Лабораторная работа №2

Шифры перестановки

Якушевич Артём Юрьевич

Содержание

Список таблиц

Список иллюстраций

1 Цель работы

Ознакомиться с шифрами перестановки и обучиться их программной реализации.

2 Задание

- Реализовать шифрование столбцовой перестановки;
- Реализовать таблицу виженера.

3 Теоретическое введение

При подготовке использовалась методичка со страницы курса в ТУИС.[1]

Шифрование столбцовой перестановки. Криптограмма получается выписыванием букв из таблицы в соответствии с некоторым маршрутом. Ключом такой криптограммы является марштур и числа m и n. Внизу таблицы приписывается слово из n неповторющихся букв и столбцы нумеруются по алфавитному порядку букв пароля.

Шифрование столбцовой перестановки (рис. 3.1):

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

Рис. 3.1: Шифрование столбцовой перестановки

Таблица Виженера. Его принцип в том, что каждая буква в исходном шифруемом тексте сдвигается по алфавиту не на фиксированное, а переменное количество символов. Величина сдвига каждой буквы задается ключом (паролем) - секретным словом или фразой, которая используется для шифрования и расшифровки. (рис. 3.2):

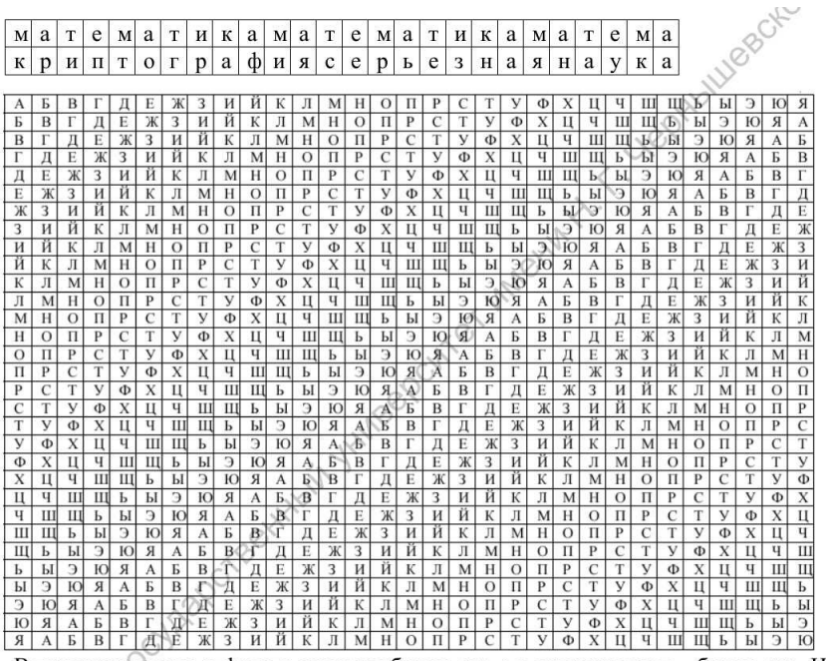


Рис. 3.2: Таблица Виженера

4 Выполнение лабораторной работы

Работа была выполнена на языке программирования Python.

Сначала реализуем шифрование столбцовой перестановки (рис. 4.1):

```python
# Encryption
def encryptMessage(msg, key):
    cipher = ""

    # убираем пробелы
    msg = msg.replace(' ', '')

    # берём длину текста
    msg_len = int(len(msg))

    # создаём список букв этого текста
    msg_lst = list(msg)

    # сортируем буквы ключа по алфавиту
    key_lst = sorted(list(key))

    # считаем количество стобцов
    col = len(key)

    # считаем количество строк
    if msg_len % col == 0:
        row = int(msg_len / col)
    else:
        row = int(msg_len // col) + 1

    # добавляем английскую a, если наша таблица не полная
    fill = int((row * col) - msg_len)
    msg_lst.extend('a' * fill)

    # создадим матрицу нужного размера для шифрования
    matrix = [msg_lst[i: i + col] for i in range(0, len(msg_lst), col)]

    # читаем получившуюся матрицу постолбцово (?)

    for i in range(col):
        # так как до этого мы сортировали в алфавитном порядке,
        # теперь нам надо найти эти буквы в исходном ключе
        # и взять их порядковый номер на рукаве
        curr_idx = key.index(key_lst[i])
        # и соединить это всё в одну строку
        cipher += ''.join([row[curr_idx] for row in matrix])

    return cipher
```

Рис. 4.1: шифрование столбцовой перестановки

Сначала реализуем шифрование столбцовой перестановки (рис. 4.2):

```python
def decryptMessage(cipher, key):
    msg = ""

    # берём длину текста
    msg_len = int(len(cipher))

    # создаём список букв этого текста
    msg_lst = list(cipher)

    # считаем количество стобцов
    col = len(key)

    # считаем количество столбцов
    # по логике этого типа шифрования, у нас будет всегда получаться целое число
    row = int(msg_len / col)

    # сортируем буквы ключа по алфавиту или возрастанию
    key_lst = sorted(list(key))

    # создаём пустую матрицу размера, который мы высчитали
    dec_cipher = []
    for _ in range(row):
        dec_cipher += [[None] * col]

    # Arrange the matrix column wise according
    # to permutation order by adding into new matrix

    # счётчик для номера элемента в списке букв нашего текста
    # не придумал, как сунуть это в цикл, поэтому решил проблему так
    msg_indx = 0

    for i in range(col):
        # так как до этого мы сортировали в алфавитном порядке,
        # теперь нам надо найти эти буквы в исходном ключе
        # и взять их порядковый номер на рукаве
        curr_idx = key.index(key_lst[i])

        for j in range(row):
            dec_cipher[j][curr_idx] = msg_lst[msg_indx]
            msg_indx += 1

    # объединяем воедино
    msg = ''.join(sum(dec_cipher, []))

    return msg
```

Рис. 4.2: шифрование столбцовой перестановки

Вывод программы (рис. 4.3):

Шифрование

```python
while True:
    msg = input(bold + "What message do you want to encrypt?\n" + end + "Note that only russian and english characters and spac
    if (False in [x in a for x in msg]):
        continue
    else:
        break

while True:
    key = input(bold + "\nEnter the key\n" + end + "Note that repeated characters are prohibited:\n")
    if len(set(key)) != len(key):
        continue
    else:
        break

print("\nYour encrypted message is: " + bold + ul + encryptMessage(msg, key))
```

```
What message do you want to encrypt?
Note that only russian and english characters and space are allowed:
inforom

Enter the key
Note that repeated characters are prohibited:
odod

Enter the key
Note that repeated characters are prohibited:
fkdfg

Enter the key
Note that repeated characters are prohibited:
yandex

Your encrypted message is: naoarafaoaim
```

Рис. 4.3: Вывод программы

Реализация таблицы Виженера (рис. 4.4):

Таблица Виженера

In [7]:
```
# повторяем (убираем?) буквы ключа до тех пор, пока не станет
# столько же, сколько у сообщения
def genKey(msg, key):
#     key.replace(' ', '')
#     msg.replace(' ', '')
    key = list(key)
    if len(msg) == len(key):
        return(key)
    else:
        for i in range(len(msg) -
                       len(key)):
            key.append(key[i % len(key)])
    return("" . join(key))
```

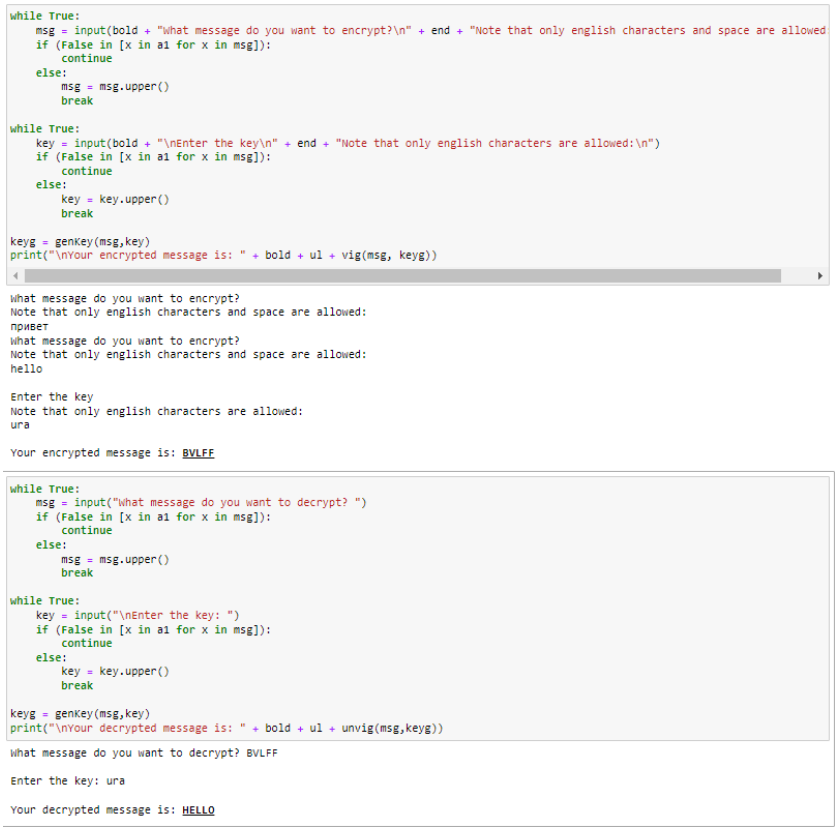
In [8]:
```
# шифрование
def vig(msg, key):
    cipher_text = []
    # убираем пробелы, потому что мы их ненавидим
#     msg.replace(' ', '')
    #key.replace(' ','')
    for i in range(len(msg)):
        x = (ord(msg[i]) + ord(key[i])) % 26
        x += ord('A')
        cipher_text.append(chr(x))
    return("" . join(cipher_text))
```

In [9]:
```
def cipherText(string, key):
    cipher_text = []
    for i in range(len(string)):
        x = (ord(string[i]) +
             ord(key[i])) % 26
        x += ord('A')
        cipher_text.append(chr(x))
    return("" . join(cipher_text))
```

In [10]:
```
# расшифровка
def unvig(cipher_text, key):
    orig_text = []
    #key.replace(' ','')
    for i in range(len(cipher_text)):
        x = (ord(cipher_text[i]) - ord(key[i]) + 26) % 26
        x += ord('A')
        orig_text.append(chr(x))
    return("" . join(orig_text))
```

Рис. 4.4: таблица Виженера

Вывод программы (рис. 4.5):



```
while True:
    msg = input(bold + "What message do you want to encrypt?\n" + end + "Note that only english characters and space are allowed
    if (False in [x in a1 for x in msg]):
        continue
    else:
        msg = msg.upper()
        break

while True:
    key = input(bold + "\nEnter the key\n" + end + "Note that only english characters are allowed:\n")
    if (False in [x in a1 for x in msg]):
        continue
    else:
        key = key.upper()
        break

keyg = genKey(msg,key)
print("\nYour encrypted message is: " + bold + ul + vig(msg, keyg))
```

```
What message do you want to encrypt?
Note that only english characters and space are allowed:
привет
What message do you want to encrypt?
Note that only english characters and space are allowed:
hello

Enter the key
Note that only english characters are allowed:
ura

Your encrypted message is: BVLFF
```

```
while True:
    msg = input("What message do you want to decrypt? ")
    if (False in [x in a1 for x in msg]):
        continue
    else:
        msg = msg.upper()
        break

while True:
    key = input("\nEnter the key: ")
    if (False in [x in a1 for x in msg]):
        continue
    else:
        key = key.upper()
        break

keyg = genKey(msg,key)
print("\nYour decrypted message is: " + bold + ul + unvig(msg,keyg))
```

```
What message do you want to decrypt? BVLFF

Enter the key: ura

Your decrypted message is: HELLO
```

Рис. 4.5: Вывод программы

5 Выводы

Ознакомился с шифрами перестановки и обучился их программной реализации.

Список литературы

1. ТУИС: Математические основы защиты информации и информационной безопасности (02.04.02) [Электронный ресурс]. РУДН, 2022. URL: https://esystem.rudn.ru/course/view.php?id=2084.