Manual Software
Esquema de Secreto Compartido de Shamir

Sobre el programa

El programa “Esquema de Secreto Compartido de Shamir” fue desarrollado para implementar un método seguro de compartir una clave de cifrado que permite descifrar archivos confidenciales. Basado en el esquema de secreto compartido de Shamir, el programa divide la clave en múltiples fragmentos distribuidos entre los miembros autorizados. Estos fragmentos, llamados shares, son generados al evaluar un polinomio matemático que tiene como término independiente la clave secreta.

El programa garantiza que la clave pueda ser reconstruida únicamente cuando se reúna el número mínimo de fragmentos requeridos, manteniendo el secreto seguro incluso si algunos fragmentos se pierden o caen en manos no autorizadas. Esto se logra mediante la interpolación de los puntos proporcionados por los participantes.

Además, el software incluye la funcionalidad de cifrar el archivo original mediante el estándar AES y descifrarlo al recuperar la clave. Este enfoque asegura que la información confidencial quede protegida hasta que un grupo autorizado colabore para restaurar el acceso. El uso del programa debe realizarse con responsabilidad y de acuerdo con las normativas de seguridad de la información aplicables.

Legislación y lineamientos aplicables

Tratados Internacionales

Este proyecto se desarrolla bajo el marco de tratados internacionales enfocados en la protección de datos y ciberseguridad. específicamente:

1. **Convenio 108 del consejo de europa**:
   Este tratado establece estándares para la protección de datos personales en sistemas de tratamiento de información, del cual México es parte. Aunque este proyecto maneja claves y fragmentos más que grandes volúmenes de datos personales, cualquier implementación que contemple almacenamiento o transmisión de información debe adherirse a estas normas para garantizar la seguridad de los datos (consejo de europa, 1981).
2. **Reglamento general de protección de datos (gdpr)**:
   Si el proyecto estuviera disponible para ciudadanos de la unión europea, el gdpr regula estrictamente la gestión y protección de datos, asegurando que las prácticas cumplan con estándares europeos (unión europea, 2016).
3. **Tratado méxico-estados unidos-canadá (t-mec)**:
   El t-mec incluye disposiciones sobre protección de datos y comercio digital, promoviendo prácticas responsables en el uso de tecnologías y garantizando la privacidad de los usuarios (gobierno de méxico, 2020).
4. **Convenio de budapest**:
   Este tratado aborda ciberdelitos, como violaciones de privacidad y manipulación indebida de datos. El proyecto debe ser desarrollado y utilizado exclusivamente dentro de un marco ético y legal (consejo de europa, 2001).

Leyes Nacionales e Internacionales

1. **Ley federal de protección de datos personales en posesión de los particulares (lfpdppp)**:
   Regula el manejo de datos personales en méxico, asegurando la privacidad y protección de los derechos de los usuarios. Este proyecto cumple con estas disposiciones al garantizar que las claves y fragmentos no estén vinculados directamente con información identificable (congreso de la unión, 2010).
2. **Ley de protección al consumidor**:
   Asegurar que el software sea transparente y cumpla con estándares de calidad. En este proyecto, la interfaz es clara en sus funciones, y comprensibles para los usuarios (congreso de la unión, 1976).
3. **Ley de firma electrónica avanzada**:
   Si el sistema incorpora autenticación con firma digital, esta ley asegura su validez jurídica en méxico, garantizando la autenticidad e integridad de las transacciones electrónicas (congreso de la unión, 2012).
4. **Ley de seguridad nacional**:
   Regula el uso de tecnologías de cifrado, asegurando que no vulneren la seguridad nacional. Este proyecto utiliza aes como método de cifrado robusto y seguro, adherido a estándares internacionales (congreso de la unión, 2005).

Códigos Éticos

1. **Código de ética profesional del ingeniero en sistemas computacionales:** Subraya la importancia de desarrollar software seguro y responsable. En este proyecto, se aplican prácticas como la validación de entradas y el uso de bibliotecas confiables para garantizar la privacidad y seguridad de los datos (aniac, 2015).
2. **Código de conducta de la industria de software:** Promueve la transparencia en el manejo de datos y el cumplimiento de normativas legales. El esquema de shamir implementado sigue estos lineamientos mediante un diseño modular, seguro y confiable (aniac, 2015).

Reglamentos

1. **Reglamento de la ley federal de protección de datos personales en posesión de los particulares:** Regula cómo se deben proteger los datos durante su manejo y transmisión. En este proyecto, se emplea cifrado aes para garantizar la integridad y confidencialidad de las claves y fragmentos generados (congreso de la unión, 2011).
2. **Reglamento del instituto federal de telecomunicaciones (ift):** Establece normas para proteger la confidencialidad de los datos en telecomunicaciones. Si se transmite información entre usuarios o sistemas, el uso de protocolos seguros como https sería esencial para cumplir con estas disposiciones (ift, 2014).
3. **Reglamento de la ley de protección al consumidor:** Asegura la claridad en la presentación de los servicios digitales. Este proyecto sigue estas normas al ofrecer una interfaz intuitiva y documentación completa para garantizar la comprensión por parte de los usuarios (congreso de la unión, 2015).

Generalidades

Estructura del paquete del proyecto:

- **proyecto_3ESCS/**
  - **docs/**

    - Archivo.txt
- **src/**
    - Codificador.py
    - polinomio.py
    - Consola.py
    - Gestor.py
    - Lagrange.py
- **test/**
    - test_codificador.py
    - test_polinomio.py
    - test_lagrange.py
- **.gitignore**
- **README.md**
- **install.py**
- **requirements.txt**

Arquitectura del Sistema

La aplicación, desarrollada en Python, implementa una arquitectura sólida basada en el esquema de secreto compartido de shamir. El backend se encarga de toda la lógica matemática y criptográfica necesaria para dividir y recuperar un secreto, asegurando un flujo de trabajo eficiente y seguro.

El sistema opera directamente desde la terminal.

Ventajas de utilizar python:

- Su sintaxis clara y legible permitió un desarrollo ágil, especialmente para implementar cálculos matemáticos y operaciones criptográficas complejas.
- La disponibilidad de bibliotecas especializadas como numpy y cryptography redujo significativamente el tiempo de desarrollo, proporcionando herramientas optimizadas para manejar polinomios e implementar aes.
- Su compatibilidad multiplataforma asegura que el sistema pueda ejecutarse en cualquier entorno sin modificaciones significativas.
- La integración con herramientas de pruebas como pytest facilitó la validación y robustez del sistema.

Dependencias Externas

El programa utiliza dependencias esenciales para garantizar su correcto funcionamiento y seguridad:

- cryptography: proporciona herramientas para cifrar y descifrar archivos mediante el algoritmo aes (modo cbc), asegurando la protección de la información confidencial.
- pytest: permite realizar pruebas unitarias de los diferentes módulos. Este marco asegura que cualquier cambio en el código no afecte la funcionalidad del sistema, validando las operaciones matemáticas y de cifrado.
- rich: usado para la creación del programa en consola, su función es hacer más intuitivo y estético el programa. De esta manera, el usuario puede manejar mejor el software.

Interacción con el Usuario

El programa ofrece dos modos principales de operación, accesibles desde la terminal:

- modo cifrado: permite dividir un secreto en n fragmentos mediante un polinomio de grado t−1. El usuario proporciona la contraseña, el archivo a cifrar, y los parámetros t y n. El programa genera fragmentos y un archivo cifrado separados.
- modo descifrado: utiliza al menos t fragmentos válidos para reconstruir el secreto, recuperar la clave y descifrar el archivo.

Estos modos están diseñados para ofrecer una experiencia de usuario fluida y eficiente.

Para el Fácil Uso del Programa

El archivo consola.py gestiona todas las interacciones con el usuario. Este diseño modular facilita la extensión o modificación del programa según las necesidades del proyecto. La estructura está pensada para minimizar errores y garantizar una experiencia robusta, con validaciones claras y mensajes descriptivos en cada paso.

Responsabilidad del Usuario

El uso del programa es responsabilidad exclusiva del usuario. Cualquier uso indebido, como actividades ilegales o violaciones a la privacidad, está estrictamente prohibido. Este software está diseñado para:

- Proteger información confidencial mediante técnicas matemáticas y criptográficas avanzadas.

- Respetar las leyes y regulaciones locales e internacionales.

El desarrollador no se hace responsable de actos ilícitos derivados del uso del software. Se recomienda utilizarlo de manera ética y legal, siguiendo los lineamientos descritos en el apartado de legislación aplicable.

Público Objetivo

1. Usuarios finales:
    - Buscan proteger información confidencial mediante cifrado robusto y recuperación de secretos.
    - La interfaz está diseñada para ser sencilla e intuitiva, adecuada para usuarios sin experiencia técnica avanzada.
2. Profesionales de seguridad informática:
    - Interesados en probar y personalizar el esquema para fines académicos o de investigación.
    - El diseño modular permite extensiones y análisis avanzados de robustez.
3. Administradores o equipo técnico:
    - Responsables de garantizar la implementación, funcionalidad y mantenimiento del sistema.
    - Enfocados en cumplir con estándares de seguridad y rendimiento.

Tipo de Software

El software implementa el esquema de secreto compartido de shamir, permitiendo dividir un secreto en múltiples fragmentos mediante interpolación de polinomios y recuperar el secreto únicamente cuando se reúnen al menos t de los n fragmentos generados. El programa está desarrollado en python y utiliza bibliotecas como numpy para los cálculos matemáticos y cryptography para el manejo seguro de claves y cifrado aes. El usuario puede cifrar un archivo utilizando una clave generada a partir de una contraseña, dividir esta clave en fragmentos, o descifrar un archivo recuperando la clave a partir de los fragmentos.

el flujo principal incluye:

- Modo cifrado: el usuario proporciona un archivo, una contraseña y los parámetros t y n. el programa genera n fragmentos de la clave y cifra el archivo con aes.
- Modo descifrado: el usuario proporciona al menos t fragmentos y el archivo cifrado para recuperar la clave y descifrar los datos.

Requisitos Funcionales

1. División y reconstrucción del secreto:
    - El programa divide la clave en nnn fragmentos utilizando un polinomio de grado t−1.
    - Es capaz de reconstruir el secreto original mediante interpolación lagrange al recibir al menos t fragmentos válidos.
2. Cifrado y descifrado de archivos:
    - Permite cifrar archivos utilizando aes en modo cbc con una clave generada a partir de sha-256.
    - Descifra archivos previamente protegidos al recuperar la clave original desde los fragmentos.
3. Entrada de datos:
    - Para cifrar, se requiere un archivo de entrada (datos confidenciales), una contraseña, y los parámetros t (mínimos fragmentos necesarios) y n (fragmentos totales generados).
    - Para descifrar, se proporcionan al menos t fragmentos y el archivo cifrado.
4. Validación de datos:
    - verifica que t sea menor o igual a n y que los fragmentos proporcionados sean válidos.
    - valida la contraseña y el formato del archivo de entrada.

Requisitos No Funcionales

1. Rendimiento:
    - El programa utiliza numpy para optimizar los cálculos matemáticos y cryptography.
    - Está diseñado para operar de manera eficiente con archivos y fragmentos, incluso en configuraciones de gran tamaño.
2. Compatibilidad y Portabilidad:
    - Está desarrollado en python, lo que asegura compatibilidad con múltiples plataformas como windows, linux y macos.
    - El uso de bibliotecas ampliamente soportadas como pipenv facilita la gestión de dependencias y asegura un entorno controlado.
3. Seguridad:
    - El esquema garantiza que el secreto sólo puede reconstruirse si se cumplen las condiciones del umbral t.
    - La clave utilizada para el cifrado aes no se almacena directamente, protegiendo los datos incluso en caso de acceso a los fragmentos.

Enfoque

El enfoque del programa basado en el esquema de secreto compartido de shamir es permitir la división y recuperación de secretos mediante la interpolación de polinomios, garantizando que un secreto solo pueda reconstruirse si se reúnen al menos ttt de los nnn fragmentos generados. Además, incorpora un sistema seguro para cifrar y descifrar archivos utilizando aes, protegiendo información confidencial.

El software está dirigido a usuarios finales que buscan una herramienta sencilla para proteger y recuperar datos sensibles sin necesidad de conocimientos técnicos avanzados. gracias a su diseño modular y su interfaz basada en la terminal, los usuarios pueden:

- cifrar un archivo y generar fragmentos para compartirlos de manera segura.
- recuperar un secreto y descifrar el archivo utilizando los fragmentos necesarios.

Sin embargo, también está pensado para usuarios avanzados, como técnicos o desarrolladores, ya que el programa está escrito en python utilizando bibliotecas como numpy y cryptography, lo que facilita su extensión y personalización para casos de uso específicos. La modularidad del código permite a los desarrolladores ajustar y adaptar el sistema según sus necesidades.

El objetivo principal es proporcionar una herramienta confiable, eficiente y accesible para gestionar secretos y proteger datos confidenciales, combinando una experiencia de usuario sencilla con una implementación técnica robusta.

Posibles Mejoras

1. **Interfaz gráfica de usuario (gui):**
    - Implementar una gui amigable utilizando bibliotecas como pyqt o tkinter.
    - Desarrollar una versión web simple con frameworks como flask o django para ampliar la accesibilidad.

2. **Soporte para otros algoritmos de cifrado:**
    - Incorporar algoritmos de cifrado más avanzados, como cha-cha20 o rsa, además de aes, para ofrecer mayor flexibilidad en la seguridad del sistema.
3. **Modo de prueba y análisis de errores:**
    - Agregar un modo de prueba que permita a los usuarios verificar la reconstrucción del secreto y detectar posibles inconsistencias en los fragmentos o el cifrado.
4. **Optimización del rendimiento:**
    - Optimizar los cálculos polinómicos y el manejo de archivos para reducir el tiempo de ejecución al trabajar con grandes volúmenes de datos o fragmentos.
5. **Soporte multilingüe:**
    - Añadir soporte para múltiples idiomas en los mensajes de ayuda e instrucciones para atraer a una audiencia global.
6. **Módulo de integridad de datos:**
    - Implementar un sistema de verificación de integridad que confirme que los fragmentos y los archivos cifrados no han sido alterados antes de realizar operaciones.
7. **Personalización de parámetros:**
    - Permite a los usuarios definir parámetros avanzados, como el grado del polinomio o el método de generación de claves, para personalizar la configuración del sistema según sus necesidades.
8. **Mejor compatibilidad de archivos:**
    - Ampliar la compatibilidad con diferentes formatos de archivo, como documentos pdf o imágenes cifradas, para diversificar los casos de uso.
9. **Funcionalidad de previsualización:**
    - Incluye una previsualización del archivo descifrado antes de confirmar la operación, permitiendo al usuario verificar que los datos han sido recuperados correctamente.

Mantenimiento

1. **Actualización de bibliotecas:**
    - Mantener actualizadas las dependencias clave como numpy y cryptography para garantizar seguridad, compatibilidad y acceso a nuevas funcionalidades.

2. **Pruebas unitarias y de integración:**
    - Mantener un conjunto robusto de pruebas automatizadas con pytest para asegurar que las actualizaciones no rompan funcionalidades críticas, como la generación de fragmentos o la reconstrucción de secretos.
3. **Documentación técnica:**
    - Asegurar que la documentación esté actualizada e incluya guías claras de uso, instalación y descripciones de los módulos principales, facilitando el mantenimiento y la incorporación de nuevos desarrolladores.
4. **Monitoreo de seguridad:**
    - Realizar revisiones periódicas de las implementaciones de cifrado y manejo de claves para identificar y mitigar vulnerabilidades.
5. **Refactorización periódica:**
    - Revisar y mejorar la estructura del código para mantenerlo eficiente, legible y preparado para futuras expansiones.
6. **Soporte y resolución de problemas:**
    - Establecer un canal de comunicación con los usuarios para recopilar retroalimentación y resolver problemas rápidamente, fortaleciendo la confianza en el software.

Clases

Codificador

Atributos

- **__key :: bytes**
  Clave de cifrado generada a partir de la contraseña.

Métodos

- **Codificador()**
  Constructor que inicializa el objeto Codificador con una clave nula.
- **generaSha(string password)**
  Genera la clave de cifrado utilizando SHA-256 a partir de una contraseña.
- **leer_archivo(string nombre_archivo)**
  Lee el contenido de un archivo en formato binario y lo retorna en bytes.
- **guardar_archivo(string nombre_original, bytes data)**
  Guarda los datos cifrados en un archivo con la extensión .aes.

- **cifrar_archivo(string archivo_claro, string password)**
  Cifra un archivo utilizando AES en modo CBC y guarda el archivo cifrado.
- **shamir_generar_polinomio(int grado)**
  Genera un polinomio para Shamir's Secret Sharing con la clave como término independiente.
- **shamir_generar_puntos(Polinomio polinomio, int n)**
  Genera una lista de n puntos evaluando el polinomio en diferentes valores de x.
- **guardar_fragmentos(string archivo_cifrado, list puntos)**
  Crea y guarda un archivo con los fragmentos generados por Shamir.

Consola

Atributos

- **console :: Console**
  Objeto de la biblioteca rich para manejar la interacción con la terminal.

Métodos

- **titulo()**
  Muestra un panel inicial de bienvenida con el título "Iniciador de Tareas".
    - Panel: Contiene el título "Proyecto3" y el subtítulo "Bienvenido".
- **menuTexto(string titulo, list opciones)**
  Genera un menú dinámico con opciones numeradas, utilizando un diseño estilizado con la biblioteca rich.
    - Parámetros:
        - titulo :: string -> Título del menú.
        - opciones :: list -> Lista de opciones del menú.
- **opcion_tareas(list opciones, string seleccionado)**
  Ejecuta la opción seleccionada por el usuario.
    - Parámetros:
        - opciones :: list -> Lista de funciones asociadas a las opciones.
        - seleccionado :: string -> Opción ingresada por el usuario.
    - Retorno: "salir" si el usuario selecciona 0.
- **entrada_C()**
  Maneja la entrada de datos para cifrar un archivo.
    - Acciones:
        - Verifica el nombre del archivo, el rango de valores t y n, la existencia del archivo y la validez de la contraseña.
        - Muestra mensajes de error estilizados en caso de entradas inválidas.

- **entrada_D()**
  Maneja la entrada de datos para descifrar un archivo.
    - Acciones:
        - Solicita los nombres de los archivos necesarios para descifrar.
        - Verifica la validez de los archivos y sus extensiones.
- **consol()**
  Muestra el menú principal y gestiona las interacciones del usuario.
    - Acciones:
        - Ofrece las opciones de cifrar y descifrar archivos.
        - Maneja el flujo de ejecución dependiendo de la selección del usuario.

Clases y Funcionalidades Importadas

- **nombreCorrecto**
  Verifica que el nombre del archivo sea válido.
- **verifica_archivo**
  Comprueba la existencia del archivo en el sistema.
- **verificar_extension**
  Valida la extensión del archivo.
- **rangoValido**
  Asegura que los valores de t y n estén dentro del rango permitido.
- **verifiEntrada**
  Valida la contraseña ingresada por el usuario.

Módulo de Verificación(Gestor)

Atributos

No se utilizan atributos globales en este módulo. Todas las funciones son independientes y utilizan parámetros para la entrada de datos.

Funciones

1. **verifica_archivo(str ruta)**
   Verifica si el archivo especificado existe en la ruta dada (absoluta o relativa).
    - Parámetros:
        - ruta :: str -> Ruta al archivo.
    - Excepciones:
        - TypeError -> Si la ruta no es una cadena.
        - FileNotFoundError -> Si el archivo no existe.
        - Exception -> Si ocurre un error al leer el archivo.

- Retorno: Contenido del archivo como str.

2. **guardar_evaluacion(str nombre_archivo, list datos)**
   Guarda una lista de evaluaciones en un archivo dentro de la carpeta resultados.
   - Parámetros:
     - nombre_archivo :: str -> Nombre del archivo a guardar.
     - datos :: list[tuple] -> Evaluaciones en formato (x, y).
   - Retorno:
     - True -> Si el archivo se guarda correctamente.
     - False -> Si el archivo ya existía.
3. **verifiEntrada(str entrada)**
   Verifica que la entrada sea una cadena válida y cumpla con el límite de longitud.
   - Parámetros:
     - entrada :: str -> Entrada a verificar.
   - Excepciones:
     - TypeError -> Si la entrada no es una cadena.
     - ValueError -> Si la entrada excede los 255 caracteres.
4. **genera_eval(Polinomio polinomio, int n)**
   Genera las evaluaciones de un polinomio en un rango de valores desde 1 hasta n.
   - Parámetros:
     - polinomio :: Polinomio -> Polinomio a evaluar.
     - n :: int -> Número de evaluaciones.
   - Retorno: Lista de tuplas (x, y).
5. **nombreCorrecto(str nombre)**
   Verifica si el nombre del archivo cumple con el formato válido (sin extensión).
   - Parámetros:
     - nombre :: str -> Nombre del archivo.
   - Excepciones:
     - ValueError -> Si el nombre no cumple con las reglas establecidas.
6. **verificar_extension(str nombre_archivo)**
   Comprueba que el archivo tenga la extensión .eval.
   - Parámetros:
     - nombre_archivo :: str -> Nombre del archivo.
   - Excepciones:
     - ValueError -> Si la extensión no es .eval.
7. **rangoValido(int n, int t)**
   Valida que los valores n (evaluaciones) y t (puntos mínimos) cumplan con la regla 2 < t <= n.
   - Parámetros:
     - n :: int -> Número total de evaluaciones.
     - t :: int -> Número mínimo de puntos.

- Excepciones:
  - ValueError -> Si n o t no son enteros válidos o no cumplen la regla.

Lagrange

Atributos

- pares :: list
  Lista de pares ordenados (x,y)(x, y)(x,y) utilizados para construir y evaluar el polinomio de Lagrange.

Métodos

1. **Lagrange(list pares)**
   Constructor que inicializa el polinomio de Lagrange con una lista de pares ordenados.
   - Parámetros:
     - pares :: list -> Lista de al menos dos pares (x,y)(x, y)(x,y).
   - Excepciones:
     - ValueError -> Si la lista contiene menos de dos pares.
2. **calcula_Li(int i, float x=None)**
   Calcula el polinomio base Li(x)L_i(x)Li(x) o lo evalúa en un valor específico si xxx es proporcionado.
   - Parámetros:
     - i :: int -> Índice del par (xi,yi)(x_i, y_i)(xi,yi) en la lista.
     - x :: float -> (Opcional) Valor donde evaluar Li(x)L_i(x)Li(x).
   - Retorno:
     - Polinomio -> Si xxx no se proporciona.
     - float -> Si xxx es proporcionado.
3. **evalua(float x)**
   Evalúa el polinomio de Lagrange en un valor específico de xxx.
   - Parámetros:
     - x :: float -> Valor donde evaluar el polinomio.
   - Retorno:
     - float -> Valor del polinomio evaluado en xxx.
4. **genera_polinomio()**
   Genera el polinomio completo de Lagrange como un objeto Polinomio.
   - Retorno:
     - Polinomio -> Polinomio de Lagrange completo.

5. **_multiplica_monomios(list m1, list m2)**
   Multiplica dos listas de monomios y retorna el polinomio resultante.
   - Parámetros:
     - m1 :: list -> Lista de monomios.
     - m2 :: list -> Lista de monomios.
   - Retorno:
     - Polinomio -> Polinomio resultante de la multiplicación.

Clases: Monomio y Polinomio

Monomio

Atributos

- **coef :: float**
  Representa el coeficiente del monomio.
- **exp :: int**
  Representa el exponente del monomio.

Métodos

1. **Monomio(float coef, int exp)**
   Constructor que inicializa el monomio con su coeficiente y exponente.
   - Precondición:
     - coef y exp no pueden ser None.
   - Excepciones:
     - ValueError si coef o exp son None.
2. **suma(Monomio monomio)**
   Suma dos monomios si tienen el mismo exponente.
   - Retorno:
     - True si la suma fue realizada.
     - False si los monomios tienen diferentes exponentes.
3. **multiplicacion(Monomio monomio)**
   Multiplica dos monomios sumando exponentes y multiplicando coeficientes.
   - Retorno:
     - True si la multiplicación fue realizada correctamente.
4. **precondicion(float coef, int exp)**
   Verifica que el coeficiente y el exponente no sean None.
5. **__hash__()**
   Genera un hash único para el monomio.

6. **__eq__(Monomio other)**
   Compara si dos monomios son iguales (coeficiente y exponente).
7. **__str__()**
   Representa el monomio en forma de cadena matemática (ej. 2x^2, -3x).

Polinomio

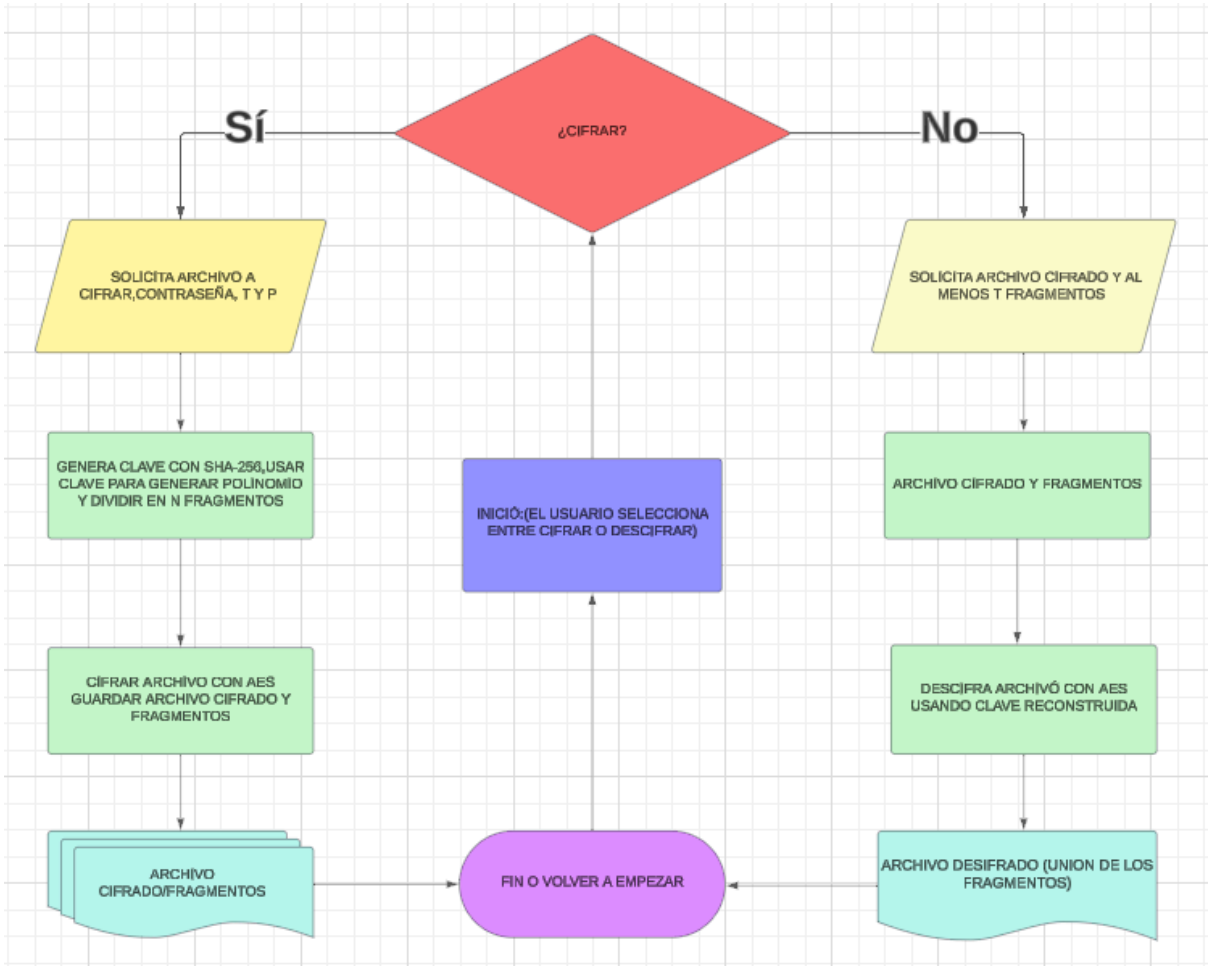
Atributos

- **monomios :: list[Monomio]**
  Lista de objetos Monomio que representan el polinomio.

Métodos

1. **Polinomio(list[Monomio] list_monomios)**
   Constructor que inicializa el polinomio con una lista de monomios.
   - Precondición:
     - Todos los elementos de la lista deben ser de tipo Monomio.
   - Excepciones:
     - TypeError si algún elemento no es un Monomio.
2. **simplificar(list[Monomio] list_monomios)**
   Combina monomios con el mismo exponente y elimina términos con coeficiente cero.
   - Retorno:
     - Una lista de monomios simplificada.
3. **evalua(float x)**
   Evalúa el polinomio en un valor específico de xxx.
   - Retorno:
     - float valor del polinomio evaluado.
4. **verifica(list[Monomio] list_monomios)**
   Verifica que todos los elementos de la lista sean instancias de Monomio.
5. **__str__()**
   Representa el polinomio en forma de cadena matemática (ej. 2x^3 + 3x^2 + 4).

Diagrama de flujo



Referencias

- Consejo de europa. (1981). *convenio para la protección de las personas con Respecto al tratamiento automatizado de datos de carácter personal*.
- Consejo de europa. (2001). *convenio de budapest sobre ciberdelincuencia*.
- Congreso de la unión. (1976). *ley federal de protección al consumidor*.
- Congreso de la unión. (2005). *ley de seguridad nacional*.
- Congreso de la unión. (2010). *ley federal de protección de datos personales en Posesión de los particulares*.
- Congreso de la unión. (2011). *reglamento de la ley federal de protección de Datos personales en posesión de los particulares*.
- Congreso de la unión. (2012). *ley de firma electrónica avanzada*.
- Congreso de la unión. (2015). *reglamento de la ley de protección al consumidor*.
- Gobierno de méxico. (2020). *tratado entre méxico, estados unidos y canadá (t-mec)*.
- Instituto federal de telecomunicaciones (ift). (2014). *reglamento de confidencialidad en telecomunicaciones*.
- Unión europea. (2016). *reglamento general de protección de datos (gdpr)*.