EncriptadorMatricesMagicas

Andrés

June 5, 2024

Historia de la Criptografía y Aplicación en Criptografía

La criptografía, el arte y la ciencia de escribir mensajes de forma secreta o codificada, ha sido una parte fundamental de la comunicación humana desde tiempos antiguos. Desde los jeroglíficos egipcios hasta los sofisticados sistemas de cifrado utilizados en la era digital, la criptografía ha evolucionado enormemente a lo largo de la historia.

Orígenes de la Criptografía Los primeros registros de criptografía se remontan a la antigua Mesopotamia, donde se utilizaron técnicas rudimentarias para ocultar mensajes. Sin embargo, fue en la antigua Grecia donde se desarrollaron algunos de los primeros métodos formales de cifrado, como la cifra de César, que implicaba desplazar cada letra del alfabeto un número fijo de posiciones.

Avances en Criptografía Durante la Edad Media y el Renacimiento, la criptografía experimentó avances significativos con la invención de métodos más complejos, como el cifrado de Vigenère y la cifra de Alberti. Estos sistemas permitían una mayor seguridad al ocultar patrones en el texto cifrado, dificultando su descifrado sin la clave adecuada.

Criptografía en la Era Moderna En los tiempos modernos, con el advenimiento de la computación y la era digital, la criptografía se ha vuelto aún más crucial. Se utilizan algoritmos de cifrado avanzados, como AES (Estándar de Cifrado Avanzado) y RSA (Algoritmo de Cifrado de Clave Pública), para proteger la información sensible en línea y garantizar la privacidad y la seguridad en las comunicaciones.

Aplicación Práctica: Ejercicio de Criptografía El ejercicio presentado aquí muestra una implementación básica de criptografía utilizando la multiplicación de matrices para encriptar y desencriptar texto. Aunque este método es bastante simplificado en comparación con los sistemas de cifrado modernos, ilustra los principios fundamentales de la criptografía, como la utilización de claves para transformar el texto original en un formato ilegible y luego revertir este proceso para recuperar el mensaje original.

Este ejercicio nos recuerda que la criptografía no solo es una herramienta importante para proteger la información, sino también un campo fascinante que ha evolucionado a lo largo de los siglos, desde las técnicas más simples hasta los sistemas de cifrado más complejos utilizados en la actualidad.

1 Historia de las Matrices y los Cuadrados Mágicos

Las matrices y los cuadrados mágicos tienen una historia fascinante que se remonta a varios siglos atrás. Desde su descubrimiento hasta sus diversas aplicaciones en diferentes campos, han dejado una marca significativa en las matemáticas, la criptografía y otras áreas.

1.1 Matrices: Un Breve Resumen Histórico

Las matrices, una estructura de datos bidimensional, tienen sus raíces en la antigua China, donde los matemáticos chinos las utilizaban para resolver sistemas de ecuaciones lineales. Sin embargo, fue en el siglo XIX cuando los matemáticos europeos, como Gauss y Cayley, formalizaron el estudio de las matrices y les dieron el nombre que utilizamos hoy en día.

1.2 Cuadrados Mágicos: Origen y Misterio

Los cuadrados mágicos, una disposición de números en una cuadrícula donde la suma de cada fila, columna y diagonal principal es la misma, han intrigado a las mentes curiosas durante siglos. Se dice que los primeros cuadrados mágicos se encontraron en la antigua India y China, donde se utilizaban con propósitos religiosos y místicos.

1.3 Aplicaciones de las Matrices y los Cuadrados Mágicos

1.3.1 Matrices en Criptografía

Las matrices juegan un papel crucial en la criptografía moderna. Los algoritmos de cifrado, como el cifrado de Hill, utilizan matrices para transformar y encriptar mensajes de manera segura. Esta aplicación demuestra cómo las matrices tienen un impacto significativo en la seguridad de la información y la protección de datos sensibles.

1.3.2 Cuadrados Mágicos en la Cultura y el Entretenimiento

Además de su importancia matemática, los cuadrados mágicos han capturado la imaginación de artistas, escritores y aficionados al entretenimiento. Han aparecido en obras literarias, obras de arte y juegos de mesa, añadiendo un elemento de misterio y enigma a diversas formas de expresión creativa.

1.4 Documentación de Funciones de Encriptación y Desencriptación

1.4.1 `asignar_numero_letra(letra)`

Función que asigna un número único a cada letra del alfabeto español.

Parámetros: - `letra`: La letra del alfabeto a la que se le asignará un número.

Valor de retorno:
- Entero: El número asignado a la letra.

1.4.2 `encriptar_letra(letra)`

Función que encripta una letra utilizando la multiplicación de matrices.

Parámetros: - `letra`: La letra que se desea encriptar.

Valor de retorno:
- NumPy Array: La matriz resultante de la encriptación de la letra.

1.4.3 `desencriptar_letra(matriz_letra_encriptada)`

Función que desencripta una letra utilizando la matriz inversa.

Parámetros: - `matriz_letra_encriptada`: La matriz que representa la letra encriptada.

Valor de retorno:
- String: La letra desencriptada.

1.4.4 `encriptar_palabra(palabra)`

Función que encripta una palabra completa utilizando la función `encriptar_letra`.

Parámetros: - `palabra`: La palabra que se desea encriptar.

Valor de retorno:
- List: Una lista de matrices que representan la encriptación de cada letra de la palabra.

1.4.5 `desencriptar_palabra(matrices_palabra_encriptada)`

Función que desencripta una palabra completa uUtilizando la función `desencriptar_letra`.

Parámetros: - 'matrices_palabra_encri

Se importa la biblioteca NumPy para realizar operaciones con matrices de manera eficiente.

MODELO 1

2 Encriptador y Desencriptador de Palabras (Modelo 1)

Este programa permite al usuario encriptar y desencriptar palabras utilizando el método de multiplicación de matrices con una matriz predefinida.

2.1 Funcionamiento del Programa

El programa utiliza una matriz base predefinida para encriptar y desencriptar palabras. El usuario solo necesita ingresar la palabra que desea encriptar, y el programa aplicará la matriz base para realizar la encriptación. Luego, muestra las matrices resultantes y el texto encriptado. Para desencriptar, utiliza la matriz inversa de la matriz base para revertir el proceso y mostrar la palabra original.

2.2 Perfección del Programa

2.2.1 Eficiencia en la Encriptación

- Al utilizar una matriz predefinida, el proceso de encriptación es rápido y eficiente. El usuario no necesita ingresar la matriz cada vez que desee encriptar una palabra, lo que simplifica el proceso y ahorra tiempo.

2.2.2 Simplicidad en la Interfaz

- El programa presenta una interfaz simple donde el usuario solo necesita ingresar la palabra a encriptar. Esto hace que el proceso sea fácil de entender y seguir, incluso para usuarios no técnicos.

2.2.3 Exactitud en la Desencriptación

- Al utilizar la matriz inversa de la matriz base predefinida, el programa puede desencriptar con precisión las palabras encriptadas. Esto garantiza que la información original se recupere correctamente sin pérdida de datos.

2.3 Conclusión

El modelo 1 ofrece una solución eficiente y simple para la encriptación y desencriptación de palabras utilizando una matriz predefinida. Su enfoque directo y su capacidad para mantener la exactitud en el proceso lo convierten en una herramienta útil para proteger la privacidad de las comunicaciones.

[1]:
```
import numpy as np

# Definimos el alfabeto español
alfabeto = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ"

# Creamos una función para asignar números a las letras
def asignar_numero_letra(letra):
    """
    Función que asigna un número único a cada letra del alfabeto español.
    """
    return alfabeto.index(letra) + 1

# Definimos una matriz constante para encriptar
matriz_clave = np.array([[8,1,6],
                         [3,5,7 ],
                         [4,9,2]])

# Calculamos la matriz inversa de la matriz de clave
matriz_inversa = np.linalg.inv(matriz_clave)

# Creamos una función para encriptar una letra
def encriptar_letra(letra):
    """
    Función que encripta una letra utilizando la multiplicación de matrices.
    """
    numero_letra = asignar_numero_letra(letra)
    return np.dot(numero_letra, matriz_clave)

# Creamos una función para desencriptar una letra
def desencriptar_letra(matriz_letra_encriptada):
    """
    Función que desencripta una letra utilizando la matriz inversa.
    """
    matriz_letra_desencriptada = np.dot(matriz_letra_encriptada, matriz_inversa)
    letras_desencriptadas = [alfabeto[int(np.round(numero)) - 1] for numero in␣
 ↪matriz_letra_desencriptada.flatten()]
```

```
    return ''.join(letras_desencriptadas)

# Creamos una función para encriptar una palabra
def encriptar_palabra(palabra):
    """
    Función que encripta una palabra completa utilizando la función␣
 ↪'encriptar_letra'.
    """
    palabra_mayusculas = palabra.upper()
    matriz_palabra = []
    for letra in palabra_mayusculas:
        matriz_letra_encriptada = encriptar_letra(letra)
        matriz_palabra.append(matriz_letra_encriptada)
    return matriz_palabra

# Creamos una función para desencriptar una palabra
def desencriptar_palabra(matrices_palabra_encriptada):
    """
    Función que desencripta una palabra completa utilizando la función␣
 ↪'desencriptar_letra'.
    """
    palabra_desencriptada = ""
    for matriz_letra_encriptada in matrices_palabra_encriptada:
        letra_desencriptada = desencriptar_letra(matriz_letra_encriptada)
        palabra_desencriptada += letra_desencriptada[-1] # Tomamos solo la␣
 ↪última letra desencriptada
    return palabra_desencriptada


# Ejemplo de uso
palabra_a_encriptar = input("Escribe la palabra a encriptar")
matrices_palabra_encriptada = encriptar_palabra(palabra_a_encriptar)

print("Matrices de la palabra encriptada:")
for matriz in matrices_palabra_encriptada:
    print("\n", matriz, "\n")

# Desencriptamos la palabra
palabra_desencriptada = desencriptar_palabra(matrices_palabra_encriptada)

# Imprimimos la palabra desencriptada
print("Palabra desencriptada:", palabra_desencriptada)
```

```
Escribe la palabra a encriptar AMOR

Matrices de la palabra encriptada:

 [[8 1 6]
```

```
[3 5 7]
[4 9 2]]


[[104  13  78]
[ 39  65  91]
[ 52 117  26]]


[[128  16  96]
[ 48  80 112]
[ 64 144  32]]


[[152  19 114]
[ 57  95 133]
[ 76 171  38]]

Palabra desencriptada: AMOR
```

MODELO 2

[2]:
```python
import numpy as np

# Definimos el alfabeto español
alfabeto = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ"

# Creamos una función para asignar números a las letras
def asignar_numero_letra(letra):
    """
    Función que asigna un número único a cada letra del alfabeto español.
    """
    return alfabeto.index(letra) + 1

def obtener_matriz_base():
    """
    Función para solicitar al usuario los valores de la matriz base y␣
 ↪convertirlos en una matriz NumPy.
    """
    print("Ingrese los valores de la matriz base (3 filas x 3 columnas):")
    matriz_base = []
    for i in range(3):
        fila = input(f"Ingrese los valores de la fila {i+1}, separados por␣
 ↪espacio: ")
        valores = fila.split()
        fila_numeros = [int(valor) for valor in valores]
        matriz_base.append(fila_numeros)
    return np.array(matriz_base)
```

```
# Solicitar al usuario la matriz base
matriz_clave = obtener_matriz_base()

# Calculamos la matriz inversa de la matriz de clave
matriz_inversa = np.linalg.inv(matriz_clave)

# Creamos una función para encriptar una letra
def encriptar_letra(letra):
    """
    Función que encripta una letra utilizando la multiplicación de matrices.
    """
    numero_letra = asignar_numero_letra(letra)
    return np.dot(numero_letra, matriz_clave)

# Creamos una función para desencriptar una letra
def desencriptar_letra_v2(matriz_letra_encriptada):
    """
    Función que desencripta una letra utilizando la matriz inversa.
    """
    matriz_letra_desencriptada = np.dot(matriz_letra_encriptada, matriz_inversa)

    # Ajustamos los valores para que estén dentro del rango del alfabeto
    ajustados = matriz_letra_desencriptada % len(alfabeto)

    # Convertimos los números ajustados en letras
    letras_desencriptadas = [alfabeto[int(np.round(numero)) - 1] for numero in␣
↪ajustados.flatten()]

    return ''.join(letras_desencriptadas)

# Creamos una función para encriptar una palabra
def encriptar_palabra(palabra):
    """
    Función que encripta una palabra completa utilizando la función␣
↪'encriptar_letra'.
    """
    # Eliminamos los espacios en blanco de la palabra ingresada
    palabra_sin_espacios = palabra.replace(" ", "")
    # Convertimos la palabra a mayúsculas para garantizar consistencia
    palabra_mayusculas = palabra_sin_espacios.upper()
    # Inicializamos una lista para almacenar las matrices encriptadas de cada␣
↪letra
    matriz_palabra = []
    # Iteramos sobre cada letra de la palabra
    for letra in palabra_mayusculas:
        # Encriptamos la letra actual y obtenemos su matriz encriptada
```

```
        matriz_letra_encriptada = encriptar_letra(letra)
        # Agregamos la matriz encriptada de la letra a la lista
        matriz_palabra.append(matriz_letra_encriptada)
    # Retornamos la lista de matrices encriptadas de las letras de la palabra
    return matriz_palabra


def desencriptar_palabra(matrices_palabra_encriptada):
    """
    Función que desencripta una palabra completa utilizando la función␣
↪'desencriptar_letra_v2'.
    """
    palabra_desencriptada = ""
    for matriz_letra_encriptada in matrices_palabra_encriptada:
        letra_desencriptada = desencriptar_letra_v2(matriz_letra_encriptada)
        palabra_desencriptada += letra_desencriptada[-1] # Tomamos solo la␣
↪última letra desencriptada
    return palabra_desencriptada

# Ejemplo de uso
palabra_a_encriptar = input("Escribe la palabra a encriptar: ")
matrices_palabra_encriptada = encriptar_palabra(palabra_a_encriptar)

print("Matrices de la palabra encriptada:")
for matriz in matrices_palabra_encriptada:
    print("\n", matriz, "\n")

# Desencriptamos la palabra
palabra_desencriptada = desencriptar_palabra(matrices_palabra_encriptada)

# Imprimimos la palabra desencriptada
print("Palabra desencriptada:", palabra_desencriptada)
```

```
Ingrese los valores de la matriz base (3 filas x 3 columnas):

Ingrese los valores de la fila 1, separados por espacio:  8 1 6
Ingrese los valores de la fila 2, separados por espacio:  3 5 7
Ingrese los valores de la fila 3, separados por espacio:  4 9 2
Escribe la palabra a encriptar:  amores que matan

Matrices de la palabra encriptada:

 [[8 1 6]
 [3 5 7]
 [4 9 2]]


 [[104  13  78]
```

```
[ 39  65  91]
[ 52 117  26]]


[[128  16  96]
[ 48  80 112]
[ 64 144  32]]


[[152  19 114]
[ 57  95 133]
[ 76 171  38]]


[[40  5 30]
[15 25 35]
[20 45 10]]


[[160  20 120]
[ 60 100 140]
[ 80 180  40]]


[[144  18 108]
[ 54  90 126]
[ 72 162  36]]


[[176  22 132]
[ 66 110 154]
[ 88 198  44]]


[[40  5 30]
[15 25 35]
[20 45 10]]


[[104  13  78]
[ 39  65  91]
[ 52 117  26]]


[[8 1 6]
[3 5 7]
[4 9 2]]
```

```
[[168  21 126]
[ 63 105 147]
[ 84 189  42]]


[[8 1 6]
[3 5 7]
[4 9 2]]


[[112  14  84]
[ 42  70  98]
[ 56 126  28]]

Palabra desencriptada: AMORESQUEMATAN
```

CONCLUSIONES DEL EXPERIMENTO DE ENCRIPTACIÓN

Usar un cuadrado mágico como matriz clave en nuestro algoritmo de cifrado tiene un atractivo especial y una implicación interesante: al encriptar una palabra utilizando una matriz mágica como clave, las matrices resultantes de la encriptación también serán cuadrados mágicos.

Este fenómeno es notable porque resalta una propiedad especial de los cuadrados mágicos: la preservación de la estructura mágica a través de ciertas operaciones matemáticas, como la multiplicación de matrices en nuestro caso. Es decir, cuando multiplicamos la matriz clave (que es un cuadrado mágico) por una matriz de letras (que representa la palabra a encriptar), el resultado es una serie de matrices que mantienen la propiedad de ser cuadrados mágicos.

Esta observación demuestra una conexión intrigante entre las propiedades matemáticas de los cuadrados mágicos y el proceso de encriptación. No solo proporciona una forma interesante de generar cuadrados mágicos adicionales, sino que también resalta la versatilidad de estos objetos en diferentes contextos, como la criptografía. La utilización de cuadrados mágicos como claves en nuestro algoritmo de cifrado aporta una capa adicional de seguridad y complejidad al proceso de encriptación. La selección de una matriz específica como clave no solo determina la forma en que se encriptará el mensaje, sino que también influirá en las matrices resultantes durante el proceso de encriptación. Esto significa que cada clave generará un conjunto único de matrices encriptadas, lo que dificulta enormemente la tarea de descifrar el mensaje sin conocer la clave correcta.

Una de las ventajas más destacadas de esta técnica es la diversidad y la imprevisibilidad inherentes a los cuadrados mágicos. Con una cantidad infinita de combinaciones posibles, la probabilidad de adivinar la clave correcta se vuelve extremadamente baja, lo que refuerza la seguridad del sistema de cifrado. En lugar de depender únicamente de la confidencialidad de la clave, la robustez del algoritmo se basa en la complejidad de los cuadrados mágicos y su relación con el proceso de encriptación. En resumen, la integración de cuadrados mágicos como claves ofrece una solución segura y efectiva para proteger la información sensible mediante cifrado.

En resumen, el uso de cuadrados mágicos como matrices clave en nuestro algoritmo de cifrado no solo añade una capa de seguridad, sino que también revela una fascinante interacción entre las propiedades matemáticas de los cuadrados mágicos y el proceso de encriptación.

Operación	Símbolo	Descripción
Suma de matrices	(A + B)	Se suma cada elemento correspondiente de la matriz (A) con el elemento correspondiente de la matriz (B).
Resta de matrices	(A - B)	Se resta cada elemento correspondiente de la matriz (B) del elemento correspondiente de la matriz (A).
Multiplicación de matrices	(A * B)	Se realiza multiplicando cada fila de la matriz (A) por cada columna de la matriz (B) y sumando los productos.
Multiplicación por escalar	(k * A)	Cada elemento de la matriz (A) se multiplica por el escalar (k).
Transposición de matriz	(A^T)	Se intercambian las filas y columnas de la matriz (A).

2.4 Multiplicación de la Matriz Clave

En nuestro algoritmo de encriptación, utilizamos una matriz clave para transformar las letras de una palabra en matrices numéricas encriptadas. Aquí explicamos cómo funciona este proceso:

1. **Asignación de Números a las Letras**: Cada letra del alfabeto se asigna a un número único. Por ejemplo, la letra "A" se asigna al número 1, la letra "B" al número 2, y así sucesivamente.
2. **Creación de la Matriz Clave**: La matriz clave es una matriz cuadrada que utilizamos como multiplicador en nuestro proceso de encriptación. Esta matriz puede ser proporcionada por el usuario o generada por el sistema.
3. **Multiplicación de la Matriz Clave por los Valores de las Letras**: Para encriptar una letra, multiplicamos su valor numérico asignado por la matriz clave. Este proceso se repite para cada letra de la palabra.
4. **Creación de las Matrices Encriptadas**: Después de la multiplicación, obtenemos una matriz numérica que representa la letra encriptada. Este proceso se repite para cada letra de la palabra, lo que resulta en una serie de matrices encriptadas.
5. **Desplazamiento y Ajuste**: Posteriormente, estos valores numéricos pueden ser ajustados para asegurarse de que estén dentro del rango del alfabeto y convertidos nuevamente en letras. Esto se realiza generalmente tomando el módulo del tamaño del alfabeto.

Este proceso de multiplicación de la matriz clave por los valores de las letras nos permite transformar una palabra en una serie de matrices encriptadas, lo que proporciona una capa adicional de seguridad en nuestro sistema de encriptación.

MODELO 3

2.5 Encriptación y Desencriptación con Suma de Matrices

En este ejemplo a continuación, en lugar de multiplicar la matriz de la letra por la matriz clave, utilizamos la operación de suma. Aquí explicamos cómo funciona este proceso:

2.5.1 Encriptación de una Letra

1. **Obtención del Número de la Letra:** Cada letra del alfabeto se asigna a un número único, comenzando desde 1. Por ejemplo, la letra 'A' es 1, 'B' es 2, y así sucesivamente.
2. **Suma con la Matriz Clave:** Para encriptar una letra, sumamos su valor numérico con la matriz clave. Esto se hace elemento por elemento, sumando cada número de la matriz de la letra con su correspondiente número en la matriz clave.
3. **Resultado:** El resultado es una nueva matriz que representa la letra encriptada. Esta matriz puede contener números que pueden estar fuera del rango del alfabeto.

2.5.2 Desencriptación de una Letra

1. **Resta con la Matriz Clave:** Para desencriptar una letra, restamos la matriz encriptada con la matriz clave. Esto se hace también elemento por elemento, restando cada número de la matriz encriptada con su correspondiente número en la matriz clave.
2. **Ajuste de Valores:** Después de la resta, algunos números pueden estar fuera del rango del alfabeto. Para corregir esto, ajustamos los valores para que estén dentro del rango del alfabeto. Esto se logra tomando el módulo del valor resultante con la longitud del alfabeto.
3. **Obtención de la Letra Desencriptada:** Finalmente, convertimos los números ajustados en letras, utilizando su valor numérico para encontrar la letra correspondiente en el alfabeto.

Este método de encriptación y desencriptación proporciona una forma alternativa de proteger y recuperar información utilizando matrices y operaciones simples de suma.

[9]:
```
import numpy as np

# Definimos el alfabeto español
alfabeto = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ"

# Creamos una función para asignar números a las letras
def asignar_numero_letra(letra):
    """
    Función que asigna un número único a cada letra del alfabeto español.
    """
    return alfabeto.index(letra) + 1

def obtener_matriz_base():
    """
    Función para solicitar al usuario los valores de la matriz base y␣
 ↪convertirlos en una matriz NumPy.
    """
    print("Ingrese los valores de la matriz base (3 filas x 3 columnas):")
    matriz_base = []
    for i in range(3):
        fila = input(f"Ingrese los valores de la fila {i+1}, separados por␣
 ↪espacio: ")
        valores = fila.split()
```

```
        fila_numeros = [int(valor) for valor in valores]
        matriz_base.append(fila_numeros)
    return np.array(matriz_base)

# Solicitar al usuario la matriz base
matriz_clave = obtener_matriz_base()

# Creamos una función para encriptar una letra
def encriptar_letra(letra):
    """
    Función que encripta una letra sumando su valor numérico con la matriz clave.
    """
    # Convertimos la letra en un número según su posición en el alfabeto
    numero_letra = asignar_numero_letra(letra)
    # Sumamos el número de la letra con la matriz clave
    return numero_letra + matriz_clave

def desencriptar_letra(matriz_letra_encriptada):
    """
    Función que desencripta una letra sumando su valor numérico con la matriz␣
 ↪inversa.
    """
    # Restamos la matriz encriptada con la matriz clave para deshacer la␣
 ↪encriptación
    matriz_letra_desencriptada = matriz_letra_encriptada - matriz_clave

    # Ajustamos los valores para que estén dentro del rango del alfabeto
    ajustados = matriz_letra_desencriptada % len(alfabeto)

    # Convertimos los números ajustados en letras
    letras_desencriptadas = [alfabeto[int(np.round(numero)) - 1] for numero in␣
 ↪ajustados.flatten()]

    return letras_desencriptadas[0]  # Devolvemos solo la primera letra␣
 ↪desencriptada



# Creamos una función para encriptar una palabra
def encriptar_palabra(palabra):
    """
    Función que encripta una palabra completa utilizando la función␣
 ↪'encriptar_letra'.
    """
    palabra_mayusculas = palabra.upper()
    matriz_palabra = []
    for letra in palabra_mayusculas:
```

```
        matriz_letra_encriptada = encriptar_letra(letra)
        matriz_palabra.append(matriz_letra_encriptada)
    return matriz_palabra

# Creamos una función para desencriptar una palabra
def desencriptar_palabra(matrices_palabra_encriptada):
    """
    Función que desencripta una palabra completa utilizando la función␣
 ↪'desencriptar_letra'.
    """
    palabra_desencriptada = ""
    for matriz_letra_encriptada in matrices_palabra_encriptada:
        letra_desencriptada = desencriptar_letra(matriz_letra_encriptada)
        palabra_desencriptada += letra_desencriptada[-1] # Tomamos solo la␣
 ↪última letra desencriptada
    return palabra_desencriptada

# Ejemplo de uso
palabra_a_encriptar = input("Escribe la palabra a encriptar: ")
matrices_palabra_encriptada = encriptar_palabra(palabra_a_encriptar)

print("Matrices de la palabra encriptada:")
for matriz in matrices_palabra_encriptada:
    print("\n", matriz, "\n")

# Desencriptamos la palabra
palabra_desencriptada = desencriptar_palabra(matrices_palabra_encriptada)

# Imprimimos la palabra desencriptada
print("Palabra desencriptada:", palabra_desencriptada)
```

```
Ingrese los valores de la matriz base (3 filas x 3 columnas):

Ingrese los valores de la fila 1, separados por espacio:  8 1 6
Ingrese los valores de la fila 2, separados por espacio:  3 5 7
Ingrese los valores de la fila 3, separados por espacio:  4 9 2
Escribe la palabra a encriptar:  amor

Matrices de la palabra encriptada:

 [[ 9  2  7]
 [ 4  6  8]
 [ 5 10  3]]


 [[21 14 19]
 [16 18 20]
 [17 22 15]]
```

```
[[24 17 22]
[19 21 23]
[20 25 18]]


[[27 20 25]
[22 24 26]
[23 28 21]]

Palabra desencriptada: AMOR
```

MODELO 4

e las comunicaciones.

[13]:
```
import numpy as np

# Definimos el alfabeto español
alfabeto = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ"

def asignar_numero_letra(letra):
    """
    Función que asigna un número único a cada letra del alfabeto español.
    """
    return alfabeto.index(letra) + 1

def obtener_matriz_base():
    """
    Función para solicitar al usuario los valores de la matriz base y␣
 ↪convertirlos en una matriz NumPy.
    """
    print("Ingrese los valores de la matriz base (3 filas x 3 columnas):")
    matriz_base = []
    for i in range(3):
        fila = input(f"Ingrese los valores de la fila {i+1}, separados por␣
 ↪espacio: ")
        valores = fila.split()
        fila_numeros = [int(valor) for valor in valores]
        matriz_base.append(fila_numeros)
    return np.array(matriz_base)

# Solicitar al usuario la matriz base
matriz_clave = obtener_matriz_base()

def encriptar_letra_multiplicacion(letra):
    """
```

```
    Función que encripta una letra utilizando la multiplicación de matrices.
    """
    numero_letra = asignar_numero_letra(letra)
    return np.dot(numero_letra, matriz_clave)

def desencriptar_letra_multiplicacion(matriz_letra_encriptada):
    """
    Función que desencripta una letra utilizando la matriz inversa.
    """
    matriz_letra_desencriptada = np.dot(matriz_letra_encriptada, np.linalg.
 ↪inv(matriz_clave))
    ajustados = matriz_letra_desencriptada % len(alfabeto)
    letras_desencriptadas = [alfabeto[int(np.round(numero)) - 1] for numero in␣
 ↪ajustados.flatten()]
    return ''.join(letras_desencriptadas)

def encriptar_letra_suma(letra):
    """
    Función que encripta una letra sumando su valor numérico con la matriz clave.
    """
    numero_letra = asignar_numero_letra(letra)
    return numero_letra + matriz_clave

def desencriptar_letra_suma(matriz_letra_encriptada):
    """
    Función que desencripta una letra sumando su valor numérico con la matriz␣
 ↪inversa.
    """
    matriz_letra_desencriptada = matriz_letra_encriptada - matriz_clave
    ajustados = matriz_letra_desencriptada % len(alfabeto)
    letras_desencriptadas = [alfabeto[int(np.round(numero)) - 1] for numero in␣
 ↪ajustados.flatten()]
    return ''.join(letras_desencriptadas)

def encriptar_palabra_multiplicacion(palabra):
    """
    Función que encripta una palabra completa utilizando la función␣
 ↪'encriptar_letra_multiplicacion'.
    """
    palabra_mayusculas = palabra.upper()
    matriz_palabra = []
    for letra in palabra_mayusculas:
        matriz_letra_encriptada = encriptar_letra_multiplicacion(letra)
        matriz_palabra.append(matriz_letra_encriptada)
    return matriz_palabra

def encriptar_palabra_suma(palabra):
```

```
    """
    Función que encripta una palabra completa utilizando la función␣
↪'encriptar_letra_suma'.
    """
    palabra_mayusculas = palabra.upper()
    matriz_palabra = []
    for letra in palabra_mayusculas:
        matriz_letra_encriptada = encriptar_letra_suma(letra)
        matriz_palabra.append(matriz_letra_encriptada)
    return matriz_palabra

def desencriptar_palabra_multiplicacion(matrices_palabra_encriptada):
    """
    Función que desencripta una palabra completa utilizando la función␣
↪'desencriptar_letra_multiplicacion'.
    """
    palabra_desencriptada = ""
    for matriz_letra_encriptada in matrices_palabra_encriptada:
        letra_desencriptada =␣
↪desencriptar_letra_multiplicacion(matriz_letra_encriptada)
        palabra_desencriptada += letra_desencriptada[-1] # Tomamos solo la␣
↪última letra desencriptada
    return palabra_desencriptada

def desencriptar_palabra_suma(matrices_palabra_encriptada):
    """
    Función que desencripta una palabra completa utilizando la función␣
↪'desencriptar_letra_suma'.
    """
    palabra_desencriptada = ""
    for matriz_letra_encriptada in matrices_palabra_encriptada:
        letra_desencriptada = desencriptar_letra_suma(matriz_letra_encriptada)
        palabra_desencriptada += letra_desencriptada[-1] # Tomamos solo la␣
↪última letra desencriptada
    return palabra_desencriptada

def elegir_metodo_encriptacion():
    """
    Función que solicita al usuario que elija el método de encriptación deseado.
    """
    while True:
        metodo = input("¿Qué método de encriptación desea utilizar?␣
↪(multiplicacion/suma): ").lower()
        if metodo == 'multiplicacion':
            return 'multiplicacion'
        elif metodo == 'suma':
```

```
            return 'suma'
        else:
            print("Por favor, ingrese 'multiplicacion' o 'suma'.")

# Ejemplo de uso
palabra_a_encriptar = input("Escribe la palabra a encriptar: ")
metodo_encriptacion = elegir_metodo_encriptacion()
if metodo_encriptacion == 'multiplicacion':
    matrices_palabra_encriptada =␣
 ↪encriptar_palabra_multiplicacion(palabra_a_encriptar)
elif metodo_encriptacion == 'suma':
    matrices_palabra_encriptada = encriptar_palabra_suma(palabra_a_encriptar)

print("Matrices de la palabra encriptada:")
for matriz in matrices_palabra_encriptada:
    print("\n", matriz, "\n")

# Desencriptamos la palabra
if metodo_encriptacion == 'multiplicacion':
    palabra_desencriptada =␣
 ↪desencriptar_palabra_multiplicacion(matrices_palabra_encriptada)
elif metodo_encriptacion == 'suma':
    palabra_desencriptada =␣
 ↪desencriptar_palabra_suma(matrices_palabra_encriptada)

print("Palabra desencriptada:", palabra_desencriptada)
```

```
Ingrese los valores de la matriz base (3 filas x 3 columnas):

Ingrese los valores de la fila 1, separados por espacio:  8 1 6
Ingrese los valores de la fila 2, separados por espacio:  3 5 7
Ingrese los valores de la fila 3, separados por espacio:  4 9 2
Escribe la palabra a encriptar:  amor
¿Qué método de encriptación desea utilizar? (multiplicacion/suma):  suma

Matrices de la palabra encriptada:

 [[ 9  2  7]
 [ 4  6  8]
 [ 5 10  3]]


 [[21 14 19]
 [16 18 20]
 [17 22 15]]


 [[24 17 22]
```

```
 [19 21 23]
 [20 25 18]]


 [[27 20 25]
 [22 24 26]
 [23 28 21]]

Palabra desencriptada: AMOR
```

3 Encriptador y Desencriptador de Palabras

Este programa permite al usuario encriptar y desencriptar palabras utilizando dos métodos diferentes: multiplicación de matrices y suma de matrices.

3.1 Funcionamiento del Programa

El programa solicita al usuario que elija el método de encriptación deseado: multiplicación o suma. Luego, el usuario ingresa la palabra que desea encriptar. El programa encripta la palabra letra por letra y muestra las matrices resultantes. Finalmente, desencripta la palabra utilizando el mismo método elegido y muestra el resultado.

3.2 Perfección del Programa

3.2.1 Flexibilidad en el Método de Encriptación

- El programa permite al usuario elegir entre dos métodos de encriptación: multiplicación y suma de matrices. Esto brinda flexibilidad y adaptabilidad a las preferencias del usuario.

3.2.2 Claridad en el Proceso

- El programa guía al usuario a través de cada paso, desde la elección del método de encriptación hasta la entrada de la palabra a encriptar. Esto garantiza una experiencia de usuario clara y sin confusiones.

3.2.3 Funcionalidad Completa

- El programa realiza tanto la encriptación como la desencriptación de palabras de manera efectiva utilizando los métodos elegidos. Esto asegura que el usuario pueda cifrar y descifrar sus mensajes con facilidad y precisión.

3.3 Conclusión

En resumen, el programa proporciona una solución completa y flexible para la encriptación y desencriptación de palabras. Su capacidad para adaptarse a diferentes métodos de encriptación y su claridad en el proceso lo convierten en una herramienta útil y eficaz para proteger la privacidad de las comunicaciones. AEON MERX INICIO DE AOEN MERX VER MÀS CURSOS

[ ]: