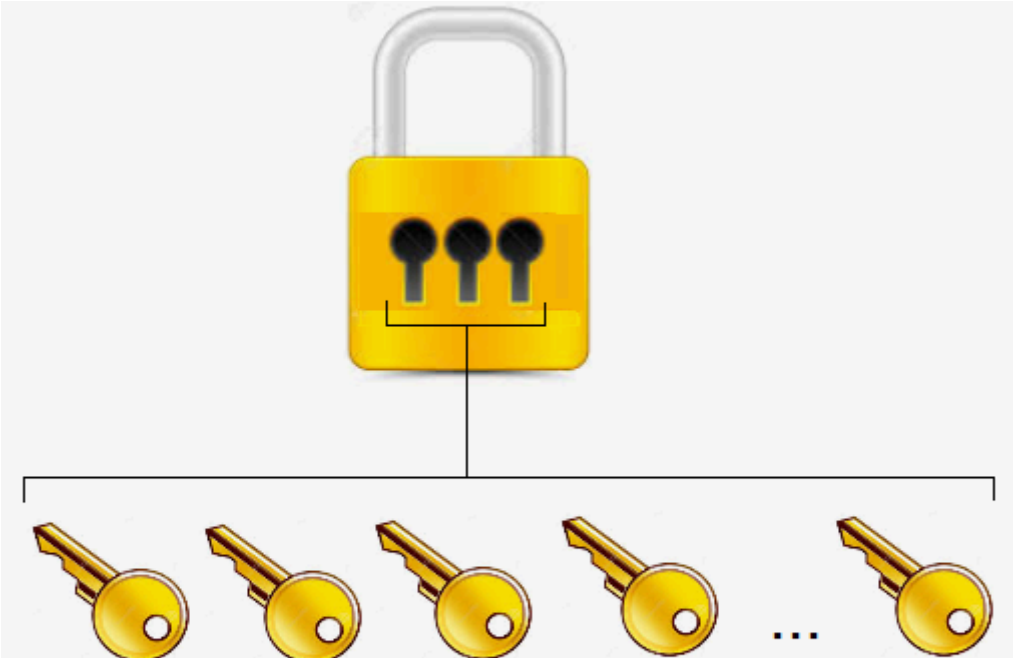
Manual

Esquema De Secreto Compartido De Shamir



Diaz Quijada Alan Joseph

Vega Alonso Diego Hazael

Índice.

Introducción

En la era digital, proteger la información sensible se ha convertido en una necesidad fundamental tanto para individuos como para organizaciones, es por esto que este proyecto surge como una solución robusta para proteger datos sensibles mediante técnicas avanzadas de criptografía. Este sistema implementa dos pilares fundamentales de la criptografía moderna: el esquema de secreto compartido de Shamir y la Advanced Encryption Standard (AES-256), lo que lo convierte en una herramienta ideal para garantizar tanto la confidencialidad como la integridad de la información.

Esquema de secreto compartido de Shamir.
El esquema de Shamir para compartir secretos es un método criptográfico innovador que permite dividir una contraseña o clave en múltiples fragmentos (también llamados participaciones). Este enfoque garantiza que el secreto original solo pueda ser reconstruido si se reúnen al menos un número mínimo de estos fragmentos (denotado como t), mientras que cualquier número menor de fragmentos no proporciona información útil. Este esquema es especialmente útil en situaciones donde la confianza en un único poseedor de la clave puede ser un riesgo, o donde es necesario distribuir el acceso entre varios participantes.

AES-256
El Advanced Encryption Standard (AES) es uno de los algoritmos de cifrado más seguros y ampliamente adoptados en la actualidad. AES-256, la variante más robusta del estándar, utiliza claves de 256 bits, proporcionando un nivel de seguridad prácticamente impenetrable con la tecnología computacional actual. El sistema utiliza AES-256 para cifrar archivos, transformando la información legible en un formato cifrado que solo puede ser descifrado con la clave correcta. En este proyecto, AES-256 trabaja en conjunto con el esquema de Shamir: primero, se genera una clave para cifrar el archivo; luego, esta clave es dividida en fragmentos mediante el esquema de Shamir, garantizando la seguridad de los datos.

El objetivo principal es combinar estos dos métodos criptográficos para crear una solución que permita cifrar archivos utilizando AES-256, protegiéndolos contra accesos no autorizados, del mismo modo que permita compartir la clave de cifrado de manera segura utilizando el esquema de Shamir. Este enfoque híbrido asegura que los datos estén protegidos tanto durante el almacenamiento como durante el acceso, ofreciendo una solución integral para la protección de la información en diversos entornos, desde aplicaciones personales hasta corporativos.

Lineamientos Generales

Este proyecto se desarrolla cumpliendo con una serie de leyes y reglamentos que garantizan con cumplir estándares y normativas en materia de protección de datos y uso de criptografía. El uso de herramientas de cifrado como AES-256 y esquemas de distribución de claves debe alinearse con las normativas de protección de datos en las regiones donde se implemente el sistema. Este apartado detalla los lineamientos legales, reglamentarios y normativos aplicables al proyecto. A continuación, se describen algunos de los principales:

-Ley Federal de Protección de Datos Personales en Posesión de los Particulares (LFPDPPP) - México

Es el marco legal clave que rige el tratamiento de datos personales por parte de empresas y particulares en México. Este proyecto, está alineado con los principios y disposiciones de la ley, ya que proporciona herramientas técnicas que permiten cumplir con los requisitos de protección de datos. Sus principales disposiciones relevantes para este sistema incluyen:

- **Artículo 19**: Obliga a los responsables a implementar medidas de seguridad técnicas y administrativas para proteger los datos personales (como el uso de tecnologías de cifrado).
- **Artículo 21**: Establece que los responsables deben garantizar la confidencialidad de los datos personales, incluso después de terminada su relación con el titular.

Por otra parte, la LFPDPPP otorga a los titulares de datos los derechos **ARCO** (Acceso, Rectificación, Cancelación y Oposición). El proyecto respalda el cumplimiento de estos derechos al facilitar:

-Acceso: Los titulares pueden solicitar su información cifrada, la cual puede ser descifrada únicamente con las claves necesarias.

-Rectificación: Si los datos necesitan ser actualizados o corregidos, el responsable puede realizar el proceso después de descifrarlos de forma segura.

-Cancelación: Los archivos cifrados pueden ser eliminados de manera definitiva, y las claves correspondientes pueden invalidarse.

-Oposición: El uso del sistema garantiza que los datos no sean accesibles a terceros si el titular decide ejercer su derecho de oposición.

Aunque el proyecto está enfocado en México, es importante tener en cuenta las normativas internacionales que impactan la protección de datos. El uso de criptografía y distribución de claves en este proyecto también está alineado con estándares internacionales:

-Reglamento General de Protección de Datos (GDPR) - Unión Europea: Esta normativa de la Unión Europea protege los datos personales de los ciudadanos europeos y es aplicable si el proyecto se llega a extender a dicha región, ya que implementa medidas técnicas como el cifrado AES-256 y el esquema de Shamir, garantizando la confidencialidad y seguridad de los datos personales frente a accesos no autorizados. Además, facilita el cumplimiento de derechos de los titulares, como acceder, modificar o eliminar sus datos, proporcionando una herramienta eficaz para cumplir con los estándares de privacidad y responsabilidad en la Unión Europea.

-Decretos y Reglamentos Locales: Dependiendo de la región o país donde se despliegue la aplicación, se deberán cumplir los decretos y regulaciones locales específicas sobre protección de datos y transparencia. Esto puede incluir leyes adicionales sobre la gestión de información sensible y la ciberseguridad.

-Usos permitidos y Restricciones:

El cifrado y distribución de claves deben utilizarse exclusivamente para fines legales y éticos:

- Usos Permitidos:
    - Protección de datos sensibles en instituciones públicas o privadas.

    - Implementación de seguridad en sistemas personales o corporativos.
    - Cumplimiento de requisitos legales y normativos en materia de privacidad.
- Restricciones:
    - Queda estrictamente prohibido el uso del sistema para actividades ilícitas, como el cifrado de datos con fines de extorsión (ransomware) o espionaje.
    - El proyecto debe respetar las leyes de cada jurisdicción en cuanto a la implementación y exportación de tecnologías criptográficas.

-<u>Responsabilidades del Usuario</u>:

Los usuarios del sistema son responsables de:

- Cumplir con las leyes de su jurisdicción relacionadas con la protección de datos y el uso de criptografía.
- No divulgar de manera indebida los fragmentos de clave generados mediante el esquema de Shamir.
- Adoptar medidas adicionales para proteger los archivos generados (por ejemplo, el archivo .aes y el archivo .frg).

Este marco de lineamientos asegura que el sistema no solo cumpla con las normativas legales, sino que también promueva un uso ético y responsable de la tecnología.

Generalidades

Estructura

El proyecto ha sido diseñado con una arquitectura modular que permite su fácil comprensión, mantenimiento y escalabilidad. Está dividido en componentes que trabajan en conjunto para ofrecer cifrado robusto y gestión segura de claves, con una implementación clara. A continuación, se describe la estructura del proyecto en detalle.

1. Componentes Clave del Proyecto

Clases de Cifrado y Gestión de Claves:

Estas clases implementan la lógica principal del sistema relacionada con la protección de datos y el manejo de claves de acceso:

ProcesadorContraseña:

Encargada de generar un hash SHA-256 a partir de una contraseña proporcionada por el usuario. Este hash se utiliza como base para las claves de cifrado, asegurando que las contraseñas sean seguras.

AES:

Implementa el cifrado y descifrado de archivos utilizando el estándar AES-256.
Proporciona métodos robustos que convierten archivos de texto en datos cifrados y también descifra.

SecretoShamir:

Divide una clave generada en múltiples fragmentos utilizando el esquema de secreto compartido de Shamir. Permite configurar el número total de fragmentos generados (n) y el número mínimo necesario (t) para recuperar la clave. Este enfoque garantiza que el acceso a la información sea controlado y distribuido entre múltiples participantes.

Gestión de Comandos:

Comando:

Interfaz que define la estructura básica de un comando con el método ejecutar().

ComandoCifrar:

Implementa la lógica para cifrar archivos. Utiliza las clases ProcesadorContraseña, AES y SecretoShamir para generar los archivos cifrados y las contraseñas distribuidas.

ComandoDescifrar:

Responsable de descifrar un archivo cifrado. Verifica la existencia de los fragmentos de clave necesarios y los utiliza para recuperar el archivo original.

Procesamiento y Coordinación:

Estas clases hacen el flujo del sistema y procesan las entradas del usuario:

ProcesadorEntrada:

Gestiona los argumentos proporcionados por el usuario en la línea de comandos.
Verifica la validez de los parámetros y ejecuta el comando correspondiente (ComandoCifrar o ComandoDescifrar). Maneja errores relacionados con el uso incorrecto del sistema y muestra mensajes descriptivos para guiar al usuario.

Main:

Es el punto de entrada del programa. Su función principal es inicializar el sistema.

Pruebas Unitarias:

Para garantizar la calidad del proyecto, se implementaron pruebas unitarias que validan cada componente crítico:

SecretoShamirTest:

Comprueba la funcionalidad del proyecto.

Clases de Apoyo para Calificación y Reporte:

Estas clases facilitan la organización y reporte de las pruebas:

Calificador:

Clase base que categoriza y califica las pruebas unitarias, permitiendo medir la efectividad y calidad del sistema.

Categoria:

Representa una categoría dentro del sistema de calificación, asociando pruebas específicas con sus respectivas ponderaciones.

Impresora:

Ayuda a generar reportes claros y formateados sobre los resultados de las pruebas unitarias.

2. Organización de Archivos

El proyecto sigue una estructura de carpetas estándar para proyectos Java gestionados con Maven:

Código Fuente (src/main/java):

Contiene las implementaciones principales del sistema en el paquete mx.unam.criptografia.

Pruebas (src/test/java):

Contiene las pruebas unitarias que validan la funcionalidad del sistema, organizadas en el mismo paquete.

Archivos de Configuración:

pom.xml:

Define las dependencias necesarias, como JUnit para pruebas, y los plugins para compilar y empaquetar el proyecto como un JAR ejecutable.

Dirigido para

El proyecto está diseñado para ser utilizado por una amplia variedad de usuarios y organizaciones que buscan proteger información sensible.

1. Profesionales y Organizaciones que Manejan Datos Sensibles
   - Sector financiero: Protección de información bancaria, transacciones y datos de clientes.

   - Instituciones gubernamentales: Protección de documentos confidenciales relacionados con políticas, investigaciones y procesos legales.

2. Individuos con Necesidades de Seguridad Avanzada
   - Profesionales independientes: Abogados, contadores y consultores que necesitan proteger datos de clientes o documentos legales.

   - Investigadores y académicos: Protección de resultados de investigaciones, datos estadísticos antes de su publicación.

   - Usuarios generales conscientes de la privacidad: Personas que buscan asegurar información personal, como contraseñas, archivos confidenciales o comunicaciones sensibles.

3. Aplicaciones en Escenarios Colaborativos
   - Equipos distribuidos: En proyectos donde la gestión de acceso a datos sensibles requiere la colaboración de varios participantes.

   - Administración de claves compartidas: Situaciones donde se necesita asegurar que ningún individuo tenga control exclusivo sobre una clave de cifrado, como en instituciones bancarias o gubernamentales.

Roles

El desarrollo de este proyecto fue posible gracias a la colaboración de los siguientes integrantes:

-Alan Joseph Diaz Quijada - Desarrollador, Tester y ayudó a la arquitectura del proyecto: Ayudó al diseño de la estructura general del sistema, asegurando una organización modular y eficiente. Implementó las funcionalidades principales de cifrado y descifrado utilizando AES-256 y contribuyó en la corrección del esquema de secreto compartido de Shamir, garantizando la robustez del sistema frente a accesos no autorizados.

-Diego Hazael Vega Alonso - Desarrollador, Testing y ayudó a la arquitectura del proyecto: Encargado de la implementación de las pruebas unitarias para validar la correcta funcionalidad del proyecto. Desarrolló métodos para gestionar errores y asegurar que las operaciones de cifrado y descifrado se ejecutarán conforme a los estándares de calidad, así mismo implementó el funcionamiento del secreto compartido de Shamir, así como del procesamiento de la entrada del usuario y contraseña.

En caso de que el proyecto sea implementado, los roles en el sistema serían:

- Usuarios en general: Personas o corporaciones que necesiten proteger información confidencial mediante el cifrado de archivos y gestión de claves distribuidas para compartir accesos de forma segura.
- Web Master: Responsable del funcionamiento técnico completo del sistema. También deberá encargarse de la resolución de problemas técnicos y el mantenimiento del sistema.
- Desarrollador: Persona encargada de realizar mejoras continuas en el código, corregir errores y añadir nuevas funcionalidades basadas en las necesidades de los usuarios o cambios en las normativas de protección de datos.

Información de software

Tipo de software

El programa desarrollado es una **herramienta criptográfica multiplataforma** orientada al cifrado y descifrado de archivos mediante el algoritmo simétrico **AES-256**. Además, incluye una implementación avanzada del **Esquema de Secreto Compartido de Shamir**, permitiendo dividir una clave de cifrado en fragmentos distribuidos. Esta herramienta está diseñada para ser ejecutada desde la línea de comandos (CLI), ofreciendo una solución práctica y segura para proteger información sensible.

Está dirigida principalmente a **profesionales de TI**, **investigadores** y **administradores de sistemas** que buscan una solución confiable y personalizable para la gestión de datos cifrados y secretos compartidos. Gracias a su implementación en Java, es compatible con los principales sistemas operativos y puede integrarse fácilmente en flujos de trabajo existentes.

Requerimientos

Para ejecutar este software, es necesario cumplir con los siguientes requisitos:

Hardware:

- Procesador Intel/AMD x86-64 o ARM con soporte para Java.
- Mínimo 512 MB de RAM, 1 GB o más recomendado.
- Espacio libre en disco: 50 MB para instalación, más espacio según el tamaño de los archivos cifrados/descifrados.
- Sistema operativo compatible: Windows, macOS o Linux.

Software:

- **Java Development Kit (JDK) 8** o superior instalado y configurado en el sistema.
- Opcional: Apache Maven para compilar el código fuente.

Conocimientos Requeridos:

- Uso básico de línea de comandos.
- Familiaridad con conceptos criptográficos como cifrado AES y secreto compartido de Shamir.
- Habilidad para manejar archivos en el sistema operativo.

Con estos requisitos, el software puede ejecutarse de manera eficiente para cumplir con sus objetivos de cifrado y gestión de secretos.

¿En que está enfocado?

Enfoque del Software

Este software está diseñado para abordar la necesidad de proteger información confidencial en sistemas donde la recuperación de datos o la gestión de claves requiere la colaboración de múltiples partes. Su enfoque principal es combinar dos potentes técnicas criptográficas: el cifrado simétrico con AES-256 y el esquema de secreto compartido de Shamir.

El enfoque del software se basa en los siguientes principios:

- **Seguridad:** Garantizar la confidencialidad e integridad de los datos mediante técnicas de cifrado avanzadas.
- **Flexibilidad:** Permitir al usuario definir el número total de fragmentos (`n`) y el mínimo necesario (`t`) para reconstruir la clave.
- **Distribución del Riesgo:** Los fragmentos de la clave pueden almacenarse en diferentes ubicaciones o ser manejados por diferentes personas, eliminando el riesgo de un único punto de fallo.
- **Accesibilidad:** Ser una herramienta multiplataforma y ejecutable desde la línea de comandos, adecuada para usuarios avanzados y administradores de sistemas.
- **Educación y Aprendizaje:** Proporcionar un ejemplo práctico y funcional de conceptos avanzados de criptografía.

Con este enfoque, el software no solo es útil para aplicaciones prácticas, sino que también sirve como herramienta educativa y de investigación para quienes desean profundizar en la criptografía y la seguridad de la información.

Pseudocódigo:

A continuación, se presenta un pseudocódigo general que describe las principales funciones y flujo de trabajo del programa. Este pseudocódigo incluye las operaciones de cifrado, descifrado y manejo de fragmentos del esquema de secreto compartido de Shamir.

Flujo Principal

plaintext

Copiar código

```
INICIO del programa
    Leer argumentos desde la línea de comandos (args)
    Si el número de argumentos es insuficiente:
        Mostrar error y terminar programa

    Leer bandera (args[0]):
        Si bandera es "-c":
            Llamar a función Cifrar(args)
        Si bandera es "-d":
            Llamar a función Descifrar(args)
        Caso contrario:
            Mostrar error de bandera desconocida
FIN del programa
```

Función: Cifrar

plaintext

Copiar código

```
Función Cifrar(args):
    Extraer parámetros:
        archivoConContraseñas = args[1]
        numeroTotalEvaluaciones = args[2]
        minimoEvaluaciones = args[3]
        archivoDocumentoClaro = args[4]

    Pedir contraseña al usuario
    Validar la contraseña
    Derivar clave a partir de la contraseña (SHA-256)

    Dividir la clave utilizando el esquema de Shamir:
        Generar polinomio aleatorio de grado t-1
        Evaluar polinomio en n puntos únicos
        Guardar los puntos en archivoConContraseñas.frg

    Cifrar archivoDocumentoClaro usando AES-256:
        Generar clave AES a partir de la contraseña
        Leer contenido del archivo claro
        Cifrar contenido
        Guardar el archivo cifrado como archivoDocumentoClaro.aes

    Mostrar éxito del cifrado
FIN de la función
```

Función: Descifrar

plaintext

Copiar código

```
Función Descifrar(args):
    Extraer parámetros:
        archivoConContraseñas = args[1]
        archivoCifrado = args[2]

    Leer puntos (x, y) desde archivoConContraseñas
      Verificar que el número de puntos es suficiente para mínimo
requerido (t):
        Si no es suficiente:
            Mostrar error y terminar

    Recuperar la clave mediante interpolación de Lagrange:
        Inicializar claveRecuperada = 0
        Para cada punto en los fragmentos:
            Calcular término de Lagrange para el punto actual
            Sumar a claveRecuperada
        Modificar claveRecuperada usando el módulo definido (modulo)

    Descifrar archivoCifrado usando AES-256:
        Leer contenido del archivo cifrado
        Descifrar usando claveRecuperada
        Guardar archivo descifrado con su nombre original

    Mostrar éxito del descifrado
FIN de la función
```

Función: Generar Fragmentos con Shamir

plaintext
Copiar código

```
Función                    GenerarFragmentos(numeroTotalEvaluaciones,
minimoEvaluaciones, claveSecreta):
    Representar claveSecreta como BigInteger
    Inicializar lista de coeficientes:
        coeficientes[0] = claveSecreta
        Para i desde 1 hasta (minimoEvaluaciones - 1):
            coeficientes[i] = número aleatorio módulo `modulo`

    Inicializar lista de puntos
    Para cada x en 1..numeroTotalEvaluaciones:
        y = EvaluarPolinomio(coeficientes, x)
        Agregar punto (x, y) a la lista de puntos

    Retornar lista de puntos
FIN de la función
```

Función: Recuperar Secreto

plaintext
Copiar código

```
Función RecuperarSecreto(fragmentos):
    Inicializar secreto = 0
    Para cada fragmento i en fragmentos:
        yi = valor y del fragmento
        Calcular término de Lagrange Li(0) para fragmento i
        secreto += yi * Li(0)

    Aplicar módulo sobre secreto
    Retornar secreto
FIN de la función
```

Función: Evaluar Polinomio

plaintext

Copiar código

```
Función EvaluarPolinomio(coeficientes, x):
    Inicializar resultado = 0
    Para cada coeficiente en coeficientes:
        resultado = resultado * x + coeficiente
    Retornar resultado % modulo
FIN de la función
```

Resumen del Flujo

1. El programa comienza leyendo los argumentos de la línea de comandos.
2. Según la bandera proporcionada:
    - **Cifrar (-c):**
        1. Pide una contraseña y genera una clave derivada.
        2. Divide la clave en fragmentos usando Shamir.
        3. Cifra el archivo claro con AES-256 y guarda el resultado.
    - **Descifrar (-d):**
        1. Lee los fragmentos necesarios desde un archivo.
        2. Recupera la clave utilizando interpolación de Lagrange.
        3. Usa la clave recuperada para descifrar el archivo.
3. En ambos casos, muestra mensajes de éxito o error.

Resumen del Proyecto

Este documento describe un software desarrollado para la protección de información confidencial mediante la combinación de **cifrado simétrico AES-256** y el **Esquema de Secreto Compartido de Shamir**. Su propósito principal es garantizar la seguridad y gestión controlada de claves criptográficas, permitiendo dividir y distribuir dichas claves de manera segura entre múltiples partes.

Características Principales

1. **Cifrado Simétrico con AES-256:**
   - Utiliza el estándar **Advanced Encryption Standard (AES)** con una clave de 256 bits para proteger archivos de manera confiable.
   - Incluye metadatos en los archivos cifrados para facilitar su recuperación y descifrado.
2. **Esquema de Secreto Compartido de Shamir:**
   - Divide la clave de cifrado en `n` fragmentos mediante un polinomio aleatorio de grado `t-1`.
   - Solo es posible recuperar la clave original si al menos `t` fragmentos están disponibles.
3. **Recuperación del Secreto:**
   - Implementa interpolación de Lagrange para reconstruir la clave a partir de los fragmentos disponibles.
4. **Interfaz de Línea de Comandos:**
   - Ejecutable desde la terminal, con comandos claros para realizar operaciones de cifrado y descifrado.
   - Multiplataforma, compatible con sistemas Windows, macOS y Linux.
5. **Validación Robusta:**
   - Asegura la integridad de los parámetros y archivos proporcionados antes de realizar cualquier operación.
   - Verifica que los fragmentos sean suficientes para reconstruir la clave antes de intentar descifrar.

Objetivo del Proyecto

El objetivo del software es proporcionar una herramienta robusta y flexible para:

- **Proteger información sensible:** Cifrando archivos con seguridad de nivel avanzado.
- **Gestión distribuida de claves:** Dividiendo claves entre múltiples partes para reducir el riesgo de un único punto de fallo.

- **Control colaborativo:** Permitir que grupos confiables colaboren para acceder a información cifrada solo si cumplen con los requisitos definidos (mínimo de fragmentos).

Público Objetivo

El software está diseñado para ser utilizado por:

- **Profesionales de TI y Criptografía:**
  - Administradores, desarrolladores y analistas de seguridad que gestionen datos confidenciales.
- **Sistemas de Recuperación Distribuida:**
  - Organizaciones que necesiten gestionar claves criptográficas de manera distribuida y segura.
- **Proyectos Académicos y Educativos:**
  - Estudiantes e investigadores interesados en explorar implementaciones prácticas de conceptos avanzados de criptografía.

Casos de Uso

1. **Protección de Documentos Confidenciales:**
   - Cifrar documentos críticos y dividir la clave entre varios ejecutivos, asegurando que nadie pueda descifrar el archivo sin colaboración.
2. **Gestión de Claves Críticas:**
   - En sistemas distribuidos, las claves maestras se almacenan en fragmentos y solo se reconstruyen si las partes confiables proporcionan los fragmentos requeridos.
3. **Demostración Educativa:**
   - Mostrar cómo se pueden implementar técnicas avanzadas de criptografía en proyectos prácticos.

Características Técnicas

- **Lenguaje:** Java
- **Algoritmos Utilizados:**
  - AES-256 para cifrado simétrico.
  - Esquema de Secreto Compartido de Shamir para dividir y recuperar claves.
- **Entrada y Salida:**
  - Archivos de texto y binarios soportados para cifrado.
  - Fragmentos almacenados en formato legible como pares `(x, y)`.
- **Compatibilidad:** Multiplataforma (Windows, macOS, Linux).

Ventajas del Proyecto

1. **Seguridad Robusta:**
    - Combina el cifrado AES-256, ampliamente reconocido por su resistencia, con la flexibilidad del esquema de Shamir.
2. **Flexibilidad:**
    - Permite configurar el número total de fragmentos (`n`) y el mínimo necesario (`t`) para recuperar la clave, adaptándose a diversos escenarios.
3. **Distribución del Riesgo:**
    - Los fragmentos pueden almacenarse en diferentes ubicaciones o asignarse a diferentes usuarios, minimizando el riesgo de acceso no autorizado.
4. **Facilidad de Uso:**
    - La interfaz de línea de comandos permite su integración en flujos de trabajo automatizados y su uso por usuarios avanzados.

Procedimientos:

Mantenimiento

Mantenimiento del Software

El mantenimiento de este software es fundamental para garantizar su funcionamiento óptimo y su adaptación a nuevas necesidades o entornos. Se incluyen los siguientes tipos de mantenimiento:

- **Correctivo:** Resolución de errores detectados en las funcionalidades existentes.
- **Preventivo:** Optimización del código y mejora de la documentación para prevenir problemas futuros.
- **Evolutivo:** Incorporación de nuevas características, como soporte para otros formatos de archivo o una interfaz gráfica.
- **Adaptativo:** Ajustes para asegurar compatibilidad con nuevas tecnologías o plataformas.

Recomendaciones para el 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.

Plan de Mantenimiento:

1. Revisiones trimestrales del código y dependencias.
2. Sistema de reporte de errores a través de GitHub Issues u otro medio.
3. Publicación de nuevas versiones con changelogs detallados.

El mantenimiento garantizará que este software siga siendo una herramienta segura, eficiente y confiable para la protección de información sensible y la gestión de secretos.

Futuras Actualizaciones

Este software está diseñado con un enfoque modular, lo que permite agregar nuevas funcionalidades y mejorar las existentes. A continuación, se presentan posibles futuras actualizaciones que enriquecerán la herramienta:

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.
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.

Costo del proyecto:

El desarrollo de este proyecto involucró la participación de un equipo de dos programadores, quienes trabajaron durante un período de **6 días**, dedicando un promedio de **3 horas diarias** al diseño, desarrollo e implementación del software. En total, se invirtieron **36 horas de trabajo**, considerando tareas como:

- Investigación de algoritmos criptográficos.
- Implementación y pruebas de las funcionalidades principales.
- Documentación y ajustes finales.

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.

Este costo es razonable y competitivo en el mercado, dado el nivel de especialización y utilidad del producto final.

Referencias

- **Cámara de Diputados del H. Congreso de la Unión.** (2010). *Ley Federal de Protección de Datos Personales en Posesión de los Particulares*. Diario Oficial de la Federación. Recuperado de https://www.diputados.gob.mx/LeyesBiblio/pdf/LFPDPPP.pdf

- Caldwell, Chris K. y G.L. Honaker, "Prime Curios! The dictionary of prime number trivia", CreateSpace, 2009. Prime Curios Database: http://primes.utm.edu/curios/page.php?number_id=3746

- **Shamir, A.** (1979). How to share a secret. *Communications of the ACM, 22*(11), 612–613.
- Aldeco, R., Solano, J. A., Sáenz, R. y Mejía, A. (2021). Algoritmo AES (advanced encryption standard). Unidades de Apoyo para el Aprendizaje. CUAIEED/Facultad de Ingeniería-UNAM. https://repositorio-uapa.cuaieed.unam.mx/repositorio/moodle/pluginfile.php/2184/mod_resource/content/2/contenido-uapa/index.html
- Unión Europea. (2016). Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE . Diario Oficial de la Unión Europea, L 119, 1-88.