

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Информационная безопасность систем и технологий»

Отчет

по лабораторной работе №4

на тему «Программная реализация алгоритма блочного шифра »

Дисциплина: МиСКЗИ

Группа: 21ПИ1

Выполнил: Гусев Д. А.

Количество баллов:

Дата сдачи:

Принял: Липилин О. В.

2024

1 Цель работы: получение навыков программной реализации блочного шифра Магма.

2 Задание на лабораторную работу.

2.1 Программно реализовать блочный шифр по ГОСТ Р 34.12-2015 с размером блока 64 бита в режиме простой замены по ГОСТ Р 34.13-2015 в соответствии со следующими требованиями:

- в программе должна быть реализована процедура ввода ключа (ключ шифрования должен считываться из отдельно сформированного бинарного файла с расширением .key);

- ключ должен заранее формироваться с использованием программного генератора случайных чисел, реализованного ранее;

- размер ключа шифрования 56 бит, в программе должна отсутствовать возможность ввода ключа другого размера;

- развертывание ключа размером 56 бит до размера в 256 бит осуществляется повторением ($4 \cdot 56 +$ старшие 4 байта ключа);

- в программе должен быть предусмотрен контроль срока действия ключа шифрования – при зашифровании более 10 Кбайт открытого текста должно выводиться предупреждение о необходимости смены ключа зашифрования; программа не должна допускать возможность зашифрования на одном ключе более 20 Кбайт открытого текста;

- дополнение блока открытого текста выполняется в соответствии с процедурой 2 по ГОСТ Р 34.13-2015 (в младшие разряды дописывается бит «1», затем биты «0»);

- результат зашифрования записывается в бинарный файл с расширением .enc; - результат расшифрования записывается в бинарный файл с расширением .txt.

3 Выполнение лабораторную работы:

3.1 На основе ранее разработанных модулей генератора ПСП [*generator.cpp*](#), а также с помощью модуля ввода-вывода [*io.cpp*](#) была создана программа генерации ключей [*keygen.cpp*](#).

3.2 Для развертывания ключа и удобного получения раундовых ключей был написан класс RoundKey.

Конструктор класса получает на вход путь к файлу с ключем и если количество байт в файле не равняется 56 битам (7 байтам) вызывает исключение *range_error("The key must be 56 bits in size")*. 7 байт ключа хранятся в атрибуте *vector<uint8_t> bytes*.

Также в классе реализован единственный метод получения раундового ключа по индексу раунда *uint32_t operator[](int index) const*. В классе реализован подсчет использования ключа (для блока *uint64_t* на 1Кб информации ключ вызывается 4096 раз). Метод извлекает по 4 байта из вектора *bytes* по маске *bytes[(index * 4 + i) % 7]*, где *i* — номер байта (0, 1, 2, 3), *index* — индекс раундового ключа. Затем конвертирует извлеченные байты в число *uint32_t*. Если *index* больше 23, метод возвращает раундовые ключи в обратном порядке. Тестирование класса представлено на рисунке 1. Результат тестирования представлен в таблице 1. Код класса и программы, реализующей функционал шифра Магма представлен [*в Приложении Г*](#).

```

D:\Projects\PGU\6s-MiSKZI-Li X + v
key: 0xDA 0xE8 0x86 0xD6 0x58 0x32 0x4C

Round 1 : 0xDAE886D6
Round 2 : 0x58324CDA
Round 3 : 0xE886D658
Round 4 : 0x324CDAE8
Round 5 : 0x86D65832
Round 6 : 0x4CDAE886
Round 7 : 0xD658324C
Round 8 : 0xDAE886D6

Round 9 : 0x58324CDA
Round 10 : 0xE886D658
Round 11 : 0x324CDAE8
Round 12 : 0x86D65832
Round 13 : 0x4CDAE886
Round 14 : 0xD658324C
Round 15 : 0xDAE886D6
Round 16 : 0x58324CDA

Round 17 : 0xE886D658
Round 18 : 0x324CDAE8
Round 19 : 0x86D65832
Round 20 : 0x4CDAE886
Round 21 : 0xD658324C
Round 22 : 0xDAE886D6
Round 23 : 0x58324CDA
Round 24 : 0xE886D658

Round 25 : 0xDAE886D6
Round 26 : 0xD658324C
Round 27 : 0x4CDAE886
Round 28 : 0x86D65832
Round 29 : 0x324CDAE8
Round 30 : 0xE886D658
Round 31 : 0x58324CDA
Round 32 : 0xDAE886D6
Для продолжения нажмите любую клавишу . . . |

```

Рисунок 1 — Тестирование получения раундовых ключей

Таблица 1 — Раундовые ключи

key: 0xDA 0xE8 0x86 0xD6 0x58 0x32 0x4C			
Round 1 – 8	Round 9 – 16	Round 17 – 24	Round 25 – 32
0xDAE886D6	0x58324CDA	0xE886D658	0xDAE886D6
0x58324CDA	0xE886D658	0x324CDAE8	0xD658324C
0xE886D658	0x324CDAE8	0x86D65832	0x4CDAE886

0x324CDAE8	0x86D65832	0x4CDAE886	0x86D65832
0x86D65832	0x4CDAE886	0xD658324C	0x324CDAE8
0x4CDAE886	0xD658324C	0xDAE886D6	0xE886D658
0xD658324C	0xDAE886D6	0x58324CDA	0x58324CDA
0xDAE886D6	0x58324CDA	0xE886D658	0xDAE886D6

3.3 Были протестированы функции расшифрования и зашифрования. Результаты представлены на рисунках 2 — 7.

```

Windows PowerShell
Для продолжения нажмите любую клавишу . . .
PS D:\Projects\PGU\6s-MiSKZI-Lipilin\4LB\source> ./main
Enter 'encrypt' for encryption or 'decrypt' for decryption: decrypt
Enter the file name: encrypt-1byte.enc
Для продолжения нажмите любую клавишу . . .

D:\Projects\PGU\6s-MiSKZI-Lipilin\4LB\source\files\decrypt.txt - Notepad++
Файл Правка Поиск Вид Кодировки Синтаксис Опции Инструменты Макросы Запуск Плагины Вкладки ?
decrypt.txt
1 Yb-4X-KXJwE7#* %j*,S1a0S.A9*8bVTP°Afхђй-ПдсУК€CANf]*yY SY0вEOTюё[ж°PdчDLErKвl° Kк(<zRSJ,иJ№ё?g*sn+
2 %4 у.хщЕ2ГТDС12ь°с1ьS0°9ӨWӨ°ньи.wwmNUL0ГNj3е(ђблK*сJSYU,ънЕ! чГ9°EW°j6EXjc1L°5nBNUL*
3 чй
4 щBEL(ES2о&NULbXI))уПл60:‘,‘,ф’)9@DС1фдби GS«щг$>ёёбRXI{SprчSTXb4Ib»1ГH.äv,FFXH;ETBS16мсdёX€g-бyачкC
5 чSYNh:EOE!:-USMb1nёNUL†8%и(ывAXыб°‘§DLENUL°°°Tёf €eP[с€X)ЭуУд€T†jшбёACK6P\A0°шS0xS@DС3°X°4эto;ЙUJ4
6 bGsf1Xe6 ё~,8лGH5DС1йvLfbL н;Г2ЙD-5ETBfv6fbL н;Г26D-5ETBfv6fbL н;Г2сбGS2@BELf3_b.$юYв4€cm05DLE:H8u8!
7 мJ5HaE04б:28u8$
8 мJ5Ha
9 02ђ:1:38
10 лГ5?Д-7F†^6Xи& Pю,04U•1@DС3NOнЮ. F<Г42б51B{N5нЮ. F<Г40б-1FfN6nv.$FbJ4...
11 42°Z1Sjux sF,7чU•1EnNDиЯ& EYГ0х&}1D(ETBN0иЯ& EYГ0yбGS1@BELN3иЯ& EYГ0иDu5DLEyv4ESQ±| |^,7QCme@BS|Hнб-
12 eG€|1лў-‘Йму<$ESu8т-q7w,,p ;ю,g†GHe@!;Lн&V юУГаёD-eFG†бн&v юУГаbD5еB+†5и9V!юУГа°4ugB°37II«6сVTZd0°hPYb
13 *I€<рХТрVInsyTuyIYT:ГeC>бЦ?MECшГшS.SYN°ё~jAfKj ЧCT+j>MA°°ГNAKTЮ°SёсCAXбQDLErHбл°°HACK(Iz3p,he 0IT4jC
14 иш&яHёis 9Y|KCANHлГФHиЩКи°YЛr†PСУBТiФ=«ЮХьYГ°OK±XoBSDLBfГ°x yоN°S†Mшsла(T8:сI†PьUESCдёбje•eDGS<Yг0
15 (Qq>6EB†IY4,q°Yй56LSTXpaabJ$,ENUL)0CANл3мEN0)_№t:7сX{[EM†°GcpDС1()нYСӨ†0еSYN(†°мEЛ0т,i-фQACK.ЧDС2/
16 °ф...°[°шN°5.^h&iяi)/н 5IbI
17 xЖ°*KuE9,.
18 ы3]mSUBfcG<MSTXIш9ьexKXTи2RrEглмье°;mfESQAo WGN%I6B №Гd<сИ2bђk7Etq%-Y6уьu2IzDС146STXb2RSBELNULg
19 юDС3U8DuKэZ8T°DС1 !-ruP-!y9°n:6/SYNi5o!°Hu,jgъaVT UAmVTюDС3U8DuKэ]ETDС1{EW-bCANё(+чс#9.q4Pb °l
20 юDС3U8DuKэZиT2DС1{ -eST.ASUB. Iифю00>=d€pLHKIjбъaVT FAmVTюDС3U8DuKэ°BT0DС1{!-1Pa)DС1°$°шщSOH7ENObV
21 jBиit=TX«°K°(сrФoBSDLBfГ°x yоN°S†Mшsк6(C8:щLлEBGT°b°FdTTUoP1Цъ°°,№°ф?<°°,»ор,CANMCOH
22 ENU2ёvIUJ)(SUBC$;ag@eNUL93STXjю(0_ў)DLEГ60еe8YQ...pcJf)X|M
23 алђh2]W$°фэSYN°wSOH(NULЦCФл3юю°°хJCANPSjђsеN€Tя5Yё&Юбм<h(еe4Б,»b(T
24 *I€<рХТрVInsyTuyIYT:ГeC>бЦ?MECшГшS.SYN°ё~jAfKj ЧCT+j>MA°°ГNAKT_eS&iю€Xя5Yё&Юбм;8(се!И,$4(Ya1CANIa)°
25 ETBPI°@†vөtp6,UswYBSTP1cENODС1,|KQccBSb1и0i9YACKFq5M37Fyђи,BELD00PX;]«бB°,jб3BLGS>SO°3SYN
26 -2е°NUL4.GxE\6рpFMб §иe(;&ђёю<6EOTсBS8ь:03XДиe,°JdђA&1>J)EOTбГGr jX°&юBел°EWVT4°SOHNUL3)€CANзиCAI
27 GS14snZ0RFFq7°°26USЭ;HМ°rFFP1DС3&ђhK,°^ETDС2{EW-†°P°C!e°DMueб6т°brLH 2j2шс<Zёь
28 ETBPI°@†vөtpBS6°UsaYFF-P3
29 L
Normal text file length: 5456 lines: 38 Ln: 2 Col: 73 Pos: 241 Windows (CR LF) ANSI INS

```

Рисунок 2— Результат расшифрования удаления 1 байта данных из шифртекста


```
Windows PowerShell
Для продолжения нажмите любую клавишу . . .
PS D:\Projects\PGU\6s-MiSKZI-Lipilin\4LB\source> ./main.exe
Enter 'encrypt' for encryption or 'decrypt' for decryption: decrypt
Enter the file name: encrypt+8bytes.enc
Для продолжения нажмите любую клавишу . . .

D:\Projects\PGU\6s-MiSKZI-Lipilin\4LB\source\files\decrypt.txt - Notepad++
Файл  Правка  Поиск  Вид  Кодировки  Синтаксисы  Опции  Инструменты  Макросы  Запуск  Плагины  Вкладки  ?
decrypt.txt
1  Rp
2  A.2 Алгоритм блочного шифрования с длиной блока n = 64 бит
3  ÷.2.1 Преобразование t
4  t(fdb97531) = 2a196f34,
5  t(2a196f34) = ebd96059,
6
7  t(ebd9f03a) = b039bb3d,
8  t(b039bb3d) = 68695433.
9  A.2.2 Преобразование g
10
11  g[87654321](fedcba98) = fdcba20c,
12  g[fdcba20c](87654321) = 7e791a4b,
13  g[7e791a4b](fdcba20c) = c76549ec,
14  g[c76549ec](7e791a4b) = 9791c85.
15
16  E.2.3 Алгоритм развертывания ключа
17  В настоящем контрольном примере ключ имеет значение:
18  K = ffeeddccbbaa99c7766254433221100f0f1f2f3f4f5f6f7f8f9fafbfcdfeffv.
19
20  Итерационные ключи Ki, i = 1, 2, ..., 32, принимают следующие значения:
21
22  K1 = ffeeddccc, K9 = ffeeddccc, K17 = ffeeddccc, K25 = fcdfeff,
23  K2 = bbaa9988, K10 = bbaa9988, K18 = bbaa9988, K26 = f8f9fafb,
24  K3 = 77665544, K11 = 77665544, K19 = 77665544, K27 = f4f5f6f7,
25  K4 = 33221100, K12 = 33221100, K20 = 33221100, K28 = f0f1f2f3,
26  K5 = f0f1f2f3, K13 = f0f1f2f3, K21 = f0f1f2f3, K29 = 33221100,
27  K6 = f4f5f6f7, K14 = f4f5f6f7, K22 = f4f5f6f7, K30 = 77665544,
28  K7 = f8f9fafb, K15 = f8f9fafb, K23 = f8f9fafb, K31 = bbaa9988,
29  K8 = fcdfeff, K16 = fcdfeff, K24 = fcdfeff, K32 = ffeeddccc.
```

Рисунок 5 — Результат расшифрования добавления блока данных в шифртекст, размер которого кратен 64 битам;

```
Windows PowerShell
Для продолжения нажмите любую клавишу . . .
PS D:\Projects\PGU\6s-MiSKZI-Lipilin\4LB\source> ./main.exe
Enter 'encrypt' for encryption or 'decrypt' for decryption: decrypt
Enter the file name: encrypt_sub.enc
Для продолжения нажмите любую клавишу . . .

D:\Projects\PGU\6s-MiSKZI-Lipilin\4LB\source\files\decrypt.txt - Notepad++
Файл  Правка  Поиск  Вид  Кодировки  Синтаксисы  Опции  Инструменты  Макросы  Запуск  Плагины  Вкладки  ?
decrypt.txt
1  ГОСТ Р 34.12 - 2015
2  A.2 Алгоритм блочного шифрования с длиной блока n = 64 бит
3  ÷.2.1 Преобразование t
4  t(fdb97531) = 2a196f34,
5  t(2a196f34) = ebd96059,
6
7  t(ebd9f03a) = b039bb3d,
8  t(b039bb3d) = 68695433.
9  A.2.2 Преобразование g
10
11  g[87654321](fedcba98) = fdcba20c,
12  g[fdcba20c](87654321) = 7e791a4b,
13  g[7e791a4b](fdcba20c) = c76549ec,
14  g[c76549ec](7e791a4b) = 9791c85.
15
16  E.2.3 Алгоритм развертывания ключа
17  В настоящем контрольном примере ключ имеет значение:
18  K = ffeeddccbbaa99c7766254433221100f0f1f2f3f4f5f6f7f8f9fafbfcdfeffv.
19
20  Итерационные ключи Ki, i = 1, 2, ..., 32, принимают следующие значения:
21
22  K1 = ffeeddccc, K9 = ffeeddccc, K17 = ffeeddccc, K25 = fcdfeff,
23  K2 = bbaa9988, K10 = bbaa9988, K18 = bbaa9988, K26 = f8f9fafb,
24  K3 = 77665544, K11 = 77665544, K19 = 77665544, K27 = f4f5f6f7,
25  K4 = 33221100, K12 = 33221100, K20 = 33221100, K28 = f0f1f2f3,
26  K5 = f0f1f2f3, K13 = f0f1f2f3, K21 = f0f1f2f3, K29 = 33221100,
27  K6 = f4f5f6f7, K14 = f4f5f6f7, K22 = f4f5f6f7, K30 = 77665544,
28  K7 = f8f9fafb, K15 = f8f9fafb, K23 = f8f9fafb, K31 = bbaa9988,
29  K8 = fcdfeff, K16 = fcdfeff, K24 = fcdfeff, K32 = ffeeddccc.
```

Рисунок 6 — Результат расшифрования перестановки двух блоков данных шифртекста, размер которых кратен 64 битам;

Приложение А

Код программы io.cpp

```
#include <iostream>
#include <fstream>
#include <vector>
#include <iomanip>

using namespace std;

class Type
{
private:
    string type;
public:
    Type(const string type)
    {
        this->type = type;
    }
    operator string() const
    {
        return type;
    }
    bool operator==(const Type &type) const
    {
        return this->type == string(type);
    }
};

class By
{
public:
    static Type HEX;
    static Type BIN;
    static Type DEC;
    static Type STR;
```

```

};

Type By::HEX = Type("HEX");
Type By::BIN = Type("BIN");
Type By::DEC = Type("DEC");
Type By::STR = Type("STR");

template <typename T>
void printBytes(const T value, const Type &type = By::DEC)
{
    uint8_t size = sizeof(T);
    if (type == By::HEX)
        cout << "0x" << setfill('0') << setw(size * 2) << uppercase << hex <<
uint64_t(value) << " ";
    else if (type == By::BIN)
    {
        cout << "0b";
        for (int i = size * 8 - 1; i >= 0; i--)
            cout << ((value >> i) & 1);
        cout << " ";
    }
    else if (type == By::DEC)
        cout << dec << uint64_t(value) << " ";
    else if (type == By::STR)
    {
        for (int i = size - 1; i >= 0; --i)
            cout << static_cast<char>((value >> (8 * i)) & 0xFF);
        cout << " ";
    }
}

template <typename T>
void printBytes(const vector<T> values, const Type &type = By::DEC)
{
    for (int i = 0; i < values.size(); i++)
    {
        if (i * sizeof(values[i]) >= 16)
            cout << endl;
    }
}

```

```

        printBytes(values[i], type);
    }
}

template <typename T>
vector<T> convert(const vector<uint8_t> &bytes)
{
    vector<T> result;
    for (size_t i = 0; i < bytes.size(); i += sizeof(T))
    {
        T value = 0;
        for (size_t j = 0; j < sizeof(T) && i + j < bytes.size(); ++j)
        {
            value |= static_cast<T>(bytes[i + j]) << (8 * (sizeof(T) - 1 - j));
        }
        result.push_back(value);
    }
    return result;
}

template <typename T>
vector<uint8_t> convert(const vector<T> &values)
{
    vector<uint8_t> bytes;
    for (const T &value : values)
    {
        for (int i = sizeof(T) - 1; i >= 0; --i)
            bytes.push_back(static_cast<uint8_t>(value >> (8 * i)));
    }
    return bytes;
}

template <typename T>
void writeBytes(const string &path, const vector<T> &values)
{
    ofstream file(path, ios::binary);
    if (!file)

```

```

        throw runtime_error(path);
    vector<uint8_t> bytes = convert(values);
    file.write(reinterpret_cast<const char *>(bytes.data()), bytes.size());
    file.close();
}

template <typename T>
vector<T> readBytes(const string &path)
{
    ifstream file(path, ios::binary);
    if (!file)
        throw runtime_error(path);
    vector<uint8_t> bytes((istreambuf_iterator<char>(file)),
                          istreambuf_iterator<char>());
    return convert<T>(bytes);
}

```

Приложение Б

Код программы generator.cpp

```
#include <vector>
#include <numeric>
#include <bitset>
using namespace std;
class Generator
{
private:
    /* Инициализация полиномов */
    bitset<128> firstPolynomial;
    bitset<128> secondPolynomial;

    /* Инициализация дефолтных значений */
    int ids[6] = {13, 16, 17, 100, 110, 111};
public:
    /* Конструктор */
    Generator(long long int firstSeed,
              long long int secondSeed)
    {
        firstPolynomial = bitset<128>(firstSeed);
        secondPolynomial = bitset<128>(secondSeed);
        secondPolynomial <<= ids[5] / 2;
    }
    bool getBit()
    {
        /* Получение суммы по модулю 2 для полиномов */
        bool firstSum = firstPolynomial[ids[0]] ^ firstPolynomial[ids[1]];
        bool secondSum = firstPolynomial[ids[3]] ^ firstPolynomial[ids[4]];

        /* Сдвигаем полиномы */
        firstPolynomial <<= 1;
        secondPolynomial <<= 1;

        /* Устанавливаем нулевые биты как соответствующую сумму */
```

```

firstPolynomial[0] = firstSum;
secondPolynomial[0] = secondSum;

/* Обрезаем полиномы по их размеру
this->firstPolynomial &= (1ULL << ids[2]) - 1;
this->secondPolynomial &= (1ULL << ids[5]) - 1;
return firstSum ^ secondSum;
}
uint8_t getByte()
{
    uint8_t byte = 0;
    for (int i = 0; i < 8; i++)
    {
        byte <= 1;
        byte |= getBit();
    }
    return byte;
}
vector<bool> bitSequence(int length)
{
    vector<bool> sequence;
    for (int i = 0; i < length; i++)
        sequence.push_back(getBit());
    return sequence;
}
vector<uint8_t> byteSequence(int length)
{
    vector<uint8_t> bytes;
    for (int i = 0; i < length; ++i)
        bytes.push_back(getByte());
    return bytes;
}
};

```

Приложение В

Код программы keygen.cpp

```
#include <iostream>
#include <fstream>
#include "../common/io.cpp"
#include "../common/generator.cpp"

using namespace std;

int main()
{
    long long int firstSeed, secondSeed;
    cout << "\n# ----- Start keygen ----- #\n";
    cout << "\nEnter integers seeds to generate the key:\nEnter first seed: ";
    cin >> firstSeed;
    cout << "Enter second seed: ";
    cin >> secondSeed;

    Generator generator(firstSeed, secondSeed);
    vector<uint8_t> key = generator.byteSequence(7);
    writeBytes("../files/key.key", key);
    cout << "The key has been successfully generated and saved to the key.key file\n";
    cout << "Your key: ";
    printBytes(key, By::HEX);
    system("pause");
    return 0;
}
```

Приложение Г

Код класса RoundKey

```
#include <iostream>
#include <fstream>
#include <vector>
#include "../common/io.cpp"

using namespace std;

vector<uint8_t> S[8] = {
    {12, 4, 6, 2, 10, 5, 11, 9, 14, 8, 13, 7, 0, 3, 15, 1},
    {6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15},
    {11, 3, 5, 8, 2, 15, 10, 13, 14, 1, 7, 4, 12, 9, 6, 0},
    {12, 8, 2, 1, 13, 4, 15, 6, 7, 0, 10, 5, 3, 14, 9, 11},
    {7, 15, 5, 10, 8, 1, 6, 13, 0, 9, 3, 14, 11, 4, 2, 12},
    {5, 13, 15, 6, 9, 2, 12, 10, 11, 7, 8, 1, 4, 3, 14, 0},
    {8, 14, 2, 5, 6, 9, 1, 12, 15, 4, 11, 0, 13, 10, 3, 7},
    {1, 7, 14, 13, 0, 5, 8, 3, 4, 15, 10, 6, 9, 12, 11, 2}};

template <typename T>
T substitute(const T &value)
{
    T replaced = 0;
    for (int i = 0; i < sizeof(value); i++)
    {
        uint8_t bits = (value >> (i * 4)) & 0xF;
        replaced |= (S[i][bits] << (i * 4));
    }
    return replaced;
}

template <typename T>
T cycle_shift(const T &value, const int &shift)
{
    const size_t bits = sizeof(T) * 8;
    if (shift < 0)
        return (value << -shift) | (value >> (bits + shift));
    return (value >> shift) | (value << (bits - shift));
}

class RoundKey
{
private:
    vector<uint8_t> bytes;
    mutable uint32_t counter = 0;
    uint32_t sync = 4096; // Количество вызовов RoundKey[] на 1 Кб данных

public:
    RoundKey(const string &path)
    {
        bytes = readBytes<uint8_t>(path);
        if (bytes.size() != 7)
```



```

        throw range_error("\nThe key must be 56 bits in size\n");
};

uint32_t operator[](int index) const
{
    if (index < 0 || index > 31)
        throw range_error("\nThe index must be in range [0, 32)\n");

    counter += 1;
    if (counter == sync * 10)
        cout << "\nWarning!!! The key is about to expire!\n";
    if (counter >= sync * 20)
        throw length_error("\nThe validity period of the key has expired.\n");

    if (index > 23)
        index = 31 - index;
    uint32_t nkey = 0;
    for (int i = 0; i < 4; i++)
        nkey |= (bytes[(index * 4 + i) % 7] << (3 - i) * 8);
    return nkey;
}
};

uint64_t crypt(const uint32_t &xkey, const uint64_t &block)
{
    uint32_t N1 = block & 0xFFFFFFFF; // Правые 32 бита блока
    uint32_t N2 = block >> 32;        // Левые 32 бита блока
    uint32_t N1s = N1 + xkey;          // Сложение с ключем
    uint32_t R = substitute(N1s);      // Подстановка
    uint32_t Rs = cycle_shift(R, -11); // Перестановка
    uint32_t N2s = Rs ^ N2;            // XOR
    return (uint64_t(N1) << 32) | N2s; // Объединяем N1 и N2s в одно 64-битное число
}

int main()
{
    // Чтение файла
    vector<uint64_t> data64 = readBytes<uint64_t>("./files/open.txt");

    // Зашифрование
    RoundKey key("./files/key.key");
    for (int i = 0; i < data64.size(); i++)
    {
        for (uint8_t j = 0; j < 31; j++)
            data64[i] = crypt(key[j], data64[i]);
        data64[i] = cycle_shift(crypt(key[31], data64[i]), 32);
    };
    writeBytes("./files/encrypt.enc", data64);

    // Расшифрование
    key = RoundKey("./files/key.key");
    for (int i = 0; i < data64.size(); i++)
    {
        for (int j = 31; j > 0; j--)
            data64[i] = crypt(key[j], data64[i]);
        data64[i] = cycle_shift(crypt(key[0], data64[i]), 32);
    }
    writeBytes("./files/decrypt.txt", data64);
}

```

```
    system("pause");  
    return 0;  
};
```

Приложение Д

Код программы keygen.cpp