

ESTRATEGIAS DE SEGURIDAD

PRÁCTICA 3



Universitat d'Alacant
Universidad de Alicante

INTEGRANTES

- ALEJANDRO MOLINES MOLINA
- AKIRA LLORENS MONTERO
- HUGO HERNÁNDEZ LÓPEZ

CURSO: 2023/2024

Índice

Descripción general de los archivos entregados.....	3
Manual de usuario.....	5
Aplicación de cifrado.....	5
1. Instalación de dependencias.....	5
2. Inicio del Programa.....	5
3. Descripción de la Interfaz y Funcionalidades.....	5
4. Doble factor de autenticación: OTP (One-Time Password).....	15
5. Notas Adicionales.....	16
Cliente Malicioso.....	17
1. Servidor en modo inseguro.....	17
2. Acceso sin iniciar sesión.....	17
3. Descripción de la interfaz y funcionalidades.....	17
Implementación.....	19
Principales funcionalidades.....	19
Cifrado.....	19
Algoritmo de cifrado.....	19
Gestión de ficheros.....	19
Interfaces.....	20
Comunicación Cliente-Servidor.....	20
Compartir ficheros.....	21
Diagramas.....	22
Registro.....	22
Login.....	23
Subida de documentos.....	24
Descarga de documentos.....	24
Glosario de claves.....	26

Descripción general de los archivos entregados

Esta práctica se centra en el desarrollo y manejo de un sistema de archivos seguro y su interfaz gráfica, con la capacidad de encriptar y desencriptar archivos, así como de gestionar el acceso a ellos de manera segura o insegura.

En el directorio principal del proyecto están todos los ficheros, pero se han separado en Servidor y Cliente porque ambos no los necesitan todos. Aunque en el código están todos los ficheros, los ejecutables cuentan únicamente con los ficheros necesarios.

Directorio Servidor:

- **certificates/:** Guarda los certificados y claves SSL para asegurar las comunicaciones.
- **logs/:** (logfile_server.log) Almacena los registros de eventos y errores que ocurren en el servidor, esenciales para la auditoría y depuración del sistema.
- **server/:** Contiene los datos específicos de los usuarios en users.json y los documentos encriptados, keys y json de los archivos que suben los usuarios.
- **sockets/:** Incluye los módulos para la gestión de conexiones y transmisión de datos. A diferencia del Directorio Cliente aquí se encuentra socketServidor.py
- **utils/:** Contiene herramientas de soporte para la operación del servidor, como la generación de claves.

Directorio Cliente:

- **certificates/:** Mantiene los certificados necesarios para la verificación del servidor y la comunicación segura mediante SSL.
- **files/:** Aloja los archivos del cliente que serán enviados al servidor, esta subcarpeta es un repositorio local para cada usuario.
- **interfaces/:** Contiene Interfaz.py para la interacción gráfica con el usuario, y ClienteMalicioso.py, diseñado para descifrar contraseñas inseguras.
- **logs/:** (logfile.log) Almacena los registros de eventos y errores que ocurren con cada cliente, a la hora de ver los eventos ocurridos para una buena auditoría.
- **sockets/:** Gestiona las conexiones desde el punto de vista del cliente, facilitando el intercambio de datos con el servidor. En este directorio se encuentra socketCliente.py.
- **utils/:** Ofrece scripts auxiliares para tareas como la encriptación de archivos.
- **./ClienteInterfaz.py:** Se encarga del registro e inicio de sesión de los usuarios mediante una interfaz.
- **./GetDataUploaded.py:** Gestiona la organización de documentos en el sistema y la recuperación de metadatos.

Además ambos directorios comparten los siguientes ficheros:

- **config.json:** fichero de configuración con información necesaria para la comunicación.
- **requirements.txt:** fichero con los requisitos necesarios para ejecutar la aplicación.
- **Cifrado.py:** contiene los métodos necesarios para el cifrado y descifrado de documentos.

A continuación, se mostrará una estructura de directorios y archivos en forma de lista para que quede ilustrada la distribución en nuestro proyecto, y cómo quedarían los ficheros tanto en el servidor como en el cliente:

```

Practica3/
├── certificates/
│   ├── certificate.pem
│   └── key.pem
├── files_user1/
│   └── document1/
│       ├── document1.zip.enc
│       ├── document1.key
│       └── document1.json
├── interfaces/
│   ├── Interfaz.py
│   └── ClienteMalicioso.py
├── logs/
│   ├── logfile.log
│   └── logfile_server.log
├── server/
│   ├── users.json
│   ├── public_keys.json
│   └── user1/
│       ├── private_key.pem.enc
│       ├── shared/
│       │   └── shared_document1/
│       │       ├── shared_document1.key.enc
│       │       └── shared_document1.json
│       └── document1/
│           ├── document1.zip.enc
│           ├── document1.key.enc
│           └── document1.json
├── sockets/
│   ├── socketPadre.py
│   ├── socketServidor.py
│   └── socketCliente.py
├── utils/
│   └── secure_key_gen.py
├── Cifrado.py
├── ClienteInterfaz.exe
├── ClienteInterfaz.py
├── Servidor.py
├── GetDataUploaded.py
├── requirements.txt
└── config.json

```

```

Servidor/
├── certificates/
│   ├── certificate.pem
│   └── key.pem
├── logs/
│   └── logfile_server.log
├── server/
│   ├── users.json
│   ├── public_keys.json
│   └── user1/
│       ├── private_key.pem.enc
│       ├── shared/
│       │   └── shared_document1/
│       │       ├── shared_document1.key.enc
│       │       └── shared_document1.json
│       └── document1/
│           ├── document1.zip.enc
│           ├── document1.key.enc
│           └── document1.json
├── sockets/
│   ├── socketPadre.py
│   └── socketServidor.py
├── utils/
│   └── secure_key_gen.py
├── Cifrado.py
├── Servidor.py
├── GetDataUploaded.py
├── requirements.txt
└── config.json

```

```

Cliente/
├── certificates/
│   └── certificate.pem
├── files_user1/
│   └── document1/
│       ├── document1.zip.enc
│       ├── document1.key
│       └── document1.json
├── interfaces/
│   ├── Interfaz.py
│   └── ClienteMalicioso.py
├── logs/
│   └── logfile.log
├── sockets/
│   ├── socketPadre.py
│   └── socketCliente.py
├── utils/
│   └── secure_key_gen.py
├── Cifrado.py
├── ClienteInterfaz.exe
├── ClienteInterfaz.py
├── GetDataUploaded.py
├── requirements.txt
└── config.json

```

Manual de usuario

Aplicación de cifrado

1. Instalación de dependencias

Antes de iniciar el programa, asegúrate de tener instaladas las librerías necesarias. Si aún no las has instalado, puedes hacerlo ejecutando el siguiente comando en tu terminal:

```
pip install -r requirements.txt
```

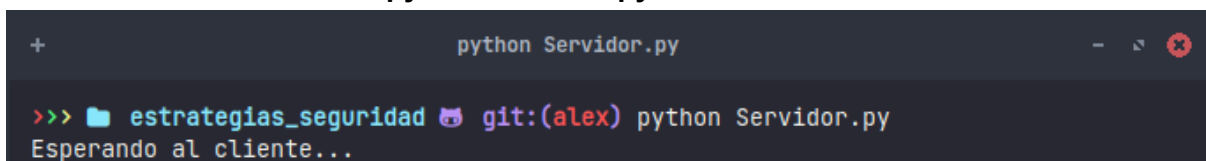
NOTA: Los ejecutables son *standalone*, por lo que en caso de utilizarlos no es necesario instalar las dependencias porque vienen ya incluidas.

2. Inicio del Programa

Mediante terminal:

Para iniciar el programa correctamente, debes situarte en la carpeta raíz del programa, que es donde se encuentran los archivos `ClienteInterfaz.py` y `Servidor.py`. Puedes hacerlo navegando a través de la terminal o ubicándote directamente en el directorio y abriendo el terminal en esa ubicación.

1. `python Servidor.py`



```
+ python Servidor.py
>>> estrategias_seguridad git:(alex) python Servidor.py
Esperando al cliente...
```

2. `python ClienteInterfaz.py`

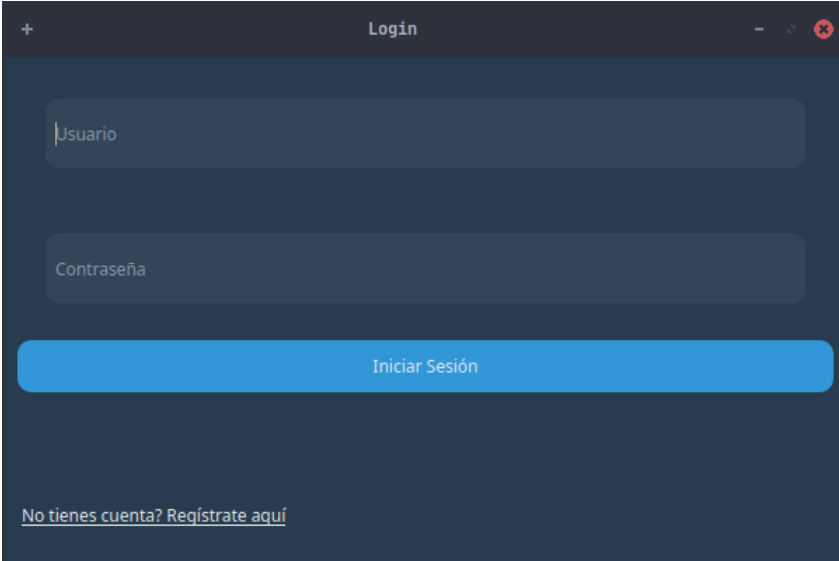
Mediante ejecutable:

También es posible iniciar el programa mediante el ejecutable. El del cliente se llama `ClienteInterfaz.exe`, y viene únicamente el fichero ejecutable. El ejecutable del servidor es `Servidor.exe` y viene en un zip donde se pueden ver todos los archivos y directorios que utiliza.

3. Descripción de la Interfaz y Funcionalidades

- **Inicio de Sesión en la Interfaz del Cliente:** Al ejecutar `ClienteInterfaz.py`, el usuario se encontrará con una interfaz gráfica intuitiva y accesible, diseñada para facilitar el proceso de autenticación y acceso al sistema. Esta pantalla inicial es el punto de entrada al servicio, donde se solicitan las credenciales de acceso para verificar la identidad del usuario.

- **Campos de Texto:** La ventana de inicio de sesión dispone de dos campos de texto esenciales:
 - **Nombre de Usuario:** Aquí el usuario debe introducir su identificador único dentro del sistema.
 - **Contraseña:** Este campo está destinado a la clave secreta asociada al nombre de usuario. Por seguridad, el texto introducido aparecerá oculto para evitar la exposición de la contraseña.
- **Botón de Iniciar Sesión:** Una vez ingresadas las credenciales, el usuario puede pulsar este botón para intentar acceder al sistema. El sistema verificará la información proporcionada; si las credenciales son correctas, se concederá el acceso. En caso contrario, se mostrará un mensaje de error indicando que el nombre de usuario o la contraseña son incorrectos.
- **Texto de Registro:** Debajo del botón de iniciar sesión, este elemento actúa como un enlace para aquellos usuarios que aún no disponen de una cuenta en el sistema. Al hacer clic sobre el texto se abrirá la ventana de registro, donde los nuevos usuarios podrán crear una cuenta siguiendo un proceso sencillo y seguro.



- **Registro en la Interfaz del Cliente:** La ventana de registro es un componente esencial de nuestro sistema, diseñada para facilitar a los usuarios la creación de cuentas seguras y personalizadas para acceder al servidor. Con el objetivo de garantizar la seguridad de la información y la integridad del sistema, se han establecido criterios específicos para la creación de credenciales de usuario.
 - **Nombre de Usuario:** El nombre de usuario debe contener al menos 5 caracteres.

- **Contraseña:** La contraseña constituye una barrera crítica de seguridad para la protección de la cuenta de usuario. Por ello, debe tener un mínimo de 8 caracteres.
- **Generador de Contraseñas:** Esta herramienta ofrece varias herramientas desde la longitud de la contraseña hasta la inclusión de símbolos, permitiendo a los usuarios generar automáticamente una contraseña que cumpla con los estándares de seguridad establecidos.

Registro

Nuevo Usuario

Nueva Contraseña

☒ Autenticación OTP

Registrar

50%

☒ Incluir mayúsculas

☒ Incluir minúsculas

☒ Incluir números

☒ Incluir símbolos

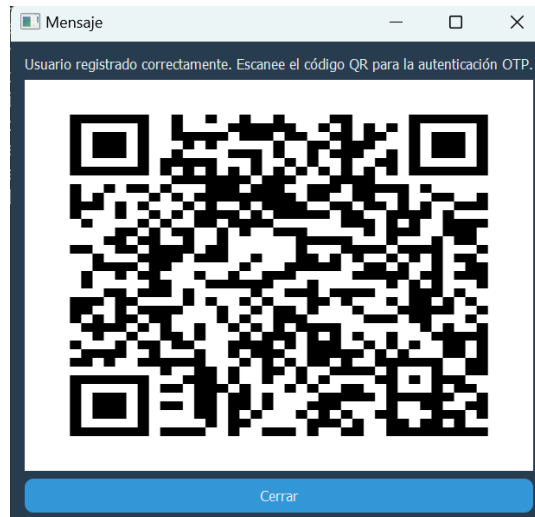
☐ Fácil de leer

☐ Fácil de decir

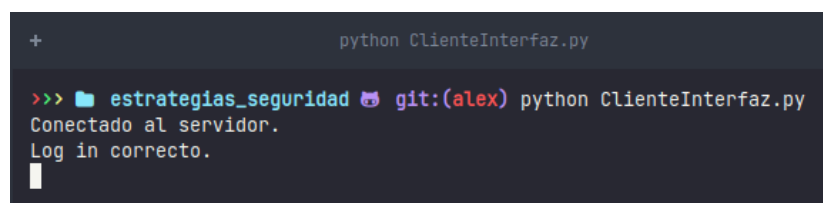
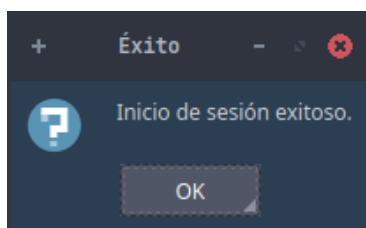
☒ Manual

Generar Contraseña

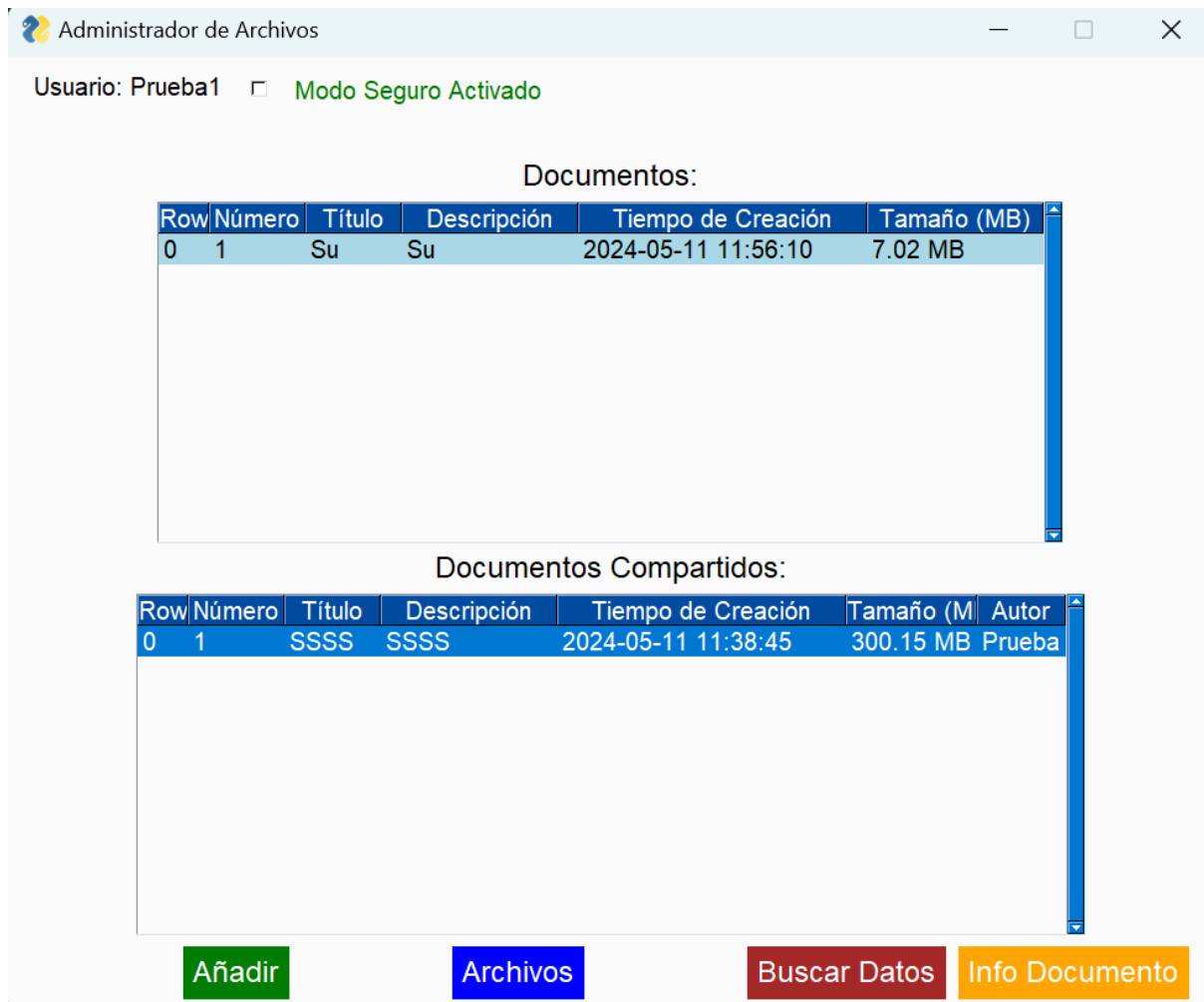
- **Autenticación OTP:** Al seleccionar esta opción, al darle al botón de **Registrar** aparecerá un código QR, el cual su función es la de doble factor de autenticación. Este apartado está mejor explicado en el **punto 4**.



Tras completar los campos de nombre de usuario y contraseña, y satisfacer los criterios de validación, el usuario puede proceder con el registro de su cuenta. Finalmente, una vez validadas y aceptadas las credenciales, el usuario puede completar el registro y acceder al servidor con su nueva cuenta.



- **Interfaz:** La interfaz de usuario se compone de varios elementos interactivos diseñados para ofrecer una experiencia de usuario fluida y funcional. Esta ventana sirve como interfaz de navegación y punto de interacción para el usuario. Además, inmediatamente después de la inicialización, la interfaz intenta establecer una conexión con el servidor.



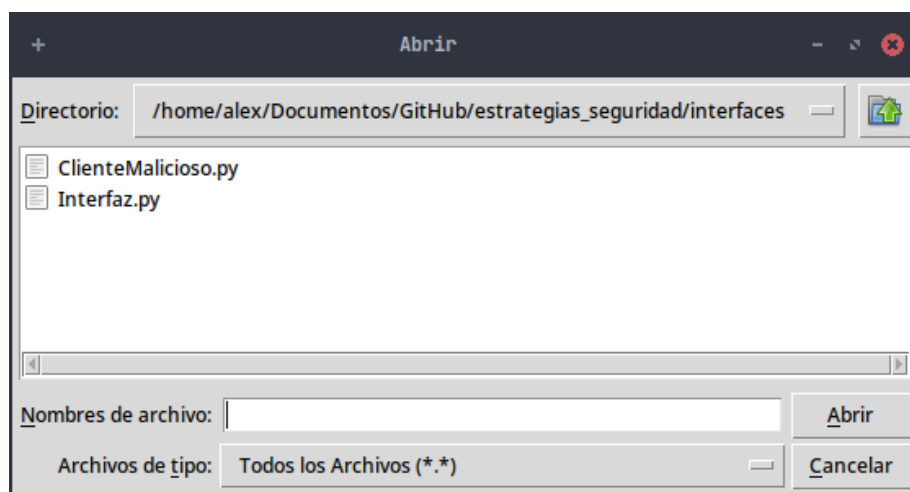
- **Modo Seguro/Inseguro:** Hay un checkbox que permite alternar entre el modo seguro y el modo inseguro. Cambiar esta opción actualizará el estado mostrado justo al lado, indicando si el modo inseguro está activado o no.



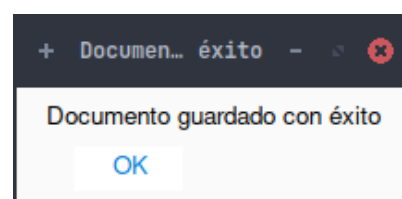
- **Botón 'Añadir':** Al hacer clic en este botón, se abre una nueva ventana que permite ingresar los detalles de un nuevo archivo para ser procesado. Debes proporcionar un título, una descripción y seleccionar los archivos que deseas comprimir y cifrar.
 - **Ventana de Añadir Nuevo Archivo:** Al darle al botón de "Añadir" en la ventana principal, se abre una nueva ventana con varias opciones para ingresar la información del archivo que deseas procesar.



- **Título:** Campo de texto para ingresar el título del nuevo archivo.
- **Descripción:** Espacio para ingresar una descripción del archivo.
- **Archivos:** Aquí puedes ingresar la ruta del archivo que deseas añadir o utilizar el botón '**Buscar**' para abrir un explorador de archivos y seleccionar los archivos manualmente. Puedes seleccionar tanto un solo archivo como varios.
- **Compartir:** Aunque no aparece en la captura de pantalla, si en el servidor hay más usuarios registrados, permitirá seleccionarlos en el caso de que se quiera compartir con esos usuarios mediante un checkbox.



- **Botón 'Guardar':** Después de llenar toda la información requerida y seleccionar los archivos, haciendo clic en este botón ahora enviará los archivos cifrados al servidor, organizados por nombre de usuario. Este proceso incluye compresión, cifrado, y transmisión segura, con una confirmación de éxito o un mensaje de error en caso de problemas.



Añadir Nuevo Archivo

Título

Descripción

Archivos

```

server
  aaaaa / File79b6326d-d588-4...
    {} File79b6326d-d588-4e60-99...
    ≡ File79b6326d-d588-4e60-99...
    ≡ File79b6326d-d588-4e60-99...

```

```

python ClienteInterfaz.py

>>> estrategias_seguridad git:(alex) python ClienteInterfaz.py
Conectado al servidor.
Log in correcto.
Enviando archivos...
File79b6326d-d588-4e60-99f0-da336711506b
/home/alex/Documentos/GitHub/estrategias_seguridad
b'\xa9^\x99\xa1\xa0e"\n;\xd9\xa77\x8d4\x19['
Data key provided
data_key: 30e36724313b7e3d9fff50e3772f769d

```

```

python Servidor.py

127.0.0.1:47390 conectado.
Active threads: 2
Esperando opción...
Iniciando sesion...
Esperando al cliente...
Esperando el tamaño del nombre del archivo...
Nombre de archivo recibido: File79b6326d-d588-4e60-99f0-da336711506b.zip.enc
Recibiendo archivo...
Archivo recibido correctamente.
Archivo recibido.
Esperando el tamaño del nombre del archivo...
Nombre de archivo recibido: File79b6326d-d588-4e60-99f0-da336711506b.key.enc
Recibiendo archivo...
Archivo recibido correctamente.
Archivo recibido.
Esperando el tamaño del nombre del archivo...
Nombre de archivo recibido: File79b6326d-d588-4e60-99f0-da336711506b.json
Recibiendo archivo...
Archivo recibido correctamente.
Archivo recibido.
Esperando el tamaño del nombre del archivo...
Everything sent correctly
Archivos recibidos correctamente.

```

- **Botón 'Archivos':** Este botón permite visualizar los archivos actualmente procesados. Al seleccionar un archivo de la lista y hacer clic en este botón, puedes ver y descargar los archivos individuales contenidos dentro del archivo comprimido seleccionado.
- **Botón 'Buscar Datos':** Al hacer clic en este botón, el programa intenta cargar los datos de los archivos previamente procesados para cada usuario, ya sean los documentos propios como los que otros usuarios han querido compartir con el. Muestra un mensaje de carga y, una vez completada la carga, actualiza la tabla con la información de los archivos. Si hay algún error en la carga, se mostrará una notificación.

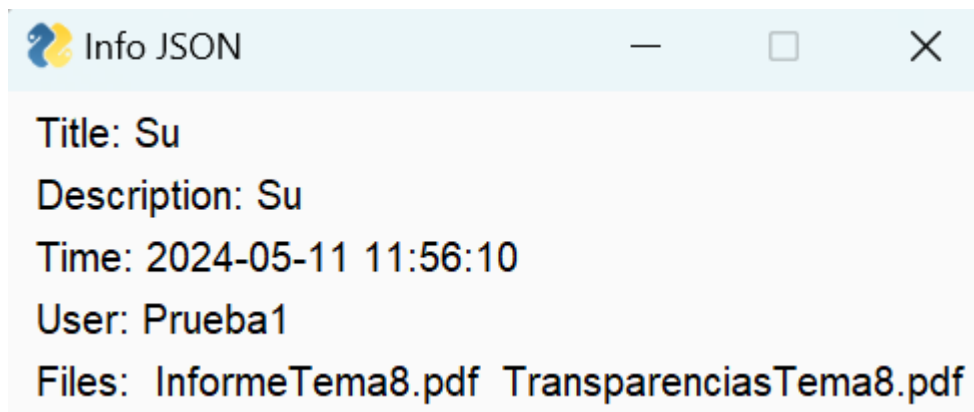
Documentos:						
Row	Número	Título	Descripción	Tiempo de Creación	Tamaño (MB)	
0	1	Su	Su	2024-05-11 11:56:10	7.02 MB	
Documentos Compartidos:						
Row	Número	Título	Descripción	Tiempo de Creación	Tamaño (M)	Autor
0	1	SSSS	SSSS	2024-05-11 11:38:45	300.15 MB	Prueba

```

PS C:\Users\34634\Desktop\UA\3º\CUATRIMESTRE 2\ES\PRÁCTICAS\estrategias_seguridad> python .\ClienteInterfaz.py
Conectado al servidor.
Log in correcto.
False
Recibiendo archivo... private_key.pem.enc
Guardando archivo en: files_Prueba\private_key.pem.enc
1674
Conexion abierta: <ssl.SSLSocket fd=1160, family=2, type=1, proto=0, laddr=('127.0.0.1', 50802), raddr=('127.0.0.1', 65535)>
Response: OTP_false
Recibiendo JSON
Sharing en wait_files: False
Esperando el tamaño del nombre del archivo...
NameSize: 45
Nombre de archivo recibido: File29835a98-bfac-4f63-a489-27c130fa3d14.json
FileNameBeforeCreate: File29835a98-bfac-4f63-a489-27c130fa3d14.json
Sharing: False
folderAfterCreate: files_Prueba\File29835a98-bfac-4f63-a489-27c130fa3d14
self.Folder: files_Prueba
Recibiendo archivo...
filename: File29835a98-bfac-4f63-a489-27c130fa3d14.json
Archivo recibido correctamente.
Archivo recibido.

```

- **Botón 'Info Documento':** Brinda información detallada sobre el archivo seleccionado, incluyendo metadatos como título, descripción, fecha de creación, usuario creador y los ficheros que lo componen..

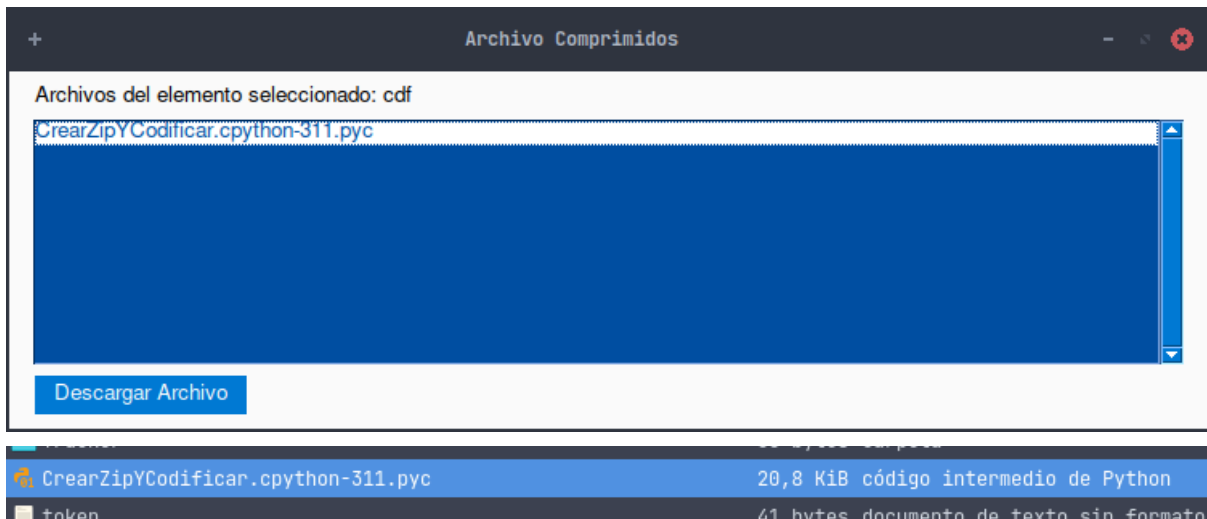


- **Tabla de Archivos:** Presenta una lista de los archivos que se han procesado, mostrando detalles como el número, título, descripción y el tiempo de creación. Hay un total de dos tablas de archivos: la de los propios **documentos** y la de los **documentos compartidos** por otros usuarios. Ambas tablas tienen las mismas columnas con la misma información exceptuando por que la de **Documentos Compartidos** contiene la columna de **Autor** que enseñará qué usuario ha querido compartir el documento con él.

Documentos:					
Row	Número	Título	Descripción	Tiempo de Creación	Tamaño (MB)
0	1	d	d	2024-05-11 11:39:58	0.16 MB
1	2	ss	ss	2024-05-11 11:39:25	57.81 MB
2	3	SSSS	SSSS	2024-05-11 11:38:45	300.15 MB
3	4	s	s	2024-05-09 18:27:05	927.63 KB

Documentos Compartidos:						
Row	Número	Título	Descripción	Tiempo de Creación	Tamaño (MB)	Autor

- **Descargar Archivo:** Al darle al botón de ‘Archivos’, se visualizarán los archivos dentro del archivo comprimido seleccionado, permitiendo seleccionar y descargar los archivos individuales. Antes de descargar los ficheros, se descifran utilizando la clave almacenada en el fichero .key en la carpeta del documento.

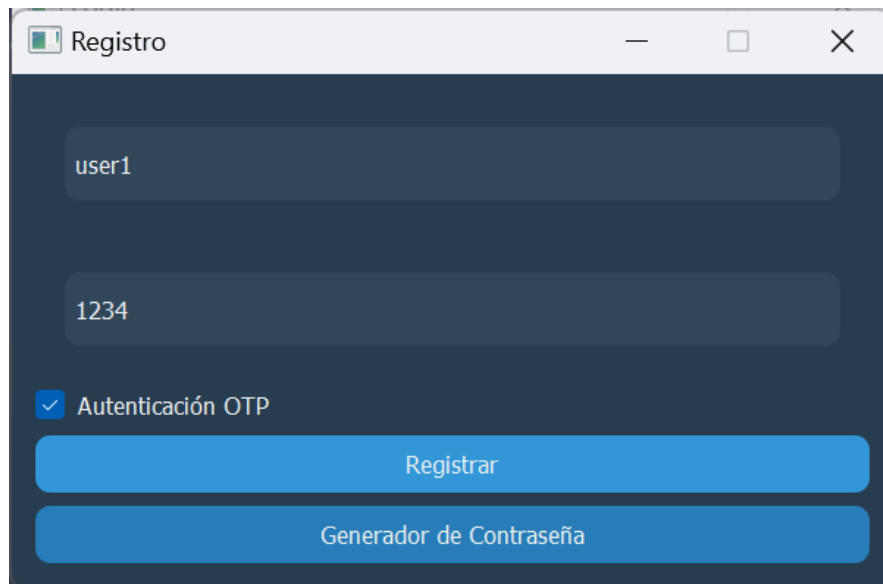


```
python ClienteInterfaz.py

Esperando el tamaño del nombre del archivo...
Nombre de archivo recibido: File79b6326d-d588-4e60-99f0-da336711506b.json
Recibiendo archivo...
filename: File79b6326d-d588-4e60-99f0-da336711506b.json
Archivo recibido correctamente.
Archivo recibido.
Esperando el tamaño del nombre del archivo...
Nombre de archivo recibido: File79b6326d-d588-4e60-99f0-da336711506b.key.enc
Recibiendo archivo...
filename: File79b6326d-d588-4e60-99f0-da336711506b.key.enc
data_key: 30e36724313b7e3d9fff50e3772f769d
Archivo recibido correctamente.
Archivo recibido.
Esperando el tamaño del nombre del archivo...
Everything sent correctly
Archivos recibidos correctamente.
file_path: files/File79b6326d-d588-4e60-99f0-da336711506b/File79b6326d-d588-4e60-99f0-da336711506b.zip.enc
```

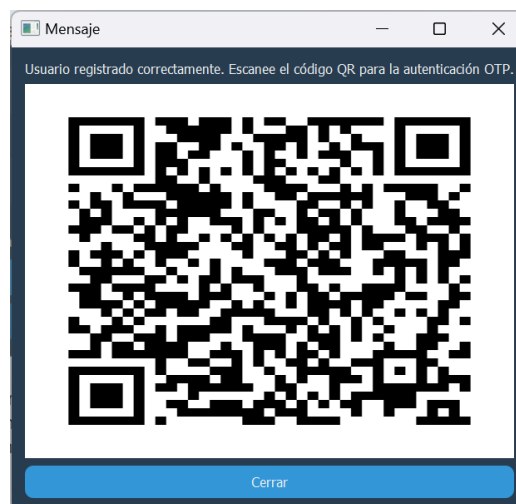
4. Doble factor de autenticación: OTP (One-Time Password)

Al momento de registrar un usuario, es posible seleccionar la opción *Autenticación OTP* si queremos activar el doble factor de autenticación.



A screenshot of a web application window titled "Registro". It features two input fields: the first contains "user1" and the second contains "1234". Below these fields is a checkbox labeled "Autenticación OTP" which is checked. At the bottom, there are two blue buttons: "Registrar" and "Generador de Contraseña".

Una vez registrado el usuario se nos proporcionará un código QR que tendremos que escanear con una aplicación de tokens OTP, por ejemplo: Google Authenticator. Si perdemos este código no seremos capaces de iniciar sesión.



A partir de este momento, cada vez que iniciemos sesión la aplicación nos pedirá un token temporal que recibiremos en la aplicación de autenticación.



A screenshot of a web application window titled "OTP". It features a single input field labeled "OTP" and a blue button labeled "Iniciar Sesión" at the bottom.

5. Notas Adicionales

- El script de 'Interfaz.py' tiene tres dependencias con dos módulos externos (ZipFile y is_unsafe_mode de Cifrado.py', y gdu de 'GetDataUploaded.py') hay que asegurarse de que estén correctamente configurados y accesibles desde este script.
- El programa utiliza **hilos** (threading) para la carga de datos, lo que permite que la interfaz gráfica siga en funcionamiento durante operaciones que puedan tomar tiempo.
- Es importante que cuando estés trabajando con el programa no cierres la ventana principal mientras realizas operaciones en ventanas secundarias, ya que esto podría interrumpir el proceso y cerrar el programa.
- Se aplican patrones de diseño de software, como MVC (Modelo-Vista-Controlador) para separar la lógica de la interfaz de usuario de la lógica de negocio, promoviendo un código más organizado y mantenible.
- Se ha utilizado Git como sistema de control de versiones para mantener un registro de los cambios, permitiendo la colaboración y facilitando la integración continua.
- Los ejecutables se han generado con el programa *pyinstaller*, con el comando para el cliente:

```
PyInstaller -F -w ClienteInterfaz.py
--add-data=./certificates/certificate.pem:./certificates
--add-data=./interfaces/Interfaz.py:./interfaces --add-data=./logs/*:./logs
--add-data=./sockets/*:./sockets --add-data=./utils/secure_key_gen.py:./utils
--add-data=Cifrado.py:. --add-data=config.json:. --add-data=GetDataUploaded.py:.
```

Opción -F para generar el ejecutable en un solo fichero.

Opción -w para no tener terminal.

La opción --runtime-tmpdir selecciona un directorio para almacenar el directorio temporal de ejecución, por defecto lo almacena en un directorio temporal del SO.

- El fichero *requirements.txt* se ha generado con los programas *pipreqs* y *pip-tools*, con el comando:

```
pipreqs --savepath=requirements.in && pip-compile
```


Cliente Malicioso

1. Servidor en modo inseguro

Para permitir el acceso a los documentos sin necesidad de iniciar sesión hay que iniciar el servidor en modo inseguro. Para ello tendremos que seleccionar la opción de inicio inseguro al iniciar la aplicación. Si esta opción no está habilitada, será imposible acceder a documentos que no sean de nuestra propiedad.

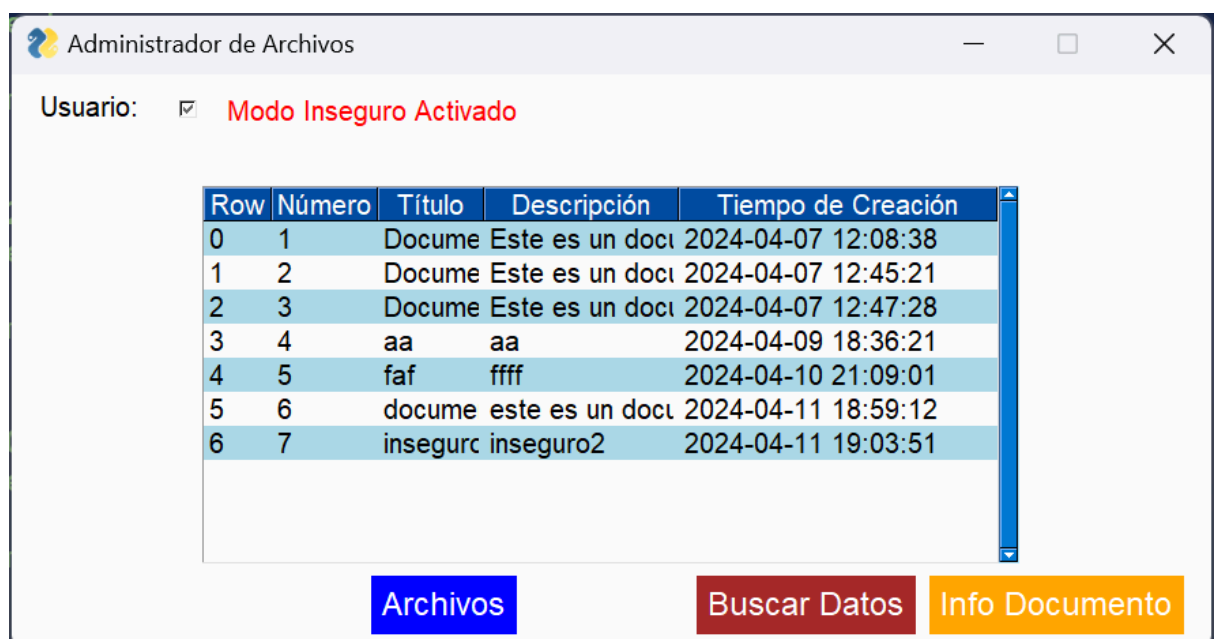
```
Folder: C:\AKIRA\UNI\4\semestre_2\ES\en
Habilitar modo inseguro? (s/N): s
HABILITADO MODO INSEGURO
Esperando al cliente...
```

2. Acceso sin iniciar sesión

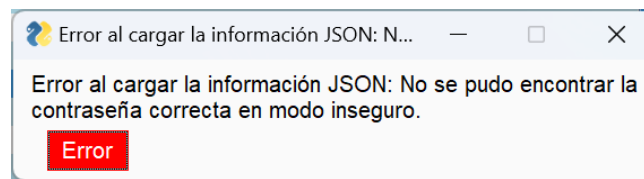
Si el servidor tiene habilitado el modo inseguro, aún será posible acceder iniciando sesión en un usuario concreto de la manera habitual. Para acceder siendo un cliente malicioso habrá que iniciar sesión sin introducir ningún usuario ni contraseña.

3. Descripción de la interfaz y funcionalidades

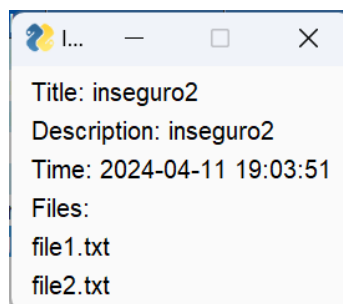
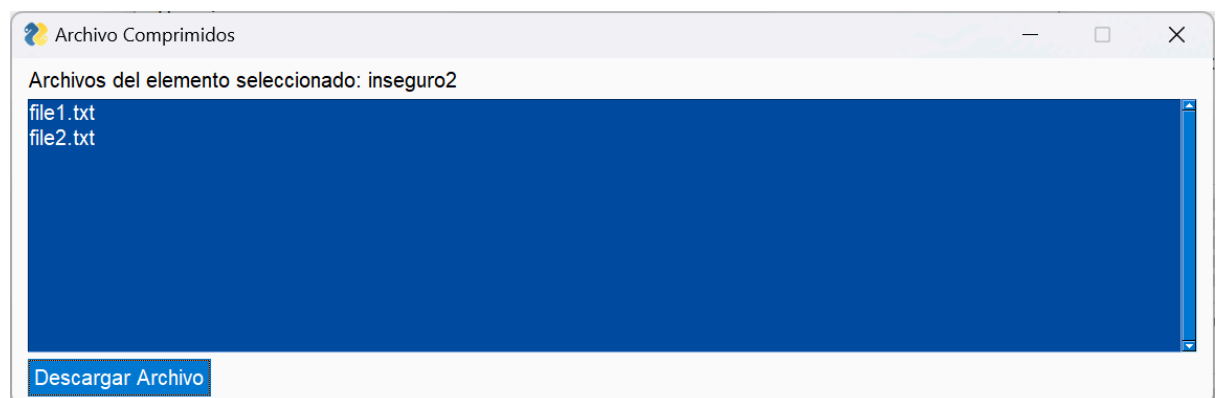
A continuación vamos a explicar el funcionamiento de la aplicación en modo inseguro. Una vez iniciado el cliente en modo inseguro podremos ver que no existe un botón para subir documentos al servidor. Además, si buscamos documentos existentes nos aparecerán todos los documentos del servidor, pertenecientes a cualquier usuario.



En caso de intentar obtener la información de un documento cifrado de forma segura, nos aparecerá el siguiente mensaje de error:



Si el documento fue cifrado con una contraseña insegura, podremos ver su información y descargar los ficheros que contiene:



Será posible descargar cada fichero individualmente o varios al mismo tiempo. Después de descargarse los ficheros se descifrarán y se descomprimirán automáticamente, no será necesario hacerlo manualmente después.

Implementación

Principales funcionalidades

Cifrado

El cifrado se realiza en el fichero Cifrado.py. En este fichero se realiza también la compresión de los ficheros a cifrar y el cifrado de la información sensible en los JSON.

Las funciones más relevantes son:

- **encrypt_file**: esta función cifra un fichero y lo almacena en el directorio especificado. Para el cifrado genera una clave aleatoria que almacena junto a los ficheros cifrados en un archivo con extensión .key.
- **decrypt_file**: esta función descifra un fichero. Se puede pasar la clave de cifrado como parámetro, lo cual es útil para utilizar en el cliente malicioso. Si no se le pasa la clave, esta se obtendrá del fichero .key junto al fichero encriptado.
- **decrypt_file_unsafe**: esta función prueba con las diferentes claves almacenadas hasta que una es la correcta o no consigue descifrarlo. Esta función utiliza la función anterior para descifrar los ficheros.
- **encrypt_file_asimetric**: esta función cifra un fichero con RSA. Se utiliza para cifrar las claves de fichero.
- **decrypt_file_asimetric**: esta función descifra un fichero con RSA. Se utiliza para descifrar las claves de fichero.

Algoritmo de cifrado

Para la realización de esta práctica se ha utilizado el algoritmo de cifrado AES (Advanced Encryption Standard) 256 con el modo CTR. Se ha escogido este modo sobre otros porque puede utilizar paralelización, además de no requerir padding, por lo que es seguro contra ciertos ataques criptográficos que utilicen el padding. Además, en este modo el cifrado de un bloque depende de su posición, por lo que este modo es más seguro ante ciertos ataques que buscan patrones en el cifrado.

Gestión de ficheros

En el fichero GetDataUploaded.py se realizan funciones como por ejemplo listar los documentos existentes, obtener datos de los JSON o listar los ficheros de un documento cifrado a partir de la información del JSON.

Interfaces

Nuestro proyecto cuenta con 3 interfaces diferentes, dos de estas se encuentran en el directorio interfaces/ dentro de la raíz del proyecto, y la última interfaz se encuentra en la raíz del proyecto

Las interfaces disponibles son:

- interfaces/Interfaz.py
 - Es la interfaz del programa principal de cifrado y descifrado.
- interfaces/ClienteMalicioso.py
 - Es la interfaz del cliente malicioso que permite descifrar ficheros cifrados con claves inseguras.
- ./ClienteInterfaz.py
 - Es la interfaz que asegura un registro e inicio de sesión seguros, permitiendo el acceso al servidor sólo tras una correcta autenticación.

Para la creación de las interfaces se ha utilizado la librería PySimpleGUI y PyQt5.

Desde las interfaces se llama a las funciones correspondientes de los ficheros Cifrado.py, secure_key_gen.py que se encargan de realizar las acciones necesarias según la interacción del usuario con la interfaz.

Comunicación Cliente-Servidor

Para la comunicación entre el cliente y el servidor se han utilizado sockets. Los sockets, por defecto, no cifran la información que se envía por ellos, por lo que no son seguros. Para solventar este fallo de seguridad se ha utilizado el protocolo TLS (Transport Layer Security) de la librería ssl. Este protocolo se encarga de cifrar los datos con un par de certificados de clave pública y privada generados previamente. El servidor dispone de ambos certificados mientras que el cliente dispone únicamente del certificado público.

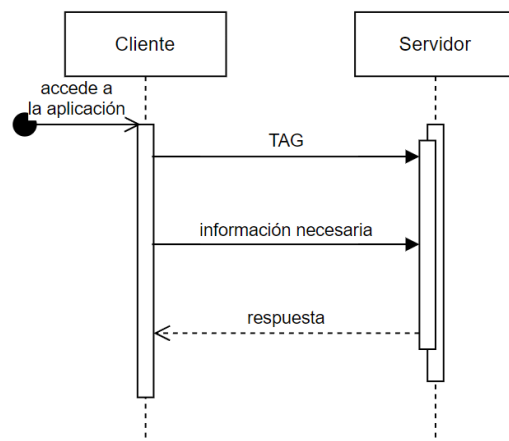
Una vez que el cliente ha iniciado sesión, la conexión de los sockets no se cierra hasta que finaliza la comunicación. De esta forma podemos asegurarnos de que el cliente con el que se comunica el servidor es el mismo que inició sesión al principio.

La secuencia de comunicación es la siguiente:

El TAG indica al servidor cuál es la operación que debe realizar. Por ejemplo, REG para registrar un usuario o LOG para iniciar sesión.

La información necesaria para realizar la operación puede ir en uno o varios mensajes. Por ejemplo, para el registro se enviará un mensaje con el nombre de usuario y otro con la clave de login.

La respuesta le indica al usuario cómo ha ido la operación.



Compartir ficheros

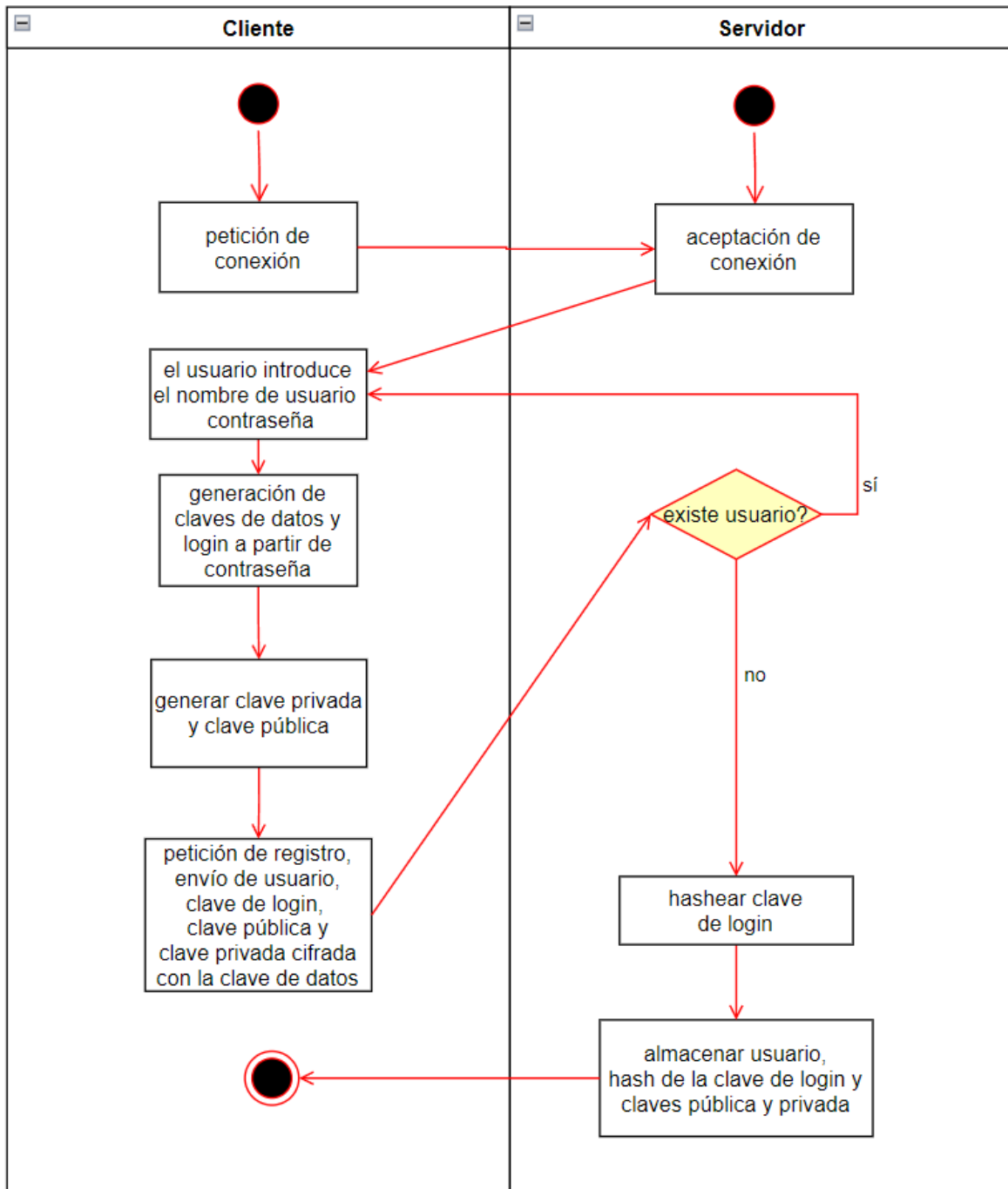
Para poder compartir ficheros se utiliza un cifrado de clave pública y privada, concretamente RSA.

Los documentos se cifran con la clave de fichero generada aleatoriamente. Después la clave de fichero se cifra, pero no con la clave de datos del usuario. Se genera una copia de la clave de fichero por cada usuario con el que se quiere compartir el documento, incluido el usuario que ha creado el documento, y cada una de las copias se cifra con la clave pública de cada usuario.

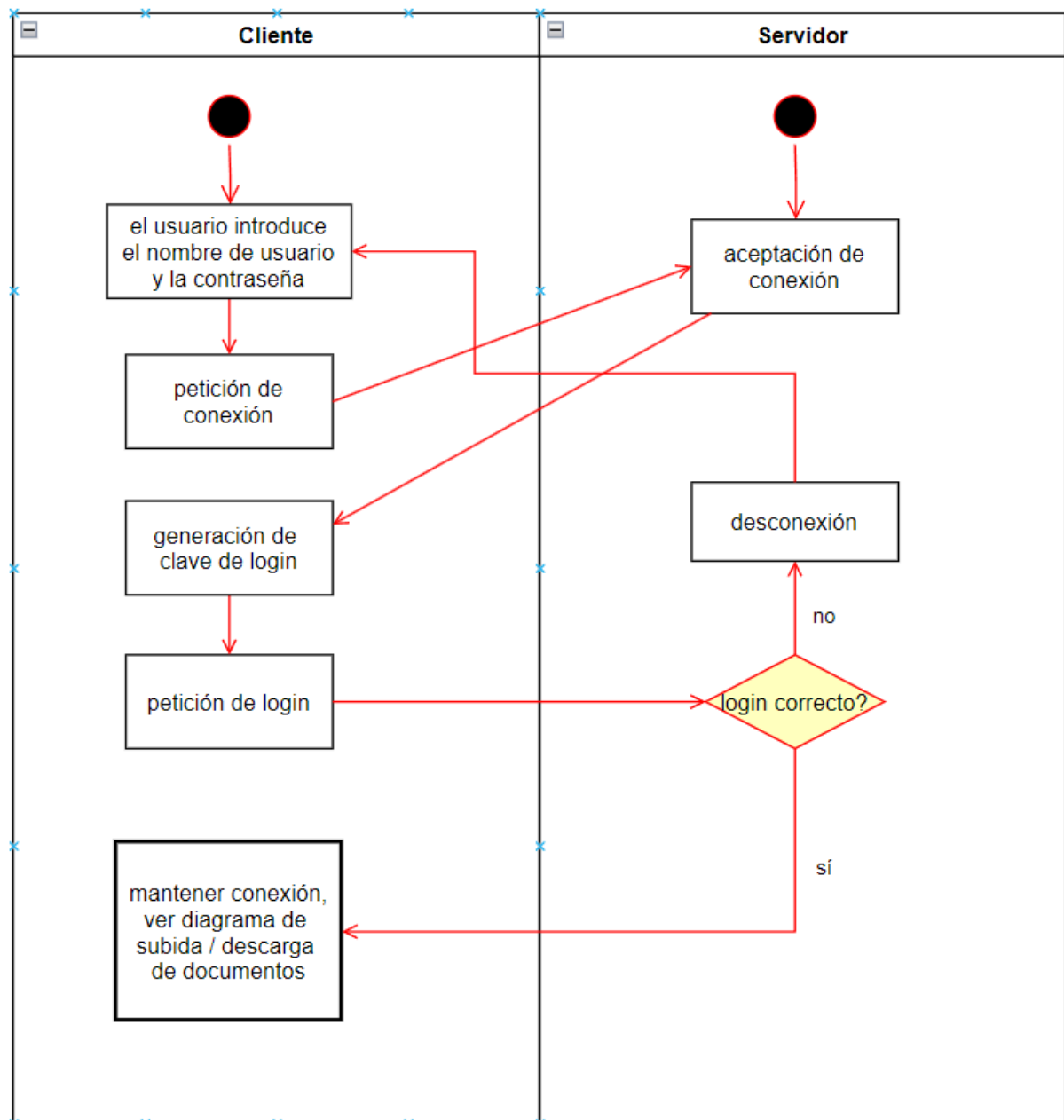
Una vez enviados los ficheros al servidor, se almacena el documento en el directorio del usuario que lo ha creado y se reparten las claves a cada usuario con los que se ha compartido el documento. De esta forma cada usuario podrá descifrar el documento con la clave que se le ha proporcionado, que será la misma que la original pero cifrada con su clave pública. Al estar cifrada la clave con la clave pública de cada usuario solo se podrá descifrar con la clave privada del mismo usuario. Además, esta clave privada está cifrada con la clave de datos del usuario, que nunca sale del cliente, por lo que no es posible descifrar los documentos si no eres un usuario con permisos sobre ese documento.

Diagramas

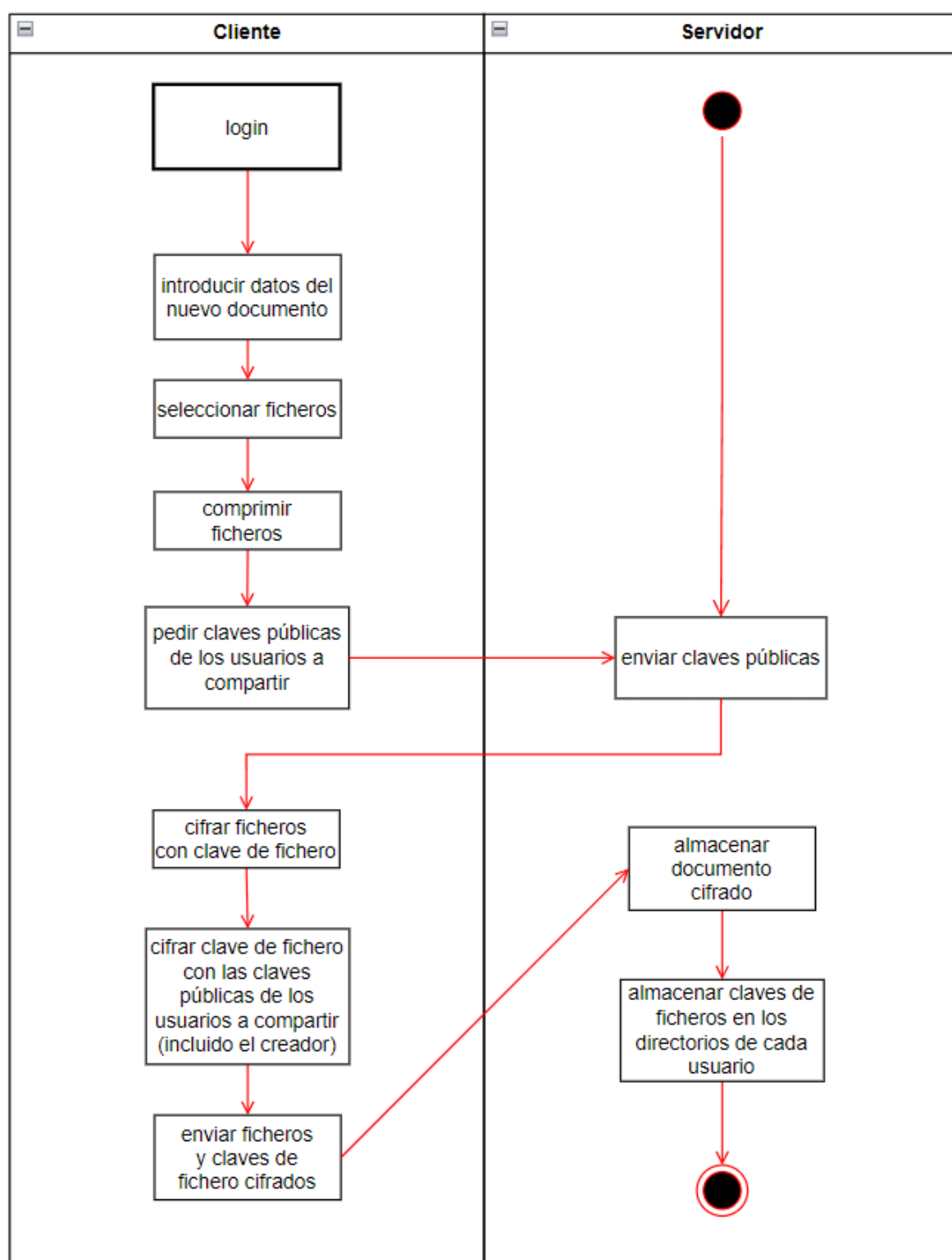
Registro



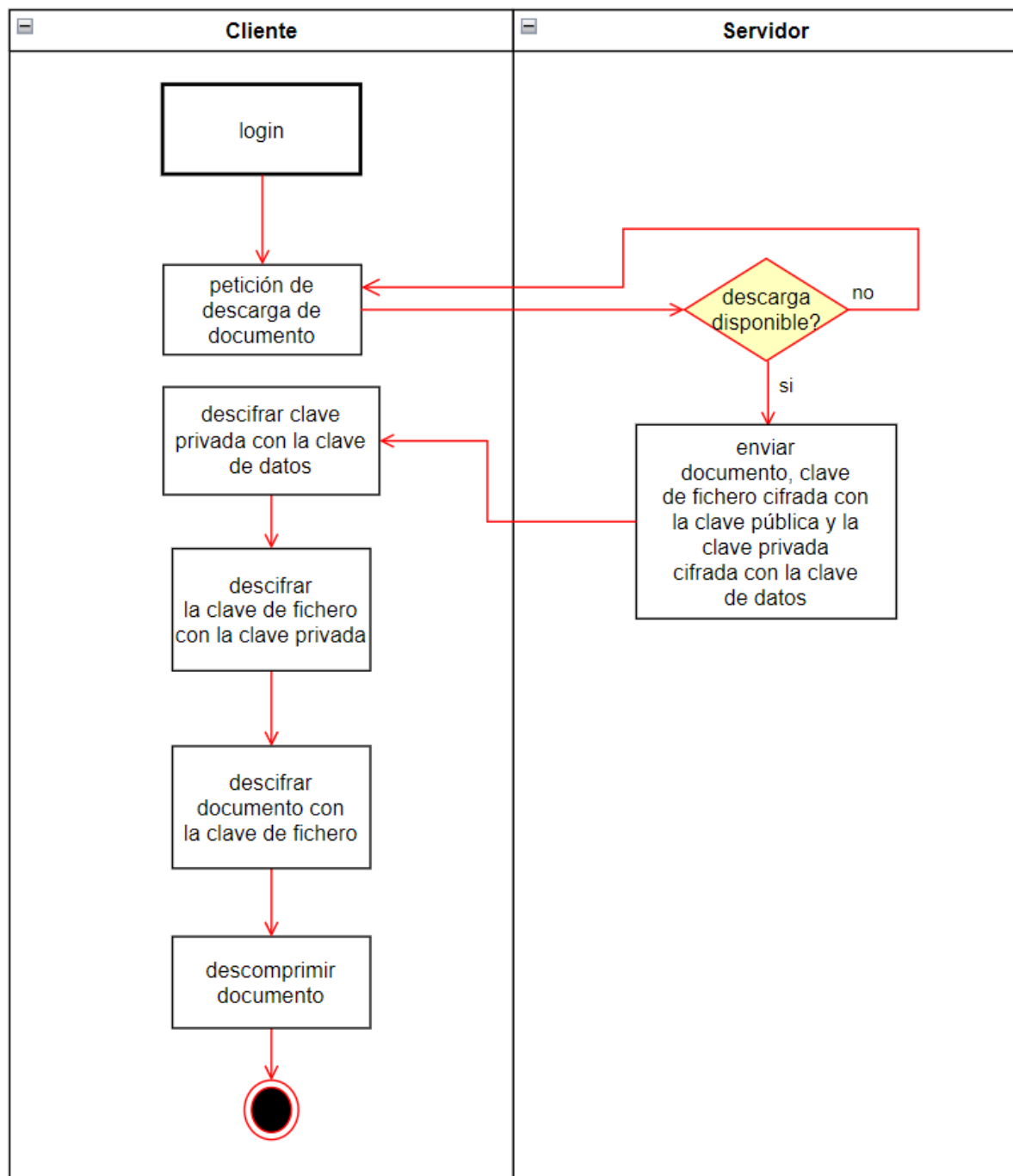
Login



Subida de documentos



Descarga de documentos



Glosario de claves

Clave de fichero

Cada fichero se cifra con una clave que se genera automáticamente y de forma aleatoria. Esta clave es diferente para cada fichero.

$$K_{File}^{AES}$$
$$E_{K_{File}^{AES}}^{AES}(File) = File.enc$$

Contraseña

El usuario debe proporcionar una contraseña para registrarse y para iniciar sesión. Esta contraseña tiene un número mínimo de caracteres, pero es escogida por el usuario, por lo que la seguridad no está garantizada. Se utiliza la contraseña para generar las claves de login y datos mediante el algoritmo SHA3-256.

$$Hash_{SHA3}(contraseña) = klogin, kdatos$$

Clave de login

Esta es la clave que se mandará al servidor a la hora de registrarse e iniciar sesión. Cuando se envíe al servidor, se obtendrá un hash mediante un algoritmo no determinista, concretamente bcrypt. De esta forma se logra que en el servidor no se almacenen las claves de los usuarios y también que si dos usuarios utilizan la misma contraseña, el hash no sea el mismo, aumentando así la dificultad de posibles ataques de fuerza bruta o diccionario.

Clave de datos

Con esta clave se cifran las claves de los ficheros antes de enviarse al servidor. Esta clave se genera al iniciar sesión y nunca se envía al servidor, por lo que la única forma de descifrar un fichero es que se inicie sesión en el usuario al que pertenece.

Clave pública

Se implementará en la siguiente práctica.

$$KU_{pub}^{RSA}$$
$$\text{Cifrado de ficheros: } E_{KU_{pub}^{RSA}}^{RSA}(K_{File}^{AES})$$

Clave privada

Esta clave, se utiliza para descifrar los documentos que han sido cifrados con la clave pública. Cada usuario tendrá su propia clave privada, que no será compartida a ningún usuario, y se almacenará en el servidor cifrada con la clave de datos:

$$E_{Upass}^{AES}(KU_{priv}^{RSA})$$

En el usuario, se descripta y se obtiene la clave privada en claro usando AES:

$$D_{Upass}^{AES}(E_{Upass}^{AES}(KU_{priv}^{RSA}))=KU_{priv}^{RSA}$$