**МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Защита информации

Отчет по лабораторной работе № 3

Блочный шифр «Кузнечик»

Выполнил студент группы М3О-409Б-20:

Мокшин И. Н.

Проверил, Ведьманов И. С.

Москва 2023 г.

Оглавление

[1. Код программы 3](#_Toc153353627)

[2. Результат работы 8](#_Toc153353628)

# Код программы

#pragma once

#include <mpirxx.h>

#include <time.h>

#include <iostream>

#include <sstream>

typedef union {

uint64\_t q[2];

uint8\_t b[16];

} uint128\_t;

typedef struct {

uint128\_t k[10];

} kuznechik\_key;

const uint8\_t pi\_values[256] = {

0xfc, 0xee, 0xdd, 0x11, 0xcf, 0x6e, 0x31, 0x16,

0xfb, 0xc4, 0xfa, 0xda, 0x23, 0xc5, 0x4, 0x4d,

0xe9, 0x77, 0xf0, 0xdb, 0x93, 0x2e, 0x99, 0xba,

0x17, 0x36, 0xf1, 0xbb, 0x14, 0xcd, 0x5f, 0xc1,

0xf9, 0x18, 0x65, 0x5a, 0xe2, 0x5c, 0xef, 0x21,

0x81, 0x1c, 0x3c, 0x42, 0x8b, 0x01, 0x8e, 0x4f,

0x05, 0x84, 0x02, 0xae, 0xe3, 0x6a, 0x8f, 0xa0,

0x06, 0x0b, 0xed, 0x98, 0x7f, 0xd4, 0xd3, 0x1f,

0xeb, 0x34, 0x2c, 0x51, 0xea, 0xc8, 0x48, 0xab,

0xf2, 0x2a, 0x68, 0xa2, 0xfd, 0x3a, 0xce, 0xcc,

0xb5, 0x70, 0x0e, 0x56, 0x08, 0x0c, 0x76, 0x12,

0xbf, 0x72, 0x13, 0x47, 0x9c, 0xb7, 0x5d, 0x87,

0x15, 0xa1, 0x96, 0x29, 0x10, 0x7b, 0x9a, 0xc7,

0xf3, 0x91, 0x78, 0x6f, 0x9d, 0x9e, 0xb2, 0xb1,

0x32, 0x75, 0x19, 0x3d, 0xff, 0x35, 0x8a, 0x7e,

0x6d, 0x54, 0xc6, 0x80, 0xc3, 0xbd, 0x0d, 0x57,

0xdf, 0xf5, 0x24, 0xa9, 0x3e, 0xa8, 0x43, 0xc9,

0xd7, 0x79, 0xd6, 0xf6, 0x7c, 0x22, 0xb9, 0x03,

0xe0, 0x0f, 0xec, 0xde, 0x7a, 0x94, 0xb0, 0xbc,

0xdc, 0xe8, 0x28, 0x50, 0x4e, 0x33, 0x0a, 0x4a,

0xa7, 0x97, 0x60, 0x73, 0x1e, 0x00, 0x62, 0x44,

0x1a, 0xb8, 0x38, 0x82, 0x64, 0x9f, 0x26, 0x41,

0xad, 0x45, 0x46, 0x92, 0x27, 0x5e, 0x55, 0x2f,

0x8c, 0xa3, 0xa5, 0x7d, 0x69, 0xd5, 0x95, 0x3b,

0x07, 0x58, 0xb3, 0x40, 0x86, 0xac, 0x1d, 0xf7,

0x30, 0x37, 0x6b, 0xe4, 0x88, 0xd9, 0xe7, 0x89,

0xe1, 0x1b, 0x83, 0x49, 0x4c, 0x3f, 0xf8, 0xfe,

0x8d, 0x53, 0xaa, 0x90, 0xca, 0xd8, 0x85, 0x61,

0x20, 0x71, 0x67, 0xa4, 0x2d, 0x2b, 0x09, 0x5b,

0xcb, 0x9b, 0x25, 0xd0, 0xbe, 0xe5, 0x6c, 0x52,

0x59, 0xa6, 0x74, 0xd2, 0xe6, 0xf4, 0xb4, 0xc0,

0xd1, 0x66, 0xaf, 0xc2, 0x39, 0x4b, 0x63, 0xb6

};

const uint8\_t pi\_values\_inverse[256] = {

0xa5, 0x2d, 0x32, 0x8f, 0x0e, 0x30, 0x38, 0xc0,

0x54, 0xe6, 0x9e, 0x39, 0x55, 0x7e, 0x52, 0x91,

0x64, 0x03, 0x57, 0x5a, 0x1c, 0x60, 0x07, 0x18,

0x21, 0x72, 0xa8, 0xd1, 0x29, 0xc6, 0xa4, 0x3f,

0xe0, 0x27, 0x8d, 0x0c, 0x82, 0xea, 0xae, 0xb4,

0x9a, 0x63, 0x49, 0xe5, 0x42, 0xe4, 0x15, 0xb7,

0xc8, 0x06, 0x70, 0x9d, 0x41, 0x75, 0x19, 0xc9,

0xaa, 0xfc, 0x4d, 0xbf, 0x2a, 0x73, 0x84, 0xd5,

0xc3, 0xaf, 0x2b, 0x86, 0xa7, 0xb1, 0xb2, 0x5b,

0x46, 0xd3, 0x9f, 0xfd, 0xd4, 0x0f, 0x9c, 0x2f,

0x9b, 0x43, 0xef, 0xd9, 0x79, 0xb6, 0x53, 0x7f,

0xc1, 0xf0, 0x23, 0xe7, 0x25, 0x5e, 0xb5, 0x1e,

0xa2, 0xdf, 0xa6, 0xfe, 0xac, 0x22, 0xf9, 0xe2,

0x4a, 0xbc, 0x35, 0xca, 0xee, 0x78, 0x05, 0x6b,

0x51, 0xe1, 0x59, 0xa3, 0xf2, 0x71, 0x56, 0x11,

0x6a, 0x89, 0x94, 0x65, 0x8c, 0xbb, 0x77, 0x3c,

0x7b, 0x28, 0xab, 0xd2, 0x31, 0xde, 0xc4, 0x5f,

0xcc, 0xcf, 0x76, 0x2c, 0xb8, 0xd8, 0x2e, 0x36,

0xdb, 0x69, 0xb3, 0x14, 0x95, 0xbe, 0x62, 0xa1,

0x3b, 0x16, 0x66, 0xe9, 0x5c, 0x6c, 0x6d, 0xad,

0x37, 0x61, 0x4b, 0xb9, 0xe3, 0xba, 0xf1, 0xa0,

0x85, 0x83, 0xda, 0x47, 0xc5, 0xb0, 0x33, 0xfa,

0x96, 0x6f, 0x6e, 0xc2, 0xf6, 0x50, 0xff, 0x5d,

0xa9, 0x8e, 0x17, 0x1b, 0x97, 0x7d, 0xec, 0x58,

0xf7, 0x1f, 0xfb, 0x7c, 0x09, 0x0d, 0x7a, 0x67,

0x45, 0x87, 0xdc, 0xe8, 0x4f, 0x1d, 0x4e, 0x04,

0xeb, 0xf8, 0xf3, 0x3e, 0x3d, 0xbd, 0x8a, 0x88,

0xdd, 0xcd, 0x0b, 0x13, 0x98, 0x02, 0x93, 0x80,

0x90, 0xd0, 0x24, 0x34, 0xcb, 0xed, 0xf4, 0xce,

0x99, 0x10, 0x44, 0x40, 0x92, 0x3a, 0x01, 0x26,

0x12, 0x1a, 0x48, 0x68, 0xf5, 0x81, 0x8b, 0xc7,

0xd6, 0x20, 0x0a, 0x08, 0x00, 0x4c, 0xd7, 0x74

};

const uint8\_t l\_coef[16] = {

0x94, 0x20, 0x85, 0x10, 0xC2, 0xC0, 0x01, 0xFB,

0x01, 0xC0, 0xC2, 0x10, 0x85, 0x20, 0x94, 0x01

};

// Умножение в поле Галуа GT(2^8)

uint8\_t multiply\_in\_gf\_256(uint8\_t x, uint8\_t y)

{

uint8\_t z;

z = 0;

while (y) {

if (y & 1)

z ^= x;

x = (x << 1) ^ (x & 0x80 ? 0xC3 : 0x00);

y >>= 1;

}

return z;

}

void linear(uint128\_t\* w)

{

int i, j;

uint8\_t x;

for (j = 0; j < 16; j++) {

x = w->b[15];

for (i = 14; i >= 0; i--) {

w->b[i + 1] = w->b[i];

x ^= multiply\_in\_gf\_256(w->b[i], l\_coef[i]);

}

w->b[0] = x;

}

}

void inverse\_linvear(uint128\_t\* w)

{

int i, j;

uint8\_t x;

for (j = 0; j < 16; j++) {

x = w->b[0];

for (i = 0; i < 15; i++) {

w->b[i] = w->b[i + 1];

x ^= multiply\_in\_gf\_256(w->b[i], l\_coef[i]);

}

w->b[15] = x;

}

}

void set\_encryption\_key(kuznechik\_key\* kuz, const uint8\_t key[32])

{

int i, j;

uint128\_t c, x, y, z;

for (i = 0; i < 16; i++) {

x.b[i] = key[i];

y.b[i] = key[i + 16];

}

kuz->k[0].q[0] = x.q[0];

kuz->k[0].q[1] = x.q[1];

kuz->k[1].q[0] = y.q[0];

kuz->k[1].q[1] = y.q[1];

for (i = 1; i <= 32; i++) {

// C Value

c.q[0] = 0;

c.q[1] = 0;

c.b[15] = i; // load round in lsb

linear(&c);

z.q[0] = x.q[0] ^ c.q[0];

z.q[1] = x.q[1] ^ c.q[1];

for (j = 0; j < 16; j++)

z.b[j] = pi\_values[z.b[j]];

linear(&z);

z.q[0] ^= y.q[0];

z.q[1] ^= y.q[1];

y.q[0] = x.q[0];

y.q[1] = x.q[1];

x.q[0] = z.q[0];

x.q[1] = z.q[1];

if ((i & 7) == 0) {

kuz->k[(i >> 2)].q[0] = x.q[0];

kuz->k[(i >> 2)].q[1] = x.q[1];

kuz->k[(i >> 2) + 1].q[0] = y.q[0];

kuz->k[(i >> 2) + 1].q[1] = y.q[1];

}

}

}

void encrypt\_block(kuznechik\_key\* key, uint64\_t\* block)

{

int i, j;

uint128\_t x;

x.q[0] = (block)[0];

x.q[1] = (block)[1];

for (i = 0; i < 9; i++) {

x.q[0] ^= key->k[i].q[0];

x.q[1] ^= key->k[i].q[1];

for (j = 0; j < 16; j++)

x.b[j] = pi\_values[x.b[j]];

linear(&x);

}

(block)[0] = x.q[0] ^ key->k[9].q[0];

(block)[1] = x.q[1] ^ key->k[9].q[1];

}

void decrypt\_block(kuznechik\_key\* key, uint64\_t\* block)

{

int i, j;

uint128\_t x;

x.q[0] = (block)[0] ^ key->k[9].q[0];

x.q[1] = (block)[1] ^ key->k[9].q[1];

for (i = 8; i >= 0; i--) {

inverse\_linvear(&x);

for (j = 0; j < 16; j++)

x.b[j] = pi\_values\_inverse[x.b[j]];

x.q[0] ^= key->k[i].q[0];

x.q[1] ^= key->k[i].q[1];

}

(block)[0] = x.q[0];

(block)[1] = x.q[1];

}

void lr3(std::string str) {

std::vector<uint64\_t\*> blocks;

for (size\_t i = 0; i < str.size(); i++)

{

blocks.push\_back(new uint64\_t(str[i]));

}

gmp\_randclass r(gmp\_randinit\_default);

r.seed(time(NULL));

uint8\_t key[32];

for (size\_t i = 0; i < 32; i++)

{

mpz\_class random\_number = r.get\_z\_bits(8);

key[i] = random\_number.get\_ui();

}

auto subkeys = new kuznechik\_key();

set\_encryption\_key(subkeys, key);

std::cout << "Message: " << str << std::endl;

for (auto block : blocks)

{

encrypt\_block(subkeys, block);

}

for (auto block : blocks)

{

decrypt\_block(subkeys, block);

}

std::stringstream ss;

for (auto block : blocks)

{

ss << (char)(\*block);

}

std::cout << "Decrypted message: " << ss.str() << std::endl;

}

# Результат работы

Message: This is very long test string.

Decrypted message: This is very long test string.

Сообщение было успешно зашифровано/расшифровано по блокам размером типа uint64\_t, т.е. 64 бита.