

Отчёт по лабораторной работе №2

Математические основы защиты информации и информационной безопасности

Шифры простой замены

**Выполнил: Мануэл Марсия Педру,
НФИмд-02-25, 1032255503**

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
3 Реализация шифрования с помощью решеток	7

Список иллюстраций

3.1 Реализация маршрутного шифрования	7
3.2 Реализация маршрутного шифрования	8
3.3 Реализация шифрования с помощью решеток	8
3.4 РРеализация шифрования с помощью решеток	9
3.5 Проверка (рис. 3.5):	9
3.6 Р Проверка	10

Список таблиц

1 Цель работы

Изучить шифры простой замены и научиться их реализовывать.

2 Выполнение лабораторной работы

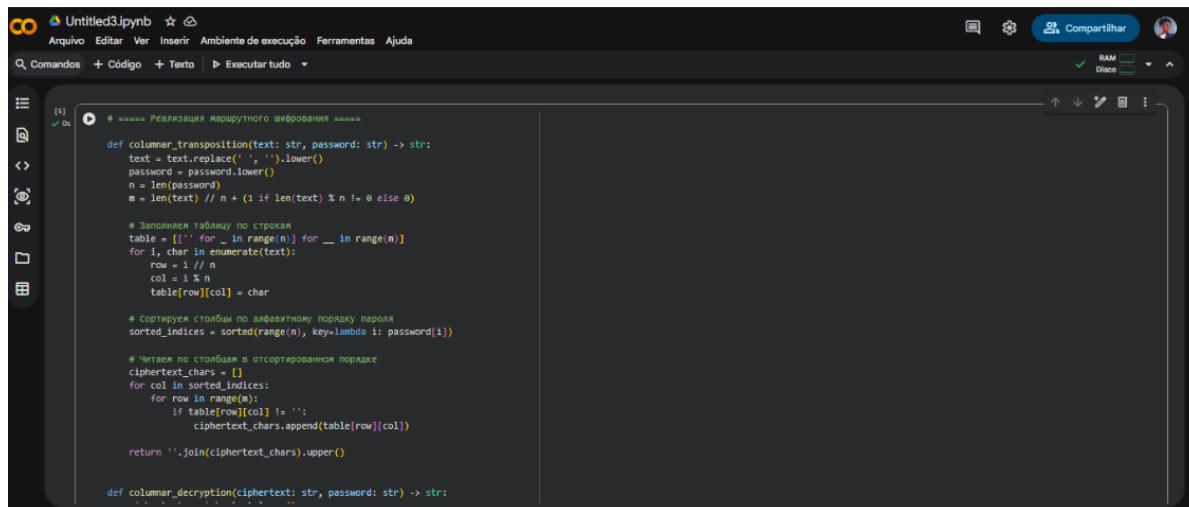
Реализация маршрутного шифрования
Данный способ шифрования разработал французский математик Франсуа Виет. Открытый текст записывают в некоторую геометрическую фигуру (обычно прямоугольник) по некоторому пути, а затем, выписывая символы по другому пути, получают шифртекст.

3 Реализация шифрования с помощью решеток

Данный способ шифрования предложил австрийский криптограф Эдуард Флейснер в 1881 году. Суть этого способа заключается в следующем. Выбирается натуральное число $\times > 1$, строится квадрат размерности \times и построчно заполняется числами $1, 2, \dots, \times^2$.

#Реализация таблицы Виженера 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.

Выполним реализацию этого алгоритма на языке Python (рис. 3.1):



```
# ===== Реализация маршрутного шифрования =====

def columnar_transposition(text: str, password: str) -> str:
    text = text.replace(' ', '').lower()
    password = password.lower()
    n = len(password)
    m = len(text) // n + (1 if len(text) % n != 0 else 0)

    # Заполняем таблицу по строкам
    table = [[None for _ in range(n)] for __ in range(m)]
    for i, char in enumerate(text):
        row = i // n
        col = i % n
        table[row][col] = char

    # Сортируем столбцы по алфавитному порядку первая
    sorted_indices = sorted(range(n), key=lambda i: password[i])

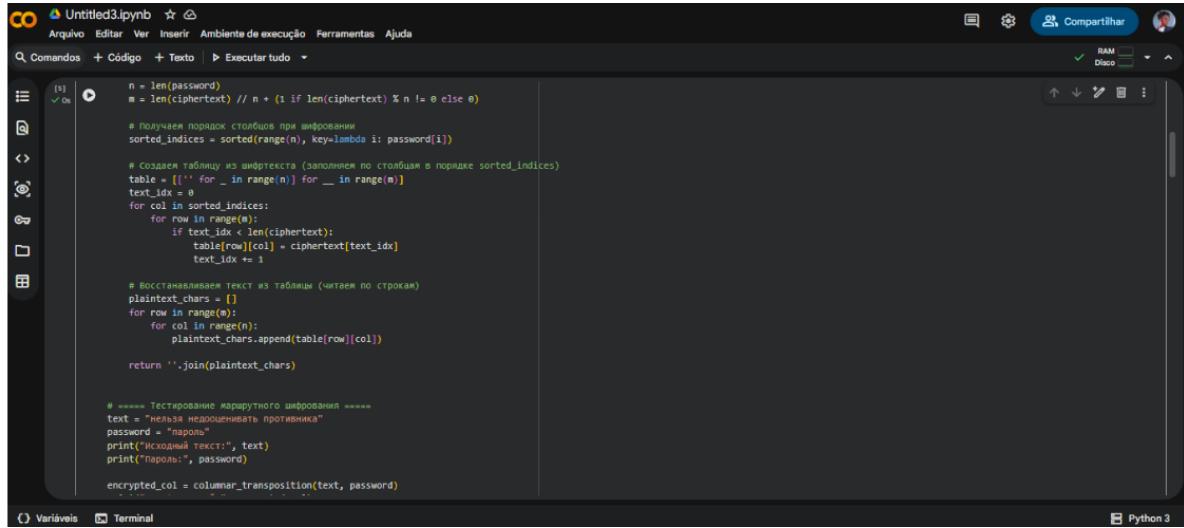
    # Читаем по столбцам в отсортированном порядке
    ciphertext_chars = []
    for col in sorted_indices:
        for row in range(m):
            if table[row][col] != '':
                ciphertext_chars.append(table[row][col])

    return ''.join(ciphertext_chars).upper()

def columnar_decryption(ciphertext: str, password: str) -> str:
```

Рис. 3.1: Реализация маршрутного шифрования

Проверим работу алгоритма (рис. 3.2):



```
n = len(password) // n + (1 if len(ciphertext) % n != 0 else 0)

# Получаем порядок столбцов при шифровании
sorted_indices = sorted(range(n), key=lambda i: password[i])

# Создаем таблицу из шифртекста (заполним по столбцам в порядке sorted_indices)
table = ['' for _ in range(n) for __ in range(m)]
text_idx = 0
for col in sorted_indices:
    for row in range(n):
        if text_idx < len(ciphertext):
            table[row][col] = ciphertext[text_idx]
            text_idx += 1

# Восстанавливаем текст из таблицы (читаем по строкам)
plaintext_chars = []
for row in range(n):
    for col in range(m):
        plaintext_chars.append(table[row][col])

return ''.join(plaintext_chars)

# ***** Тестирование маршрутного шифрования *****
text = "Нельзя недооценивать противника"
password = "пароль"
print("Исходный текст:", text)
print("Пароль:", password)

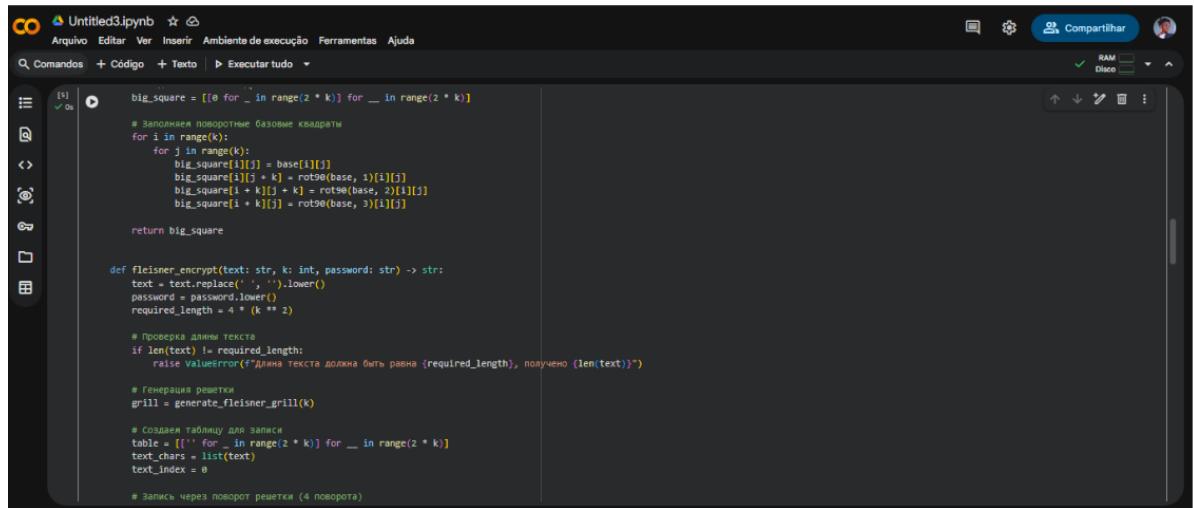
encrypted_col = columnar_transposition(text, password)
```

Рис. 3.2: Реализация маршрутного шифрования

Проверим работу алгоритма (рис. 3.2):

Реализация шифрования с помощью решеток (рис. ??):](image/3.png){ #fig:003 width=100% height=100% }

Выполним реализацию этого алгоритма на языке Python (рис. 3.3):



```
big_square = [[0 for _ in range(2 * k)] for __ in range(2 * k)]

# Заполняем поворотные базовые квадраты
for i in range(k):
    for j in range(k):
        big_square[i][j] = base[1][j]
        big_square[i][j + k] = rot90(base, 1)[1][j]
        big_square[i + k][j + k] = rot90(base, 2)[1][j]
        big_square[i + k][j] = rot90(base, 3)[1][j]

return big_square

def fleisner_encrypt(text: str, k: int, password: str) -> str:
    text = text.replace(' ', '').lower()
    password = password.lower()
    required_length = 4 * (k ** 2)

    # Проверка длины текста
    if len(text) != required_length:
        raise ValueError(f"Длина текста должна быть равна {required_length}, получено {len(text)}")

    # Генерация решетки
    grille = generate_fleisner_grille(k)

    # Создаем таблицу для записи
    table = [['' for _ in range(2 * k)] for __ in range(2 * k)]
    text_chars = list(text)
    text_index = 0

    # Запись через поворот решетки (4 поворота)
```

Рис. 3.3: Реализация шифрования с помощью решеток

Проверим работу алгоритма (рис. 3.4):

```
[1]: table = [['' for _ in range(2 * k)] for __ in range(2 * k)]
text_chars = list(text)
text_index = 0

# Запись через поворот решетки (4 поворота)
for rotation in range(4):
    rotated_grill = rotate(grill, rotation)

    # Поиск ячек для записи (значения от 1 до k**2)
    for i in range(2 * k):
        for j in range(2 * k):
            if 1 <= rotated_grill[i][j] <= k ** 2:
                if text_index < len(text_chars):
                    table[i][j] = text_chars[text_index]
                text_index += 1

# Сортируем столбцы по алфавитному порядку букв пароля
sorted_indices = sorted(range(2 * k), key=lambda i: password[i % len(password)])

ciphertext_chars = []
for col in sorted_indices:
    for i in range(2 * k):
        ciphertext_chars.append(table[i][col])

return ''.join(ciphertext_chars).upper()

# ===== Тестирование шифрования решетками =====
k = 2
grill_text = "договор подписан"
grill_text_clean = grill_text.replace(' ', '').lower()
```

Рис. 3.4: Реализация шифрования с помощью решеток

Проверим работу алгоритма (рис. 3.5):

Рис. 3.5: Проверка (рис. 3.5):

Выполним реализацию этого алгоритма на языке Python (рис. 3.6):

The screenshot shows a Jupyter Notebook interface with a single code cell containing Python code for testing the Vigenère cipher. The code imports the `vigenere` module and defines two strings: `vigenere_text` ("криптография серьезная наука") and `vigenere_key` ("математика"). It prints the original text and key, then encrypts and decrypts the text using the key. The output shows the original text, the key, the encrypted text ("зашифрованный"), and the decrypted text ("расшифрованный"). The decrypted text matches the original input.

```
# ===== Тестирование шифра Виженера =====
vigenere_text = "криптография серьезная наука"
vigenere_key = "математика"
print("Исходный текст:", vigenere_text)
print("Ключ:", vigenere_key)

encrypted_vig = vigenere_cipher(vigenere_text, vigenere_key)
print("Зашифрованный:", encrypted_vig)

decrypted_vig = vigenere_decipher(encrypted_vig, vigenere_key)
print("Расшифрованный:", decrypted_vig)

... Исходный текст: нельзя недооценивать противника
Пароль: пароль
Зашифрованный: ЕЕННЭЗДАТЬСВОКННЬВДИРИЯДИ
Расшифрованный: нельзя недооцениватьпротивника

дополнен ТЕКСТ до 16 символов
исходный ТЕКСТ: договор подписан
Подготовленный ТЕКСТ: договорподписан
длина текста: 16
к: 2
Пароль: инфо
Зашифрованный: ООДЛЮХГРНЧвдос

исходный ТЕКСТ: криптография серьезная наука
Ключ: математика
Зашифрованный: ЧРЫЮДУХЕХДЫЧРМННПИА
Расшифрованный: криптография серьезная наука
```

Рис. 3.6: Р Проверка