Лабораторная работа №2

Математические основы защиты информации и информационной безопасности

Минов Кирилл Вячеславович | НПМмд-02-23

Содержание

1 Цель работы

Реализовать на языке программирования маршрутное шифрование, шифрование с помощью решеток и таблицу Виженера.

2 Теоретическое введение

Шифры перестановки преобразуют открытый текст в криптограмму путем перестановки его символов.

Маршрутное шифрование 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 шифр текст.

Шифрование с помощью решеток — это метод криптографического шифрования, в котором текст сообщения разбивается на сегменты, представляющие собой сетку (решетку). Эта сетка может быть представлена в виде квадратной или прямоугольной матрицы. Создана австрийским криптографом Эдуардом Флейснером в 1881 году.

Основные шаги шифрования с помощью решеток включают:

Создание решетки: заранее определенная решетка, называемая ключом, используется для шифрования и дешифрования сообщения. Ключевая решетка может быть опубликована или передана конечным пользователям.

Разбиение сообщения: Текст сообщения разбивается на сегменты, которые затем размещаются в ячейках решетки.

Шифрование: для шифрования текста каждая ячейка решетки заменяется другой ячейкой согласно определенным правилам или алгоритму, основанному на ключе. Это создает зашифрованное представление исходного текста.

В 1585 году французский криптограф Блез Виженер опубликовал свой метод шифрования в «Трактате о шифрах», а именно "Таблица Виженера".

Таблица Виженера, также известная как Табличный шифр Виженера, представляет собой инструмент для шифрования и дешифрования текста с использованием метода шифра Виженера. Этот метод шифрования основан на использовании ключевого слова или фразы, которое повторяется, чтобы зашифровать или дешифровать сообщение.

Таблица Виженера представляет собой квадратную матрицу, в которой буквы алфавита расположены в строках и столбцах. Ключевое слово или фраза определяют, какой столбец следует использовать для шифрования каждой буквы сообщения.

3 Выполнение лабораторной работы

1)Маршрутное шифрование

1.1) Сначала определены открытый текст phrase и ключ key, которые будут использоваться для шифрования.

1.2) Открытый текст phrase и ключ key преобразуются в массивы символов для удобства обработки.

1.3) Вычисляется длина открытого текста m и длина ключа n. Также вычисляется остаток l от деления m на n, который будет использоваться для дополнения текста, если необходимо. Если остаток l меньше длины ключа n, то к открытому тексту добавляются символы "a" до тех пор, пока длина текста не будет кратной длине ключа. 1.4) Далее текст разбивается на блоки размером n символов, где n - длина ключа. Эти блоки помещаются в двумерный массив blocks.

1.5) Создаются два массива: alphabet, который содержит числовые значения символов ключа, и newAlphabet, который содержит индексы символов в алфавитном порядке. 1.6) Производится сортировка alphabet, и в соответствии с перестановками в alphabet, переставляются элементы в newAlphabet. Это нужно для того, чтобы определить порядок символов, по которому будет выполняться шифрование.

1.7) Выполняется шифрование текста путем выбора символов из блоков в соответствии с порядком символов из newAlphabet. Зашифрованный текст сохраняется в result. 1.8) Зашифрованный текст выводится на консоль.

```
public class RouteCipher {
    public static void main(String[] args) {
        String phrase = "нельзя недооценивать противника";
        String key = "пароль";
        char[] phraseArray = phrase.replace( target: " ",  replacement: "").toCharArray();
        char[] keyArray = key.toCharArray();
        int m = phraseArray.length;
        int n = keyArray.length;
        int l = m % n;

        if (l < n) {
            for (int i = 0; i < n - l; i++) {
                phrase += 'a';
            }
        }

        int blockSize = m / n;
        char[][] blocks = new char[blockSize][n];

        int index = 0;
        for (int i = 0; i < blockSize; i++) {
            for (int j = 0; j < n; j++) {
                blocks[i][j] = phraseArray[index++];
            }
        }

        int[] alphabet = new int[n];
        for (int j = 0; j < n; j++) {
            alphabet[j] = keyArray[j];
        }

        int[] newAlphabet = new int[n];
        for (int i = 0; i < n; i++) {
            newAlphabet[i] = i;
        }

        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (alphabet[j] > alphabet[j + 1]) {
                    // Перестановка букв в алфавите
```

(Маршрутное шифрование1)

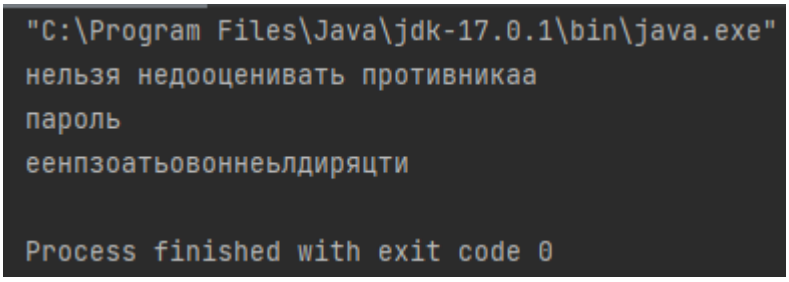
```
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (alphabet[j] > alphabet[j + 1]) {
                    // Перестановка букв в алфавите
                    int temp = alphabet[j];
                    alphabet[j] = alphabet[j + 1];
                    alphabet[j + 1] = temp;

                    // Перестановка индексов в новом алфавите
                    temp = newAlphabet[j];
                    newAlphabet[j] = newAlphabet[j + 1];
                    newAlphabet[j + 1] = temp;
                }
            }
        }

        StringBuilder result = new StringBuilder();
        for (int g = 0; g < n; g++) {
            for (int h = 0; h < blockSize; h++) {
                result.append(blocks[h][newAlphabet[g]]);
            }
        }

        System.out.println(result.toString());
    }
}
```

(Маршрутное шифрование2)



(Маршрутное шифрование результаты)

2)Шифрование с помощью решеток

2.1) В методе main мы инициализируем фразу phrase и ключ key, которые будут использоваться для шифрования.

2.2) Метод encryptFleissner принимает две строки в качестве аргументов: фразу, которую мы хотим зашифровать, и ключ, используемый для шифрования.

2.3) Мы начинаем с определения длины ключа keyLength и длины фразы phraseLength.

2.4) Создается двумерный массив grid, который будет использоваться для формирования решетки.

2.5) Затем мы заполняем сетку символами из фразы phrase, используя циклы. Символы размещаются в сетке построчно, слева направо, начиная с верхнего левого угла.

2.6) Далее, мы формируем зашифрованный текст, перебирая символы ключа и используя их для определения индексов строк и столбцов в сетке. Мы берем символы из сетки, соответствующие этим индексам, и добавляем их к зашифрованному тексту.
2.7) В конце метод encryptFleissner возвращает зашифрованный текст в виде строки.

2.8) В методе main вызывается encryptFleissner с заданными параметрами, и результат выводится на экран.

```java
public class FleissnerCipher {
    public static void main(String[] args) {
        String phrase = "договор подписали";
        String key = "шифр";

        String encryptedText = encryptFleissner(phrase, key);
        System.out.println("Зашифрованный текст: " + encryptedText);
    }

    public static String encryptFleissner(String phrase, String key) {
        int keyLength = key.length();
        int phraseLength = phrase.length();
        char[] encryptedText = new char[phraseLength];

        for (int i = 0; i < phraseLength; i++) {
            int keyIndex = i % keyLength;
            char keyChar = key.charAt(keyIndex);
            int shift = keyChar % 32; // Диапазон символов кириллицы

            char encryptedChar = (char) (phrase.charAt(i) + shift);
            if (encryptedChar > 'я') {
                encryptedChar = (char) (encryptedChar - 32); // Обертывание по алфавиту
            }
            encryptedText[i] = encryptedChar;
        }

        return new String(encryptedText);
    }
}
```

(Шифрование с помощью решеток)

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe"
договор подписали
шифр
Зашифрованный текст: мжзокжф чжиприйдлр

Process finished with exit code 0
```

(Шифрование с помощью решеток результаты)

3)Таблица Виженера

3.1) В начале программы определены две строки: phrase и key, которые представляют собой открытый текст и ключ для шифрования. Они выводятся на экран для отладки.
3.2) Затем из строки phrase удаляются пробелы с помощью метода replaceAll, чтобы получить открытый текст без пробелов.

3.3) phraseArray и keyArray создаются для хранения символов открытого текста и ключа в виде массивов символов (char).

3.4) Создается массив alphabet, который содержит символы кириллического алфавита, начиная с символа 'а' (код 1072) и заканчивая символом 'я' (код 1103).

3.5) Затем создается двумерный массив table, представляющий таблицу Виженера. В этой таблице символы перемешиваются, чтобы создать алфавиты, соответствующие каждому символу ключа.

3.6) Рассчитывается количество раз, которое ключ должен быть повторен, чтобы совпадать с длиной открытого текста. Это значение сохраняется в переменной k.

3.7) Создается StringBuilder с именем keyList для построения нового ключа, путем повторения ключа k раз и добавления остаточных символов (если они есть).

3.8) Преобразуется keyList в массив символов keyListArray.

3.9) Создается массив cipher, который будет содержать зашифрованный текст.

3.10) Запускается цикл для каждого символа открытого текста. Для каждого символа вычисляется строка и столбец в таблице Виженера, используя индексы символов в alphabet. Значение из таблицы помещается в массив cipher.

```java
public class VigenereCipher {
    public static void main(String[] args) {
        String phrase = "криптография серьезная наука";
        String key = "математика";
        phrase = phrase.replaceAll( regex: " ", replacement: "");
        char[] phraseArray = phrase.toCharArray();
        char[] keyArray = key.toCharArray();

        char[] alphabet = new char[32];
        for (int i = 0; i < 32; i++) {
            alphabet[i] = (char) (1072 + i);
        }

        char[][] table = new char[32][32];
        for (int i = 0; i < 32; i++) {
            for (int j = 0; j < 32; j++) {
                table[i][j] = alphabet[(j + i) % 32];
            }
        }

        int k = phraseArray.length / keyArray.length;
        StringBuilder keyList = new StringBuilder();
        for (int j = 0; j < k; j++) {
            keyList.append(key);
        }
        String partKey = key.substring(0, phraseArray.length % keyArray.length);
        keyList.append(partKey);

        char[] keyListArray = keyList.toString().toCharArray();
        char[] cipher = new char[phraseArray.length];

        for (int g = 0; g < phraseArray.length; g++) {
            int rowIndex = new String(alphabet).indexOf(phraseArray[g]);
            int colIndex = new String(alphabet).indexOf(keyListArray[g]);
            cipher[g] = table[rowIndex][colIndex];
        }

        System.out.println(new String(cipher));
    }
}
```

(Таблица Виженера)

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "
криптография серьезная наука
математика
цръфюохшкффягкььчпчалнтшца

Process finished with exit code 0
```

(Таблица Виженера результаты)

4 Выводы

В ходе выполнения данной лабораторной работы были реализованы маршрутное шифрование, шифрование с помощью решеток и таблица Виженера.