МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №1.3
“Поточное симметричное шифрование”
по дисциплине
“Информационная безопасность”

Студент:

Алексеев Даниил Иннокентьевич

Группа P34302

Преподаватель:

Рыбаков Степан Дмитриевич

Санкт-Петербург
2023

Цели работы:

- Цель работы: изучение структуры и основных принципов работы современных алгоритмов поточного симметричного шифрования, приобретете.

Задание (вариант 9)

Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью алгоритма RC4

Алгоритм RC4

Является методом симметричного поточного шифрования.

Симметричное шифрование – шифрование при котором для шифрования и дешифрования сообщения используется один и тот же ключ.

При поточном шифровании символ открытого текста преобразовывается в зависимости от ключа и от положения символоа в потоке открытого текста.

Алгоритм шифрования.

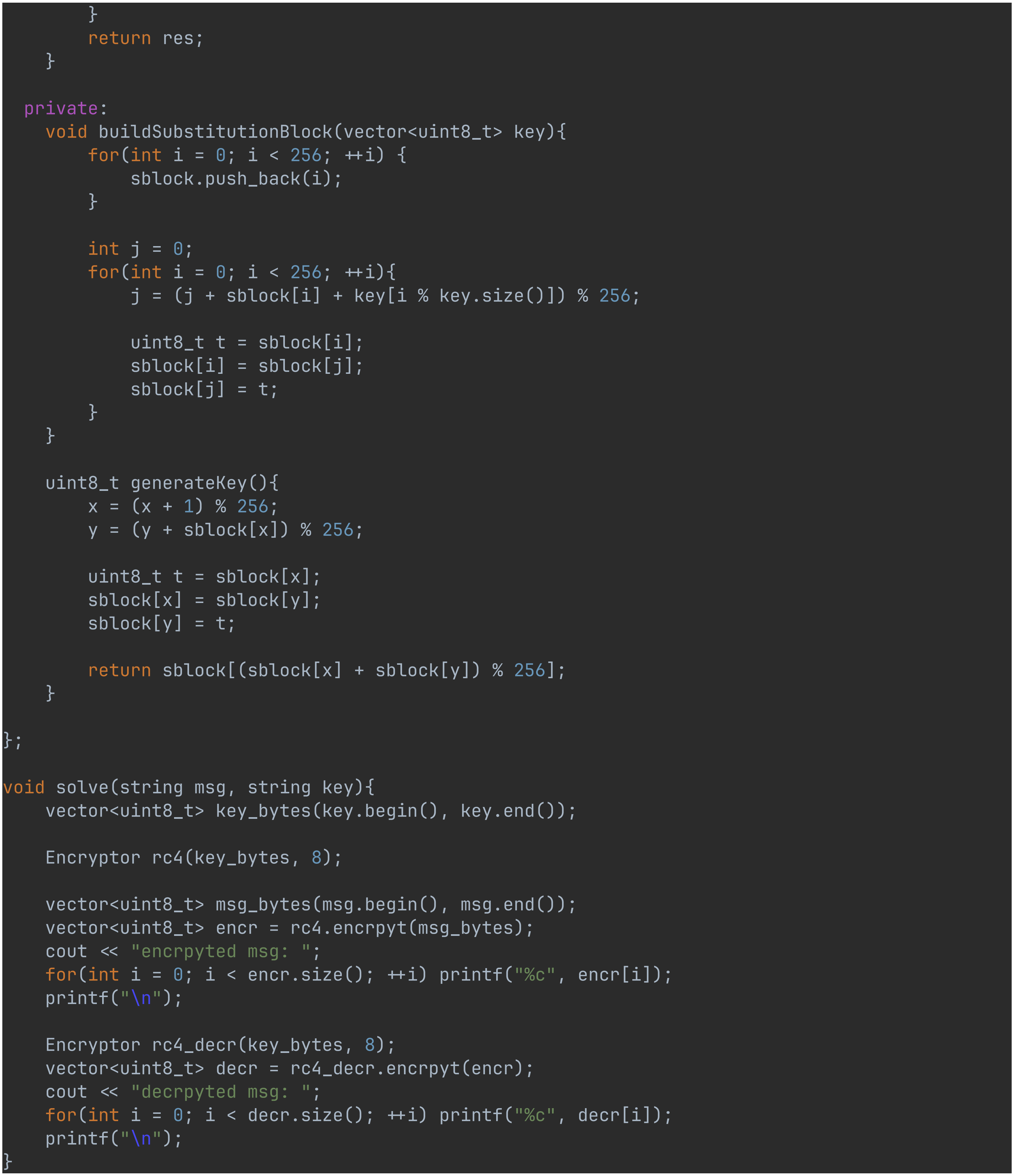
1. Сначала генерируется s-block (блок перестановки) на основе ключа. Далее этот массив позволит генерировать ключевой поток.
2. Выполняется операция суммирования по модулю 2 (xor) байтов ключевого потока и байтов исходного текста.

Алгорит дешифрования абсолютно идентичен: снова генерируется ключевой поток и выполеняется xor.

Решение задачи

Метод можно перевести в код на C++ и выполнить шифрование и дешифрование сообщения.

```cpp
#include <cstdint>
#include <string>
#include <vector>
#include <cmath>
#include <algorithm>
#include <queue>
#include <set>
#include <map>
#include <iostream>
#include <stack>
#include <iomanip>
#include <unordered_set>
#include <ctime>
#include <cassert>
#include <random>
#include <chrono>
using namespace std;

void solve(string, string);

int main(int argc, char* argv[]){
    solve(argv[1], argv[2]);

}

class Encryptor {
  public:
    int n;
    uint8_t x;
    uint8_t y;
    vector<uint8_t> sblock;

    Encryptor(vector<uint8_t> key, int blockSz) {
        n = blockSz;
        x = 0;
        y = 0;
        buildSubstitutionBlock(key);
    }

    vector<uint8_t> encrpyt(vector<uint8_t> msg){

        vector<uint8_t> res;
        for(int i = 0; i < msg.size(); ++i){
            res.push_back(msg[i] ^ generateKey());
```



```cpp
        }
        return res;
    }

  private:
    void buildSubstitutionBlock(vector<uint8_t> key){
        for(int i = 0; i < 256; ++i) {
            sblock.push_back(i);
        }

        int j = 0;
        for(int i = 0; i < 256; ++i){
            j = (j + sblock[i] + key[i % key.size()]) % 256;

            uint8_t t = sblock[i];
            sblock[i] = sblock[j];
            sblock[j] = t;
        }
    }

    uint8_t generateKey(){
        x = (x + 1) % 256;
        y = (y + sblock[x]) % 256;

        uint8_t t = sblock[x];
        sblock[x] = sblock[y];
        sblock[y] = t;

        return sblock[(sblock[x] + sblock[y]) % 256];
    }

};

void solve(string msg, string key){
    vector<uint8_t> key_bytes(key.begin(), key.end());

    Encryptor rc4(key_bytes, 8);

    vector<uint8_t> msg_bytes(msg.begin(), msg.end());
    vector<uint8_t> encr = rc4.encrpyt(msg_bytes);
    cout << "encrpyted msg: ";
    for(int i = 0; i < encr.size(); ++i) printf("%c", encr[i]);
    printf("\n");

    Encryptor rc4_decr(key_bytes, 8);
    vector<uint8_t> decr = rc4_decr.encrpyt(encr);
    cout << "decrpyted msg: ";
    for(int i = 0; i < decr.size(); ++i) printf("%c", decr[i]);
    printf("\n");
}
```

Вывод программы:

```
❯ ./build/result sometext key
encrpyted msg: xYP�>
decrpyted msg: sometext

❯ ./build/result sometext key1
encrpyted msg: o�
decrpyted msg: sometext

❯ ./build/result sometext key1asdf
encrpyted msg: ?p)
decrpyted msg: sometext

❯ ./build/result ssss key
encrpyted msg: xG
decrpyted msg: ssss
```

Вывод

В ходе выполнения лабораторной работы я ознакомился с основами симметричного шфирования и изучил шифр RC4.