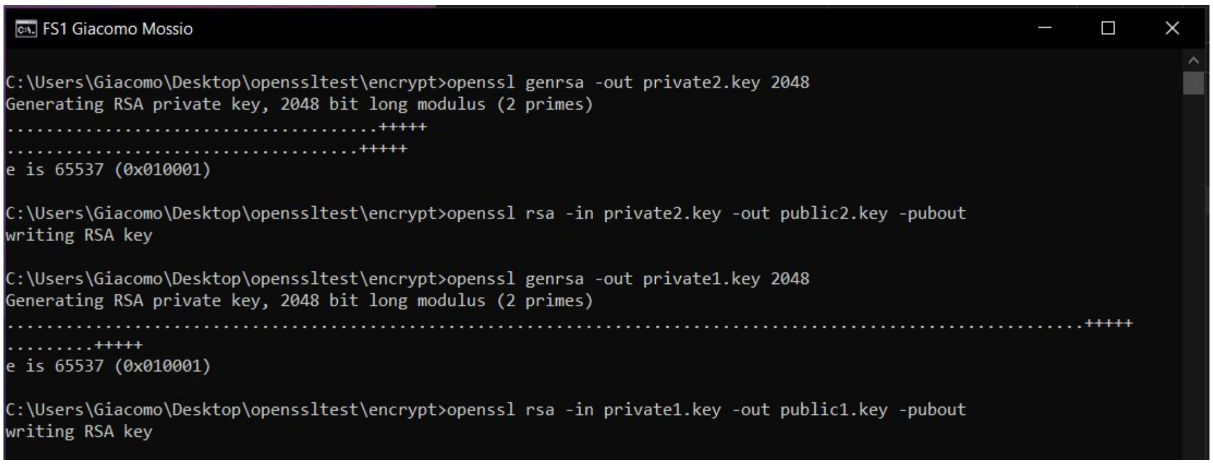
```
C:\Users\Giacomo\Desktop\openssltest\keys>openssl genpkey -algorithm X25519 -out p1.key

C:\Users\Giacomo\Desktop\openssltest\keys>openssl genpkey -algorithm X25519 -out p2.key

C:\Users\Giacomo\Desktop\openssltest\keys>openssl pkey -pubout -in p1.key -out pub1.key

C:\Users\Giacomo\Desktop\openssltest\keys>openssl pkey -pubout -in p2.key -out pub2.key

C:\Users\Giacomo\Desktop\openssltest\keys>openssl pkeyutl -derive -inkey p1.key -peerkey pub2.key -out c1

C:\Users\Giacomo\Desktop\openssltest\keys>openssl pkeyutl -derive -inkey p2.key -peerkey pub1.key -out c2

C:\Users\Giacomo\Desktop\openssltest\keys>fc c1 c2
Confronto in corso dei file c1 e C2
FC: nessuna differenza riscontrata


C:\Users\Giacomo\Desktop\openssltest\keys>
```

fc is just a command for compare files and the output is that there are no differences.

1.3 Cifrado Asimétrico de documentos

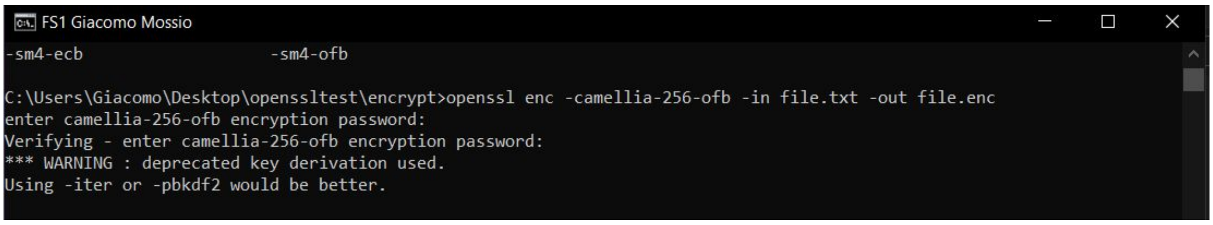
- Seguir los siguientes pasos para crear un documento de texto y cifrarlo con openssl, enviando a un compañero el documento cifrado y la clave, cifrada a su vez con su clave pública RSA (que previamente ha de conocerse). Codificarlo todo en Base64 y enviar un correo electrónico con tres partes:
    - 1.- Documento cifrado (indicando algoritmo utilizado)
    - 2.- Clave simétrica empleada, cifrada con la clave pública del receptor
    - 3.- Resumen del documento original (indicando algoritmo) cifrado con la clave privada del emisor
    -

We begin by creating the 4 keys needed



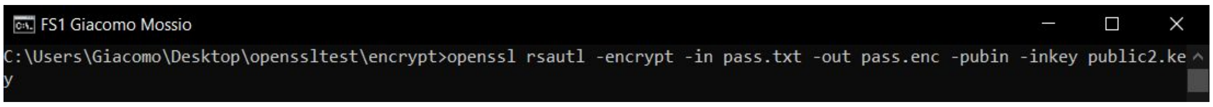
```
C:\Users\Giacomo\Desktop\openssltest\encrypt>openssl genrsa -out private2.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
......................................+++++
....................................+++++
e is 65537 (0x010001)

C:\Users\Giacomo\Desktop\openssltest\encrypt>openssl rsa -in private2.key -out public2.key -pubout
writing RSA key

C:\Users\Giacomo\Desktop\openssltest\encrypt>openssl genrsa -out private1.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
..............................................................................................................+++++
.........+++++
e is 65537 (0x010001)

C:\Users\Giacomo\Desktop\openssltest\encrypt>openssl rsa -in private1.key -out public1.key -pubout
writing RSA key
```

then we encrypt our document (file.txt)



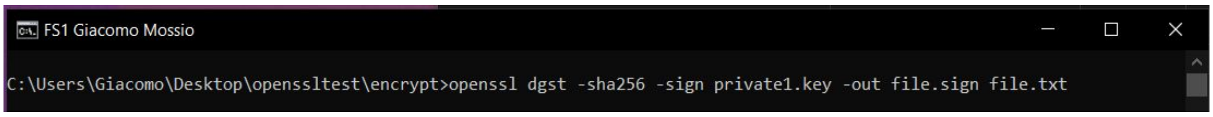
```
-sm4-ecb                   -sm4-ofb

C:\Users\Giacomo\Desktop\openssltest\encrypt>openssl enc -camellia-256-ofb -in file.txt -out file.enc
enter camellia-256-ofb encryption password:
Verifying - enter camellia-256-ofb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

Then we encrypt the password with the public key of the receiver (public2.key)



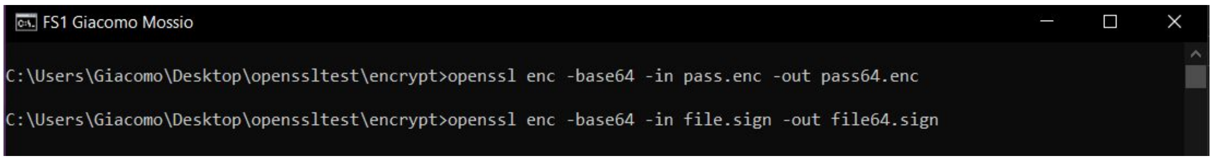
```
C:\Users\Giacomo\Desktop\openssltest\encrypt>openssl rsautl -encrypt -in pass.txt -out pass.enc -pubin -inkey public2.ke
y
```

Now we create the resume of the original file.txt with the private key of the sender



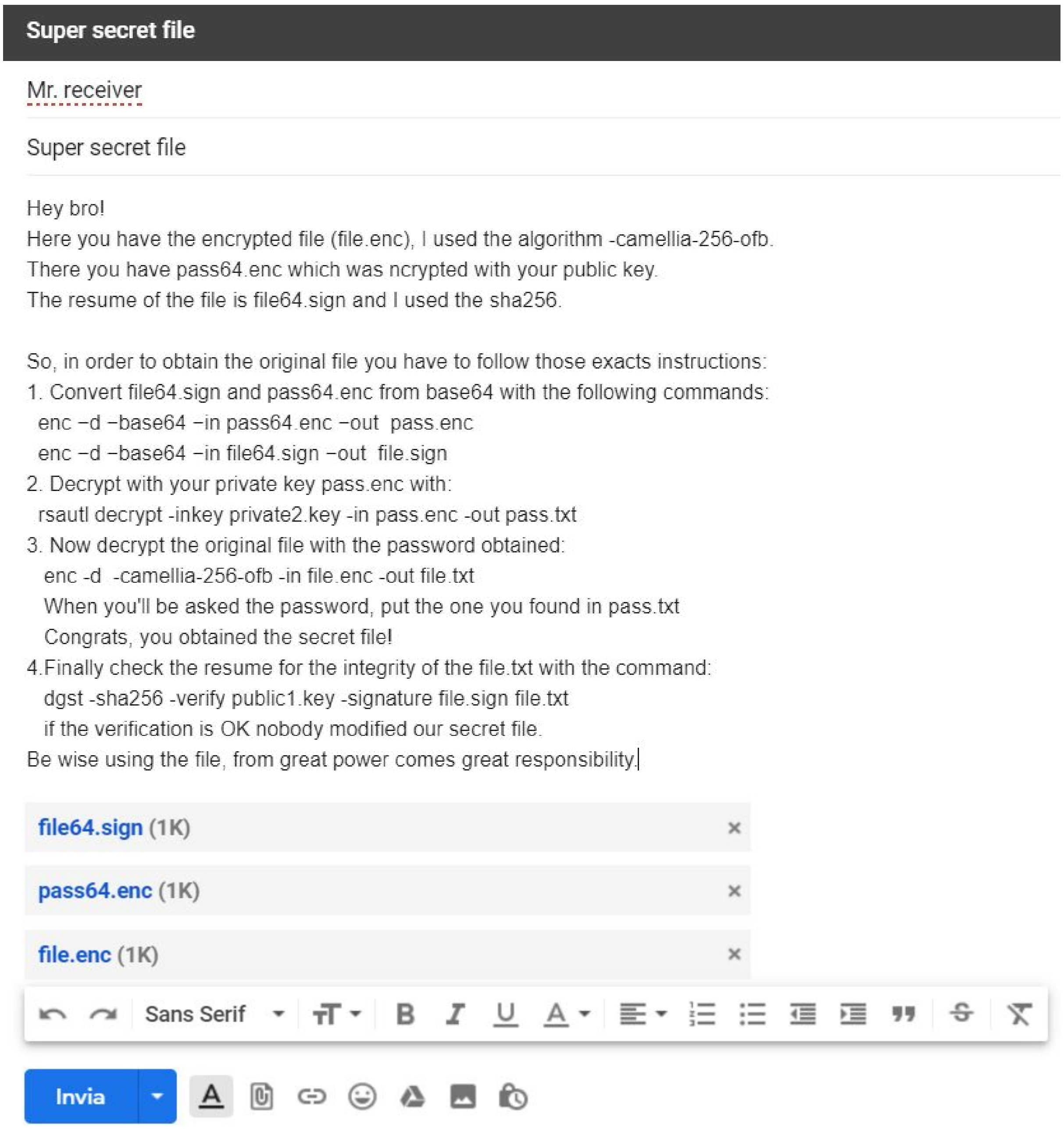
```
C:\Users\Giacomo\Desktop\openssltest\encrypt>openssl dgst -sha256 -sign private1.key -out file.sign file.txt
```

Then we proceed with the conversion in base64 of the encrypted password and the resume



```
C:\Users\Giacomo\Desktop\openssltest\encrypt>openssl enc -base64 -in pass.enc -out pass64.enc

C:\Users\Giacomo\Desktop\openssltest\encrypt>openssl enc -base64 -in file.sign -out file64.sign
```

Now we'll send our email like this:


Super secret file

Mr. receiver

Super secret file

Hey bro!
Here you have the encrypted file (file.enc), I used the algorithm -camellia-256-ofb.
There you have pass64.enc which was ncrypted with your public key.
The resume of the file is file64.sign and I used the sha256.

So, in order to obtain the original file you have to follow those exacts instructions:
1. Convert file64.sign and pass64.enc from base64 with the following commands:

```
 enc –d –base64 –in pass64.enc –out  pass.enc
 enc –d –base64 –in file64.sign –out  file.sign
```

2. Decrypt with your private key pass.enc with:

```
 rsautl decrypt -inkey private2.key -in pass.enc -out pass.txt
```

3. Now decrypt the original file with the password obtained:

```
  enc -d  -camellia-256-ofb -in file.enc -out file.txt
```

  When you'll be asked the password, put the one you found in pass.txt
  Congrats, you obtained the secret file!

4.Finally check the resume for the integrity of the file.txt with the command:

```
  dgst -sha256 -verify public1.key -signature file.sign file.txt
```

  if the verification is OK nobody modified our secret file.

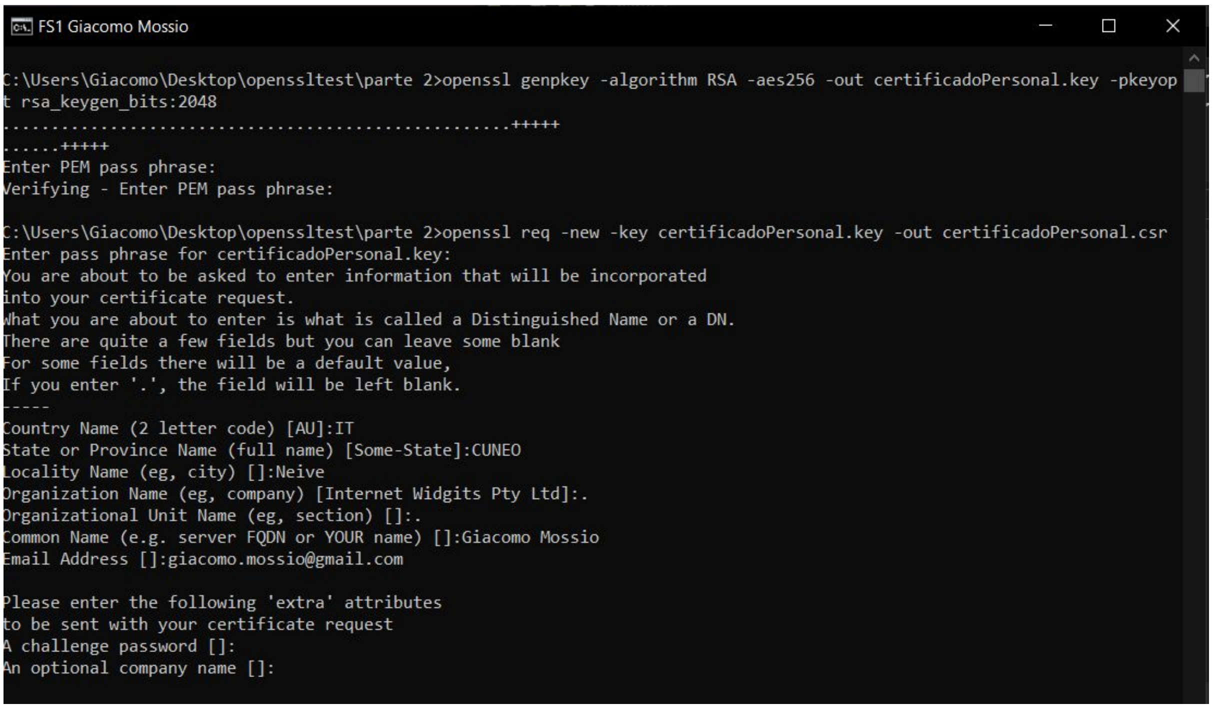
Be wise using the file, from great power comes great responsibility.

PARTE 2 - Envío, recepción y DECODIFICACIÓN MANUAL de mensajes S/MIME firmados y cifrados, empleando certificados creados por el estudiante y firmados con el certificado raíz de prácticas de esta asignatura.

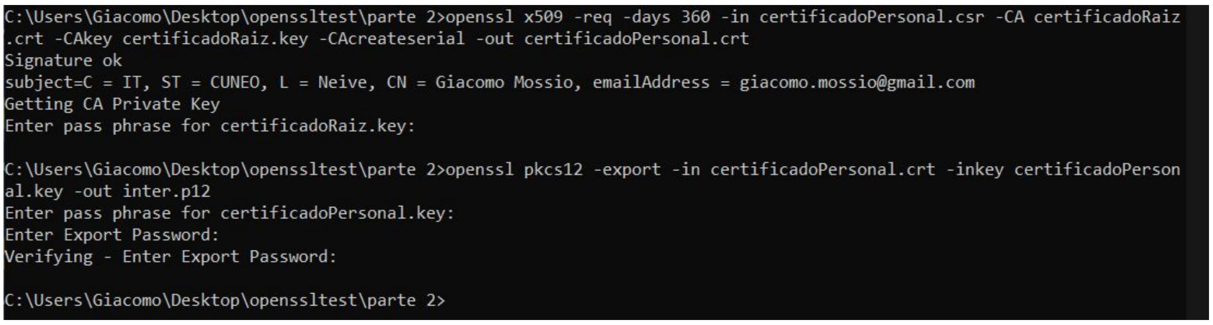
2.1.- El estudiante previamente ha de crear un certificado X.509 y firmarlo con el certificado de Autoridad disponible en la página de la asignatura

The steps to do in order to create our personal certificate are the followings:

1. Generate a private key for our certificate
   ```
   openssl genpkey -algorithm RSA -aes256 -out
   certificadoPersonal.key -pkeyopt rsa_keygen_bits:2048
   ```
2. Create the CSR for our certificate
   ```
   openssl req -new -key certificadoPersonal.key -out
   certificadoPersonal.csr
   ```
3. Create our personal certificate using our CSR, the root certificate and the root key
   ```
   openssl x509 -req -days 360 -in certificadoPersonal.csr
   -CA certificadoRaiz.crt -CAkey certificadoRaiz.key
   -CAcreateserial -out certificadoPersonal.crt
   ```
4. Create the “inter.p12” folder for saving personal keys and the certificate grouped in a encrypted folder:
   ```
   openssl pkcs12 -export -in certificadoPersonal.crt -inkey
   certificadoPersonal.key -out inter.p12
   ```



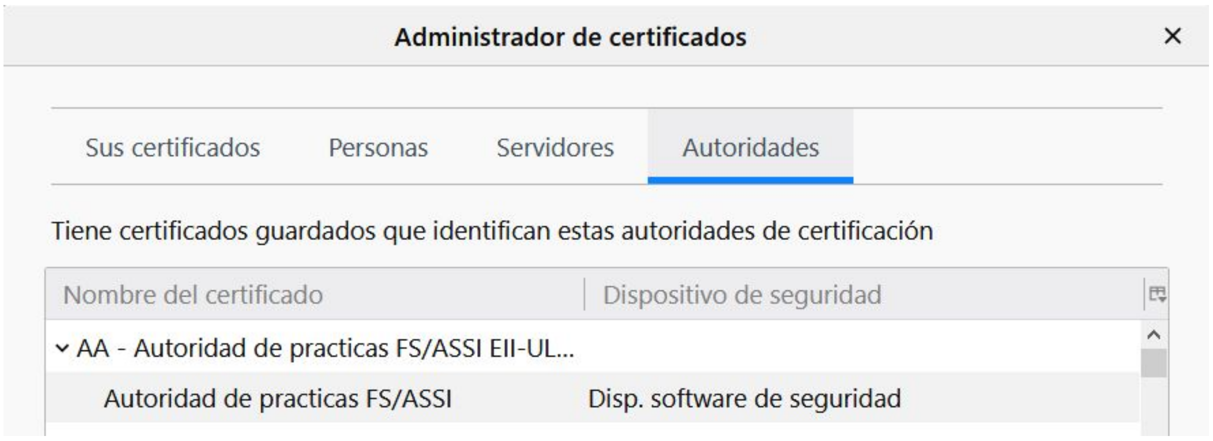
```
:\Users\Giacomo\Desktop\openssltest\parte 2>openssl genpkey -algorithm RSA -aes256 -out certificadoPersonal.key -pkeyop
t rsa_keygen_bits:2048
...............................................+++++
......+++++
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

:\Users\Giacomo\Desktop\openssltest\parte 2>openssl req -new -key certificadoPersonal.key -out certificadoPersonal.csr
Enter pass phrase for certificadoPersonal.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:CUNEO
Locality Name (eg, city) []:Neive
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
Organizational Unit Name (eg, section) []:.
Common Name (e.g. server FQDN or YOUR name) []:Giacomo Mossio
Email Address []:giacomo.mossio@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```



```
C:\Users\Giacomo\Desktop\openssltest\parte 2>openssl x509 -req -days 360 -in certificadoPersonal.csr -CA certificadoRaiz
.crt -CAkey certificadoRaiz.key -CAcreateserial -out certificadoPersonal.crt
Signature ok
subject=C = IT, ST = CUNEO, L = Neive, CN = Giacomo Mossio, emailAddress = giacomo.mossio@gmail.com
Getting CA Private Key
Enter pass phrase for certificadoRaiz.key:

C:\Users\Giacomo\Desktop\openssltest\parte 2>openssl pkcs12 -export -in certificadoPersonal.crt -inkey certificadoPerson
al.key -out inter.p12
Enter pass phrase for certificadoPersonal.key:
Enter Export Password:
Verifying - Enter Export Password:

C:\Users\Giacomo\Desktop\openssltest\parte 2>
```

2.2.- Instalar ambos certificados (usuario y autoridad) en un cliente de correo Thunderbird y enviar un mensaje firmado y cifrado a la dirección de un compañero de la asignatura, cuyo certificado ha de ser previamente importado en el cliente de correo, después de recibir un mensaje firmado de ese compañero.

We proceed with the importation and installation of the certificates in Thunderbird:

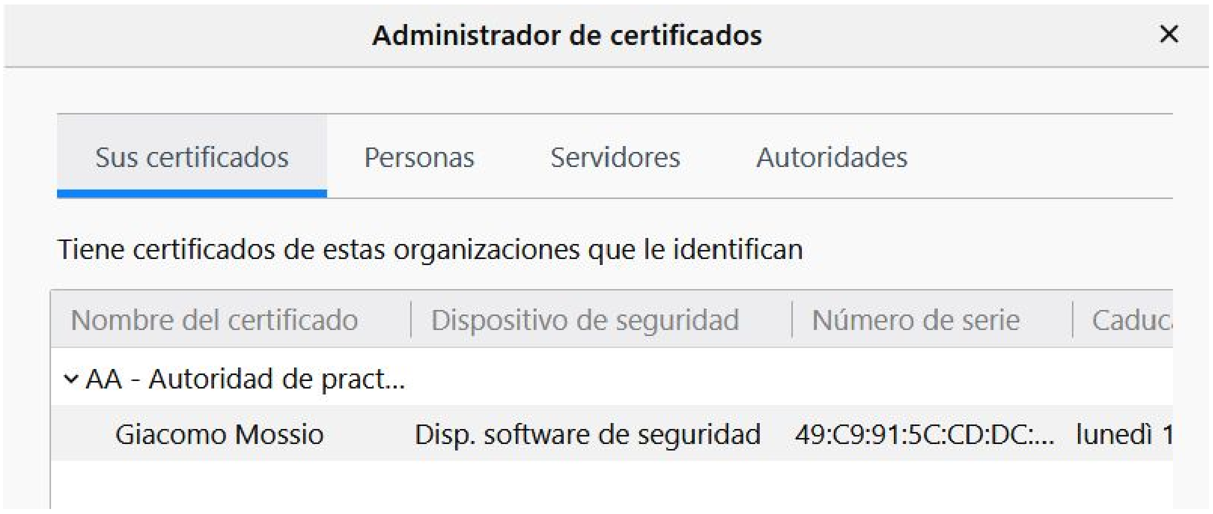


Administrador de certificados

Sus certificados | Personas | Servidores | Autoridades

Tiene certificados guardados que identifican estas autoridades de certificación

Nombre del certificado	Dispositivo de seguridad
AA - Autoridad de practicas FS/ASSI EII-UL...	
Autoridad de practicas FS/ASSI	Disp. software de seguridad



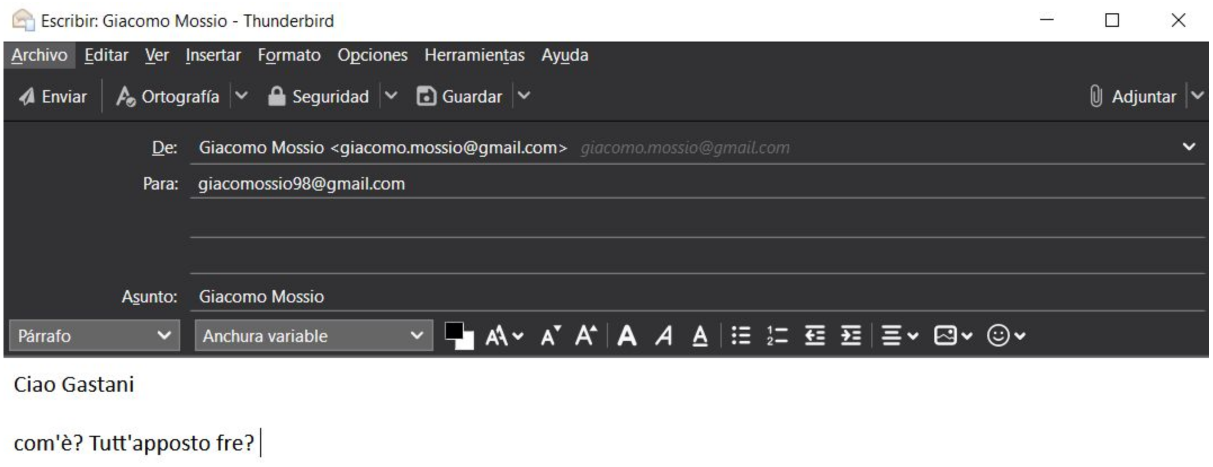
Administrador de certificados

Sus certificados | Personas | Servidores | Autoridades

Tiene certificados de estas organizaciones que le identifican

Nombre del certificado	Dispositivo de seguridad	Número de serie	Caduc
AA - Autoridad de pract...			
Giacomo Mossio	Disp. software de seguridad	49:C9:91:5C:CD:DC:...	lunedì 1

Now we’re gonna send an email encrypted and signed to a friend (another email mine) so that we can send our public key.



Escribir: Giacomo Mossio - Thunderbird

De: Giacomo Mossio <giacomo.mossio@gmail.com> giacomo.mossio@gmail.com

Para: giacomossio98@gmail.com

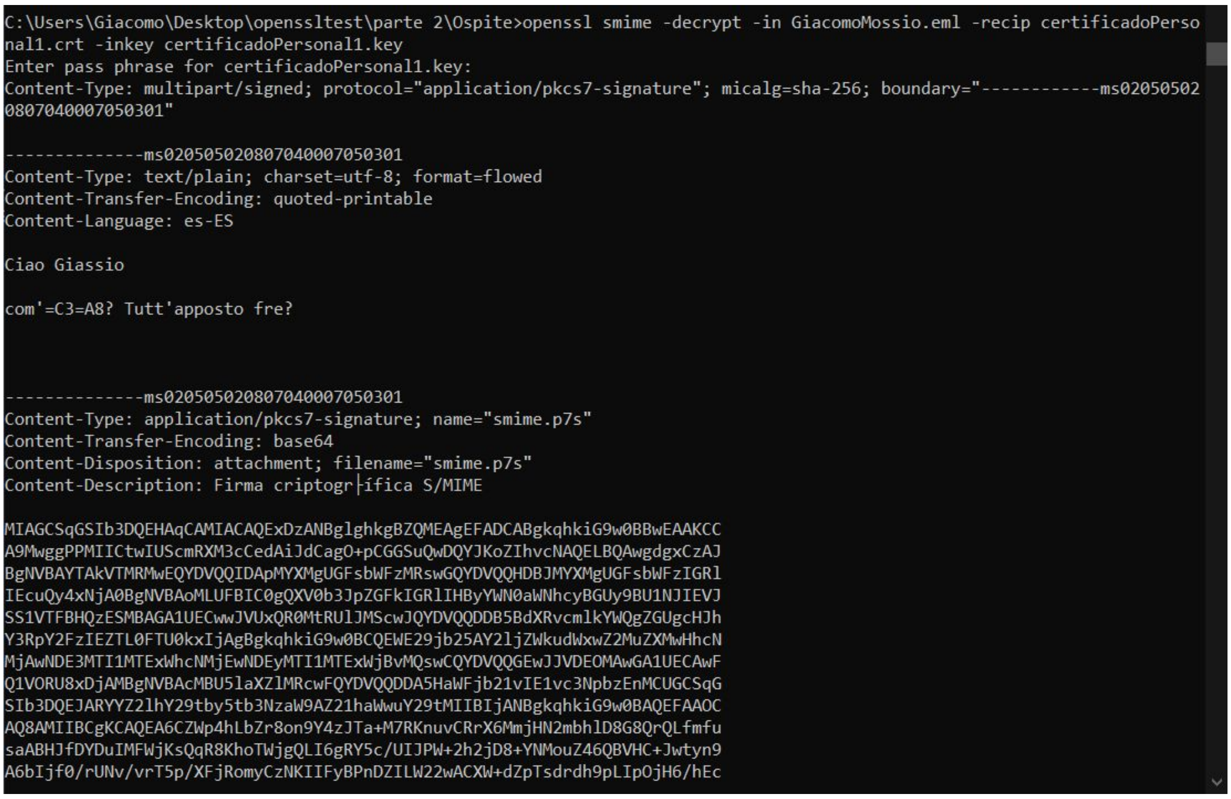
Asunto: Giacomo Mossio

Ciao Gastani

com'è? Tutt'apposto fre?

2.3.- Una vez recibido el correo firmado y cifrado, hemos de decodificar manualmente, con la ayuda de la utilidad “openssl smime” el texto del mensaje recibido, empleando nuestro certificado original (con la clave privada) y el certificado del compañero.

Once we’ve received the message, we have to decrypt it, in order to do so we need:
-the encrypted message
-the personal certificate of the sender
-the key of the certificate of the sender
we can then execute the command:
`openssl smime -decrypt -in GiacomoMossio.eml -recip certificadoPersonal1.crt -inkey certificadoPersonal1.key`
and we’ll obtain our original sent message decrypted



```
C:\Users\Giacomo\Desktop\openssltest\parte 2\Ospite>openssl smime -decrypt -in GiacomoMossio.eml -recip certificadoPerso
nal1.crt -inkey certificadoPersonal1.key
Enter pass phrase for certificadoPersonal1.key:
Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg=sha-256; boundary="------------ms02050502
0807040007050301"

--------------ms020505020807040007050301
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: quoted-printable
Content-Language: es-ES

Ciao Giassio

com'=C3=A8? Tutt'apposto fre?



--------------ms020505020807040007050301
Content-Type: application/pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"
Content-Description: Firma criptogr├ífica S/MIME
```

```
MIAGCSqGSIb3DQEHAqCAMIACAQExDzANBglghkgBZQMEAgEFADCABgkqhkiG9w0BBwEAAKCC
A9MwggPPMIICtwIUScmRXM3cCedAiJdCagO+pCGGSuQwDQYJKoZIhvcNAQELBQAwgdgxCzAJ
BgNVBAYTAkVTMRMwEQYDVQQIDApMYXMgUGFsbWFzMRswGQYDVQQHDBJMYXMgUGFsbWFzIGRl
IEcuQy4xNjA0BgNVBAoMLUFBIC0gQXV0b3JpZGFkIGRlIHByYWN0aWNhcyBGUy9BU1NJIEVJ
SS1VTFBHQzESMBAGA1UECwwJVUxQR0MtRUlJMScwJQYDVQQDDB5BdXRvcmlkYWQgZGUgcHJh
Y3RpY2FzIEZTL0FTU0kxIjAgBgkqhkiG9w0BCQEWE29jb25AY2ljZWkudWxwZ2MuZXMwHhcN
MjAwNDE3MTI1MTExWhcNMjEwNDEyMTI1MTExWjBvMQswCQYDVQQGEwJJVDEOMAwGA1UECAwF
Q1VORU8xDjAMBgNVBAcMBU5laXZlMRcwFQYDVQQDDA5HaWFjb21vIE1vc3NpbzEnMCUGCSqG
SIb3DQEJARYYZ2lhY29tby5tb3NzaW9AZ21haWwuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEA6CZWp4hLbZr8on9Y4zJTa+M7RKnuvCRrX6MmjHN2mbhlD8G8QrQLfmfu
saABHJfDYDuIMFWjKsQqR8KhoTWjgQLI6gRY5c/UIJPW+2h2jD8+YNMouZ46QBVHC+Jwtyn9
A6bIjf0/rUNv/vrT5p/XFjRomyCzNKIIFyBPnDZILW22wACXW+dZpTsdrdh9pLIpOjH6/hEc
```

PARTE 3 - Configuración de un SERVIDOR WEB SEGURO (SHTTP) utilizando un certificado AUTOFIRMADO, con privilegios de Administrador en un servidor web – Apache, ISS, etc.

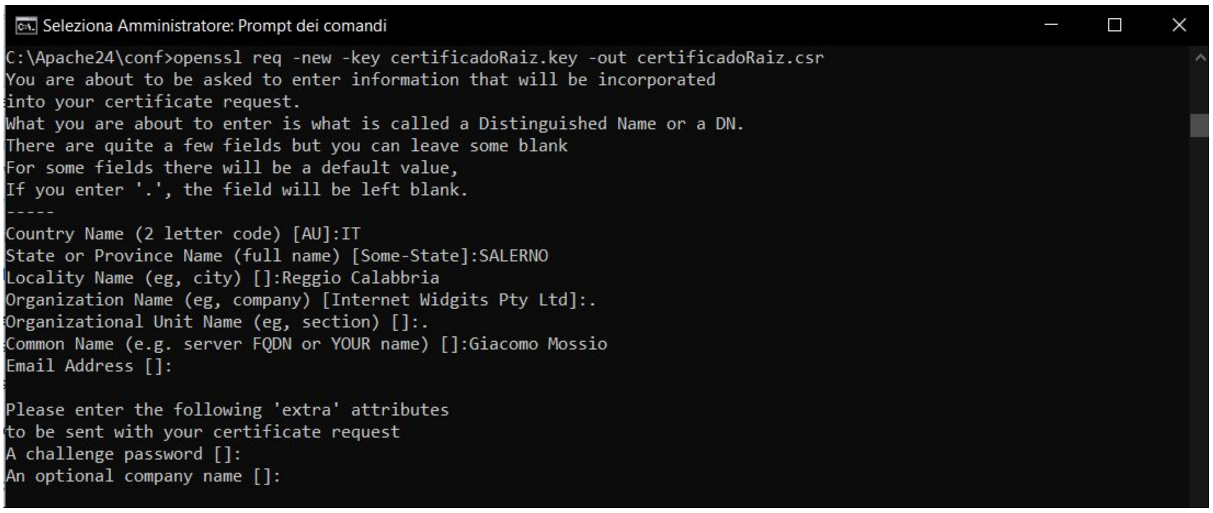
This tutorial is made for windows users.

The first step is to download the Apache web server from the official page (https://httpd.apache.org/) then we go to the download section, we click on “Files for Microsoft Windows” and then we download the “Apache Lounge” file.
Now we unzip it in our “C:\” (note that we must have the administration permissions to do so).
Then we open a prompt windows as administrator and we navigate to the apache folder, once we’re in we execute the command **`httpd -k install`** and we have the server installed.

Then we have to create our root certificate, in order to do so we execute those commands:

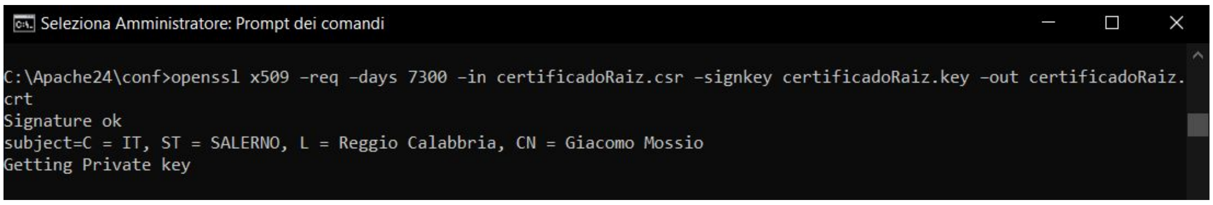
```
openssl genpkey -algorithm RSA -out certificadoRaiz.key
-pkeyopt rsa_keygen_bits:2048
```

```
openssl req -new -key certificadoRaiz.key -out
certificadoRaiz.csr
```


Seleziona Amministratore: Prompt dei comandi

```
C:\Apache24\conf>openssl req -new -key certificadoRaiz.key -out certificadoRaiz.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:SALERNO
Locality Name (eg, city) []:Reggio Calabbria
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
Organizational Unit Name (eg, section) []:.
Common Name (e.g. server FQDN or YOUR name) []:Giacomo Mossio
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

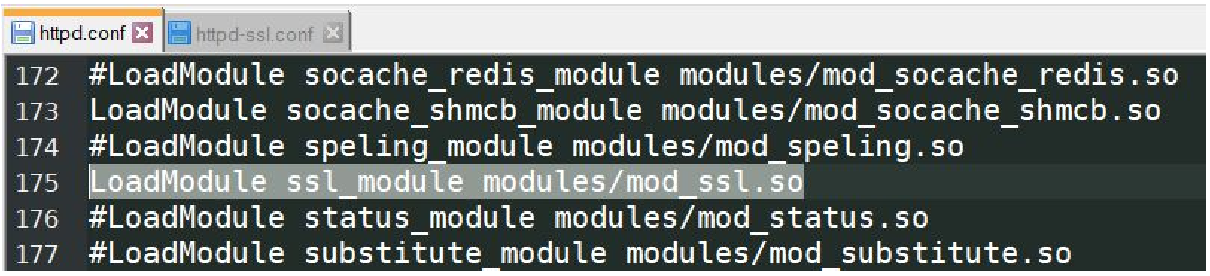
`openssl x509 -req -days 365 -in certificadoRaiz.csr -signkey certificadoRaiz.key -out certificadoRaiz.crt`



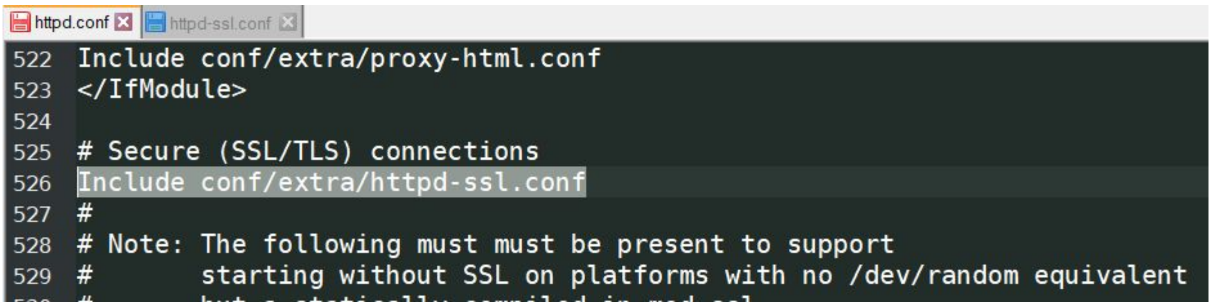
```
C:\Apache24\conf>openssl x509 -req -days 7300 -in certificadoRaiz.csr -signkey certificadoRaiz.key -out certificadoRaiz.
crt
Signature ok
subject=C = IT, ST = SALERNO, L = Reggio Calabbria, CN = Giacomo Mossio
Getting Private key
```

Now we have our certificate files and we have to install them in Apache, in order to do so we have to modify some parameters in the configuration files of Apache that will be founded in \Apache24\conf:

1. httpd.conf
   we search for those two lines and we modify them as shown

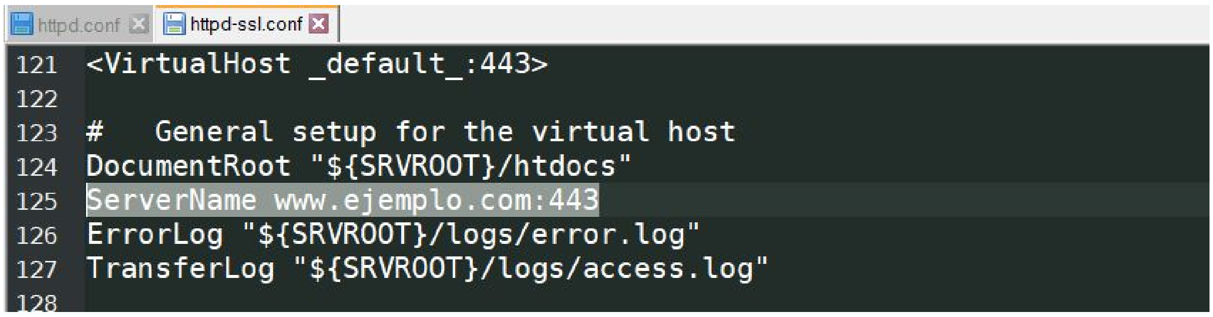


```
#LoadModule socache_redis_module modules/mod_socache_redis.so
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
#LoadModule speling_module modules/mod_speling.so
LoadModule ssl_module modules/mod_ssl.so
#LoadModule status_module modules/mod_status.so
#LoadModule substitute_module modules/mod_substitute.so
```

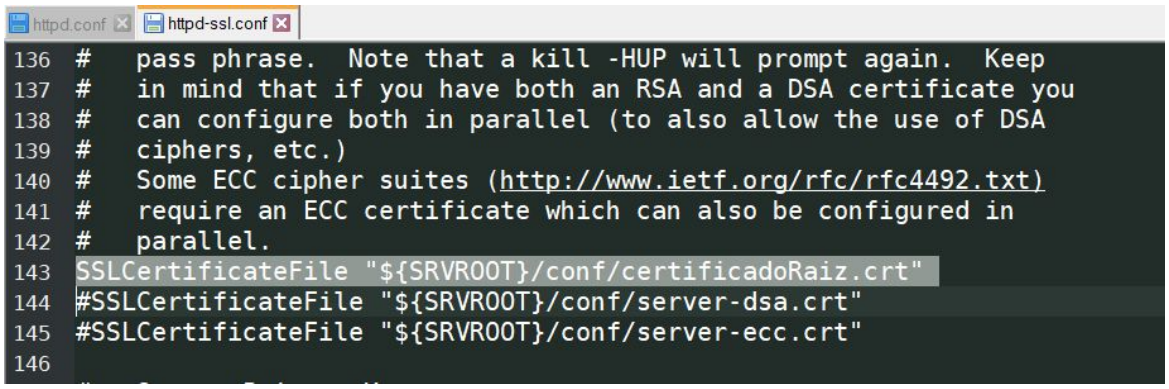


```
Include conf/extra/proxy-html.conf
</IfModule>

# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
#
# Note: The following must must be present to support
#       starting without SSL on platforms with no /dev/random equivalent
```

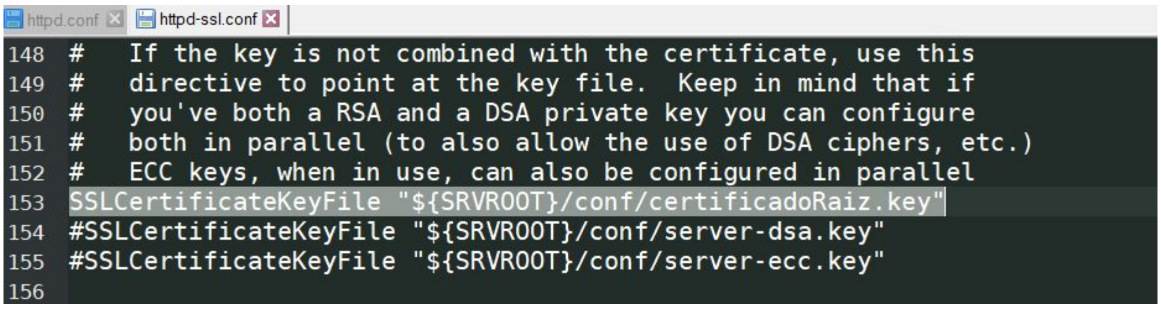
2. httpd-ssl.conf (in \Apache24\conf\extra)



```
<VirtualHost _default_:443>

#   General setup for the virtual host
DocumentRoot "${SRVROOT}/htdocs"
ServerName www.ejemplo.com:443
ErrorLog "${SRVROOT}/logs/error.log"
TransferLog "${SRVROOT}/logs/access.log"
```



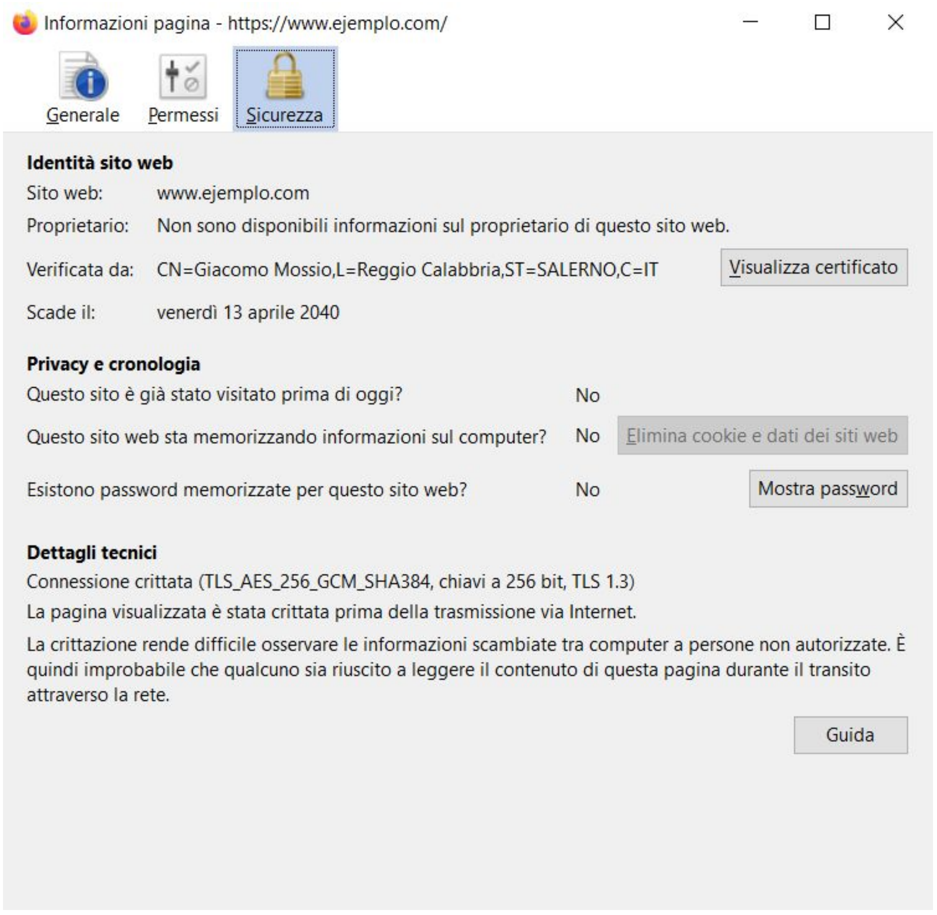
```
#   pass phrase.  Note that a kill -HUP will prompt again.  Keep
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
#   Some ECC cipher suites (http://www.ietf.org/rfc/rfc4492.txt)
#   require an ECC certificate which can also be configured in
#   parallel.
SSLCertificateFile "${SRVROOT}/conf/certificadoRaiz.crt"
#SSLCertificateFile "${SRVROOT}/conf/server-dsa.crt"
#SSLCertificateFile "${SRVROOT}/conf/server-ecc.crt"
```



```
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
#   ECC keys, when in use, can also be configured in parallel
SSLCertificateKeyFile "${SRVROOT}/conf/certificadoRaiz.key"
#SSLCertificateKeyFile "${SRVROOT}/conf/server-dsa.key"
#SSLCertificateKeyFile "${SRVROOT}/conf/server-ecc.key"
```

Now that we have our files configured we save them and then we execute the command **`httpd -k start`** and it's done.

If we visit the site https://www.ejemplo.com we can check our certificate


Informazioni pagina - https://www.ejemplo.com/
Generale
Permessi
Sicurezza
Visualizza certificato
Elimina cookie e dati dei siti web
Mostra password
Guida

Identità sito web

Sito web: www.ejemplo.com

Proprietario: Non sono disponibili informazioni sul proprietario di questo sito web.

Verificata da: CN=Giacomo Mossio,L=Reggio Calabbria,ST=SALERNO,C=IT

Scade il: venerdì 13 aprile 2040

Privacy e cronologia

Questo sito è già stato visitato prima di oggi? No

Questo sito web sta memorizzando informazioni sul computer? No

Esistono password memorizzate per questo sito web? No

Dettagli tecnici

Connessione crittata (TLS_AES_256_GCM_SHA384, chiavi a 256 bit, TLS 1.3)

La pagina visualizzata è stata crittata prima della trasmissione via Internet.

La crittazione rende difficile osservare le informazioni scambiate tra computer a persone non autorizzate. È quindi improbabile che qualcuno sia riuscito a leggere il contenuto di questa pagina durante il transito attraverso la rete.

Bibliografia:

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation used for 1.1
https://en.wikipedia.org/wiki/PBKDF2 used for 1.1
https://tools.ietf.org/html/rfc2898#section-3 used for 2.1
https://openssl.cicei.com used for section 2
https://www.openssl.org/docs/man1.1.1/man1/ used for kind of all the relation