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

Суровцева А.С.

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 ГОСТ 28147-89 шифра в EСВ режиме шифрования.

**Основные сведения**

Отечественным стандартом блочного шифрования является алгоритм ГОСТ 28147-89, который использует 256-битный ключ шифрования и 32 цикла преобразования 64-битных блоков исходного сообщения. Алгоритм реализует классическую схему сети Фейштеля. При этом образующая функция реализована следующим образом (рис.1) .

Сначала правая половина блока и ключ раунда складываются по модулю 232. Результат сложения разбивается на восемь 4-битовых последовательностей, каждая из которых поступает на вход соответствующего S-блока. Каждый блок представляет собой таблицу подстановки, которая заменяет поступающее на вход число в диапазоне [0..15] на другое число в том же диапазоне. Выходы всех S-блоков объединяются в 32-битное слово, которое затем циклически сдвигается влево на 11 битов и объединяется с левой частью блока операцией XOR.

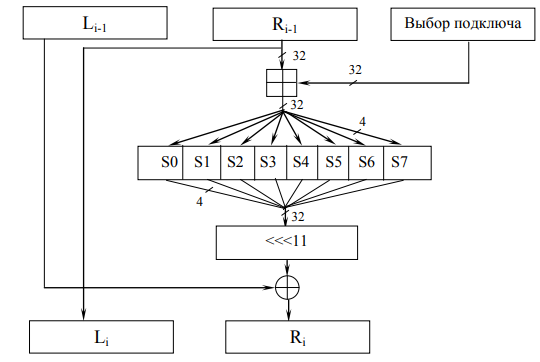


Рисунок 1 – Образующая функция алгоритма ГОСТ 28147-89

Формирование ключей раунда осуществляется по простой схеме. 256-битный ключ разбивается на восемь 32-битных машинных слов. Они нумеруются с К0 по К7. 32 ключа раунда получаются применением этих машинных слов в следующем порядке:

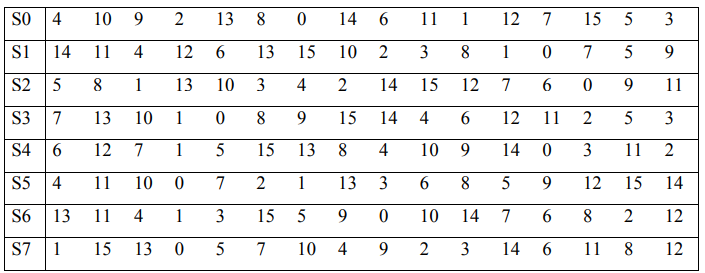


то есть в последних 8 раундах ключи подаются в обратном порядке.

Алгоритм ГОСТ является симметричным, и для дешифрации достаточно подать на вход алгоритма блоки зашифрованных сообщений и ключи раундов в порядке, обратном их следованию при шифрации.

Особенностью алгоритма является отсутствие в стандарте каких-либо рекомендаций по выбору содержимого таблиц подстановок (S-блоков). Первоначально 8\*16\*4=512 бит таблиц подстановок являлись также частью ключевой информации. Впоследствии требование к секретности содержимого таблиц было упразднено, однако статическое, обеспечивающее высокую криптостойкость алгоритма, содержимое таблиц подстановки так и не было опубликовано. Набор S-блоков, рекомендуемый стандартом хеширования ГОСТ Р 34.11-94, использующего блочный шифр ГОСТ 28147-89 в качестве основной преобразующей операции, приведен в таблице 1.

Таблица 1. S-блоки алгоритма ГОСТ 28147-89



Достоинствами алгоритма ГОСТ можно назвать простоту реализации, причем как программной, так и аппаратной, поскольку алгоритм не использует битовых перестановок, большой размер ключа делает малоперспективной силовую атаку на шифр, большое количество раундов затрудняют дифференциальный и линейный криптоанализ. Алгоритм оптимизирован под 32-разрядные процессоры. Единственной проблемой практического применения алгоритма является, как уже упоминалось, формирование таблиц подстановки.

**Приложение**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

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

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

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

using namespace std;

using namespace CryptoPP;

byte key[GOST::DEFAULT\_KEYLENGTH];

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::GOST::Encryption gostEncryption(key, CryptoPP::GOST::DEFAULT\_KEYLENGTH);

CryptoPP::ECB\_Mode\_ExternalCipher::Encryption ebcEncryption(gostEncryption, key);

// Local variable !

std::string ciphertext;

CryptoPP::StreamTransformationFilter stfEncryptor(ebcEncryption, 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::GOST::Decryption gostDecryption(key, CryptoPP::GOST::DEFAULT\_KEYLENGTH);

CryptoPP::ECB\_Mode\_ExternalCipher::Decryption ebcDecryption(gostDecryption, key);

// Local variable !

std::string decryptedtext;

CryptoPP::StreamTransformationFilter stfDecryptor(ebcDecryption, 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[])

{

setlocale(LC\_ALL, "rus");

string k, name\_file;

while (true) {

do {

system("cls");

cout << "\n Доступные операции: ";

cout << "\n1. Зашифровать файл";

cout << "\n2. Расшифровать файл";

cout << "\n0. Выйти";

cout << "\n Ваш выбор: ";

cin >> k;

} while (k.length() > 1 || k != "1" && k != "2" && k != "0");

if (k == "1") {

system("cls");

cout << "\n Введите ключь 16 символов: ";

cin >> key;

system("cls");

cout << "\n Введите название файла: ";

cin >> name\_file;

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

string crypt = Encrypt(my\_file);

ofstream fout;

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

fout << crypt;

fout.close();

}

if (k == "2") {

system("cls");

cout << "\n Введите ключь 16 символов: ";

cin >> key;

system("cls");

cout << "\n Введите название файла: ";

cin >> name\_file;

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

string decrypt = Decrypt(my\_file);

ofstream fout1;

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

fout1 << decrypt;

fout1.close();

}

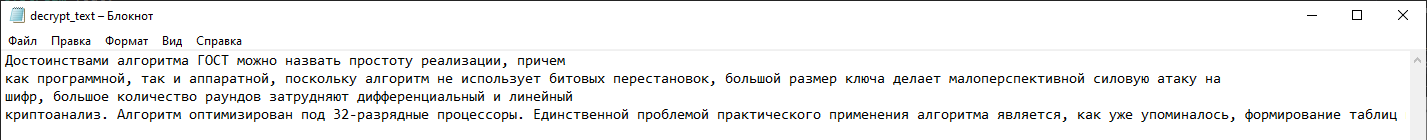
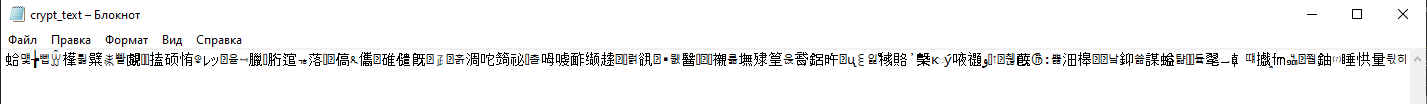
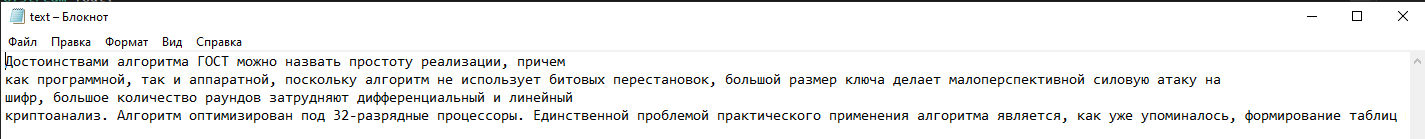
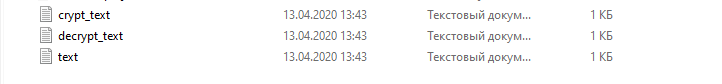
if (k == "0") return 0;

}

return 0;

}

**Результата работы программы**



**Ответы на вопросы**

* 1. Для чего был разработан, где применяется.

Работы над алгоритмом, положенным впоследствии в основу стандарта, начались в рамках темы «Магма» (защита информации криптографическими методами в ЭВМ ряда Единой Системы) по поручению Научно-Технического Совета Восьмого Главного управления КГБ СССР (ныне в структуре ФСБ), в марте 1978 года после длительного предварительного изучения опубликованного в 1976 году стандарта DES. В действительности работы по созданию алгоритма (или группы алгоритмов) схожего с алгоритмом DES начались уже в 1976 году.

Изначально работы имели гриф «Совершенно секретно». Затем были понижены до грифа «Секретно». В 1983 году гриф алгоритма был понижен до грифа «Для служебного пользования». Именно с последним грифом алгоритм был подготовлен для публикации в 1989 году. 9 марта 1987 года группа разработчиков-криптографов (заявитель - в/ч 43753) получила авторское свидетельство с приоритетом №333297 на изобретение на устройство шифрования по алгоритму «Магма-2».

* 1. Принцип работы и характеристики

Сначала правая половина блока и ключ раунда складываются по модулю 232. Результат сложения разбивается на восемь 4-битовых последовательностей, каждая из которых поступает на вход соответствующего S-блока. Каждый блок представляет собой таблицу подстановки, которая заменяет поступающее на вход число в диапазоне [0..15] на другое число в том же диапазоне. Выходы всех S-блоков объединяются в 32-битное слово, которое затем циклически сдвигается влево на 11 битов и объединяется с левой частью блока операцией XOR.

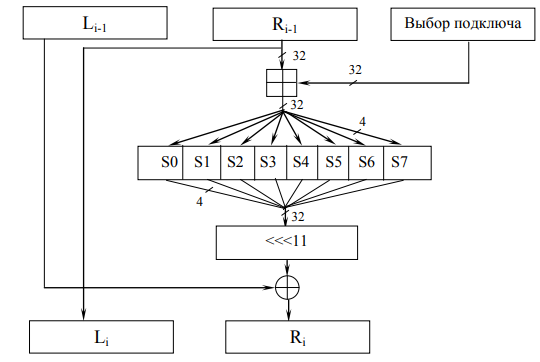


Рисунок 1 – Образующая функция алгоритма ГОСТ 28147-89

Формирование ключей раунда осуществляется по простой схеме. 256-битный ключ разбивается на восемь 32-битных машинных слов. Они нумеруются с К0 по К7. 32 ключа раунда получаются применением этих машинных слов в следующем порядке:

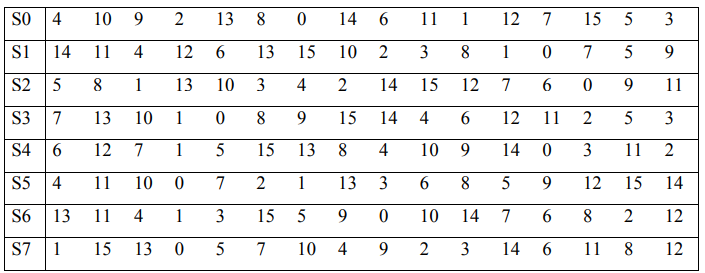


то есть в последних 8 раундах ключи подаются в обратном порядке.

Алгоритм ГОСТ является симметричным, и для дешифрации достаточно подать на вход алгоритма блоки зашифрованных сообщений и ключи раундов в порядке, обратном их следованию при шифрации.

Особенностью алгоритма является отсутствие в стандарте каких-либо рекомендаций по выбору содержимого таблиц подстановок (S-блоков). Первоначально 8\*16\*4=512 бит таблиц подстановок являлись также частью ключевой информации. Впоследствии требование к секретности содержимого таблиц было упразднено, однако статическое, обеспечивающее высокую криптостойкость алгоритма, содержимое таблиц подстановки так и не было опубликовано. Набор S-блоков, рекомендуемый стандартом хеширования ГОСТ Р 34.11-94, использующего блочный шифр ГОСТ 28147-89 в качестве основной преобразующей операции, приведен в таблице 1.

Таблица 1. S-блоки алгоритма ГОСТ 28147-89



3) Криптостойкость для алгоритма

Считается, что ГОСТ устойчив к таким широко применяемым методам, как линейный и дифференциальный криптоанализ. Обратный порядок использования ключей в последних восьми раундах обеспечивает защиту от атак скольжения (slide attack) и отражения (reflection attack). Ростовцев А. Г., Маховенко Е. Б., Филиппов А. С., Чечулин А. А. в своей работе описали вид криптоанализа, который сводится к построению алгебраической целевой функции и нахождению её экстремума. Были выделены классы слабых ключей, в частности, показано, что разреженные ключи (со значительным преобладанием 0 или 1) являются слабыми. По мнению авторов, их метод в любом случае лучше, чем полный перебор, однако без численных оценок.

* 1. Криптоанализ (известные уязвимости)

В мае 2011 года известный криптоаналитик Николя Куртуа доказал существование атаки на данный шифр, имеющей сложность в 28 (256) раз меньше сложности прямого перебора ключей при условии наличия 264 пар открытый текст/закрытый текст. Данная атака не может быть осуществлена на практике ввиду слишком высокой вычислительной сложности. Более того, знание 264 пар открытый текст/закрытый текст, очевидно, позволяет читать зашифрованные тексты, даже не вычисляя ключа. В большинстве других работ также описываются атаки, применимые только при некоторых предположениях, таких как определенный вид ключей или таблиц замен, некоторая модификация исходного алгоритма, или же требующие все ещё недостижимых объёмов памяти или вычислений. Вопрос о наличии применимых на практике атак без использования слабости отдельных ключей или таблиц замены остается открытым.

ECB:

В ГОСТ 28147—89 этот режим называется режимом простой замены.  
Расшифровкавыполняется функцией {\displaystyle D\_{k}} с использованием того же ключа *k*:{\displaystyle P\_{i}=D\_{k}\left(C\_{i},k\right).}

Особенности:

* каждый блок шифруется/расшифровывается независимо от других блоков.

Недостатки ECB:

* сохранение статистических особенностей открытого текста (поскольку одинаковым блокам шифротекста соответствуют одинаковые блоки открытого текста).

Достоинства ECB:

* постоянная скорость обработки блоков (скорость определяется эффективностью реализации шифра);
* возможно распараллеливание вычислений (так как блоки не связаны между собой).