Firma de archivos con DNIe

Carla Ballesteros (880156), Aranzazu Araguás (873300)

Resumen

Se ha realizado una aplicación en Python que permite firmar digitalmente archivos usando el propio DNIe y su correspondiente clave privada almacenada en él. Se han empleado librerías como Tkinter, PyKCS11 y cryptography. La aplicación dispone de un menú que ofrece tres funcionalidades: exportar el certificado digital, firmar documentos y verificar firmas. Además, con el objetivo de un uso más intuitivo para el usuario, se ha desarrollado una interfaz gráfica que actúa como GUI wrapper. De esta manera se mejora su experiencia frente al uso típico en consola y se cuidan aspectos de seguridad.

Palabras clave: python, tkinter, pyKCS11, cryptography, DNIe, firma digital, clave privada, clave pública, certificado, token, seguridad.

1 Requisitos iniciales para el desarrollo del programa

Antes de empezar a desarrollar el programa se debe preparar el entorno adecuado. Primero, es necesario instalar OpenSC, disponible en un repositorio oficial de GitHub, que proporciona la librería PKCS11 para poder establecer una comunicación con el chip del DNIe. Además, es indispensable disponer de un lector de tarjetas, donde poder introducir el DNIe físico y así acceder a sus certificados y claves privadas. Acerca del software de Python se han de instalar mediante el comando "pip install" los paquetes cryptography (para la gestión de certificados) y PyKCS11 (para el acceso de PKCS11 de OpenSC). Asimismo, como se ha incorporado una interfaz gráfica para el usuario con un fondo específico, se necesitan importar tanto las librerías tkinter y pillow. Una vez se disponga de todos estos elementados configurados se da comienzo a la creación y ejecución del programa.

2 Diseño y arquitectura

El programa desarrollado se basa en una arquitectura modular, divide el código en varias funciones independientes que encapsulan las operaciones criptográficas y de interfaz. El diseño del código se estructura en las siguientes funciones independientes: verificar_pin, mostrar_menu, exportar_certificado, firmar_documento, verificar_firma y cerrar_sesion.

El flujo de la aplicación comienza con una primera pantalla inicial donde se debe introducir el PIN del DNIe para validar su acceso. Una vez que el usuario es autenticado, se muestra un menú con cuatro funciones: exportar el certificado a un archivo .pem, firmar un documento de manera electrónica descargando un archivo .sig, verificar una firma digital y el cierre de sesión. A partir de unos botones que aparecen, el usuario podrá elegir una de las funciones anteriores, dependiendo de la acción seleccionada se invocará a la función correspondiente. En la Figura 1 se muestra la pantalla inicial y el menú del programa.

En el caso de que el usuario introdujese mal el PIN del DNIe, la aplicación no funcionaría y saltaría un mensaje de error. Además, la aplicación gestiona otras excepciones como por ejemplo: si la verificación de la firma falla se informa “Firma no válida” y se detiene la operación, si no se encuentran el certificado o la clave privada del DNI se notifica y se cierra la sesión de forma segura.

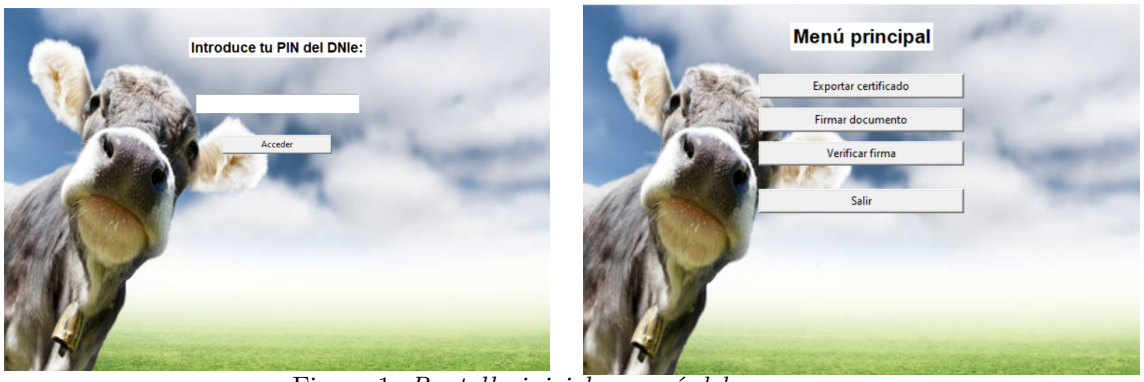

Figura 1: *Pantalla inicial y menú del programa.*

3 Funciones del programa

Al iniciar el programa, se ejecuta de manera automática un bloque de configuración que permite preparar el entorno antes de interactuar con el DNIe. En primer lugar, se carga la librería PKCS11, una vez cargada el programa busca todo tipo de lectores conectados (slots) y en caso de encontrar alguno disponible comprueba si dispone de un DNIe insertado (token). Como para la ejecución de este programa solo se ha insertado un lector, seleccionamos el primero disponible. A continuación, se inicializan unas variables globales que posteriormente almacenarán la sesión activa con el chip del DNIe, el certificado digital extraído y la clave privada asociada.

3.1 Función verificar_pin()

La función verificar_pin se encarga de la gestión del acceso inicial al DNIe a través del PIN introducido por el usuario. Tras leer el valor del PIN se valida y se procede a borrar inmediatamente el contenido del campo de texto y la variable que lo almacenaba, para evitar que el dato sensible quede en memoria. A continuación, se localizan los certificados que contiene el DNIe y se selecciona el correspondiente a la firma, que en este caso es el tercer certificado que aparece. Más tarde, se busca la clave privada asociada al certificado de firma para realizar operaciones criptográficas posteriormente. Si todo este proceso se realiza de manera satisfecha, se da paso a la interfaz del menú principal.

3.2 Función mostrar_menu()

Esta función muestra el menú principal tras validar el PIN. Para ello, elimina la ventana anterior y genera botones que permiten al usuario elegir la actividad que desee. Cada botón ejecutará la función correspondiente, facilitando la interacción.

3.3 Función exportar_certificado()

Exportar_certificado permite al usuario guardar en su equipo el certificado digital extraído del DNIe. Lo primero que hace es abrir una ventana de guardado mediante filedialog.asksaveasfilename(), donde el usuario elige dónde quiere guardar su certificado de extensión .pem. Para obtener el certificado a partir del DNI en el formato adecuado, debemos abrir el archivo donde queremos guardar el certificado en formato binario, y en él escribir el certificado extraído con el siguiente comando cert.public_bytes(serialization.Encoding.PEM). Si la exportación termina con éxito se muestra un mensaje informativo al usuario mediante messagebox.showinfo(). Por el contrario, si se produce cualquier error producido durante el proceso quedará capturado por el bloque except y será notificado mediante messagebox.showerror().

3.4 Función firmar_documento()

Esta función ofrece al usuario la posibilidad de seleccionar un archivo y firmarlo digitalmente a través de la clave privada almacenada en el DNIe. En caso de que no se seleccionara ningun archivo, la función se detendría.

El archivo seleccionado se abre en modo binario y se lee su contenido. Luego se define el mecanismo de firma RSA con hash SHA-256 y se genera la firma gracias a la clave asociada al DNIe con session.sign(clave_privada, data, mechanism). La firma resultante se convierte a bytes y se guarda como un nuevo archivo con extensión .sig, en la misma ruta del original. Tanto si el proceso finaliza correctamente como si se produce un error, se muestra un mensaje al usuario: messagebox.showinfo() o messagebox.showerror(), respectivamente.

3.5 Función verificar_firma()

La función verificar_firma comprueba la validez de una firma digital previamente generada con el DNIe. Abre dos cuadros de diálogo con el comando filedialog.askopenfilename() para que el usuario seleccione tanto el archivo original que quiere verificarse como la firma correspondiente al archivo. Una vez obtenidos ambos archivos se procede a leer su contenido en formato binario y extraer la clave pública asociada al certificado previamente adquirido en la función verificar_pin() mediante cert.public_key(). A continuación, se ejecuta el método verify() que verifica que la firma corresponda al archivo y persona correcta empleando el algoritmo de relleno PKCS1v15 y el hash SHA-256. Se termina informando al usuario de si la verificación es correcta o no, de esta forma, la función garantiza la autenticidad e integridad del documento firmado.

3.6 Función cerror_sesion()

La función cerrar_sesion() pone fin a la comunicación con el DNIe y a la aplicación. Si hay una sesión activa, ejecuta logout() y closeSession() para liberar los recursos de forma segura.

Después, invoca root.quit() para cerrar la interfaz de Tkinter. Con ello, se cierra la ventana y concluye la ejecución del programa.

4 Análisis de las limitaciones y seguridad de la aplicación

Actualmente, la aplicación cumple con sus funciones principales de firmar, verificar y exportar certificados correctamente, pero presenta algunas limitaciones. Su mayor debilidad reside en la gestión del PIN, ya que, aunque se elimine del campo de texto y de las variables inmediatamente después de usarse, no se garantiza que alguien no pueda obtenerlo realizando un debug o un análisis de la memoria del programa mientras se está ejecutando. Esto es debido a que Python no permite controlar como se guardan los datos sensibles en memoria. Además se debe manejar el certificado con cuidado ya que no está cifrado, simplemente es el texto plano codificado en Base64. Aunque este archivo no contiene la clave privada, en él se encuentra información personal como el nombre, el DNIe o la clave pública. Un factor importante en el diseño de esta aplicación es que ha sido probada para un único lector de tarjetas, el comportamiento con otros modelos de dispositivos puede variar ya que no todos gestionan de igual forma el acceso a los certificados del DNIe. Esto puede provocar que el orden en que se enumeren los certificados cambie y no se puedan ejecutar correctamente las operaciones de la aplicación. En general, la aplicación es adecuada para un uso básico y educativo, pero si queremos usarla en entornos más exigentes sería necesario reforzar la seguridad del PIN, cifrar los archivos generados y mejorar la dependencia con el lector de tarjetas y detección de certificados. Reforzando estos aspectos el programa se podría convertir en una herramienta mucho más segura y robusta para uso profesional.