Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.01 – «Информатика и вычислительная техника»

Дисциплина: «Защита информации»

Профиль: «Автоматизированные системы обработки информации и управления»

Семестр 7

ОТЧЕТ

по лабораторной работе №4

Тема: «Блочное шифрование DES»

Выполнил: студент группы АСУ-17-1б

Хохряков Денис

Проверил: доцент кафедры ИТАС

Шереметьев В.Г.\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_

# ЦЕЛЬ РАБОТЫ

Получить практические навыки по созданию и применению блочных шифров

# ЗАДАНИЕ

Реализовать шифрование бинарного файла, методом блочного шифрования, используя блоки длиной 32 бит, ключ длиной 16 бит, реализуя в алгоритме шифрования методику DES.

# КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

**Блочный шифр** — разновидность симметричного шифра, оперирующего группами бит фиксированной длины — блоками, характерный размер которых меняется в пределах 64‒256 бит. Если исходный текст (или его остаток) меньше размера блока, перед шифрованием его дополняют. Фактически, блочный шифр представляет собой подстановку на алфавите блоков, которая, как следствие, может быть моно- или полиалфавитной. Блочный шифр является важной компонентой многих криптографических протоколов и широко используется для защиты данных, передаваемых по сети.

**DES** (Data Encryption Standard) — алгоритм для симметричного шифрования, разработанный фирмой IBM и утверждённый правительством США в 1977 году как официальный стандарт (FIPS 46-3). Размер блока для DES равен 64 битам. В основе алгоритма лежит сеть Фейстеля с 16 циклами (раундами) и ключом, имеющим длину 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки E, IP, IP-1) преобразований.

# ХОД РАБОТЫ

В нашей работе требуется блочное шифрование с размером блока 32бита и длиной ключа 16бит (2 байта – 4 символа HEX)

Последовательность действий программы при шифровании следующая:

1. Делим весь файл на блоки размером 4 байта. (Каждый блок – 4х байтовое число)
2. Применяем битовую перестановку IP
3. Делим число на два 2х байтовых числа
4. К ключу также применяем свою битовую перестановку, и сдвигаем циклически вправо
5. Складываем по модулю *левую* часть с вычисленным ключом и правой частью
6. Меняем левую и правую часть местами
7. Сдвигаем ключ циклически *вправо*
8. Повторяем пункты 5-8 N раз.
9. Выполняем обратную битовую перестановку IP-1
10. Соединяем все зашифрованные блоки и файл зашифрован!

А при дешифровании следующая последовательность:

1. Делим весь файл на блоки размером 4 байта. (Каждый блок – 4х байтовое число)
2. Выполняем битовую перестановку IP
3. Делим число на два 2х байтовых числа
4. К ключу также применяем свою битовую перестановку и сдвигаем циклически вправо на N шагов
5. Складываем по модулю *правую* часть с вычисленным ключом и левой частью
6. Меняем левую и правую часть местами
7. Сдвигаем ключ циклически *влево*
8. Повторяем пункты 5-8 N раз.
9. Выполняем обратную битовую перестановку IP-1
10. Соединяем все зашифрованные блоки и файл расшифрован!

# РАБОТА ПРОГРАММЫ

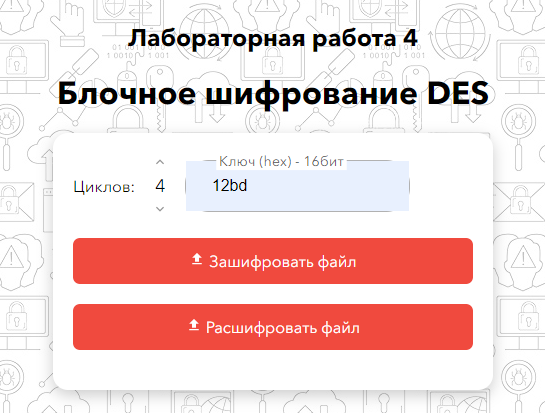


Рисунок 1. Общий вид программы

Загружаем в программу файл (изображение), кол-во блоков – 5

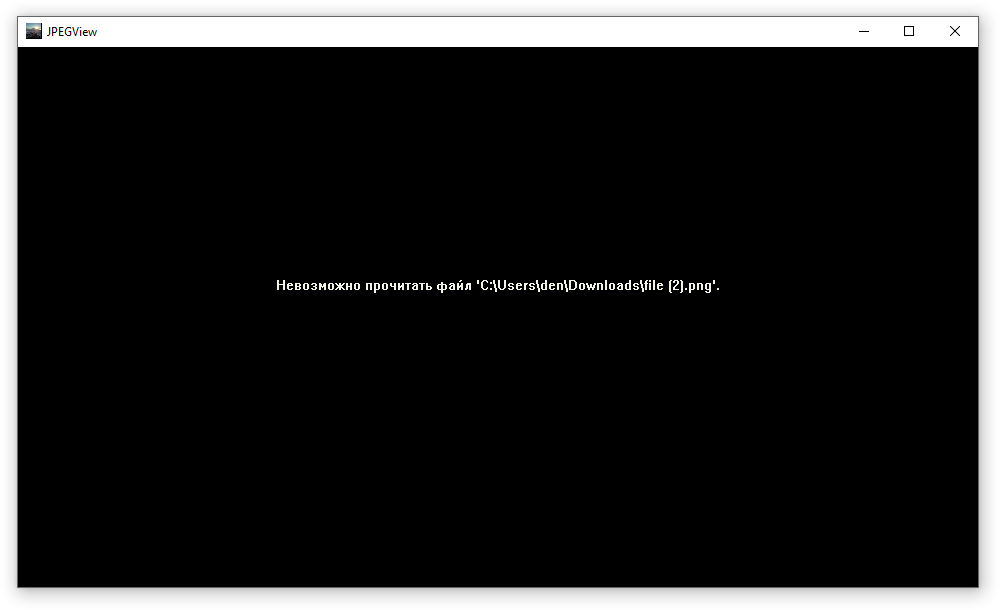


Рисунок 2. Зашифрованный файл. Прочитать - невозможно

Данный файл очень сильно изменился по сравнению с исходным, и, не зная ключа, расшифровать его будет очень практически невозможно

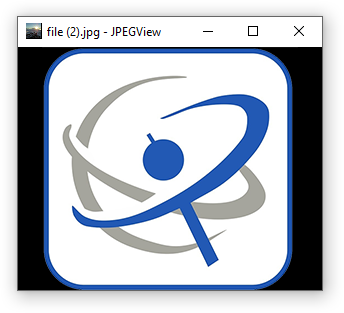


Рисунок 3. Расшифрованный файл

А вот после того, как мы расшифровали файл, его снова можно посмотреть (предварительно сменив расширение с .bin на .png). При этом есть вероятность, что в конце файла будут добавлены пустые блоки, но они никак не сказываются на чтении файла)

# ЛИСТИНГ ПРОГРАММЫ (Язык – JavaScript)

function cycled (int){ //Циклический сдвиг вправо

return (int >> 1) + ((int & 1) << 15);

}

//Функция шифрования файла

function cipher (int, \_key, cycles){

let newInt = 0;

for(let i = 0; i < initialPermutation.length; i++)

newInt += ((int >> i & 1) << initialPermutation[i]); //Здесь мы получаем нужный бит и сдвигаем его на нужную позицию перестановки

let l0 = newInt >>> 16;

let r0 = (newInt & ~(~0 << 16)); //Ну что сказать - здесь мы делим число на две части

let key = 0;

for(let i = 0; i < keyPermutation.length; i++)

key += ((\_key >> i & 1) << keyPermutation[i]);

for(let i = 0; i < cycles; i++){

key = cycled(key);

const a = l0;

l0 = l0 ^ key ^ r0;

r0 = a;

}

const \_newInt = (l0 << 16) + r0;

let finalInt = 0;

for(let i = 0; i < initialPermutation.length; i++)

finalInt += ((\_newInt >> initialPermutation[i] & 1) << i);

return finalInt;

}

//Функция дешифрования файла

function decypher (int, \_key, cycles){

let newInt = 0;

for(let i = 0; i < initialPermutation.length; i++)

newInt += ((int >> i & 1) << initialPermutation[i]); //Здесь мы получаем нужный бит и сдвигаем его на нужную позицию перестановки

let l0 = (newInt >>> 16) | 0;

let r0 = (newInt & ~(~0 << 16)); //Ну что сказать - здесь мы делим число на две части

let key = 0;

for(let i = 0; i < keyPermutation.length; i++)

key += ((\_key >> i & 1) << keyPermutation[i]);

const keyArr = [];

for(let i = 0; i < cycles; i++){

key = cycled(key);

keyArr.push(key);

}

for(let i = 0; i < cycles; i++){

key = keyArr[cycles-i-1];

const a = r0;

r0 = r0 ^ key ^ l0;

l0 = a;

}

const \_newInt = (l0 << 16) + r0;

let finalInt = 0;

for(let i = 0; i < initialPermutation.length; i++)

finalInt += ((\_newInt >> initialPermutation[i] & 1) << i);

return finalInt;

}