Министерство образования Российской Федерации
Московский Государственный Технический
Университет им. Н.Э. Баумана

Отчет по лабораторной работе №8
По курсу «Анализ алгоритмов»

Тема: «Алгоритм поточной обработки»

Студент: **Медведев А.В.**
Группа: **ИУ7-51**

Преподаватель: **Волкова Л.Л.**

Москва, 2017

Содержание

Постановка задачи

В ходе лабораторной работы предстоит:

1. реализовать муравьиный алгоритм на языке программирования;
2. сравнить работу алгоритма при разных значениях параметров, задающих веса феромона и времени жизни колонны.

Реализация

Суть алгоритма

Поточный шифр - это симметричный шифр, в котором каждый символ открытого текста преобразуется в символ шифрованного текста в зависимости не только от используемого ключа, но и от его расположения в потоке открытого текста.

Алгоритм RC4, как и любой потоковый шифр, строится на основе генератора псевдослучайных битов. На вход генератора записывается ключ, а на выходе читаются псевдослучайные биты. Идеальным вариантом с точки зрения стойкости для потокового шифра, является размер ключа, сопоставимый с размером шифруемых данных. Тогда каждый бит открытого текста объединяется с соответствующим битом ключа посредством суммирования по модулю 2 (XOR), образуя зашифрованную последовательность. Для расшифровки требуется проделать ту же операцию еще раз на принимающей стороне.

Алгоритм шифрования:

- Функция генерирует последовательность битов (k_i).
- Затем последовательность битов посредством операции «суммирование по модулю два» (xor) объединяется с открытым текстом (mi). В результате получается шифрограмма $(c_i) : c_i = m_i \oplus k_i$.

Алгоритм расшифровки:

- Повторно создаётся (регенерируется) поток битов ключа (ключевой поток) (k_i).
- Поток битов ключа складывается с шифрограммой (c_i) операцией «xor». В силу свойств операции «xor» на выходе получается исходный (не зашифрованный) текст$(m_i) : m_i = c_i \oplus k_i = (m_i \oplus k_i) \oplus k_i$

Программная реализация алгоритма

Листинг 1: Класс RC4

```
public class RC4
{

    byte[] S = new byte[256];
    int x = 0;
    int y = 0;
    public RC4(byte[] key)
    {
        init(key);
    }
    private static void Swap(byte[] s, int i, int j)
    {
        byte c = s[i];
        s[i] = s[j];
        s[j] = c;
    }
    private byte keyItem()
    {
        x = (x + 1) % 256;
        y = (y + S[x]) % 256;

        Swap(S,x, y);

        return S[(S[x] + S[y]) % 256];
    }
    private void init(byte[] key)
    {
        int keyLength = key.Length;

        for (int i = 0; i < 256; i++)
        {
            S[i] = (byte)i;
        }

        int j = 0;

        for (int i = 0; i < 256; i++)
        {
            j = (j + S[i] + key[i % keyLength]) % 256;
            Swap(S,i, j);
```

```
41          }
42      }
43
44      public byte[] Encode(byte[] dataB, int size)
45      {
46          byte[] data = dataB.Take(size).ToArray();
47
48          byte[] cipher = new byte[data.Length];
49
50          for (int m = 0; m < data.Length; m++)
51          {
52              cipher[m] = (byte)(data[m] ^ keyItem());
53          }
54
55          return cipher;
56      }
57
58      public byte[] Decode(byte[] dataB, int size)
59      {
60          return Encode(dataB, size);
61      }
62
63  }
```

Методы:

- keyItem() - генератор псевдослучайной последовательности
- init() - начальной инициализация вектора-перестановки ключём
- Encode() - Шифрование
- Decode() - Расшифровка

Листинг 2: Вызов поточного алгоритма

```
1  static void Main(string[] args)
2  {
3      byte[] key = ASCIIEncoding.ASCII.GetBytes("Key");
4      RC4 rc4= new RC4(key);
5      RC4 rc42= new RC4(key);
6
7      Encoder encoder= new Encoder(rc4);
8      Decoder decoder= new Decoder(rc42);
9      encoder.SetNext onveyer(decoder);
```

```
10      string testString = "Final laba";
11
12      foreach (char c in testString)
13      {
14          byte[] testBytes = ASCIIEncoding.ASCII.GetBytes(c.
                ToString());
15          encoder.Enqueue(testBytes);
16      }
17      encoder.Run();
18      decoder.Run();
19
20
21  }
```

Заключение

Во время выполнения работы был изучен и реализован алгоритм поточной обработки данных.