Отчёт по лабораторной работе №8

Шифр гаммирования

Альсид Мона НФИбд-03-18

Содержание

[1 Цель работы 1](#_Toc90730214)

[2 Теоретические сведения 1](#_Toc90730215)

[2.1 Шифр гаммирования 1](#_Toc90730216)

[2.2 Идея взлома 2](#_Toc90730217)

[3 Выполнение работы 3](#_Toc90730218)

[3.1 Реализация взломщика, шифратора и дешифратора на Python 3](#_Toc90730219)

[3.2 Контрольный пример 6](#_Toc90730220)

[4 Выводы 6](#_Toc90730221)

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

# 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.

## 2.2 Идея взлома

Шифротексты обеих телеграмм можно получить по формулам режима однократного гаммирования:

Открытый текст можно найти, зная шифротекст двух телеграмм, зашифрованных одним ключом. Для это оба равенства складываются по модулю 2. Тогда с учётом свойства операции XOR получаем:

Предположим, что одна из телеграмм является шаблоном — т.е. имеет текст фиксированный формат, в который вписываются значения полей. Допустим, что злоумышленнику этот формат известен. Тогда он получает достаточно много пар (известен вид обеих шифровок). Тогда зная имеем:

Таким образом, злоумышленник получает возможность определить те символы сообщения , которые находятся на позициях известного шаблона сообщения . В соответствии с логикой сообщения , злоумышленник имеет реальный шанс узнать ещё некоторое количество символов сообщения . Затем вновь используется равенство с подстановкой вместо полученных на предыдущем шаге новых символов сообщения . И так далее. Действуя подобным образом, злоумышленник даже если не прочитает оба сообщения, то значительно уменьшит пространство их поиска.

# 3 Выполнение работы

## 3.1 Реализация взломщика, шифратора и дешифратора на Python

import java.util.HashMap;  
import java.util.Iterator;  
import java.util.Map;  
import java.util.Scanner;  
public class Shifrovka {  
public static void main(String [] args) {  
 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");  
 //System.out.println(shifrovanie("14 15 15 ","41 43 42",map));  
 String text="";  
 String cipher;  
 String cipher2;   
 Scanner in = new Scanner(System.in);  
 System.out.println("enter '1' if you want to determine ciphertext by key and plaintext \n or '2' if you want to determine plaintext by ciphertext:");  
 int input = in.nextInt();  
 if(input==1) {  
 Scanner in2 = new Scanner(System.in);  
 System.out.println("enter encryption key: ");   
 cipher= in2.nextLine();  
 System.out.println("enter plaintext: ");  
 cipher2 = in2.nextLine();  
 cipher2= characterto16(cipher2,map);  
 String shifr = shifrovanie(cipher,cipher2,map);  
 System.out.println("ciphertext : "+shifr);  
   
 }else {  
 Scanner in2 = new Scanner(System.in);  
 System.out.println("enter the first ciphertext (через пробелы) : ");  
 cipher= in2.nextLine();  
 System.out.println("enter the second ciphertext (через пробелы) : ");  
 cipher2= in2.nextLine();  
 System.out.println("enter the plain text of one of the messages in order to decrypt the plain text of the second message:");  
 text =in2.nextLine();  
 String C1xorC2= shifrovanie(cipher,cipher2,map);  
 String cipher16=characterto16(text,map);  
 String result = shifrovanie(C1xorC2,cipher16,map);  
 System.out.println("открытый текст второго сообщения: "+tocharacter(result,map));   
 }  
}  
public static String characterto16 (String cipher,HashMap<Character, String> map) {  
 char[] chararray = cipher.toCharArray();  
 String finalcode="";  
 for(int i=0;i<chararray.length;i++) {  
 char character = chararray[i];  
 int ascii = (int) character;  
 String code = Integer.toString(ascii,2);  
 String curcode=code;  
 for(int j=0;j<8-code.length();j++) {  
 curcode="0"+curcode;  
 }  
 code= curcode;  
 String val = code.substring(0, 4);  
 String val2= code.substring(4);  
 char nval=' ';  
 char nval2=' ';  
 Iterator it = map.entrySet().iterator();  
 while (it.hasNext()) {  
 Map.Entry pair = (Map.Entry)it.next();  
 if(pair.getValue().equals(val)) {  
 nval=(char)pair.getKey();  
 }  
 if(pair.getValue().equals(val2)) {  
 nval2=(char)pair.getKey();  
 }  
 }  
 String v = String.valueOf(nval)+String.valueOf(nval2);  
 finalcode=finalcode+v+" ";   
 }  
 return finalcode;  
}  
public static String tocharacter(String cipher, HashMap<Character, String> map) {  
 String[] splt = cipher.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]);  
 int number = Integer.parseInt(symbol, 2);  
 finalcode+=Character.toString ((char) number);  
}  
 return finalcode;  
 }  
public static String shifrovanie(String cipher, String cipher2,HashMap<Character, String> map) {  
   
   
 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]);  
 String newsymbol="";  
 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+=number^number2;  
 }  
 String val = newsymbol.substring(0, 4);  
 String val2= newsymbol.substring(4);  
 char nval=' ';  
 char nval2=' ';  
 Iterator it = map.entrySet().iterator();  
 while (it.hasNext()) {  
 Map.Entry pair = (Map.Entry)it.next();  
 if(pair.getValue().equals(val)) {  
 nval=(char)pair.getKey();  
 }  
 if(pair.getValue().equals(val2)) {  
 nval2=(char)pair.getKey();  
 }  
 }  
 String v = String.valueOf(nval)+String.valueOf(nval2);  
 finalcode=finalcode+v+" ";   
 }  
 return finalcode;  
}  
}

## 3.2 Контрольный пример

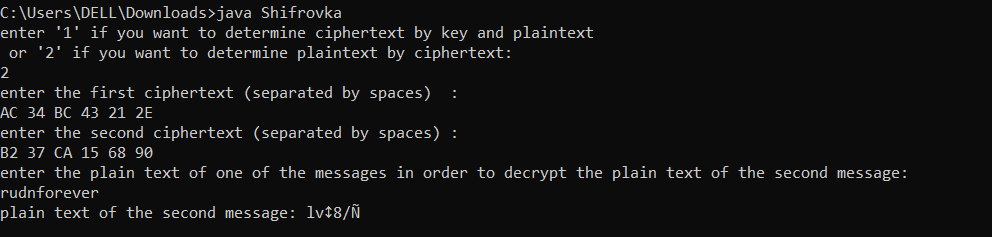


Figure 1: Работа алгоритма взлома ключа

# 4 Выводы

В ходе выполнения лабораторной работы было разработано приложение, позволяющее шифровать тексты в режиме однократного гаммирования.