

Информационная безопасность. Отчет по лабораторной работе № 2

Шифры перестановки

Мухамеджанов Исматулло Иззатуллоевич

Содержание

1	Цель работы	5
2	Указание к работе	6
3	Выводы	10
4	Список литературы	11

List of Figures

2.1	Программа (1)	8
2.2	Программа (2)	8
2.3	Программа (3)	9
2.4	Программа (4)	9

List of Tables

1 Цель работы

Освоить на практике применение шифрование перестановкой [1].

2 Указание к работе

Исходные данные. Маршрутное шифрование Шифрование с помощью решёток Таблица Виженера # Выполнение лабораторной работы 1. Маршрутное шифрование.

Данный способ шифрования разработал французский математик Франсуа Виет. Открытый текст записывают в некоторую геометрическую фигуру (обычно прямоугольник) по некоторому пути, а затем, выписывая символы по другому пути, получают шифртекст. Пусть m и n — целые положительные числа, большие Открытый текст разбивается на блоки равной длины, состоящие из n символов, равному произведению $m \cdot n$. Если последний блок получится меньше остальных, то в него следует дописать требуемое количество произвольных символов. Составляется таблица размерности $m \cdot n$. Блоки вписываются построчно в таблицу. Криптограмма получается выписыванием букв из таблицы в соответствии с некоторым маршрутом. Ключом такой криптограммы является маршрут и числа m и n . Обычно буквы выписывают по столбцам, которые упорядочивают согласно паролю: внизу таблицы приписывается слово из n неповторяющихся букв и столбцы нумеруются по алфавитному порядку букв пароля.

Рассмотренный способ шифрования (столбцовая перестановка) в годы первой мировой войны использовала легендарная немецкая шпионка Мата Хари.

2. Шифрование с помощью решеток.

Данный способ шифрования предложил австрийский криптограф Эдуард Флейснер в 1891 году. Суть этого способа заключается в следующем. Выбирается

натуральное число $k > 1$, строится квадрат размерности k и построено заполняется числами $1, 2, \dots, k^2$. В качестве примера рассмотрим квадрат размерности $k = 2$.

Повернем его по часовой стрелке на 90° и присоединим к исходному квадрату справа. Прделаем еще дважды такую процедуру и припишем получившиеся квадраты снизу. Получился большой квадрат размерности $2k$.

Далее из большого квадрата вырезаются клетки, содержащие числа от 1 до k^2 . В каждой клетке должно быть только одно число. Получается своего FОділ решето. Шифрование осуществляется следующим образом. Решето накладывается на чистый квадрат $2k \times 2k$ и в прорези вписываются буквы

исходного текста по порядку их следования. Когда заполнятся все прорези, решето поворачивается на 90° и вписывание букв продолжается. После третьего поворота все клетки большого квадрата окажутся заполненными. Подобрал подходящий пароль (число букв пароля должно равняться k^2 и они не должны повторяться), выпишем буквы по столбцам. Очередность столбцов определяет алфавитным порядком букв пароля.

3. Шифрование с помощью решеток. В 1585 году французский криптограф Блез Виженер опубликовал свой метод шифрования в «Трактате о шифрах». Шифр считался нераскрываемым до 1863 года, когда австриец Фридрих Казиски взломал его. Открытый текст разбивается на блоки длины p . Ключ представляет собой Следовательность из p натуральных чисел: m, p_2, \dots . Дал е в каждом блоке первая буква циклически сдвигается вправо по алфавиту на m позиций, вторая буква — на p_2 позиций, последняя — на p_t позиций. Для лучшего запоминания в качестве ключа можно взять осмысленное слово, а алфавитные номера входящих в него букв использовать для осуществления сдвигов.

The screenshot shows a Jupyter Notebook window titled "Lab_2_IPynb". The code defines a function `path_cipher` that takes a text string, a key word, and two integers `n` and `m`. It implements a cipher where characters are grouped into a matrix of size `m` by `n`. The matrix is filled with characters from the text in a row-major order, and then the key word is used to permute the columns. The final result is the concatenation of the rows of the permuted matrix.

```
import string
import random
#Маршрутное шифрование

def path_cipher(text:str, key_word:str, n: int , m:int):
    """Cipher the text according to path cipher method"""
    alphabet = "абвгдежзийклмнопстуфхцчшщъыьэюя"
    text = text.replace(" ", "")
    matrix = []
    counter = 0
    result = ""
    for i in range(len(text) // (m)):
        matrix.append(text[i*n:i*n+m])

    while len(matrix[-1]) != n:
        # matrix[-1] += random.choice(string.ascii_lowercase)
        matrix[-1] += random.choice(alphabet)
        # print(matrix[-1][-1])
    # Adding keyword to the matrix
    # matrix.append(key_word)
    for char in alphabet:
        if char in key_word:
            position = key_word.index(char)
            for j in matrix:
                result += j[position]
    print(result)
```

Figure 2.1: Программа (1)

The screenshot shows a Jupyter Notebook window titled "Шифрование с помощью решёток". The code implements a lattice cipher. It takes a text input and generates a key. The key is a list of random integers. The text is converted into a list of symbols (characters). The symbols are then encrypted using the key. The final result is the encrypted text.

```
from random import choice, randint
from collections import Counter

text = input("Write the message: ")
symbols = [chr(x) for x in range(65,91)] # A - Z
symbols += [chr(x) for x in range(97,123)] # a - z
# A - z
while True:
    keys = [randint(1,100) for x in range(len(text))]
    k = Counter(keys)
    switch = 0; n = 0
    for l in k:
        if k[keys[n]] > 1:
            switch = 1
            break
        n += 1
    if switch == 0:
        break
    keys.sort()
    print("keys:",keys)

lattice = [choice(symbols) for x in range(0,101)]
```

Figure 2.2: Программа (2)


```

lattice = [choice(symbols) for x in range(0,101)]

n = 0
for i in range(len(lattice)):
    if n < len(text):
        if i == keys[n]:
            lattice[i] = text[n]
            n += 1
print()

for j in range(0,11):
    if j == 0:
        print(" ", end = " ")
    else:
        print(j, end = " ")
print()

n = 0
for j in range(1,len(lattice)):
    if j == 1:
        print(n, end = " | "); n += 1
        print(lattice[j], end = " | ")
    elif j % 10 != 0:
        print(lattice[j], end = " | ")
    else:
        print(lattice[j], end = " | ")
        print()
        if n < 10:
            print(n, end = " | "); n += 1
print()

```

Figure 2.3: Программа (3)

Шифр Виженера

```

def encrypt_vegener(plaintext, key):
    key_length = len(key)
    key_as_int = [ord(i) for i in key]
    plaintext_int = [ord(i) for i in plaintext]
    result = ''
    for i in range(len(plaintext_int)):
        value = (plaintext_int[i] + key_as_int[i % key_length]) % 26
        result += chr(value + 65)
    return result

encrypt_vegener("cryptography crucial science", "math")

```

[65]

... 'ADDIRALKYBMRLONNAUFELEHBCZHX'

Figure 2.4: Программа (4)

3 Выводы

Освоены шифры методом перестановки

4 Список литературы

1. Методические материалы курса