"Año de la unidad, la paz y el desarrollo"

Universidad Nacional de Ingeniería
Facultad de Ciencias

Cifrado Hill

Integrantes	Código	Correo UNI
Iman Noriega Melissa	20224041G	melissa.iman.n@uni.pe
Llanos Rosadio José Ismael	20220107C	jose.llanos.r@uni.pe
Olivera Calderón Renato Steven	20224111E	renato.olivera.c@uni.pe
Ortiz Lozano Eric Hernan	20220587E	eric.ortiz.l@uni.pe

Docente: Ronald Mas

Julio 2023

Índice

1. Resumen

Este informe presenta un proyecto de matemáticas discretas centrado en el cifrado Hill. El objetivo del proyecto es estudiar y analizar este algoritmo clásico de cifrado desde una perspectiva matemática. El informe incluye una breve historia del cifrado Hill, la implementación del algoritmo en Python, la discusión de sus desventajas y las conclusiones obtenidas.

En el marco teórico, se introduce la importancia de la criptografía clásica y se exploran los fundamentos matemáticos del cifrado Hill, incluyendo el uso de matrices y operaciones algebraicas. A continuación, se presenta el código en Python que permite cifrar y descifrar mensajes utilizando el cifrado Hill, acompañado de ejemplos prácticos que ilustran el proceso paso a paso.

En cuanto a los resultados, se destacan las limitaciones y vulnerabilidades del cifrado Hill, incluyendo su susceptibilidad a ataques criptoanalíticos. Se analizan los aspectos matemáticos y prácticos del algoritmo, resaltando su importancia histórica en la criptografía clásica y su contribución al desarrollo de algoritmos de cifrado más seguros.

Finalmente, las conclusiones del proyecto reflejan los resultados y análisis realizados. Se reconoce la relevancia de comprender los aspectos matemáticos y prácticos de los algoritmos criptográficos para una implementación segura. Se resalta la necesidad de considerar las desventajas y vulnerabilidades de los algoritmos clásicos como el cifrado Hill, y la importancia de seguir desarrollando métodos de cifrado más robustos.

En resumen, este proyecto ofrece una mirada detallada al cifrado Hill, desde su historia hasta su implementación en Python, y proporciona un análisis crítico de sus desventajas. El informe destaca la importancia de seguir investigando y desarrollando nuevos enfoques criptográficos para garantizar la seguridad de la información en el mundo digital actual.

2. Marco teórico

2.1. Criptografía

La criptografía es la ciencia de representar información de forma opaca para que solo los agentes autorizados (personas o dispositivos diversos) sean capaces de desvelar el mensaje oculto. El proceso de ocultar la información se llama cifrado, pero a menudo también se le llama encriptado por influencia del inglés. El proceso de desvelarla se llama descifrado o desencriptado.

Tipos de criptografía más importantes:

- **Criptografía simétrica:** En este tipo de criptografía, se utiliza una clave única tanto para el cifrado como para el descifrado de la información. Los algoritmos de criptografía simétrica, como AES (Advanced Encryption Standard) y DES (Data Encryption Standard), son ampliamente utilizados en aplicaciones que requieren una alta velocidad de procesamiento y confidencialidad. Además, existen otros algoritmos de criptografía simétrica, como el cifrado Hill y el cifrado Vigenère, que también se utilizan en diferentes contextos.

- **Criptografía asimétrica:** También conocida como criptografía de clave pública, utiliza un par de claves: una clave pública para el cifrado y una clave privada correspondiente para el descifrado. RSA (Rivest, Shamir y Adleman) y ECC (Elliptic Curve Cryptography) son ejemplos populares de algoritmos de criptografía asimétrica. Este tipo de criptografía se utiliza principalmente para el intercambio seguro de información y la autenticación.

- **Criptografía de hash:** En lugar de cifrar información, los algoritmos de criptografía de hash generan una representación única y fija de un conjunto de datos de entrada, conocido como hash. Los hashes son ampliamente utilizados para verificar la integridad de los datos y garantizar la autenticidad de los archivos. Ejemplos comunes de algoritmos de hash incluyen MD5, SHA-1 y SHA-256.

- **Criptografía de curva elíptica:** Este tipo de criptografía se basa en las propiedades matemáticas de las curvas elípticas y ofrece un alto nivel de seguridad con tamaños de clave más pequeños en comparación con otros algoritmos criptográficos. ECDSA (Elliptic Curve Digital Signature Algorithm) y ECDH (Elliptic Curve Diffie-Hellman) son ejemplos de algoritmos de criptografía de curva elíptica.

2.2. Cifrado Hill

Es un cifrado de sustitución poligráfica basado en el álgebra lineal. Inventado por Lester S. Hill en 1929, fue el primer cifrado poligráfico que era práctico para operar sobre más de tres símbolos inmediatamente. Este cifrado utiliza conceptos matemáticos y algebraicos para cifrar y descifrar mensajes. A diferencia de otros métodos de cifrado, como el cifrado César o el cifrado de sustitución, el cifrado Hill trabaja con grupos de letras en lugar de letras individuales. Esto significa que puede cifrar bloques de texto en lugar de caracteres individuales, lo que proporciona un nivel adicional de seguridad.

Cifrado

Se refiere al proceso que convierte la representación original de la información, conocida como texto plano, en una forma alternativa conocida como texto cifrado.

Descifrado

Es el proceso de convertir el texto cifrado en el texto en claro.

2.3. Pasos del Cifrado Hill

1. **Generación de la clave:**

   Determine la dimensión de la matriz clave que se utilizará para el cifrado Hill. Por ejemplo, si se utilizará una matriz 2x2, se necesitarán 4 elementos de clave.

   Elija los elementos de la clave, que deben ser números enteros y coprimos con el tamaño del alfabeto utilizado.

2. **Preparación del texto:**

   Divida el texto en bloques de tamaño igual al número de elementos de la matriz clave. Asegúrese de que la longitud del texto sea un múltiplo del tamaño del bloque. Agregue espacios en blanco al final si es necesario.

3. **Asignación de números a las letras:**

   Asigne a cada letra del bloque un número correspondiente según su posición en el alfabeto utilizado. Por ejemplo, si está utilizando el alfabeto inglés, puede asignar .ªçomo 0, "Bçomo 1, y así sucesivamente.

4. **Conversión del bloque a una matriz:**

   Cree una matriz utilizando los números asignados a las letras en el bloque. Asegúrese de que la matriz tenga la misma dimensión que la matriz clave.

5. **Cifrado:**

   Multiplique la matriz del bloque por la matriz clave.

   Si el tamaño del alfabeto utilizado es mayor que el número resultante de la multiplicación tome el módulo del resultado utilizando el tamaño del alfabeto. Esto asegurará que los números estén dentro del rango válido.

6. **Conversión del bloque cifrado a texto:**

   Convierta la matriz cifrada nuevamente a letras utilizando la asignación inversa de números a letras.

7. **Repita los pasos 4 al 6 para cada bloque del texto original.**

2.4. Pasos del Descifrado Hill

1. **Obtenga la matriz clave de descifrado:**

   Si la matriz clave de cifrado es una matriz invertible, puede calcular la matriz clave de descifrado invirtiendo la matriz clave de cifrado. En caso contrario, el descifrado no será posible.

2. **Preparación del texto cifrado:**

   Divida el texto cifrado en bloques de tamaño igual a la dimensión de la matriz clave. Asegúrese de que la longitud del texto cifrado sea un múltiplo del tamaño del bloque.

3. **Asignación de números a las letras:**

   Asigne a cada letra del bloque cifrado un número correspondiente según su posición en el alfabeto utilizado. Utilice la misma asignación que se utilizó durante el cifrado.

4. **Conversión del bloque cifrado a una matriz:**

   Cree una matriz utilizando los números asignados a las letras en el bloque cifrado. Asegúrese de que la matriz tenga la misma dimensión que la matriz clave.

5. **Descifrado:**

   Multiplique la matriz del bloque cifrado por la matriz clave de descifrado.

6. **Conversión del bloque descifrado a texto:**

   Convierta la matriz descifrada nuevamente a letras utilizando la asignación inversa de números a letras.

7. **Repita los pasos 4 al 6 para cada bloque del texto cifrado.**

3. Resultados

En este código se presenta una implementación del cifrado de Hill. El código consta de varias funciones que permiten generar una matriz clave aleatoria, cifrar un mensaje y descifrar un mensaje cifrado.

- `Importación de bibliotecas`: El código comienza importando las bibliotecas necesarias, como numpy y sympy, para realizar operaciones matemáticas y manipulación de matrices.
- `Función para generar una matriz clave aleatoria`: La función `generate_key(n)` genera una matriz clave aleatoria de tamaño $n \times n$. La matriz se genera seleccionando números enteros aleatorios en el rango de 0 a 26 y luego se verifica si la matriz es invertible calculando su determinante. Si el determinante es invertible módulo 27, se devuelve la matriz clave.
- `Función para cifrar un mensaje`: La función `encrypt(plaintext, key)` toma un mensaje de texto plano y una matriz clave como entrada. El mensaje se convierte a mayúsculas y se eliminan los espacios en blanco. Luego, se convierten los caracteres en números utilizando la codificación ASCII y se realiza la multiplicación de la matriz clave con bloques de caracteres del mensaje. El resultado se convierte nuevamente en caracteres y se concatena para formar el texto cifrado.
- `Función para descifrar un mensaje`: La función `decrypt(ciphertext, key)` toma un mensaje cifrado y una matriz clave como entrada. Se realiza el proceso inverso al cifrado, utilizando la matriz inversa modular de la clave para obtener el mensaje original. El resultado se convierte nuevamente en caracteres y se concatena para formar el texto descifrado.

```
import numpy as np
import sympy as sp

# Funcion para generar una matriz clave aleatoria invertible
def generate_key(n):
    while True:
        key = np.random.randint(0, 27, size=(n, n))
        det = int(np.round(np.linalg.det(key))) % 27
        try:
            det_inv = sp.mod_inverse(det, 27)
            return key
        except ValueError:
            pass

# Funcion para cifrar un mensaje usando la matriz clave
def encrypt(plaintext, key):
    n = key.shape[0]
    plaintext = plaintext.replace(" ", "")  # Eliminar espacios en blanco
    plaintext = plaintext.upper()  # Convertir a mayusculas
    plaintext = [ord(c) - ord("A") for c in plaintext]  # Convertir
        caracteres a numeros
    plaintext = np.array(plaintext)
    ciphertext = ""
    for i in range(0, len(plaintext), n):
        block = plaintext[i:i + n]
        if len(block) < n:
            block = np.append(block, [26]*(n - len(block)))  # Rellenar
                con 26s si el bloque es menor a n
        block = np.dot(key, block) % 27
        block = [chr(int(c) + ord("A")) if c != 26 else ' ' for c in
            block]  # Convertir numeros a caracteres
        ciphertext += "".join(block)
    return ciphertext

# Funcion para descifrar un mensaje usando la matriz clave
def decrypt(ciphertext, key):
    n = key.shape[0]
    ciphertext = [ord(c) - ord("A") if c != ' ' else 26 for c in
        ciphertext]  # Convertir caracteres a numeros
    ciphertext = np.array(ciphertext)
    decrypted_text = ""
    key_inv = sp.Matrix(key.tolist()).inv_mod(27).tolist()  # Calcular la
        matriz inversa de la clave
    key_inv = [[int(x) % 27 for x in row] for row in key_inv]
    key_inv = np.array(key_inv)
    print("Inverso Modular de la Matriz clave:\n",key_inv)
    for i in range(0, len(ciphertext), n):
        block = ciphertext[i:i + n]
        block = np.dot(key_inv, block) % 27
        block = [chr(int(c) + ord("A")) if c != 26 else ' ' for c in
```

```
            block]  # Convertir numeros a caracteres
46          decrypted_text += "".join(block)
47      return decrypted_text.strip()  # Eliminar espacios al final
48
49  # Obtener la palabra a cifrar por teclado
50  plaintext = input("Ingrese la palabra a cifrar: ")
51  key_size = int(input("Ingrese la dimension de la matriz clave: "))
52  # Generar una matriz clave aleatoria de tamano nxn
53  key = generate_key(key_size)
54
55  print("Matriz clave:")
56  print(key)
57
58  ciphertext = encrypt(plaintext, key)
59  print("Texto cifrado:", ciphertext)
60
61  decrypted_text = decrypt(ciphertext, key)
62  print("Texto descifrado:", decrypted_text)
```

Listing 1: Descripción del código

En la ejecución del programa se observa que solo se podrá apreciar de forma estética la matriz clave hasta la dimensión 24. Después de eso, esta comenzará a distorsionarse debido al espacio en pantalla.

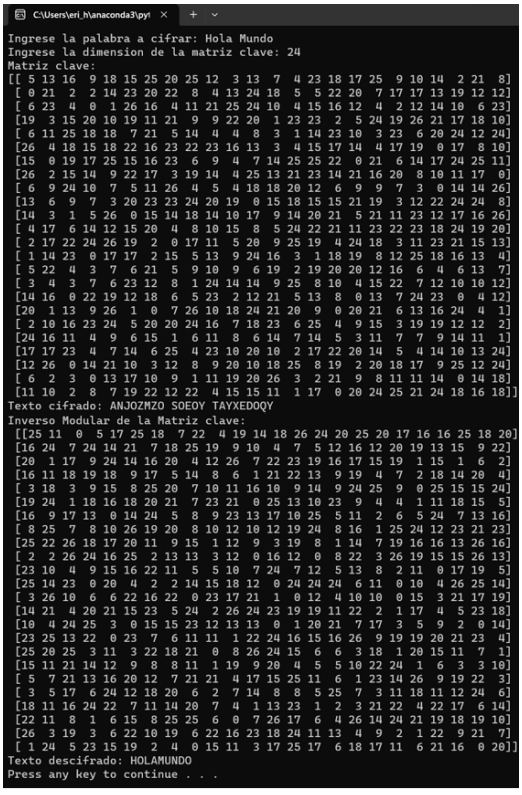

Figura 1: Ejecucion con matriz clave de dimension 24

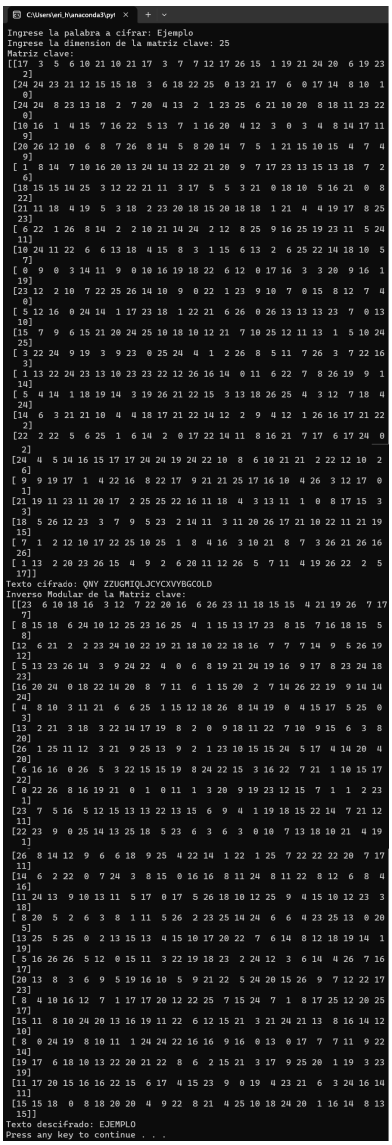

Figura 2: Ejecucion con matriz clave de dimension 25

4. Conclusiones

a. La determinante de la matriz clave debe ser primo entre sí con el módulo de encriptación, es decir, sea A, la matriz clave y n el modulo de encriptación: 27, entonces: MCD(det(A), n) = 1.

b. El cifrado de Hill, a través de su código, permitirá transformar los datos originales en una forma ilegible, utilizando una clave matriz, para proteger su confidencialidad y así poder implementarlo para cifrar datos sensibles o confidenciales. Además del cifrado, también podrás utilizar el código para descifrar los datos cifrados previamente. Al proporcionar la clave matriz correcta, podrás revertir el proceso de cifrado y recuperar los datos originales.

c. El cifrado de Hill permite proteger la privacidad y confidencialidad de los datos durante la transmisión o el almacenamiento. Al cifrar los datos antes de enviarlos o guardarlos, te aseguras de que solo las personas autorizadas puedan acceder a la información, ya que, sin la clave correcta, los datos cifrados son incomprensibles.

d. El cifrado, se basa en conceptos matemáticos, como por ejemplo: las operaciones de álgebra lineal y la teoría de números.

e. Además, es adecuado para: pequeñas cantidades de datos, comunicaciones seguras en redes,

protección de datos locales, transmisión segura de datos, entre otros.

f. Sin embargo, puede no ser la opción preferida en los siguientes casos: Grandes volúmenes de datos,autenticación de datos, escenarios de alta seguridad, volviendolo vulnerable a ciertos ataques como el criptoanalisis diferencial.

g. Se concluye que este tipo de proyecto promueve la comprensión y el interés en temas relacionados con la seguridad de la información, la criptografía y la protección de datos.

h. Con la realización de este proyecto se ha querido diseñar una herramienta con la que los alumnos puedan familiarizarse con el uso de un criptosistema. Con la ayuda de esta herramienta los alumnos pueden comprobar de una manera práctica los conocimientos adquiridos referentes a este sistema de cifrado.

5. Recomendaciones

a. Hay que asegurarnos de utilizar una matriz de clave de un tamaño relativamente adecuado. Es más, esta matriz debe de ser invertible y cuadrada de tamaño mayor o igual a 2X2 para evitar debilidades en la seguridad.

b. Evitar el uso de matrices predefinidas o patrones predecibles que puedan comprometer la seguridad del cifrado.

c. Utilizar técnicas adecuadas de gestión de claves, así como el almacenamiento encriptado y el acceso restringido a las claves.

d. Evitar compartir las claves y almacenarlas junto con los datos cifrados.

6. Bibliografía

Stallings, W. (2017). Criptografía y seguridad de la información. Pearson Educación.

Immune Technology Institute. Tipos de criptografía ¿Cuáles son los más comunes para la privacidad de las comunicaciones? https://immune.institute/blog/tipos-de-criptografia-seguridad-online/