Шаблон отчёта по лабораторной работе №2

Дисциплина: Математические основы защиты информации и информационной безопасности

Миличевич Александра

Содержание

Список иллюстраций

Список таблиц

Цель работы

Познакомиться с шифрами перестановки.

Задание

1. Программно реализовать маршрутное шифрование.
2. Программно реализовать шифрование с помощью решёток.
3. Программно реализовать шифр Виженера.

Выполнение лабораторной работы

Маршрутное шифроваие

1) Код сначала подготавливает текст для шифрования, заменяя пробелы на “sp” и определяя размеры матрицы на основе длины текста и заданного ключа. Затем, символы текста последовательно записываются в матрицу по строкам, заполняя её слева направо и сверху вниз. Эта матрица используется для маршрутного шифрования.

```python
def route_cipher_encrypt(text, key):
    """
    Шифрует текст с помощью маршрутного шифрования.

    Args:
        text (str): Текст для шифрования.
        key (int): Количество столбцов в маршрутной матрице.

    Returns:
        str: Зашифрованный текст.
    """

    # 1. Подготовка
    text = text.replace(" ", "sp")  # Удаляем пробелы из текста
    text_length = len(text)
    num_columns = key  # Количество столбцов задается ключом
    num_rows = (text_length + num_columns - 1) // num_columns  #  Вычисляем необходимое количество строк

    # 2. Заполнение матрицы
    matrix = [['' for _ in range(num_columns)] for _ in range(num_rows)]  # создем пустую матрицу
    index = 0
    for row in range(num_rows):
        for column in range(num_columns):
            if index < text_length: # Записываем символы текста в матрицу
                matrix[row][column] = text[index]
                index += 1
```

Рис. 1: реализация маршрутного шифрования

1.1 # 3. Чтение матрицы по столбцам cipher_text = “” После заполнения матрицы, код считывает ее по столбцам сверху вниз, формируя зашифрованный текст. Символы каждого столбца добавляются в общий шифрованный текст,

который затем возвращается как результат работы алгоритма. Таким образом текст шифруется путем записи по строкам и чтения по столбцам.

```
    # 3. Чтение матрицы по столбцам
    cipher_text = ""
    for column in range(num_columns): # читаем матрицу по столбцам и формируем шифрованный текст
        for row in range(num_rows):
            cipher_text += matrix[row][column]

    return cipher_text


# пример текста
text = "this is a secret message"
key = 4

encrypted_text = route_cipher_encrypt(text, key)


print(f"Original text: {text}")
print(f"Encrypted text: {encrypted_text}")
```

```
Original text: this is a secret message
Encrypted text: tssprpshppsemaiiaetegssscsse
```

Рис. 2: реализация маршрутного шифрования2

Шифрование с помощью решеток

Этот код реализует шифрование с использованием решетки (grill e cipher). Сначала он подготавливает текст, удаляя пробелы и проверяя, что текст не превышает размер решетки. Затем, он создает матрицу (решетку) нужного размера, заполняет ее символами текста в порядке, заданном списком grille, и, наконец, считывает матрицу построчно, формируя зашифрованный текст.

В примере, решетка (grille) задается списком кортежей, где каждый кортеж указывает строку и столбец, куда должен быть помещен соответствующий символ текста. Результат работы алгоритма — зашифрованный текст, сформированный из прочитанных по строкам элементов матрицы.

Шифрование с помощью решеток

```python
def grille_cipher_encrypt(text, grille):
    """
    Шифрует текст с использованием метода шифрования решеткой.

    Args:
        text (str): Текст для шифрования.
        grille (list of tuples): Решетка, определяющая позиции, куда будут помещены символы.

    Returns:
        str: Зашифрованный текст.
    """

    # 1. Подготовка:
    text = text.replace(" ", "s")  # Удаляем пробелы из текста
    text_length = len(text) # Определяем длину текста
    grille_size = len(grille) # Определяем размер решетки (сторона квадрата)

    if text_length > grille_size * grille_size:
        raise ValueError("Текст длиннее, чем вместимость решетки.")

    # 2. Создаем пустую матрицу (квадрат)
    matrix_size = grille_size
    matrix = [['' for _ in range(matrix_size)] for _ in range(matrix_size)]

    # 3. Заполняем матрицу в соответствии с решеткой и текстом
    index = 0
    for row, column in grille:
        if index < text_length:
            matrix[row][column] = text[index]
            index += 1

    # 4. Читаем матрицу построчно
    cipher_text = ""
    for row in range(matrix_size):
        for column in range(matrix_size):
            cipher_text += matrix[row][column]

    return cipher_text

# Пример использования:
text = "криптография"
# Определяем решетку как список кортежей (строка, столбец)
grille = [(0, 1), (1, 3), (2, 0), (3, 2), (0,0), (1,2), (2,1), (3,3), (0,2), (1,0), (2,3), (3,1), (0,3),(1,1),(2,2),(3,0)]

encrypted_text = grille_cipher_encrypt(text, grille)

print(f"Исходный текст: {text}")
print(f"Зашифрованный текст: {encrypted_text}")
```

```
Исходный текст: криптография
Зашифрованный текст: ткафоригияпр
```

Рис. 3: реализация шифрования с помощью решеток

Таблица Виженера

Этот код реализует шифрование Виженера, где каждый символ текста сдвигается на величину, определяемую соответствующим символом ключа, повторяющегося по циклу. Функция vigenere_cipher_encrypt сначала приводит текст и ключ к верхнему регистру, а затем для каждой буквы текста вычисляет её сдвиг, используя соответствующую букву ключа и выполняя операцию взятия остатка от деления на 26. Результатом является шифрованный текст, где неалфавитные символы остаются неизменными.В этом конкретном примере текст “HELLO” шифруется с помощью ключа “KEY”, что приводит к зашифрованному тексту, где буквы сдвигаются на величину, задаваемую буквами ключа. Разница между шифрованием Виженера и другими шифрами,

например, маршрутным или решеточным, заключается в том, что Виженер использует полиалфавитный шифр с циклическим ключом, в то время как маршрутный и решеточный шифры используют перестановку символов.

Шифрование Виженера

```python
def vigenere_cipher_encrypt(text, key):
    text = text.upper()
    key = key.upper()
    key_length = len(key)
    cipher_text = ""
    for i, char in enumerate(text):
         if 'A' <= char <= 'Z':
            text_char_code = ord(char) - ord('A')
            key_char_code = ord(key[i % key_length]) - ord('A')
            encrypted_char_code = (text_char_code + key_char_code) % 26
            encrypted_char = chr(encrypted_char_code + ord('A'))
            cipher_text += encrypted_char
         else:
              cipher_text += char
    return cipher_text

# Пример использования:
text = "HELLO"
key = "KEY"
encrypted_text = vigenere_cipher_encrypt(text, key)

print(f"Исходный текст: {text}")
print(f"Зашифрованный текст: {encrypted_text}")
```

```
Исходный текст: HELLO
Зашифрованный текст: RIJVS
```

Рис. 4: реализация шифрования таблица виженера

В коде для шифрования Виженера на русском языке, добавлен русский алфавит (alphabet_ru) и его длина (alphabet_length), чтобы обрабатывать символы русского языка. Индексы букв берутся из русского алфавита alphabet_ru.find(), и для шифрования используется модуль от деления на длину русского алфавита. Если символ не найден в русском алфавите, он добавляется в зашифрованный текст без изменений.

```python
def vigenere_cipher_encrypt_ru(text, key):
    """
    Шифрует текст на русском языке, используя шифр Виженера.

    Args:
        text (str): Текст для шифрования на русском.
        key (str): Ключ шифрования на русском.

    Returns:
        str: Зашифрованный текст на русском.
    """
    text = text.upper() # Приводим текст к верхнему регистру
    key = key.upper() # Приводим ключ к верхнему регистру
    key_length = len(key) # Запоминаем длину ключа
    cipher_text = "" # Создаём пустую строку для зашифрованного текста

    alphabet_ru = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ" # Русский алфавит
    alphabet_length = len(alphabet_ru) # Запоминаем длину алфавита

    for i, char in enumerate(text):
        if char in alphabet_ru: # Проверяем, является ли символ русской буквой
            text_char_code = alphabet_ru.find(char) # Получаем индекс буквы в русском алфавите
            key_char_code = alphabet_ru.find(key[i % key_length]) # Получаем индекс буквы ключа
            encrypted_char_code = (text_char_code + key_char_code) % alphabet_length # Шифрование
            encrypted_char = alphabet_ru[encrypted_char_code] # Получаем зашифрованную букву
            cipher_text += encrypted_char # Добавляем зашифрованную букву в результирующую строку
        else: # Если символ не является буквой русского алфавита
              cipher_text += char # Добавляем его в результирующую строку без изменений

    return cipher_text

# Пример использования:
text = "ПРИВЕТМИР"
key = "КЛЮЧ"
encrypted_text = vigenere_cipher_encrypt_ru(text, key)

print(f"Исходный текст: {text}")
print(f"Зашифрованный текст: {encrypted_text}")
```

```
Исходный текст: ПРИВЕТМИР
Зашифрованный текст: ЬЬЖЩПЮКАЫ
```

Рис. 5: с русским алфавитом таблица Виженера

Выводы

Программно реализованы шифры перестановки.