Practica 1 FS

Alberto Brandon Velázquez

**Indice**

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

**1-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.).**

Para proceder a cifrar o descifrar mensajes se hará uso del comando -enc. Se utiliza tanto para cifrados de bloque como de flujo, haciendo uso de claves basadas en contraseñas o claves proporcionadas explícitamente (clave y vector de inicialización, en su caso).

Estos son algunos de sus comandos:

* -in filename

Se introduce el fichero que se quiere cifrar.

* -out filename

Se escribe el nombre del fichero de salida que contendrá el texto cifrado.

* -e

Para cifrar el fichero de entrada.

* -d

Para descifrar el fichero de entrada.

* -a

Para cifrar el fichero de entrada en Base64.

* -k password

Para especificar una contraseña (en desuso). Su equivalente y el que se recomienda usar hoy es -pass pass: password (aunque desaconsejamos esto)

* -kfile filename

Fichero que contiene la contraseña (en desuso). Su equivalente y el que se recomienda usar hoy es -pass file.

* -pass pass: password

Igual que "-k password".

* -pass file

Igual que "-kfile filename".

* -p

Cifra y muestra en pantalla la clave, el vector de inicialización y el valor de sal.

* -P

No cifra, simplemente muestra en pantalla la clave, el vector de inicialización y el valor de sal.

* -K key

Se proporciona la clave real directamente. Siempre que se introduce la clave, obligatoriamente se introducirá también el vector de inicialización (-iv IV) salvo que el algoritmo no lo use.

* -iter count

Utilice un número dado de iteraciones en la contraseña para derivar la clave de cifrado. Los valores altos aumentan el tiempo requerido para forzar el archivo resultante. Esta opción permite el uso del algoritmo PBKDF2 para derivar la clave (recordemos que PKBDF1 está obsoleto).

* -pbkdf1

Leer -pbkdf2. Tiene la misma funcionalidad que su sucesor, pero con la diferencia de que este último tiene capacidad para generar claves de más de 160 bits (128, 256, 512 bits). Hay que tener en cuenta que, si se usa una de las dos versiones a la hora del cifrado, se tiene que usar la misma función para el descifrado, ya que no son compatibles. Por ello, cuando se manipulen ficheros cifrados, hay que saber cuál de las dos versiones se ha usado. Aunque -pbkdf1 está en desuso, hay que conocerla debido a que existen ficheros antiguos cifrados con este sistema de derivación de claves.

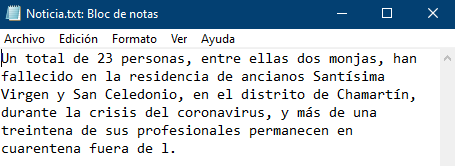
* -pbkdf2

Usar algoritmo PBKDF2 con el número de iteraciones predeterminado al no ser que diga lo contrario. PBKDF2 aplica una función pseudoaleatoria (HMAC con cualquier función hash aprobada: HMAC-SHA1) a la contraseña de entrada o frase de contraseña junto con un valor de sal y repite el proceso muchas veces (mínimo se recomienda 1000 iteraciones) para producir una clave derivada, que luego se puede usar como una clave criptográfica en operaciones posteriores. El trabajo computacional agregado hace que el "craqueo" de contraseñas por fuerza bruta sea mucho más difícil. La norma recomienda una longitud de sal de al menos 64 bits, que es la que utiliza OpenSSL.

* -md messagedigest

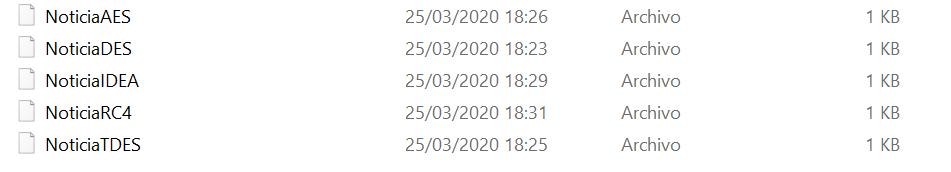
Especifica el resumen del mensaje que se utiliza para la derivación de clave. Puede tomar uno de los valores md2, md5 , sha o sha2 . El algoritmo predeterminado es sha2-256.

**2-Creamos un fichero texto plano llamado noticia con 279 caracteres**



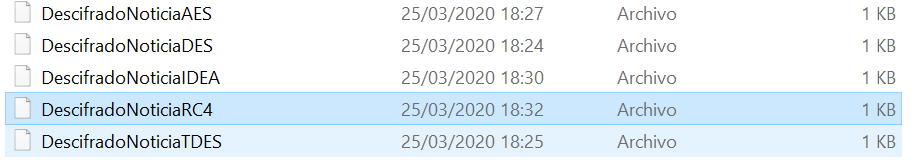
**3**-**Cifrarlo con CINCO algoritmos simétricos (AES y TDES obligatorios y un cifrador de flujo como mínimo).**

* enc -des-cbc -in Noticia.txt -out NoticiaDES
* enc des-ede3-cbc -in Noticia.txt -out NoticiaTDES
* enc -aes-256-cbc -in Noticia.txt -out NoticaAES
* enc -idea-cbc -in Noticia.txt -out NoticiaIDEA
* enc -rc4 -in Noticia.txt -out NoticiaRC4



**4-** **Descifrarlos y comprobar el resultado**

* enc -aes-256-cbc -in NoticiaCifrada -out DescifradoAES
* enc -rc4-in NoticiaRC4 -out DescifradoRC4
* enc -des-cbc -in NoticiaDES -out DescifradoDES
* enc -idea-cbc -in NoticiaIDEA -out DescifradoIDEA
* enc des-ede3-cbc -in NoticiaTDES-out DescifradoTDES



Para comprobar que todo está bien vemos que el descifrado es igual a la noticia inicial

**5-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-.**

Al comprobar el tamaño y los bytes de los archivos cifrados se puede comprobar que los métodos de cifrado por bloques como AES, DES, TDES y el IDEA generan archivos del mismo tamaño de 3.680 bytes y los métodos de cifrado de flujo RC4 tienen un tamaño de 3.665 bytes.

**6-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).**

En criptografía, **PKCS** (Public-Key Cryptography Standards) se refiere a un grupo de estándares de criptografía de clave pública concebidos y publicados por los laboratorios de RSA en California. A RSA Security se le asignaron los derechos de licenciamiento para la patente de algoritmo de clave asimétrica RSA y adquirió los derechos de licenciamiento para muchas otras patentes de claves.

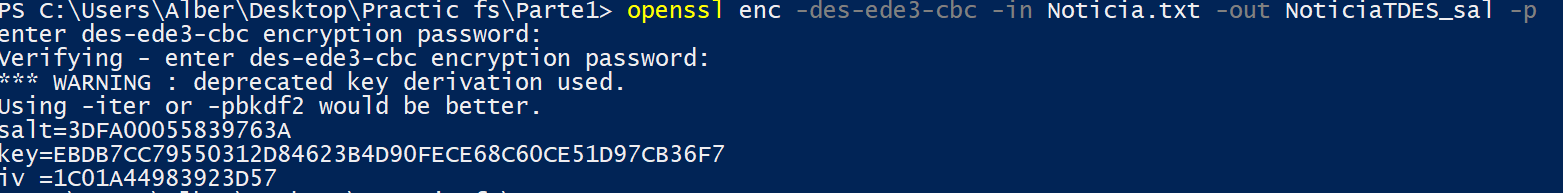
**El PKCS#5**

PBKDF2 aplica una función pseudoaleatoria, tales como código de autenticación de mensajes basado en hash (HMAC), a la entrada de contraseña o frase de contraseña junto con una sal de valor y repite el proceso varias veces para producir una clave derivada, que luego se puede utilizar como una clave criptográfica en las operaciones subsiguientes. El trabajo computacional añadido hace contraseña agrietamiento mucho más difícil, y es conocida como estiramiento clave

Practica:

*1-Caso:Haciendo que el sistema le asigne una clave,un iv y la sal*

* openssl enc -des-ede3-cbc -in Noticia.txt -out NoticiaTDES\_sal -p



* openssl enc -des-ede3-cbc -d -in NoticiaTDES\_sal -out DescifradoNoticiaTDES\_sal.txt -K F6F36323FEA0CC97C95AF2612044BDDB269C343AAC9E87E9 -iv F3F622C5E9E363AB

Nos deberia de dar:

Un total de 23 personas, entre ellas dos monjas, han fallecido en la residencia de ancianos Santísima Virgen y San Celedonio, en el distrito de Chamartín, durante la crisis del coronavirus, y más de una treintena de sus profesionales permanecen en cuarentena fuera de l.

Nos da como resultado:

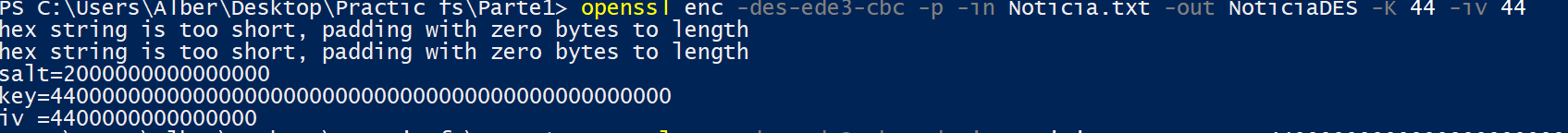
Ï¤¦j¿›F£è\*PL3^v lýòžÎã de 23 personas, entre ellas dos monjas, han fallecido en la residencia de ancianos SantÃ­sima Virgen y San Celedonio, en el distrito de ChamartÃ­n, durante la crisis del coronavirus, y mÃ¡s de una treintena de sus profesionales permanecen en cuarentena fuera de l.

Explicación de la diferencia:

Esta diferencia viene dada por la sal dada,por eso hay una diferencia entre el texto original y el descifrado.

*2-Caso: Dando nosotros la clave y el iv*

* openssl enc -des-ede3-cbc -p -in Noticia.txt -out NoticiaDES -K 44 -iv 44



* openssl enc -des-ede3-cbc -d -in NoticiaDES -out NoticiaDES\_sal.txt -K 440000000000000000000000000000000000000000000000 -iv 4400000000000000

Nos da como resultado:

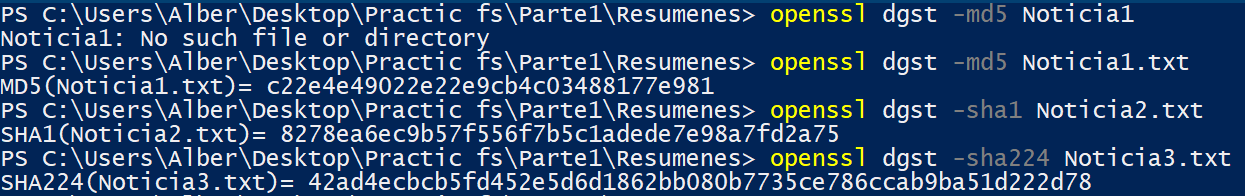
Un total de 23 personas, entre ellas dos monjas, han fallecido en la residencia de ancianos Santísima Virgen y San Celedonio, en el distrito de Chamartín, durante la crisis del coronavirus, y más de una treintena de sus profesionales permanecen en cuarentena fuera de l.

Nos debería de dar:

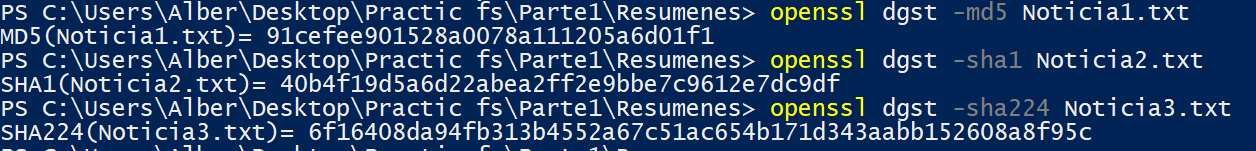
Un total de 23 personas, entre ellas dos monjas, han fallecido en la residencia de ancianos Santísima Virgen y San Celedonio, en el distrito de Chamartín, durante la crisis del coronavirus, y más de una treintena de sus profesionales permanecen en cuarentena fuera de l.

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

**1- 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.**



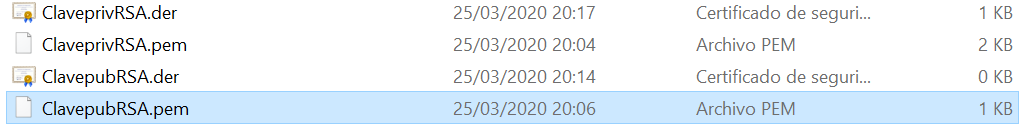
Cambiamos una mayúscula por una minúscula para ver si hay diferencia



Comprobamos que al cambiar solo una letra del texto se ha producido una variación en los resúmenes.

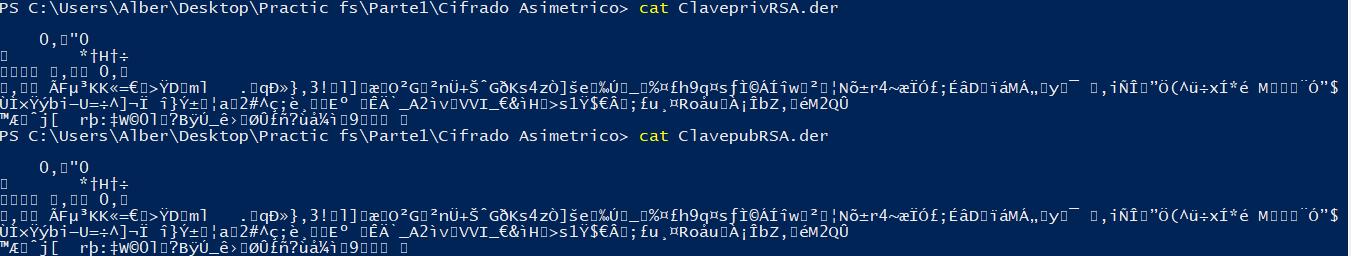
**2-Generar un par de claves asimétricas RSA de 2048 bits, de acuerdo con las indicaciones del apéndice A del manual básico (para RSA).**

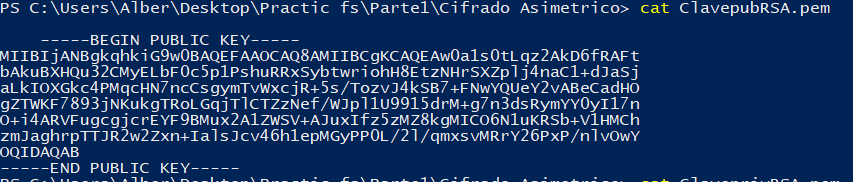
* openssl genpkey -algorithm RSA -aes256 -out ClaveprivRSA.pem -pkeyopt rsa\_keygen\_bits:2048(Generamos la clave privada).
* openssl pkey -in ClaveprivRSA.pem -pubout -out ClavepubRSA.pem(Obtenemos la clave publica).

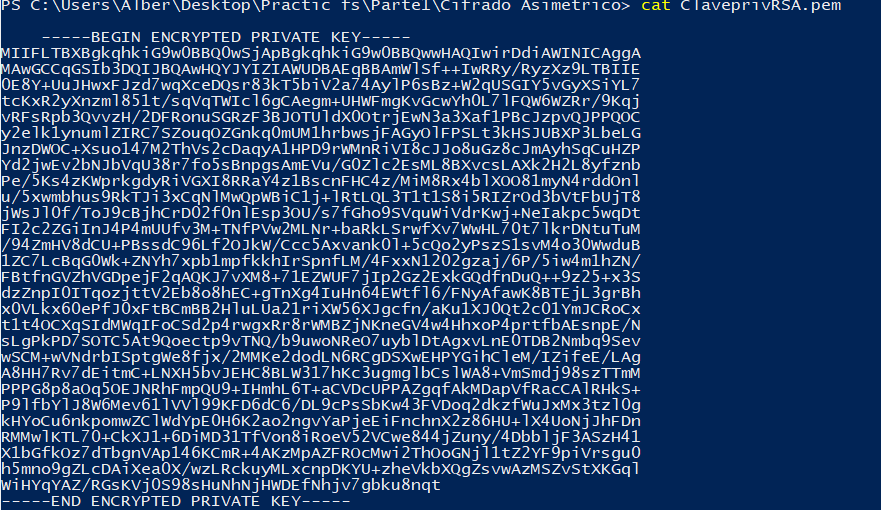


**3-Exportar dicho par de claves (pública y privada) en formato PEM (textual) y DER (binario). Utilizar los comandos de conversión de PEM a DER y viceversa.**

* openssl pkey -in ClavepubRSA.pem -pubin -outform DER -out ClavepubRSA.der(Cambiamos la clave pública de formato).
* openssl pkey -in ClaveprivRSA.pem -pubout -outform DER -out ClaveprivRSA.der(Cambiamos la clave privada de formato).

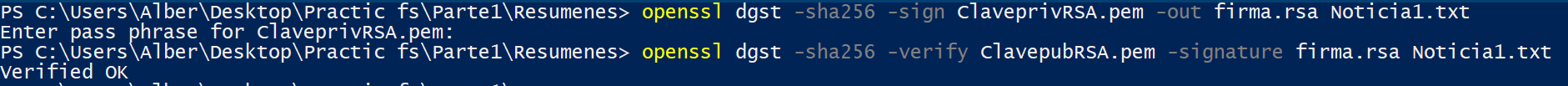






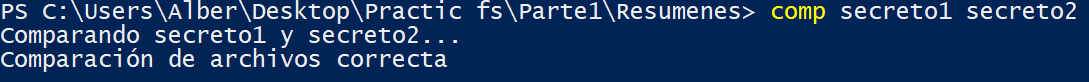
**4-Con los dos pares de claves asimétricas creadas, firmar y comprobar la firma del resumen (con SHA-256) de un texto cualquiera.**

* openssl dgst -sha256 -sign ClaveprivRSA.pem -out firma.rsa Noticia1.txt
* openssl dgst -sha256 -verify ClavepubRSA.pem -signature firma.rsa Noticia1.txt



**5-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.**

* openssl genpkey -algorithm X25519 -out priv1.pem (Creamos la clave priv con curva elíptica)
* openssl pkey -in priv1.pem -pubout -out pub1.pem (Creamos la clave pública)
* openssl genpkey -algorithm X25519 -out priv2.pem (Creamos la clave priv con curva elíptica)
* openssl pkey -in priv2.pem -pubout -out pub2.pem (Creamos la clave pública)
* openssl pkeyutl -derive -inkey priv1.pem -peerkey pub2.pem -out secreto1
* openssl pkeyutl -derive -inkey priv2.pem -peerkey pub1.pem -out secreto2
* comp secreto1 secreto2 (Comparamos los secretos)



**1.3 Cifrado Asimétrico de documentos**

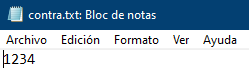
**1.-Documento cifrado (indicando algoritmo utilizado)**

* openssl enc -aes-256-cbc -pbkdf2 -in Noticia.txt -out texto.aes (ciframos con clave simétrica AES-256-CBC el fichero).
* pkeyutl -pubin -encrypt -in contrasena.txt -out contrasena.cif -inkey [Clave pública de la otra persona] (cifra de forma asimétrica el fichero)
* dgst -sha256 -sign ClaveprivRSA.pem -out firma.rsa texto.aes(firmamos con su clave privada RSA el resumen generado con SHA2-256 del fichero texto.txt)

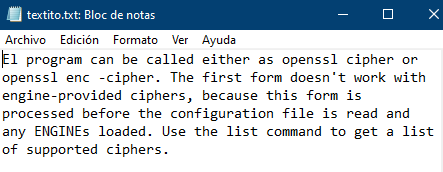
**2.-Clave simétrica empleada, cifrada con la clave pública del receptor**

* openssl pkeyutl -decrypt -in contraseña.cif -out contra.txt -inkey privadoBobRSA.pem





* openssl enc -d -aes-256-cbc -pbkdf2 -in texto.aes -out test.txt



* openssl dgst -sha256 -verify publickey.pem -signature firma.rsa test.ae



**3.-Resumen del documento original (indicando algoritmo) cifrado con la clave privada del emisor**

**RSA**

* openssl dgst -sha256 -sign ClaveprivRSA.pem -out firma\_res.rsa Noticia.txt
* openssl dgst -sha256 -verify ClavepubRSA.pem -signature firma\_res.rsa Noticia.txt

