Отчёт по лабораторной работе №7

Шифр гаммирования

Шемякин Алексей НФИбд-02-18

Содержание

List of Figures

1 Цель работы

Изучение алгоритма шифрования гаммированием

2 Теоретические сведения

2.1 Шифр гаммирования

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, т.е. последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных.

Принцип шифрования гаммированием заключается в генерации гаммы шифра с помощью датчика псевдослучайных чисел и наложении полученной гаммы шифра на открытые данные обратимым образом (например, используя операцию сложения по модулю 2). Процесс дешифрования сводится к повторной генерации гаммы шифра при известном ключе и наложении такой же гаммы на зашифрованные данные. Полученный зашифрованный текст является достаточно трудным для раскрытия в том случае, если гамма шифра не содержит повторяющихся битовых последовательностей и изменяется случайным образом для каждого шифруемого слова. Если период гаммы превышает длину всего зашифрованного текста и неизвестна никакая часть исходного текста, то шифр можно раскрыть только прямым перебором (подбором ключа). В этом случае криптостойкость определяется размером ключа.

Метод гаммирования становится бессильным, если известен фрагмент исходного текста и соответствующая ему шифрограмма. В этом случае простым вычитанием по модулю 2 получается отрезок псевдослучайной последовательности и по нему восстанавливается вся эта последовательность.

Метод гаммирования с обратной связью заключается в том, что для получения сегмента гаммы используется контрольная сумма определенного участка шифруемых данных. Например, если рассматривать гамму шифра как объединение непересекающихся множеств H(j), то процесс шифрования можно пердставить следующими шагами:

1. Генерация сегмента гаммы H(1) и наложение его на соответствующий участок шифруемых данных.
2. Подсчет контрольной суммы участка, соответствующего сегменту гаммы H(1).
3. Генерация с учетом контрольной суммы уже зашифрованного участка данных следующего сегмента гамм H(2).
4. Подсчет контрольной суммы участка данных, соответствующего сегменту данных H(2) и т.д.

3 Выполнение работы

3.1 Реализация шифратора и дешифратора Java

```
public class Main {
    public static void main(String [] args) throws IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.i
        BufferedReader reader2 = new BufferedReader(new InputStreamReader(System.
        HashMap<Character, String> map = new HashMap<Character ,String>();
        map.put('0',"0000");
        map.put('1',"0001");
        map.put('2',"0010");
        map.put('3',"0011");
        map.put('4',"0100");
        map.put('5',"0101");
        map.put('6',"0110");
        map.put('7',"0111");
        map.put('8',"1000");
        map.put('9',"1001");
        map.put('A',"1010");
        map.put('B',"1011");
        map.put('C',"1100");
        map.put('D',"1101");
        map.put('E',"1110");
```

```
map.put('F',"1111");


String text="";
String cipher;
String cipher2;
System.out.println("Введите: " +
        "\n'1' если хотите определить шифротекст по ключу и открытому тек
        "\n'2' если хотите определить ключ по открытому тексту и шифротек

int input = Integer.parseInt(reader.readLine());
if(input==1) {
    System.out.println("Введите ключ шифрования (ключ должен быть в шестн
    cipher= reader2.readLine();
    System.out.println("Введите открытый текст (размерность текста должна
}else {
    System.out.println("Введите  шифротекст  : ");
    cipher= reader.readLine();

    System.out.println("Введите открытый текст (размерность текста должна
}
cipher2 = reader.readLine();
cipher2=  characterto16(cipher2,map);

String shifr = shifrovanie(cipher,cipher2,map);

if(input==1) {
    System.out.println("Шифротекст : "+shifr);
}else {
```

```
        System.out.println("Ключ : "+shifr);
    }
}

public static String characterto16 (String cipher,HashMap<Character, String>
    char[] charArray = cipher.toCharArray();
    String finalcode="";
    for (char character : charArray) {
        String code = Integer.toString((int) character, 2);
        StringBuilder curcode = new StringBuilder(code);
        for (int j = 0; j < 8 - code.length(); j++) {
            curcode.insert(0, "0");
        }
        code = curcode.toString();
        finalcode = getString(map, finalcode, code);
    }
    return finalcode;
}

private static String getString(HashMap<Character, String> map, String finalc
    String val = code.substring(0, 4);
    String val2= code.substring(4);
    char nval=' ';
    char nval2=' ';

    for (Map.Entry<Character, String> characterStringEntry : map.entrySet())
        if (((Map.Entry) characterStringEntry).getValue().equals(val)) {
            nval = (char) ((Map.Entry) characterStringEntry).getKey();
        }
```

```
        if (((Map.Entry) characterStringEntry).getValue().equals(val2)) {
            nval2 = (char) ((Map.Entry) characterStringEntry).getKey();
        }
    }
    String v = String.valueOf(nval)+String.valueOf(nval2);
    finalcode=finalcode+v+" ";
    return finalcode;
}

public static String shifrovanie(String cipher, String cipher2,HashMap<Charac
    String[] splt = cipher.split("\\s+");
    String[] splt2 = cipher2.split("\\s+");

    String finalcode="";
    for(int i=0;i<splt.length;i++) {

        char[] symbols = splt[i].toCharArray();
        String symbol = map.get(symbols[0])+map.get(symbols[1]);

        char[] symbols2 = splt2[i].toCharArray();
        String symbol2 = map.get(symbols2[0])+map.get(symbols2[1]);

        StringBuilder newSymbol = new StringBuilder();
        for(int j=0;j<symbol2.length();j++) {
            int number= Character.digit(symbol2.charAt(j), 10);
            int number2 = Character.digit(symbol.charAt(j), 10);
            newSymbol.append(number ^ number2);
        }
        finalcode = getString(map, finalcode, newSymbol.toString());
```

```
        }
        return finalcode;
    }
}
```

3.2 Контрольный пример 1

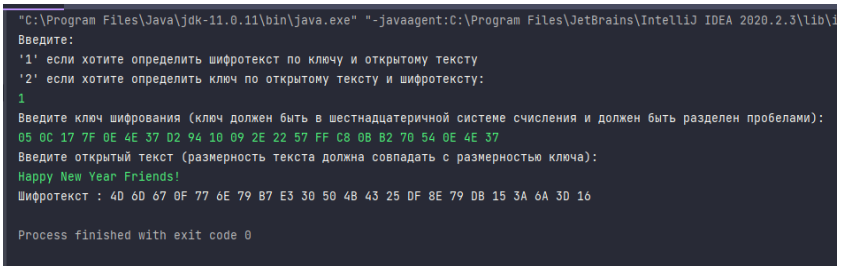

Figure 3.1: Работа алгоритма гаммирования

3.3 Контрольный пример 2

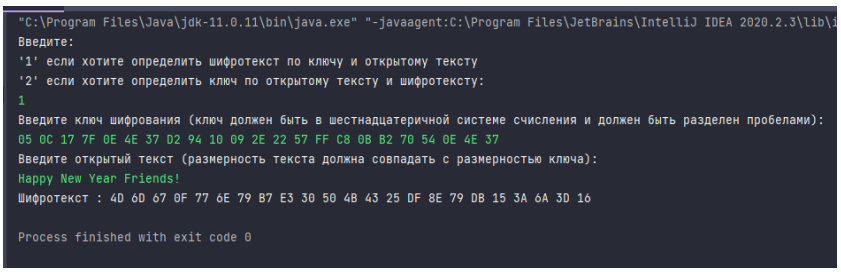

Figure 3.2: Работа алгоритма гаммирования

4 Выводы

Изучили алгоритмы шифрования на основе гаммирования

Список литературы

1. Шифрование методом гаммирования
2. Режим гаммирования в блочном алгоритме шифрования