Universidad Nacional Autónoma de México

Facultad de Ciencias

Modelado y programación

Góngora Ramírez Dania Paula 318128274
Daniel

README.

Nota: Mi compañero dejo de contestarme mensajes y ya era tarde asi que no pude saber su nombre completo pero si participo en el proyecto.

1. Ejecución

Se necesita instalar java(yo tengo hasta java 13 asi que supongo que ese) y se necesita instalar ant, para instalar ant en ubuntu es necesario abrir la terminal y poner los siguientes comandos:
sudo apt update y después
sudo apt install ant Ejecución del proyecto Para correr el proyecto en necesario poner:
ant jar y depués
java -jar build/jar/proyecto.jar (el argumento que se necesiten 'c' o 'd')
Si se quiere cifrar(argumento 'c')

java -jar build/jar/proyecto.jar c nombreDelArchivoAGuardar numeroDeEvaluaciones numeroMinimodeEv NombreDelArchivoOriginal
Si se quiere descifrar(argumento 'd') java -jar build/jar/proyecto.jar d nombreDelArchivoconEval nombreArchivoCifrado
Para correr la pruebas es necesario poner:
ant tst
y después

ant test
Para generar la documentación es con:
ant doc

2. Definición del problema

El esquema de secreto compartido de Shamir hace posible que un sólo dato pueda ser ocultado de manera que, a partir de él, se generan n diferentes datos y que con, al menos $t \leq n$ cualesquiera de ellos sea posible recuperar el dato original.

El propósito del proyecto final, es elaborar un programa que implemente el esquema de Shamir para compartir la clave necesaria para descifrar un archivo que suponemos contiene información confidencial. Dicha información debe poder ser accedida siempre que estén presentes, al menos u, de los n miembros de un grupo de personas con autorización para acceder a ella. A partir de un documento u otro tipo de archivo que suponemos contiene información confidencial, se debe generar una versión cifrada de él. Tanto el documento cifrado, como las n evaluaciones del polinomio, pueden ser distribuidos entre las personas con acceso autorizado al documento claro. Si luego se logran reunir al menos $t \leq n$

de estas personas, usando el programa, pueden descifrar el documento cifrado recuperando el claro.

3. Analisis del problema

Datos de entrada:
El programa debe funcionar en dos modalidades, para cifrar (opción c) y para descifrar (opción d).
Cifrar. Se debe proporcionar, en la línea de llamada: 1. La opción c.
2. El nombre del archivo en el que serán guardadas las n evaluaciones del polinomio.
3. El número total de evaluaciones requeridas (n >2).
4. El número mínimo de puntos necesarios para descifrar ($1 < t \leq n$).
5. El nombre del archivo con el documento claro.
6. Ya con el programa en ejecución se debe solicitar al usuario una contraseña (sin hacer eco en la terminal).

Descifrar. Se debe proporcionar, en la línea de llamada:
1. La opción d.
2. El nombre del archivo con, al menos, t de las n evaluaciones del polinomio.
3. El nombre del archivo cifrado.

Datos de salida
Dependiendo de la opción seleccionada el programa debe entregar lo que se especifica.
Cifrar.
El archivo con el documento cifrado usando AES. El archivo con n parejas $(x_i, P(x_i))$ de las evaluaciones del polinomio.
Descifrar.
El archivo con documento claro y con el nombre original.

4. Selección de la mejor alternativa

Para el cifrado fue importante buscar una forma leer y escribir cualquier tipo de documento, para esto se utilizo BufferedReader y almacenar la información del documento con una lista, de esta forma el manejo se simplifico. Además de utilizar lo especificado en el pdf del proyecto que es SHA-256 y enteros grandes para manejar la contraseña, para esto lo que consideramos adecuado es BigInteger de java el cual nos permite manejar número enteros grandes, justo como lo requiere nuestro problema, para SHA-256 MessageDigest ya que nos permite usarlo adecuadamente, pero hay que tener en cuenta que este no es thread-safe.

Para encriptar el mensaje consideramos que Cipher era la mejor opción ya que nos parecio una biblioteca segura que java ya ofrecia, además de permitirnos

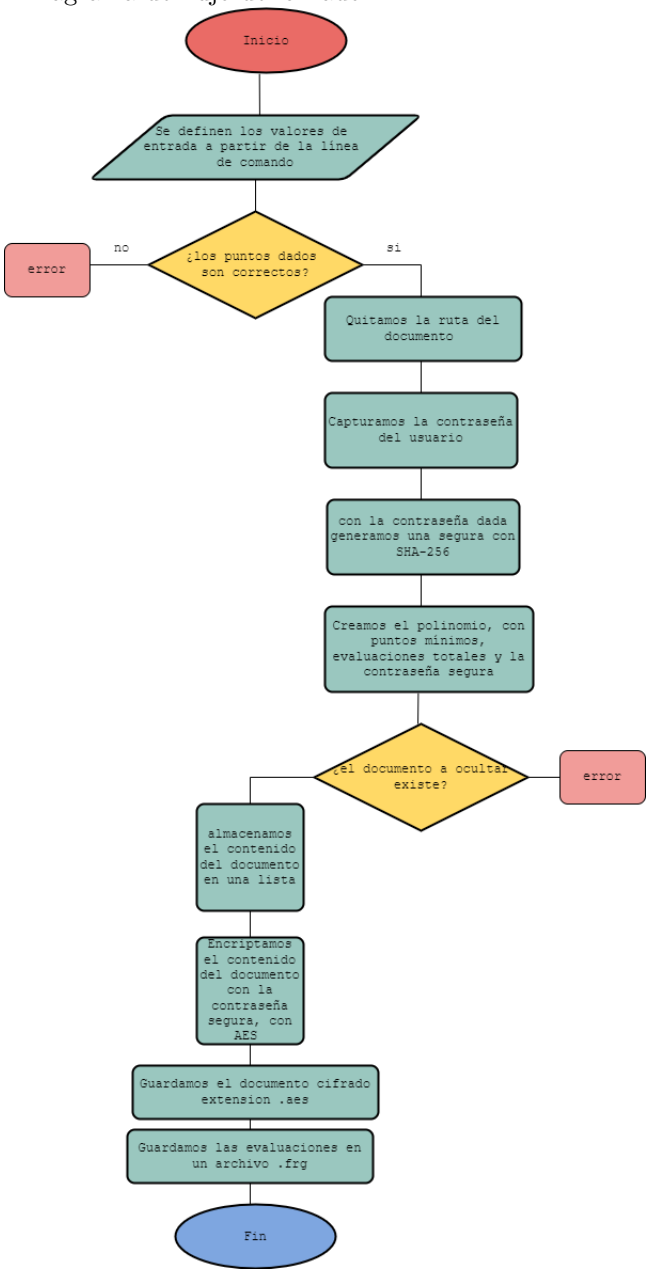
incluir encriptar y decriptar de manera segura con las especificación requerida la cual es AES, lo mismo ocurrio con SecretKeySpec.

5. Pensamiento a futuro

Hya varias consideraciones que se podrian tener con este proyecto a futuro, una de ellas que consideramos importante destacar es debido a MessageDigest que se utiliza al generar una contraseña segura no es thread-safe, asi que podria buscarse añlternativas para mejorar este aspecto del programa, otro aspecto a considerar es que podria tener interfaz gráfica la cual ayude a que cualquier tipo de usuario pueda utilizarla.

6. Diagrama de flujo

Diagrama de flujo del cifrado:



7. Recursos

El método crearContrasenia de la clase contrasenia se baso en: https://howtodoinjava.com/java/java-security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bc

El método leerDocumentos de la clase Documentos se baso en: https://www.logicbig.com/how-to/code-snippets/jcode-java-io-files-write.html

El método guardarDocumentos de la clase Documentos se baso en: https://www.logicbig.com/how-to/code-snippets/jcode-java-io-files-write.html

La clase PolinomioAleatorio baso su estructura y método constructor en: https://github.com/terappy/ShamirSecretSharing/blob/master/src/main/java/scheme/SecretShare.java

El método generaEnteros de PolinomioAleatorio se basa en: https://github.com/RicoVergara94/ShamirsSecretSharingAlgo/blob/main/src/main/java/Polynomial.java

El método cifrar y todos sus métodos auxiliares utilizados en el se basaron en: https://github.com/Armando122/Proyectos-Modelado-y-Programacion/blob/master/esquema_shamir/src/main/java/com/MyP/proyecto/EsquemaShamir.java

El método encriptar de Cifrar se baso en: https://stackoverflow.com/questions/140131/convert-a-string-representation-of-a-hex-dump-to-a-byte-array-using-java
https://www.geeksforgeeks.org/java-program-to-convert-hex-string-to-byte-array/
https://stackoverflow.com/questions/20770072/aes-cbc-pkcs5padding-vs-aes-cbc-pkcs7padding-w