**Práctica 4 – Uso de Openssl para cifrado de Mensajes y Ficheros**

***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.**

Antes de comenzar con el manual voy a comentar que es openssl. Es una herramienta que contiene un conjunto de funciones que son de gran utilidad para la criptografía aplicada, además OpenSSL implementa los protocolos más conocidos y utilizados en el mundo de la computación, como SSL y TLS.

Para usarlo voy a utilizar una de máquinas virtuales proporcionadas como por ejemplo Mallet, esta es una máquina Linux, y por tanto dispone de la herramienta openssl. Esto lo sabemos ya que ponemos *“openssl”* en la línea de comandos y se modifica el promt a OpenSSL > . Ahora ejecutamos enc -help, y nos va a devolver como se ve en la captura de la parte inferior, una serie de opciones, así como tipos de cifrado. Viendo las opciones, así como esta página web que muestra ejemplos de cifrado (<https://www.openssl.org/docs/man1.1.1/man1/enc.html>) los comandos para cifrar con openssl serían: “openssl *algoritmoCifrado -in ficheroACifrar -out ficheroCifrado”* para codificar y *“openssl algoritmoCifrado -d -in ficheroCifrado -out ficheroACifrar”.*

**Texto

Descripción generada automáticamente**

**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.**

Para ello creo un archivo llamado textoClaro.txt con algo escrito, y lo cifro primero con el algoritmo AES 256, y este algoritmo tiene 3 modos diferentes de cifrado por tanto primero utilizando ECB, como clave de encriptación utilizo “123456”:

Texto

Descripción generada automáticamente

Ahora utilizando CFB:

Texto

Descripción generada automáticamente

Y por último utilizo CBC:

Texto

Descripción generada automáticamente

Ahora en vez de utilizar AES 256 cambio el algoritmo de cifrado y utilizo DES con sus 3 modos diferentes de cifrado, y cambio la clave de encriptación a “abcd”, primero empiezo utilizando ECB:

Texto

Descripción generada automáticamente

Ahora utilizando CFB:

Texto

Descripción generada automáticamente

Y por último utilizo CBC:

Texto

Descripción generada automáticamente

Pero antes un pequeño inciso, como se puede ver en la captura del apartado 1 de la práctica en Cipher Types, mallet no tiene CAMELLIA, por tanto, a partir de ahora cambio la máquina en la que voy a continuar con la realización de la práctica, voy a pasar a utilizar el Ubuntu 20.0.4 LTS de la práctica anterior. Como podemos ver ahora, esta nueva máquina sí que dispone de CAMELLIA.

Interfaz de usuario gráfica

Descripción generada automáticamente

Y para acabar con el cifrado de texto claro, el último algoritmo de cifrado que se pide, voy a utilizar CAMELLIA-256 con sus 3 modos diferentes de cifrado, y modifico la clave de cifrado a “a1b2c3”, primero empiezo utilizando ECB:

Texto

Descripción generada automáticamente

Ahora utilizando CFB:

Texto

Descripción generada automáticamente

Y por último utilizo CBC:

Texto

Descripción generada automáticamente

Pero de momento solo hemos cifrado textos en claro, como pide el enunciado también necesitamos archivos binarios. Para ello he buscado en internet y la ejecución de este comando: “xxd -b textoClaro.txt textoBin.bin” te convierte el contenido de textoClaro.txt a binario y lo guarda en textoBin.bin como se puede ver en la siguiente captura.

Texto, Chat o mensaje de texto

Descripción generada automáticamente

Como ahora este nuevo fichero es binario, voy a realizar el cifrado, usando este fichero como fichero de entrada. Al igual que antes primero voy a utilizar el algoritmo AES-256 con sus 3 modos de cifrado ECB, CFB, CBC en este orden:

Texto

Descripción generada automáticamente

Como podemos observar el archivo que contiene la información cifrada es bastante extenso, y para evitar alagar la práctica, ya no voy a mostrar el cifrado únicamente el comando de cifrado que utilizo.

Texto

Descripción generada automáticamente

Ahora que ya he cifrado con AES al igual que antes voy a cambiar el algoritmo de cifrado a DES, y siguiendo el mismo orden con los distintos 3 modos de cifrado primero ECB, CFB y CBC:

Interfaz de usuario gráfica, Texto, Chat o mensaje de texto

Descripción generada automáticamente

Y por último y de la misma manera que en los anteriores, pero modificando el algoritmo de cifrado a CAMELLIA.

Texto

Descripción generada automáticamente

**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.**

Primero voy a comentar que es y en qué consiste la entropía de Shannon. El concepto de entropía o cantidad de información sirve para medir la incertidumbre de una fuente de información.

La entropía de Shannon se podría calcular de esta forma, dada una variable aleatoria X:

Un dibujo de una persona

Descripción generada automáticamente con confianza baja

Como se puede observar en la fórmula la entalpía depende muy directamente de la distribución de probabilidad.

En esta página web (<https://code.activestate.com/recipes/577476-shannon-entropy-calculation/>) he conseguido encontrar un código que calcula la entropía de Shannon de un fichero que recibe como parámetro. Por tanto, creo un archivo Python llamado entropy\_shannon.py donde voy a copiar dicho código de internet, realizando unas pequeñas modificaciones ya que realizando únicamente copia y pega no compila.

Este es el código que lo realiza:

Texto

Descripción generada automáticamente

Y ahora voy a ejecutar este código para todos los archivos que pide el enunciado.

Primero voy a empezar por los ficheros descifrados, es decir, el fichero de texto claro (textoClaro.txt) y también el fichero binario (textoBin.bin).

Texto

Descripción generada automáticamente

Ahora que ya he hallado las entropías de los textos sin descifrar, voy a hallar las entropías de los ficheros cifrados generados a partir del fichero textoClaro.txt que como indica su nombre contiene texto claro:

**CIFRADOS CON AES 256**

**Texto

Descripción generada automáticamente**

**CIFRADOS CON DES**

**Texto

Descripción generada automáticamente**

**CIFRADOS CON CAMELLIA 256**

**Texto

Descripción generada automáticamente**

Hasta ahora he hallado las entropías de los ficheros de texto claro cifrados, pero faltan para los ficheros binarios cifrados.

**CIFRADOS CON AES 256**

**Texto

Descripción generada automáticamente**

**CIFRADOS CON DES**

**Texto

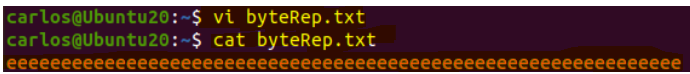
Descripción generada automáticamente**

**CIFRADOS CON CAMELLIA 256**

**Texto

Descripción generada automáticamente**

Puesto que todavía no he comenzado a realizar el apartado 4 primero voy a calcular la entropía de un fichero que contenga un único byte repetido un número de veces arbitrario. El fichero se va a llamar byteRep.txt, como se ve en esta captura.



Ahora como el fichero contiene un único byte que esta repetido muchas veces, el valor de la entropía de Shannon debería de ser muy bajo, ya que puede tomar valores ente 0 y 8. Siendo 0 el mínimo valor posible que puede tomar (cuando el fichero está vacío) y 8 el valor máximo. De nuevo voy a ejecutar el programa para ver qué valor toma la entropía:

Texto

Descripción generada automáticamente

Como hemos supuesto en el párrafo anterior el valor de la entropía es bajo 0.1176.

Ahora voy a calcular la entropía de la imagen que he seleccionado para el apartado 4, así como de ambas imágenes cifradas “copia\_imagen\_ecb.bmp” y “copia\_imagen\_cbc.bmp”. Primero voy a hallar la entropía de la imagen original.

Texto

Descripción generada automáticamente

Y ahora de ambas copias cifradas, primero la hallada con el modo ECB y después con el modo CBC:

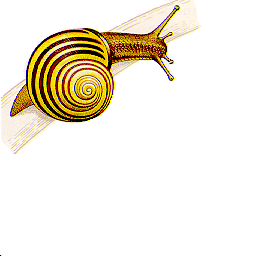
Texto

Descripción generada automáticamente

Como se puede ver, tanto en los textos como en las imágenes cifradas, pero sobretodo en estas últimas, el valor de la entropía es mucho mayor y cercano a 8. Sin embargo, en los fichero o imágenes sin cifrar la entropía toma valores menores, entorno a 5. Esto es debido a que al cifrarlo altera los bytes para que no puedan ser interpretados, y como hemos visto la repetición de los bytes está muy relacionado con el valor que toma la entropía de Shannon.

**4. Obtenga de la red un fichero que contenga una imagen en formato BMP (libre de derechos, a poder ser) y cífrelo 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’**

Para ello he buscado en internet imágenes con formato bmp, (<https://people.math.sc.edu/Burkardt/data/bmp/snail.bmp>) y en esta página he encontrado una serie de imágenes que te permitían descargar formato bmp, gratis. La imagen elegida es esta:

****

Ahora voy cifrar dicha imagen con algunos de los algoritmos de cifrado que hemos utilizado antes, nos pide con AES con los modos ECB y CBC:

**Texto

Descripción generada automáticamente**

Realizo las dos copias de las imágenes cifradas con el comando *“cp ficheroACopiar nombreCopia”.* Y después ejecuto el comando *“ls”* para comprobar que se han creado dichas copias.

Texto

Descripción generada automáticamente

Ahora falta sustituir la cabecera de las copias por la cabecera de la imagen original (la descargada). Utilizando el comando que dice el enunciado del apartado.

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

A la hora de intentar visualizar las imágenes, las imágenes que han sido cifradas tanto con ECB como con CBC, no deja visualizarlas.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Pero las imágenes que son copias de las cifradas cuya cabecera ha sido modificada, sustituyendo la cabecera de la imagen cifrada, por la de la imagen original nos permite visualizarla. Esto es el resultado que obtenemos:

**CIFRADA CON AES, MODO ECB**

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**CIFRADA CON AES, MODO CBC**

**Interfaz de usuario gráfica, Código QR

Descripción generada automáticamente**

Como las imágenes han sido cifradas, para ninguno de los dos métodos se permite distinguir la imagen original, únicamente los píxeles alterados de forma que sea imposible. Aunque en el modo de cifrado ECB, se permite distinguir la forma de la figura del caracol, pero no se llega a distinguir nada, es decir, se sabe que hay algo por la forma pero no el qué.

***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?**

Para ello selecciono el servidor web de la UVa ([www.uva.es](http://www.uva.es)). Los puertos a nivel de transporte que utiliza para brindar el servicio web y por tanto están abiertos, son el puerto 80 y 443 TCP.

El puerto 80 es el que se utiliza para la navegación web usando el protocolo HTTP (HyperText Transfer Protocol) este protocolo no es seguro ya que la conexión no está protegida de posibles ataques y escuchas.

El puerto 443 es el que se utiliza para la navegación web utilizando el protocolo HTTPS (HyperText Transfer Protocol Secure), este protoco sí que es seguro ya que implementa el protocolo de seguridad TLS (Transport Layer Security). Además aporta al cliente un certificado digital que asegura que el sitio web en concreto posee confidencialidad, integridad, autenticidad, disponibilidad y no repudio.

Por tanto, con la herramienta nmap podemos ver si estos dos puertos de nivel de tansporte se encuentran abiertos. Como podemos ver en la siguiente captura ejecutando el comando *“nmap* [*www.uva.es*](http://www.uva.es) *-p80”* y *“nmap* [*www.uva.es*](http://www.uva.es) *-p443”*, ambos puertos para dicho servidor web, están abiertos.

Texto

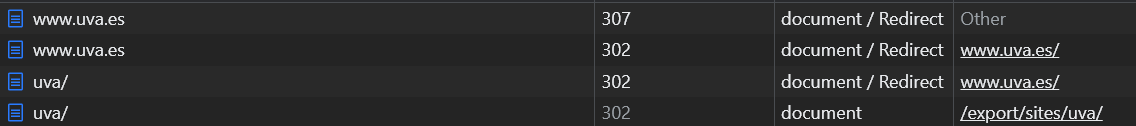
Descripción generada automáticamente

Cuando nosotros realizamos una búsqueda a una página web que utiliza el puerto 443, es decir conexión segura, podemos ver que nos aparece un candado en la parte superior izquierda de la barra de búsqueda.



Ahora suponga que en vez de conectarnos a dicho servicio de manera segura (<https://www.uva.es>). Nos conectamos poniendo (http://www.uva.es).

De forma automática el servidor web nos redirige del puerto 80 TCP al 443 TCP de esta forma pasamos de usar HTTP a usar HTTPS, es decir pasamos a un modo seguro. Esta redirección se puede observar en el apartado Network del menú de desarrollador.





El código de estado 302 que se ve en la captura, es el código de respuesta que http cuando se produce una redirección. En este caso del puerto 80 al 443, o lo que es lo mismo de utilizar el protocolo HTTP a HTTPS.

Respecto a la pregunta ¿Qué sentido tiene?, nos permite que la comunicación entre el cliente y el servidor web vaya cifrada. Ya que si únicamente tuviese abierto el puerto 80 TCP las conexiones se realizarían sin seguridad y serian mucho más vulnerables a posibles ataques. Esto es más difícil que ocurra si la conexión se efectúa a través del puerto 443, ya que las conexiones se realizan de manera segura ya que están cifradas.

**2. Analiza el certificado digital presenta en el servidor web. Puedes hacer uso de las herramientas sslscan y ssltest del apartado anterior e indica:**

Texto

Descripción generada automáticamenteAntes de comenzar a responder las preguntas concretas sobre el certificado digital que presenta el servidor web, en este caso [www.uva.es](http://www.uva.es) voy a comentar brevemente como se accede a este, siguiendo los pasos de las siguientes capturas:

Captura de pantalla de un celular

Descripción generada automáticamente

Clicando en “El certificado es válido” ya estaríamos accediendo al certificado digital que presenta en este caso el servidor web de la UVa.

**a. Protocolo/s criptográfico y versión utilizado a nivel de transporte.**

Como no dispongo de una máquina Ubuntu que me permita instalar sslscan, ya que a la hora de intentarlo instalar todo el rato obtengo el mismo error use el comando que use, como se puede ver en esta captura:

Texto

Descripción generada automáticamente

Voy a utilizar la herramienta online proporcionada por el profesor, (<https://www.ssllabs.com/ssltest/>) donde en la barra de búsqueda nos pide introducir el servidor web que queremos que escaneé, donde introducimos [www.uva.es](http://www.uva.es).

Aquí vemos que tras unos segundos nos ha generado un informe con toda la información.



En el apartado de protocolos podemos ver que se esta usando el protocolo TLS (Transport Layer Security) en su versión 1.2.

Imagen que contiene Gráfico

Descripción generada automáticamente

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

Esta información podemos conocerla a través del apartado de detalles, del certificado digital que utiliza dicho servidor web. Como algoritmo de criptografía asimétrica utiliza el RSA y la longitud de la clave pública es de 2048 bits**.**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

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

Para obtener la clave pública accedemos a la herramienta sslscan online y vemos:



Como podemos ver la clave pública es 65537, número primo (4º de Fermat, suponiendo que no tenemos en cuenta el 0), la representación de este número en binario es, 10000000000000001. Es un primo bastante grande por lo que se garantiza la seguridad, además su representación y los cálculos son bastante fáciles, puesto que este compuesto por 15 ceros y 2 unos.

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

En el apartado Cipher Suites, sslscan muestra los algoritmos de criptografía de clave privada en orden de preferencia. Como se puede ver en la captura aparecen 4 algoritmos de los cuales los dos últimos se consideran débiles, por tanto, elegimos de los dos primeros el mejor. TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384.

Este algoritmo es el ECDH (Elliptic-Curve Diffie-Hellman), y en este caso de 256, es decir, la clave privada es de 256. Dicha clave privada ha de tenerla el servidor, ya que como alguien descubra o consiga esa clave va a poder conocer todas las comunicaciones.

**Imagen que contiene Texto

Descripción generada automáticamente**

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

Observando el certificado digital en el apartado “Algoritmo de firma de certificado” podemos ver que el que se utiliza en el certificado digital es SHA-384 con RSA.

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente con confianza media**

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

**Diagrama

Descripción generada automáticamente**

En el diagrama anterior podemos ver los pasos o fases que tienen lugar cuando intentamos acceder a un servidor web utilizando el puerto 80 TCP es decir, una conexión HTTP en vez de una conexión HTTPS con el puerto 443.

Como podemos ver primero realiza la petición HTTP, y este automáticamente realiza un redireccionamiento de puertos, como comento en el diagrama. Después el servidor devuelve la solicitud del certificado digital, dentro de este se encuentra la clave pública, K\_pub\_RSA. Ahora el cliente crea su clave privada con el algoritmo AES obteniendo K\_priv\_AES. Ahora RSA con esa clave privada, la pública cifra el mensaje a enviar M, obteniendo C (criptograma) y este criptograma se envía al servidor. Cuando el mensaje cifrado (criptograma) llega al servidor este puede conocer la clave privada del cliente y acceder al contenido del mensaje. Con esto habría finalizado la conexión que el cliente ha iniciado como HTTP con el servidor.

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.**

Primero voy a crear un directorio donde voy a crear tanto el certificado como las claves. Una vez creado el directorio generamos la clave privada de RSA. La longitud de dicha clave es 1024 bits, ya que lo indica el enunciado, pero en la realidad se utilizan de 2048 o 4096.

Como estamos generando una clave la extensión de esta ha de ser .key

Texto

Descripción generada automáticamente

Utilizando OpenSSL podemos ver información sobre la clave “alice.key”, entre toda la información se puede ver por ejemplo el valor de n:

Texto

Descripción generada automáticamente

Y podemos ver la clave privada, mediante el comando “*sudo cat alice.key”,* ya que sin permisos de super usuario nos deniega el permiso:

Texto

Descripción generada automáticamente

Y ahora procedemos a crear la solicitud de creación de certificado digital, que tiene como clave la privada que acabamos de crear para ello:

Texto

Descripción generada automáticamente

Debemos de ir rellenando los campos que nos van solicitando, aunque tenemos la posibilidad de dejar alguno de ellos en blanco, no tenemos por qué completar todo. Como podemos ver ya tenemos tanto la clave como la solicitud de certificado

Texto

Descripción generada automáticamente

Antes de continuar, dos cosas importantes sobre la solicitud que acabamos de crear. La primera es que nunca debemos de poner el correo en texto claro cuando nos lo solicita, en este caso lo hemos hecho porque es un ejemplo académico pero en la realidad nunca deberíamos de hacerlo.

Y la segunda es la extensión de la solicitud ¿Por qué se pone csr?, CSR contiene un bloque encriptado de texto que identifica el solicitante del certificado e incluye datos encriptados para cada uno de los datos que nos solicita. Además, este CSR está utilizado por una Potestad de Certificado para establecer prueba de identidad para sitios Web.

Ahora podemos ver el contenido de dicha solicitud con OpenSSL:

Texto

Descripción generada automáticamente

Ya tenemos tanto la clave “alice.key” como la solicitud de certificado “alice.csr”, con ello podemos crear un certificado digital autofirmado compatible con el estándar X.509. Para ello ejecutamos el siguiente comando:

Pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Ahora el certificado utiliza la extensión .crt, y lo hemos creado con el nombre “alice.crt”.

Para que no todo el mundo pueda acceder a la clave y certificado, vamos a mover (en nuestro caso copiar) los archivos creados a sus correspondientes directorios, ya que estos como luego veremos tienen los permisos adecuados.



Con el comando *“sudo ls –l /etc/ssl/”* podemos ver los permisos de los directorios que hay dentro del directorio /etc/ssl, y como vemos, /private únicamente tiene permisos de lectura el root. De esta forma, no puede acceder cualquiera a la clave privada, solo el dueño de la máquina o quién pueda acceder a los permisos de super usuario de dicha máquina.

Texto

Descripción generada automáticamente

Ahora ya estaría creado, pero para poderlo analizar con sslscan o con ssltest he tenido problemas puesto que sería necesario crear un servidor web. Por tanto, la forma de poder analizarlo con estas dos herramientas es creando un servidor web e introduciendo el certificado ahí. Esto es debido a que sslscan y ssltest solo permite analizar servidores web, y con ello certificados presentes dentro de ellos, no únicamente certificados aislados.

Aunque con las herramientas que nos pide el enunciado no se puede analizar el certificado OpenSSL si que lo permite, por tanto con el comando “openssl x509 -in alice.crt -text -noout” que he obtenido de la página web (<https://www.ibm.com/support/pages/openssl-commands-check-and-verify-your-ssl-certificate-key-and-csr>) podemos analizar el certificado y la salida que obtenemos es la que se ve en la siguiente captura.

Texto

Descripción generada automáticamente