Documentación de Técnicas de Cifrado y Permutación

1. Introducción

Este documento describe diversas técnicas de cifrado implementadas en los lenguajes de programación Python y Java. Las técnicas aquí detalladas son parte fundamental de la criptografía clásica, que si bien ha sido superada por métodos modernos, aún representa una base sólida para comprender los principios fundamentales del cifrado de información.

2. Permutaciones de Caracteres (Anagramas)

Lenguaje: Python

Este algoritmo genera todas las permutaciones posibles de una palabra ingresada por el usuario.

Pasos del algoritmo:

1. 1. Solicitar al usuario que ingrese una palabra sin espacios.
2. 2. Eliminar espacios en blanco alrededor del texto.
3. 3. Usar `itertools.permutations` para obtener todas las combinaciones posibles de los caracteres.
4. 4. Convertir las combinaciones a un conjunto para eliminar duplicados.
5. 5. Ordenar alfabéticamente las permutaciones.
6. 6. Mostrar las primeras 10 permutaciones y el número total generado.

3. Cifrado por Permutación de Filas

Lenguaje: Python

Este algoritmo organiza los caracteres en una matriz por filas y luego lee la matriz en orden de filas.

Pasos del algoritmo:

1. Solicitar al usuario el valor de n (filas) y el mensaje a cifrar.
2. Eliminar espacios del mensaje.
3. Verificar que el mensaje tenga como máximo n × n caracteres.
4. Rellenar el mensaje con '*' si no se completa la matriz.
5. Crear una matriz de n filas por n columnas y llenarla por filas.
6. Leer la matriz por filas y generar el mensaje cifrado.

7. Imprimir la matriz de cifrado, el mensaje original y el cifrado.

4. Cifrado por Permutación de Columnas

Lenguaje: Python

Este algoritmo organiza los caracteres en una matriz por filas y luego lee la matriz en orden de columnas.

Pasos del algoritmo:

1. Solicitar al usuario el valor de n (columnas) y el mensaje a cifrar.
2. Eliminar espacios del mensaje.
3. Verificar que el mensaje tenga como máximo n × n caracteres.
4. Rellenar el mensaje con '*' si no se completa la matriz.
5. Crear una matriz de n filas por n columnas y llenarla por filas.
6. Leer la matriz por columnas y generar el mensaje cifrado.
7. Imprimir la matriz de cifrado, el mensaje original y el cifrado.

5. Cifrado por Sustitución Monoalfabética (César)

Lenguaje: Python

Este algoritmo desplaza cada letra del alfabeto un número fijo de posiciones.

Pasos del algoritmo:

1. Solicitar al usuario el mensaje y el número de desplazamiento n.
2. Construir el alfabeto original (a-z).
3. Generar el alfabeto cifrado desplazando cada letra n posiciones.
4. Reemplazar cada letra del mensaje con su equivalente cifrado.
5. Imprimir el alfabeto original, el cifrado, el mensaje original y el mensaje cifrado.

6. Cifrado por Sustitución Polialfabética (Vigenère)

Lenguaje: Python

Este algoritmo cifra cada letra del mensaje con una letra distinta, de acuerdo a una clave que se repite.

Pasos del algoritmo:

1. Solicitar al usuario el mensaje y la clave de cifrado.
2. Repetir la clave hasta igualar la longitud del mensaje.
3. Para cada letra del mensaje, calcular su valor cifrado desplazándola según la letra correspondiente de la clave.
4. Imprimir el mensaje original, la clave utilizada y el mensaje cifrado.

7. Cifrado con Tabla de Coordenadas

Lenguaje: Python

Este algoritmo reemplaza cada letra del mensaje por su posición (fila, columna) en una tabla predefinida.

Pasos del algoritmo:

1. Crear una matriz de cifrado omitiendo la letra 'w'.
2. Solicitar al usuario el mensaje a cifrar.
3. Recorrer cada letra del mensaje y buscar su coordenada (fila, columna) en la matriz.
4. Si una letra no está en la matriz, se representa como '**'.
5. Imprimir la matriz de cifrado, el mensaje original y el mensaje cifrado en coordenadas.

8. Conclusiones

Estos algoritmos ofrecen una introducción práctica a la criptografía clásica. Aunque su nivel de seguridad es bajo en entornos reales, son ideales para entender los principios de sustitución, permutación y uso de claves. Python y Java ofrecen herramientas robustas para implementar estos métodos de forma clara y efectiva.

9. Referencias

[1] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed. New York: Wiley, 1996.
[2] D. R. Stinson and M. B. Paterson, *Cryptography: Theory and Practice*, 4th ed. CRC Press, 2018.
[3] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed. Pearson, 2017.
[4] Python Software Foundation. 'Python 3 Documentation'. [Online]. Available: https://docs.python.org/3/
[5] Oracle. 'Java Platform, Standard Edition Documentation'. [Online]. Available: https://docs.oracle.com/javase/