

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Факультет программной инженерии и компьютерной техники

**ЛАБОРАТОРНАЯ РАБОТА № 1.1
ПО ДИСЦИПЛИНЕ «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»
Основы шифрования данных**

Выполнил: Давыдов Иван Денисович

Группа: Р3400

Вариант 5

Санкт-Петербург
2020/2021

Цель работы

Изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

Задание

Реализовать в программе шифрование и дешифрацию файла с использованием квадрата Кардано размером 4x4.

Листинг

CardanGrille.java

```
package lab1;

public class CardanGrille {

    private int[][] grille;

    private final int GRILLE_SIZE = 4;
    private final int GRILLE_ELEM_AMOUNT = 16;
    private boolean toLeft = false;

    CardanGrille(boolean toLeft) {
        this.toLeft = toLeft;
        grille = new int[4][4];
        generateGrille();
    }

    /*
        Метод генерации решетки Карадано
        Вырезы в решетке должны быть сделаны таким образом, чтобы несколько разных вырезов
        не попадали на одну ячейку.
        Построим такую решетку следующим образом: разбиваем матрицу на 4 части (на квадраты
        2x2)
        Одну часть заполняем цифрами от 1 до 4, поворачиваем ее на 90 градусов и записываем
        в соседний квадрат и т.д.
        Случайным образом вырезаем все цифры, причем каждая может вырезаться лишь раз
        (1) 2 3 1
        3 4 (4) 2
        2 4 4 (3)
        1 3 (2) 1
    */
    private void generateGrille() {
        int[][] tempQuad = new int[2][2];
        int[][] tempGrille = new int[4][4];

        // случайным образом заполняем квадрант
        int quad = (int) (Math.random() * 4);
        for (int i = 1; i < 4 + 1; i++) {
            tempQuad[quad / 2][quad % 2] = i;
            quad = (quad + 1) % 4;
        }
    }
}
```

```

// заполняем маску решетки, перемещая и поворачивая исходный квадрант
int k=0;
while(true) {
    if(k == 4)
        break;

    for (int i = 0; i < tempQuad.length; i++) {
        for (int j = 0; j < tempQuad[i].length; j++) {
            int rowBase = quad < 2 ? i:i+2;
            int clmnBase = quad < 2 ? j+quad*2: (quad == 2?j+2:j);
            tempGrille[rowBase][clmnBase] = tempQuad[i][j];
        }
    }
    tempQuad = rotateMatrixRight(tempQuad);
    quad = (quad + 1) % 4;
    k++;
}

// случайным образом делаем вырезы в итоговой решетке
k = 1;
while(true) {
    if(k==5)
        break;
    int quadToCut = (int)(Math.random() * 4);
    for(int i =0; i<tempQuad.length; i++){
        for(int j =0; j<tempQuad[i].length; j++){
            int rowBase = quadToCut < 2 ? i:i+2;
            int clmnBase = quadToCut < 2 ? j+quadToCut*2: (quadToCut == 2?j+2:j);

            if(tempGrille[rowBase][clmnBase] == k)
                grille[rowBase][clmnBase]=1;
        }
    }
    k++;
}

}

// метод поворота матрицы на 90 градусов по часовой стрелке
private int[][] rotateMatrixRight(int[][] srcArr){
    int[][] resultArray = new int[srcArr[0].length][srcArr.length];
    for (int i = 0; i < srcArr.length; i++) {
        for (int j = 0; j < srcArr[i].length; j++) {
            resultArray[j][srcArr.length - i - 1] = srcArr[i][j];
        }
    }
    return resultArray;
}

// метод поворота матрицы на 90 градусов против часовой стрелки
private int[][] rotateMatrixToLeft(int[][] srcArr) {
    int[][] resultArray = new int[srcArr[0].length][srcArr.length];
    for (int i = 0; i < srcArr.length; i++) {
        for (int j = 0; j < srcArr[i].length; j++) {
            resultArray[srcArr[i].length - j - 1][i] = srcArr[i][j];
        }
    }
    return resultArray;
}
}

```

```

/*
 * Метод для шифрации исходно текста решеткой Крдано
 */
public String encrypt(String text)
{
    int[][] tempMatrix = grille;
    int rightAngleTurns = 0, circleSpinsCounter = 0, encryptedCharsCount = 0;
    String paddedText = text;

    // добавление мусорных символов
    if (paddedText.length() % GRILLE_ELEM_AMOUNT != 0)
        paddedText = paddedText.concat(Util.getRandomString(GRILLE_ELEM_AMOUNT -
(paddedText.length() % GRILLE_ELEM_AMOUNT)));

    String result = " ";
    result = String.format("%"+paddedText.length()+"s", result);

    while (true) {
        // При текущем положении решетки проходим по всем ячейкам, находим все вырезы и
        // вписываем символы
        for (int i=0; i<GRILLE_SIZE; i++) {
            for (int j =0; j<GRILLE_SIZE; j++) {
                if (tempMatrix[i][j] == 1) {

                    //построчно переносим матрицу в итоговую строку
                    StringBuilder stringBuilder = new StringBuilder(result);
                    stringBuilder.setCharAt(i * GRILLE_SIZE + j + GRILLE_ELEM_AMOUNT *
circleSpinsCounter, paddedText.charAt(encryptedCharsCount));
                    result = stringBuilder.toString();

                    encryptedCharsCount++;

                    if (encryptedCharsCount == paddedText.length())
                        return result;
                }
            }
        }
        // Поворачиваем решетку на 90 градусов
        tempMatrix = this.toLeft ?
rotateMatrixToLeft(tempMatrix):rotateMatrixRight(tempMatrix);
        rightAngleTurns++;

        // Если решетка повернулась на 360 градусов, смещаем ее на другой блок
        if (rightAngleTurns == 4) circleSpinsCounter++;
        rightAngleTurns %= 4;
    }
}

```

```
// Метод для дешифрации текста решеткой Кардано
public String decrypt(String text)
{
    int[][] tempMatrix = grille;
    int rightAngleTurns = 0, circleSpinsCounter = 0, decryptedCharsCount = 0;
    String result = "";

    while (true) {
        // Идем по решетке в том же порядке, что при шифровании, и выписываем символы из
        // зашифрованной строки
        for (int i=0; i<GRILLE_SIZE; i++) {
            for (int j =0; j<GRILLE_SIZE; j++) {
                if (tempMatrix[i][j] == 1) {
                    result = result+text.charAt(i * GRILLE_SIZE + j + GRILLE_ELEM_AMOUNT
* circleSpinsCounter);

                    decryptedCharsCount++;
                    if (decryptedCharsCount == text.length())
                        return result;
                }
            }
        }
        // Поворачиваем решетку на 90 градусов
        tempMatrix = this.toLeft ?
rotateMatrixToLeft(tempMatrix):rotateMatrixRight(tempMatrix);
        rightAngleTurns++;

        // Если решетка повернулась на 360 градусов, смещаем ее на другой блок
        if (rightAngleTurns == 4) circleSpinsCounter++;
        rightAngleTurns %= 4;
    }
}
}
```

Main.java

```
package lab1;

import org.apache.commons.cli.*;
import java.io.IOException;
import java.nio.file.StandardOpenOption;
import java.nio.file.Files;
import java.nio.file.Paths;

public class Main {

    public static void main(String[] args){
        //опции командной строки
        Options options = new Options();

        Option input = new Option("s", "source", true, "input text file path");
        input.setRequired(true);
        options.addOption(input);

        Option encrypted = new Option("e", "encrypted", true, "file path to encrypted
text");
        encrypted.setRequired(true);
        options.addOption(encrypted);
    }
}
```

```

Option decrypted = new Option("d", "decrypted", true, "file path to decrypted
text");
decrypted.setRequired(true);
options.addOption(decrypted);

Option toLeft = new Option("l", "left", false, "direction of rotation");
toLeft.setRequired(false);
options.addOption(toLeft);

CommandLineParser parser = new DefaultParser();

try {
    CommandLine cmd = parser.parse(options, args);

    //получаем аргументы командной строки
    String inputFile = cmd.getOptionValue("source");
    String encryptedFile = cmd.getOptionValue("encrypted");
    String decryptedFile = cmd.getOptionValue("decrypted");

    String fileContents = new String(Files.readAllBytes(Paths.get(inputFile)));

    //шифруем
    CardanGrille encryptor = new CardanGrille(cmd.hasOption("left"));

    String encryptedText = encryptor.encrypt(fileContents);

    Files.write(Paths.get(encryptedFile), encryptedText.getBytes("utf-8"),
StandardOpenOption.CREATE, StandardOpenOption.TRUNCATE_EXISTING);

    //дешифруем
    String decryptedText = encryptor.decrypt(encryptedText);

    Files.write(Paths.get(decryptedFile), decryptedText.getBytes("utf-8"),
StandardOpenOption.CREATE, StandardOpenOption.TRUNCATE_EXISTING);

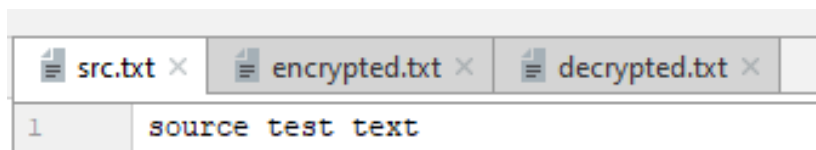
    } catch (ParseException | IOException e) {
        System.out.println(e.getMessage());

        System.exit(1);
    }
}

```

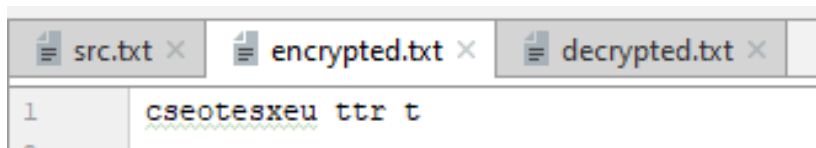
Результаты

Исходный текст:



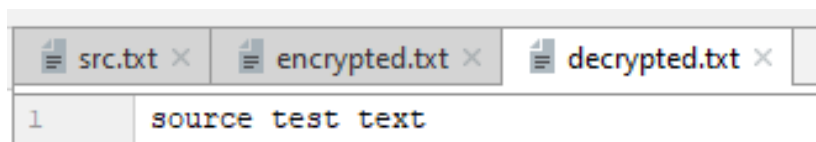
```
src.txt x encrypted.txt x decrypted.txt x
1 source test text
```

Текст, полученный в результате шифрования:



```
src.txt x encrypted.txt x decrypted.txt x
1 cseotesxeu ttr t
2
```

Текст, полученный в результате дешифрования:



```
src.txt x encrypted.txt x decrypted.txt x
1 source test text
```

Выводы

В ходе данной лабораторной работы был изучен метод шифрования и дешифрования текстовых сообщений с использованием решетки Кардано. Была написана программа на основе этого метода, реализующая решетку размером 4x4.