Manual de Procedimientos

Este manual de procedimientos está elaborado con el fin de ofrecer una orientación para la creación, conservación y actualización del proyecto de Esquema de Secreto Compartido de Shamir con un enfoque especial en la creación de una clave; la cual es de suma importancia, pues está elaborada para que descifre un archivo con información confidencial.

Propósito

El objetivo principal del proyecto del Esquema de Secreto Compartido de Shamir es crear un software capaz de implementar el esquema de Shamir para compartir la clave necesaria desencriptar archivos con información confidencial; así mismo como a partir de un documento u otro tipo de archivo con información confidencial se pueda generar una versión cifrada del mismo

---

Leyes y lineamientos

Esta sección establece las bases legales así mismo como las regulaciones que sustentan el progreso del proyecto de Esquema de Secreto Compartido de Shamir, el cual comprende las fases de análisis de documentos confidenciales, así como la extracción para la creación de una clave, así mismo como el descifrar dicho documento con la clave proporcionada. Se garantiza que el proyecto se adapte a las normas locales e internacionales importantes, así mismo como el establecimiento de los fundamentos para el desarrollo ético y responsable en este proyecto; así mismo como honrar y respetar la privacidad, propiedad de información y respetando los principios de transparencia. En la siguiente sección están los fundamentos del marco legal, acuerdos internacionales, códigos, leyes y normativas relacionados al proyecto.

Marco Juridico

Este proyecto basado en el esquema de Shamir se lleva a cabo bajo el marco legal internacional y nacional, respetando las normativas de protección de datos y seguridad de la información. Cumple con la **Ley Federal de Protección de Datos Personales en Posesión de los Particulares (LFPDPPP)** en México y el **Reglamento General de Protección de Datos (RGPD)** de la Unión Europea, garantizando el manejo adecuado, seguro y transparente de información sensible.

Asimismo, se asegura el cumplimiento de la **Ley Federal del Derecho de Autor**, empleando exclusivamente algoritmos y herramientas desarrollados o utilizados de manera legítima,

previniendo cualquier violación de derechos de autor o reproducciones no autorizadas de código o tecnología criptográfica.

Este marco jurídico sustenta la implementación del esquema de Shamir para garantizar la privacidad, seguridad y distribución responsable de datos confidenciales, alineándose con las normativas que rigen el uso ético y legal de tecnologías de encriptación.

Tratados Internacionales

El plan de manejo y protección de datos digitales concuerda con acuerdos internacionales fundamentales en la salvaguarda de la privacidad y la seguridad de la información. La **Convención de Budapest sobre la Ciberdelincuencia (2001)** establece directrices esenciales para combatir delitos cibernéticos, asegurando un tratamiento ético y seguro de los datos electrónicos, mientras que el **Acuerdo de Wassenaar (1996)** regula el uso de tecnologías de cifrado, garantizando su empleo responsable y protegiendo los flujos de información digital. Asimismo, se adopta como marco de referencia el **Reglamento General de Protección de Datos (GDPR)** de la Unión Europea, que refuerza la privacidad y el control sobre los datos personales, promoviendo estándares éticos y transparentes en la gestión de datos digitales.

Leyes

A escala nacional, las normativas vinculadas al manejo y protección de datos personales y la seguridad de la información son esenciales en este proyecto. En nuestra situación, se cumple con la **Ley General de Protección de Datos Personales en Posesión de Sujetos Obligados** en México, asegurando que cualquier información personal gestionada por entidades públicas o privadas adheridas al marco legal se maneje de manera ética y segura. Asimismo, se considera la **Ley de Seguridad Nacional** de los Estados Unidos, que establece medidas para proteger la información crítica y prevenir posibles vulnerabilidades en el ámbito digital, alineando este proyecto con los estándares de seguridad internacional y el uso responsable de datos electrónicos.

Códigos

En el ámbito tanto ético como profesional, podemos destacar el **Código Penal Federal** en materia de seguridad informática y manejo indebido de información confidencial. Aseguramos que este proyecto respeta los contenidos de información sensible, confidencial o de cualquier índole, empleando de manera cuidadosa los documentos que sean proporcionados de forma legal y autorizada.
Así mismo, se considera tanto la integridad del análisis, como en el procedimiento de

encriptación y desencriptación del mismo documento, sin modificación del mismo, para evitar y prevenir manipulaciones fraudulentas y preservando la imparcialidad en el debido procesamiento.

Reglamentos

1. Reglamento General de Protección de Datos (RGPD)

El programa implementado cumple con el Reglamento General de Protección de Datos (RGPD), normativa de la Unión Europea que regula la protección y privacidad de los datos personales. Los principios establecidos por el RGPD, como transparencia, limitación de finalidad, minimización de datos, seguridad y restricciones en la transferencia internacional de datos, son fundamentales en el manejo de información sensible.

En este contexto, el uso del esquema de Shamir asegura que las claves para descifrar archivos confidenciales se almacenen y distribuyan de manera segura, limitando su uso exclusivamente a los usuarios autorizados. Esto garantiza que la privacidad de los datos se mantenga intacta, incluso en casos de pérdida o robo de información parcial.

2. Seguridad de la Información en el Proyecto Shamir

Para garantizar la protección de la información en este proyecto, se implementan las siguientes medidas de seguridad:

- **Confidencialidad:**
  El archivo cifrado utiliza AES-256, uno de los algoritmos más seguros para la protección de datos, y las claves generadas a partir de contraseñas pasan por SHA-256 para asegurar su longitud y resistencia a ataques.
- **Integridad:**
  Los archivos generados incluyen verificaciones de integridad para asegurar que no se modifiquen durante el almacenamiento o la transferencia.
- **Disponibilidad:**
  El esquema de Shamir garantiza que los datos puedan ser recuperados siempre que se reúnan el número mínimo de evaluaciones requeridas, mitigando el impacto de la pérdida de fragmentos individuales.
- **Acceso restringido:**
  Solo los usuarios con acceso a las evaluaciones autorizadas del polinomio pueden recuperar el archivo claro, limitando el riesgo de acceso no autorizado.

Generalidades

¿Cómo está formado el sistema?

El sistema consta de dos módulos principales:

1. Cifrado: Convierte un archivo en un documento cifrado utilizando AES y divide la clave de cifrado en fragmentos basados en el esquema de Shamir.
2. Descifrado: Reconstruye la clave a partir de los fragmentos y descifra el archivo para recuperar su contenido original.

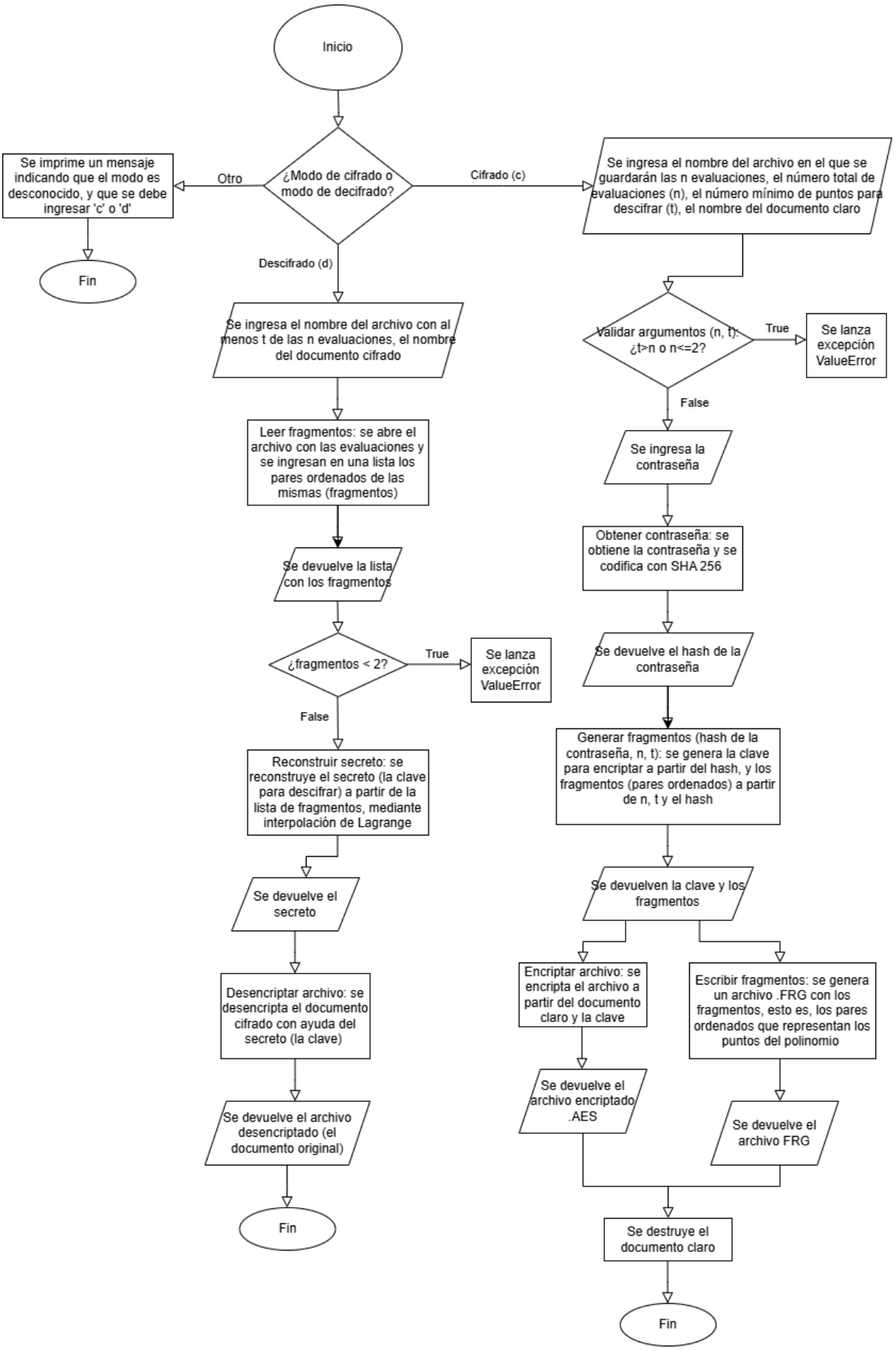
¿A quién está dirigido?
Este sistema está diseñado para organizaciones o personas que requieren compartir información confidencial entre múltiples partes autorizadas, como instituciones gubernamentales, empresas y organizaciones académicas.

Roles

- **Usuarios:** Miembros autorizados para recibir fragmentos de la clave y recuperar el archivo.
- **Administrador de contenido:** Persona(s) responsable(s) de generar y distribuir el archivo cifrado y los fragmentos, garantizando que la información pueda ser desencriptada. Asignados a: Rodríguez Jiménez Brenda, Sánchez Gonzalez Oscár Iván, Peña Villegas Diego Eduardo y Barrera Hernández Tomás.
- **Web Master:** Encargado de supervisar el progreso tanto técnico como de aplicación de los instrumentos del proyecto, así como el mantenimiento y actualizaciones del software de ser necesario. Este papel requiere garantizar que el sistema funcione eficazmente y logre los objetivos establecidos anteriormente. Asignado a: Peña Villegas Diego Eduardo.

Información del software

Software:
1. Linux Fedora 40 (64 bits)
2. Ubuntu (64 bits)
Requisitos para ejecución y pruebas unitarias:
Tener instalado:
1.Sympy
2. Pycryptodomex
3. Python
4. Pytest
Enfoque:
El programa está enfocado en la encriptación y descifrado de archivos confidenciales mediante una llave, con ayuda del esquema de Shamir.
Diagrama de flujo:


Inicio
¿Modo de cifrado o modo de decifrado?
Otro
Se imprime un mensaje indicando que el modo es desconocido, y que se debe ingresar 'c' o 'd'
Fin
Cifrado (c)
Se ingresa el nombre del archivo en el que se guardarán las n evaluaciones, el número total de evaluaciones (n), el número mínimo de puntos para descifrar (t), el nombre del documento claro
Validar argumentos (n, t): ¿t>n o n<=2?
True
Se lanza excepción ValueError
False
Se ingresa la contraseña
Obtener contraseña: se obtiene la contraseña y se codifica con SHA 256
Se devuelve el hash de la contraseña
Generar fragmentos (hash de la contraseña, n, t): se genera la clave para encriptar a partir del hash, y los fragmentos (pares ordenados) a partir de n, t y el hash
Se devuelven la clave y los fragmentos
Encriptar archivo: se encripta el archivo a partir del documento claro y la clave
Se devuelve el archivo encriptado .AES
Escribir fragmentos: se genera un archivo .FRG con los fragmentos, esto es, los pares ordenados que representan los puntos del polinomio
Se devuelve el archivo FRG
Se destruye el documento claro
Fin
Descifrado (d)
Se ingresa el nombre del archivo con al menos t de las n evaluaciones, el nombre del documento cifrado
Leer fragmentos: se abre el archivo con las evaluaciones y se ingresan en una lista los pares ordenados de las mismas (fragmentos)
Se devuelve la lista con los fragmentos
¿fragmentos < 2?
True
Se lanza excepción ValueError
False
Reconstruir secreto: se reconstruye el secreto (la clave para descifrar) a partir de la lista de fragmentos, mediante interpolación de Lagrange
Se devuelve el secreto
Desencriptar archivo: se desencripta el documento cifrado con ayuda del secreto (la clave)
Se devuelve el archivo desencriptado (el documento original)
Fin

Librerías usadas:

1.Para el programa:

-Sys: Se utiliza para interactuar con el sistema y gestionar argumentos de línea de comandos, en específico para leer los argumentos proporcionados a la hora de ejecución, facilitando la salida del programa en caso de errores.

-Os: Se utiliza para proporcionar funciones para interactuar con el sistema operativo, específicamente para separar la extensión del archivo permitiendo cambiarla de .txt a .aes durante el cifrado y de manera inversa.

-Random: Se utiliza para hacer coeficientes aleatorios del polinomio, de la misma manera que obtiene números enteros aleatorios dentro de un rasgo específico.

-Hashlib: Se utiliza para generar un hash SHA-256 de la contraseña proporcionada, para posteriormente usarla como término independiente del polinomio.

-Getpass: Se utiliza para la entrada segura de contraseñas desde la línea de comandos, ocultando los carácteres que son ingresados, así mismo asegurando que la contraseña del usuario no sea visible durante la entrada.

-Sympy: Se utiliza para crear una variable simbólica para representar el polinomio con "symbols", ayuda a generar un polinomio a partir de un conjunto e puntos (x,y) usando interpolación de Lagrange con "interpolate" y convierte el polinomio interpolador en una representación polinómica que puede evaluarse en un valor específico con "Poly".

-Pycryptodome: Se utiliza con la implementación de el modo CBC que usa un vector de inicialización (IV) para garantizar que el mismo texto plano que está cifrado múltiples veces produce resultados distintos, de igual forma con Cryptodome.Util.Padding se maneja el padding y la eliminación del mismo durante el cifrado y descifrado.

2.Para las pruebas unitarias:

-Unittest: Se utiliza como un marco estándar de pruebas en Python, el cual nos permite estructurar y ejecutar las pruebas de manera ordenada.

- Os: Se usa para interactuar con el sistema de archivos, ya sea crear, leer o eliminar archivos y trabajar con sus rutas; para las pruebas unitarias se usa para archivos temporales y comprobación de que existan.

-Pytest: Se está usando para hacer las pruebas unitarias en python, probando que el código se comporte conforme a lo esperado

-Hashlib: Se utiliza para trabajar con funciones hash como (SHA256), el cual es usado para la creación de un hash de contraseña.

-Tempfile: Se utiliza para trabajar con archivos temporales de manera segura y limpia, permitiendo escribir y leer fragmentos sin crear archivos permanentes en el sistema de archivos.

Funciones y clases:

Clase Shamir:

La clase contiene las funciones que trabajan con el polinomio, ya sea para crearlo o para interpolar los puntos (fragmentos). Utiliza las bibliotecas hashlib, random y sympy para generar los coeficientes, trabajar con el polinomio, calcular el término independiente y realizar la interpolación.

- **generar_fragmentos(contraseña: str, n: int, t: int):**

La función generar_fragmentos recibe la contraseña, el número total de fragmentos a generar ‘n’ y el número mínimo de fragmentos necesarios para reconstruir el secreto ‘t’.
Primero, convierte la contraseña a un número de 256 bits utilizando la función SHA256 de la biblioteca hashlib. Este número se transforma en un entero que representa el “secreto” que se desea proteger. Posteriormente, se genera aleatoriamente un polinomio de grado t−1, donde el término independiente corresponde al secreto y los demás coeficientes se crean con números aleatorios de 256 bits. Luego, se evalúa el polinomio en n valores distintos para producir los fragmentos. Cada fragmento se compone de un par (x,y), donde ‘x’ es el índice del fragmento y ‘y’ es el resultado de evaluar el polinomio en ‘x’. Finalmente, la función devuelve el secreto original (en formato de 32 bytes) junto con una lista de los fragmentos generados.

- **reconstruir_secreto(fragmentos):**

La función reconstruir_secreto recibe una lista de fragmentos, donde cada fragmento es un par (x,y) que representa un punto en el polinomio. Se requieren al menos ‘t’ fragmentos para poder reconstruir el secreto original. Primero, se valida que se proporcionen suficientes fragmentos. Luego, separa los fragmentos en dos listas: una con los valores de x y otra con los valores de y. A continuación, utiliza la función interpolate de la biblioteca sympy para reconstruir el polinomio original a partir de los fragmentos proporcionados. Después, evalúa el polinomio reconstruido en x=0, para obtener el término independiente, que corresponde al secreto buscado. Se convierte el secreto a un entero y luego a un arreglo de 32 bytes para finalmente retornarlo.

Clase cifrado_AES:

La clase contiene las funciones necesarias para cifrar y descifrar archivos utilizando el algoritmo AES (Advanced Encryption Standard) en modo CBC (Cipher Block Chaining). Utiliza las bibliotecas Cryptodome.Cipher para realizar las operaciones de cifrado/descifrado y Cryptodome.Util.Padding para manejar el relleno necesario en los bloques de datos.

- **encriptar_archivo(entrada_archivo, clave, ruta_carpeta):**

La función encriptar_archivo cifra el contenido de un archivo usando AES en modo CBC.
Primero, genera un objeto de cifrado AES en modo CBC con una clave proporcionada. El vector de inicialización (IV) se crea automáticamente al instanciar el objeto de cifrado. Luego, lee el contenido del archivo de entrada y lo rellena (pad) para garantizar que su tamaño sea un múltiplo del tamaño del bloque de AES (16 bytes). Después, cifra el contenido relleno y escribe el IV seguido del texto cifrado en un nuevo archivo. El archivo resultante tendrá la misma ruta base que el original pero con la extensión .aes. Finalmente, la función retorna la ruta del archivo cifrado generado.

- **desencriptar_archivo(entrada_archivo, clave, ruta_carpeta):**

La función desencriptar_archivo descifra el contenido de un archivo previamente cifrado con AES en modo CBC. Primero, lee el archivo cifrado para extraer el IV (los primeros 16 bytes) y el contenido cifrado restante. Luego, crea un objeto de descifrado AES en modo CBC utilizando la clave proporcionada y el IV extraído. Descifra el contenido y elimina el relleno

aplicado durante el cifrado, obteniendo así el texto original. Por último, escribe el contenido descifrado en un nuevo archivo con la misma ruta base pero con la extensión .txt. La función retorna la ruta del archivo descifrado generado.

Clase auxiliares:
La clase contiene funciones utilitarias para el manejo de contraseñas, fragmentos de datos, y validación de parámetros. Facilita operaciones como la generación de un hash SHA-256 para contraseñas, la lectura y escritura de fragmentos en archivos, y la validación de argumentos para asegurar su correcto uso.

- **obtener_contraseña():**

La función obtener_contraseña solicita al usuario una contraseña y devuelve su hash generado con el algoritmo SHA-256. Primero, utiliza la función getpass para solicitar la contraseña de forma segura, sin mostrar los caracteres ingresados en pantalla. Luego, convierte la contraseña ingresada en una secuencia de bytes y calcula su hash utilizando la biblioteca hashlib. Finalmente, retorna el hash como una cadena en formato hexadecimal, asegurando que la contraseña nunca se almacene en texto plano.

- **escribir_fragmentos(archivo_salida, fragmentos, ruta_carpeta):**

La función escribir_fragmentos escribe una lista de fragmentos en un archivo en formato CSV con extensión *.frg*. Abre el archivo indicado por archivo_salida en modo escritura y recorre la lista de fragmentos. Cada fragmento, representado como una tupla (x,y), se escribe en el archivo como una línea con los valores separados por comas. El archivo generado puede ser leído posteriormente para recuperar los fragmentos necesarios.

- **leer_fragmentos(archivo_entrada, ruta_carpeta):**

La función leer_fragmentos lee fragmentos de un archivo en formato CSV y extensión *.frg* y los devuelve como una lista de tuplas. Primero, abre el archivo especificado por archivo_entrada en modo lectura. Luego, procesa cada línea dividiéndola por la coma, convirtiendo los valores a enteros y almacenándolos como una tupla (x,y). Retorna una lista con todos los fragmentos válidos encontrados en el archivo.

- **validar_argumentos(n, t):**

La función validar_argumentos comprueba que los valores de n (número total de fragmentos) y t (número mínimo de fragmentos necesarios para reconstruir el secreto) cumplan las restricciones necesarias. Primero, verifica que $1 < t \leq n$ y que $n > 2$. Si alguna de estas condiciones no se cumple, lanza un error con un mensaje explicativo. Esta validación asegura que los parámetros proporcionados sean válidos para dividir y reconstruir un secreto usando el esquema de Shamir.

Clase Main: Administra el flujo del programa según los argumentos proporcionados por el usuario. Admite dos modos de operación:
Cifrar (c): Cifra un archivo y genera fragmentos necesarios para reconstruir la clave.

En este modo de operación del programa, el usuario tendrá que especificar siguiendo el siguiente el orden: el nombre del archivo de salida, el número de fragmentos que se crearán, el número mínimo de fragmentos necesarios para reconstruir la clave y el nombre del archivo que se quiere cifrar
Descifrar (d): Reconstruye la clave a partir de los fragmentos y descifra el archivo.
En este modo de operación, el usuario tendrá que especificar, siguiendo el siguiente orden: el nombre del archivo que contiene los fragmentos y el nombre del archivo a descifrar.

Procedimientos

Mantenimiento:

Para garantizar la seguridad, funcionalidad y escalabilidad a largo plazo del sistema basado en el esquema de Shamir, es esencial tener en cuenta los siguientes aspectos de mantenimiento:

- Validación de Bibliotecas Criptográficas: Verificar periódicamente que las bibliotecas de cifrado utilizadas (como AES y SHA-256) cumplan con los estándares actuales y actualizarlas según sea necesario para garantizar su resistencia a ataques modernos.
- Auditorías de Seguridad: Realizar revisiones periódicas del código para identificar vulnerabilidades y garantizar que se sigan las mejores prácticas de desarrollo seguro, especialmente en la generación y distribución de claves.
- Monitoreo de Polinomios Generados: Validar regularmente la integridad de los archivos que contienen las evaluaciones del polinomio, asegurando que las líneas no se corrompan o modifiquen durante su almacenamiento o transferencia.
- Optimización del Desempeño: Revisar y optimizar los algoritmos de generación de polinomios y cifrado para mantener un rendimiento eficiente, especialmente con grandes valores de n y t.
- Gestión de Dependencias: Documentar las versiones exactas de las dependencias utilizadas, como bibliotecas criptográficas y de manejo de archivos, para facilitar actualizaciones y resolver problemas de compatibilidad.

Actualizaciones:

Para mejorar la funcionalidad del sistema y mantenerlo relevante a las necesidades del usuario, se proponen las siguientes actualizaciones futuras:

- Soporte para Otros Algoritmos de Cifrado: Incorporar alternativas como ChaCha20 o Twofish, permitiendo a los usuarios elegir el método que mejor se adapte a sus necesidades de seguridad.

- Integración con Sistemas de Almacenamiento en la Nube: Habilitar el almacenamiento de los fragmentos generados y documentos cifrados en servicios como AWS, Google Drive o Nextcloud para facilitar su distribución y recuperación.
- Automatización del Proceso de Recuperación: Incorporar una función que identifique automáticamente los puntos necesarios para reconstruir la clave a partir de los fragmentos disponibles, simplificando el proceso de descifrado.
- Compatibilidad Multiplataforma: Ampliar el soporte del sistema para su uso en diferentes sistemas operativos, incluyendo Windows, macOS y distribuciones de Linux, con versiones empaquetadas y listas para instalar.
- Interfaces de Línea de Comando Mejoradas: Incorporar opciones más intuitivas y claras, con menús de ayuda interactivos que expliquen cada paso del proceso.
- Integración con Hardware de Seguridad: Compatibilidad con dispositivos de hardware como HSMs (Hardware Security Modules) o llaves USB para la generación y almacenamiento seguro de claves.

Aportaciones al manual:

En este manual de procedimientos se contendrá con información detallada y ejemplos prácticos para facilitar su comprensión y aplicación:

- **Documentación Detallada de Funciones:** Se incluirá una descripción exhaustiva de cada componente del sistema, desde la generación del polinomio hasta la implementación del cifrado y descifrado.
- **Guía de Escenarios Prácticos:** Se presentarán casos de uso específicos, como la generación de fragmentos para compartir en un grupo de trabajo, y la recuperación de la clave con diferentes combinaciones de puntos.
- **Ejemplos de Configuración:** Se detallará cómo configurar el sistema para diferentes escenarios, como cifrado de documentos grandes o manejo de valores personalizados para n y t.
- **Directrices de Seguridad:** Incluirá mejores prácticas para el manejo seguro de los fragmentos generados, como la encriptación adicional de los archivos .frg para su transferencia.
- **Solución de Problemas:** Una sección dedicada a la resolución de errores comunes, como problemas de compatibilidad con bibliotecas, archivos corruptos o configuración incorrecta de parámetros.
- **Actualizaciones Futuras:** Un apartado para documentar nuevas funciones añadidas al sistema, detallando los pasos para su instalación y uso, junto con una descripción de sus beneficios y limitaciones.

Bibliografía

1. *Protección de Datos conforme al reglamento RGPD - Your Europe*. (2022, 7 junio). Your Europe. https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_es.htm
2. *Código Penal Federal › Libro Segundo› Título Noveno - Revelación de Secretos y Acceso Ilícito a Sistemas y Equipos de Informática› Capítulo I - Revelación de Secretos*. (2024, 15 noviembre). Justia. https://mexico.justia.com/federales/codigos/codigo-penal-federal/libro-segundo/titulo-noveno/capitulo-i/#:~:text=de%20car%C3%A1cter%20industrial.-,Art%C3%ADculo%20211%20Bis,trescientos%20a%20seiscientos%20d%C3%ADas%20multa.
3. De Fomento Educativo, C. N. (s. f.). *Ley General de Protección de Datos Personales en Posesión de Sujetos Obligados*. gob.mx. https://www.gob.mx/conafe/documentos/ley-general-de-proteccion-de-datos-personales-en-posesion-de-sujetos-obligados-289438
4. *Objetivo 2.1: Proteger la seguridad nacional*. (2023, 7 junio). https://www.justice.gov/es/objetivo-21-proteger-la-seguridad-nacional
5. *Aviso de privacidad y seguridad | US EPA*. (2024, 31 octubre). US EPA. https://espanol.epa.gov/espanol/aviso-de-privacidad-y-seguridad#:~:text=La%20Ley%20de%20Privacidad%20del,o%20alg%C3%BAn%20otro%20identificador%20personal.)
6. Digitales, D. (2022, 16 mayo). *Convenio de Budapest sobre la Ciberdelincuencia en América Latina*. Derechos Digitales. https://www.derechosdigitales.org/18451/convenio-de-budapest-sobre-la-ciberdelincuencia-en-america-latina/
7. *Qué es el AW - The Wassenaar Arrangement*. (2024, 9 agosto). The Wassenaar Arrangement. https://www.wassenaar.org/es/about-us/
8. *La protección de datos en la UE*. (s. f.). Comisión Europea. https://commission.europa.eu/law/law-topic/data-protection/data-protection-eu_es
9. *Ley Federal del Derecho de Autor - CONAMER*. (s. f.). https://catalogonacional.gob.mx/FichaRegulacionid?regulacionId=5677
10. *Normativa y legislación en PDP – Marco Internacional de Competencias de Protección de Datos Personales para Estudiantes*. (s. f.). https://micrositios.inai.org.mx/marcocompetencias/?page_id=370
11. *Esquemas para compartir secretos*. (2018, 18 mayo). https://web.mat.upc.edu/jorge.villar/esamcid/rep/accs/reportaccessse2.html
12. *random —Generar números pseudoaleatorios — documentación de Python - 3.10.15*. (s. f.). https://docs.python.org/es/3.10/library/random.html
13. *hashlib — Hashes seguros y resúmenes de mensajes — documentación de Python - 3.10.15*. (s. f.). https://docs.python.org/es/3.10/library/hashlib.html?highlight=hashlib#module-hashlib

14. *tempfile — Generar archivos y directorios temporales — documentación de Python - 3.10.15*. (s. f.). https://docs.python.org/es/3.10/library/tempfile.html?highlight=tempfile#module-tempfile
15. *getpass — Entrada de contraseña portátil — documentación de Python - 3.10.15*. (s. f.). https://docs.python.org/es/3.10/library/getpass.html?highlight=getpass#module-getpass