Memoria de la entrega 1

Criptografía y Seguridad Informática

Raúl Aguilar Arroyo

Alberto Penas Díaz

Grupo 8001

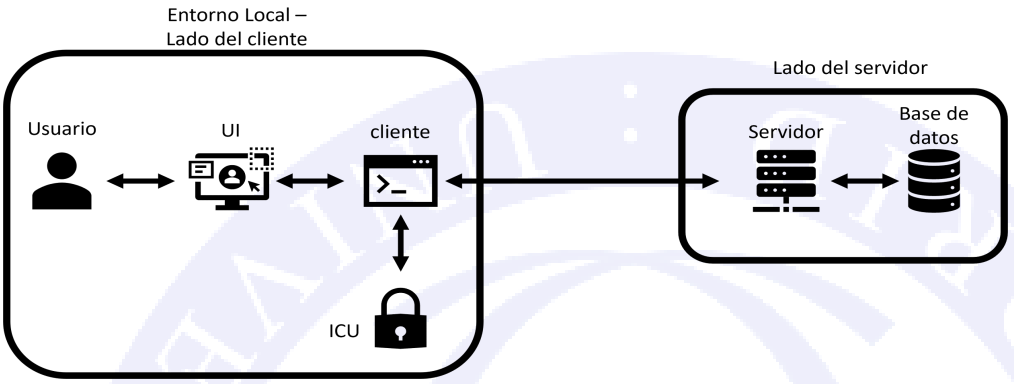
27 oct 2023

Índice

Propósito de la aplicación

El propósito general de la aplicación es encriptar localmente secciones de imágenes que se almacenan en un servidor.

Funcionamiento de la aplicación



La aplicación está compuesta de dos partes muy diferenciadas, el lado del cliente y el lado del servidor.

Lado del cliente

Este es el encargado de encriptar las imágenes y almacenar (mientras dure la sesión) la contraseña del usuario. También es el encargado de enviar correctamente las imágenes y la información al servidor. Esta parte se divide a su vez en 3 componentes:

- **Interfaz de usuario (UI):**
  - Es simplemente la interfaz gráfica contra la que interactúa el usuario, está recopila las entradas, se las manda al cliente y muestra las imágenes e información que el cliente le proporciona.
- **Cliente:**
  - Se encarga de enviar la información al servidor y a la interfaz gráfica, así como de hacer las solicitudes al servidor. Proporciona las funciones a las que tiene acceso el usuario.
- **Paquete Image Crypto Utils (ICU):**
  - Es el encargado de encriptar y desencriptar imágenes, así como de generar los MAC de las mismas.

Lado del servidor

Este es el encargado de autenticar a los usuarios y de administrar las imágenes que se suben. A nivel lógico se divide en servidor y la base de datos.

- **Servidor:**
  - Es el encargado de autenticar a los usuarios, exigir la robustez de las contraseñas, la generación de KDFs de las contraseñas para su almacenamiento y la verificación de la originalidad de las imágenes proporcionadas
- **Base de datos:**
  - Es la encargada de administrar la localización y búsqueda tanto de imágenes como de la información de los usuarios.

Autenticación de usuarios

Registro de usuarios

Los usuarios se registran en la aplicación con un nombre y una contraseña. Esta información se envía al servidor, que comprueba su previa inexistencia del usuario y la robustez de la contraseña (la cual debe tener al menos una mayúscula, una minúscula, un número y un símbolo).

Después de comprobar la validez del nombre y la contraseña se genera un salt aleatorio de 128 bits (lo recomendado para Scrypt). Con este salt y la función de KDF "Scrypt", especialmente diseñada para almacenar contraseñas se genera un Hash de 256 bits (también lo recomendado para Scrypt). Este Hash, junto al salt usado para generarlo se guarda en la base de datos.

Inicio de sesión

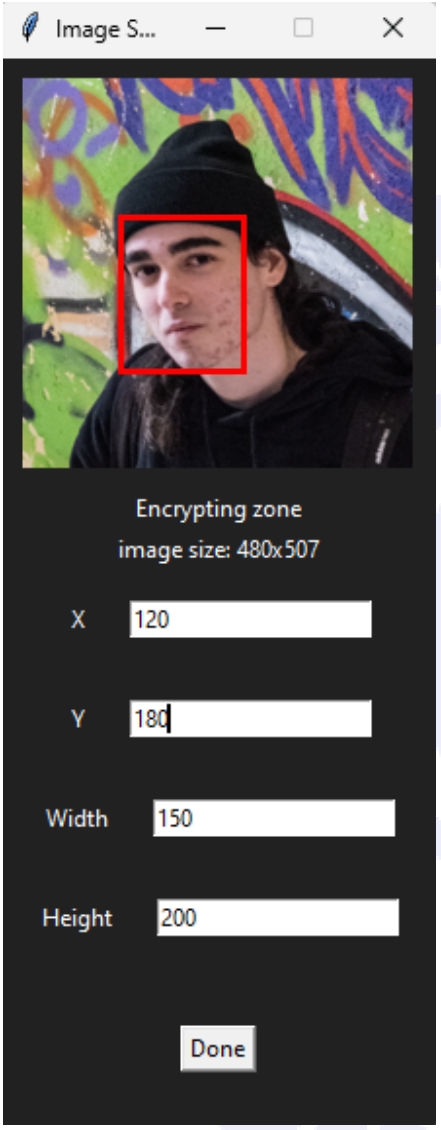
Si un usuario ya está registrado en la aplicación puede iniciar sesión introduciendo su nombre y contraseña. Esta información se envía al servidor, el cual usando el salt correspondiente a este usuario regenera el hash y comprueba la coincidencia. Después de comprobar la identidad del usuario se procede a regenerar el salto y generar un hash nuevo para actualizarlo en la base de datos. Esto hace que en cada inicio de sesión el hash derivado de la contraseña del usuario cambie para mayor seguridad.

Cabe destacar que el servidor únicamente tiene la contraseña del usuario mientras dura este proceso, esta no se almacena en ningún lado ni por parte del cliente ni por parte del servidor.
Ejemplo de entrada de la base de usuarios:

```
Unset
[
  {
    "name": "admin",
    "password": "44424dfd649a044823b3938b24cefca22049a87803c9ed1ae1ca2d2510e3b8af",
    "salt_p": "1062e18dc1e44e1f80e392c0f97e7b62"
  }
]
```

Cifrado simétrico

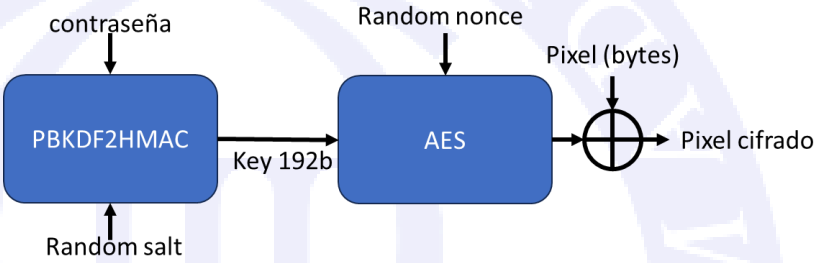
El cifrado simétrico se emplea en las imágenes. Una vez que un usuario se ha registrado en la aplicación puede añadir imágenes en el siguiente menú:



Aquí seleccionas una imagen y el área a encriptar.

Después el ICU se encarga de encriptar únicamente esa área, pixel a pixel.

Como clave de encriptación se utiliza una clave derivada de la contraseña con el método PBKDF2HMAC (que está pensado para esto), para ello se genera un salt aleatorio de 128 bits que junto a la contraseña generarán nuestra clave



de 192 bits.

Con esta clave y un nonce aleatorio se realiza AES con modo CTR y se encriptan todos los píxeles de la imagen.

Posteriormente el nonce y el salt se guardan en la metadata de la imagen para poder regenerar la clave más adelante para el desencriptado.

Hay que tener en cuenta que la contraseña únicamente es conocida por el usuario y es suficientemente compleja, por lo tanto el servidor no debe ni puede recrear la clave para descifrar la imagen y por lo tanto en ningún momento tiene acceso al contenido de esta.

Se ha decidido utilizar el método de derivación de clave ya que permite almacenar las imágenes durante tiempo indefinido sin necesidad de almacenar la clave, lo que consideramos que sería inseguro, así en un futuro únicamente sabiendo la contraseña se podrá descifrar la imagen.

Cabe destacar que todas las imágenes utilizan claves distintas ya que el salt usado para derivar las claves es aleatorio, además obviamente utilizan nonces distintos y aleatorios.

Autenticación de mensajes

Cuando el cliente envía imágenes al servidor, se genera un código hash para que el servidor verifique la autenticidad del mensaje. Esto se lleva a cabo para impedir la manipulación de las imágenes en el canal de transmisión cliente-servidor. El tipo de hash que se genera es un **SHA-2 256** (Estandarizado por el NIST), cuya clave es un valor aleatorio generado con *os.urandom()* que posteriormente se guarda en los metadatos de la propia imagen (ya que es necesario transmitir de alguna manera la clave que se ha usado para generar el hash al servidor). El hash como tal, está generado a partir de los bytes de la imagen una vez ha sido encriptada, además de el nonce del AES, la salt y la propia clave del código hash, como ya se ha mencionado anteriormente y se almacena de nuevo en los metadatos de la imagen.

Si alguno de estos valores se ve modificado, el servidor descartará la imagen recibida y notificará al cliente del error, al comprobar que el código de autenticación hash extraído de los metadatos de la imagen no coincide con el hash generado.

Asunciones/Requisitos del sistema

Estamos asumiendo que la comunicación cliente-servidor se realiza por un canal privado que puede tener pérdidas de bits (como cualquier red), es por ello que permitimos que se mande la contraseña sin cifrar al servidor, así como la clave de generación del MAC. Cabe destacar que el MAC sigue siendo necesario por si se producen pérdidas en el paquete.

Por tal y como funciona nuestro sistema de encriptado (por pixeles), es necesario que las imágenes se almacenan en formato *png*. Esto sucede porque cualquier otro formato utiliza una compresión que distorsiona el color de los píxeles en la imagen, lo que hace imposible su posterior descifrado. En principio se pueden cargar imágenes en otros formatos y se transformaran a png, sin embargo no está completamente probado el correcto funcionamiento de todo el proceso cargando formatos distintos a png.

Contenido extra

Nuestra aplicación cuenta con funcionalidades extra que agregan seguridad y dinamismo a nuestro software:

- Verificación de robustez de la contraseña al registrar un nuevo usuario, notificando (si así fuera preciso) que es necesario 12 caracteres como mínimo, la utilización de dígitos y algún carácter especial.
- Hemos implementado una base de datos bastante compleja:
  - Almacenamiento de la información de los usuarios.
  - Almacenamiento e indexación de las imágenes para poder clasificarlas por usuario y fecha mejorando así el manejo de las mismas.
  - Está programada para limpiar automáticamente la información de usuarios no existentes, limpiando las distintas rutas cuando se eliminan imágenes o usuarios para no dejar restos y mantenerse limpia.
- Renovación de hashes de contraseña en cada login. Cada vez que un usuario inicia sesión en la aplicación, el servidor renueva automáticamente el salt y el hash derivado de la contraseña del mismo para mayor seguridad.
- Algunas mejoras en la interfaz que mejoran la experiencia del usuario:
  - Se ha añadido una pantalla de carga[1]
  - Ventanas emergentes y mensajes de error.
  - Hemos conseguido que la selección de la región de imagen a encriptar sea visible en tiempo real cuando se selecciona una imagen a encriptar.
- También consideramos que el problema que hemos decidio enfrentar, y el enfoque que le hemos dado es bastante original e implica ciertas complejidades en el desarrollo, como por ejemplo:
  - Aprender el funcionamiento de los diferentes formatos de imagen

[1] Implementamos una pantalla de carga ya que si un usuario tiene un número grande de imágenes, el cliente tarda un tiempo considerable en desencriptarlas. Una sola imágen de 1500x1500 requiere ir pixel a pixel en un bucle de 2.250.000 iteraciones solo para una única foto.

- Desarrollar un paquete capaz de tratar pixel a pixel las imágenes
- Implementar los algoritmos para seleccionar las áreas, encriptar únicamente esas áreas y sobre escribirlas en la imagen de forma correcta

Link al Repositorio

https://github.com/seniorbeto/Criptografia_2023-24