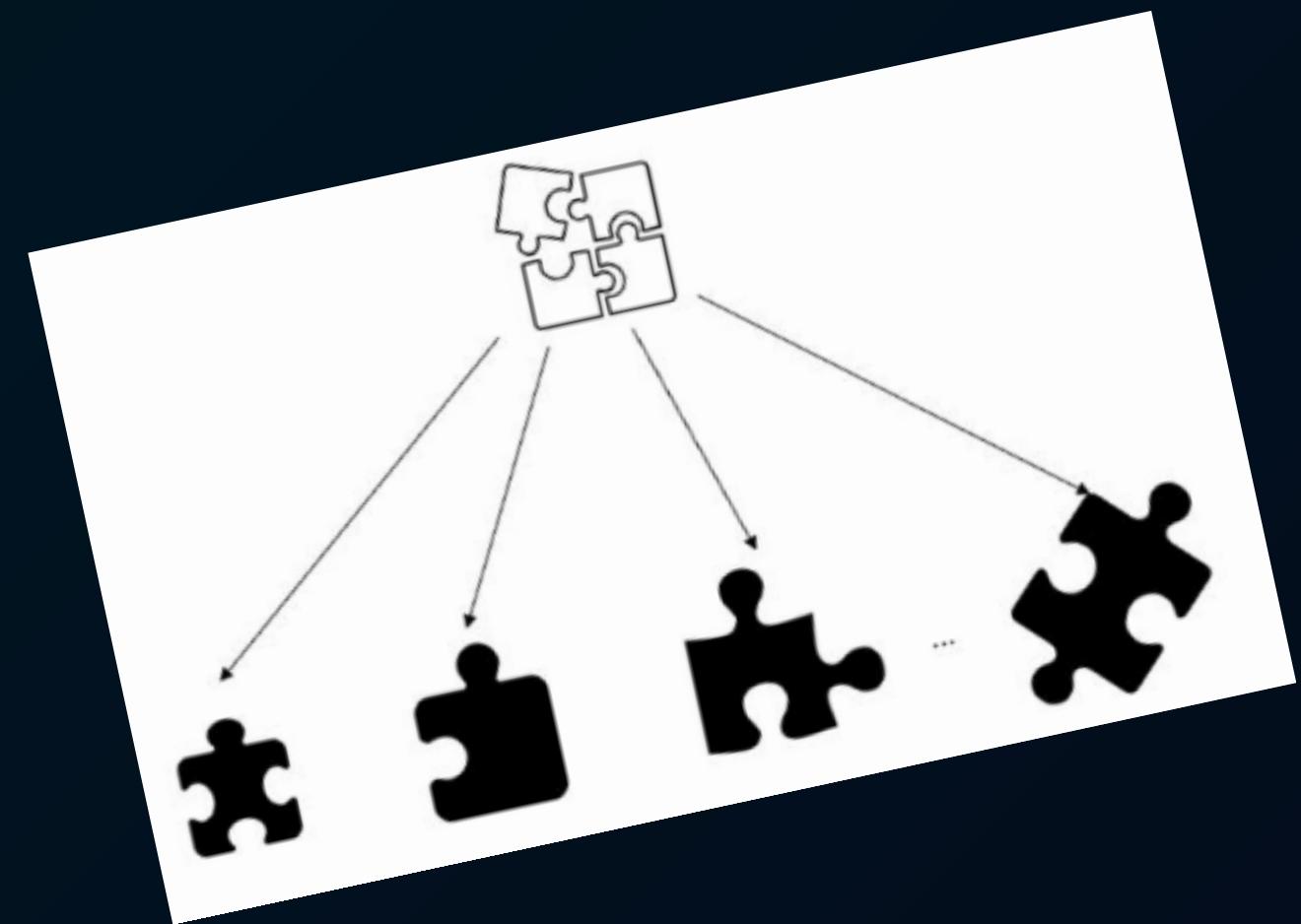


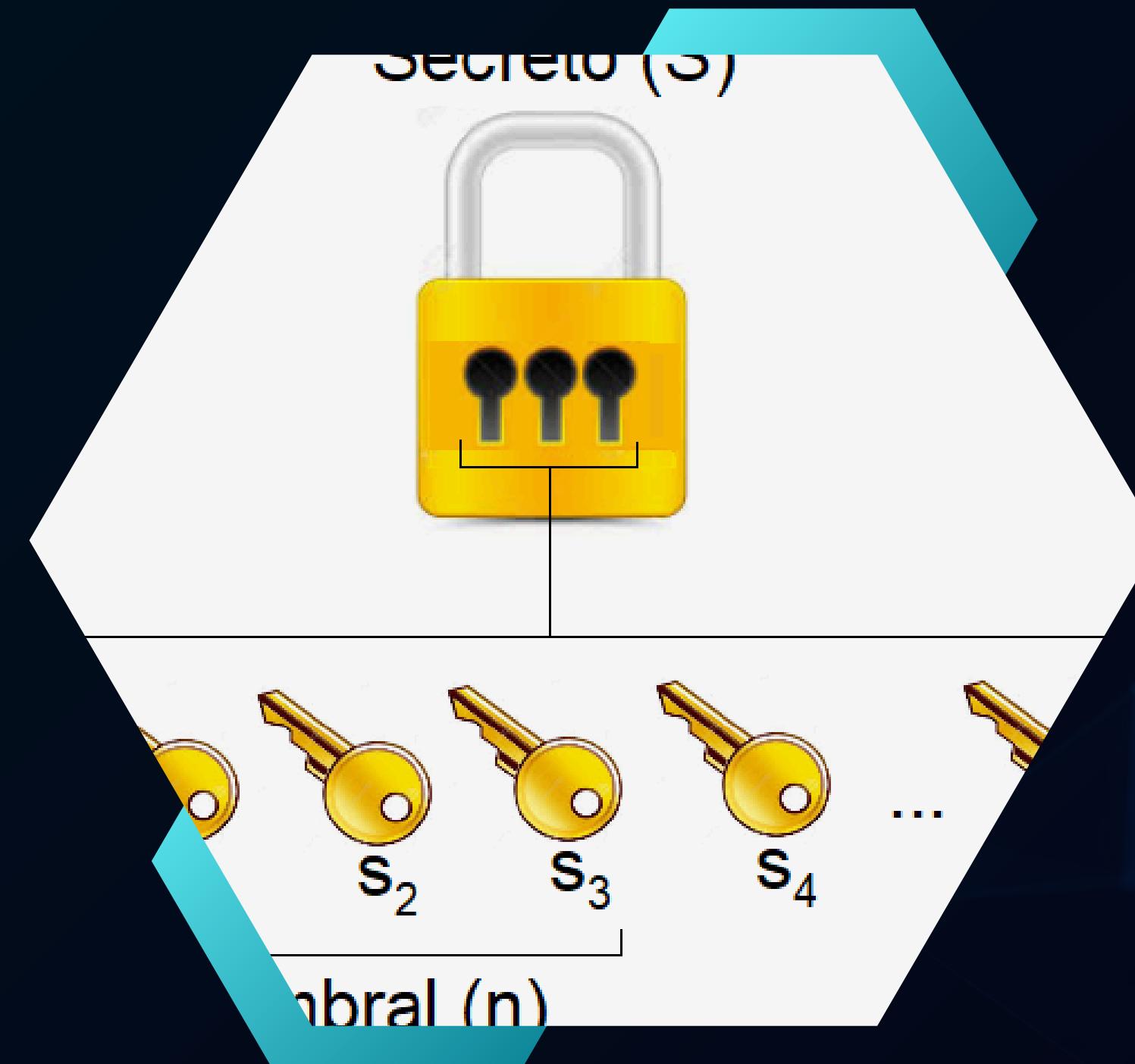
ESQUEMA DE SECRETO COMPARTIDO DE SHAMIR





CONTENIDO

- Definición del problema
 - Entendimiento del problema
 - Arsenal
 - Requisitos Funcionales
 - Requisitos No Funcionales
- Análisis del problema
- Selección de la mejora alternativa
- Pseudocódigo
- Costos y Mantenimiento

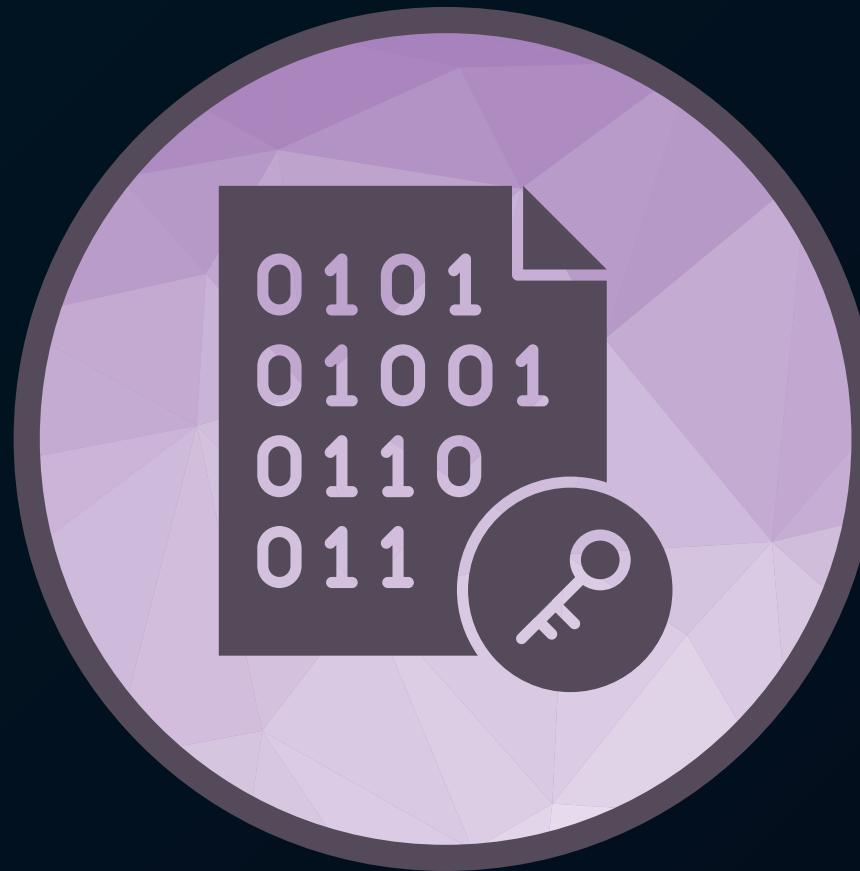




INTRODUCCIÓN

Este proyecto es una herramienta que permite a los usuarios cifrar y descifrar archivos de forma sencilla y segura. Utilizando algoritmos avanzados como AES-256, garantizando que los datos permanezcan protegidos frente a accesos no autorizados. Además, de integrar el esquema de secreto compartido de Shamir para gestionar claves de forma distribuida, asegurando un acceso controlado y colaborativo, todo sin comprometer la integridad o confidencialidad de la información.

DEFINICIÓN DEL PROBLEMA





ENTENDIMIENTO DEL PROBLEMA

La digitalización ha incrementado los riesgos de acceso no autorizado y robo de datos, afectando a individuos y organizaciones. Además la gestión centralizada de claves puede ser vulnerable si no se administra adecuadamente. Este proyecto enfrenta estos desafíos combinando cifrado avanzado con gestión distribuida de claves, asegurando la protección y control de datos sensibles.





ARSENAL



Java

Lenguaje de programación principal utilizado para desarrollar el proyecto, ideal para proyectos de manipulación de datos.



Maven

Herramienta de gestión de dependencias y construcción de proyectos que facilita la compilación y organización del código de manera eficiente.



java.security

Proporciona las herramientas para implementar algoritmos de hashing (SHA-256) y gestionar claves criptográficas.



ARSENAL



javax.crypto

Biblioteca que incluye las clases necesarias para el cifrado y descifrado utilizando AES-256, asegurando un manejo seguro de datos sensibles.



java.math

Utilizada para manejar números grandes y operaciones aritméticas necesarias en el esquema de Shamir.



java.nio.file

Facilita la lectura y escritura de archivos, optimizando el manejo de datos durante las operaciones de cifrado y descifrado.



ARSENAL



java.util

Herramienta clave para manejar estructuras de datos como listas y colecciones.



JUnit

Biblioteca incluida para la creación y ejecución de pruebas unitarias.



REQUISITOS FUNCIONALES

- 01** Permitir cifrar archivos de texto con AES-256 y generar un archivo cifrado y un archivo de contraseñas utilizando el esquema de Shamir.
- 02** Dividir la clave en fragmentos (n) con un mínimo necesario (t) para su recuperación.
- 03** Recuperar la clave original con al menos t fragmentos y descifrar el archivo cifrado para generar el archivo de texto original.
- 04** Ofrecer comandos para cifrar (-c) y descifrar (-d) archivos, validando entradas y solicitando contraseñas.
- 05** Proteger contraseñas con SHA-256 y asegurar los archivos generados contra accesos no autorizados.
- 06** Verificar la existencia y formato de los archivos, así como la coherencia de los parámetros informando de errores al usuario.



REQUISITOS NO FUNCIONALES



- 01** Utilizar algoritmos robustos y estándares internacionales como AES-256 y SHA-256.
- 02** El sistema debe procesar archivos en un tiempo razonable, independientemente de su tamaño.
- 03** La generación y recuperación de claves mediante el esquema de Shamir debe ser eficiente.
- 04** Proveer mensajes de error detallados y útiles en caso de entradas inválidas o fallas.
- 05** Ser compatible con archivos estándar y formatos de salida definidos (.aes y .frg).
- 06** Garantizar que las operaciones de cifrado, descifrado y manejo de claves sean precisas y consistente.

[Home](#)[Video](#)[About Us](#)[Contact](#)

ANÁLISIS DEL PROBLEMA

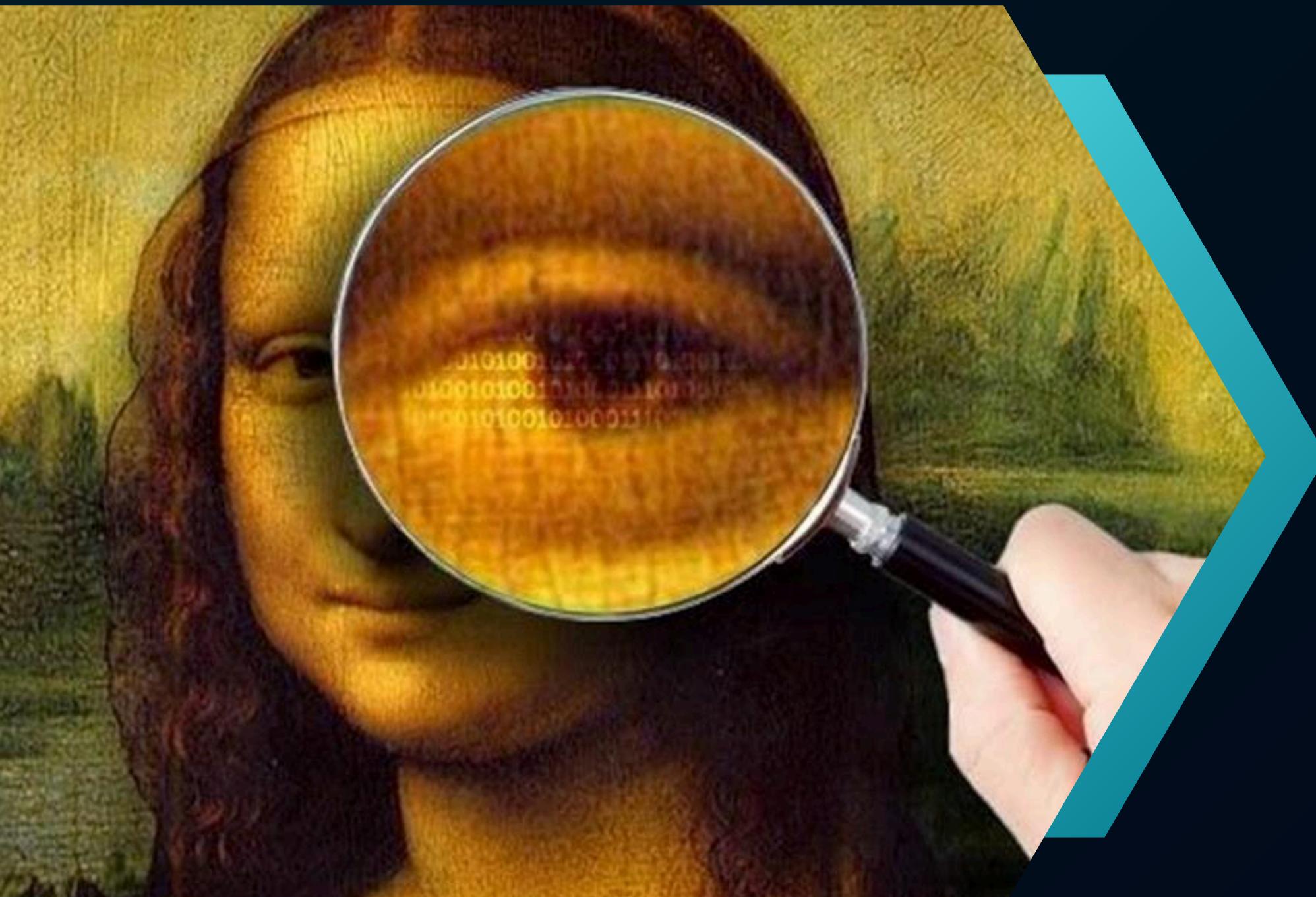




IDENTIFICACIÓN DEL PROBLEMA PRINCIPAL

- Necesidad de Protección de Datos Sensibles:
La creciente digitalización ha resaltado la importancia de proteger información confidencial contra accesos no autorizados e inseguridad, especialmente en sectores como salud, finanzas y gobierno.

- Riesgos Asociados a la Gestión Insegura de Claves:
El manejo de una sola clave puede ser un punto débil, ya que su pérdida o exposición compromete la seguridad de los datos.



DESAFÍOS TÉCNICOS

Implementar correctamente AES-256 para proteger datos contra accesos no autorizados.

Optimizar el cifrado y descifrado para manejar archivos grandes sin afectar la velocidad.

Integrar el esquema de Shamir para dividir y recuperar claves de forma precisa y eficiente.

Crear una línea de comandos clara y accesible para usuarios con diferentes niveles técnicos.



RIESGOS Y LIMITACIONES

- La pérdida o corrupción de los fragmentos de clave puede impedir el descifrado.
- Diseñado para archivos, los de gran tamaño pueden afectar el rendimiento así como la generación de muchas contraseñas.
- El archivo de contraseñas (.frg) debe protegerse adecuadamente.
- Un mal manejo de fragmentos de clave puede comprometer la seguridad o el acceso.



MEJOR OPCIÓN



- Dependencia de gestión centralizada de claves, aumentando riesgos de pérdida o accesos no autorizados.
- Procesos de cifrado y descifrado lentos para archivos de gran tamaño.
- Falta de integración de estándares modernos de seguridad y normativas de protección de datos.
- Uso de AES-256.
- Integración del esquema de Shamir para compartir secretos.
- Operaciones rápidas y eficientes incluso para archivos de mayor tamaño.
- Escalabilidad y flexibilidad para añadir nuevas funcionalidades en el futuro.



PSEUDOCODIGO

El Pseudocodigo completo de este proyecto podr encontrarlo en el Manual de Proyecto.

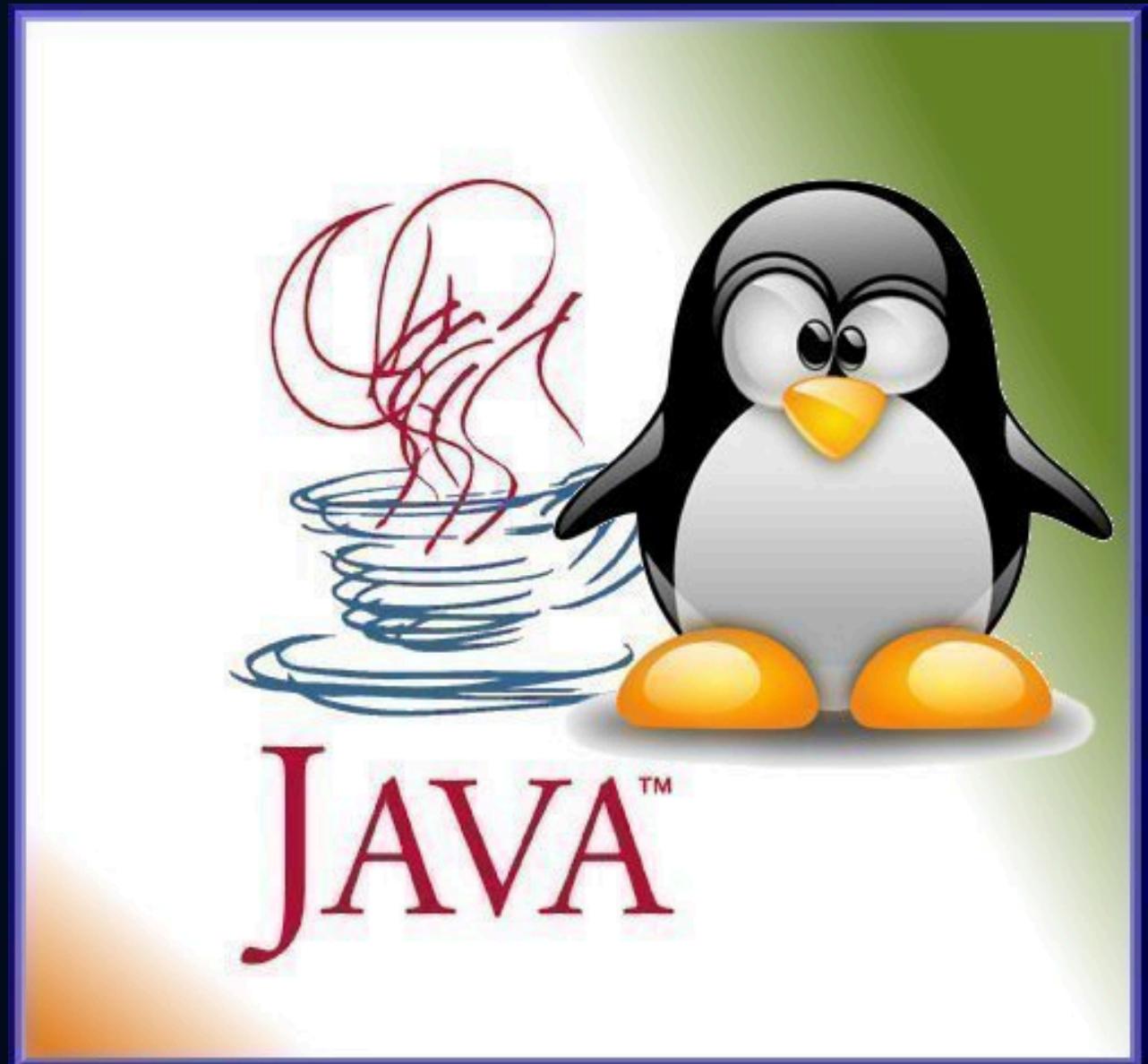
En esta seccion se describira lo que hace cada uno de los modulos del programa.





CLASE: MAIN

- Punto de entrada del programa.
- Funciones principales:
- Iniciar la ejecución leyendo los argumentos de la línea de comandos.
- Delegar el control al ProcesadorEntrada para interpretar y ejecutar los comandos.





CLASE: AES

- Implementa el cifrado y descifrado de archivos utilizando el algoritmo AES-256.
Funciones principales:
 - Cifrar un archivo con una clave generada a partir de una contraseña.
 - Descifrar un archivo utilizando la clave previamente compartida.
 - Gestionar la generación y validación de claves AES.



CLASE: SECRETOS DE SHAMIR

- Implementa el Esquema de Secreto Compartido de Shamir para dividir y recuperar claves.
- Generar un polinomio de grado $t-1$ con un término independiente que representa el secreto.
- Dividir el secreto en n fragmentos (puntos (x, y)).
- Recuperar el secreto utilizando interpolación de Lagrange a partir de al menos t fragmentos.





CLASE: COMANDO (INTERFAZ)

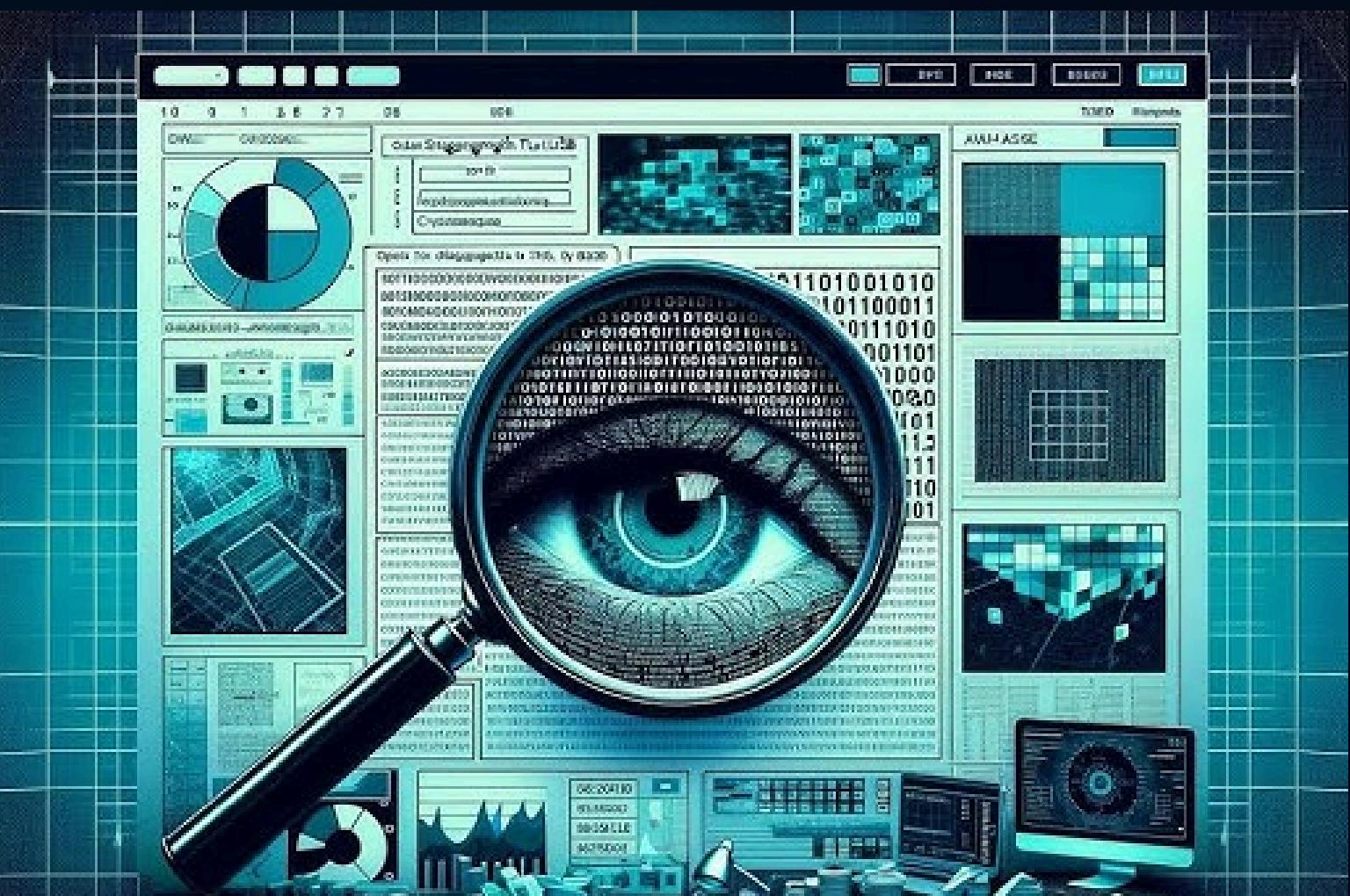


- Define la estructura básica para los comandos de cifrado y descifrado.
- Función principal:
- Establecer el método ejecutar() que será implementado por los comandos específicos.



CLASE: COMANDOCIFRAR

- Implementa el comando para cifrar un archivo.
- Funciones principales:
- Solicitar una contraseña al usuario y generar la clave correspondiente.
- Dividir la clave utilizando el esquema de Shamir.
- Cifrar el archivo con AES y guardar el resultado.





CLASE: COMANDO DESCIFRAR

- Implementa el comando para descifrar un archivo.
- Funciones principales:
- Recuperar la clave utilizando los fragmentos proporcionados.
- Descifrar el archivo con la clave recuperada.
- Validar que se cumpla el número mínimo de fragmentos requeridos (t).

$$l_i = \frac{x - x_0}{x_i - x_0} \times \dots \times \frac{x - x_{i-1}}{x_i - x_{i-1}} \times \frac{x - x_{i+1}}{x_i - x_{i+1}} \times \dots \times \frac{x - x_{k-1}}{x_i - x_{k-1}}$$

$$f(x) = \sum_{i=0}^{K-1} y_i l_i(x)$$



CLASE: PROCESADORENTRADA

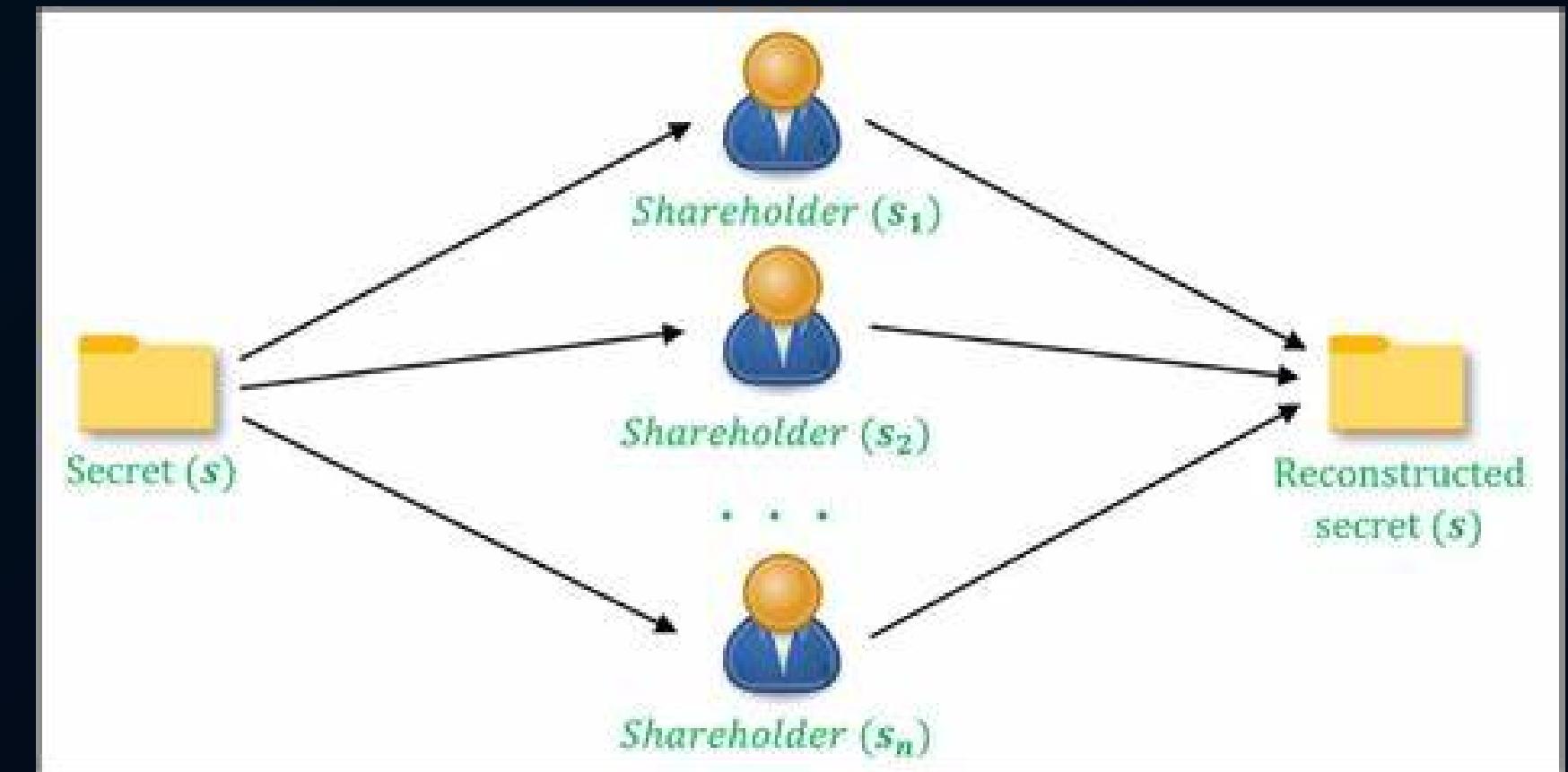
- Gestiona los argumentos de la línea de comandos y ejecuta los comandos correspondientes.
- Funciones principales:
- Interpretar y validar los argumentos proporcionados por el usuario.
- Crear y ejecutar instancias de ComandoCifrar o ComandoDescifrar según la bandera proporcionada.
- Manejar errores comunes como parámetros insuficientes o incorrectos.





CLASE: PROCESADORCONTRASEÑA

- Maneja la generación de claves a partir de contraseñas.
- Funciones principales:
- Derivar una clave segura mediante el algoritmo de hash SHA-256.
- Validar que las contraseñas cumplan con los requisitos de seguridad.



RESUMEN DEL PROYECTO

Este proyecto implementa un sistema criptográfico avanzado que combina el cifrado simétrico AES-256 con el Esquema de Secreto Compartido de Shamir, proporcionando una solución robusta para la protección de información sensible y la gestión de claves criptográficas. Su propósito principal es garantizar la seguridad de los datos mediante el cifrado de archivos y la división de la clave de cifrado en fragmentos, distribuidos entre varias partes confiables.

Objetivo del Proyecto

- Proteger datos confidenciales mediante un sistema de cifrado robusto.
- Facilitar la recuperación segura de claves en entornos colaborativos o distribuidos.
- Brindar una herramienta educativa y práctica para demostrar conceptos avanzados de criptografía.



COSTOS, MANTENIMIENTO Y ACTUALIZACIONES





MANTENIMIENTO

- Utilizar un sistema de control de versiones como Git para gestionar el desarrollo.
- Mantener la documentación actualizada y clara.
- Ejecutar pruebas automatizadas para garantizar la estabilidad del sistema.
 - Revisar periódicamente las dependencias para asegurarse de que estén actualizadas y seguras.





**Ademas, la validacion de entradas
y el manejo de errores deben
revisarse y mejorarse
continuamente para ofrecer una
experiencia de usuario mas
robusta.**





ACTUALIZACIONES

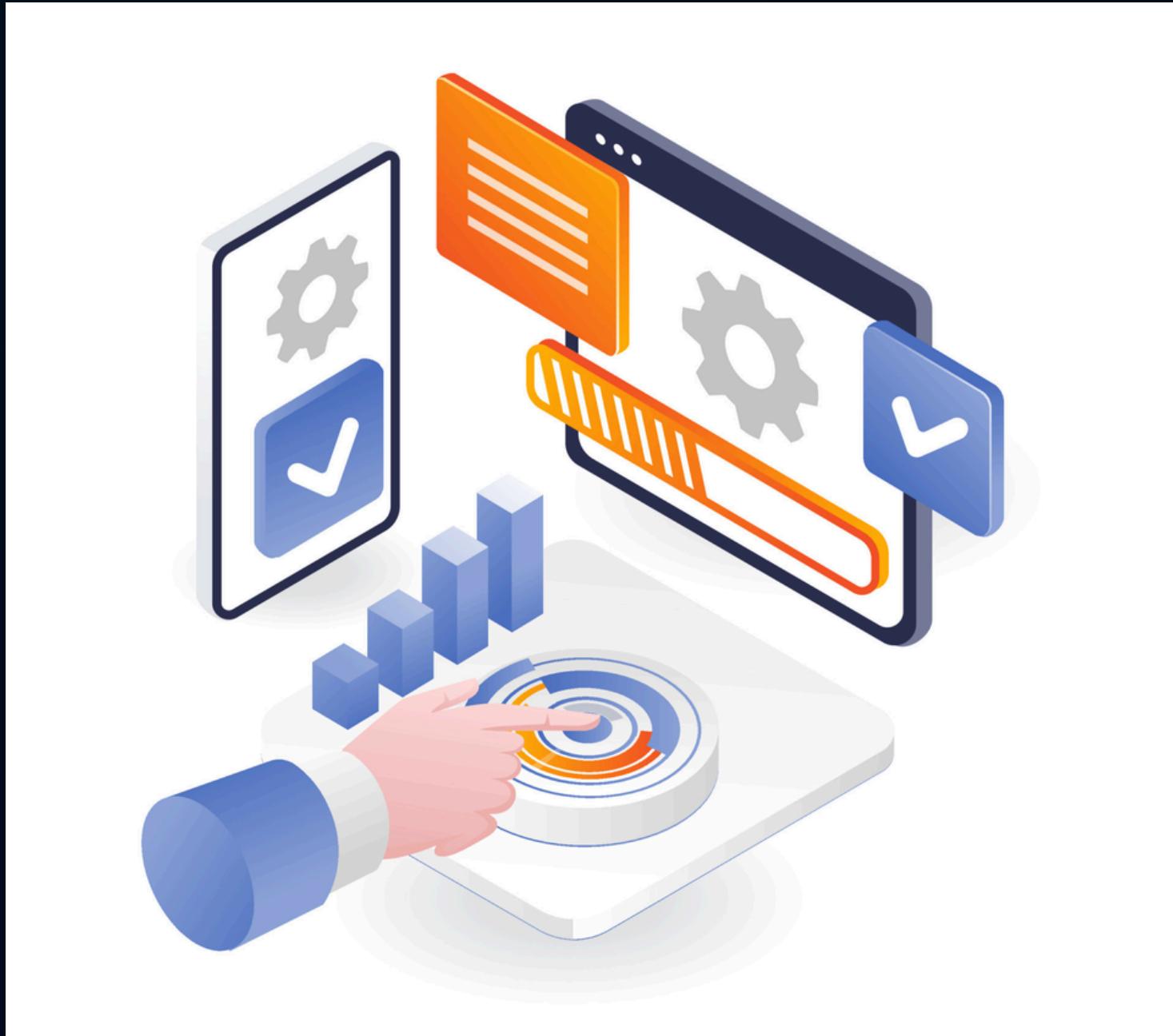
- 1. Soporte para más formatos de archivo:** Ampliar el soporte a documentos, imágenes y archivos comprimidos.
- 2. Cifrado híbrido:** Integrar algoritmos de cifrado asimétrico como RSA para mejorar la seguridad en la transmisión de claves.
- 3. Interfaz Gráfica de Usuario (GUI):** Facilitar el uso del programa mediante una interfaz visual amigable.
- 4. Almacenamiento distribuido de fragmentos:** Permitir guardar fragmentos en plataformas como la nube o redes descentralizadas.





ACTUALIZACIONES

- 5. Optimización del rendimiento:** Mejorar la eficiencia para manejar archivos grandes y fragmentos numerosos.
- 6. Internacionalización:** Agregar soporte multilingüe para una mayor accesibilidad.
- 7. Compatibilidad con hardware criptográfico:** Integrar dispositivos de seguridad física para gestionar claves.
- 8. Mejoras de seguridad:** Fortalecer la derivación de claves y agregar verificaciones de integridad mediante firmas digitales.





Estas actualizaciones se desarrollarán según las necesidades de los usuarios y la evolución tecnológica, garantizando que el software continúe siendo seguro, eficiente y relevante.





COSTOS

Dado el nivel de complejidad técnica del software, el costo estimado se calcula en función

del tiempo invertido y el valor del producto final. Con base en un rango estándar para

programadores con experiencia intermedia, el costo del proyecto se estima entre \$700 y

\$1,000 USD, considerando los siguientes factores:

1. Tiempo de desarrollo: 36 horas de trabajo efectivo.
2. Complejidad técnica: Implementación de cifrado AES-256 y Secreto

Compartido de
Shamir.

3. Aplicación práctica: Una herramienta funcional para la protección y gestión de datos sensibles.





PROPUESTA DE COSTOS DE MANTENIMIENTO Y ACTUALIZACIONES

- **Mantenimiento Preventivo:** Una tarifa mensual fija que cubre revisión de funcionalidades y corrección de errores menores. Propuesta: \$1,500 a \$4,000 MXN al mes.
- **Actualizaciones Menores:** Mejoras de la interfaz o funcionalidades básicas. Propuesta: \$500 a \$1,000 MXN por hora.
- **Actualizaciones Mayores:** Implementación de nuevas características como soporte para más formatos de imagen o cifrado avanzado. Propuesta: \$800 a \$1,500 MXN por hora, según la complejidad.