



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	3
Реализация	3
Суть алгоритма	3
Программная реализация алгоритма	3
Заключение	6

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» (хор) объединяется с открытым текстом (m_i) . 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 «хор». В силу свойств операции «хор» 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

```
1 public class RC4
2 {
3
4     byte[] S = new byte[256];
5     int x = 0;
6     int y = 0;
7     public RC4(byte[] key)
8     {
9         init(key);
10    }
11    private static void Swap(byte[] s, int i, int j)
12    {
13        byte c = s[i];
14        s[i] = s[j];
15        s[j] = c;
16    }
17    private byte keyItem()
18    {
19        x = (x + 1) % 256;
20        y = (y + S[x]) % 256;
21
22        Swap(S, x, y);
23
24        return S[(S[x] + S[y]) % 256];
25    }
26    private void init(byte[] key)
27    {
28        int keyLength = key.Length;
29
30        for (int i = 0; i < 256; i++)
31        {
32            S[i] = (byte)i;
33        }
34
35        int j = 0;
36
37        for (int i = 0; i < 256; i++)
38        {
39            j = (j + S[i] + key[i % keyLength]) % 256;
40            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.
15             ToString());
16         encoder.Enqueue(testBytes);
17     }
18     encoder.Run();
19     decoder.Run();
20
21 }
```

Заключение

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