

Лабораторная работа 2

Серенко Данил Сергеевич, 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: Серенко Данил Сергеевич

Группа: НФИмд-01-23

МОСКВА

2023 г.

Прагматика выполнения лабораторной работы

Требуется реализовать:

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

Цель работы

Освоить на практике шифры перестановки.

Выполнение лабораторной работы

1. Для реализации маршрутного шифрования:

1. Функции заполнения матрицы текстом
2. Функция, шифрующая матрицу

```
##### ROUTE #####
text_2 = "нельзя недооценивать противника".replace(" ", "")
password_2 = "пароль"

def fill_matrix(text, num_cols):
    num_rows = math.ceil(int(len(text)) / int(num_cols))
    matrix = []
    for i in range(num_rows):
        matrix.append([letter for letter in text_2[num_cols * i:num_cols * (i + 1)]])

    while len(matrix[-1]) != num_cols:
        matrix[-1].append("a")

    return matrix

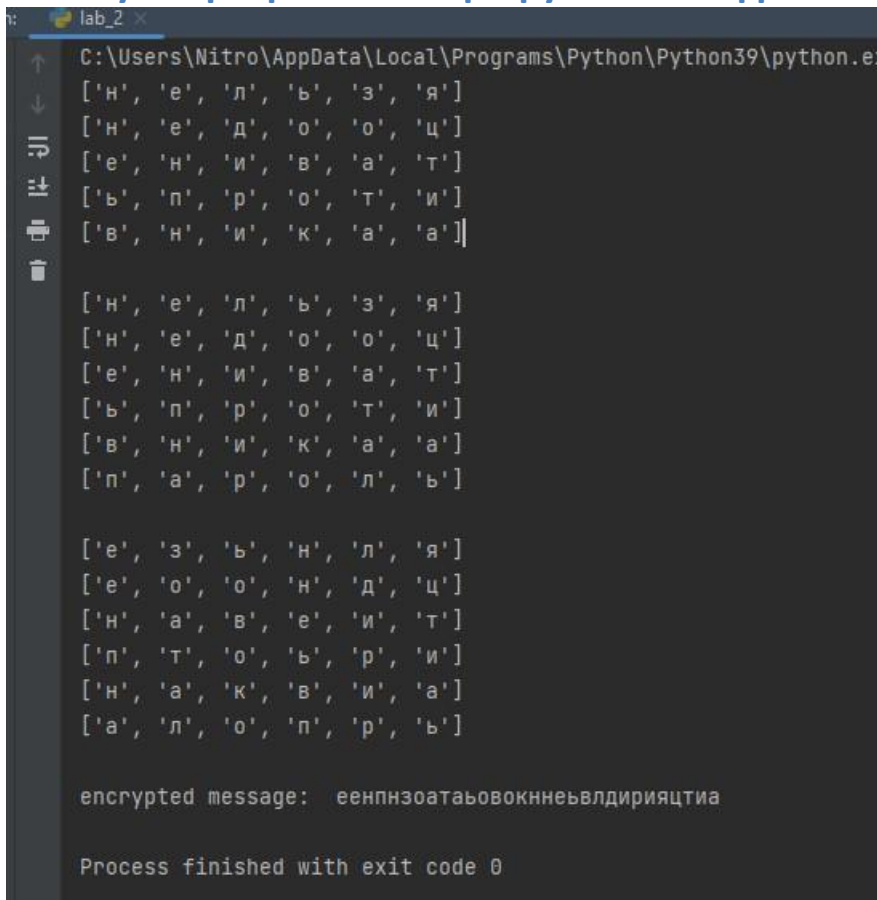
def encrypt_route(text, num_cols):
    matrix = fill_matrix(text, num_cols)
    print_matrix(matrix)
    matrix.append([letter for letter in password_2])
    print_matrix(matrix)
    sorted_matrix(matrix, len(matrix), len(matrix[0]))
    print_matrix(matrix)
    shifr = ""

    for i in range(len(matrix)):
        for j in range(len(matrix) - 1):
            shifr += matrix[j][i]

    print("encrypted message: ", shifr)
```

route_funcs

2. Запуск программы маршрутного шифрования



```
lab_2 x
C:\Users\Nitro\AppData\Local\Programs\Python\Python39\python.exe
['н', 'е', 'л', 'ь', 'з', 'я']
['н', 'е', 'д', 'о', 'о', 'ц']
['е', 'н', 'и', 'в', 'а', 'т']
['ь', 'п', 'р', 'о', 'т', 'и']
['в', 'н', 'и', 'к', 'а', 'а']

['н', 'е', 'л', 'ь', 'з', 'я']
['н', 'е', 'д', 'о', 'о', 'ц']
['е', 'н', 'и', 'в', 'а', 'т']
['ь', 'п', 'р', 'о', 'т', 'и']
['в', 'н', 'и', 'к', 'а', 'а']
['п', 'а', 'р', 'о', 'л', 'ь']

['е', 'з', 'ь', 'н', 'л', 'я']
['е', 'о', 'о', 'н', 'д', 'ц']
['н', 'а', 'в', 'е', 'и', 'т']
['п', 'т', 'о', 'ь', 'р', 'и']
['н', 'а', 'к', 'в', 'и', 'а']
['а', 'л', 'о', 'п', 'р', 'ь']

encrypted message: еенпнзоатаьовокннеьвлдирияцтиа

Process finished with exit code 0
```

route_output

3. Для реализации шифрования с помощью решеток:

Чтобы реализовать программу был написан след. код на python:

```

7      ##### GRID #####
8
9  def print_matrix(matrix):
10      for row in matrix:
11          print(row)
12      print()
13
14
15  def sorted_matrix(a, N, M):
16      k = M-1
17      while k > 0:
18          ind = 0
19          for j in range(k+1):
20              if a[N-1][j] > a[N-1][ind]:
21                  ind = j
22          for i in range(N):
23              m = a[i][ind]
24              a[i][ind] = a[i][k]
25              a[i][k] = m
26          k -= 1
27
28      return a
29
30
31  def encrypt_with_grid(k):
32      encrypted = np.zeros((2*k, 2*k))
33
34      for _ in range(4):
35          for i in range(k):
36              for j in range(k):
37                  encrypted[i][j] = i * k + j + 1

```

grid_funcs_1

```

34     for _ in range(4):
35         for i in range(k):
36             for j in range(k):
37                 encrypted[i][j] = i * k + j + 1
38
39         encrypted = np.rot90(encrypted)
40         print(encrypted)
41         print()
42     uniq = [num for num in range(1, k**2+1)]
43     indexes = []
44     while len(uniq) > 0:
45         for i, lst in enumerate(encrypted):
46             if len(uniq) == 0:
47                 break
48             index = np.where(lst == uniq[0])
49             index = index[0]
50             if len(index) == 1:
51                 indexes.append((i, index[0]))
52                 uniq.pop(0)
53             elif len(index) > 1:
54                 prob = randint(0, 1)
55                 if prob == 1:
56                     indexes.append((i, index[-1]))
57                 else:
58                     indexes.append((i, index[0]))
59                 uniq.pop(0)
60     print("indexes: ", indexes)
61     print()
62     encrypted_matrix = np.chararray((k * 2, k * 2), unicode=True)
63     ind_text = 0
64     for _ in range(k*2):

```

grid_funcs_2

```

print()
encrypted_matrix = np.chararray((k * 2, k * 2), unicode=True)
ind_text = 0
for _ in range(k*2):
    for ind in indexes:
        encrypted_matrix[ind[0]][ind[1]] = text[ind_text]
        ind_text += 1
    encrypted_matrix = np.rot90(encrypted_matrix)
    print(encrypted_matrix)
    print()

order_passw = []
for letter in password:
    order_passw.append(rus_alp.index(letter))
matrix = [np.ndarray.tolist(row) for row in encrypted_matrix]
matrix.append(order_passw)
print_matrix(matrix)
sorted_matrix(matrix, len(matrix), len(matrix[0]))
print_matrix(matrix)
shifr = ""
for i in range(len(matrix) - 1):
    for j in range(len(matrix) - 1):
        shifr += matrix[j][i]

print("encrypted message: ", shifr)

```

grid_funcs_3

```

print("encrypted message: ", shifr)

def encrypt(text, password):
    k = int(len(text)**0.25)
    if k**4 != len(text):
        raise ValueError("Length of the text should be a perfect square.")

    if len(password) != k**2:
        raise ValueError(f"Length of the password should be {k}.")

    encrypt_with_grid(2)

text = "договор подписали".replace(" ", "")
password = "шифр"
encrypt(text, password)

```

grid_funcs_4

4. Запуск программы

```
lab_2
C:\Users\Nitro\AppData\Local\Programs\Python\Python39\python.exe C:\Users\N
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [2. 4. 0. 0.]
 [1. 3. 0. 0.]]

[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [2. 4. 4. 3.]
 [1. 3. 2. 1.]]

[[0. 0. 3. 1.]
 [0. 0. 4. 2.]
 [2. 4. 4. 3.]
 [1. 3. 2. 1.]]

[[1. 2. 3. 1.]
 [3. 4. 4. 2.]
 [2. 4. 4. 3.]
 [1. 3. 2. 1.]]

indexes: [(0, 0), (1, 3), (2, 3), (1, 1)]

[[' 'o' 'r' '']]
[[' ' ' ' '']]
[[' 'o' ' ' '']]
[['д' ' ' ' '']]

[[' 'o' 'p' '']]
[['r' ' ' ' '']]
[['o' 'n' 'o' '']]
[['в' ' ' ' 'д']]
```

grid_output

```

[[' ' 'о' 'р' ' ']]
[['г' ' ' ' ' ' ']]
[['о' 'п' 'о' ' ']]
[['в' ' ' ' ' 'д']]

[[' ' 'д' 'п' 'д']]
[['р' ' ' 'о' ' ']]
[['о' 'и' 'п' ' ']]
[['о' 'г' 'о' 'в']]

[['д' 'а' 'л' 'в']]
[['п' 'о' 'п' 'о']]
[['д' 'и' 'и' 'г']]
[['с' 'р' 'о' 'о']]

[['д', 'а', 'л', 'в']]
[['п', 'о', 'п', 'о']]
[['д', 'и', 'и', 'г']]
[['с', 'р', 'о', 'о']]
[25, 9, 21, 17]

[['а', 'в', 'л', 'д']]
[['о', 'о', 'п', 'п']]
[['и', 'г', 'и', 'д']]
[['р', 'о', 'о', 'с']]
[9, 17, 21, 25]

encrypted message: аoirвоголпиодпдс

Process finished with exit code 0
|

```

grid_output_2

6. Для реализации Таблицы Виженера:

1. Функция шифрования (построение таблицы Виженера)


```

alph = 'абвгдежзийклмнопрстуфхцшщъыэюя'
def vigenere_encrypt(text, key):
    encrypted_text = ''

    # размерность пароля = размерности текста
    key_expanded = ''
    while len(key_expanded) < len(text):
        key_expanded += key
    key_expanded = key_expanded[:len(text)]

    alph_matrix = create_alf(alph)

    # сравниваем по таблице и получаем на пересечении букву
    for t, k in zip(text, key_expanded):
        encrypted_text += alph_matrix[alph.index(k)][alph.index(t)]

    return encrypted_text

def create_alf(alph):
    alph_matrix = []
    for i in range(len(alph)):
        new_row = alph[i:] + alph[:i]
        alph_matrix.append([letter for letter in new_row])

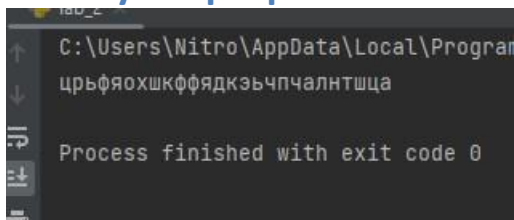
    return alph_matrix

text_3 = "криптография серьезная наука".replace(" ", "")
password_3 = "математика"
encrypted_text = vigenere_encrypt(text_3, password_3)
print(encrypted_text)

```

vigenere_funcs

7. Запуск программы Таблицы Виженера



```

C:\Users\Nitro\AppData\Local\Programs
црьфяохшкффядкэьчпчалнтщца
Process finished with exit code 0

```

vigenere_output

Выводы

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