МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования  
 «Ижевский государственный технический университет имени М.Т.Калашникова»

Факультет «Информатика и вычислительная техника»

Кафедра «Вычислительная техника»

Отчет

по лабораторной работе №2

на тему «Блочное симметричное шифрование»

по дисциплине «Методы и способы защиты компьютерной информации»

Выполнил:

Проверил:

студент группы Б08-781-1

Мушегов В.А.

к.т.н., доцент

Марков Е.М.

Ижевск 2020

**Цель работы**

Изучение структуры и основных принципов работы современных алгоритмов блочного симметричного шифрования, приобретение навыков программной реализации блочных симметричных шифров.

**Задание**

Реализовать систему симметричного блочного шифрования, позволяющую шифровать и дешифровать файл на диске с использованием RC6 шифра в СВС режиме шифрования.

**Основные сведения**

Шифр RC6 был представлен на конкурсе AES группой разработчиков во главе с известным криптографом Рональдом Ривестом. Шифр очень прост, что делает его очень привлекательным для использования в различных прикладных за-дачах. Он представляет собой сеть Фейштеля из 20 раундов с 4 ветвями смешанного типа – результат образующих функций, вычисленных от четных ветвей, накладывается на нечетные ветви, затем ветви меняются местами. Размер блока – 128 бит. Структура алгоритма приведена на рис.1.

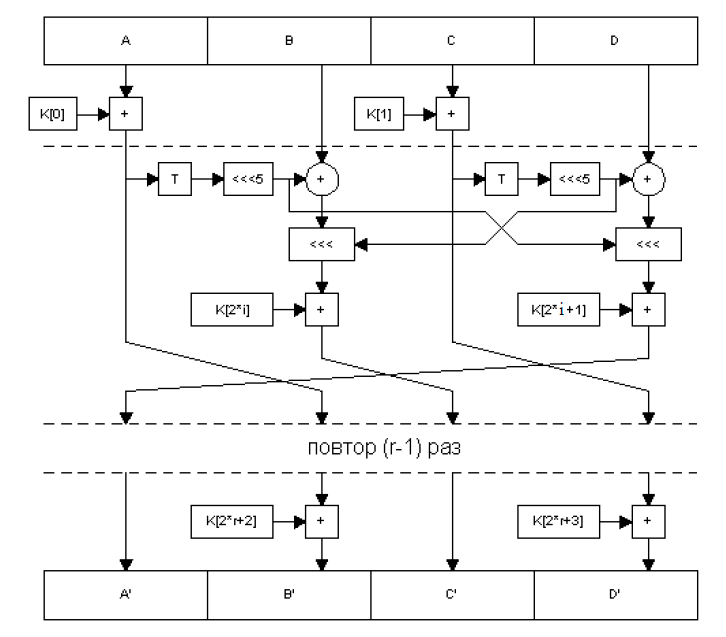


Рисунок 1 – Алгоритм шифрования RC6

Преобразование T представляет собой функцию: . Оно используется в качестве нелинейного преобразования с хорошими показателями перемешивания битового значения входной величины. В качестве параметра циклического сдвига на переменное число бит поступает результат умножения, циклически смещенный влево на 5 бит. Таким образом, реально величину переменного сдвига определяют 5 старших бит результата 32-битного умножения – а именно эти биты находятся в центре 64 битного общего (до взятия вычета) произведения, и, следовательно, зависят от всех бит входного параметра X.

Дешифрование алгоритма RC6 производится инвертированием порядка выполняемых в сети Фейштеля действий и заменой операции сложения на вычитание.

Формирование ключей раунда в RC6 происходит следующим образом. Сначала ключ, размер которого может иметь произвольную длину, выравнивается по 32-битной границе нулями. В результате получается массив K из с машинных слов K[0]..K[c-1]. Массив ключей раунда назовем k, всего потребуется 44 элемента. На первом этапе массив k заполняется с использованием специальных констант, полученных из двоичной записи чисел e (основание натурального логарифма) и φ (золотое сечение):

На втором этапе производится рекурсивное перемешивание между собой массивов *K* и *k* :

A=0;B=0;i=0;j=0;

for( x=0;x<44\*3;x++)

{

k[i]=(k[i]+A+B) ROL 3; B=k[i];

i=(i+1) %44;

K[j]=(K[j]+A+B) ROL (A+B);A=k[j];

j=(j+1)%c;

}

Троекратное рекурсивное перемешивание материала ключа с ключом при наличии сдвига на переменное количество бит существенно затрудняет крипто-анализ алгоритма.

Основное преимущество алгоритма – высокая скорость работы на 32-разрадных аппаратных платформах. Его криптостойкость определяется стойкостью к атакам его предшественника (RC5), который в свое время очень хорошо изучен криптографами. К недостатком алгоритма относят его слабую распараллеливаемость.

**Режим шифрования CBC.**

Режим CBC предполагает следующие алгоритмы шифрации/дешифрации:

В режиме CBC каждый блок открытого текста складывается с блоком шифро-текста, полученным на предыдущем этапе. Таким образом, происходит сцепление блоков друг с другом и независимая манипуляция с каждым из них невозможна, а одинаковые входные блоки будут давать на выходе разные блоки. Однако задача распараллеливания процедуры кодирования в этом режиме затруднена. Дополнительным параметром процедур шифрования/дешифрования является параметр .

**Приложение**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

#include "C:\Libraries\cryptopp820\modes.h"

#include "C:\Libraries\cryptopp820\RC6.h"

#include "C:\Libraries\cryptopp820\filters.h"

using namespace std;

using namespace CryptoPP;

byte key[RC6::DEFAULT\_KEYLENGTH], iv[RC6::BLOCKSIZE];

string readFile(const std::string& fileName)

{

ifstream f(fileName, ios::binary);

stringstream ss;

ss << f.rdbuf();

return ss.str();

}

//функция Encrypt:

std::string Encrypt(std::string plaintext)

{

CryptoPP::RC6::Encryption rc6Encryption(key, CryptoPP::RC6::DEFAULT\_KEYLENGTH);

CryptoPP::CBC\_Mode\_ExternalCipher::Encryption cbcEncryption(rc6Encryption, iv);

// Local variable !

std::string ciphertext;

CryptoPP::StreamTransformationFilter stfEncryptor(cbcEncryption, new CryptoPP::StringSink(ciphertext));

stfEncryptor.Put(reinterpret\_cast<const unsigned char\*>(plaintext.c\_str()), plaintext.length() + 1);

stfEncryptor.MessageEnd();

return ciphertext;

}

//функция Decrypt:

std::string Decrypt(std::string plaintext)

{

CryptoPP::RC6::Decryption rc6Decryption(key, CryptoPP::RC6::DEFAULT\_KEYLENGTH);

CryptoPP::CBC\_Mode\_ExternalCipher::Decryption cbcDecryption(rc6Decryption, iv);

// Local variable !

std::string decryptedtext;

CryptoPP::StreamTransformationFilter stfDecryptor(cbcDecryption, new CryptoPP::StringSink(decryptedtext));

stfDecryptor.Put(reinterpret\_cast<const unsigned char\*>(plaintext.c\_str()), plaintext.size());

stfDecryptor.MessageEnd();

return decryptedtext;

}

int main(int argc, char\* argv[])

{

memset(key, 0x00, RC6::DEFAULT\_KEYLENGTH);

memset(iv, 0x00, RC6::BLOCKSIZE);

string name\_file = "foo.txt";

string my\_file = readFile(name\_file);

string plaintext = my\_file;

string crypt = Encrypt(plaintext);

ofstream fout;

fout.open("crypt\_" + name\_file, ios::trunc | ios::binary);

fout << crypt;

fout.close();

string decrypt = Decrypt(crypt);

ofstream fout1;

fout1.open("decrypt\_" + name\_file, ios::trunc | ios::binary);

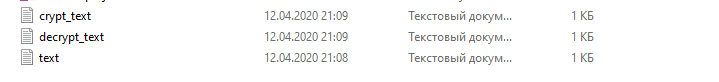
fout1 << decrypt;

fout1.close();

return 0;

}

**Результата работы программы**

****

