UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ciencias

Proyecto de Modelado y Programación

Asignatura: Modelado y Programación

Alumnos:

- **Urzua Contreras Cristian Josué**
- **Reyes Arteaga Ángel David**

Profesores:

- **José de Jesús Galaviz Casas**
- **María Ximena Lezama Hernández**
- **Karla Adriana Esquivel Guzmán**

Proyecto de Modelado y Programación

1 Descripción del Programa

Este programa permite ocultar un mensaje o archivo, usando la técnica de Shammir, consiste en un polinomio generado de n coeficientes y n+1 evaluacion con números distintos. De igual forma permite desencritarlo usando la misma técnica y guardarlo en un .txt

1.1 Requisitos

- Python 3.x
- Biblioteca Random para generar evaluaciones y coeficientes aleatorios
- Biblioteca Base64 para operar con la contraseña Haslib en el polinomio
- Biblioteca Hashlib para generar la clave única
- Biblioteca Io para manejar archivos y rutas de los mismo
- Biblioteca Argparse para poder ejecutar funciones desde la terminal
- Biblioteca Cryptography para codificar el texto.

Instala Criptography ejecutando

```
pip install cryptography
```

- Biblioteca Sympy para operar usando interpolación de Lagrange

Instala Sympy ejecutando

```
pip install sympy
```

2 Estructura del Código

1. **codificar(archivo, contraseña)**: Codifica un archivo en formato AES con la contraseña solicitada
2. **decodificar(archivo_codificado, fragmentos)**: Desencripta el mensaje del archivo.aes con la n evaluaciones
3. **Ejecución desde línea de comandos**: La función __main__ interpreta los argumentos proporcionados.

3 Funcionalidades

3.1 Función genera_polinomio(grado, numero)

Genera una lista del tamaño del número de coeficientes equivalentes al grado y el numero se transformará en

el termino independiente del mismo. El polinomio al menos tendrá dos coeficientes y será almenos de máximo 2 a 5 coeficientes (se puede generar otro polinomio mayor pero nopude ejecutarlo en mi computadora, aunque es funcional)

Ejemplo de uso:

```
genera_polinomio(5, 57)
```

3.2 Función evaluar (polinomio, numero)

Evalua el numero en el polinomio interpretando los indices

Ejemplo de uso:

```
evaluar ([1,5,7], 3)
```

3.3 Función codificar (archivo, contraseña):

Dado un archivo, será codificado en un archivo.aes y la contraseña será ocultada en n evaluaciones de un polinomio aleatorio:

Funciones auxiliares

a) **Funcion convierte256(contraseña)**: La contraseña será transformada en una contraseña única hash en formato 64 bits

b) **Función to_dec(n64):** Transforma un número en base 64 a decimal

c) **Función genera_frags(polinomio):** Generará un archivo.frags que contenga las n evaluaciones del polinomio aleatoriamente generado

Parámetros:

Archivo: nombre o ruta del archivo que se deseé codificar
Contraseña: Clave que usará para poder decodificar el archivo encriptado

Ejemplo de uso:

```
codificar("claves_nucleares.txt", "Contraseña_secreta123")
```

3.4 Función descodificar(archivo_aes, archivo_frags)

Decodificará el archivo encriptado con las n evaluaciones contenidas en el archivo.frags.

- Usando la función auxiliar interpolación(archivo_frags) Leerá cada evaluación de las n filas correspondientes y usando la interpolación de lagrange, se obtendrá el polinomio original y posteriormente se obtendrá el número correspondiente a la contraseña.
- El numero obtenido será transformado en base 64 para conseguir la contraseña que decodifique elarchivo.aes

Parámetro:

- archivo_aes: Ruta o nombre del archivo encriptado
- archivo_frags: Ruta o nombre del archivo que contendrá las n evaluaciones del polinomio

Ejemplo de uso:

descodificar ("claves_nucleares.aes", "fragmentos_frags")

4 Ejecución desde Línea de Comandos

El programa acepta dos modos de operación:

- 1)**Codificar (-c):** Encripta el archivo con formato AES
- **2)Decodificar (-d):** Desencripta el archivo.

.

4.1 Opciones de Ejecución

4.1.1 Opción -c (Codificar)

python nombre_del_proyecto.py -c --archivo 'ruta_del_archivo' --contraseña "nombre_de_la_constraseña"

- ruta_del_archivo: Nombre del archivo que contiene el texto a ocultar.
- nombre_de_la_constraseña: Contraseña que se deseé poner.
-

Ejemplo:

python bow.py -c --'mis claves nucleares.txt' --"moratdela"

4.1.2 Opción -d (Decodificar)

python python nombre_del_proyecto -d --archivo_codificado 'archivo_codificado.aes' --fragmentos 'n_evaluaciones.frags'

- archivo_codificado: Archivo que este codificado en .aes.
- n_evaluaciones.frags: Archivo que conténgalas n evaluaciones aleatorias
- Ejemplo:

python bow.py -d --archivo_codificado 'mis claves nucleares.aes' --fragmentos 'fragmentos.frags

5 Ejemplos de Uso

- **Codificar**

  python bow.py -c --'mis claves nucleares.txt' --"moratdela"

  Esto codificará el archivo y será generado en un archivo del mismo nombre .aes, con ello generará un archivo ..frags que contengan las n evaluaciones de un polinomio aleatorio

- **Decodificar**

python bow.py -d --archivo_codificado 'mis claves nucleares.aes' --fragmentos 'fragmentos.frag

  Esto generará la contraseña original y será usada para genera un archivo con el nombre original que tendrá la información decodificada del texto.

6 Precauciones y Advertencias

- Se recomienda usar la codificación con un archivo.txt puesto que es incierto lo que puede pasar usando otro formato

- Cuando se usa una ruta ajena se generará el archivo.aes y .frags en la carpeta donde se encuentre el texto que se deseé encriptar, estos son losque la funcionalidad decodificar usará