Mª del Carmen Llorente Benedicto (carmenllorenteb@gmail.com | mcllorente@eiposgrados.edu.es)

CRIPTOLOGÍA

1. Diseña una presentación con alguna herramienta online (por ejemplo, un genial.ly, prezi, presentación en Google Drive o alguna herramienta similar). En dicha presentación tendrás que poner cinco ejemplos de nuestra vida cotidiana (Transacciones económicas, mensajería instantánea, ejemplos de IoT) en el que se utilice la criptografía e indicar qué tipo de algoritmo utiliza. Tendrás que explicar brevemente en qué consiste dicha técnica criptográfica.

   1 - Acceso seguro a servidores remotos: en lugar de utilizar usuario/contraseña para el acceso y gestión de servidores remotos las buenas prácticas recomiendan un acceso configurando el servicio OpenSSH/SSH de forma que solo se pueda acceder con clave público-privada.

   Uso el comando `ssh-keygen` para generar un par de claves público-privada para utilizar para conectar vía protocolo `ssh`. En el parámetro `-t rsa` se indica el algoritmo criptográfico a utilizar en la generación de claves, en este caso RSA, además, vía parámetro `-b 4096` indicamos que queremos un tamaño –en bits– de la clave de 4096 bits, conforme mayor tamaño, mayor seguridad. Se ha capturado la secuencia completa para generar la clave pública y privada, a saber:

```
ssh-keygen -t rsa -b 4096

Generating public/private rsa key pair.
Enter file in which to save the key (/Users/carmenllorenteb/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/carmenllorenteb/.ssh/id_rsa
Your public key has been saved in /Users/carmenllorenteb/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:z5q5IkUrLyxlSrK3InFf4j6sN1jLgAdcEX2/nmX2Mjg carmenllorenteb@MacBook-pro.local
The key's randomart image is:
+---[RSA 4096]----+
| .o              |
|. o . .          |
|.o    ..   .     |
| . . . .S o .    |
|o = * =. E o .   |
| B B @..+ + o    |
|+ + O Oo +       |
|.o.+oB o=.       |
+----[SHA256]-----+
```

Mª del Carmen Llorente Benedicto (carmenllorenteb@gmail.com | mcllorente@eiposgrados.edu.es)

La correspondiente clave privada (`id_rsa`) y pública (`id_rsa.pub`) que se usarán para las conexiones a los sistemas remotos se almacenan con los permisos adecuados en el directorio `~/.ssh/`, tal y como se aprecia en la siguiente ilustración:

```
❯ ls -la
total 48
drwx------   7 carmenllorenteb  staff   224 10 jul 21:49 .
drwxr-xr-x+ 56 carmenllorenteb  staff  1792  6 ago 18:17 ..
-rw-------   1 carmenllorenteb  staff  3454 10 jul 21:41 id_rsa
-rw-r--r--   1 carmenllorenteb  staff   759 10 jul 21:41 id_rsa.pub
-rw-------   1 carmenllorenteb  staff  4164 18 jul 18:16 known_hosts
-rw-------   1 carmenllorenteb  staff  3429 10 jul 21:43 known_hosts.old
-rw-------   1 carmenllorenteb  staff  2520 25 ene  2023 known_hosts_copia
~/.ssh
```

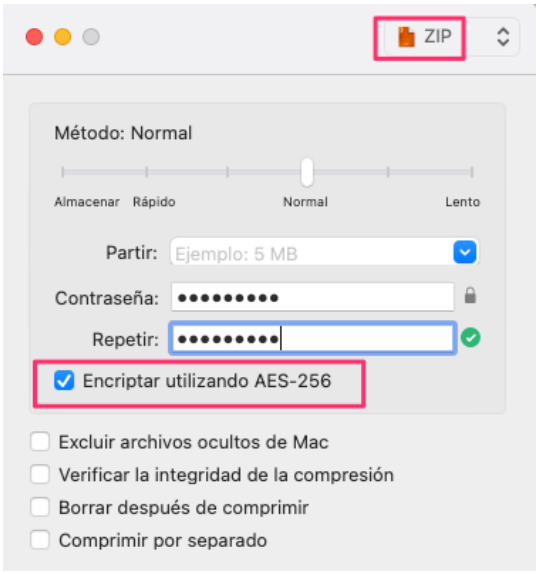
En pocas palabras, RSA se basa en la dificultad de factorizar un número entero que resulta de multiplicar dos números primos convenientemente grandes. Descomponer ese número en sus factores primos es muy costoso en términos computacionales, por eso la clave pública –en la que aparece el número entero resultante de multiplicar los dos primos escogidos– puede compartirse sin poner en riesgo la seguridad, ya que, aunque está relacionada con la privada, no puede obtenerse esta a partir de la pública –al revés sí es posible–.

2 – Acceso vía especificación FIDO2: la tecnología passwordless –sin contraseña– nos permite autenticarnos sin necesidad de las tradicionales contraseñas que son susceptibles a robos y fugas. Aunque hay distintos enfoques, mi favorito es la autenticación vía dispositivos, como una llave hardware –personalmente uso Yubikey–, de forma que puedo registrar en un servicio que soporte autenticación WebAuthn mi llave hardware, el proceso de registro realmente consiste en almacenar la clave pública del dispositivo registrado de forma que en el futuro podré autenticarme sin contraseñas, simplemente respondiendo un desafío que será cifrado con mi clave pública almacenada y solo el dispositivo original que contiene la clave privada asociada puede descifrar ese desafío, momento en el que la autenticación queda probada.

3 – Cifrado fichero zip con AES 256: es muy habitual cifrar información para enviarla adjunta a un correo de forma segura a un cliente, para ello me gusta utilizar un compresor que soporte **criptografía simétrica fuerte** como **aes 256**, de esta forma si tengo

Mª del Carmen Llorente Benedicto (carmenllorenteb@gmail.com | mcllorente@eiposgrados.edu.es)

que enviar a un cliente información cifro los documentos en un fichero comprimido y cifrado utilizando aes 256, cuya clave secreta única, que debemos conocer tanto emisor como receptor, presenta una gran seguridad con una longitud de 256 bits. Para comunicar la clave podría hacer uso de un **canal paralelo seguro** previamente fijado, por ejemplo, un canal Telegram, plataforma conocida por su enfoque hacia la seguridad y privacidad del usuario, los canales Telegram presentan cifrado extremo a extremo y ni siquiera los empleados de Telegram pueden acceder a los mensajes intercambiados.



4 - Protocolo Nostr: me preocupa la censura o que puedan afirmar que he publicado algo que no sea cierto en redes sociales, por ello utilizo el protocolo Nostr para comunicarme en una red descentralizada, resistente a la censura y en la que se puede autenticar el autor e integridad de cada mensaje dado que Nostr hace uso de criptografía de clave pública-privada, de forma que, por defecto, los mensajes van firmados. Nostr es un protocolo sencillo, estándar abierto, sobre el que se puede construir cualquier aplicación, en mi caso utilizo en iOS el cliente Damus.

Mi clave pública –que también es mi id, por tanto, se me puede localizar con él en Nostr– es:

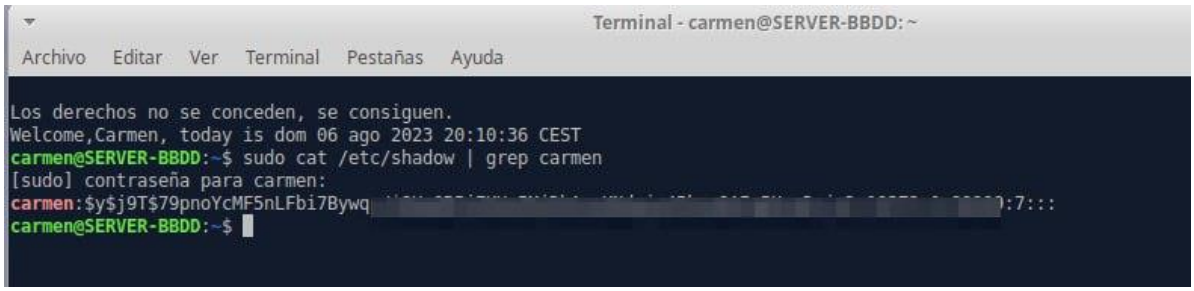
`npub182eksedpe79m7f26h4kdnmx2trtz5exhtxrtathc8cnn4ufln06qxmeat7`

Mª del Carmen Llorente Benedicto (carmenllorenteb@gmail.com | mcllorente@eiposgrados.edu.es)

Todas mis publicaciones se firman con mi clave privada de forma automática y como mi clave pública está disponible en mi perfil cualquiera puede comprobar que soy yo, no se me puede suplantar. Los mensajes privados se cifran con la clave pública del destinatario de forma que solo él puede acceder al contenido.

En el lateral derecho se muestra una captura de mi cliente Damus de Nostr para dispositivo iOS, también existen clientes web y multitud de implementaciones distintas.

5 – Almacenamiento seguro de contraseñas: al introducir nuestra contraseña en un sistema Unix, por ejemplo, la misma se puede comprobar gracias a que en el fichero `/etc/shadow` –si no se ha conectado con un repositorio o sistema de autenticación centralizado– se almacena el resumen criptográfico de la contraseña original. Dado que el resumen criptográfico o función hash, por ejemplo, bcrypt –ideado específicamente para ser resistente a ataques de fuerza bruta por su alto coste computacional, ideal para guardar contraseñas–, es una función de sentido único, significa que a partir del resumen criptográfico o hash resultante no puede obtenerse el texto de entrada. De esta forma, es posible verificar que la contraseña introducida, tras aplicar la misma función hash bcrypt, obtiene un resumen criptográfico o hash idéntico al almacenado en el fichero `/etc/shadow`. Si ambos hashes coinciden, es que la contraseña es correcta, dado que con bcrypt es prácticamente imposible que haya una colisión –que dos entradas distintas tengan como resultado la misma salida en la función hash–. En la siguiente ilustración se aprecia una entrada del fichero `/etc/shadow` para el usuario “carmen”, junto al usuario hay un campo que incluye el resumen criptográfico o función hash Bcrypt.



Mª del Carmen Llorente Benedicto (carmenllorenteb@gmail.com | mcllorente@eiposgrados.edu.es)

Se ha realizado una presentación del apartado previo que puede consultarse en línea. Para ver la presentación **pulse el siguiente enlace**.

Mª del Carmen Llorente Benedicto (carmenllorenteb@gmail.com | mcllorente@eiposgrados.edu.es)

2. Crea un acertijo utilizando la esteganografía. Para ello tendrás que proponer como ocultar la información, el programa utilizando y los pasos realizados y luego mostrar, mediante capturas de pantalla y explicándolas, los pasos a seguir para obtener la solución. Tendrás que elaborar un documento que entregarás en PDF a través de la plataforma. En el mismo tendrás que poner una portada con tus datos y con los del trabajo y explicarás mediante la utilización de texto y capturas de pantalla todos los pasos realizados. Además de la solución entregada se evaluará el formato y la calidad del documento entregado como solución.

El programa de esteganografía elegido es Steghide –licencia GNU/Linux, de código abierto– el cual permite ocultar información sensible en diferentes medios digitales. Muy cómodo e incluso automatizable, dado que se trata de una herramienta que se ejecuta en línea de comandos.

Entre sus características principales:

- Steghide puede ocultar datos dentro de archivos de imagen (formatos JPEG, BMP) y archivos de audio (formatos WAV y AU). Esto permite que los datos se mezclen con el contenido visual o auditivo, lo que dificulta su detección para el ojo u oído humano, incapaz de detectar las sutiles diferencias que el proceso de esteganografía produce en el fichero resultante.

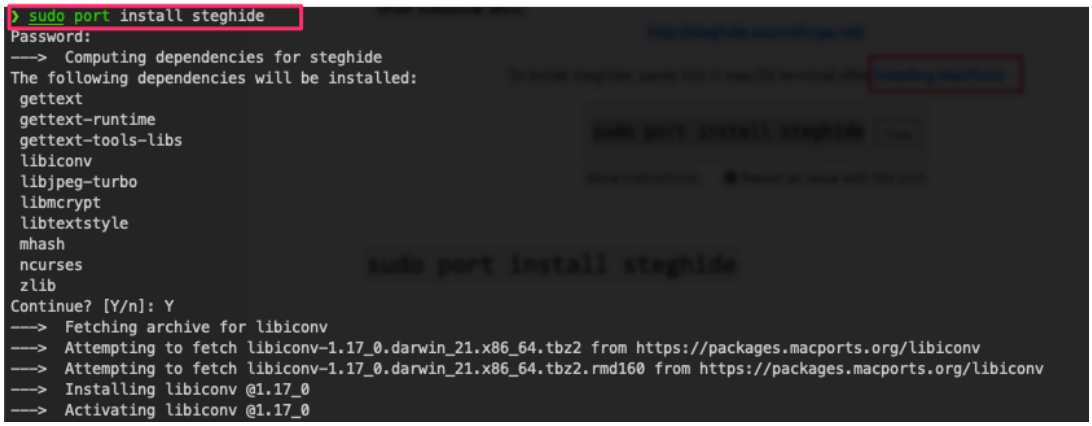
- Es interesante que la herramienta admite el uso de contraseñas simétricas –AES, Advance Encryption Standard– para proteger los datos ocultos. De esta manera, solo las personas que conocen la contraseña pueden extraer la información oculta, si bien lo que se suele pretender con la esteganografía es el pasar desapercibido, aunque no deja de ser una capa de seguridad adicional, por si se descubre que hay un mensaje oculto (probablemente lo usaría un espía en un entorno hostil y lo que quiere es evitar ser detectado).

Por fortuna la herramienta es multiplataforma y he podido instalarla en mi Mac vía MacPorts.

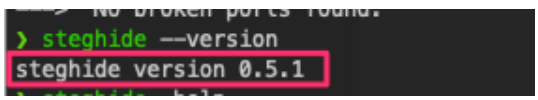
Mª del Carmen Llorente Benedicto (carmenllorenteb@gmail.com | mcllorente@eiposgrados.edu.es)

Una vez instalada, ejecutamos en el terminal:

```
sudo port install steghide
```



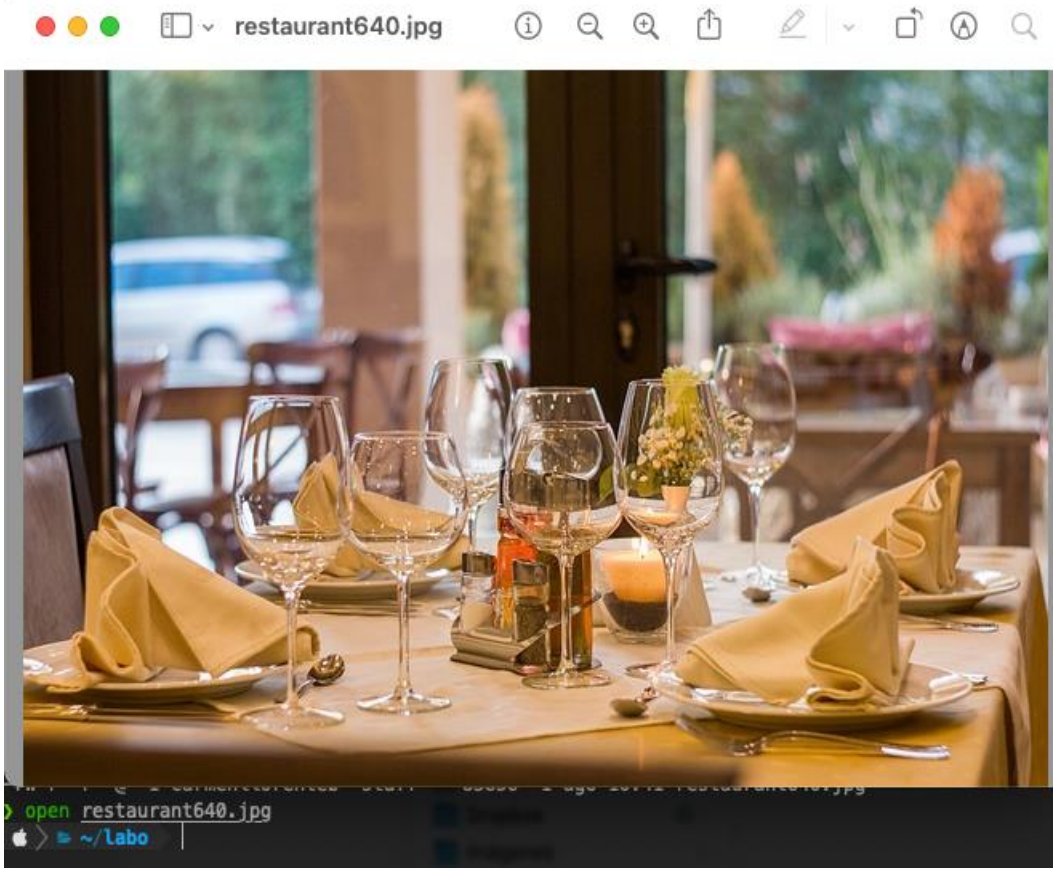

```
steghide --version
```



Antes de **ocultar el mensaje** he obtenido un **resumen criptográfico SHA 256 de la imagen** que voy a usar para ocultar el mensaje, de esta forma podré comparar su integridad tras embeber el mensaje. Se ha de notar que, aunque `steghide` afirma ser **resistente** a **análisis estadístico**, este control serviría para detectar fugas de información mediante esteganografía en una empresa.

Mª del Carmen Llorente Benedicto (carmenllorenteb@gmail.com | mcllorente@eiposgrados.edu.es)

A continuación, se muestra la imagen en la que se ocultará el mensaje secreto o adivinanza. Al ser una imagen con una alta resolución y profundidad de colores, los cambios que el proceso de esteganografía produzca no deberían ser apreciables al ojo humano.



Como se ha comentado se muestra el resumen criptográfico sha 256 previo a la operación de esteganografía, posteriormente se podrá verificar que efectivamente la integridad de la imagen –incluso el tamaño en octetos, marcado en la captura en azul– ha sido modificada.

```
shasum -a 256 restaurant640.jpg
127a16a3e00a4e6dd6202a4394b264dfb9d27d1b966ff71e01ccdd69d59e0d28 restaurant640.jpg
```

Creamos el fichero **cita.txt** con el mensaje de Robert Collier que vamos a ocultar en la imagen haciendo uso de un simple “echo”, ver ilustración siguiente.

Mª del Carmen Llorente Benedicto (carmenllorenteb@gmail.com | mcllorente@eiposgrados.edu.es)

```
❯ echo "El éxito es la suma de pequeños esfuerzos repetidos día tras día. Robert Collier." > cita.txt
❯ steghide embed -ef cita.txt -cf restaurant640.jpg
Anotar salvoconducto:
```

Para ocultar el **contenido de cita.txt en la imagen** se ha ejecutado –ver imagen previa– el siguiente comando.

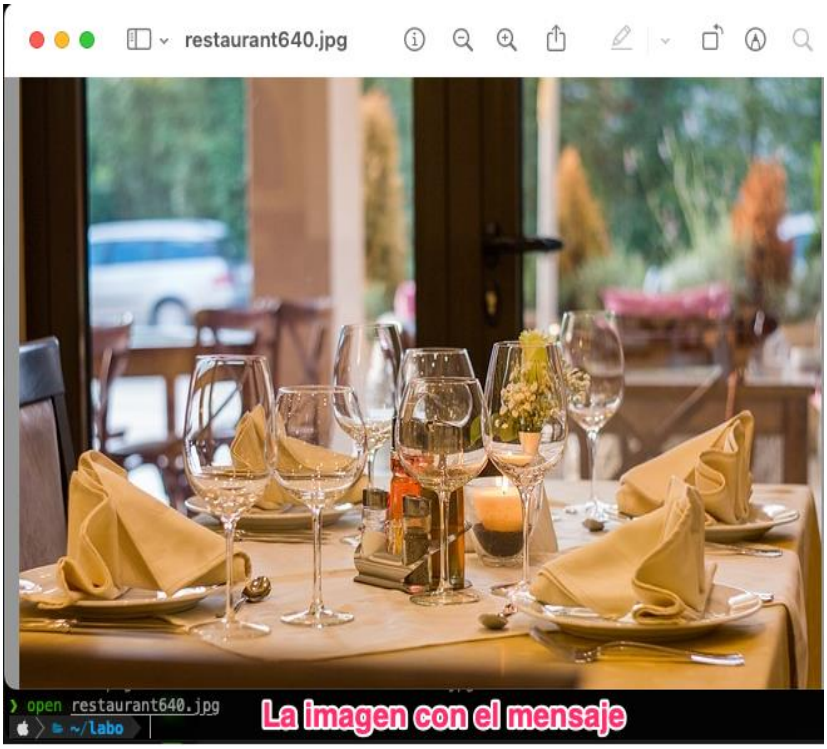
```
steghide embed -ef cita.txt -cf restaurant640.jpg
```

Se ha de notar que, se solicita una contraseña o "salvoconducto", simplemente pulsamos enter ya que lo único interesante aquí es que es posible, si se quiere, prefijar una contraseña dada para poder obtener el mensaje oculto, si bien lo interesante es el hecho mismo de que exista un mensaje oculto.

Una vez que hemos incorporado el fichero en la imagen volvemos a realizar un resumen criptográfico y vemos que ha cambiado tanto el **resumen criptográfico** o **hash** como el **tamaño del fichero**. Ver ilustración siguiente.

```
adjuntando "cita.txt" en "restaurant640.jpg"... hecho
❯ shasum -a 256 restaurant640.jpg
7e40cf6fe91ec2b733ccfd5a8ad50cbff0ce4e6ea0098d2ab597568854690867  restaurant640.jpg
❯ ls -l restaurant640.jpg
-rw-r--r--@ 1 carmenllorenteb  staff  86155 1 ago 11:31 restaurant640.jpg
```

En la siguiente ilustración se aprecia la imagen ya modificada –con el mensaje oculto embebido–, si bien resulta inapreciable al ojo humano.

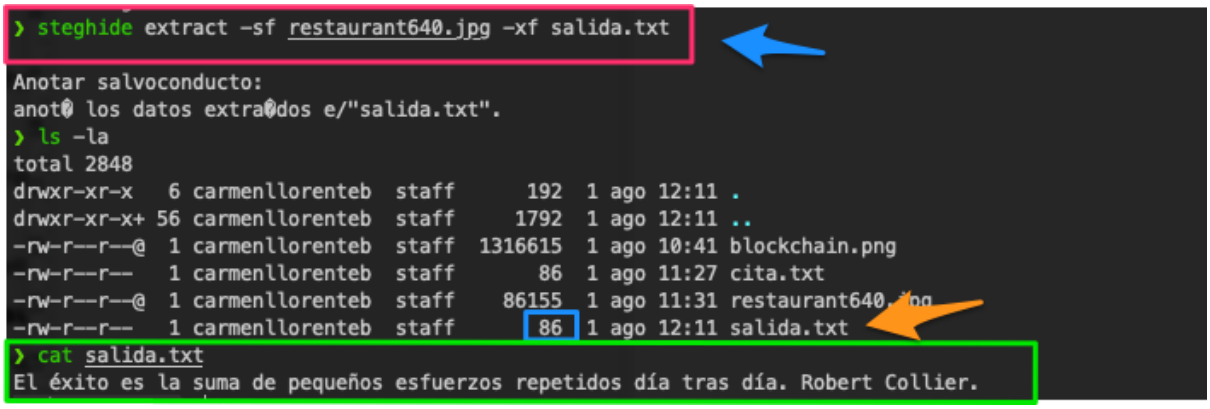


Mª del Carmen Llorente Benedicto (carmenllorenteb@gmail.com | mcllorente@eiposgrados.edu.es)

Para **extraer la información oculta en una imagen** utilizamos el siguiente comando, el mensaje oculto se almacenará en el fichero `salida.txt`, de esta forma podemos comprobar que el resultado es idéntico al mensaje original comparando los ficheros salida.txt y cita.txt (podríamos hacer un diff o simplemente leer los mensajes o realizar un hash md5 de ambos ficheros a ver si coincide).

```
steghide extract -sf restaurant640.jpg -xf salida.txt
```

En la siguiente ilustración se comprueba que el fichero resultante "`salida.txt`", coincide con el mensaje oculto que se introdujo vía fichero "cita.txt".



Para finalizar me gustaría comentar que, aunque no he realizado la prueba, es muy probable que, si aplicamos **hashes difusos**, se pueda detectar que una imagen es muy similar a otra, y esto podría servir como mecanismo para detectar imágenes que pudieran contener algún tipo de mensaje oculto. Ya que los hashes difusos permiten encontrar ficheros que son muy similares, que presentan solo ligeras alteraciones.