PRÁCTICA 04

TEORÍA DE CÓDIGOS Y CRIPTOGRAFÍA

APLICACIÓN DE ENCRIPTACIÓN DE ARCHIVOS

Autores
Adrián Jiménez Benítez
Antonio Miguel Almagro Valles
David Montoya Segura
Javier Diaz Vilchez

Asignatura
Teoría de Códigos y Criptografía

Titulación
Grado en Ingeniería Informática

1. Enunciado.

La aplicación debe incluir la siguiente funcionalidad:

1. *Generación de certificados básicos para diferentes usuarios.*
2. *El certificado (básico) debe incluir la identidad del propietario, su clave pública, firma del público clave realizada a través de la clave privada de la autoridad. La autoridad, en este caso será la propia aplicación, cuya clave privada debe almacenarse de forma segura.*
3. *Para almacenar la clave privada de la aplicación utilizaremos un número de licencia. Un hash de esta licencia proporcione una clave AES que cifre la clave privada de la aplicación. Para no hacer un sistema complejo, no es necesario proteger el número de licencia.*
4. *Para cifrar cualquier archivo o carpeta, la aplicación debe pedirle al usuario que seleccione un usuario, cuyo certificado correspondiente deberá obtenerse previamente a partir de un listado.*
5. *La solicitud deberá seleccionar el(los) certificado(s) correspondiente(s), comprobar su validez a través de la clave pública de la aplicación y extraer la clave pública.*
6. *Habilitar la posibilidad de cifrar un fichero o carpeta para varios usuarios.*
7. *Las claves privadas correspondientes de cada usuario deben protegerse mediante una contraseña como en el caso de la clave privada de la aplicación.*
8. *Cuando un usuario autorizado quiere recuperar el archivo o carpeta, se le solicita la contraseña correspondiente.*
9. *Tras introducir la contraseña, se recupera la clave privada y ésta será utilizada para recuperar el archivo o carpeta.*

2. Introducción

La protección de información confidencial es un pilar fundamental en los sistemas modernos, especialmente en entornos donde los datos se transmiten a través de redes no seguras. Para garantizar la seguridad, existen dos tipos principales de cifrado: el simétrico, que emplea una misma clave tanto para cifrar como para descifrar, y el asimétrico, que utiliza un par de claves (una pública y otra privada) para ofrecer mayor flexibilidad y seguridad en la comunicación.

En la práctica anterior, integramos AES (cifrado simétrico) para garantizar un manejo eficiente de grandes volúmenes de datos, y RSA (cifrado asimétrico) para proteger las claves utilizadas por AES, combinando así lo mejor de ambos métodos para ofrecer una solución robusta.

En esta nueva práctica, hemos llevado la implementación un paso más allá al introducir un sistema de generación de certificados para los usuarios registrados en la aplicación. Este mecanismo permite garantizar la autenticidad y confianza en las comunicaciones, alineándose con las instrucciones del ejercicio. En los apartados siguientes, detallaremos el diseño y la implementación de este sistema, junto con los algoritmos RSA y Kyber que refuerzan su seguridad.

3. Manual de usuario

Registro de Usuario

1. Abre la aplicación y selecciona **Registrar**.
2. Ingresa un nombre de usuario y una contraseña.
3. El sistema generará automáticamente:
    - Un par de claves RSA (pública y privada).
    - Un certificado digital que identifica al usuario.
4. Al finalizar, recibirás un mensaje confirmando el registro y el certificado será almacenado en el directorio del usuario.

Inicio de Sesión

1. Selecciona "**Seleccionar Certificado"** para cargar tu certificado.
2. Introduce tu contraseña en el campo correspondiente y presiona "**Login"**.
3. Si la validación es correcta:
    - El sistema cargará tus claves y configurará el entorno de trabajo.
    - Accederás a la ventana principal de la aplicación.

Seleccionar Archivos

1. En la ventana principal, presiona "**Seleccionar Archivos"**.
2. Aparecerá un cuadro de diálogo para elegir uno o varios archivos desde tu sistema.
3. Los archivos seleccionados se mostrarán en la lista correspondiente.

Cifrar Archivos

1. Presiona "**Cifrar Archivo"** en la ventana principal.
2. Selecciona los usuarios para quienes deseas cifrar los archivos:
    - Se mostrará una lista de usuarios disponibles.
    - Agrega o elimina usuarios de la selección.
3. Confirma la selección y el sistema:
    - Cifrará el archivo con AES.
    - Cifrará la clave AES con la clave pública de los usuarios seleccionados.

4. Los archivos cifrados se almacenarán en el directorio del usuario actual.

Descifrar Archivos

1. Selecciona un archivo cifrado presionando "**Seleccionar Archivos"**.
2. Presiona "**Descifrar Archivo"**.
3. Si el archivo incluye una clave cifrada con tu clave pública:
    - El sistema descifrará la clave AES con tu clave privada.
    - Luego descifrará el contenido del archivo.
4. El archivo descifrado se guardará en tu directorio.

Salir de la Aplicación

1. Presiona "**Salir de la Aplicación"** para cerrar el programa de forma segura.
2. Tu sesión será cerrada y tus claves permanecerán protegidas.

Implementación

Gestión de Usuarios

- **login()**
    - **Descripción:** Autentifica a los usuarios mediante un certificado (archivo JSON) y una contraseña.
    - **Invoca:**
        1. **validarUsuario(contraseña)**: Valida el certificado y descifra la clave privada del usuario con su contraseña.
        2. **listarUsuariosEncriptar():** Lista los usuarios registrados disponibles para la encriptación.
- **registrarUsuarioApp(nombre, contraseña)**
    - **Descripción:** Genera claves RSA para el usuario (pública y privada).
    - **Pasos:**
        1. Firma un certificado con la clave privada de la aplicación.
        2. Cifra la clave privada del usuario con AES, usando como clave derivada la contraseña del usuario.
        3. Almacena los certificados y claves en los archivos correspondientes.

Creación de Certificados

- **guardarUsuarioApp(certificate, private_key)**
    - **Descripción**: Guarda el certificado y la clave privada cifrada en el archivo **appCertificates.json.**
- **leerUsuariosApp()**
    - **Descripción**: Carga todos los certificados registrados desde el archivo **appCertificates.json.**
- **validarUsuario(contraseña)**
    - **Descripción**: Valida la firma del certificado con la clave pública de la aplicación y descifra la clave privada del usuario si el certificado es válido.

Cifrado y Descifrado de Archivos

- **cifrar_archivo()**
    - **Descripción**: Presenta una interfaz para seleccionar usuarios destinatarios del cifrado.
    - **Utiliza**:
        - **encriptarArchivosUsers(lista_usuarios_encriptan, lista_rutas_archivos)**: Cifra un archivo usando AES. La clave AES es cifrada con la clave pública RSA de cada usuario seleccionado. Almacena el archivo cifrado junto con las claves cifradas y metadatos (extensión, nombre).
- **descifrar_archivo()**
    - **Descripción**: Recupera la clave AES descifrando la copia cifrada con la clave privada del usuario autenticado.
    - **Llama a:**
        - **desencriptarArchivosUsers(lista_rutas_archivos)**: Descifra el archivo con AES y lo guarda en el directorio de salida.

Generación y Gestión de Claves

- **RSA**:
    - **crear_RSA()**
        - **Descripción**: Genera un par de claves RSA (2048 bits) para el usuario y guarda las claves en el directorio del usuario.

- **Kyber:**
    - **crear_kyber()**
        - **Descripción**: Genera claves públicas y privadas usando Kyber512 (criptografía poscuántica) y guarda las claves en formato .pem.
    - **encriptar_kyber() y desencriptar_kyber(ruta_archivo_cifrado)**
        - **Descripción**: Cifran y descifran un archivo de claves utilizando Kyber.
- **crearClavesAplicacion()**
    - **Descripción**: Genera claves RSA internas para la aplicación y cifra la clave privada de la aplicación con AES usando una clave fija **(clave_aes_app).**
- **encriptarPrivate_app() y desencriptarPrivate_app()**
    - **Descripción**: Manejan el cifrado y descifrado de la clave privada de la aplicación.

Opciones Avanzadas

- **encriptarArchivosUsers(lista_usuarios_encriptan, lista_rutas_archivos)**
    - **Descripción**: Cifra un archivo para múltiples usuarios. Cada usuario obtiene una copia cifrada de la clave AES.
- **desencriptarArchivosUsers(lista_rutas_archivos)**
    - **Descripción**: Permite que cualquier usuario autorizado recupere la clave AES y descifre el archivo.

Selección de Usuarios

- **listarUsuariosEncriptar()**
    - **Descripción**: Obtiene una lista de usuarios registrados que pueden ser seleccionados para cifrar archivos.
- **verificarUsuarioSeleccionado(usuario)**
    - **Descripción**: Valida un usuario seleccionado mediante su certificado y clave pública.

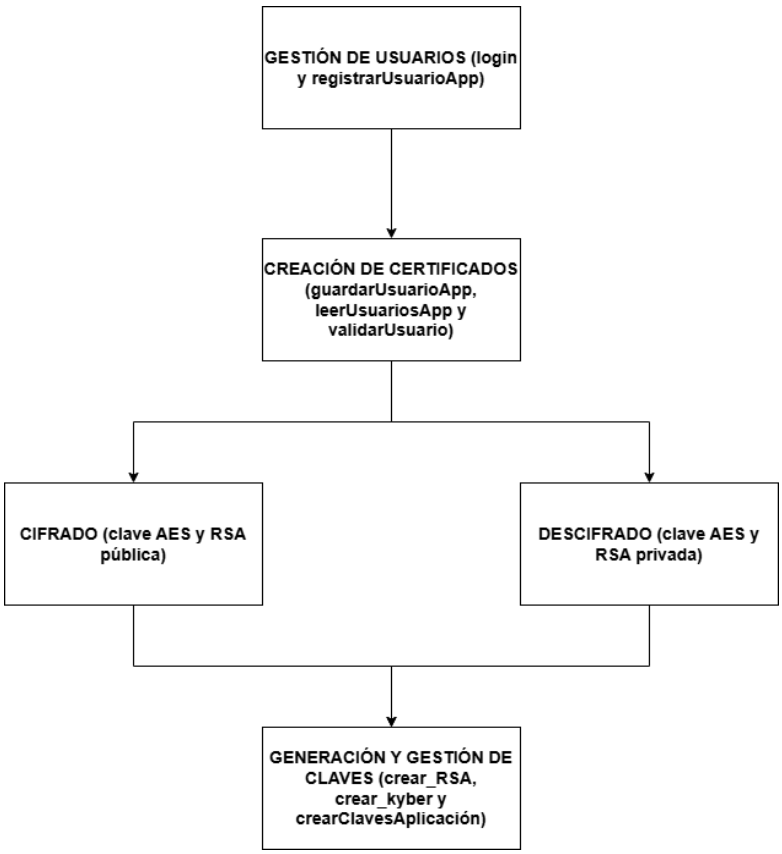
Interfaz Gráfica

- **Ventana de Login**
    - **Descripción**: Permite seleccionar el certificado y proporcionar la contraseña.

- **Ventana Principal**
    - **Funciones disponibles:**
        - Seleccionar archivos para cifrar/descifrar.
        - Cifrar o descifrar usando claves de usuarios registrados.
        - Generar claves o salir de la aplicación.
- **Ventana de Selección de Usuarios**

Descripción: Lista los usuarios disponibles para incluir en el cifrado.

4. Diagrama de bloques



El diagrama de bloques presentado describe los componentes principales de un sistema diseñado para implementar cifrado y descifrado de datos, con funcionalidades adicionales para la gestión de usuarios y la emisión de certificados digitales. A continuación, se detallan cada uno de los bloques y sus funciones:

1. Gestión de Usuarios

Este módulo es responsable de la administración de los usuarios en la aplicación, garantizando un acceso seguro y controlado. Las principales funciones de este bloque son:

- **Inicio de sesión (Login):** Autentica a los usuarios registrados permitiendo el acceso a las funcionalidades del sistema.
- **Registro de usuarios (registrarUsuarioApp):** Permite agregar nuevos usuarios al sistema, registrando la información necesaria, como datos personales y claves asociadas.

2. Creación de Certificados

Este bloque implementa la funcionalidad de generación y gestión de certificados digitales, esenciales para la validación y autenticación de los usuarios dentro del sistema.

- **Guardar información de usuarios (guardarUsuarioApp):** Almacena los datos de los usuarios, incluyendo información relevante para la creación y uso de certificados.
- **Lectura de usuarios registrados (leerUsuariosApp):** Recupera los datos de usuarios previamente registrados, permitiendo el acceso a sus certificados y claves asociadas.
- **Validación de usuarios (validarUsuario):** Comprueba la autenticidad de los usuarios mediante el uso de certificados, asegurando la legitimidad de las operaciones de cifrado y descifrado.

3. Cifrado de Datos

Este bloque implementa los procesos necesarios para proteger la información mediante técnicas de cifrado.

- Se utiliza **AES** como método de cifrado simétrico para garantizar la eficiencia al cifrar archivos.
- Las claves utilizadas por AES se protegen con **RSA**, aprovechando la fortaleza de este algoritmo asimétrico. En este módulo, se emplea la clave pública de RSA para garantizar que solo el destinatario autorizado pueda descifrar los datos.

4. Descifrado de Datos

Este bloque es el inverso al de cifrado y permite recuperar la información original.

- Se emplean las mismas técnicas mencionadas en el cifrado, pero en este caso, utilizando la **clave privada de RSA** para descifrar las claves de AES y, posteriormente, los datos cifrados.
- Este proceso asegura que solo el destinatario autorizado con la clave privada correspondiente pueda acceder a la información.

5. Generación y Gestión de Claves

Este módulo está dedicado a la creación de las claves necesarias para el funcionamiento del sistema.

- **Creación de claves RSA (crear_RSA):** Genera los pares de claves (pública y privada) para el algoritmo RSA, utilizados tanto en cifrado como en descifrado.
- **Creación de claves Kyber (crear_kyber):** Introduce un sistema de claves basado en el algoritmo de cifrado post-cuántico Kyber, proporcionando una alternativa robusta frente a posibles amenazas futuras derivadas de la computación cuántica.
- **Creación de claves para la aplicación (crearClavesAplicación):** Centraliza la gestión de claves necesarias para las diferentes operaciones del sistema.

5. Conclusión.

La integración de firmas digitales junto con los algoritmos AES, RSA, Kyber y Dilithium permite la creación de un sistema de registro y autenticación de usuarios altamente seguro y resistente frente a amenazas presentes y futuras. AES, con su elevado rendimiento, se emplea para el cifrado eficiente de grandes volúmenes de datos, mientras que RSA garantiza una distribución segura de las claves simétricas.

En este contexto, Kyber, como esquema de intercambio de claves basado en criptografía post-cuántica, añade un nivel adicional de seguridad frente a posibles ataques de la computación cuántica. Por su parte, Dilithium, un esquema de firma digital post-cuántico, asegura la integridad y autenticidad de las identidades, reforzando la confianza en el sistema.

Además, cada usuario registrado recibe un certificado digital único, que no solo almacena las claves RSA, sino que también incorpora la flexibilidad y resistencia que ofrecen las tecnologías post-cuánticas. Esta arquitectura no solo garantiza la protección de los datos frente a accesos no autorizados, sino que posiciona al sistema como una solución avanzada que cumple con los estándares más exigentes de la seguridad moderna.

6. Bibliografía.

[1] Cryptography - enlace (03-10-2024)

[2] pyAesCrypt - enlace (03-10-2024)

[3] Fernet - enlace (03-10-2024)

[4] pyCryptodome - enlace (03-10-2024)

[5] Jupyter lab - enlace (04-10-2024)

[6] IEEE Xplore - enlace (04-11-2024).

[7] Practical Networking - enlace (04-11-2024).

[8] Post Quantum Cryptography - enlace (25-11-2024).

[9] Cryptology ePrint Archive - enlace (25-11-2024).

[10] The concepts behind Kyber and Dilithium - enlace (25-11-2024).