Proyecto final. Esquema de Secreto Compartido de Shamir

Pérez Romero Natalia Abigail

enero 2021

1. Introducción

En este documento explicaré el proceso de resolución del proyecto.

2. Definición de problema

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.

3. Análisis del programa

Utilicemos el proceso de solución de problemas:

3.1. Entrada

El programa debe funcionar en dos modalidades, para cifrar (opción c) y para descifrar (opción d)

Cifrar Se debe proporcionar:

1. El nombre del archivo en el que serán guardadas las n evaluaciones del polinomio.
2. El número total de evaluaciones requeridas $n > 2$
3. El número mínimo de puntos necesarios para descifrar $(1 < t \leq n)$
4. El nombre del archivo con el documento claro.
5. Contraseña (sin hacer eco en la terminal)

Descifrar Se debe proporcionar:

1. El nombre del archivo con, al menos, t de las n evaluaciones del polinomio.

2. El nombre del archivo cifrado.

3.2. 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.

En la siguiente sección explicaré la elección de las herramientas para solucionar el problema.

4. Selección de la mejor alternativa

4.1. Elegir arsenal

El paradigma de programación que utilizaré será modular y el lenguaje de programación Python.

Escogí el paradigma de programación modular porque es posible dividir el proyecto en problemas más simples:

1. Manejar la entrada del programa, es decir, recuperar un archivo de texto o una imagen de un fichero.

2. Manejar la salida del programa, en este caso crear un archivo de texto o una imagen.

3. Cifrar y descifrar el mensaje.

De forma que el modulo `reader_writer.py` se encarga de 1 y 2 por lo que le envía el texto recuperado del archivo al modulo principal `__init__.py` que resuelve de 3. A su vez el cifrar y descifrar el mensaje se puede dividir en otros problemas:

1. Generar un polinomio para después

2. calcular las n evaluaciones del polinomio $f(x)$

3. con las que se obtiene la variable independiente del polinomio $f(x)$ cuando $x = 0$ para

4. obtener la llave para descifrar

Python cuenta con librerías y entornos de trabajo que resultan útiles en este tipo de proyectos, como `secrets` y algunas son herramientas que he utilizado anteriormente, como `unittest`.

4.2. Escatimar trabajo inútil

Use distintos módulos que ofrece Python:

1. `hashlib`. Este módulo implementa una interfaz común para muchos hash seguros diferentes como SHA256 que utilice en este proyecto, para generar la variable independiente del polinomio.

2. `secrets`. Se usa para generar números aleatorios criptográficamente fuertes adecuados para administrar datos como contraseñas, autenticación de cuentas, tokens de seguridad y secretos relacionados. Use este modulo en lugar de `random` para generar números aleatorios para los coeficientes del polinomio.

3. `getpass`. Proporciona dos funciones. Una de ellas `getpass` solicita al usuario una contraseña sin hacer eco.

4. `unittest`. Utilice este modulo para crear pruebas unitarias.

Un paquete vital para este proyecto es `Crypto.Cipher`, el cual contiene algoritmos de cifrado, como `AES (Advanced Encryption Standard)` el cual es un algoritmo de cifrado de bloque simétrico.

Debido a problemas que tuve para transformar el número que obtuve al recuperar la variable independiente probé utilizar el modulo `struct` el cual realiza conversiones entre valores de Python y estructuras C representadas como objetos de bytes de Python.

4.3. Elegir el modelo de datos

Utilice listas porque tienen la ventaja de poderse acceder como arreglos y permite crear una lista de listas que facilita el almacenamiento y las operaciones de los coeficientes del polinomio.

4.4. Pruebas

Pruebas se hicieron y porque:

1. test_reader. Este test verifica que la lectura de un archivo sea lo esperado.

2. test_reader_diff. Este test verifica que la lectura de un archivo sea lo esperado considerando caracteres de espacio.

3. test_read_write. Este test verifica que los métodos de escribir y leer estén correctos, comprobando que lo que se escribe un método es lo mismo que va leer el otro método.

4. test_generate_coefficients. Prueba que se generen los coeficientes necesarios

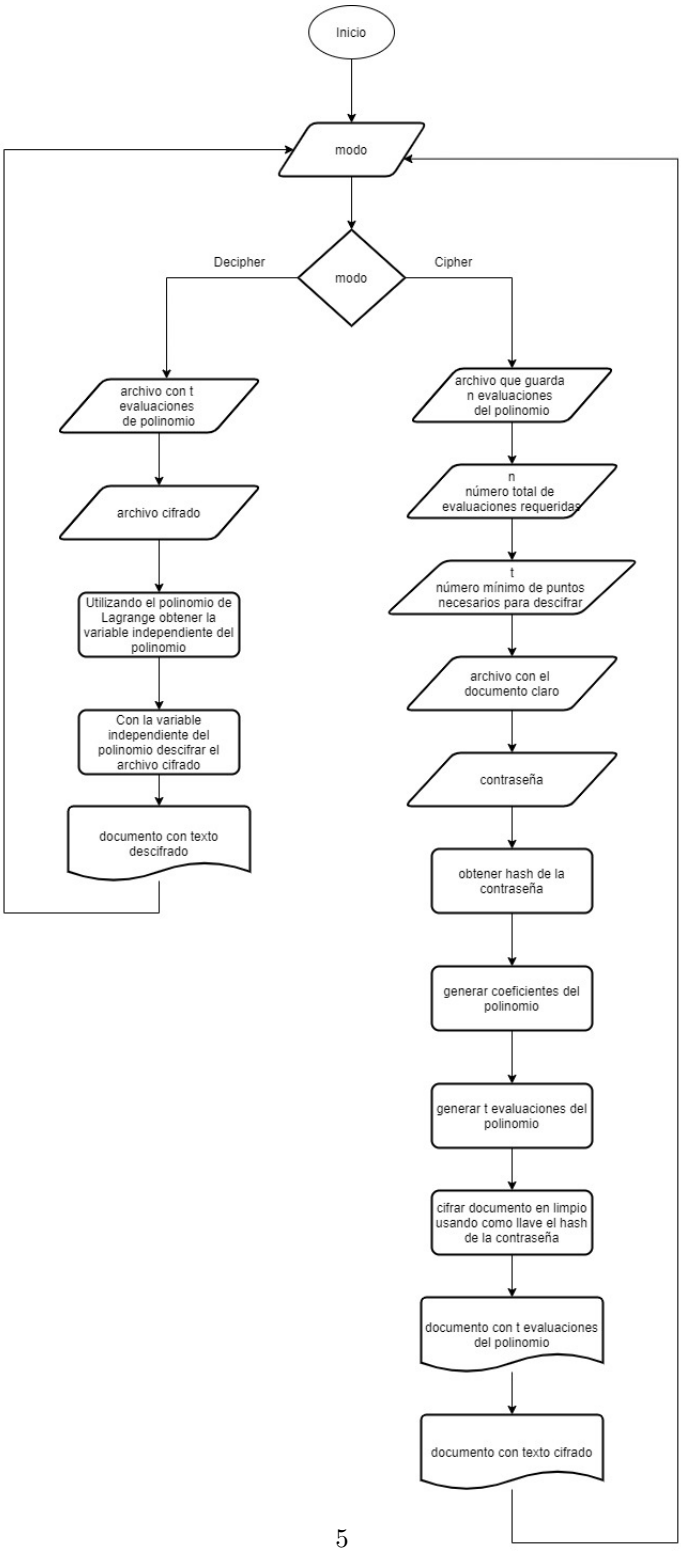
5. test_horner_algorithm. Prueba que el algorimo este correctamente implementado

6. test_lagrande_polynomial. Prueba que dado dos puntos cuando se evalué el polinomio $f(x) = Ax^2 + k$ en $f(0) = k$ en este caso 0

5. Diagrama de flujo o pseudocódigo

6. El futuro

El proyecto me parece complicado por lo que, una vez que pueda garantizar que el proyecto es correcto, cobraría $13,000$ por que a mi parecer requiere conocimientos intermedios de un lenguaje de programación además de que conservar la integridad de información confidencial es importante. Me gustaría realizar más pruebas para verificar que todos los métodos son correctos. Y tal vez una interfaz gráfica.


Inicio
modo
modo
Decipher
Cipher
archivo con t evaluaciones de polinomio
archivo cifrado
Utilizando el polinomio de Lagrange obtener la variable independiente del polinomio
Con la variable independiente del polinomio descifrar el archivo cifrado
documento con texto descifrado
archivo que guarda n evaluaciones del polinomio
n
número total de evaluaciones requeridas
t
número mínimo de puntos necesarios para descifrar
archivo con el documento claro
contraseña
obtener hash de la contraseña
generar coeficientes del polinomio
generar t evaluaciones del polinomio
cifrar documento en limpio usando como llave el hash de la contraseña
documento con t evaluaciones del polinomio
documento con texto cifrado