Memoria de la entrega 2

Criptografía y Seguridad Informática

Raúl Aguilar Arroyo

Alberto Penas Díaz

Grupo 8001

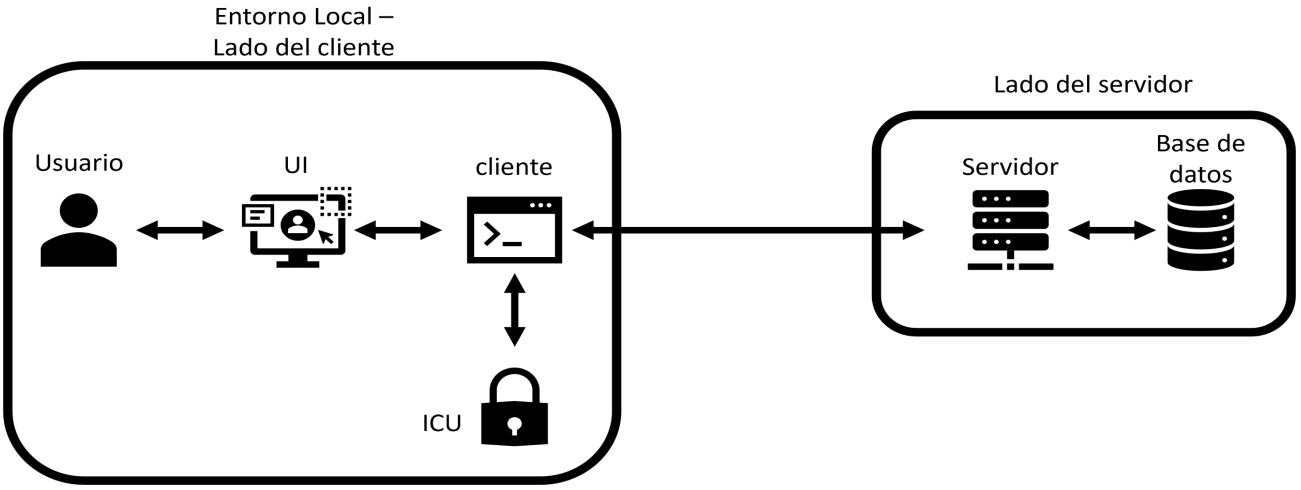
27 oct 2023

Índice

Propósito de la aplicación

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

Estructura de la aplicación



La aplicación está compuesta de dos partes muy diferenciadas, el lado del cliente y el lado del servidor.
También están implementadas las autoridades de certificación, aunque lógicamente, en una aplicación real estas serían completamente ajenas.

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.

Autoridades de Acreditación

Hemos creado una serie de autoridades que se encargan de firmar los certificados que utilizan el servidor y el cliente (se explica en la página 4)

- **El Papa**: Es la autoridad máxima y se autoafirma su certificado.
- **Pedro Sanchez** y **Ursula** (von der Leyen) que certifican cliente y al servidor respectivamente

Se indaga más profundamente en el funcionamiento de la aplicación en la página 4

Firma Digital

Utilizamos la firma digital para acreditar cada una de las imágenes que envía el cliente al servidor. De esta manera, aportamos a ésta comunicación un método para verificar la identidad del cliente así como el "no repudio". Hemos decidido emplear el método de cifrado RSA para firmar el hash de autenticación de imágen (véase la memoria de la primera entrega), de esta manera, el cliente envía **directamente** al servidor la siguiente información:

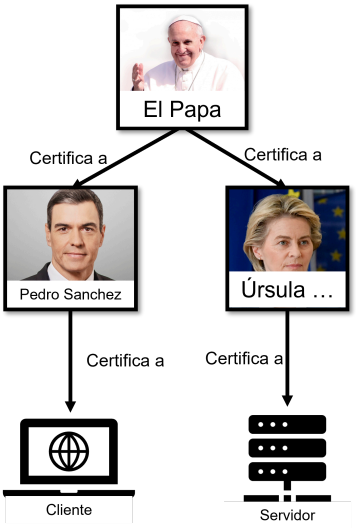
1. **hash:** resumen de la imagen previamente encriptada utilizando como generador una clave de 32 bytes generada aleatoriamente, el nonce y el salt.
2. **key:** la clave que se ha utilizado para generar el resumen de la imagen. Este key va encriptado con la clave pública (obtenida del certificado) del servidor.
3. **signature:** la firma (con la clave privada del usuario) del hash.

De esta forma, el servidor, haciendo uso de la clave pública del usuario (que el mismo hace llegar al servidor a través del certificado) es capaz de comprobar que, efectivamente, el hash de la imagen coincide con el resultado de descifrar la *signature* con la clave pública del cliente.

Es conveniente explicar que este par de clave pública y clave privada se renueva cada vez que se empieza una nueva comunicación entre cliente y servidor.

Certificados de clave pública

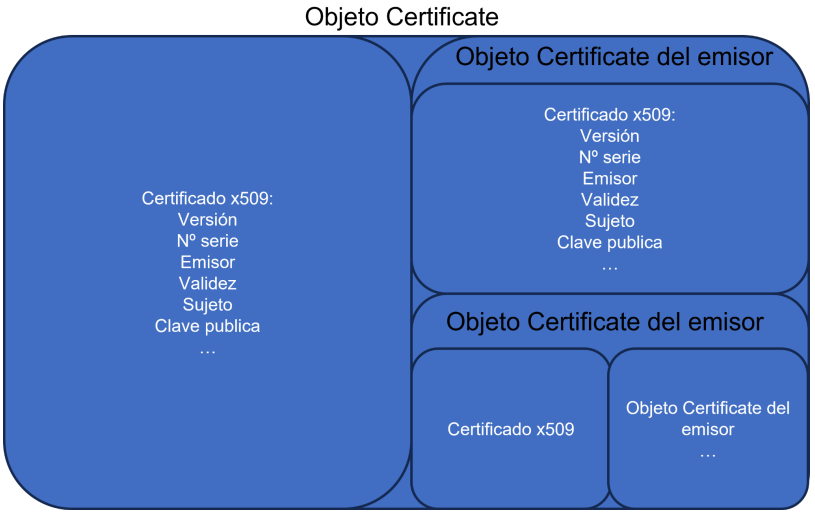
Infraestructura de clave pública



Consta de 3 niveles (aunque por la implementación es sencillo escalar a tantos niveles como se quiera), hemos designado al Papa como CA raiz y a Pedro Sanchez y Ursula Von Der Lien como autoridades subordinadas.
Los usuarios finales son el cliente y el servidor, cuya única autoridad en común es el CA raíz.
Por lo tanto cuando se verifican los certificados siempre acaban acudiendo a la CA raíz.

Generación de certificados y PKs

La generación de certificados y claves públicas se lleva a cabo de la siguiente manera.
Primero el CA raíz emite su certificado autofirmado después los CA subordinados emiten un CSR firmado por ellos en el que adjuntan su clave pública y alguna otra información. Esta solicitud se envía a la autoridad superior, en este caso el Papa, el cual revisa la solicitud y genera un certificado firmado por el. Este certificado se guarda junto al certificado de la autoridad que lo ha firmado (para simplificar luego el proceso de buscar autoridades comunes).
Un usuario final (el servidor y el cliente) solicita los certificados a sus respectivas autoridades seleccionadas. Y al igual que antes se genera un csr que se envía a la autoridad, la cual lo "revisa" y firma.



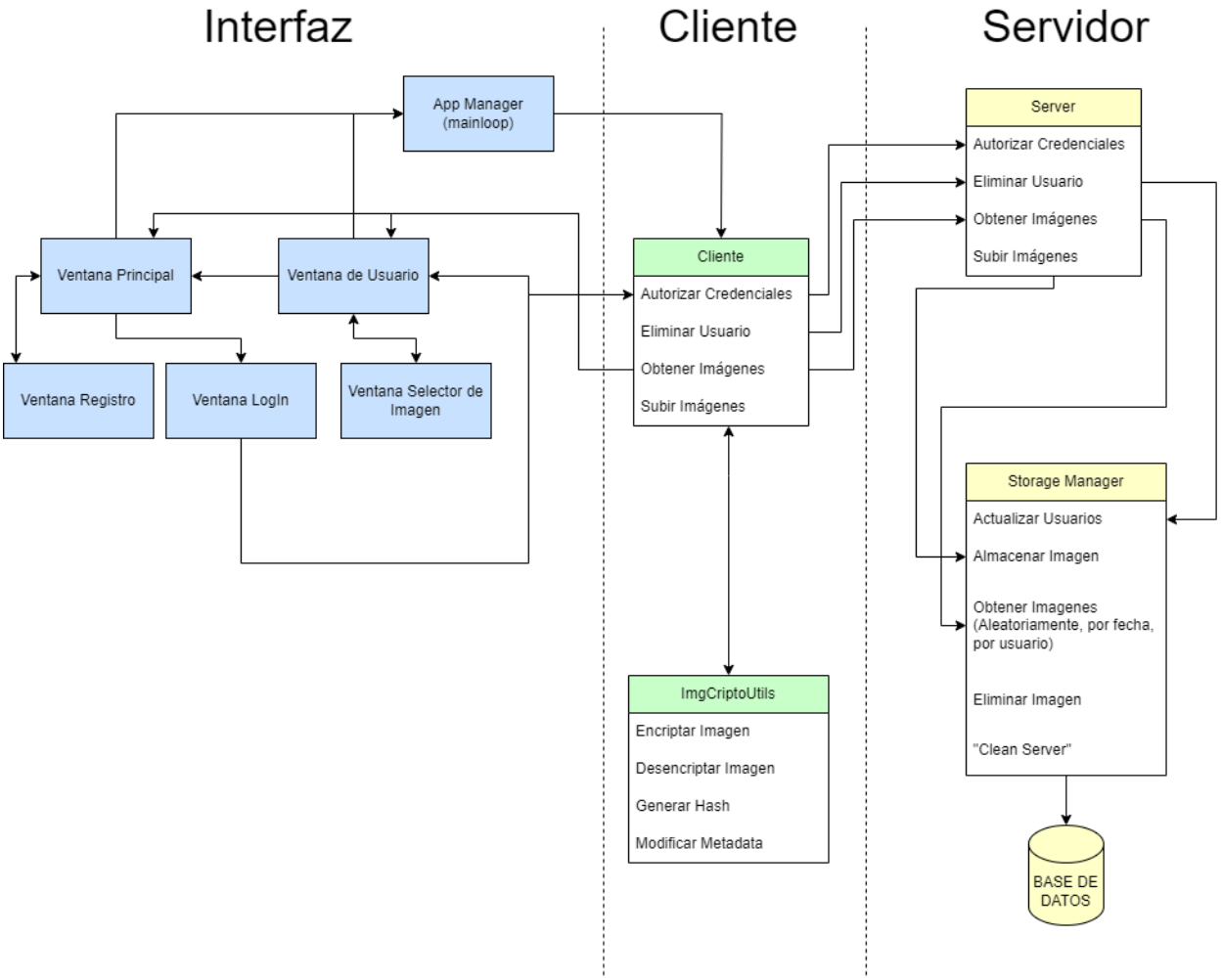
A nivel implementación hemos incluido un certificado dentro de otro de forma recursiva para simplificar la búsqueda de autoridades comunes

Para los procesos que lo requieren, las claves públicas siempre se obtienen de los certificados y siempre que se recibe un certificado (ya sea cliente o servidor) se verifica la validez de este, así como su firma y que haya sido emitido por una autoridad de confianza.

Las autoridades tienen la capacidad de emitir y revocar certificados.

Complejidad y código de la aplicación

Consideramos que nuestra aplicación es considerablemente compleja, en lo que a su funcionamiento interno se refiere. Hemos preferido afianzar una correcta y potente implementación de nuestros conocimientos en lugar de centrarnos en la parte estética de la misma. De esta manera, hacemos que sea muy fácil extender el código en el futuro y que quede muy claro cuál es la estructura de nuestro código, el cual podría ser modelizado resumidamente de la siguiente manera:



En el diagrama, se puede contemplar cómo es posible obtener imágenes del servidor desde la ventana principal así como desde la ventana de usuario, esto es porque es posible pedir imágenes al servidor a través del cliente sin haberse autenticado inicialmente, de esta forma, el cliente dará a la interfaz las imágenes tal y como se las ha entregado el servidor, es decir; encriptadas. Cabe destacar, además, que es notable cómo no es posible que ninguna imagen salga del lado del cliente sin encriptar.
Además ha sido necesario implementar algunos patrones de diseño como singleton para las clases de las autoridades.

Mejoras

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
  - 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

- Se ha implementado un sistema de logs en el que tanto cliente como servidor escriben en un archivo SYSTEM.log, aquí registran todos los procesos que ejecutan y si ocurre algún error se registra también. Esto se implementó para facilitar el proceso de debug pero también es muy útil para analizar los distintos procesos que ocurren y así determinar el orden de las operaciones y visualizarlo con más claridad.

- Hemos empleado ambos tipos de cifrado tanto simétrico como asimétrico.

- Nos hemos cerciorado de que la aplicación es realmente segura y estamos convencidos de que una aplicación real, con autoridades reales y estando el cliente y el servidor en hosts diferenciados el sistema sería completamente seguro.

[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.

Anexo

Link al repositorio

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

Otros cambios respecto a la entrega anterior

Hemos realizado cambios en la aplicación para mejorar su seguridad aunque no estuvieran entre los apartados de esta fase.
En la entrega anterior asumimos un canal (sin PITM) seguro entre cliente y servidor, por lo que la clave que genera el hash de la imagen se mandaba sin ningún cifrado. Así como la contraseña y el usuario, que se mandan en cada solicitud de subir imágenes o borrar imágenes, ... (cualquier operación que pueda requerir de autenticación), se mandaban también en texto claro. Esto lo hicimos ya que ya habíamos implementado cifrado con AES y no teníamos tiempo de añadir la segunda capa de cifrado con RSA para compartir claves. En esta segunda entrega esto ha cambiado:

Subida de imágenes:
Antes el cliente generaba un hash de la imagen, y mandaba en claro el hash y la clave usada para generarlo.
Ahora el cliente genera el hash de la imagen, encripta con la publica del servidor la clave que ha usado para generar ese hash, después firma el hash con su privada y adjunta un certificado, el cual el servidor verifica y de él extrae la pública para verificar la firma.

De esta forma es imposible que si alguien intercepta la imagen pueda modificarla y generar un hash y firma nueva.

Proceso de autenticación:
Antes cada vez que te auténticas se enviaba la contraseña y el usuario sin cifrar por el canal, ahora, antes de autenticarte se lee el certificado del servidor y se verifica. Después de verificarlo, se extrae la publica del servidor y se cifra la contraseña con ella.