Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

Московский государственный технический университет имени Н.Э.Баумана
(МГТУ им. Н.Э.Баумана)

# ОТЧЕТ
## По лабораторной работе № 8
## По курсу «Анализ алгоритмов»
## на тему «Алгоритм поточной обработки»

Выполнил Студент: Московец Н.С
Группа: ИУ7-51

Москва, 2017

## Оглавление



## Постановка задачи

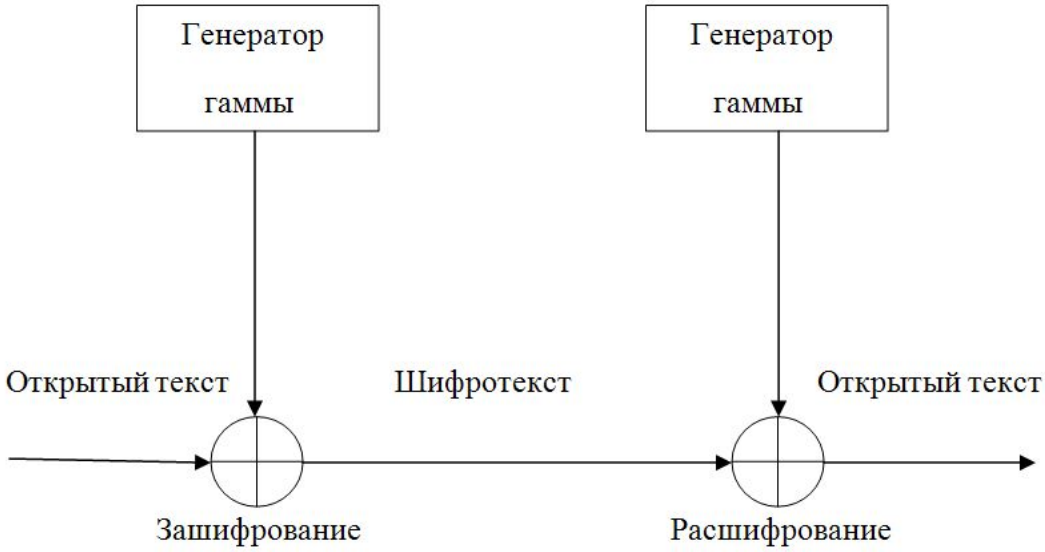
Изучить и реализовать поточный алгоритм обработки данных.

## Описание алгоритма

Поточный алгоритм обработки данных был реализован для шифрации входной последовательности символов и ее дешифрации с помощью потокового шифра RC4.

Алгоритм RC4, как и любой потоковый шифр, строится на основе генератора псевдослучайных битов. На вход генератора записывается ключ, а на выходе читаются псевдослучайные биты. Длина ключа может составлять от 40 до 2048 бит. Генерируемые биты имеют равномерное распределение.

Ядро алгоритма поточных шифров состоит из функции — генератора псевдо случайных битов (гаммы), который выдаёт поток битов ключа (ключевой поток, гамму, последовательность псевдослучайных битов).



Алгоритм шифрования.

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

Алгоритм расшифровки.

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

## Реализация

RC4

```
1. void RC4::swap(byte *arr, int i, int j)
2. {
3.    byte tmp = arr[i];
4.    arr[i] = arr[j];
5.    arr[j] = tmp;
6. }
7.
8. RC4::RC4(unsigned char *key, size_t size)
9. {
10.      init(key, size);
11.   }
12.
13.   void RC4::init(unsigned char *key, size_t size)
14.   {
15.      for (int i = 0; i < 256; i++) {
16.         gamma[i] = (byte)i;
17.      }
18.
19.      int j = 0;
20.      for (int i = 0; i < 256; i++) {
21.         j = (j + gamma[i] + key[i % size]) % 256;
22.         swap(gamma, i, j);
23.      }
24.      indI = indJ = 0;
25.   }
26.
27.   byte RC4::kword()
```

```
28.  {
29.      indI = (indI + 1) % 256;
30.      indJ = (indJ + gamma[indI]) % 256;
31.      swap(gamma, indI, indJ);
32.      byte K = gamma[(gamma[indI] + gamma[indJ]) % 256];
33.      return K;
34.  }
35.
36.  char RC4::code(char ch)
37.  {
38.      return (char)(ch ^ kword());
39.  }
```

Функция кодирования:

```
1.  void codeData(RingBuffer<char, BUFSIZE> &buff, timeType sleep, const
    char* str,
2.                                      unsigned char* key, size_t size)
3.  {
4.      RC4 code(key, size);
5.      size_t i = 0;
6.      while(i < strlen(str)) {
7.          std::this_thread::sleep_for(std::chrono::milliseconds(sleep));
8.
9.          if(!buff.isFull())
10.                 buff.push(code.code(str[i++]));
11.         }
12.     }
```

Функция дешифрации:

```
1.  void decodeData(RingBuffer<char, BUFSIZE> &buff, timeType sleep,
    size_t strSize,
2.                                          unsigned char* key, size_t size)
3.  {
4.      RC4 decode(key, size);
5.      size_t i = 0;
6.      while(i < strSize) {
7.          std::this_thread::sleep_for(std::chrono::milliseconds(sleep));
8.
9.          if(!buff.isEmpty()) {
10.                 char x;
11.                 buff.pop(x);
12.                 x = decode.code(x);
13.                 cout << x << flush;
14.                 i++;
15.             }
16.         }
17.         cout << endl;
18.     }
```

Основная программа:

```
1. int main(int argc, char *argv[])
2. {
3.     RingBuffer<char, BUFSIZE> data;
4.     unsigned char K[] = { 2, 3, 1, 0, 3 };
5.     size_t KSize = 5;
6.     const char* str = "Hello World!";
7.
8.     std::thread coder(&codeData, std::ref(data), 1, str, K, KSize);
9.     std::thread decoder(&decodeData, std::ref(data), 500, strlen(str), K,
   KSize);
10.
11.        coder.join();
12.        decoder.join();
13.        return 0;
14.    }
```

## Заключение

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