

USO DE OPENSSL PARA CIFRADO DE MENSAJES Y FICHEROS

Práctica 4

Informe de prácticas

Parte 1

1. Usa la página de manual de openssl para obtener información sobre la forma de usar esta herramienta para cifrar y descifrar ficheros.

OpenSSL es un proyecto de software libre que consiste en un paquete de herramientas de administración y bibliotecas relacionadas con la criptografía, que suministran funciones criptográficas a otros paquetes como OpenSSH y navegadores web.

Estas herramientas ayudan al sistema a implementar el Secure Sockets Layer (SSL), así como otros protocolos relacionados con la seguridad, como el Transport Layer Security (TLS). OpenSSL también permite crear certificados digitales que pueden aplicarse a un servidor.

El protocolo SSL trabaja en la capa de aplicación proporcionando seguridad a otros protocolos como el HTTP (que pasa a llamarse HTTPS) o FTP (que pasa a llamarse FTPS).

Cifrado de los archivos

1. Genera una clave aleatoriamente, de gran longitud.
2. Cifra el archivo con la clave generada en el punto anterior.
3. Ciframos la clave generada en el primer punto con nuestra clave pública.

Descifrado de los archivos

1. Desciframos la clave generada en el punto 1 anterior que fue cifrada con la clave pública en el punto 3.
2. Desciframos el archivo con la clave que acabamos de descifrar.

Algunos de los comandos de OpenSSL para encriptar o desencriptar son los siguientes:

GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN

Práctica 4. Uso de OpenSSL para Cifrado de Mensajes y Ficheros

Jhon Steeven Cabanilla Alvarado

```
Encoding and Cipher Commands
The following aliases provide convenient access to the most used encodings and ciphers.

Depending on how OpenSSL was configured and built, not all ciphers listed here may be present. See enc(1) for more information and command usage.

aes128, aes-128-cbc, aes-128-cfb, aes-128-ctr, aes-128-ecb, aes-128-ofb
AES-128 Cipher

aes192, aes-192-cbc, aes-192-cfb, aes-192-ctr, aes-192-ecb, aes-192-ofb
AES-192 Cipher

aes256, aes-256-cbc, aes-256-cfb, aes-256-ctr, aes-256-ecb, aes-256-ofb
AES-256 Cipher

aria128, aria-128-cbc, aria-128-cfb, aria-128-ctr, aria-128-ecb, aria-128-ofb
Aria-128 Cipher

aria192, aria-192-cbc, aria-192-cfb, aria-192-ctr, aria-192-ecb, aria-192-ofb
Aria-192 Cipher

aria256, aria-256-cbc, aria-256-cfb, aria-256-ctr, aria-256-ecb, aria-256-ofb
Aria-256 Cipher

base64
Base64 Encoding

bf, bf-cbc, bf-cfb, bf-ecb, bf-ofb
Blowfish Cipher

camellia128, camellia-128-cbc, camellia-128-cfb, camellia-128-ctr, camellia-128-ecb, camellia-128-ofb
Camellia-128 Cipher

camellia192, camellia-192-cbc, camellia-192-cfb, camellia-192-ctr, camellia-192-ecb, camellia-192-ofb
Camellia-192 Cipher

camellia256, camellia-256-cbc, camellia-256-cfb, camellia-256-ctr, camellia-256-ecb, camellia-256-ofb
Camellia-256 Cipher

cast, cast-cbc
CAST Cipher

cast5-cbc, cast5-cfb, cast5-ecb, cast5-ofb
CAST5 Cipher
```

Como comando para cifrar ficheros utilizaremos el siguiente:

openssl -encryption_cipher -salt -in fich_input -out fich_output

- encryption_cipher: cifrado que se va a utilizar.
- salt: añade fuerza a la encriptación.
- in fich_input: especifica el archivo de entrada.
- out fich_output: especifica el archivo de salida.

Como comando para descifrar utilizaremos el siguiente:

openssl -encryption_cipher -d -in fich_output -out fich_original

- d: descifra los datos.
- in fich_output: especifica los datos a descifrar.
- out fich_original: especifica el archivo donde se colocan los datos descifrados.

2. Experimenta con diversos algoritmos (AES, DES, CAMELLIA,) y modos de cifrado (ECB, CFB, CBC). Utiliza diferentes ficheros de entrada (texto y binarios) y con diferentes claves y vectores de inicialización.

Como ficheros de entrada utilizaremos los ficheros *fichero1.txt* (texto), el cual contiene lo siguiente:

GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN
Práctica 4. Uso de OpenSSL para Cifrado de Mensajes y Ficheros
Jhon Steeven Cabanilla Alvarado

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cat fichero1.txt
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Received: from murder (mail.umich.edu [141.211.14.90])
        by frankenstein.mail.umich.edu (Cyrus v2.3.8) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
X-Sieve: CMU Sieve 2.3
Received: from murder ([unix socket])
        by mail.umich.edu (Cyrus v2.2.12) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$
```

y *fichero2.sh* (binario) que es una copia del contenido del fichero situado en la siguiente ruta:

/usr/lib/udev/hwdb.bin

1. Comenzamos utilizando el algoritmo **AES** con el modo de cifrado **ECB**, el cual se trata del modo de cifrado más simple de todos, pues se limita a partir el mensaje en bloques y cifrarlos por separado.

Como clave privada utilizamos: *password*.

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl aes-128-ecb -salt -in fichero1.txt -out fich1.encode.aes
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cat fich1.encode.aes
By@h]/\*uG@5@L) @d@^ER1'~Lg0;42cu@7VuH@B i@B@}B@B?~`ju"bP14"Yh+mvK\2Br,@@ q@B>2WxcI*s(HKq@ne@B@B@B@S@e@sf?oae@.sDh
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$
```

Una vez que hemos comprobado que efectivamente el fichero ha sido encriptado, procedemos a realizar la descryptación correspondiente.

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl aes-128-ecb -d -in fich1.encode.aes -out fich1.dec.aes
enter aes-128-ecb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ ls
fich1.dec.aes  fich1.encode.aes  fichero1.txt  fichero2.bin  shannon_entropy.py
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cat fich1.dec.aes
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Received: from murder (mail.umich.edu [141.211.14.90])
        by frankenstein.mail.umich.edu (Cyrus v2.3.8) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
X-Sieve: CMU Sieve 2.3
Received: from murder ([unix socket])
        by mail.umich.edu (Cyrus v2.2.12) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$
```

Como se puede comprobar, el fichero obtenido contiene exactamente el mismo contenido que el fichero original, por lo tanto, podemos concluir que la prueba ha sido exitosa.

1.1. A continuación, utilizamos los algoritmos **DES** y **CAMELLIA** junto con el modo de cifrado **ECB**.

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl des-ecb -salt -in fichero1.txt -out fich1.encode.des
enter des-ecb encryption password:
Verifying - enter des-ecb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl des-ecb -d -in fich1.encode.des -out fich1.dec.des
enter des-ecb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cat fich1.dec.des
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Received: from murder (mail.umich.edu [141.211.14.90])
        by frankenstein.mail.umich.edu (Cyrus v2.3.8) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
X-Sieve: CMU Sieve 2.3
Received: from murder ([unix socket])
        by mail.umich.edu (Cyrus v2.2.12) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
```

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl camellia-128-ecb -salt -in fichero1.txt -out fich1.encode.camellia
enter camellia-128-ecb encryption password:
Verifying - enter camellia-128-ecb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl camellia-128-ecb -d -in fich1.encode.camellia -out fich1.dec.camellia
enter camellia-128-ecb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cat fich1.dec.camellia
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Received: from murder (mail.umich.edu [141.211.14.90])
        by frankenstein.mail.umich.edu (Cyrus v2.3.8) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
X-Sieve: CMU Sieve 2.3
Received: from murder ([unix socket])
        by mail.umich.edu (Cyrus v2.2.12) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
```

2. Utilizamos ahora los ficheros binarios. Empleamos de nuevo el algoritmo **AES**, pero ahora utilizamos el modo de cifrado **CFB**, el cual para producir el keystream cifra el último bloque de cifrado. En CFB, el cifrado no puede ser paralelizado, pero el descifrado sí.

Como clave privada utilizamos: *password*.

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl aes-128-cfb -salt -in fichero2.bin -out fich2.encode.aes
enter aes-128-cfb encryption password:
Verifying - enter aes-128-cfb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl aes-128-cfb -d -in fich2.encode.aes -out fich2.dec.aes
enter aes-128-cfb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ █
```

Al tratarse de ficheros extensos, empleo la siguiente sentencia para verificar que el fichero descifrado es exactamente igual que el fichero original.

GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN

Práctica 4. Uso de OpenSSL para Cifrado de Mensajes y Ficheros

Jhon Steeven Cabanilla Alvarado

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cmp --silent fichero2.bin fich2.dec.aes && echo 'Ficheros iguales!' || echo "Ficheros distintos!"
Ficheros iguales!
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$
```

2.1. A continuación, utilizamos los algoritmos **DES** y **CAMELLIA** junto con el modo de cifrado **CFB**.

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl des-cfb -salt -in fichero2.bin -out fich2.encode.des
enter des-cfb encryption password:
Verifying - enter des-cfb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl des-cfb -d -in fich2.encode.des -out fich2.dec.des
enter des-cfb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cmp --silent fichero2.bin fich2.dec.des && echo 'Ficheros iguales!' || echo "Ficheros distintos!"
Ficheros iguales!
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$
```

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl camellia-128-cfb -salt -in fichero2.bin -out fich2.encode.camellia
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.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl camellia-128-cfb -d -in fich2.encode.camellia -out fich2.dec.camellia
enter camellia-128-cfb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cmp --silent fichero2.bin fich2.dec.camellia && echo 'Ficheros iguales!' || echo "Ficheros distintos!"
Ficheros iguales!
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$
```

3. Por último, vamos a utilizar el modo de cifrado **CBC**. Este modo de cifrado es una extensión de ECB que añade cierta seguridad. Este modo divide el mensaje en bloques y usa XOR para combinar el cifrado del bloque anterior con el texto plano del bloque actual.

Comenzamos utilizando el algoritmo AES junto con el fichero de texto.

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl aes-128-cbc -salt -in fichero1.txt -out fich1.encode.aes
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl aes-128-cbc -d -in fich1.encode.aes -out fich1.dec.aes
enter aes-128-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cat fich1.dec.aes
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Received: from murder (mail.umich.edu [141.211.14.90])
        by frankenstein.mail.umich.edu (Cyrus v2.3.8) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
X-Sieve: CMU Sieve 2.3
Received: from murder ([unix socket])
        by mail.umich.edu (Cyrus v2.2.12) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$
```

3.1 Utilizamos de nuevo los algoritmos DES y CAMELLIA junto con CBC.

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl des-cbc -salt -in fichero1.txt -out fich1.encode.des
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl des-cbc -d -in fich1.encode.des -out fich1.dec.des
enter des-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cat fich1.dec.des
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Received: from murder (mail.umich.edu [141.211.14.90])
        by frankenstein.mail.umich.edu (Cyrus v2.3.8) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
X-Sieve: CMU Sieve 2.3
Received: from murder ([unix socket])
        by mail.umich.edu (Cyrus v2.2.12) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
```

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl camellia-128-cbc -salt -in fichero1.txt -out fich1.encode.camellia
enter camellia-128-cbc encryption password:
Verifying - enter camellia-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ openssl camellia-128-cbc -d -in fich1.encode.camellia -out fich1.dec.camellia
enter camellia-128-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ cat fich1.dec.camellia
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Received: from murder (mail.umich.edu [141.211.14.90])
        by frankenstein.mail.umich.edu (Cyrus v2.3.8) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
X-Sieve: CMU Sieve 2.3
Received: from murder ([unix socket])
        by mail.umich.edu (Cyrus v2.2.12) with LMTPA;
        Sat, 05 Jan 2008 09:14:16 -0500
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$
```

3. Obtenga información sobre el concepto de entropía de Shannon y elabore un breve informe y discusión sobre el tema. Usando el programa Python (enlaces interesantes), obtenga la entropía de los ficheros usados en el apartado segundo (tanto cifrados como descifrados), así como la del fichero obtenido en el apartado cuarto y la de un fichero que contenga un único byte repetido un número de veces arbitrario.

Entropía de Shannon

La entropía de Shannon es método de la Teoría de Información que, dada una variable aleatoria e histórica sobre la ocurrencia de esta variable, puede cuantificar el nivel promedio de información/incertidumbre/complejidad.

Uno de los ingredientes centrales cuando observamos la información es la probabilidad de que aparezca un componente del mensaje. Si un componente de un mensaje tiene una probabilidad muy alta de aparecer en una secuencia, no es de extrañar que aparezca. Sin embargo, si la probabilidad de que aparezca es más cercana a 0, será más *sorprendente*.

Por lo tanto, existe una relación inversa entre la información y la probabilidad de ocurrencia.

Fórmula de la Entropía de Shannon

$$H(X) = - \sum_{i=1}^n p_i \log_2 p_i$$

Para calcular la entropía de Shannon, primero debemos obtener toda la probabilidad de su variable aleatoria X. Recordemos que dicha entropía funciona en **secuencias**, por lo tanto, necesita tener un historial ya construido de cómo se ve su secuencia de interés.

```
# imports
import math
import sys

def shannon_entropy(sequence):

    freqList = [0] * 256
    for b in sequence:
        freqList[b] += 1

    shannon_entr_value = 0.0

    for f in freqList:
        if f > 0:
            freq = float(f) / len(sequence)
            shannon_entr_value += freq * math.log(freq,2)

    return shannon_entr_value * -1

def main():
    # Read arguments
    name = sys.argv[1]
    fich = open(name, "rb")
    byteArr = bytearray(fich.read())
    fich.close()

    entropy = shannon_entropy(byteArr)
    print(entropy)

if __name__ == '__main__':
    main()
```

ENTROPÍAS OBTENIDAS:

- ECB

```
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ python3 shannon_entropy.py fich1_ecb.encode.aes
7.49469769788467
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ python3 shannon_entropy.py fich1_ecb.encode.des
7.466147104270279
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ python3 shannon_entropy.py fich1_ecb.encode.camellia
7.512990102871292
jhon@LAPTOP-91HTN3Q5:~/GSI/Prac_4$ python3 shannon_entropy.py fich1_ecb.des.aes
5.309194959952497
```


4. Obtenga de la red un fichero que contenga una imagen en formato BMP (libre de derechos, a poder ser) y cifrelo usando AES con modos ECB y CBC. Salve copias de la imagen cifrada, sustituya por la cabecera (54 bytes) del fichero original las cabeceras de los ficheros obtenidos y cárguelos en un programa de visualización de imágenes. Comente el resultado.

Nota: Para trasladar la cabecera de un fichero in.bmp a otro out.bmp dejando el resto inalterado, se puede usar: 'dd if=in.bmp of=out.bmp bs=54 count=1 conv=notrunc'

Como imagen original con formato BMP utilizamos la siguiente:



Ciframos la imagen con el algoritmo AES con los modos ECB y CBC tal y como se indica:

```
jhon@jhon-HP-Pavillon-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI$ openssl aes-128-ecb -in lena.bmp -out img_aes_ecb.bmp
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@jhon-HP-Pavillon-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI$ openssl aes-128-cbc -in lena.bmp -out img_aes_cbc.bmp
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
jhon@jhon-HP-Pavillon-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI$
```

Salvamos copias de las imágenes cifradas:

```
jhon@jhon-HP-Pavillon-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI$ cp img_aes_ecb.bmp copy_img_aes_ecb.bmp
jhon@jhon-HP-Pavillon-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI$ cp img_aes_cbc.bmp copy_img_aes_cbc.bmp
jhon@jhon-HP-Pavillon-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI$ ls
copy_img_aes_cbc.bmp  copy_img_aes_ecb.bmp  img_aes_cbc.bmp  img_aes_ecb.bmp  lena.bmp
jhon@jhon-HP-Pavillon-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI$
```

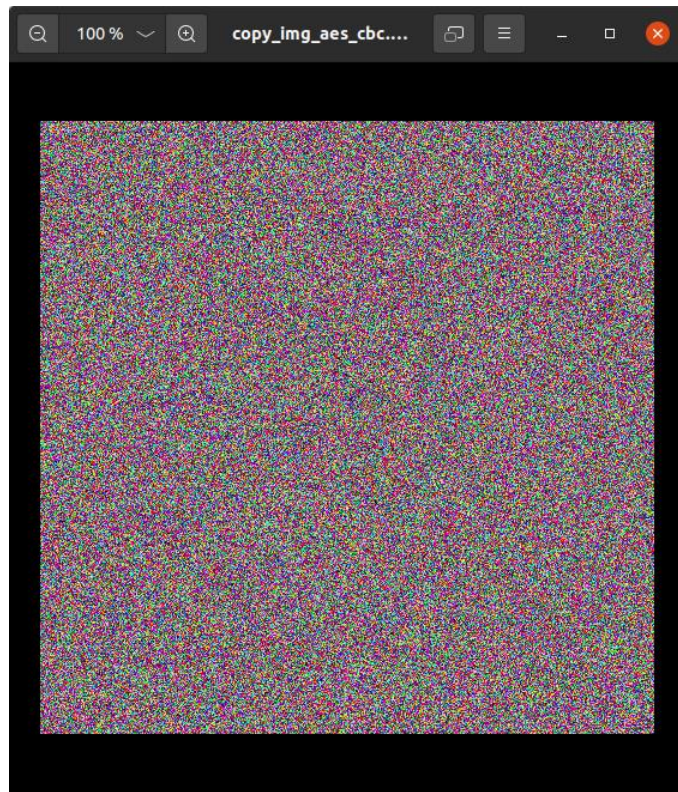
Por último, sustituimos la cabecera del fichero original por las cabeceras de los ficheros obtenidos. Para ello utilizamos el comando que se nos proporciona:

```
jhon@jhon-HP-Pavillon-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI$ dd if=lena.bmp of=copy_img_aes_ecb.bmp bs=54 count=1 conv=notrunc
1+0 registros leídos
1+0 registros escritos
54 bytes copied, 0,000503689 s, 107 kB/s
jhon@jhon-HP-Pavillon-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI$ dd if=lena.bmp of=copy_img_aes_cbc.bmp bs=54 count=1 conv=notrunc
1+0 registros leídos
1+0 registros escritos
54 bytes copied, 0,000576426 s, 93,7 kB/s
```

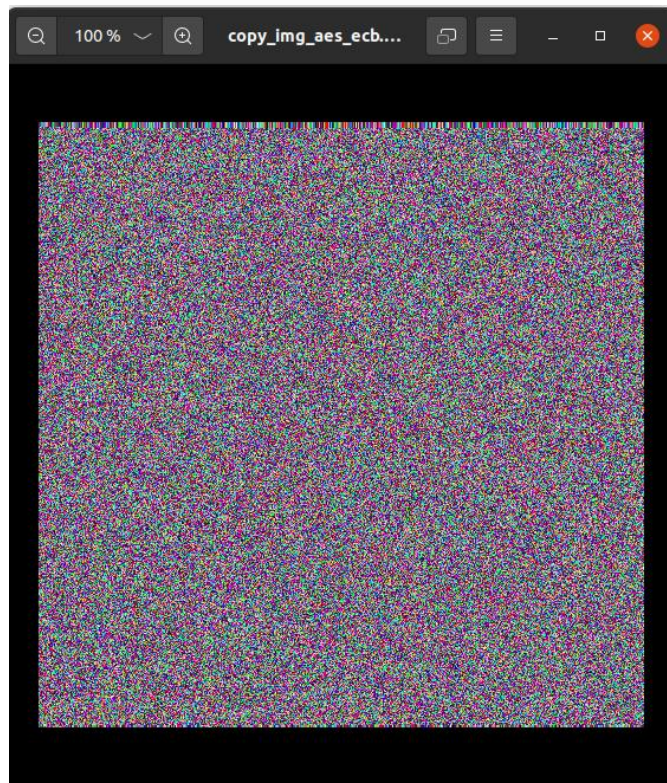
GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN
Práctica 4. Uso de OpenSSL para Cifrado de Mensajes y Ficheros
Jhon Steeven Cabanilla Alvarado

- Resultados obtenidos

CBC



ECB



PARTE 2

1. Selecciona un servidor web público (puedes utilizar el de la UVa) e investiga cuáles son los puertos a nivel de transporte que utiliza para brindar el servicio web. ¿Qué sentido tiene?

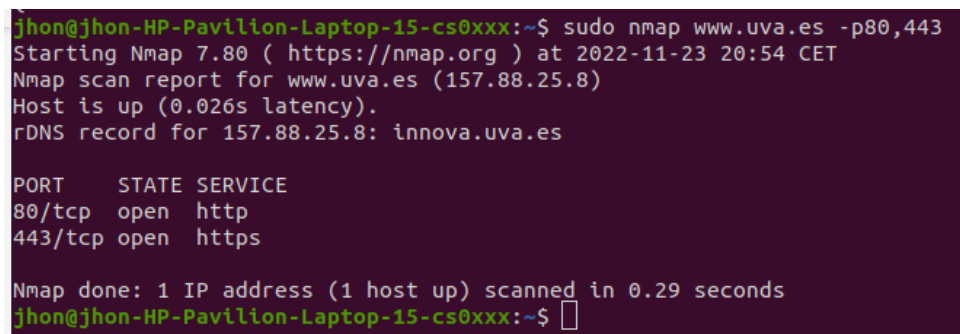
Como servidor web público seleccionamos el de la Uva. Los puertos a nivel de transporte utilizados para brindar el servicio web son: Puerto 80 y Puerto 443 para cifrar información.

- El puerto 80/tcp es el puerto que se utiliza para la navegación web de **forma no segura** HTTP.

- El puerto 443/tcp es también un puerto de navegación web, pero en este caso usa el protocolo HTTPS que es **seguro** y utiliza el protocolo TLS por debajo.

Mediante la herramienta *nmap* podemos comprobar si estos 2 puertos se encuentran abiertos utilizando el siguiente comando:

sudo nmap www.uva.es -p80, 443



```
jhon@jhon-HP-Pavilion-Laptop-15-cs0xxx:~$ sudo nmap www.uva.es -p80,443
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-23 20:54 CET
Nmap scan report for www.uva.es (157.88.25.8)
Host is up (0.026s latency).
rDNS record for 157.88.25.8: innova.uva.es

PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
jhon@jhon-HP-Pavilion-Laptop-15-cs0xxx:~$
```

Si en vez de conectarnos al servicio web de forma segura, nos conectamos utilizando http, podemos comprobar cómo de manera automática el servidor web nos redirige al puerto 443/tcp. El código 302 es el código de respuesta http cuando se produce una redirección.

De esta manera, la comunicación entre el cliente y el servidor irá cifrada.

2. Analiza el certificado digital presente en el servidor web. Puedes hacer uso de la herramienta *sslsca* y *sslttest* del apartado anterior e indica:

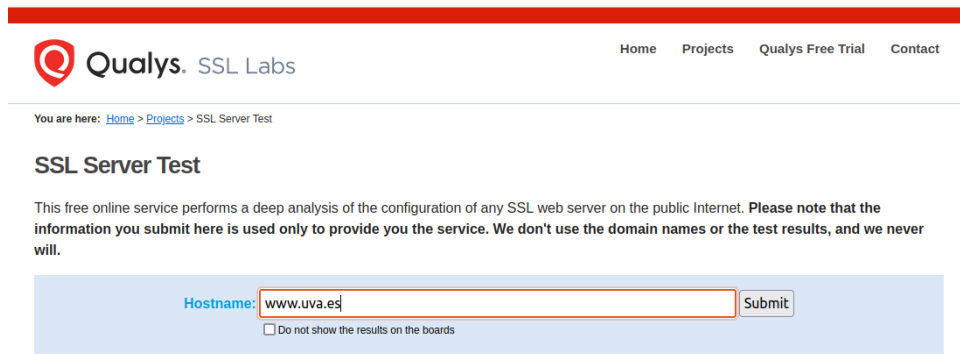
a) Protocolo/s criptográficos y versión utilizado a nivel de transporte.

Comenzamos introduciendo el nombre del servidor web: www.uva.es.

GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN

Práctica 4. Uso de OpenSSL para Cifrado de Mensajes y Ficheros

Jhon Steeven Cabanilla Alvarado



Qualys. SSL Labs

Home Projects Qualys Free Trial Contact

You are here: [Home](#) > [Projects](#) > SSL Server Test

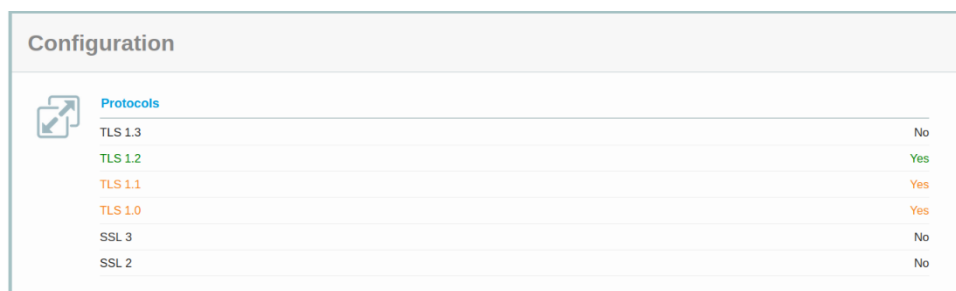
SSL Server Test


This free online service performs a deep analysis of the configuration of any SSL web server on the public Internet. Please note that the information you submit here is used only to provide you the service. We don't use the domain names or the test results, and we never will.

Hostname:

☐ Do not show the results on the boards

Una vez finalice el proceso, obtenemos una ventana con los resultados. Nos dirigimos a la parte de configuración y obtenemos los protocolos:

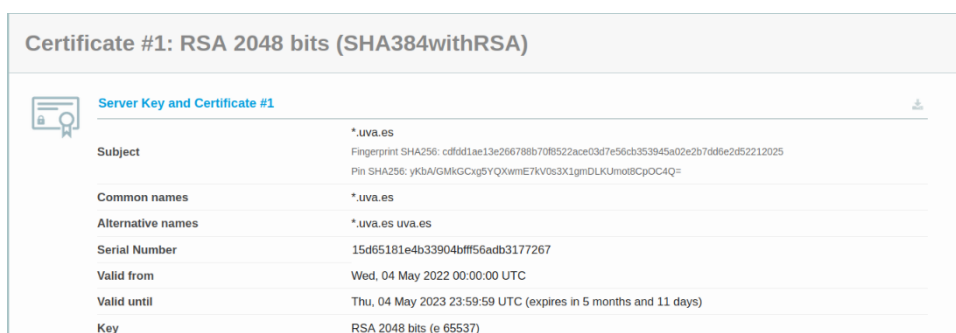



Configuration	
 Protocols	
TLS 1.3	No
TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	No
SSL 2	No

Como podemos observar, se utiliza el protocolo **TLS en la versión 1.2**.

b) Algoritmo de criptografía asimétrica (clave pública) utilizado y longitud de la clave pública.

Si nos dirigimos al inicio del certificado, podemos observar que el algoritmo de criptografía asimétrica que se utiliza es el RSA y la longitud de la clave pública es de 2048 bits.



Certificate #1: RSA 2048 bits (SHA384withRSA)	
 Server Key and Certificate #1	
Subject	*.uva.es Fingerprint SHA256: cdfdd1ae13e266788b70f8522ace03d7e56cb353945a02e2b7dd6e2d52212025 Pin SHA256: ykBA/GMkGCGxg5YQXwmE7KvOs3X1gmDLUkumot8CpOC4Q=
Common names	*.uva.es
Alternative names	*.uva.es uva.es
Serial Number	15d65181e4b33904bfff56adb3177267
Valid from	Wed, 04 May 2022 00:00:00 UTC
Valid until	Thu, 04 May 2023 23:59:59 UTC (expires in 5 months and 11 days)
Key	RSA 2048 bits (e 65537)

c) Indica la clave pública presente en el certificado digital. ¿Por qué se utiliza esa y no otra?

Si nos fijamos en el apartado *key* de la imagen anterior, podemos ver la clave pública ($e = 65537$). Este valor es el mayor número primo conocido de la forma $2^{2^n} + 1$ para $n=4$. En la teoría de números, los primos de esta forma son conocidos como **número primos de Fermat**.

d) Algoritmo de criptografía simétrica (clave privada) utilizado y longitud de la clave privada.

De nuevo en la parte de configuración, en el apartado *Cipher Suites* se muestran los algoritmos de criptografía simétrica de clave privada.

Cipher Suites				
# TLS 1.2 (suites in server-preferred order)				
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)	ECDH secp384r1 (eq. 7680 bits RSA)	FS		256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	ECDH secp384r1 (eq. 7680 bits RSA)	FS		128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)	ECDH secp384r1 (eq. 7680 bits RSA)	FS	WEAK	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	ECDH secp384r1 (eq. 7680 bits RSA)	FS	WEAK	256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	ECDH secp384r1 (eq. 7680 bits RSA)	FS	WEAK	128
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f)	DH 1024 bits	FS	WEAK	256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e)	DH 1024 bits	FS	WEAK	128

Como se puede ver en la imagen, el primer algoritmo que aparece es:

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

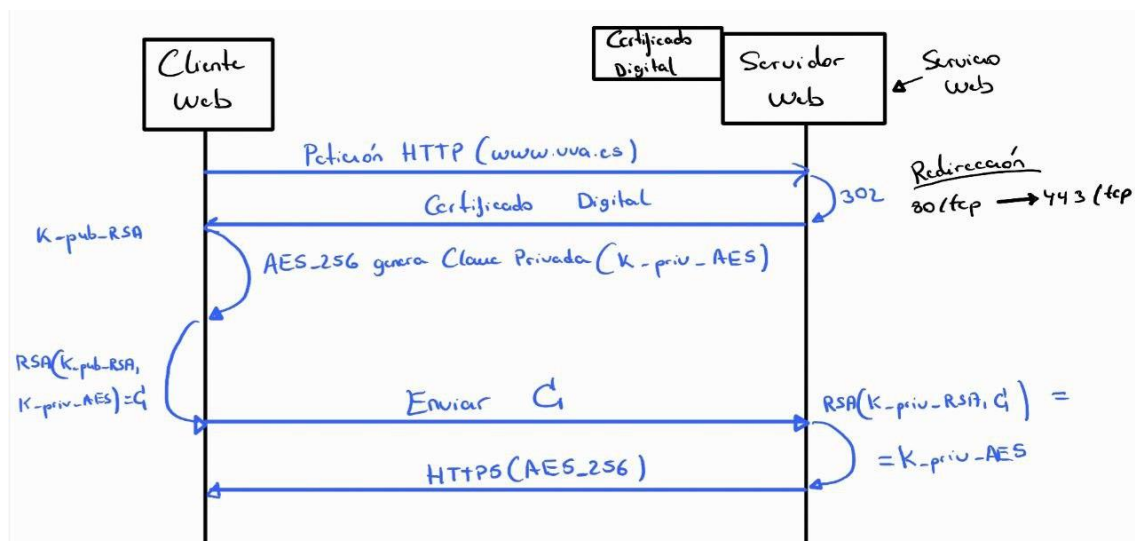
el cual se corresponde con el algoritmo **ECDH** y en este caso, la longitud de la clave privada es de **256 bits**.

e) Algoritmo de firma digital utilizado en el certificado digital.

Signature algorithm	SHA384withRSA
---------------------	---------------

Como podemos observar en la imagen, el algoritmo de firma digital que se utiliza en el certificado digital es **SHA-384 con RSA**.

3. Explica con un diagrama de secuencia por qué en un certificado digital se usa criptografía de clave pública y privada.



Comenzamos realizando una petición HTTP y se produce una redirección de puertos. Seguidamente, el servidor devuelve la solicitud del certificado digital, dentro del cual se encuentra la clave pública de RSA.

El cliente crea su clave privada con el algoritmo de cifrado AES-256 y con RSA obtiene el mensaje cifrado, a partir de la clave privada del cliente y la clave pública de RSA. Se realiza el envío del criptograma al servidor, el cual conoce la clave privada del cliente y con ello, accede al mensaje original.

El certificado digital se utiliza en criptografía de clave pública y privada para evitar que otros usuarios no deseados interfieran en la comunicación cliente-servidor.

4. Utiliza la herramienta OpenSSL para la generación de un certificado digital que tenga una longitud de clave de 1024 bits y analízalo con sslscan y ssltest.

- Comenzamos creando un directorio en la carpeta /opt para realizar este ejercicio. Procedemos a abrir dicha carpeta para generar en ella nuestra clave privada.

Para ello, utilizamos el siguiente comando: *sudo openssl genrsa -out alice.key 1024*, donde cada argumento significa lo siguiente:

- *genrsa*: este comando genera una clave privada RSA.
- *-out alice.key 1024*: genera la clave privada de 1024 bits en el archivo de salida *alice.key*.

```
jhon@LAPTOP-91HTN3Q5:~$ sudo mkdir /opt/rsa
[sudo] password for jhon:
jhon@LAPTOP-91HTN3Q5:~$ cd /opt/rsa/
jhon@LAPTOP-91HTN3Q5:/opt/rsa$ sudo openssl genrsa -out alice.key 1024
Generating RSA private key, 1024 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
jhon@LAPTOP-91HTN3Q5:/opt/rsa$ S
```

- A continuación, vamos a observar el contenido del fichero creado. Para ello, utilizamos el siguiente comando: *sudo openssl rsa -text -in alice.key*.

GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN
Práctica 4. Uso de OpenSSL para Cifrado de Mensajes y Ficheros
Jhon Steeven Cabanilla Alvarado

```
jhon@LAPTOP-91HTN3Q5:/opt/rsa$ sudo openssl rsa -text -in alice.key
RSA Private-Key: (1024 bit, 2 primes)
modulus:
 00:d3:fc:87:2f:00:82:b4:58:87:70:28:36:c1:ee:
 fc:02:82:b3:b6:20:9c:b2:2f:52:67:fa:9c:1e:1d:
 e4:41:9a:d8:cd:16:9a:5b:03:f6:a1:fe:e3:bd:58:
 db:04:f3:80:94:87:e2:3d:16:6e:06:d6:dd:dc:a1:
 fa:6e:fe:7b:59:0e:d8:c5:a0:e4:d3:39:9a:f1:c3:
 24:9e:8b:62:a8:51:1e:06:fe:82:da:18:0e:25:d8:
 a9:cd:7c:31:35:23:f5:2c:7d:51:8b:fe:23:0e:ef:
 04:27:26:c7:03:4d:f9:4e:e6:95:bc:0e:5a:dc:34:
 dc:1c:fc:12:d4:db:92:00:3b
publicExponent: 65537 (0x10001)
privateExponent:
 7e:5e:9f:c7:a9:2a:41:90:34:cc:eb:6c:19:17:fa:
 2b:14:f7:44:af:14:3a:34:73:8f:80:99:e8:6a:31:
 53:da:97:3b:4b:2a:20:ef:af:ea:ca:23:c1:10:63:
 20:04:78:b6:12:7b:0f:a7:7b:57:a0:67:09:cb:95:
 9a:ed:21:61:d3:72:f1:d0:e9:12:85:63:1f:b3:d9:
 cd:bc:2d:a2:0b:d5:de:24:92:43:cb:e7:6f:76:e4:
 a3:b1:71:ca:b6:97:8d:f8:2a:14:e9:2f:a6:fc:6b:
 36:9f:fc:6e:5a:93:5c:7b:19:9a:4a:60:86:cd:8f:
 10:f6:45:2c:27:7d:20:61
prime1:
 00:f7:7f:fa:42:03:33:5d:ec:9a:7a:ca:30:ed:c6:
 07:38:e7:50:70:17:bc:f7:e3:f9:8e:e0:21:72:db:
 98:7e:f7:70:ea:fd:dc:5b:ff:80:0e:9e:81:5f:af:
 b5:25:c8:22:d5:c0:00:6f:ba:df:9d:48:4f:56:e1:
 d1:c2:dd:78:3d
prime2:
 00:db:44:50:c8:ba:3b:c4:ff:7b:6f:15:31:ff:e1:
 42:b3:40:29:86:d7:41:ba:50:a0:d4:fd:ff:73:95:
 39:68:82:17:d5:ca:fd:7b:e4:ce:fc:cb:88:ec:c9:
 c8:ef:39:97:8a:2d:44:7d:f9:62:39:3b:96:76:85:
 bc:45:55:69:d7
exponent1:
 4e:f1:9b:39:83:1b:d7:51:a8:d4:91:b9:99:9e:18:
 9f:a5:e8:9a:58:78:05:f9:c9:98:31:15:7a:35:61:
 26:de:76:3d:fe:4b:53:6c:e3:c3:c5:fb:2e:32:35:
 6d:9f:b0:bc:cd:49:56:5a:1d:09:66:0f:28:7f:4f:
 11:00:e3:61
exponent2:
 06:94:bb:53:fa:fa:f6:43:b0:c1:b9:c2:21:6a:f2:
 64:8c:e5:72:2d:9d:c4:68:cd:1a:f7:70:a6:58:71:
 3b:a8:c3:ec:5f:c6:51:e4:a1:2d:c7:32:19:e1:48:
 8d:9d:8d:e5:d0:cc:00:77:ee:b1:c7:0f:12:09:9b:
 87:fd:33:4b
```

- Realizamos una petición de certificado digital mediante el siguiente comando: `openssl req -new -key alice.key -out alice.csr`.

- req: crea y procesa principalmente solicitudes de certificados en formato PKCS#10.
- -new: esta opción genera una nueva solicitud de certificado, pidiendo al usuario los valores de los campos pertinentes.
- -key alice.key: especifica el archivo del que se va a leer la clave privada.
- -out alice.csr: especifica dónde guardar el archivo csr. Un CSR es un archivo codificado que contiene información que la *Certificate Authority (CA)* utilizará para crear el certificado. También contiene la clave pública que se incluirá en el certificado y está firmada con la correspondiente clave privada.

GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN
Práctica 4. Uso de OpenSSL para Cifrado de Mensajes y Ficheros
Jhon Steeven Cabanilla Alvarado

```
jhon@LAPTOP-91HTN3Q5:/opt/rsa$ sudo openssl req -new -key alice.key -out alice.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]:ES
State or Province Name (full name) [Some-State]:Valladolid
Locality Name (eg, city) []:Valladolid
Organization Name (eg, company) [Internet Widgits Pty Ltd]:4ck.com
Organizational Unit Name (eg, section) []:Uva
Common Name (e.g. server FQDN or YOUR name) []:www.4ck.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
jhon@LAPTOP-91HTN3Q5:/opt/rsa$
```

Como se puede ver en la imagen, hay que ir rellenando los campos que se van solicitando.

- A continuación, procedemos a mostrar el contenido del certificado de solicitud de firma generado, utilizando el siguiente comando: `sudo openssl req -noout -text -in alice.csr`.

```
jhon@LAPTOP-91HTN3Q5:/opt/rsa$ sudo openssl req -noout -text -in alice.csr
Certificate Request:
Data:
  Version: 1 (0x0)
  Subject: C = ES, ST = Valladolid, L = Valladolid, O = 4ck.com, OU = Uva, CN = www.4ck.com
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
      RSA Public-Key: (1024 bit)
      Modulus:
        00:d3:fc:87:2f:00:82:b4:58:87:70:28:36:c1:ee:
        fc:02:82:b3:b6:20:9c:b2:2f:52:67:fa:9c:1e:1d:
        e4:41:9a:d8:cd:16:9a:5b:03:f6:a1:fe:e3:bd:58:
        db:04:f3:80:94:87:e2:3d:16:6e:06:d6:dd:dc:a1:
        fa:6e:fe:7b:59:0e:d8:c5:a0:e4:d3:39:9a:f1:c3:
        24:9e:8b:62:a8:51:1e:06:fe:82:da:18:0e:25:d8:
        a9:cd:7c:31:35:23:f5:2c:7d:51:8b:fe:23:0e:ef:
        04:27:26:c7:03:4d:f9:4e:e6:95:bc:0e:5a:dc:34:
        dc:1c:fc:12:d4:db:92:00:3b
      Exponent: 65537 (0x10001)
  Attributes:
    a0:00
  Signature Algorithm: sha256WithRSAEncryption
    19:89:e0:02:dc:85:9d:58:fd:c0:8e:21:25:f3:5f:cf:ff:17:
    95:d5:67:93:0c:dd:4e:9f:b9:21:17:a8:64:f0:39:e2:89:ce:
    8e:44:ec:96:39:f3:64:6f:93:a2:36:b4:49:78:c4:ed:61:85:
    c7:09:06:b1:f5:ac:60:7d:e5:01:1e:c2:69:1e:74:91:0f:6a:
    23:e5:50:e5:da:2b:a4:e8:8a:52:ad:44:63:ba:74:e2:10:79:
    2f:e5:e3:75:95:77:2f:1b:9e:13:dd:cc:c2:66:a3:01:c9:f0:
    38:86:c3:45:2f:f0:ab:9f:82:30:33:4a:b4:16:57:05:73:9d:
    01:5b
jhon@LAPTOP-91HTN3Q5:/opt/rsa$
```


GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN
Práctica 4. Uso de OpenSSL para Cifrado de Mensajes y Ficheros
Jhon Steeven Cabanilla Alvarado

- Una vez hemos que hayamos creado la clave (alice.key) y la solicitud de certificado (alice.csr), podemos crear un certificado digital autofirmado compatible con el estándar X.509 empleando el siguiente comando:

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

```
jhon@LAPTOP-91HTN3Q5:/opt/rsa$ sudo openssl x509 -req -days 365 -in alice.csr -signkey alice.key -out al
ice.crt
Signature ok
subject=C = ES, ST = Valladolid, L = Valladolid, O = 4ck.com, OU = Uva, CN = www.4ck.com
Getting Private key
jhon@LAPTOP-91HTN3Q5:/opt/rsa$
```

- Disposición del público.

Para que no todo el público pueda acceder a la clave y certificado, movemos dichos ficheros a sus correspondientes directorios.

- o Llevamos el certificado al siguiente directorio:
 - sudo cp alice.crt /etc/ssl/certs/
- o Llevamos la clave privada al siguiente directorio:
 - sudo cp alice.key /etc/ssl/private/

Finalmente, con el comando `sudo ls -l /etc/ssl/`, podemos comprobar los permisos de los directorios que se encuentran dentro de /etc/ssl.

```
jhon@LAPTOP-91HTN3Q5:/opt/rsa$ sudo ls -l /etc/ssl/
total 12
drwxr-xr-x 1 root root 4096 Aug 29 23:39 certs
-rw-r--r-- 1 root root 10909 Apr 20 2020 openssl.cnf
drwx----- 1 root root 4096 Apr 20 2020 private
jhon@LAPTOP-91HTN3Q5:/opt/rsa$
```

- Verificado de certificado digital

Como se nos comentó en clase, la forma de poder analizar el certificado con `sslsca` o `ssltest` sería creando un servidor web e introduciendo dicho certificado en él. Esto se debe a que `sslsca` y `ssltest` sólo permiten analizar servidores web.

Por este motivo, procedemos a utilizar la propia herramienta de `openSSL` para analizar el certificado creado. Para ello utilizamos el siguiente comando:

```
openssl x509 -in alice.crt -text -noout
```

GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN
Práctica 4. Uso de OpenSSL para Cifrado de Mensajes y Ficheros
Jhon Steeven Cabanilla Alvarado

```
jhon@LAPTOP-91HTN3Q5:/opt/rsa$ openssl x509 -in alice.crt -text -noout
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number:
      06:47:5f:e7:d9:4e:be:0a:64:fd:05:70:b5:37:07:73:b9:1d:a5:36
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = ES, ST = Valladolid, L = Valladolid, O = 4ck.com, OU = Uva, CN = www.4ck.com
    Validity
      Not Before: Nov 10 12:21:47 2022 GMT
      Not After : Nov 10 12:21:47 2023 GMT
    Subject: C = ES, ST = Valladolid, L = Valladolid, O = 4ck.com, OU = Uva, CN = www.4ck.com
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (1024 bit)
      Modulus:
        00:d3:fc:87:2f:00:82:b4:58:87:70:28:36:c1:ee:
        fc:02:82:b3:b6:20:9c:b2:2f:52:67:fa:9c:1e:1d:
        e4:41:9a:d8:cd:16:9a:5b:03:f6:a1:fe:e3:bd:58:
        db:04:f3:80:94:87:e2:3d:16:6e:06:d6:dd:dc:a1:
        fa:6e:fe:7b:59:0e:d8:c5:a0:e4:d3:39:9a:f1:c3:
        24:9e:8b:62:a8:51:1e:06:fe:82:da:18:0e:25:d8:
        a9:cd:7c:31:35:23:f5:2c:7d:51:8b:fe:23:0e:ef:
        04:27:26:c7:03:4d:f9:4e:e6:95:bc:0e:5a:dc:34:
        dc:1c:fc:12:d4:db:92:00:3b
      Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
      5e:2a:29:01:d6:28:1c:bf:9a:11:d3:3f:3e:ca:58:de:13:ec:
      ef:de:bd:c2:a3:29:ce:e5:84:76:44:c4:9a:59:66:52:2a:47:
      18:d4:20:17:fd:4f:41:26:79:f2:f1:70:0e:2f:a6:36:58:56:
      9b:01:19:82:50:53:f2:19:aa:ed:c7:fd:c1:9d:82:96:cd:74:
      aa:c2:bd:ce:15:de:c3:92:5b:6c:7b:43:5f:21:c4:b5:8b:36:
      7b:2c:45:5f:d1:90:61:7b:57:45:3d:ed:ad:c8:eb:f8:64:22:
      9b:ef:2b:fa:e2:93:9a:3d:2c:f4:45:17:7a:b8:91:81:05:bf:
      79:8c
jhon@LAPTOP-91HTN3Q5:/opt/rsa$
```