UNIVERSIDAD CENTRAL DEL ECUADOR

Facultad de ingeniería y ciencias aplicadas

Carrera: Computación

Asignatura: Criptografía y Seguridad de la Información.

Tarea 01: Documentación.

Estudiantes:

1. Andrade Cardenas Kevin Anthony
2. Chamba Carrion Jordy Pedro
3. Luna Grijalva Joel Joshua
4. Parra Vásquez Rensso Nicolay
5. Pozo Maldonado Kevin Fernando
6. Verkade Montiel Emil Thaddaeus

Anagramas:

Las permutaciones de caracteres constituyeron la base de los primeros sistemas criptográficos históricos, particularmente en cifrados por transposición como la escítala espartana y los métodos de columnas usados durante la Guerra Civil Americana [1]. Estas técnicas reorganizaban los caracteres según patrones predefinidos, demostrando que el simple reordenamiento - aunque carente de reversibilidad intrínseca - podía ofuscar mensajes cuando se combinaba con claves de permutación específicas [2]. El concepto evolucionó hacia sistemas híbridos como el cifrado de Vigenère con tablas permutadas, donde la transposición complementaba la sustitución alfabética.

El algoritmo de permutaciones implementado representa una operación fundamental no reversible en criptografía, que sirve como base conceptual para sistemas de cifrado más complejos como el César y otros cifrados por sustitución [3]. A diferencia de los cifrados reversibles, este método de reordenamiento puro no contiene mecanismos inherentes para descifrar la información.

El mapeo entre la palabra original y sus permutaciones no es biyectivo cuando hay caracteres repetidos.

Ejemplo:

Entrada: "aab" → Permutaciones: ["aab", "aba", "baa"]

No existe forma algorítmica determinar cuál fue la permutación original sin información adicional.

Cifrado por filas

El cifrado por permutación de filas es un método de transposición clásico que reorganiza el orden de las filas en una matriz según una clave determinista. Este documento describe la implementación técnica realizada en Python para el ejercicio de criptografía, junto con sus fundamentos teóricos.

2. Metodología Implementada

2.1. Algoritmo Base
Se implementó una variante del *Transposition Cipher* [4], donde:

- **Entrada**:
    - Texto plano (eliminando espacios).
    - Clave n (número de filas de la matriz cuadrada).
- **Proceso**:
    1. **Construcción de la matriz**: El mensaje se divide en una matriz de n × n, rellenando con '*' si es necesario.
    2. **Permutación de filas**:
        - Si n es **par**: Inversión del orden de filas.
        - Si n es **impar**: Rotación descendente de una posición.
    3. **Generación del cifrado**: Lectura de la matriz permutada **por columnas**.

2.2. Validaciones

- El mensaje debe cumplir: longitud ≤ n^2.
- Se garantiza que la matriz siempre sea cuadrada (n × n).

Método de permutación por columnas

El cifrado por permutación de columnas es un tipo de cifrado de transposición que reordena los caracteres del texto plano según una clave numérica que define el número de columnas (n) y su orden de lectura, El texto plano se escribe en una matriz de n columnas, la clave determina el orden en que se leen las columnas, el texto cifrado se obtiene concatenando las columnas en el orden especificado.

Ejemplo [5]:

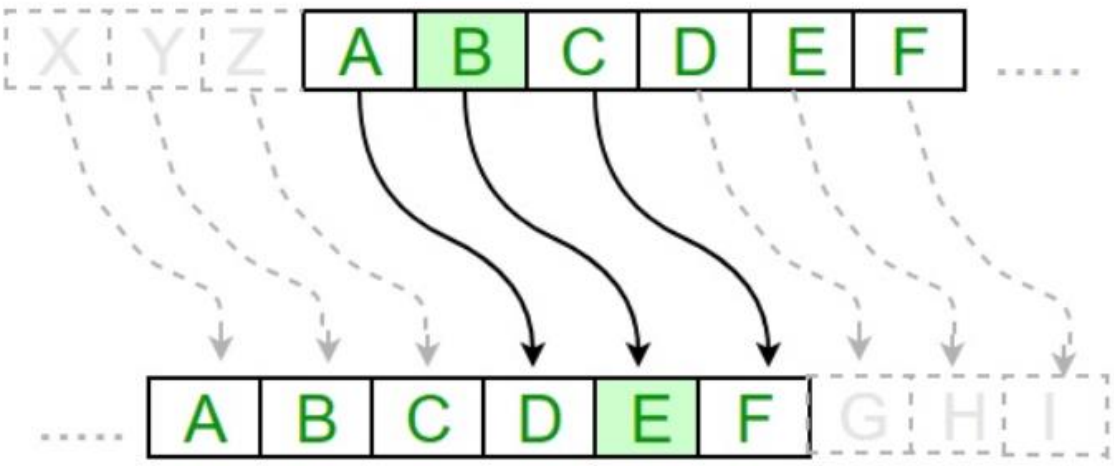
```
Key:          4 3 1 2 5 6 7
Plaintext:    a t t a c k p
              o s t p o n e
              d u n t i l t
              w o a m x y z
Ciphertext:   TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Este cifrado es vulnerable a ataques de frecuencia si n es pequeño, la fortaleza depende de la longitud de la clave y del secreto del orden de permutación.

Puede llegar a ser mejor que otros métodos de permutación por la facilidad que tiene para crear claves complejas.

Método de sustitución Mono alfabético de desplazamiento n caracteres a la derecha

Este algoritmo implementa el **cifrado de César**, un caso particular de cifrado mono alfabético de sustitución en el que cada letra del texto plano se reemplaza por otra situada un número fijo *n* de posiciones más a la derecha en el alfabeto [6].



1. **Crear alfabetos:**

- n %= 26 garantiza que *n* quede en el rango [0, 25].
- Para crear el alfabeto cifrado se desplaza a la derecha *n* posiciones: la porción final pasa al principio y el resto se “corre” hacia adelante.

```
1 alfabeto = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
[13]  1 def crear_alfabetos(desplazamiento):
      2     # Devuelve una tupla (alfabeto_original, alfabeto_cifrado) desplazado n posiciones.
      3     desplazamiento %= len(alfabeto)  # Asegura que n esté en [0, 25]
      4     alfabeto_cifrado = alfabeto[-desplazamiento:] + alfabeto[:-desplazamiento]
      5     return alfabeto, alfabeto_cifrado
```

2. **Cifrar textos:**

- El texto original se itera con un bucle.
- Se convierte en mayúscula para buscar índice en el alfabeto original.
- Si la letra existe en el alfabeto, se obtiene su posición y se toma la letra correspondiente del alfabeto cifrado.
- Si el carácter original era una minúscula, se convierte la letra cifrada a minúscula; si no, se deja en mayúscula.
- Si un carácter no está en el alfabeto (espacios, números y signos de puntuación) se añaden sin cambios.

```
[15]  1 def cifrar_texto(texto, alfabeto, alfabeto_cifrado):
      2     # Cifra el texto usando los alfabetos dado el mapeo de sustitución.
      3     resultado = []
      4     for ch in texto:
      5         ch_upper = ch.upper()
      6         if ch_upper in alfabeto:
      7             i = alfabeto.index(ch_upper)
      8             # Conserva mayúsculas/minúsculas
      9             nuevo = alfabeto_cifrado[i]
     10             resultado.append(nuevo if ch.isupper() else nuevo.lower())
     11         else:
     12             resultado.append(ch)
     13     return ''.join(resultado)
```

3. **Se solicitan inputs del usuario (texto y desplazamiento)**

[14]

```python
def main():
    # Entrada
    texto = input("Ingresa la cadena a cifrar: ")
    try:
        n = int(input("Ingresa el valor de desplazamiento n (entero): "))
    except ValueError:
        print("Desplazamiento inválido. Usa un número entero.")
        return

    # Generar alfabetos
    alfabeto_orig, alfabeto_cif = crear_alfabetos(n)

    # Cifrar
    texto_cifrado = cifrar_texto(texto, alfabeto_orig, alfabeto_cif)

    # Salida
    print(f"\nAlfabeto original: {alfabeto_orig}")
    print(f"Alfabeto cifrado : {alfabeto_cif}")
    print(f"Texto ingresado  : {texto}")
    print(f"Texto cifrado    : {texto_cifrado}")

if __name__ == "__main__":
    main()
```

```
Ingresa la cadena a cifrar: manzana
Ingresa el valor de desplazamiento n (entero): 3

Alfabeto original: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Alfabeto cifrado : XYZABCDEFGHIJKLMNOPQRSTUVW
Texto ingresado  : manzana
Texto cifrado    : jxkwxkx
```

Método de sustitución Polialfabético de Vigenère.

El cifrado Vigenère es un cifrado basado en diferentes series de caracteres o letras del cifrado César formando estos caracteres una tabla, llamada tabla de Vigenère, que se usa como clave. El cifrado de Vigenère es un cifrado polialfabético y de sustitución.

El cifrado Vigenère se ha reinventado muchas veces. El método original fue descrito por Giovan Batista Belaso en su libro de 1553 "*La cifra del Sig. Giovan Batista Belaso"*, quien construyó el cifrado basándose en la tabula recta de Trithemius, pero añadió una clave repetida para cambiar cada carácter entre los diferentes alfabetos. Sin embargo, fue incorrectamente atribuido en el siglo XIX a Blaise de Vigenère, a partir de un trabajo realizado en 1583, y por ello aún se le conoce como el *"cifrado Vigenère"* [7].

Este cifrado es conocido porque es fácil de entender e implementar, además parece irresoluble; esto le hizo valedor del apodo el *código indescifrable* (le chiffre indéchiffrable, en francés).

El primer cifrado polialfabético fue creado por Leone Battista Alberti hacia 1467 y usaba un disco de metal para cambiar entre los diferentes alfabetos del cifrado. El sistema de Alberti sólo cambiaba entre alfabetos después de muchas palabras, y los cambios se indicaban escribiendo la letra del correspondiente alfabeto en el mensaje cifrado. Más tarde, en 1508, Johannes Trithemius, en su trabajo Poligraphia, inventó

la *tabula recta*, que es básicamente la tabla de Vigenère. Trithemius, sin embargo, sólo proporcionó un progresivo, rígido y predecible sistema de cambio entre alfabetos [8].

¿Cómo usar Vigenère?

Paso 1: Seleccione una palabra clave

Seleccione una palabra clave. En nuestro caso 'Hola'.

Paso 2: Crea una clave

Escribe tu mensaje y sigue repitiendo la palabra HOLA hasta que sustituyas todas las letras del mensaje. La Figura 1 muestra cómo puede crear una clave sustituyendo cada letra de la palabra HOLA con su letra correspondiente en el mensaje.

Mensaje	P	A	G	I	N	A
Clave	H	O	L	A	H	O

Figura 1: Sustituyendo la Palabra Clave para crear una Clave

Paso 3: Usa el cuadrado de Vigenère

Una vez que tenga su sustitución mencionada en la Figura 1 lista, el siguiente paso es usar Plaza vigenère para cifrar su mensaje. La figura 2 muestra el cuadrado de Vigenère.

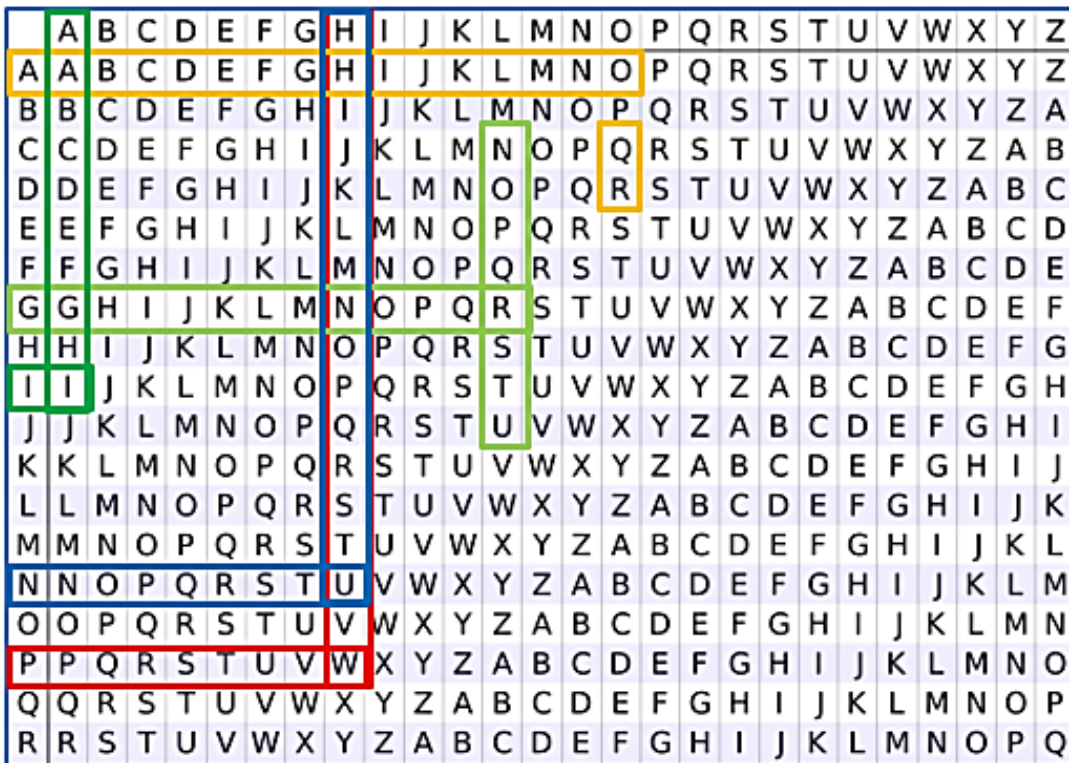


Figura 2: Cuadrado de Vigenère

Paso 3: Cifrado

En la columna más a la izquierda, encuentre la letra 'P' (la primera letra del mensaje). Luego, en la fila superior, encuentre la letra 'H' (la primera letra de la tecla). Luego,

encuentre la letra que está presente en la intersección de la fila P y la columna H. La carta en la intersección 'W' es la primera letra del mensaje cifrado. La segunda letra del mensaje en la fila A y la segunda letra de la clave en la columna O se cruzan en la letra 'O'. Del mismo modo, puede cifrar todas las letras del mensaje. La figura 3 muestra el texto cifrado después de seguir el proceso de cifrado para todas las letras del mensaje.

Mensaje	P	A	G	I	N	A
Clave	H	O	L	A	H	O
Cifrado	W	O	R	I	O	U

Figura 3: Texto cifrado después del cifrado

Vigenère Cipher usando Python

Los algoritmos de cifrado se utilizan ampliamente en la industria del software. Algunas de las aplicaciones de mensajería o correo electrónico cifran sus mensajes. Esto garantiza que nadie más pueda averiguar el contenido de sus mensajes sin conocer la palabra clave. Las compañías de software pueden usar un tipo similar de cifrado. El siguiente código en Python muestra el proceso de cifrado y descifrado utilizando el cifrado Vigenère.

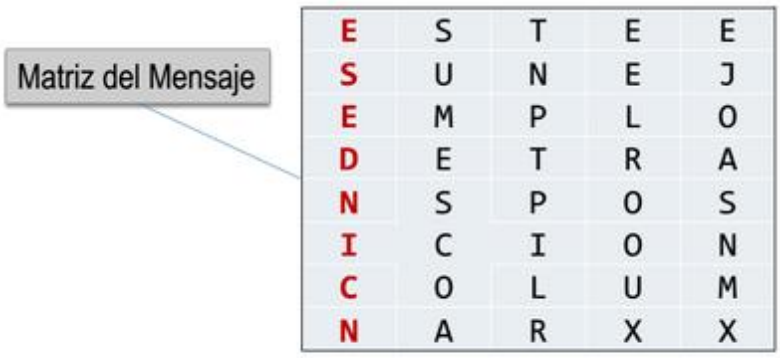
El código mencionado anteriormente toma la entrada del usuario como el mensaje de texto sin formato y la palabra clave. A continuación, el mensaje se cifra y se descifra utilizando el cifrado Vigenère. Este código da la siguiente salida [9].

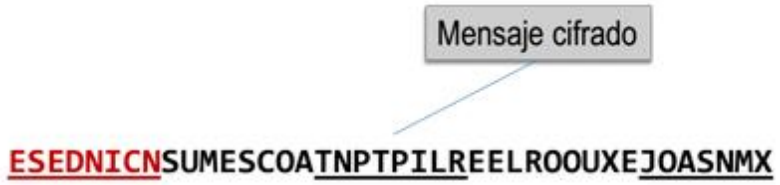
Conclusión

El cifrado Vigenère es ligeramente mejor que el cifrado César. Puede usar el código python para cifrar y descifrar los mensajes.

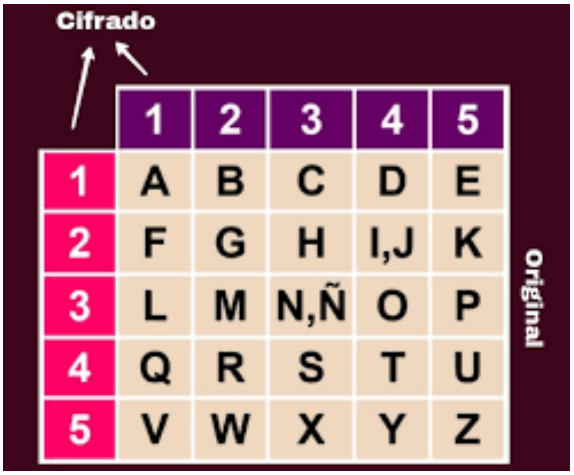
Cifrado de Sustitución Monoalfabética (Basado en coordenadas de una matriz de sustitución)

El algoritmo implementado pertenece a la familia de cifrados por sustitución monoalfabética, un tipo de cifrado clásico muy utilizado en la historia de la criptografía. En particular, a una variante matricial del cifrado de coordenadas, muy parecido al cifrado de Polibio, también conocido como el cuadrado de Polibio. Este tipo de cifrado fue creado por el griego Polibio en el siglo II a.C. Este cifrado utiliza una tabla que generalmente es simétrica de 5x5 la cual asigna a cada letra una coordenada compuesta por una fila y columna añadidas adicionalmente para ocupar dicho enlazamiento de palabras. En su versión clásica, se utilizan números para identificar filas y columnas, en este algoritmo se reemplaza dichos números por letras provenientes del alfabeto extranjero modificado volviéndolo una variante más moderna y personalizable [10].





Este tipo de cifrado es útil para la introducción de conceptos de criptografía básicos, tales como: la unidad de las sustituciones, el uso de tablas de correspondencia, el manejo de mensajes cifrados en entornos sin algoritmos modernos. No obstante, este tipo de cifrado es muy fácil de implementar y comprender, lo cual lo hace ideal para fines educativos y para introducirse en conceptos más complejos de la misma criptografía y seguridad de la información, dada su simplicidad también lo hace mayormente débil, ya que es altamente vulnerable ante ataques de análisis de frecuencia y no ofrece un nivel de seguridad adecuados para proteger la información sensible en entornos reales.



Bibliografía

[1] D. Kahn, The Codebreakers, Scribner, 1996, pp. 82-85.

[2] F. L. Bauer, Decrypted Secrets, Springer, 2006, pp. 45-47.

[3] A. J. M. e. al., Handbook of Applied Cryptography, CRC Press, 1996, pp. 15-18.

[4] GeeksforGeeks, «GeeksforGeeks,» Junio 2024. [En línea]. Available: https://www.geeksforgeeks.org/transposition-cipher-techniques-in-cryptography/. [Último acceso: 20 Abril 2025].

[5] W. Stallings, Cryptography and Network Security: Principles and Practice, 7th ed., Pearson, 2017, pp. 38-42.

[6] «TutorialsPoint,» [En línea]. Available: https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_caesar_cipher.htm. [Último acceso: 17 abril 2025].

[7] S. Gómez, J. Arias y D. Agudelo, «Cripto-Análisis Sobre Métodos Clásicos de Cifrado | Scientia et Technica,» *Revistas UTP,* 2012.

[8] F. José, «ALGORITMO DE CIFRADO,» [En línea]. Available: https://repository.udistrital.edu.co/server/api/core/bitstreams/c8b5607e-b9be-4761-81b6-c97d869e6d39/content. [Último acceso: 16 abril 2025].

[9] elhacker.INFO, «Criptografía, MAPLE y RSA,» [En línea]. Available: https://elhacker.info/manuales/Criptografia/CRYPTO.PDF. [Último acceso: 16 abril 2025].

[10] J. L. Carbajo, «Polibio Criptografía Clásica,» [En línea]. Available: https://joseluistabaracarbajo.gitbooks.io/criptografia-clasica/content/Cripto04.html. [Último acceso: 17 abril 2025].