

Лабораторная работа 2

Терентьев Егор Дмитриевич, НФИмд-01-23

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ПРЕЗЕНТАЦИЯ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

дисциплина: Математические основы защиты информации
и информационной безопасности

Преподаватель: Кулябов Дмитрий Сергеевич

Студент: Терентьев Егор Дмитриевич

Группа: НФИмд-01-23

МОСКВА

2023 г.

Прагматика выполнения лабораторной работы

Прагматика выполнения лабораторной работы

Требуется реализовать:

1. Маршрутное шифрование.
2. Шифрование с помощью решеток.
3. Табоица Виженера

Цель работы

Цель работы

Освоить на практике шифры перестановки.

Выполнение лабораторной работы

1. Для реализации маршрутного шифрования:

1. Для реализации маршрутного шифрования:

1. Функции проверки правильности пароля, значения k
2. Функция берущая столбцы матрицы в виде ключа буквы пароля в алфавитном порядке (был использован словарь для удобства)

```
import math
import re

def is_pass_valid(password, columns):
    if len(password) != columns:
        print("pass len should be equal to row len")
        raise ValueError
    alphabeticValues = re.findall(r'[a-zA-Zа-яА-Я]', password)
    if len(alphabeticValues) != len(password):
        print("pass should contain only eng and ru letters")
        raise ValueError

def is_len_columns_valid(columns, inputString):
    if columns > len(inputString):
        print("len columns should be <= len of Input String")
        raise ValueError
    if columns <= 1:
        raise ValueError

def to_dict(inputString, password, columns, row):
    list_of_slices = []
    for i in range(row):
        list_of_slices.append(inputString[columns*i:columns*(i+1)])

    while len(list_of_slices[-1]) != columns:
        list_of_slices[-1] += 'a'
```

2. Основная функция запуска где получаем входные значения и шифруем слово

2. Основная функция запуска где получаем входные значения и шифруем слово

```
inputedString = input("Input string to encrypt: ")
columns = int(input("Input int value to determine the number columns: "))
password = input("Password: ").lower()

inputString = inputedString.replace(" ", "") # remove spaces
password = password.replace(" ", "") # remove spaces
password = ''.join([password[i] for i in range(len(password)-1) if password[i+1] != password[i]+password[-1]]) #remove duplicates
is_len_columns_valid(columns, inputString)
is_pass_valid(password, columns)
row = math.ceil(len(inputString) / int(columns))

routeDict = to_dict(inputString, password, columns, row)

cryptogram = ""
for key in routeDict:
    cryptogram += routeDict[key]

print("начальная фраза: ", inputedString)
print("криптограмма: ", cryptogram)
```

Figure 2: route_main_func

3. Запуск программы маршрутного шифрования

3. Запуск программы маршрутного шифрования

```
Input string to encrypt: нельзя недооценивать противника
Input int value to determine the number columns: 6
Password: пароль
начальная фраза:  нельзя недооценивать противника
криптограмма:  еенпнзоатаьовокннеьвлдирияцтиа
PS F:\учеба 5 курс\информационная безопасность> █
```

Figure 3: route_output

4. Для реализации шифрования с помощью решеток:

4. Для реализации шифрования с помощью решеток:

1. Функция генерирующая сетку (матрицу) (использована библиотека numpy Для удобства) и ее заполнение
2. Функция заполняющая сетку значениями букв из текста и переворачивающая матрицу
3. Функция выбирающая столбцы в алфавитном порядке пароля
4. Функция объединяющая все вышепоказанные функции и проверки правильности введенных данных

```
1 import numpy as np
2
3 def generate_grid(k):
4     grid = np.zeros((k, k))
5     for i in range(k):
6         for j in range(k):
7             grid[i][j] = i * k + j + 1
8     return grid
9
10 def encrypt_with_grid(grid, text):
11     k = len(grid)
12     encrypted = np.chararray((2*k, 2*k), unicode=True)
13     encrypted[:] = ''
14
15     idx = 0
16     for _ in range(4):
17         for i in range(k):
18             for j in range(k):
19                 if grid[i][j] and idx < len(text):
20                     encrypted[i][j] = text[idx]
21                     idx += 1
22
23     grid = np.rot90(grid)
24
```

5. Основная функция запуска функции шифрования с помощью решеток

5. Основная функция запуска функции шифрования с помощью решеток

```
def encrypt(text, password):
    k = int(len(text)**0.5)
    if k*k != len(text):
        raise ValueError("Length of the text should be a perfect square.")

    if len(password) != k:
        raise ValueError(f"Length of the password should be {k}.")

    grid = generate_grid(k)
    encrypted_grid = encrypt_with_grid(grid, text)
    result = extract_by_password(encrypted_grid, password)

    return result

# Пример использования
text = "нужно подписать новый указ"
password = "шифр"
print(encrypt(text, password))
```

Figure 5: grid_main_func

6. Запуск программы атбаш

6. Запуск программы атбаш

```
44  
45  # Пример использования  
46  text = "нужно подписать новый указ да"  
47  password = "шифрр"  
48  print(encrypt(text, password))  
49
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

PS F:\учеба 5 курс\информационная безопасность> & C:

○ уоаванпъдоинуаждтызнпсок

PS F:\учеба 5 курс\информационная безопасность> |

Figure 6: grid_output

7. Для реализации Таблицы Виженера:

7. Для реализации Таблицы Виженера:

1. Функция шифрования (построение таблицы Вижинера)
2. Функция дешифровки

```
1 def vigenere_encrypt(text, key):
2     alph = 'абвгдезийклмнопрстуфхцшщъыьэя'
3     encrypted_text = ''
4
5     #размерность пароля - размерности текста
6     key_expanded = ''
7     while len(key_expanded) < len(text):
8         key_expanded += key
9     key_expanded = key_expanded[:len(text)]
10
11     #сравниваем по таблице и получаем на пересечении букву
12     for t, k in zip(text, key_expanded):
13         if t in alph:
14             new_pos = (alph.index(t) + alph.index(k)) % len(alph)
15             encrypted_text += alph[new_pos]
16         else:
17             encrypted_text += t
18
19     return encrypted_text
20
21 def vigenere_decrypt(encrypted_text, key):
22     alph = 'абвгдезийклмнопрстуфхцшщъыьэя'
23     decrypted_text = ''
24
25     key_expanded = ''
26     while len(key_expanded) < len(encrypted_text):
27         key_expanded += key
28     key_expanded = key_expanded[:len(encrypted_text)]
29
30     for e, k in zip(encrypted_text, key_expanded):
31         if e in alph:
32             new_pos = (alph.index(e) - alph.index(k)) % len(alph)
33             decrypted_text += alph[new_pos]
34         else:
35             decrypted_text += e
36
37     return decrypted_text
38
```

Figure 7: vigenere_funcs

8. Основная функция запуска функции Таблицы Виженера

8. Основная функция запуска функции Таблицы Виженера

```
# Пример использования
text = "криптографиясерьезнаянаука"
password = "математика"
encrypted_text = vigenere_encrypt(text, password)
print(f"Encrypted: {encrypted_text}")
print(f"Decrypted: {vigenere_decrypt(encrypted_text, password)}")
|
```

Figure 8: vigenere_main_func

9. Запуск программы Таблицы Виженера

9. Запуск программы Таблицы Виженера

```
38  
39  # Пример использования  
40  text = "криптографиясерьезнаянаука"  
41  password = "математика"  
42  encrypted_text = vigenere_encrypt(text, password)  
43  print(f"Encrypted: {encrypted_text}")  
44  print(f"Decrypted: {vigenere_decrypt(encrypted_text, password)}")  
45  |
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```
PS F:\учеба 5 курс\информационная безопасность> & C:/Python311/python.exe "f:/учеба  
Encrypted: црѣѣюхшкѣѣгкѣѣчпчалнтшца  
Decrypted: криптографиясерьезнаянаука  
PS F:\учеба 5 курс\информационная безопасность> |
```

Figure 9: vigenere_output

Выводы

Выводы

В результате выполнения работы я освоил на практике применение шифров перестановки.

