Математические основы защиты информации и информационной безопасности. Лабораторная работа № 2 на тему “Шифры перестановки”

Лубышева Ярослава Михайловна

RUDN University, Moscow, Russian Federation

Содержание

Содержание

Цели и задачи

Цели и задачи

Выполнить задание к лабораторной работе № 2 [1]:

1) Изучить шифры перестановки: маршрутное шифрование, шифрование с помощью решеток, шифр по талице Виженера
2) Реализовать программно каждый шифр

Выполнение

Выполнение

```python
from itertools import islice
import numpy as np
```

```python
# алфавит
alphabet = ['а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п',
            'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я']
```

```python
# функция для транспонирования матрицы
def transpose(l1):
  l2 = []
  for i in range(len(l1[0])):
    row =[]
    for item in l1:
      row.append(item[i])
    l2.append(row)
  return l2
```

Figure 1: Первая часть программной реализации шифров

Выполнение

```
# Маршрутное шифрование
# n - длина блоков
# m - количетво блоков
# password - пароль
# mes - сообщение для шифрования
def route_encryption(password, mes):
  mes = list(mes.lower())
  password = list(password.lower())
  n = len(password)
  m = len(mes)//n if len(mes)%n == 0 else len(mes)//n+1
  print("Пароль: ", password, "m: ", m, "n: ", n)

  # массив шифрования
  # преоразовываем сообщение в матрицу размера n*m для далнешего шифрования
  it = iter(mes)
  encrypt = [list(islice(it, i)) for i in [n]*m]

  # припишем к последнему блоку буквы из алфавита,
  # если он получился меньше остальных
  encrypt[-1] += alphabet[0:n-len(encrypt[-1])]

  # транспонируем матрицу-сообщение для упрощения вывода криптограммы
  encrypt_2 = transpose(encrypt)

  # согласно алфавитному порядку пароля выпишем криптограмму
  print("Криптограмма: ", end='')
  for let in alphabet:
    if let in password:
      # выписываем строку из encrypt_2 по индексу буквы в пароле
      print(''.join(encrypt_2[password.index(let)]), end='')
```

Выполнение

```
password = "пароль" #input("пароль ")
mes = "НельзяНедооцениватьПротивника" #input("сообщение ")
route_encryption(password, mes)
```

```
Пароль:  ['п', 'а', 'р', 'о', 'л', 'ь'] m:  5 n:  6
Криптограмма: еенпнзоатаьовокннеьвлдирияцтиа
```

Figure 3: Результат работы программы для маршрутного шифрования

Выполнение

```python
# шифрование с помощью решеток
def grid_encryption(password, mes, grid):
  mes = mes.lower()
  password = list(password.lower())
  # если в сообщении недостаточно символов для шифрования, допишем буквы в конец сообщения
  mes += "".join(alphabet[0:len(grid)**2-len(mes)])
  # пустой масссив шифрования
  data = [[0, 0, 0, 0],
          [0, 0, 0, 0],
          [0, 0, 0, 0],
          [0, 0, 0, 0]]
  # счетчик для количества букв в сообщении
  let_i = 0
  # заполняем массив шифрования согласно сетке, затем поворачиваем ее по часовой стрелке
  for _ in range(4):
    # проходя по каждоу элементу сетки и для каждого значения "1" ставим букву сообщения в соответствующую клетку массива data
    for i in range(len(grid)):
      for j in range(len(grid[i])):
        if grid[i][j] == 1:
          data[i][j] = mes[let_i]
          let_i += 1
    # отрицателный знак второго аргумента - поворот по часовой стрелке
    grid = np.rot90(grid, -1)
  print("Итоговая таблица криптограмма:\n", data)
  # транспонируем матрицу-криптограмму для упрощения вывода криптограммы
  encrypt_2 = transpose(data)
  # согласно алфавитному порядку пароля выпишем криптограмму
  print("Криптограмма: ", end='')
  for let in alphabet:
    if let in password:
      # выписываем строку из encrypt_2 по индексу буквы в пароле
      print(''.join(encrypt_2[password.index(let)]), end='')
```

Figure 4: Программная реализация шифрования с помощью решеток

Выполнение

```
password = "шифр"
mes = "ДоговорПодписали"

grid = [[0, 0, 0, 1],
        [0, 0, 0, 0],
        [0, 1, 0, 1],
        [0, 0, 1, 0]]

grid_encryption(password, mes, grid)
```

```
Итоговая таблица криптограмма:
 [['с', 'о', 'а', 'д'], ['д', 'в', 'п', 'л'], ['о', 'о', 'и', 'г'], ['и', 'р', 'о', 'п']]
Криптограмма: оворддлгпапиосдои
```

Figure 5: Результат работы программы для шифрования с помощью решеток

Выполнение

```python
def vishener_encryption(password, mes):
  mes = list(mes.lower())
  password = list(password.lower())

  # сделаем, чтобы пароль password дублировался на всю длину сообщения mes
  password *= len(mes)//len(password) + 1
  password = password[:len(mes)]

  print(password)
  print(mes)
  print()

  cryptogram = ""

  # находим букву пароля в вертикальном алфавите,
  # а букву сообщения в горизонтальном - на пересечении буква криптограммы
  for let_password, let_mes in zip(password, mes):
    # если обе буквы находятся в алфавите
    if let_password in alphabet and let_mes in alphabet:
      # находим, какой индекс буква на пересечении имеет в алфавите
      num = ( alphabet.index(let_password)+1 + alphabet.index(let_mes)+1 ) % (len(alphabet)) - 1
      cryptogram += ''.join(alphabet[num])

  print("Криптограмма: ", cryptogram)
```

Figure 6: Программная реализация шифра по талице Виженера

Выполнение

```
password = "математика"
mes = "КриптографияСерьёзнаяНаука"
vishener_encryption(password, mes)
```

```
['м', 'а', 'т', 'е', 'м', 'а', 'т', 'и', 'к', 'а', 'м', 'а', 'т', 'е', 'м', 'а', 'т', 'и', 'к', 'а', 'м', 'а', 'т', 'е', 'м', 'а']
['к', 'р', 'и', 'п', 'т', 'о', 'г', 'р', 'а', 'ф', 'и', 'я', 'с', 'е', 'р', 'ь', 'ё', 'з', 'н', 'а', 'я', 'н', 'а', 'у', 'к', 'а']

Криптограмма:  шсьхапцълхцаекюэщсщбмоущшб
```

Figure 7: Результат работы программы для шифра по талице Виженера

Результаты

Результаты

Выполнено задание к лабораторной работе № 2

Список литературы

Список литературы

1. Методические материалы курса