

Instituto Tecnológico y de Estudios Superiores de Monterrey
Escuela de Ingeniería y Ciencias
Ingeniería en Ciencias de Datos y Matemáticas



Documentación de las bibliotecas ecdsa, pem, hashlib, binascii y PyPDF2 para su uso en
esquemas de firma digital

Frida Bautista Martínez - *A01235009*
Leyre Carpinteyro Palos- *A01610296*
Samuel Méndez Villegas - *A01652277*
Salette Noemi Villalobos - *A01246619*
Paola Montserrat Vega Ortega - *A01274773*
Ethan Enrique Verduzco Pérez - *A01066955*

Uso de álgebras modernas para seguridad y criptografía
Grupo: 401

Profesores:
Dr. Alberto Francisco Martínez Herrera
Dr. Daniel Otero Fadul

Organización Socio Formadora: Fundación Teletón

Monterrey, Nuevo León
22 de abril de 2022

Índice

1. Requisitos técnicos	3
2. Licencia	3
2.1. Definiciones	3
2.2. Concesión de Licencia de derechos de autor	3
2.3. Concesión de Licencia patente	3
2.4. Redistribución	4
2.5. Garantía	4
2.6. Limitaciones	4
2.7. Cómo aplicar la Licencia en su obra	4
3. Instalación, compatibilidad y dependencias.	5
3.1. Instalación de Python 3	5
3.2. Instalación de Microsoft Visual Studio Code	5
3.3. Instalación de pip	6
3.4. Instalar módulos de Python	6
4. Características principales	6
4.1. Generación de claves	6
4.2. Generación de firmas	8
4.3. Verificación de firmas	9
5. Descripción de la API o las funciones de la biblioteca con ejemplos.	11
5.1. Generación de claves	11
5.2. Generación de firmas	13
5.3. Verificación de firmas	16
5.4. Página web	17
6. Planes futuros de desarrollo	21
7. Ayuda y FAQs	22
7.1. Ayuda	22
7.2. FAQs	22
8. Anexos	23
8.1. Vectores de prueba	23

1. Requisitos técnicos

Requisitos mínimos:

- OS: Windows 8.1 de 64 bits, Windows 8 de 64 bits
- Procesador: Intel Core 2 Quad CPU Q6600 a 2.40GHz (4 CPU) / AMD Phenom 9850 Quad-Core (4 CPU) a 2.5 GHz
- Memoria: 2 GB

Requisitos recomendados:

- Windows 8.1 de 64 bits, Windows 8 de 64 bits, Windows 7 de 64 bits con Service Pack 1
- Procesador: Intel Core i5 3470 a 3.2 GHZ (4 CPU) / AMD X8 FX-8350 a 4 GHZ (8 CPU)
- Memoria: 4 GB

Otros requisitos:

- Es necesaria la conexión a Internet para descargar el Software de Python o alguna otra plataforma donde se pueda ejecutar el script como lo es Visual Studio Code. En caso de estar instalado Python en una máquina remota también es necesario tener una conexión remota con el equipo. La última versión se puede bajar desde la página principal.

2. Licencia

2.1. Definiciones

Licencia

La licencia determina las condiciones de uso, reproducción y distribución.

Licenciador

Es la persona que es dueña de la entidad y que puede autorizar la Licencia.

Usted

Hace referencia a una persona o entidad que utilice el trabajo dado por esta licencia.

Obra

Se refiere a la obra del autor, ya sea código u objetos.

2.2. Concesión de Licencia de derechos de autor

Sujeto a los términos y condiciones de esta Licencia, se otorga una licencia de derechos de autor gratuita, no exclusiva, libre de regalías e irrevocable para reproducir, exhibir y reproducir públicamente la Obra.

2.3. Concesión de Licencia patente

Se le otorga a Usted una Licencia de patente irrevocable, no exclusiva, perpetua, sin regalías ni cargas para poder utilizar, vender, transmitir, etc. la Obra.

2.4. Redistribución

Usted puede reproducir y distribuir copias de este trabajo en cualquier medida, con o sin modificaciones. Sin embargo, se requieren las siguientes condiciones:

- Se debe dar una copia de esta Licencia a las personas con las que se distribuya el contenido.
- Se debe indicar si se modificó algún archivo.
- Se debe retener y distribuir la fuente de donde se extrajo el proyecto.

2.5. Garantía

El Licenciador otorga el trabajo “tal cual”, sin garantías o condiciones de cualquier tipo. Usted es el responsable de determinar si es apropiado para su posible uso y distribución.

2.6. Limitaciones

Bajo ninguna circunstancia ni fundamento legal será responsable algún contribuyente ante usted por agravio de ningún tipo ocasionados por la presente Licencia o por mal uso de esta.

2.7. Cómo aplicar la Licencia en su obra

Para aplicar la Licencia en su trabajo añada el siguiente aviso, reemplazando el espacio en el interior de los corchetes “[]” con su propia información identificativa:

Derechos de autor [yyyy] [nombre del propietario de los derechos de autor]

Autorizado en virtud de la Licencia del autor; se prohíbe aplicar este archivo excepto en ejecución de la Licencia. Podrá conseguir una copia de la Licencia en:

<https://www.overleaf.com/read/mvfpsrytnypk>

A menos que lo demanden las leyes, el software se distribuye “tal cual”, sin garantías ni condiciones de ningún tipo.

Las Licencias usadas dentro del código fueron las siguientes:

- **ecdsa**: Free Open Source software released under the MIT license.
- **SigningKey**: Free Open Source software released under the MIT license.
- **VerifyingKey**: Free Open Source software released under the MIT license.
- **NIST256p**: Free Open Source software released under the MIT license.
- **pem**: Free Open Source software released under the MIT license.
- **PyPDF2**: Free and open-source pure-python PDF library under the BSD license.
- **hashlib**: Free Open Source software released under Python Software Foundation License.
- **binascii**: Free Open Source software released under the MIT license.
- **unhexlify**: Free Open Source software released under the MIT license.

3. Instalación, compatibilidad y dependencias.

Para poder trabajar con todas las librerías y scripts es necesario instalar lo siguiente:

3.1. Instalación de Python 3

Python es un lenguaje de programación de código abierto, está interpretado y es multiplataforma, es decir, que lo podemos usar en cualquier sistema operativo incluyendo Windows.

1. El proceso de instalación se logra a través de su página principal: [https : //www.python.org/downloads/](https://www.python.org/downloads/) , en su versión de 32 bits o de 64 bits, depende la versión de nuestro sistema operativo. Una vez descargado debemos ejecutarlo.
2. Podemos ver dos opciones del instalador, el principal encargado de tener el IDLE, PIP y documentación dentro de nuestra máquina y el botón de Customize para poder elegir la carpeta donde queremos instalarlo y las funciones.
3. Se seleccionan todas las Optional Features e iniciamos el proceso de instalación con el botón de Next. Esto puede tardar algunos segundos o minutos.
4. Una vez instalado, ya podemos trabajar con Python y podemos comprobar que lo instalamos correctamente en el explorador de archivos de Windows para ver su ruta de instalación. Sin embargo, si hemos seleccionado otro directorio, este será diferente e igual la versión del python (referencia).

3.2. Instalación de Microsoft Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

Para su instalación es necesario realizar lo siguiente:

1. Dirigirse a la página de Microsoft Visual Studio Code en Academic Software y hacer clic en el botón 'Descargar Visual Studio Code' para descargar el archivo de instalación.
2. Abrir el archivo de instalación .exe en tu carpeta de descargas para iniciar la instalación.
3. Leer y aceptar el acuerdo de licencia. Hacer clic en Next para continuar.
4. Se puede cambiar la ubicación de la carpeta de instalación o mantener la configuración predeterminada. Hacer clic en Next para continuar.
5. Elegir si desea cambiar el nombre de la carpeta de accesos directos en el menú Inicio o instalar accesos directos en absoluto. Hacer clic en Next.
6. Seleccionar las tareas adicionales, por ejemplo: crear un icono en el escritorio o añadir opciones al menú contextual de Windows Explorer. Haz clic en Next.
7. Haz clic en Install para iniciar la instalación.
8. Con eso finalmente el programa está instalado y listo para usar. Hacer clic en Finish para finalizar la instalación y lanzar el programa.

3.3. Instalación de pip

El programa pip es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python. Muchos paquetes pueden ser encontrados en el Python Package Index. Python 2.7.9 y posteriores, Python 3.4 y posteriores incluyen pip por defecto

Para instalar pip utilizando la terminal de línea de comandos, navega al directorio en el que se instaló Python y se debe obtener get-pip.py.

Para este ejemplo asumimos el directorio python37, así que se debe usar el comando C: cd python37. Una vez que se esta en ese directorio, para instalar pip ejecute el comando:

```
1 python get-pip.py
```

3.4. Instalar módulos de Python

1. **ecdsa**: Esta biblioteca está disponible en PyPI, se recomienda instalarla usando pip, la versión utilizada fue 0.17.0:

```
1 pip install ecdsa
```

En caso de que se desee un mayor rendimiento y el uso de código nativo no sea un problema, es posible especificar la instalación junto con gmpy2:

```
1 pip install ecdsa[gmpy2]
```

2. **pem**: Esta biblioteca está disponible en PyPI, se recomienda instalarla usando pip, la versión utilizada dentro del código es la 21.2.0:

```
1 pip install pem
```

3. **PyPDF2**: Esta biblioteca está disponible en PyPI, se recomienda instalarla usando pip, la versión utilizada fue 1.27.8:

```
1 pip install PyPDF2
```

4. Características principales

4.1. Generación de claves

En la generación de claves se empleó el algoritmo ECDSA (Elliptic Curve Digital Signature Algorithm) con la curva NIST 256 bits. Se asume además que únicamente el administrador o la figura autorizada tendrá acceso al sistema. El propósito general del procedimiento es tener un sistema de certificados basado en la PKI (Public Key Infrastructure), donde las llaves serán plenamente identificadas con los campos: 'ID Algoritmo', 'Emisor', 'No antes de', 'No después de', 'Sujeto', 'Algoritmo de clave pública', 'Clave

pública' y 'Estado'. A continuación se muestra en detalle el significado que se le dará a cada uno de estos campos:

- **ID Algoritmo:** número único con el que se identifica el registro de la clave.
- **Emisor:** organización que está emitiendo el certificado.
- **No antes de:** fecha que establece hasta qué día la certificación será activada.
- **No después de:** fecha que establece hasta que día la certificación tiene vigencia
- **Sujeto:** organización a la que se le está emitiendo el certificado.
- **Algoritmo de clave pública:** algoritmo criptográfico que se está utilizando para generar las claves.
- **Clave pública:** clave pública del sujeto guardada en formato de bits.
- **Estado:** indica si las llaves se encuentran activas, inválidas, revocadas o expiradas.

Además, del estado activo de las llaves, estas pueden llegar a ser revocadas, expiradas o inválidas según la circunstancia. A continuación se indica el significado de estos términos:

- **Revocada:** El administrador decide revocar una clave determinada.
- **Expirada:** En caso de que se supere la fecha 'No después de', la clave se marcará como expirada.
- **Inválida:** Si la fecha 'No antes de' todavía no se cumple, se registrará como inválida.

Para el almacenamiento y administración de las claves públicas se decidió implementar una base de datos en Python. Dicha base de datos se compondrá únicamente de los campos descritos con anterioridad. Adicionalmente, todos los cambios y nuevos registros se irán almacenando en la base de datos del sistema.

En la Figura 1. se observa el diagrama de flujo correspondiente al proceso de generación de claves públicas y certificados por parte de la entidad autorizada para llevar a cabo esta acción.

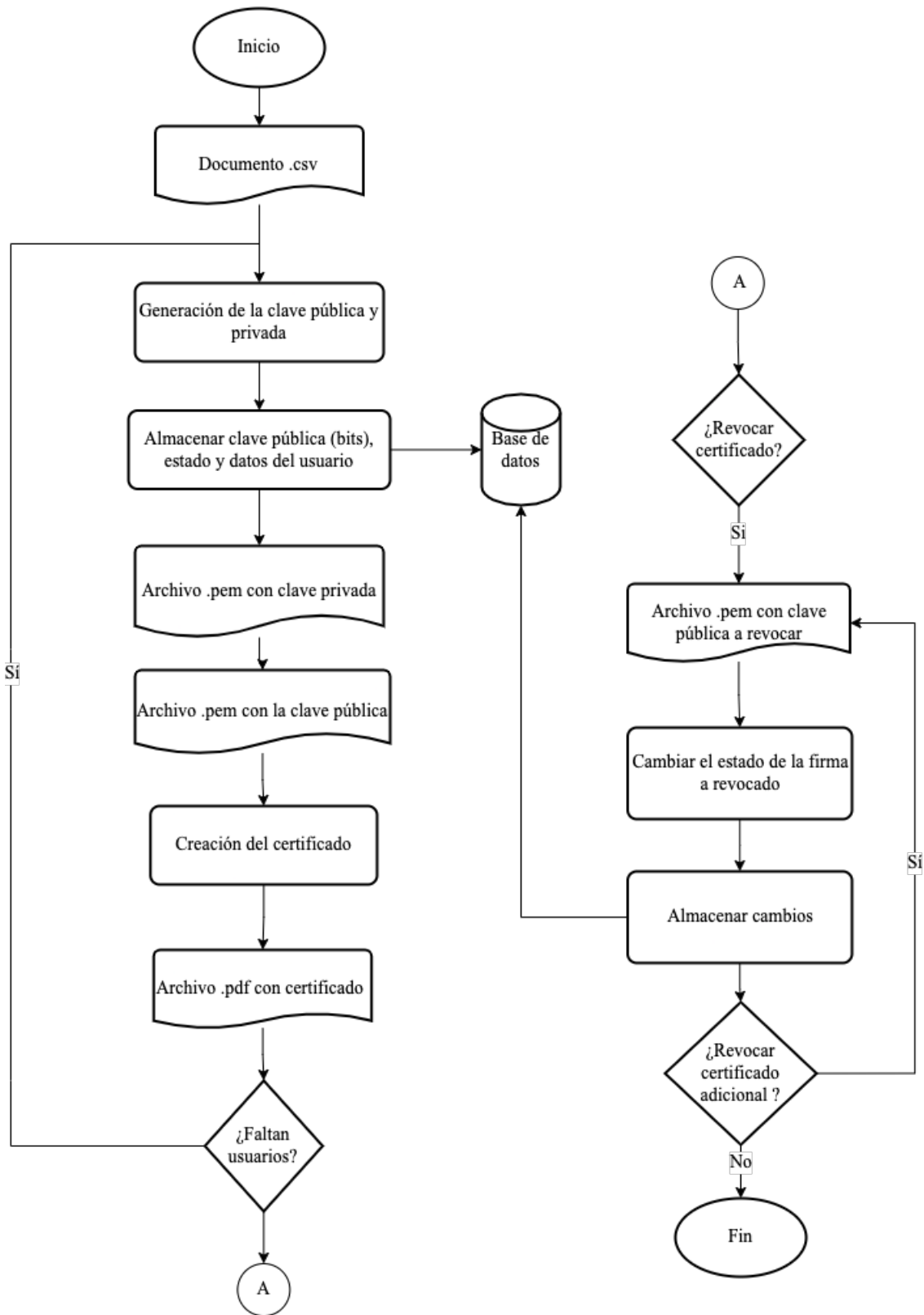


Figura 1: Diagrama de flujo de la generación de claves y certificados para los usuarios

4.2. Generación de firmas

La segunda parte fundamental del esquema de firma digital es el relacionado con la generación de la firma digital. Para su generación se utilizará el algoritmo ECDSA (Elliptic Curve Digital Signature Algorithm) y la curva NIST con 256 bits, para lo cual se requiere la clave privada del usuario y el

documento que se desea firmar. De acuerdo con lo planteado por la organización, la mayoría de los documentos que se manejan para en el Departamento de Compras son en formato pdf, por ello en esta solución se considera este formato como el tipo de archivo que el usuario desea firmar.

En la Figura 2. se muestra el diagrama de flujo relacionando con el proceso de generación de la firma digital.

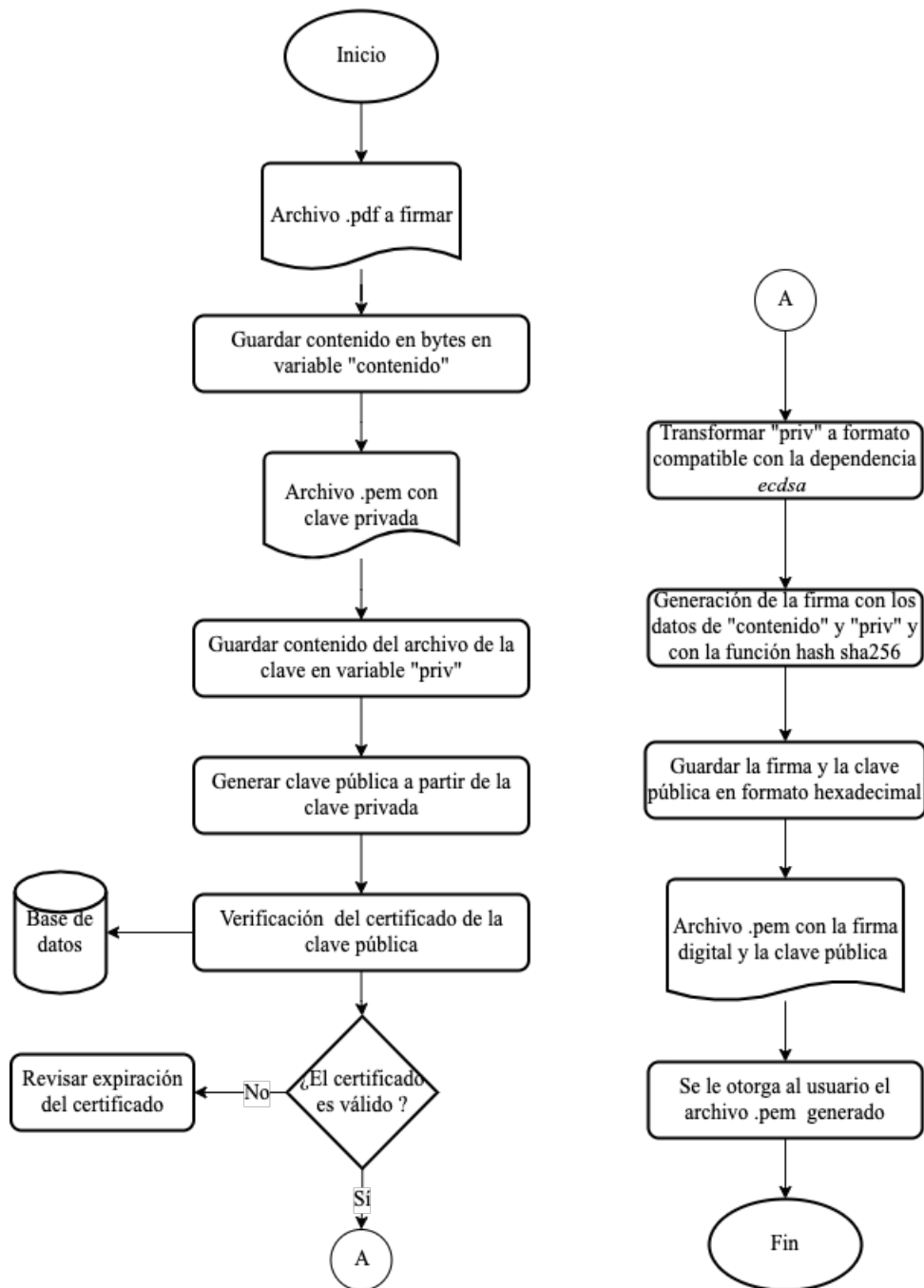


Figura 2: Diagrama de flujo de la generación de firma digital

4.3. Verificación de firmas

El último proceso fundamental dentro del esquema de firma digital, es el correspondiente a la verificación de la firma por parte de otro usuario. Dicho proceso se realiza utilizando la misma librería, **ecdsa**. En la Figura 3. se muestra el diagrama de flujo correspondiente al proceso de verificación.

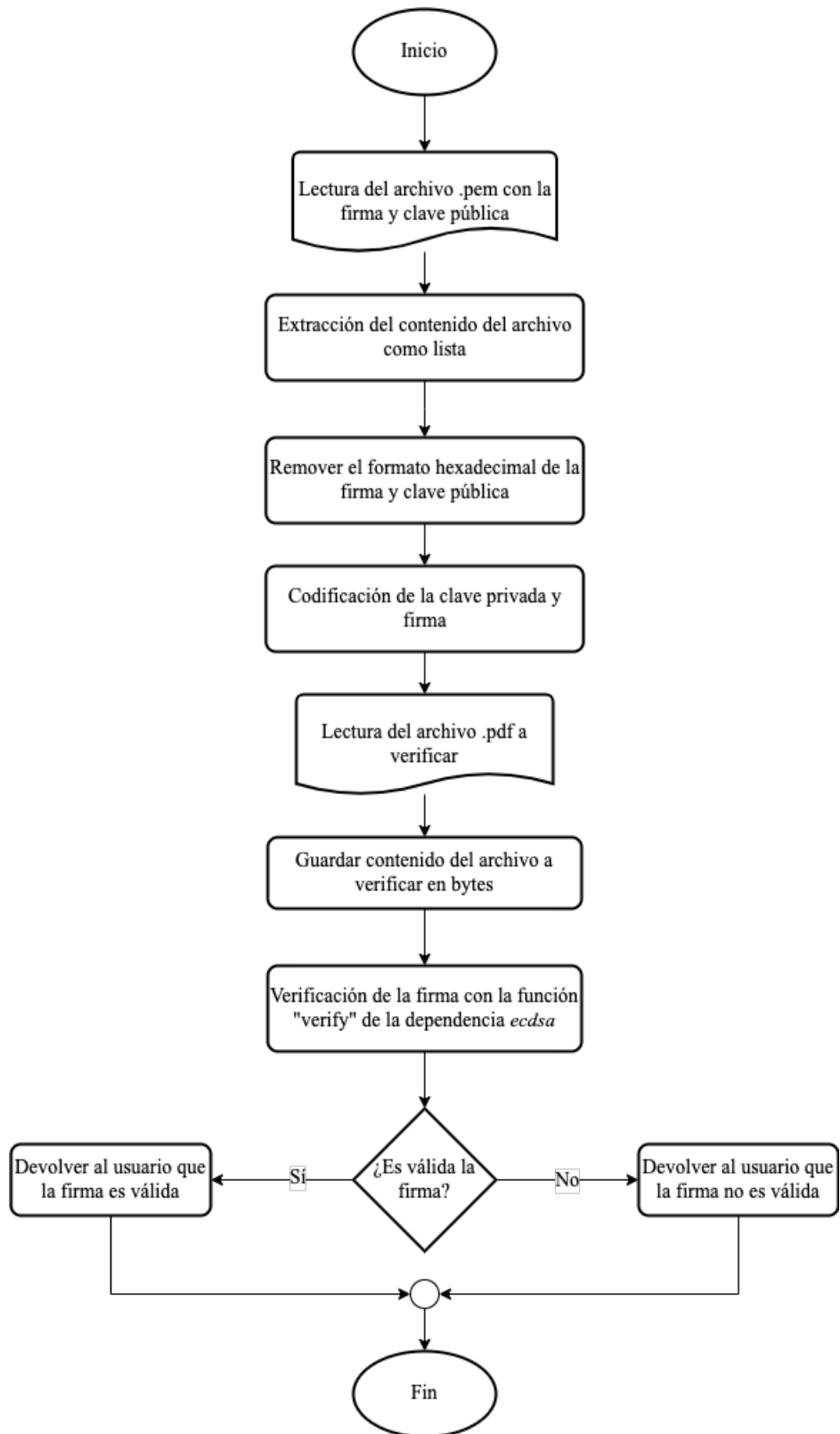


Figura 3: Diagrama de flujo de la verificación de firma digital

5. Descripción de la API o las funciones de la biblioteca con ejemplos.

5.1. Generación de claves

El primer paso consistirá en la lectura de un archivo CSV que será proporcionado por el usuario con los datos de los sujetos que adquirirán una llave pública y privada. Los campos deben de ser 'ID Algoritmo', 'Emisor', 'No antes de', 'No después de', 'Sujeto' y 'Algoritmo de clave pública'. El resto de los campos se agregarán automáticamente durante el proceso. A continuación se puede ver el código de la lectura del archivo:

```
1 def lectura_csv(nombre_csv):
2     data = []
3     with open(nombre_csv, mode='r') as file:
4
5         csvFile = csv.reader(file)
6         for lines in csvFile:
7             data.append(lines)
8
9     return data
```

Una vez almacenados estos datos, prosigue crear las claves públicas y privadas para cada individuo con el uso de la librería ECDSA. Además, se agrega el 'Estado' de los sujetos como 'Activa' y se añade la firma pública de cada sujeto a la lista de datos. A continuación, se presenta el código:

```
1 def crear_claves(data, num_sujeto):
2
3     private = ecdsa.SigningKey.generate(curve=ecdsa.NIST256p,)
4     privatepem = private.to_pem()
5     public = private.get_verifying_key()
6
7     data[num_sujeto-1].append(public)
8     data[num_sujeto-1].append('Activa')
9
10    return data, privatepem, public
```

Se crean archivos .pem de la clave pública y de la clave privada para cada uno de los usuarios. Para ello, se convierte el formato de la llave de bits a texto plano. El código es el siguiente:

```
1 def byte_a_texto(key):
2     p_key = key.to_string()
3     p_key = codecs.decode(p_key, 'latin1')
4     return p_key
```

```
1 def archivo_key(num_archivo, key, name):
2
3     name_a = name + str(num_archivo) + '.pem'
4
5     if name == 'privateKey':
6         text_file = open(name_a, "wb")
7         text_file.write(key)
8         text_file.close()
9     elif name == 'publicKey':
```

```

10     text_file = open(name_a, "w", encoding="utf-8")
11     text_file.write(byte_a_texto(key))
12     text_file.close()

```

Posteriormente, se crea un certificado para cada uno de los usuarios en formato PDF para que este pueda ser firmado por la autoridad certificadora, en este caso el administrador del sistema, con la Generación de Firma Digital.

Una vez que se ha obtenido la información necesaria, se almacenan todos los campos anteriormente mencionados en la base de datos del sistema con el siguiente código:

```

1 def cargar_base(data, nom_base):
2
3     for s_data in data:
4
5         with open(nom_base, 'a', newline='') as f_object:
6             writer_object = writer(f_object)
7             writer_object.writerow(s_data)
8             f_object.close()

```

El resultado se observa en el Cuadro 1.

ID Algoritmo	Emisor	No antes de	No despues de	Sujeto	Algoritmo CP	Clave Publica	Estado
ECDSA	Teleton	20/04/2022	30/04/2022	Teleton inc	ECDSA	b'\x15(c\x1e\x1a\x2\xddi"\xb3@\xa2Y-\x00ir\x0...	Activa
ECDSA	Uber	23/04/2022	30/04/2022	Uber inc	ECDSA	b'\x84CA\xbd\x7\x5\t\x9c\xfcPa\x02\x97\x17\x...	Activa
ECDSA	CEMEX	18/04/2022	19/04/2022	CEMEX	ECDSA	b'\x9d\xa8Y\x8\x0C\x16\rH\xfeZ-\xb9#\x9d~\xc...	Activa

Cuadro 1: Base de datos.

Para revocar, será necesario que el administrador proporcione un archivo .pem con la clave pública del sujeto que se quiere inhabilitar. El archivo se transforma primero de texto a bits, posteriormente se encuentra en la base de datos y se cambia su estado a 'Revocado'. A continuación se presenta el código para llevar a cabo las tareas:

```

1 def texto_a_bytes(nom_archivo):
2
3     with open(nom_archivo, 'r') as key:
4         key_bytes_pk = key.read()
5         key_bytes_pk = codecs.encode(key_bytes_pk, 'latin1')
6
7     return key_bytes_pk
8
9 def revocar_firma(nom_file, data_frame, base_name):
10
11     public_key_r = str(texto_a_bytes(nom_file))
12     data_frame.loc[data_frame.iloc[:,6] == public_key_r, 'Estado'] = 'Revocado'
13     data_frame.to_csv(base_name, index=False)
14     return data_frame

```

Con fines de prueba, se revoca el permiso de la llave otorgada a Teletón Inc. La base de datos se actualiza como se observa en el Cuadro 2.

ID Algoritmo	Emisor	No antes de	No despues de	Sujeto	Algoritmo CP	Clave Publica	Estado
ECDSA	Teleton	20/04/2022	30/04/2022	Teleton inc	ECDSA	b'\x15(c\x1e\x1a\x1b2\xddi"\xb3@\xa2Y-\x00ir\x0...	Revocado
ECDSA	Uber	23/04/2022	30/04/2022	Uber inc	ECDSA	b'\x84CA\xbd\x7\x5t\x9c\xfcPa\x02\x97\x17\x...	Activa
ECDSA	CEMEX	18/04/2022	19/04/2022	CEMEX	ECDSA	b'\x9d\xa8Y\xc8\xc0C\x16\rH\xfeZ-\xb9#\x9d~\xc...	Activa

Cuadro 2: Cambio de estado a revocado.

Para verificar que la fecha de la clave sea válida, se utiliza la siguiente función:

```

1 def comprobar_caducidad(data_frame, base_name):
2
3     fecha_actual = datetime.now()
4
5     data_frame.iloc[:,2]= pd.to_datetime(data_frame.iloc[:,2])
6     data_frame.iloc[:,3]= pd.to_datetime(data_frame.iloc[:,3])
7
8     data_frame.loc[data_frame.iloc[:,2] > fecha_actual, 'Estado'] = 'Invalido'
9     data_frame.loc[data_frame.iloc[:,3] < fecha_actual, 'Estado'] = 'Expirado'
10
11     data_frame.to_csv(base_name, index=False)

```

Comprueba que la fecha actual sea superior a la fecha 'No antes de', en caso contrario, se asigna un estado de 'Inválido'. Además, se revisa que todavía no se supere la fecha de caducidad, de no ser así, se asigna un estado de 'Expirado'. La base de datos después de esta función, utilizada en la fecha 21/04/2022, se observa en el Cuadro 3.

ID Algoritmo	Emisor	No antes de	No despues de	Sujeto	Algoritmo CP	Clave Publica	Estado
ECDSA	Teleton	20/04/2022	30/04/2022	Teleton inc	ECDSA	b'\x15(c\x1e\x1a\x1b2\xddi"\xb3@\xa2Y-\x00ir\x0...	Revocado
ECDSA	Uber	23/04/2022	30/04/2022	Uber inc	ECDSA	b'\x84CA\xbd\x7\x5t\x9c\xfcPa\x02\x97\x17\x...	Invalido
ECDSA	CEMEX	18/04/2022	19/04/2022	CEMEX	ECDSA	b'\x9d\xa8Y\xc8\xc0C\x16\rH\xfeZ-\xb9#\x9d~\xc...	Expirado

Cuadro 3: Revisar caducidad.

5.2. Generación de firmas

Los primeros pasos del proceso requieren de lectura del archivo **pdf** a firmar, junto con la extracción de su contenido. Cabe aclarar que se busca que la firma sea sobre el documento, en lugar de que sea sobre el texto o el contenido del archivo. Dado esto, se hace uso de bytes para leer el archivo y que en estos se almacene toda la información que contenga el **pdf** (imágenes, texto, tablas, etc.). A continuación se muestra el código para ello:

```

1 archivo = open("Nombre del documento.pdf", "rb")
2 contenido = archivo.read()
3 archivo.close()

```

Donde **rb** indica el modo de lectura de bytes del archivo, los cuales son almacenados en **contenido**. Posteriormente, se debe obtener su hash para que, junto con la clave privada se genere la firma. En este punto se debe enfatizar en la función hash que se utiliza para la solución es SHA256, pues esta cuenta con un alto nivel de seguridad que garantiza la integridad de la información de los archivos de lectura. Para la generación del hash, se utilizó la biblioteca **hashlib** de Python, la cual implementa varias funciones hash, entre las cuales se encuentra SHA256.

Para comprobar el funcionamiento de la librería, se realizó una prueba en donde utilizando la herramienta ‘Checksum Online’ se comparó el hash resultante de un archivo pdf de prueba a firmar. Dicho archivo se nombró ‘Prueba-Hash’ y contiene únicamente texto. Primero se calculó el hash con la herramienta ‘Checksum Online’, y en la Figura 17. se muestra el resultado del hash obtenido

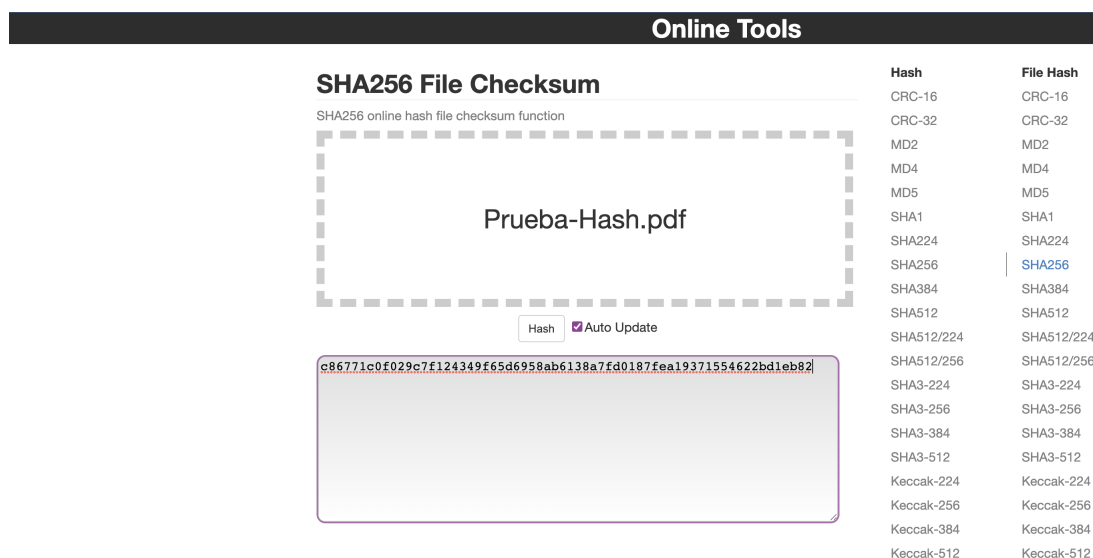


Figura 4: Hash del archivo ‘Prueba-Hash.pdf’ obtenido de la función SHA256 utilizando ‘Checksum online’

Posteriormente, se utilizó el siguiente código para realizar esta prueba utilizando **hashlib** y en la figura 5 se muestra el hash resultante.

```

1 import hashlib
2 archivo = open('Prueba-Hash.pdf', 'rb')
3 contenido = archivo.read()
4 archivo.close()
5
6 h = hashlib.new('sha256')
7 h.update(contenido)
8 h.hexdigest()

```

Out[76]: 'c86771c0f029c7f124349f65d6958ab6138a7fd0187fea19371554622bd1eb82'

Figura 5: Hash del archivo ‘Prueba-Hash.pdf’ obtenido de la función SHA256 de la librería **hashlib** de Python’.

Como se puede observar, el hash obtenido del documento fue exactamente el mismo, comprobando de esta manera el correcto funcionamiento de la librería. Sin embargo, se decidió realizar una segunda prueba utilizando otro documento pdf, pero esta vez incluyendo imágenes, tablas y otras estructuras dentro del documento. A dicho archivo se le nombró ‘Prueba-Hash2’. El proceso que se realizó para calcular el hash fue exactamente el mismo que en la prueba anterior, por que en las figuras 6 y 7 se muestran los hashes obtenidos.

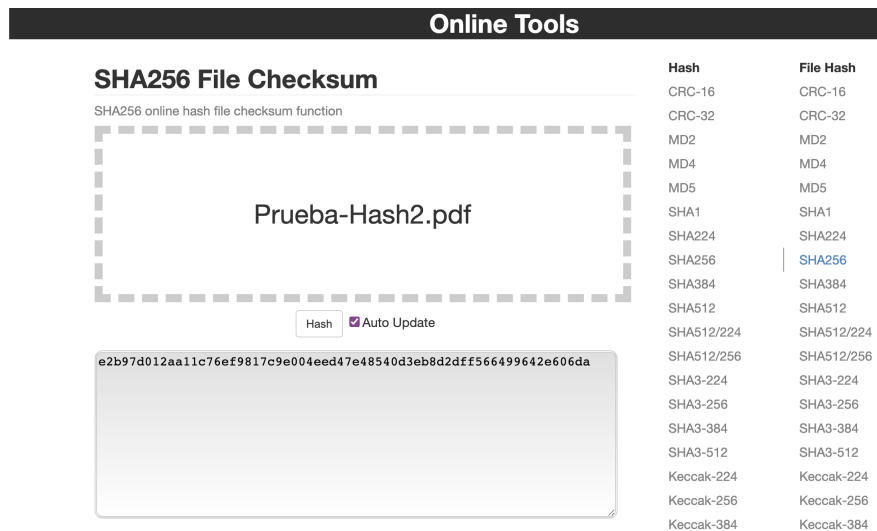


Figura 6: Hash del archivo ‘Prueba-Hash2.pdf’ obtenido de la función SHA256 utilizando ‘Checksum online’

```
Out[77]: 'e2b97d012aa11c76ef9817c9e004eed47e48540d3eb8d2dff566499642e606da'
```

Figura 7: Salida de la Prueba

Como se observa en estas dos últimas figuras, el hash obtenido fue idéntico en ambas aplicaciones, dejando aún más claro que la biblioteca de **hashlib** es adecuado para crear hashes a partir de archivos pdf.

Regresando al proceso de la generación de firma, esta última, las adquiere el usuario en un archivo con formato **.pem**, debido a que se utiliza generalmente para guardar certificados y claves públicas o privadas. De este archivo, se obtiene la clave privada con el mismo formato **.pem**. Lo anterior se realiza mediante lo siguiente:

```
1 private = open('private.pem', 'r')
2 private = private.read()
3 private = private.encode()
```

La generación de firma será por medio de la librería **ecdsa**, por lo que se requiere que la clave privada obtenida sea un objeto de esta misma. Para ello, se emplea la función **SigningKey**, como se muestra:

```
1 private_key = SigningKey.from_pem(private)
```

Una vez siendo objeto de la librería, se puede utilizar el método **.sign()** para firmar, este recibe de parámetros el contenido del PDF (**contenido**) y el tipo de hash que se desea para el archivo (SHA256 en este caso), además se puede establecer un valor **k** para en el caso de vectores de prueba.

```
1 signature = private_key.sign(contenido, hashfunc=hashlib.sha256)
```

La firma **signature** tiene el formato automático en bytes, no obstante se optará por regresar al usuario la firma en hexadecimal en un archivo **.pem**. En el mismo archivo se regresará la clave pública

del usuario. Este proceso se realiza de la siguiente manera:

```
1 public_key = private_key.get_verifying_key()
2 firma_hex = sig.hex()
3 public_hex = public_key.to_string().hex()
4
5 #Creaci n del archivo .pem
6 with open('firma_clavepublica.pem', 'w') as file:
7     file.write("%s %s" % (firma_hex, public_hex))
```

De esta manera se da por terminado el proceso de generación de la firma digital a partir de la clave privada y el documento a firmar.

5.3. Verificación de firmas

En esta etapa se necesita la clave pública, la firma y el mensaje que se envió, por lo que el primer paso es la lectura de dichos documentos. Para obtener la clave pública y la firma, que previamente se guardaron en un archivo `pem`, se realiza lo siguiente:

```
1 string = open('firma_clavepublica.pem', 'r')
2 string = string.read()
3 lista = string.split()
4 ## Clave p blica de 'string' a 'bytes'
5 public_verify =lista[1]
6 public_verify = public_verify.encode()
7 ## Firma de 'string' a 'bytes'
8 firma_verify =lista[0]
9 firma_verify = firma_verify.encode()
```

Como se ve en el bloque de código anterior, se extrajo la información del archivo en forma de lista para que de esta manera de pudiera separar la parte correspondiente a la firma digital, de la clave pública.

Antes de realizar la verificación se debe regresar la firma y la clave pública a sus formatos originales para la función de verificación. Lo que se necesita es cancelar el formato hexadecimal que se les agregó anteriormente. Para esto se utiliza la función `unhexlify` de la librería `binascii`:

```
1 public_kv = unhexlify(public_verify)
2 firma_kv = unhexlify(firma_verify)
```

De manera similar se necesita otra modificación a la clave pública, ya que necesita ser un objeto de la librería. Para esto existe una función dentro de la librería que transforma la clave de un `string` a un objeto recibiendo también la curva utilizada.

```
1 public_kv= VerifyingKey.from_string(public_kv, curve=ecdsa.NIST256p)
```

Posteriormente, el usuario sube el `pdf` que se firmó y el algoritmo realizará lo mismo que se realizó anteriormente para la lectura del documento en la parte de la generación de la firma. Es decir lee el archivo y guarda su contenido en bytes para poder ser utilizado en la función correspondiente.

Finalmente, la verificación también consiste en una sola función que recibe el mensaje, la firma y la clave pública. Se incluye la función `hash` utilizada, la cual debe ser la misma utilizada para la firma.

El resultado se presenta como un `True` o un `False`, el primero confirmando que la firma es válida y el segundo lo contrario.

```
1 public_kv.verify(firma_kv, contenido, hashfunc=hashlib.sha256)
```

De esta manera, termina el proceso de verificación de la firma, y por lo tanto, concluyendo con todo el esquema de firma digital planteado, por lo que en las siguientes secciones se dedicarán a evaluar el desempeño de la solución.

5.4. Página web

A continuación se presenta la plataforma y cómo funciona, además de visualizar los botones y opciones que podemos encontrar.

En la Figura 8 podemos encontrar la página inicial donde del lado superior izquierdo encontramos el menú para generar la clave, y abajo un botón para subir los datos del usuario en formato csv, además de un botón para enviar los archivos.

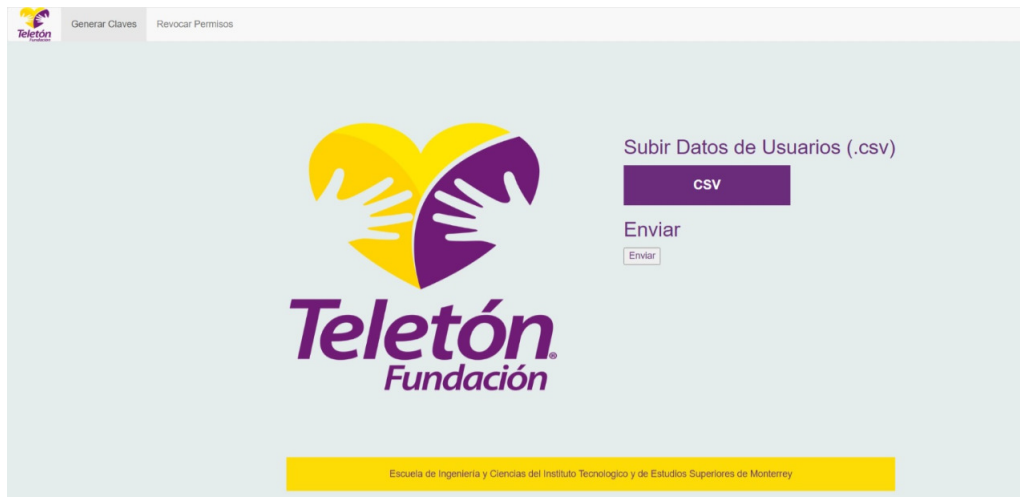


Figura 8: Página principal.

En la siguiente Figura 9 encontramos la forma en la que seleccionamos el archivo a subir.

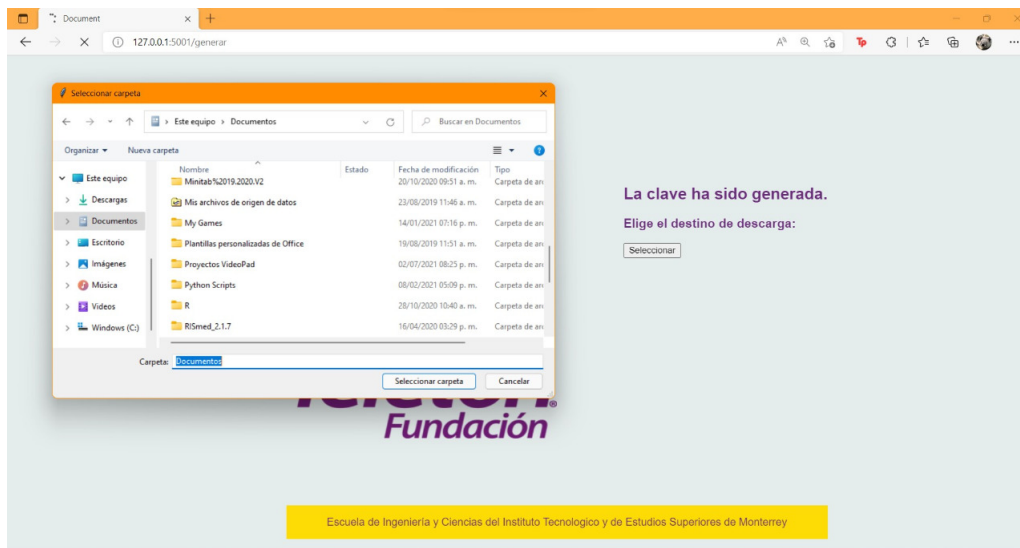


Figura 9: Subir archivo desde carpeta del usuario.

Y dentro de la Figura 10 se muestra cómo debe de aparecer el mensaje de aprobación el cual indica que el archivo ha sido cargado correctamente.



Figura 10: Verificación de carga de archivo.

En la Figura 11 podemos encontrar la página inicial al seleccionar dentro del índice la opción de revocación de permisos, donde se muestra un botón para subir la clave pública a revocar en formato .pem y un botón para enviar el archivo.

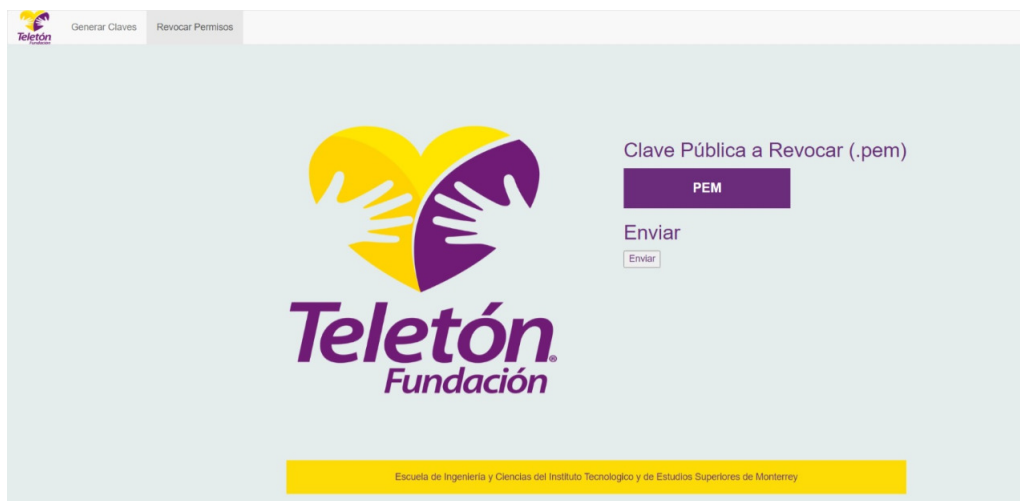


Figura 11: Página inicial de Revocación de permisos

A continuación se muestra el mensaje que debe aparecer tras hacer la carga del archivo verificando la carga de este.



Figura 12: Verificación de carga de archivo .pem

Después de realizar el proceso anterior, es necesario ingresar a la otra página donde de nuevo encontramos un menú del lado superior izquierdo. En la Figura 13 podemos ver la página inicial del menú de Firma donde se debe adjuntar la clave privada y el documento a firmar, además de tener el botón para subir los archivos pertinentes.



Figura 13: Página inicial para la firma.

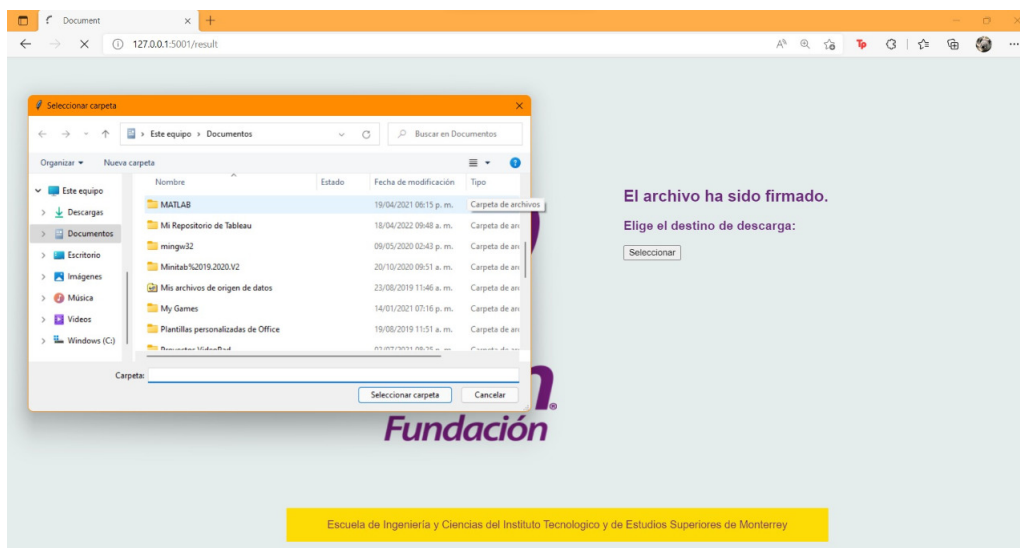


Figura 14: Selección del archivo.

Y después podemos encontrar el mensaje de confirmación sobre la carga del archivo.



Figura 15: Mensaje de comprobación de los archivos.

Y por último, dentro de la Figura 16 encontramos la página principal del menú de verificación donde de igual manera encontramos los botones de carga de archivo de la firma con el certificado y el documento.

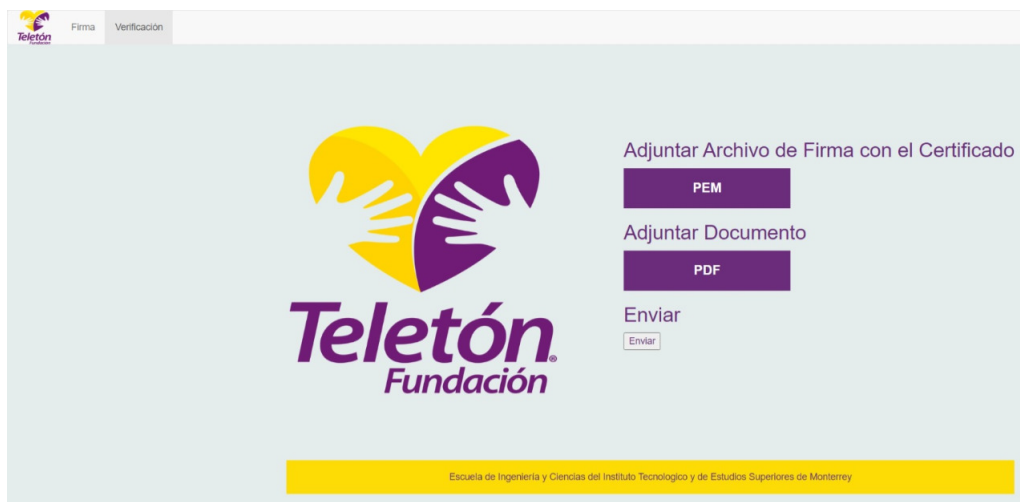


Figura 16: Menú de carga de archivos para verificación de firma.




Figura 17: Confirmación de verificación de firma.

6. Planes futuros de desarrollo

Los planes a seguir con el proyecto incluyen:

- Desarrollar este proceso para compras y recetas de los doctores de la Fundación.
- Añadir diferentes algoritmos de curvas elípticas (ECIES, ECDH).

Se debe tener cuidado con las actualizaciones de las librerías y diferentes dependencias. Es por ello que se deben considerar futuras actualizaciones.

7. Ayuda y FAQs

7.1. Ayuda

Si surge alguna duda se puede encontrar información dentro del Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Monterrey.

U otra opción es encontrarnos por medio de cualquiera de estos correos electrónicos:

- paomont45@outlook.com
- salettegnv@hotmail.com
- mendezvillegas.sam@gmail.com
- leyre.capa24@gmail.com
- a01235009@tec.mx
- ethan_verduzco@hotmail.com

7.2. FAQs

- ¿En qué fecha fue la última modificación?

La última modificación se realizó el 26 de abril del 2022.

- ¿Existen restricciones de derechos de autor sobre el uso de esta solución?

Puede hacer lo que quiera con la fuente, siempre que se deje indicado los derechos de autor y se muestren en cualquier documentación. De esta forma se puede hacer un uso comercial, hacer copias o vender productos que involucren la solución dada.

- ¿Cómo funciona el esquema de numeración de versiones de las librerías?

Las versiones de las librerías están numeradas A.B.C o A.B. A es la versión principal, donde hay cambios significativos y B representa menores cambio, lo mismo sucede con C. Lo

- ¿La solución es compatible con otro Sistema operativo?

Si, posiblemente la mayoría de la solución funcione sin ningún problema instalando las diferentes dependencias y librerías, sin embargo, no se probó mas que con lo señalado en el documento.

- ¿Por qué tengo el error de “No module named pem”?

Es porque no se instaló de manera correcta la librería o se instaló dentro de otra carpeta. Para poder instalarla de nuevo esta o cualquier otra librería puedes buscar dentro de [https : //stackoverflow.com/](https://stackoverflow.com/). Otra de las soluciones es desinstalar e instalar de nuevo la librería cono pip unistall y pip install.

8. Anexos

8.1. Vectores de prueba

Algunos de los vectores de prueba ocupados fueron los siguientes:

SHA-1, mensaje = "sample"

k = 882905F1227FD620FBF2ABF21244F0BA83D0DC3A9103DBBEE43A1FB858109DB4
r = 61340C88C3AAEBEB4F6D667F672CA9759A6CCAA9FA8811313039EE4A35471D32
s = 6D7F147DAC089441BB2E2FE8F7A3FA264B9C475098FDCF6E00D7C996E1B8B7EB

SHA-224, mensaje = "test"

k = 669F4426F2688B8BE0DB3A6BD1989BDAEFFF84B649EEB84F3DD26080F667FAA7
r = C37EDB6F0AE79D47C3C27E962FA269BB4F441770357E114EE511F662EC34A692
s = C820053A05791E521FCAAD6042D40AEA1D6B1A540138558F47D0719800E18F2D

SHA-512, mensaje = "test"

k = 6915D11632ACA3C40D5D51C08DAF9C555933819548784480E93499000D9F0B7F
r = 461D93F31B6540894788FD206C07CFA0CC35F46FA3C91816FFF1040AD1581A04
s = 39AF9F15DE0DB8D97E72719C74820D304CE5226E32DEDAE67519E840D1194E55

SHA-384, mensaje = "sample"

k = 09F634B188CEFD98E7EC88B1AA9852D734D0BC272F7D2A47DECC6EBEB375AAD4
r = 0EAFEA039B20E9B42309FB1D89E213057CBF973DC0CFC8F129EDDDC800EF7719
s = 4861F0491E6998B9455193E34E7B0D284DDD7149A74B95B9261F13ABDE940954