МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

ОТЧЕТ

по лабораторной работе №4

«Режимы шифра DES»

по дисциплине

«Методы и средства защиты информации»

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Капранов С. Н.\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сухоруков В.А.\_\_\_

(подпись) (фамилия, и.,о.)

\_\_\_\_\_\_19-ВМ\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2022

# Задание №4.2

# Реализовать стандарт шифрования данных DES в режиме «Обратная связь по шифру».

# Описание алгоритма.

В этом режиме размер блока может отличаться от 64 бит. Файл, подлежащий шифрованию (расшифрованию), считывается последовательными блоками длиной к битов (к =1 ...64).

Схема шифрования реализуется следующим образом.

1. Исходное сообщение (файл) разбивается на блоки длиной к битов каждый (остаток дописывается нулями или пробелами).

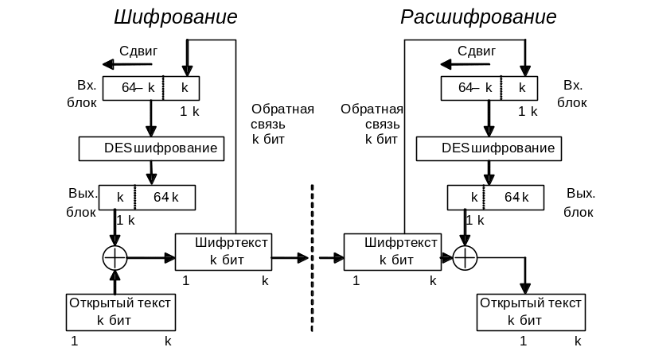
2. Задается входной блок (называется 64-битовый регистр сдвига), который содержит 64-битовый вектор инициализации.

3. Входной блок подвергается DES шифрованию.

4. Полученный после шифрования блок разделяется на k старших бит и 64-k бит. K старших бит складываются по модулю 2 с k битами открытого текста. Блок, полученный после сложения, является k битовым блоком шифротекста.

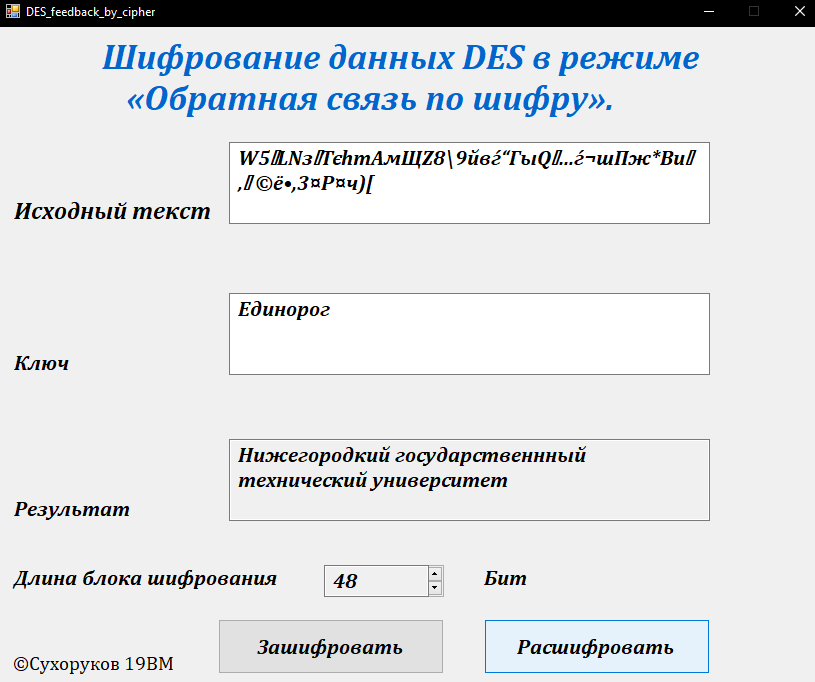
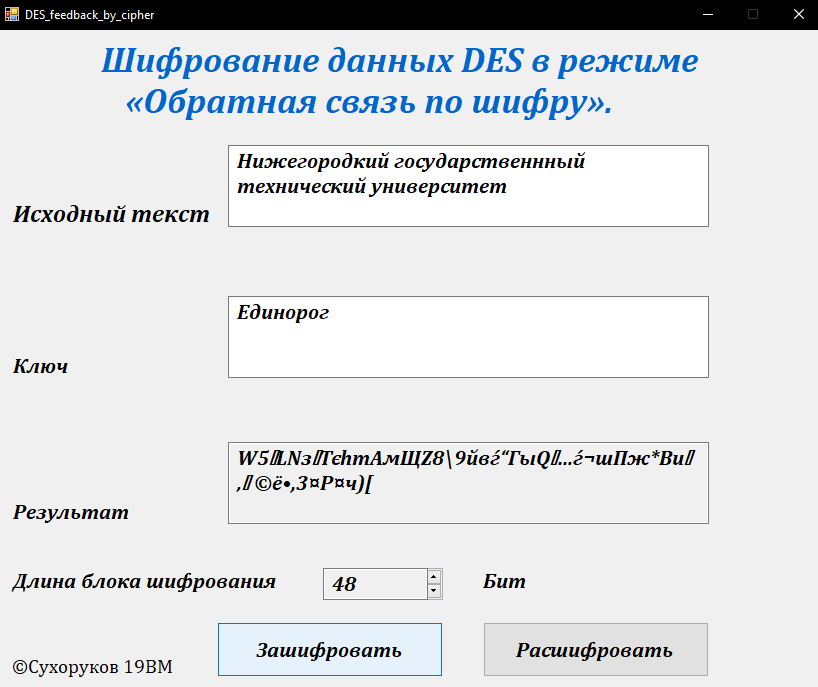
5. k битовый блок шифротекста участвует в обновлении регистра сдвига. Из входного блока удаляются k старших битов, остальные биты сдвигаются влево и на освободившееся место записывается k битовый блок шифротекста.

6. Процесс шифрования заканчивается, когда будут зашифрованы все блоки открытого текста.

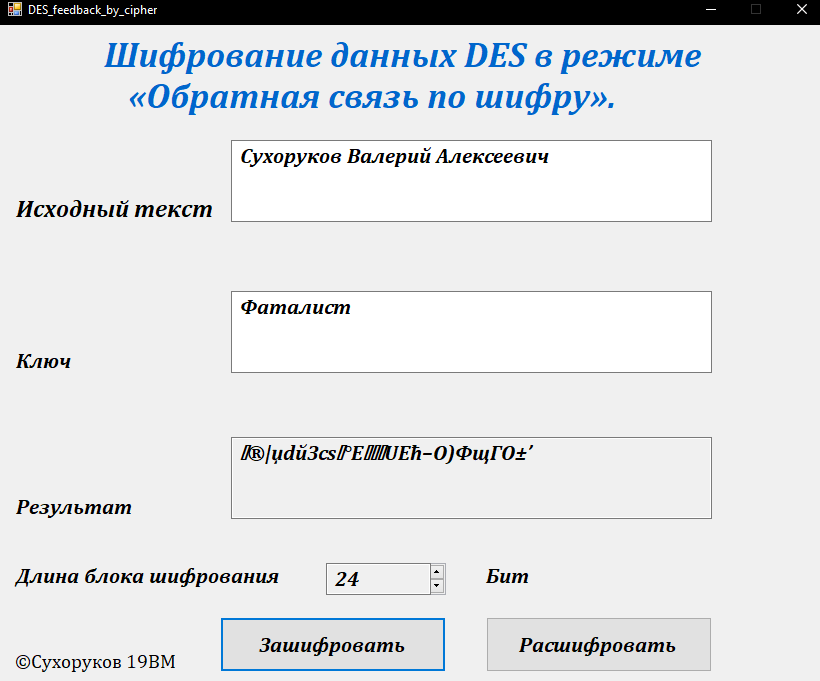


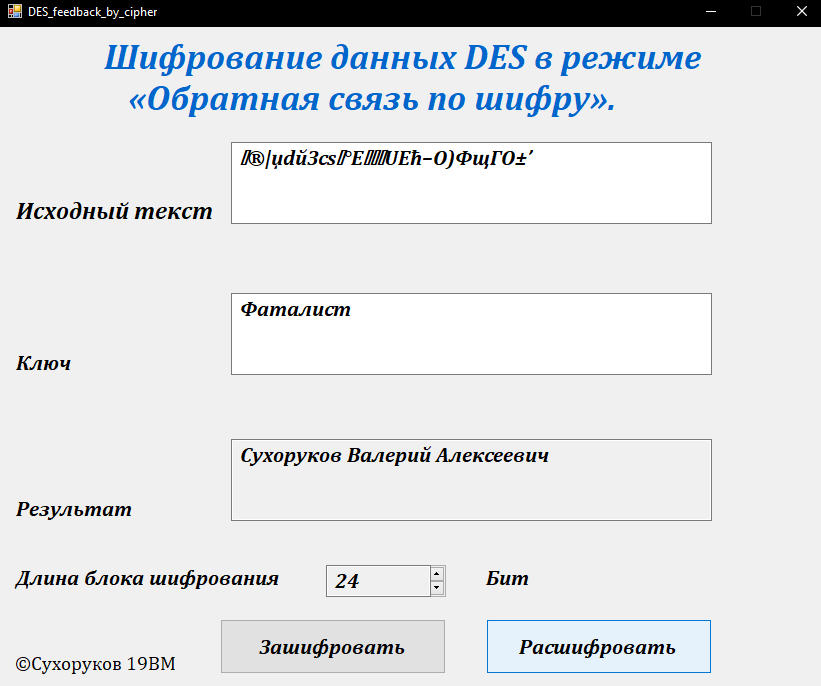
# Примеры шифрования и расшифровывания

* Нижегородский государственный технический университет



* Сухоруков Валерий Алексеевич





# Текст программы

* Функция зашифровки всего текста

//Функция, вызываемая при нажатии на кнопку "Зашифровать" в форме.

//Устанавливает в текстовое поле 3 результат функции шифрования.

System::Void LR4UI::DES\_feedback\_by\_cipher::button1\_Click(System::Object^ sender, System::EventArgs^ e) {

string text = "";

string key = "";

string result = "";

string f\_result = "";

bitset<64> resKey(0);

int offset = 0;

int input\_blok\_lenght = (int) numericUpDown1->Value\*8;

text = marshal\_as<string>(textBox1->Text);

key = marshal\_as<string>(textBox2->Text);

//конвертация ключа в последовательность бит

//проход по каждому символу

for (int itKey = 0; itKey < key.length(); itKey++) {

//конвертируем символ в биты

bitset<8> val(key[itKey]);

//Записываем значение 8 бит текущего символа

for (int i = 0; i < 8; i++) {

//заполняем результирущий список

resKey[offset + i] = val[i];

}

//Сдвигаемся на 8 бит

offset += 8;

}

//Если длина ключа меньше 8 символов,

//заполняем оставшуюся часть нулями

for (int itKey = offset; itKey < 64; itKey++) {

resKey[itKey] = 0;

}

//Конвертация строки открытого текста в последовательность бит

int lenght\_text\_in\_bit = 8 \* text.length();

vector<bool> all\_text\_bit(lenght\_text\_in\_bit);

for (int i = 0; i < text.length(); i++){

bitset<8> tmp(text[i]);

for (int j = 0; j < 8; j++){

all\_text\_bit[i \* 8 + j] = tmp[j];

}

}

//Объявление вектора инициализации

bitset<64> vector\_inital(12346789);

//Все биты шифротекста

vector<bool>cipher\_text(lenght\_text\_in\_bit);

//К бит шифротекста

vector<bool>k\_bit\_cipher\_text(input\_blok\_lenght);

//К бит результата шифрования DES

vector<bool>k\_bit(input\_blok\_lenght);

//Количество блоков по k бит

int n= lenght\_text\_in\_bit/input\_blok\_lenght;

int finish,finish1;

for (int j = 0; j < n; j++){

//DES шифрование

k\_bit = coder(vector\_inital, resKey, input\_blok\_lenght);

//Сложение по модулю 2 с открытым текстом - получение шифротекста

for (size\_t i = 0; i < input\_blok\_lenght; i++) {

k\_bit\_cipher\_text[i] = k\_bit[i] ^ all\_text\_bit[i+j\*input\_blok\_lenght];

finish = i + j \* input\_blok\_lenght;

}

//Сдвиг вектора инициализации на K бит

vector\_inital <<= input\_blok\_lenght;

//Запись в конец вектора инициализации K бит шифротекста

//Дополнение всего шифротекста K битами

for (size\_t i = 0; i < input\_blok\_lenght; i++) {

//Запись в конец вектора инициализации шифротекста

vector\_inital[63 - input\_blok\_lenght + i] = k\_bit\_cipher\_text[i];

cipher\_text[i + j \* input\_blok\_lenght]= k\_bit\_cipher\_text[i];

finish1 = i + j \* input\_blok\_lenght;

}

}

//Обработка осташихся бит

n = lenght\_text\_in\_bit % input\_blok\_lenght;

k\_bit = coder(vector\_inital, resKey, input\_blok\_lenght);

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

k\_bit\_cipher\_text[i]= k\_bit[i] ^ all\_text\_bit[finish+1+i];

}

for (size\_t i = 0; i < input\_blok\_lenght-n; i++){

k\_bit\_cipher\_text[n+i]= k\_bit[i] ^ 0;

}

int tmp = 0;

for (size\_t i = finish1+1; i < lenght\_text\_in\_bit; i++){

cipher\_text[i] = k\_bit\_cipher\_text[tmp];

tmp++;

}

//Преобразование бит в стоку

result = convertToString(cipher\_text, lenght\_text\_in\_bit);

//Приведение типов для вывода в форму

String^ myString = marshal\_as<System::String^>(result);

textBox3->Text = myString;

return System::Void();

}

* Функция зашифровки 64 бит текста и выделения старших k бит

//----------------------------------------------------------------------------------

//Функция шифрования методом DES в режиме "Обратная связь по шифру"

vector<bool> coder(bitset<64> res, bitset<64> resKey,int input\_blok\_lenght) {

//разделение сообщения на блоки

bitset<64> resIP(0);

int spot = 0;

//Начальная перестановка бит по таблице IP

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

spot = IP[i][j];

resIP[i \* 8 + j] = res[spot - 1];

}

}

//правые и левые биты сообщения

bitset<32> L(0); bitset<32> R(0);

//Разделение битов

sortLR(L, R, resIP);

//Две половины ключа

bitset<28> C(0); bitset<28> D(0);

//Получение 56 битного ключа

sortCD(C, D, resKey);

//Ключ i-того раунда

bitset<48> Ki(0);

//16 Раундов шифрования

for (int c = 0; c < 16; c++) {

getKi(C, D, c, Ki);

bitset<32> resB(0);

F(R, Ki, resB);

bitset<32> tmp1 = L;

bitset<32> tmp2 = R;

L = tmp2;

R = tmp1 ^ resB;

}

//Составляем из двух половин целую строку 64 бит

bitset<64> resultLR(0);

for (int i = 0; i < 64; i++) {

if (i < 32) { resultLR[i] = L[i]; }

else { resultLR[i] = R[i - 32]; }

}

//Конечная перестановка

bitset<64> FP(0);

spot = 0;

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

spot = finalIP[i][j];

FP[i \* 8 + j] = resultLR[spot - 1];

}

}

//Выделение старших к бит

vector<bool> k\_bit(input\_blok\_lenght);

for (int i = 0; i < input\_blok\_lenght; i++){

k\_bit[i] = FP[i];

}

return k\_bit;

}

* Функция расшифровки текста

//Функция, вызываемая при нажатии на кнопку "Расшифровать" в форме.

//Устанавливает в текстовое поле 3 результат функции расшифрования.

System::Void LR4UI::DES\_feedback\_by\_cipher::button2\_Click(System::Object^ sender, System::EventArgs^ e) {

string result = "";

string key = "";

string decoderResult = "";

int input\_blok\_lenght = (int)numericUpDown1->Value \* 8;

bitset<64> resKey(0);

int offset = 0;

result = marshal\_as<string>(textBox1->Text);

key = marshal\_as<string>(textBox2->Text);

//конвертация ключа в последовательность бит

//проход по каждому символу

for (int itKey = 0; itKey < key.length(); itKey++) {

//конвертируем символ в биты

bitset<8> val(key[itKey]);

//Записываем значение 8 бит текущего символа

for (int i = 0; i < 8; i++) {

//заполняем результирущий список

resKey[offset + i] = val[i];

}

//Сдвигаемся на 8 бит

offset += 8;

}

//Если длина ключа меньше 8 символов, заполняем оставшуюся часть нулями

for (int itKey = offset; itKey < 64; itKey++) {

resKey[itKey] = 0;

}

//Конвертация строки закрытого текста в последовательность бит

int lenght\_text\_in\_bit = 8 \* result.length();

vector<bool> all\_text\_bit(lenght\_text\_in\_bit);

for (int i = 0; i < result.length(); i++) {

bitset<8> tmp(result[i]);

for (int j = 0; j < 8; j++) {

all\_text\_bit[i \* 8 + j] = tmp[j];

}

}

//Объявление вектора инициализации

bitset<64> vector\_inital(12346789);

//Все биты открытого текста

vector<bool>open\_text(lenght\_text\_in\_bit);

//К бит открытого текста

vector<bool>k\_bit\_open\_text(input\_blok\_lenght);

//К бит результата шифрования DES

vector<bool>k\_bit(input\_blok\_lenght);

//Количество блоков по k бит

int n = lenght\_text\_in\_bit / input\_blok\_lenght;

int finish, finish1;

for (int j = 0; j < n; j++) {

//DES шифрование

k\_bit = coder(vector\_inital, resKey, input\_blok\_lenght);

//Сложение по модулю 2 с закрытым текстом - получение открытого текста

for (size\_t i = 0; i < input\_blok\_lenght; i++) {

k\_bit\_open\_text[i] = k\_bit[i] ^ all\_text\_bit[i + j \* input\_blok\_lenght];

finish = i + j \* input\_blok\_lenght;

}

//Сдвиг вектора инициализации на K бит

vector\_inital <<= input\_blok\_lenght;

//Запись в конец вектора инициализации K бит шифротекста

//Дополнение всего открытого текста K битами

for (size\_t i = 0; i < input\_blok\_lenght; i++) {

//Запись в конец вектора инициализации шифротекста

vector\_inital[63 - input\_blok\_lenght + i] = all\_text\_bit[i + j \* input\_blok\_lenght];

open\_text[i + j \* input\_blok\_lenght] = k\_bit\_open\_text[i];

finish1 = i + j \* input\_blok\_lenght;

}

}

//Обработка осташихся бит

n = lenght\_text\_in\_bit % input\_blok\_lenght;

k\_bit = coder(vector\_inital, resKey, input\_blok\_lenght);

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

k\_bit\_open\_text[i] = k\_bit[i] ^ all\_text\_bit[finish + 1 + i];

}

for (size\_t i = 0; i < input\_blok\_lenght - n; i++) {

k\_bit\_open\_text[n + i] = k\_bit[i] ^ 0;

}

int tmp = 0;

for (size\_t i = finish1 + 1; i < lenght\_text\_in\_bit; i++) {

open\_text[i] = k\_bit\_open\_text[tmp];

tmp++;

}

//Преобразование бит в стоку

decoderResult = convertToString(open\_text, lenght\_text\_in\_bit);

//Приведение типа для вывода строки в текстовое поле

String^ myString = marshal\_as<System::String^>(decoderResult);

textBox3->Text = myString;

return System::Void();

}

* Функция расшифровки 64 бит текста и выделения старших k бит

//Функция расшифрования методом DES в режиме "Обратная связь по шифру"

vector<bool> deCoder(bitset<64> res, bitset<64> resKey, int input\_blok\_lenght) {

bitset<64> resIP(0);

int spot = 0;

//Конечная перестановка

//Присваиваем результирующему списку значение бита по таблице IP наоборот

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

spot = finalIP[i][j];

resIP[spot - 1] = res[i \* 8 + j];

}

}

bitset<32> L(0);

bitset<32> R(0);

//разделение битов на левые и правые

sortLR(L, R, resIP);

bitset<28> C(0);

bitset<28> D(0);

sortCD(C, D, resKey);

bitset<48> Ki(0);

//Проходим 16 раундов в обратном порядке

for (int c = 16; c > 0; c--) {

getDecoderKi(C, D, c, Ki);

bitset<32> resB(0);

F(L, Ki, resB);

bitset<32> tmp1 = L;

bitset<32> tmp2 = R;

R = tmp1;

L = tmp2 ^ resB;

}

//Составляем из двух половин целую строку 64 бит

bitset<64> resultLR(0);

for (int i = 0; i < 64; i++) {

if (i < 32) { resultLR[i] = L[i]; }

else { resultLR[i] = R[i - 32]; }

}

bitset<64> FP(0);

spot = 0;

//Начальная перестановка в обратном порядке

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

spot = IP[i][j];

FP[spot - 1] = resultLR[i \* 8 + j];

}

}

//Выделение старших к бит

vector<bool> k\_bit(input\_blok\_lenght);

for (int i = 0; i < input\_blok\_lenght; i++) {

k\_bit[i] = FP[i];

}

return k\_bit;

}