Universidad Nacional Autónoma de México

Facultad de Ciencias

Proyecto 03

17 de Diciembre del 2021

Autores: Castro Romero Ares Gael y Prado Escamilla Axel

Documentación de la Solución.

---

1. **Definición del problema.**

   El problema planteado consiste en entregar un archivo con evaluaciones , con las cuales se pueda reconstruir un polinomio, ya que al evaluar en 0 dicho polinomio se encuentre la llave que permite la decodificación del texto que necesitamos que permanezca oculto hasta que se tenga cierto mínimo de particiones.

---

2. **Análisis del problema.** La entrada del programa solución permite tener dos opciones, codificación y decodificación, nombre del archivo en el cual se escribirán evaluaciones en puntos aleatorios de la función, número total de particiones, número mínimo de particiones y finalmente el nombre del archivo.

   Para poder decodificar se necesita forzosamente codificar primeramente. Al elegir la codificación, se le pedirá una contraseña y se le notificará que el archivo donde se encuentran sus evaluaciones ya se ha creado.Cuando se decida decodificar simplemente tiene que elegir la opción y con eso se le entrega el texto ya decodificado.

   En ambas salidas se le notifica cuales son los nombres de los archivos de evaluaciones y del texto.

---

3. **Modelo y abstracción.**

   El paradigma de programación que será usado es el orientado a objetos y el lenguaje usado será Java.

   El algoritmo central para cifrar el archivo cuya entrada es la llave creada a partir de la contraseña introducida.

```
byte[] k = llave.toByteArray();
  String archivo = this.archivoAES();

  try {
    Cipher cipher = Cipher.getInstance("AES");
    FileOutputStream textoCifrado = new FileOutputStream(archivo,true);
    SecretKeySpec kSec = new SecretKeySpec(k,"AES");
    cipher.init(Cipher.ENCRYPT_MODE,kSec);
    CipherInputStream cipherInput = new CipherInputStream(new FileInputStream(this.nombreArchivoTexto)
  ,cipher);
    int escribir = cipherInput.read();
    while(escribir != -1) {
      textoCifrado.write(escribir);
      escribir = cipherInput.read();
    }
    textoCifrado.close();
    cipherInput.close();
    System.out.println("El texto cifrado es guardo como: " + archivo);
  } catch(Exception e) {
    System.err.println(e);
      System.exit(1);
  }
```

   El pseudocódigo hace lo siguiente:

a) Se instancia de la clase cypher para poder cifrar.

b) Se utiliza la llave que se pasa como parámetro para poder inicializar el cifrado

c) Se lee el texto del archivo

d) Dentro de un bucle while se cifra cada palabra.

e) Se imprime en la consola el nombre de como se guardo el texto cifrado

.....

El método para descifrar el texto cifrado recibe como parámetro la misma llave que es la contraseña transformada.

```
    byte[] k = llave.toByteArray();
    String archivo = this.archivoTexto();
    try {
      Cipher cipher = Cipher.getInstance("AES");
      FileOutputStream textoDecifrado = new FileOutputStream(archivo,true);
      SecretKeySpec kSec = new SecretKeySpec(k,"AES");
      cipher.init(Cipher.DECRYPT_MODE,kSec);
      CipherInputStream cipherInput = new CipherInputStream(new FileInputStream(this.nombreArchivoTexto)
    ,cipher);
      int escribir = cipherInput.read();
      while(escribir != -1) {
        textoDecifrado.write(escribir);
        escribir = cipherInput.read();
      }
      textoDecifrado.close();
      cipherInput.close();
      System.out.println("El texto decifrado es guardo como: " + archivo);
    } catch(Exception e) {
      System.err.println(e);
        System.exit(1);
    }
```

El pseudocódigo hace lo siguiente:

a) Se instancia de la clase cypher para poder descifrar.

b) Se utiliza la llave que se pasa como parámetro para poder inicializar el descifrado

c) Se lee el archivo cifrado

d) Dentro de un bucle while se descifra cada byte .

e) Se imprime en la consola el nombre de como se guardo el texto descifrado

.....

Para la implementación de los pseudocódigos, modelamos el problema con clases. Se implementaron las siguientes clases:

a) **Polinomio Shamir**

Esta clase la incorporamos en el paquete *src*.

Esta clase nos permite crear el polinomio de donde se obtendrán las evaluaciones .

Contiene tres atributos, los cuales son el grado del polinomio, los coeficientes de éste y un numero primo con el cual se hará la función mod para crear la llave.

Posee los siguientes métodos:

- *evaluarValor:* Recibe como parámetro un valor en el cual será evaluado en el polinomio. Regresa el valor que regresa al ser evaluado el parámetro.
- *archivoEvaluaciones:* Método que genera el archivo con las evaluaciones.
- *parejaEvaluacion:* Método que obtiene la cadena de parejas (x,P(X)) en forma de cadena
- *evaluarNValores:* Método que obtiene una lista en forma de cadena de n evaluaciones del polinómio.

b) **CifradoShamir.**

Esta clase la incorporamos en el paquete *src*.

Esta clase sirve para cifrar y descifrar el texto y generar los archivos AES y txt.

Esta clase posee los siguientes métodos:

- *archivoAES:* Método que regresa una cadena con el nombre del archivo AES
- *archivoTexto:* Método que regresa una cadena con el nombre del archivo txt.
- *cifradoAES:* Método que instancia a la cypher y cifra el archivo de texto
- *decifradoAES:* Método que instancia a la cypher y descifra el archivo AES

c) **Lagrange.**

Esta clase la incorporamos en el paquete *src*.

Esta clase nos permite construir un polinomio mediante el método de interpolación de lagrange.

Los método que posee son:

- *lagrangeInterpolacion:* Método que realiza la construcción del polinomio dados t puntos usando el algoritmo de Lagrange.
- *numeradorLagrange:* Método que realiza la construcción del numerador del termino L_i en el algoritmo de Lagrange.
- *denominadorLagrange:* Método que realiza la construcción del denominador del termino L_i en el algoritmo de Lagrange.

d) **LlaveCifrado.**

Esta clase la incorporamos en el paquete *src*.

Esta clase nos permite manejar el password dado por el usuario y la llave de cifrado para el cifrado AES.

Los método que posee son:

- *obtenerContraseña:* Método que pide el password al usuario.
- *hashContraseña:* Método que aplica el hashing SHA-256 en una contraseña.
- *vectorEvaluaciones:* Método que recupera las evaluaciones de una archivo con parejas (x,P(X)).

e) **Menu.**

Esta clase contiene el método ***main*** que permite ejecutar el programa entero.

---

4. **Cambios Futuros**

Interfaz Gráfica

Una interfaz gráfica haría que todo sea más visible y tangible para el usuario, ya que podría ingresar en una ventana las opción que desee ya sea el de codificar y se le muestre el texto oculto mediante SHA-256 y las evaluaciones del polinomio y si desea decodificar el texto decodificado en un archivo elegido por él.

---

5. **Costos**

Se decide cobrar por proyecto dado que por línea de código o incluso por hora realmente llegan a ser cantidades absurdas para ambas partes.

El costo del proyecto se basa principalmente con respecto a la fecha de entrega de éste. Si hay poco tiempo para desarrollarlo, el monto llega a duplicarse con respecto al precio que normalmente se cobrearía.

Ahora bien **¿Cuál es ese precio?** Cobramos un precio de aproxidamente de $3000 pesos mexicanos si es que no se requiere comprar una clave para solicitar a la API. Así, $\frac{1}{3}$ del precio son destinados a los recursos por parte de los integrantes del proyecto, quedando una ganacia de $\frac{1}{3}$ para cada quien. Así las ganancias son justas y se cubren gastos como comida, internet, luz, etc.
