

Российский университет дружбы народов Научный факультет

Математические
основы защиты
информации и
информационной
безопасности

шифрование маршрута
шифр Виженера
сетевое шифрование



Подготовлено студентом:
Елиенис Санчес Родригес.
Преподаватель: Дмитрий Сергеевич

Шифрование или транспонирование маршрута

Это тип шифрования, в котором элементы открытого текста перемещаются по четко определенной схеме; «текстовые единицы» могут быть отдельными буквами (наиболее распространенный случай), парами букв, тройками букв

Шифрование или транспонирование маршрута

```
from pyfiglet import figlet_format
print(figlet_format("Cifrado de Ruta by Elienis",
font = "cybermedium"))
import math
# Función principal
def main():
    message = input('Introducir Mensaje: ')
    key = int(input('Introducir Key [2-%s]: ' %
(len(message) - 1)))
    mode = input('Cifrar/Decifrar [c/d]: ')

    if mode.lower().startswith('c'): # Si mode es
igual a "c" se llamara a "encryptMessage"
        text = cifrarMensaje(key, message)
    elif mode.lower().startswith('d'): # De lo
contrario se llamara a "decryptMessage"
        text = descifrarMensaje(key, message)

    # Imprime la cadena cifrada y usando '\n' para
indicar el fin
    #del mensaje cifrado
    print("Salida:\n%s" %(text + '\n'))
    print("\n")
    input("")

def cifrarMensaje(key, message):
#Cada cadena de texto en el mensaje cifrado
representa una columna en la matriz
    cipherText = [''] * key
    # recorremos las colimnas
    for col in range(key):
        pointer = col
```

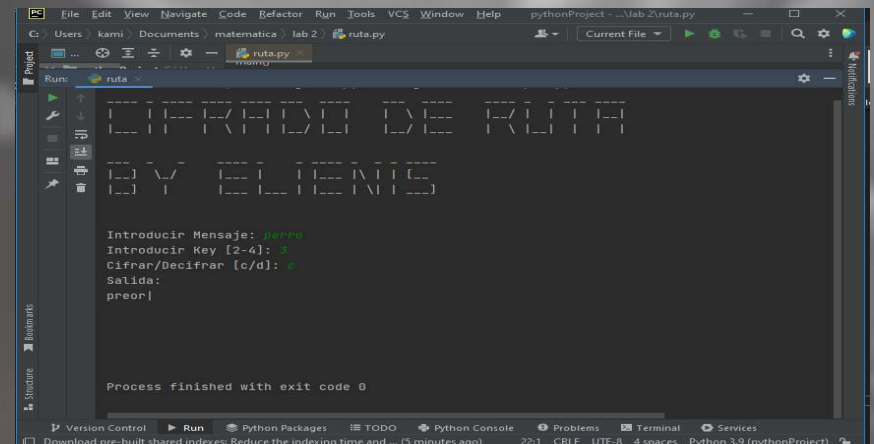
```
#Lee cada elemento de la columna
    while pointer < len(message):
        cipherText[col] += message[pointer]

    #mueve el puntero al siguiente elemento de la
columna
    pointer += key
    return "".join(cipherText)

def descifrarMensaje(key, message):
    numCols = math.ceil(len(message) / key)
    numRows = key
    numShadedBoxes = (numCols * numRows) -
len(message)
    plainText = [''] * numCols
    col = 0; row = 0;

    for symbol in message:
        plainText[col] += symbol
        col += 1

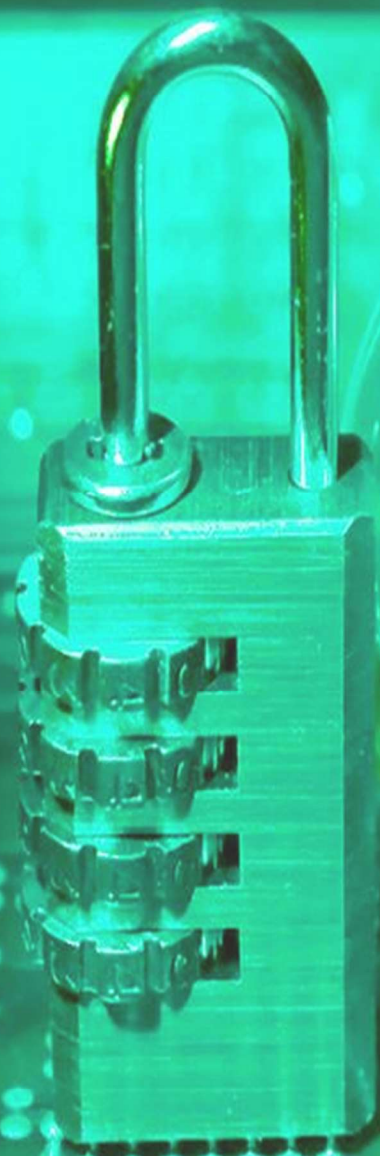
        if (col == numCols) or (col == numCols - 1)
and (row >= numRows - numShadedBoxes):
            col = 0
            row += 1
    # Convertimos la lista en una cadena de texto
y lo retornamos
    return "".join(plainText)
# si el nombre de la funcion es main la manda
a ejecutar
if __name__ == '__main__':
    main()
```



```
pythonProject - ...\lab 2\ruta.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Users \kami\Documents\matematica\lab 2\ruta.py
Run Current File
Project
Run ruta.py
Structure
Run
Introducir Mensaje: perro
Introducir Key [2-4]: 3
Cifrar/Decifrar [c/d]: c
Salida:
preor

Process finished with exit code 0
Version Control Run Python Packages TODO Python Console Problems Terminal Services
Download pre-built shared indexes: Reduce the indexing time and ... (5 minutes ago) 22:1 CRLF UTF-8 4 spaces Python 3.9 (pythonProject)
```


шифр Виженера



Шифр Виженера представляет собой шифрование, основанное на различных сериях символов или букв шифра Цезаря, эти символы образуют таблицу, называемую таблицей Виженера, которая используется в качестве ключа. Шифр Виженера — полиалфавитный шифр с подстановкой.

Шифр Виженера неоднократно изобретался заново. Оригинальный метод был описан Джован Батиста Белазо в его книге 1553 года

шифр Виженера

```
from colorama import init, Fore
from pyfiglet import figlet_format
print(figlet_format( "Cifrado Vigenere by Elienis",
font = "cybermedium"))
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

```
LETRAS = ("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
```

```
def main():
```

```
    mensaje=input("Introduce el Mensaje: ")
    myKey="MINOMBREESANTONIOALFONSO"
    print ("quiere encriptar o descifrar")
    accion=input("Modo: ")
```

```
    if accion=='encriptar':
```

```
        traducido=cifrar_mensaje(myKey,mensaje)
```

```
    elif accion=='descifrar':
```

```
        traducido=descifrar_mensaje(myKey,mensaje)
    print(traducido)
```

```
def cifrar_mensaje(clave,mensa):
```

```
    return traductor_mensaje(clave,mensa,'encriptar')
```

```
def descifrar_mensaje(clave,mensa):
```

```
    return traductor_mensaje(clave,mensa,'descifrar')
```

```
def traductor_mensaje(clave,mensa,accion):
    traducido=[]
    indice_clave=0
    clave=clave.upper()
```

```
    for symbol in mensa:
```

```
        num=LETRAS.find(symbol.upper())
```

```
        if num!=-1:
```

```
            if accion=='encriptar':
```

```
                num+=LETRAS.find(clave[indice_clave])
```

```
            elif accion=='descifrar':
```

```
                num-=LETRAS.find(clave[indice_clave])
```

```
            num%=len(LETRAS)
```

```
            if symbol.isupper():
```

```
                traducido.append(LETRAS[num])
```

```
            elif symbol.islower():
```

```
                traducido.append(LETRAS[num].lower())
```

```
            indice_clave+=1
```

```
            if indice_clave==len(clave):
```

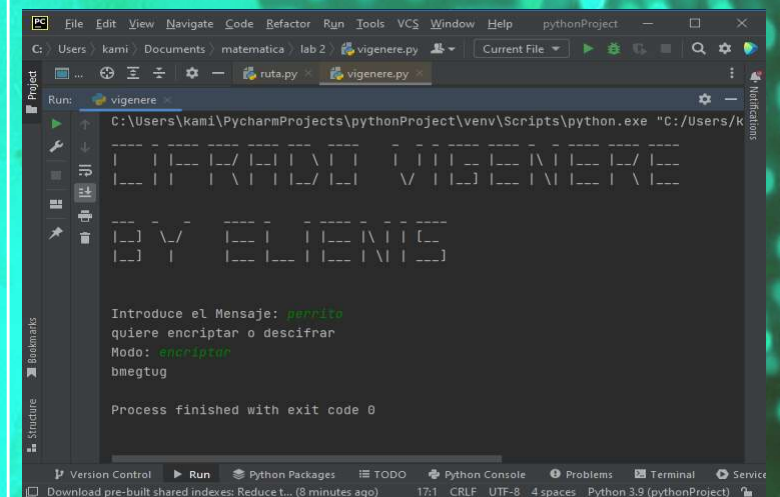
```
                indice_clave=0
```

```
        else:
```

```
            traducido.append(symbol)
```

```
    return ''.join(traducido)
```

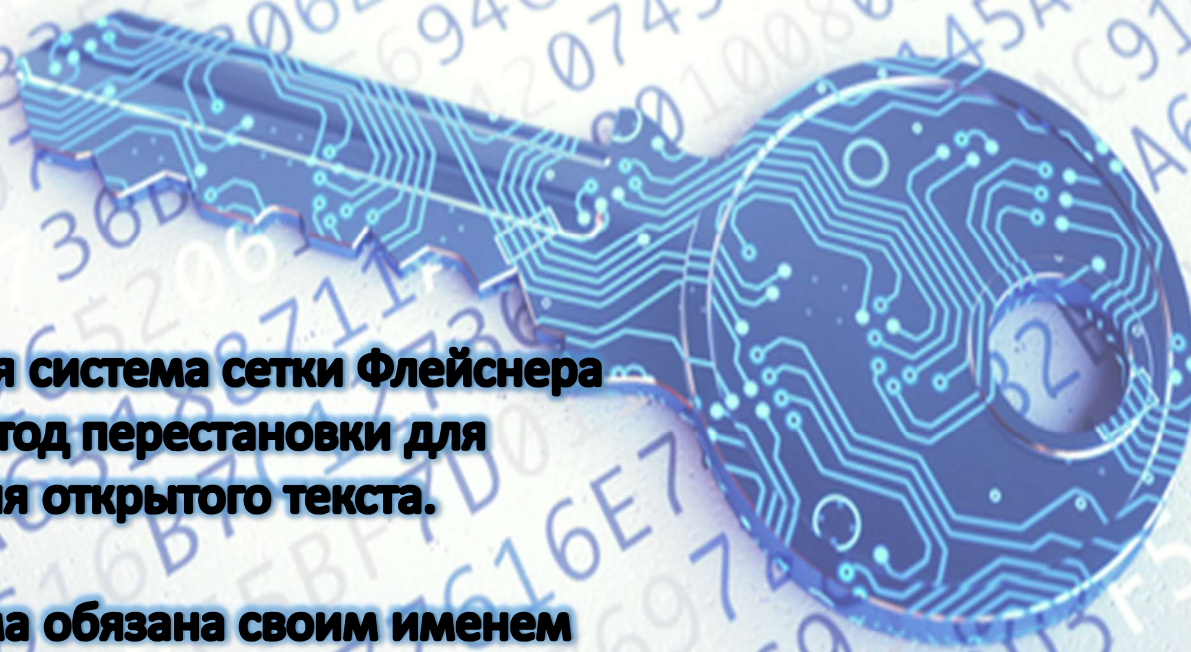
```
if __name__ == '__main__':
    main()
```



Сетки Флейснера

Криптографическая система сетки Флейснера реализует метод перестановки для шифрования открытого текста.

Эта криптосистема обязана своим именем своему изобретателю Эдуарду Фляйснеру фон Востровицу и также известна как система с вращающейся сеткой.



Сетки Флейснера

```
from typing import List, Optional
import numpy as np
from colorama import init, Fore, Back, Style
print("\033[9;35m"+"Codificación Rejilla by Elienis")
```

```
def grille_encrypt(plaintext: str, grille: List[str]) -> Optional[str]:
    # Hay un trozo de papel de cuadrícula cuadrada con
    # orificios (4*4) en el papel blanco, rejilla: posición del orificio
    # Después de girar la posición del orificio durante una
    # semana, se puede llenar el cuadrado de 4*4. Si no puede,
    # no será válido y volverá directamente a None.
    # Escriba las letras de texto sin formato en los agujeros de
    # izquierda a derecha y de arriba a abajo. Después de girar el
    # papel de orificio 90 grados en el sentido de las agujas del
    # reloj, el orificio se mueve a una posición en blanco y las
    # letras de texto sin formato continúan escribiéndose.
    # Si el mensaje no termina después de girar 3 veces,
    # continúe en la siguiente hoja de papel
    # La letra final se completará, lea el texto cifrado de
    # salida por línea
```

```
# Regrese a la posición cubierta por el orificio después de
# que la rejilla gire 3 veces
def check_grille(grille):
    # Ubicación actual del hoyo
    K = [(i, j) for i, j in enumerate(grille) for j, v in
    enumerate(l) if v == 'X']
    # Convertir lista de rejillas a matriz
    # Rotación de matriz, obtenga la posición del orificio
    # después de cada rotación, y después de 3 rotaciones,
    # obtenga la posición del orificio.
```

```
for i in range(3):
    grille = [[for j in g ] for g in grille ]
    A = np.mat(grille) # Lista matriz de par
    B = np.rot90(A,k=-1) # Rotación de la
    matriz, k es un número positivo en sentido
    antihorario, k es un número negativo en sentido
    horario.
    grille = B.tolist() # Matriz para listar
    k = [(i, j) for i, j in enumerate(grille) for j, v
    in enumerate(l) if v == 'X']
    K.extend(k)
    return K
    K = check_grille(grille) # La posición en la que
    los orificios se cubren a su vez es también la
    posición en la que las letras de texto sin
    formato se rellenan a su vez
    #
    imprimir('=====',c
    onjunto(K), len(conjunto(K)))
```

```
if len(set(K)) != 16 or len(K) > 16: # Después de
    girar 3 veces, el orificio no cubre 4*4=16
    cuadrados, o los cuadrados se cubren
    repetidamente, la rejilla no es válida
    result = None
    else:
        ciphertext = []
        for i in range(0,len(plaintext),16):
            sub_t = plaintext[i:i+16]
            # Narr = np.array([['.', '.', '.', '.'], ['.', '.', '.',
            '.', '.'], ['.', '.', '.', '.'], ['.', '.', '.', '.']])
```

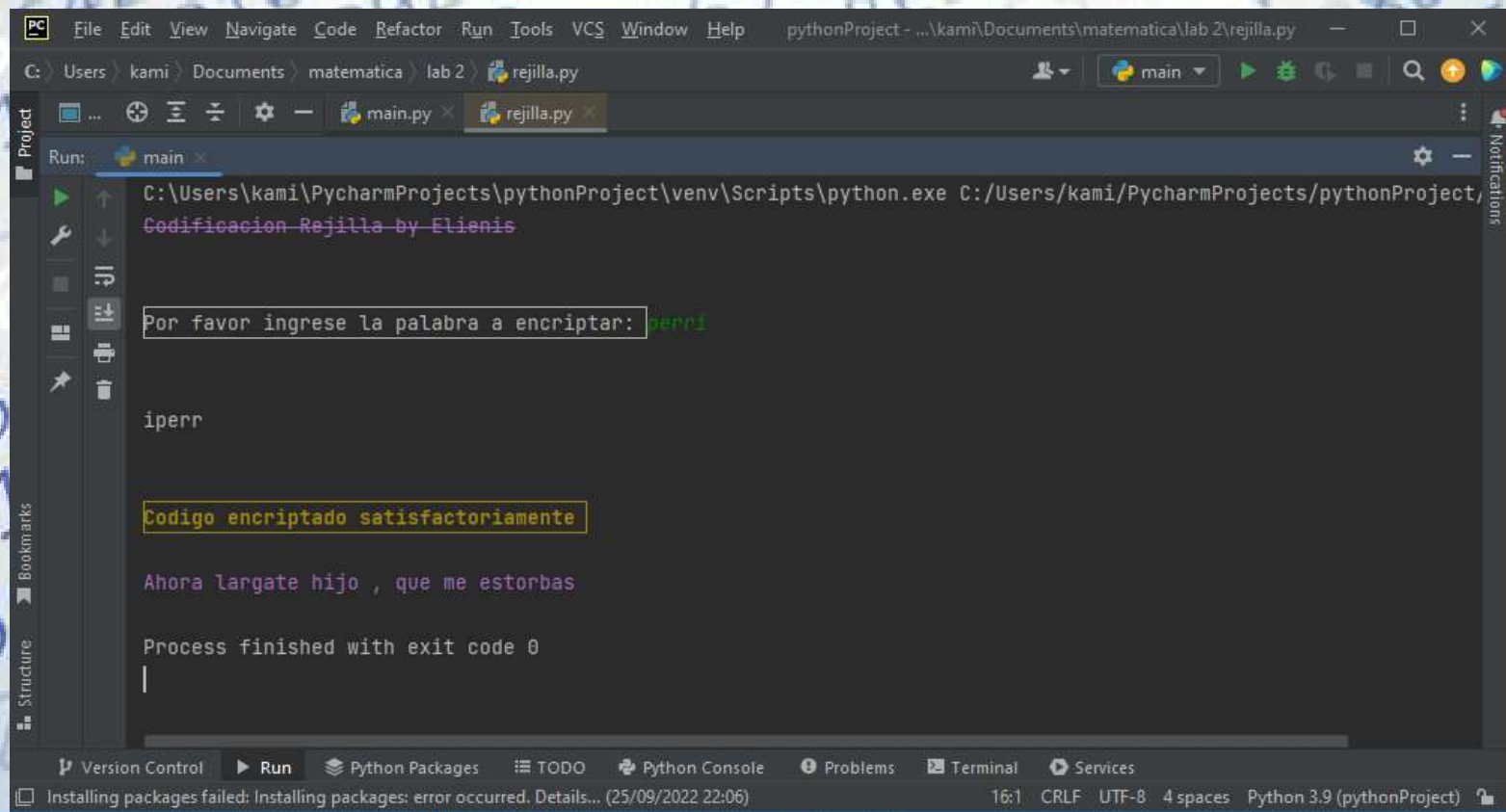
```
Narr = np.ones((4, 4)).astype(np.str_) #
# Crear una matriz de caracteres de 4*4
# Complete las letras de texto sin
# formato en la posición de K a su vez
for j in range(16):
    x = K[j][0]
    y = K[j][1]
    Narr[x][y] = sub_t[j] if j <
    len(sub_t) else " # Evita que la longitud
    del texto sin formato sea inferior a 16
    # Después de completar,
    # convierta la matriz en una cadena por
    # fila
    sub_c = Narr.tolist() # Añadir a la
    lista
    sub_c = ".join(['.join(i) for i in
    sub_c])
    ciphertext.append(sub_c)
    result = ".join(ciphertext)
    # print(result)
    return result
```

```
if __name__ == "__main__":
    print("\n")
    numero1 = input("\033[0;51m"+"Por
    favor ingrese la palabra a encriptar: ")
    print("\n")
    print(grille_encrypt(numero1, [".X..",
    ".X..", "...X", "X..."]))
    print("\n")
    # Estas "afirmaciones" se utilizan para
    # la autocorprobación y no para una
    # prueba automática
    assert (
        grille_encrypt("cardangrilletest",
        [".X..", ".X..", "...X", "X..."])
```

```
== "actilangeslrdret"
)
assert (
    grille_encrypt(
        "quickbrownfoxjumpsoverthelazydog",
        [".X..", "...X", ".X.", ".X.."]
    )
    ==
    "qwxkbnjufriumcoopyeerldsatoogvzh"
)
assert (
    grille_encrypt(
        "quickbrownfoxjumpsoverthelazydog",
        [".XX.", ".XX.", ".X.", "X..."]
    )
    == None
)
assert
grille_encrypt("cardangrilletest",
[ "...X", "...", "...", "..."]) == None

print("\033[1;33m" + "Codigo
encriptado satisfactoriamente \n" +
'\033[0;m')
print("\033[2;35m" + "Ahora largate
hijo , que me estorbas")
```

Сетки Флейснера



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject - ...kami\Documents\matematica\lab 2\rejilla.py
C:\Users\kami\Documents\matematica\lab 2\rejilla.py
main.py x rejilla.py x
Run: main x
C:\Users\kami\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/kami/PycharmProjects/pythonProject/
Codificacion Rejilla by Elienis
Por favor ingrese la palabra a encriptar: perri
iperr
Codigo encriptado satisfactoriamente
Ahora largate hijo , que me estorbas
Process finished with exit code 0
Version Control Run Python Packages TODO Python Console Problems Terminal Services
Installing packages failed: Installing packages: error occurred. Details... (25/09/2022 22:06) 16:1 CRLF UTF-8 4 spaces Python 3.9 (pythonProject)
```


Microsoft Windows [Versión 10.0.19042.1706]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\kami>

C:\Users\kami>

ВЫВОД

Криптография — это не технология, это искусство и техника сокрытия сообщений. Средства и каналы коммуникации по своей природе небезопасны, что компрометирует процесс коммуникации.

Шифрование было определено как часть криптологии, которая имеет дело с методами, применимыми к искусству или науке, которые изменяют сообщения с помощью методов шифрования или скремблирования, чтобы сделать их понятными для злоумышленников, которые перехватывают эти сообщения. Поэтому единственной целью криптографии было достижение конфиденциальности сообщений. Для этого были разработаны системы шифрования и коды. В то время единственной существовавшей криптографией была так называемая классическая криптография.

Microsoft Windows [Versión 10.0.19042.1706]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\kami>

C:\Users\kami>

Библиография

Perez P, B., & Acosta Velarde, R. (2019). Mejora en la seguridad python . Ciencia Digital, 2(3), 61-74. <https://inventwithpython.com/cracking/chapter7.html>

Cabrera R, Juan , python-transposition (2017). método transposición. Ciencia Digital, 8(5), <https://blog.finxter.com/python-transposition-algorithm/>

S Paulina . (2017). Programacion de 0 aplicada. <https://github.com/indetectables-net/manuals/blob/master/Programacion/Python/Algoritmos%20y%20Programaci%C3%B3n%201/algoritmos-programacion-1.pdf>

Barahona, B., & Yopez Velarde, R. (2016). Rejillas Digitales en seguridad y cryptografia Ciencia Digital, <https://books.google.es/books?hl=es&lr=&id=9jUjYLjJZCAC&oi=fnd&pg=PA13&dq=cifrado+en+rejillas+giratorias&ots=1MU1LQ7z32&sig=Ym3oBBN2VhP-eCrfbULjmdYzrbg#v=onepage&q=cifrado%20en%20rejillas%20giratorias&f=false>

Cley R . Naranjo, Velarde,Q. (2016). Secretos informáticos . Ciencia Digital, 62- <https://studylib.es/doc/9203034/la-biblia-de-los-c%C3%B3digos-secretos---herv%C3%A9-lehning>

Microsoft Windows [Versión 10.0.19042.1706]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\kami>

C:\Users\kami>Большое спасибо