Лабораторная работа №2. Шифры перестановки.

Alexander S. Baklashov

28 September, 2023

RUDN University, Moscow, Russian Federation

Цель работы

Цель работы

Рассмотреть шифры перестановки, а именно:

— Маршрутное шифрование

— Шифрование с помощью решеток

— Таблица Виженера

Задачи

Задачи

1. Реализовать маршрутное шифрование.
2. Реализовать шифрование с помощью решеток.
3. Реализовать шифрование с помощью таблицы Виженера.

Задача

Реализовать маршрутное шифрование.

Маршрутное шифрование

Запросим длину блоков и разобьем текст на них

In [1]:
```
text = "нельзя недооценивать противника" # Вводим фразу для шифрования
n = int(input("Введите на блоки какой длины будем разбивать (какой длины будет пароль):"))  # Convert n to an integer
text_without_spaces = text.replace(" ", "")  # Удаление пробелов
text_length = len(text_without_spaces) # Отмечаем его длину

if text_length % n != 0: # Если длина блока меньше n
    padding_length = n - (text_length % n) # Считаем насколько меньше
    text_without_spaces += "a" * padding_length # Добавляем буквы а в пустые места

matrix = [text_without_spaces[i:i+n] for i in range(0, len(text_without_spaces), n)] # Разбиваем текст на блоки
for i in range(len(matrix)):
    print(matrix[i])
```

```
Введите на блоки какой длины будем разбивать (какой длины будет пароль):6
нельзя
недооц
ениват
ьпроти
вникаа
```

Figure 1: Маршрутное шифрование (1)

Маршрутное шифрование

Запросим пароль и построим столбцы в соотв. с алф. порядком букв в пароле

In [2]:
```
flag = 1
password = ''
m = str(n)
while flag == 1: # Ввод пароля
    password = input("Введите пароль из " + m + " символов: ")
    if len(password) == n:
        flag = 0
    else:
        print ("Неправильно, нужно " + m + " символов")
```

```
Введите пароль из 6 символов: ав
Неправильно, нужно 6 символов
Введите пароль из 6 символов: пароль
```

In [3]:
```
# Создаем список индексов букв из пароля в алфавитном порядке
sorted_indices = sorted(range(n), key=lambda x: password[x])

# Выводим строки в соответствии с порядком букв в пароле
for i in range(len(matrix)):
    sorted_row = ''.join([matrix[i][j] for j in sorted_indices if j < len(matrix[i])])
    print("[" + sorted_row + "]")
```

```
[езьнля]
[еоондц]
[навеит]
[птоьри]
[наквиа]
```

Figure 2: Маршрутное шифрование (2)

Маршрутное шифрование

Выведем результат

In [5]:
```
# Создаем строки, объединяя символы из каждого столбца
result = ""
for j in sorted_indices:
    column = [matrix[i][j] for i in range(len(matrix)) if j < len(matrix[i])]
    result += ''.join(column)

# Преобразуем результат в верхний регистр
result = result.upper()

# Выводим результат
print (result)
```

```
ЕЕНПНЗОАТАЬОВОКННЕЬВЛДИРИЯЦТИА
```

Figure 3: Маршрутное шифрование (3)

Задача

Реализовать шифрование с помощью решеток.

Шифрование с помощью решеток

Заполним исх. матрицу и выявим ячейки, числа в которых будем вырезать

In [1]:
```python
import numpy as np

# Задаем размерность решетки k x k
k = 2

# Создаем список k_2, который содержит числа от 1 до k^2
k_2 = [x + 1 for x in range(k**2)]

# Создаем матрицу размером 2k x 2k, заполненную нулями
matrix = [[0 for x in range(2 * k)] for y in range(2 * k)]

# Преобразуем матрицу в массив NumPy для удобства
matrix = np.array(matrix)

# Заполняем матрицу числами из списка k_2 в специфичном порядке
for x in range(k**2):
    c = 0
    for x in range(k):
        for y in range(k):
            matrix[x][y] = k_2[c]
            c += 1
    matrix = np.rot90(matrix)  # Поворачиваем матрицу на 90 градусов

# Создаем словарь ds для подсчета встречаемости чисел в матрице
ds = {k: 0 for k in k_2}

# Задаем словарь dss, который содержит ожидаемое количество каждого числа
dss = {1: 2, 2: 4, 3: 3, 4: 3}

# Подсчитываем встречаемость чисел в матрице и помечаем некорректные числа
for x in range(k**2):
    for y in range(k**2):
        ds[matrix[x][y]] += 1
        if ds[matrix[x][y]] != dss[matrix[x][y]]:
            matrix[x][y] = -1
        else:
            matrix[x][y] = 0
```

Figure 4: Шифрование с помощью решеток (1)

Шифрование с помощью решеток

Зададим шифротекст и ключ и выведем результат, поворачивая матрицу против часовой стрелки и вставляя соотв. буквы

In [2]:
```
# Задаем открытый текст
text = 'договорподписали'

# Задаем ключ для расшифровки
key = 'шифр'

# Инициализируем переменные
t = iter(text)  # Создаем итератор для открытого текста

# Создаем матрицу matrixt для хранения расшифрованного текста
matrixt = [['0' for y in range(k**2)] for x in range(k**2)]

# Перебираем 4 поворота матрицы и заполняем расшифрованный текст
for d in range(4):
    for x in range(k**2):
        for y in range(k**2):
            if matrix[x][y] == 0:
                matrixt[x][y] = next(t)  # Заполняем буквами из открытого текста
    matrix = np.rot90(matrix, -1)  # Поворачиваем матрицу на -90 градусов (против часовой стрелки)

# Создаем строку с русским алфавитом
rus = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'

# Создаем список ps, содержащий индексы букв ключа в русском алфавите
ps = [rus.index(x) for x in key]

# Сортируем индексы по возрастанию
pss = sorted(ps)

# Инициализируем строку для зашифрованного текста
output = ''

# Собираем зашифрованный текст, учитывая порядок столбцов, определенный ключом
for letter in pss:
    for x in range(k**2):
        output += matrixt[x][ps.index(letter)]

# Выводим зашифрованный текст
print(output)
```

```
оворддгпапиосдои
```

Figure 5: Шифрование с помощью решеток (2)

Шифрование с помощью таблицы Виженера

Создадим функцию для шифрования

In [1]:

```python
def vigenere_encrypt(plain_text, key):
    """
    Функция для шифрования текста методом Виженера.

    Args:
        plain_text (str): Исходный текст для шифрования.
        key (str): Ключевое слово или фраза для шифрования.

    Returns:
        str: Зашифрованный текст.
    """
    encrypted_text = []  # Создаем пустой список для хранения зашифрованных символов
    key_length = len(key)

    for i in range(len(plain_text)):
        char = plain_text[i]
        if char.isalpha():
            # Если символ буквенный, применяем шифр Виженера
            key_char = key[i % key_length]  # Берем символ ключа с учетом цикличности
            shift = ord(key_char.lower()) - ord('a')  # Вычисляем сдвиг

            if char.isupper():
                # Обработка для заглавных букв
                encrypted_char = chr(((ord(char) - ord('A') + shift) % 33) + ord('A'))

            else:
                # Обработка для строчных букв
                encrypted_char = chr(((ord(char) - ord('a') + shift) % 33) + ord('a'))
        else:
            # Если символ не буквенный, оставляем его без изменений
            encrypted_char = char

        encrypted_text.append(encrypted_char)

    return ''.join(encrypted_text)  # Собираем зашифрованный текст в одну строку
```

Figure 6: Шифрование с помощью таблицы Виженера (1)

Создадим функцию для дешифрования

In [2]:
```python
def vigenere_decrypt(encrypted_text, key):
    """
    Функция для расшифровки текста, зашифрованного методом Виженера.

    Args:
        encrypted_text (str): Зашифрованный текст.
        key (str): Ключевое слово или фраза, используемое для шифрования.

    Returns:
        str: Расшифрованный текст.
    """
    decrypted_text = []  # Создаем пустой список для хранения расшифрованных символов
    key_length = len(key)

    for i in range(len(encrypted_text)):
        char = encrypted_text[i]
        if char.isalpha():
            # Если символ буквенный, применяем обратный шифр Виженера
            key_char = key[i % key_length]  # Берем символ ключа с учетом цикличности
            shift = ord(key_char.lower()) - ord('a')  # Вычисляем сдвиг

            if char.isupper():
                # Обработка для заглавных букв
                decrypted_char = chr(((ord(char) - ord('A') - shift) % 33) + ord('A'))
            else:
                # Обработка для строчных букв
                decrypted_char = chr(((ord(char) - ord('a') - shift) % 33) + ord('a'))
        else:
            # Если символ не буквенный, оставляем его без изменений
            decrypted_char = char

        decrypted_text.append(decrypted_char)

    return ''.join(decrypted_text)  # Собираем расшифрованный текст в одну строку
```

Figure 7: Шифрование с помощью таблицы Виженера (2)

Шифрование с помощью таблицы Виженера

Зададим шифротекст и ключ и выведем результат

```
In [3]: message = "криптография серьезная наука"  # Исходный текст
        key = "математика"  # Ключевое слово
        message = message.replace(" ", "")  # Удаление пробелов
        encrypted_message = vigenere_encrypt(message, key)
        print("Зашифрованный текст:", encrypted_message)

        decrypted_message = vigenere_decrypt(encrypted_message, key)
        print("Расшифрованный текст:", decrypted_message)

        Зашифрованный текст: цръфюохшкффявкььчпчакнтшца
        Расшифрованный текст: криптографиясерьезнаянаука
```

Figure 8: Шифрование с помощью таблицы Виженера (3)

Вывод

Вывод

В ходе данной лабораторной работы я рассмотрел и реализовал такие шифры перестановки, как маршрутное шифрование, шифрование с помощью решеток и таблица Виженера.