Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по лабораторной работе №7

«Исследование блочных шифров»

Студентка: Пунько А.А,

ФИТ 3 курс 5 группа

Преподаватель: Берников В. О.

Минск 2020

**Цель**: изучение и приобретение практических навыков разработки и использования приложений для реализации блочных шифров (рассчитана на 4 часа аудиторных занятий).

**Задачи**:

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости блочных шифров.
2. Разработать приложение для реализации указанных преподавателем методов блочного зашифрования/расшифрования.
3. Выполнить анализ криптостойкости блочных шифров.
4. Оценить скорость зашифрования/расшифрования реализованных шифров.
5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Теоретические сведения**

Блочное зашифрование (расшифрование) предполагает разбиение исходного открытого (зашифрованного) текста на равные блоки, к которым применяется однотипная процедура зашифрования (расшифрования). Указанная однотипность характеризуется, прежде всего, тем, что процедура зашифрования (расшифрования) состоит из совокупности повторяющихся наборов преобразований, называемых раундами.

Основные требования к шифрам рассматриваемого класса можно сформулировать следующим образом:

* незначительное изменение исходного сообщения должно приводить к существенному изменению зашифрованного сообщения;
* устойчивость к атакам по выбранному тексту;
* алгоритмы зашифрования/расшифрования должны быть реализуемыми на различных платформах;
* алгоритмы должны базироваться на простых операциях;
* алгоритмы должны быть простыми для написания кода, вероятность появления программных ошибок должна быть низкой;
* алгоритмы должны допускать их модификацию при переходе на иные требования по уровню криптостойкости.

Само название конструкции Фейстеля (сеть) означает ее ячеистую топологию. Формально одна ячейка сети соответствует одному раунду зашифрования или расшифрования сообщения. При зашифровании сообщение разбивается на блоки одинаковой (фиксированной) длины (как правило – 64 или 128 бит). Полученные блоки называются входными. В случае, если длина входного блока меньше, чем выбранный размер, то блок удлиняется установленным способом. Каждый входной блок шифруемого сообщения изначально делится на два подблока одинакового размера: левый (L0) и правый (R0). Далее в каждом i-ом раунде выполняются преобразования в соответствии с формальным представлением ячейки сети Фейстеля:

Li = Ri-1,

Ri = Li-1 + f (Ri-1, Ki),

По какому-либо математическому правилу вычисляется раундовый ключ Ki. В приведенном выражение знак «+» соответствует поразрядному суммированию на основе «XOR». На рис. 1.1 приведено графическое отображение сети Фейстеля. Расшифрование происходит так же, как и зашифрование, с той лишь разницей, что раундовые ключи будут использоваться в обратном порядке по отношению к зашифрованию. В своей статье Х. Фейстель описывает два блока преобразований с использованием функции f (Ri-1, Ki): блок подстановок (S-блок, англ. S-box); блок перестановок (P-блок, англ. P-box). Блок подстановок состоит из:

* дешифратора, преобразующего n-разрядное двоичное число в одноразрядное сигнал по основанию 2n;
* внутреннего коммутатора;
* шифратора, преобразующего сигнала из одноразрядного 2n-ричного в n-разрядный двоичный.

Пример реализации трехразрядного S-блок показан на рис. 1.1

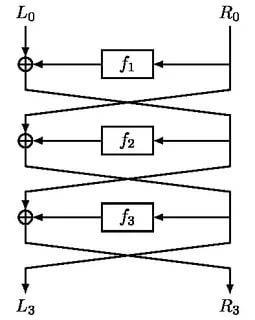


Рисунок 1.1 - Графическое отображение сети Фейстеля

В основе криптостойкости блочных шифров лежит идея К. Шеннона в представлении составного шифра таким образом, чтобы он обладал двумя важными свойствами: рассеянием и перемешиванием. Рассеивание должно скрыть отношения между зашифрованным текстом и исходным текстом.

Рассеивание подразумевает, что каждый символ (символ или бит) в зашифрованном тексте зависит от одного или всех символов в исходном тексте.

Другими словами, если единственный символ в исходном тексте изменен, несколько или все символы в зашифрованном тексте будут также изменены.

Идея относительно перемешивания заключается в том, что оно должно скрыть отношения между зашифрованным текстом и ключом.

Алгоритм DES строится на основе сети Фейстеля.

Входной блок данных, состоящий из 64 бит, преобразуется в выходной блок идентичной длины. В алгоритме широко используются рассеивания (подстановки) и перестановки битов текста, о которых мы упоминали выше. Комбинация двух указанных методов преобразования образует фундаментальный строительный блок DES, называемый раундом или циклом.

Один блок данных подвергается преобразованию (и при зашифровании, и при расшифровании) в течение 16 раундов.

После первоначальной перестановки и разделения 64-битного блока данных на правую (*R*0) и левую (*L*0) половины длиной по 32 бита выполняются 16 раундов одинаковых действий.

Вначале правая часть блока Ri расширяется до 48 битов, используя таблицу, которая определяет перестановку плюс расширение на 16 битов. Эта операция приводит размер правой половины в соответствие с размером ключа для выполнения операции XOR. Кроме того, за счет выполнения этой операции быстрее возрастает зависимость всех битов результата от битов исходных данных и ключа (это называется «лавинным эффектом»).

После выполнения перестановки с расширением для полученного 48-битного значения выполняется операция XOR с 48-битным подключом Ki. Затем полученное 48-битное значение подается на вход блока подстановки S (от англ. Substitution – подстановка), результатом которой является 32-битное значение. Подстановка выполняется в восьми блоках подстановки или восьми S-блоках (S-boxes). При выполнении этой операции 48 битов данных делятся на восемь 6-битных подблоков, каждый из которых по соответствующей таблице замен замещается четырьмя битами. Подстановка с помощью S-блоков является одним из важнейших этапом DES. Таблицы замен для этой операции спроектированы так, чтобы обеспечивать максимальную криптостойкость. В результате выполнения этого этапа получаются восемь 4-битных блоков, которые вновь объединяются в единое 32-битное значение.

Далее полученное 32-битное значение обрабатывается с помощью перестановки Р (от англ. Permutation – перестановка), которая не зависит от используемого ключа. Целью перестановки является максимальное переупорядочивание битов такое, чтобы в следующем раунде шифрования каждый бит с большой вероятностью обрабатывался другим S-блоком.

И, наконец, результат перестановки объединяется с помощью операции XOR с левой половиной первоначального 64-битового блока данных. Затем левая и правая половины меняются местами, и начинается следующий раунд (см. рис. 5.6).

После выполнения 16-раундового зашифрования 64-битного блока дан- осуществляется конечная перестановка (IP−1). Она является обратной к перестановке IP.

Общая схема алгоритма DES представлена на рисунке 1.2.

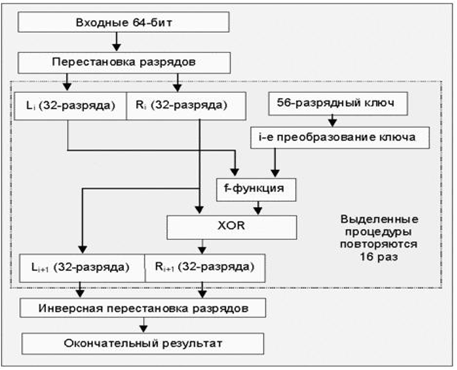
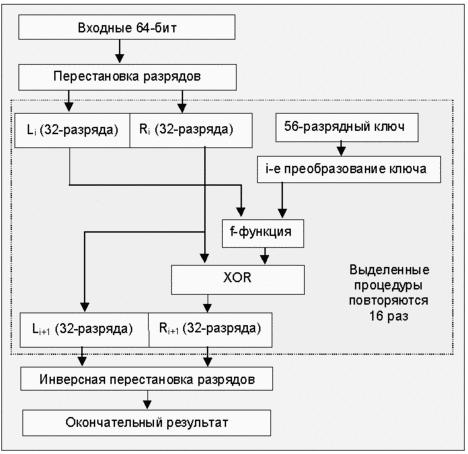
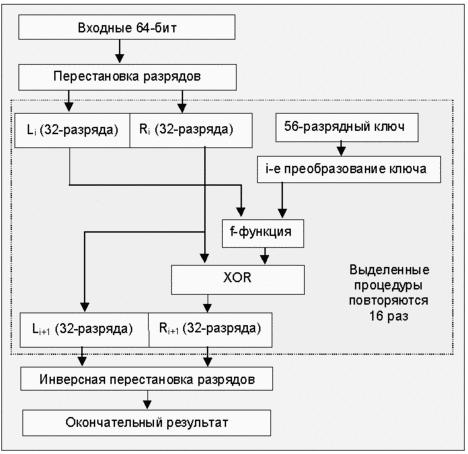
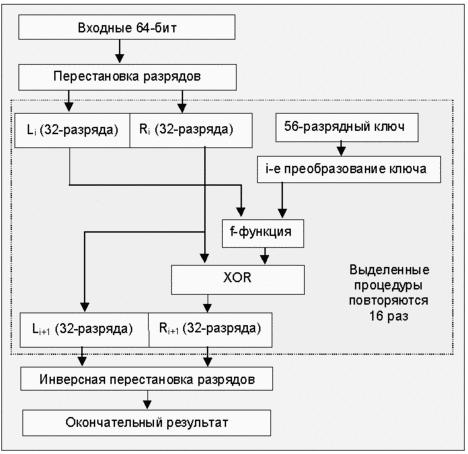
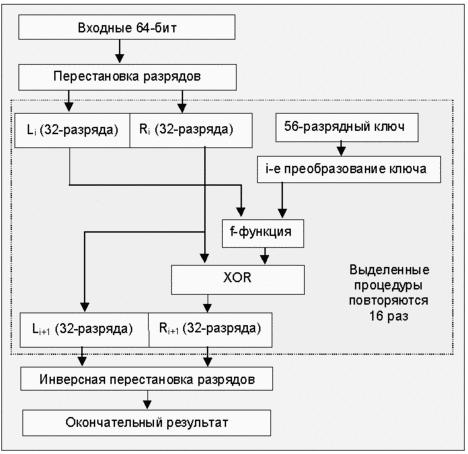


Рисунок 1.2 – Общая схема алгоритма DES

Дифференциальный криптоанализбазируется натаблице неоднородных дифференциальных распределений S-блоков в блочном шифре. Криптоанализ шифртекстов на основе рассматриваемого стандарта «работает» с парами шифртекстов, открытые тексты которых имеют определенные разности, как это отмечалось в материалах к лабораторной работе № 2. Метод анализирует эволюцию этих разностей в процессе прохождения открытых текстов раундов DES при шифровании одним и тем же ключом.

Для DES термин «разность» определяется с помощью операции XOR. Затем, используя разности полученных шифртекстов, присваивают различные вероятности различным ключам. В процессе дальнейшего анализа следующих пар шифртекстов один из ключей станет наиболее вероятным. Это и есть правильный ключ.

Линейный криптоанализ. Для того чтобы найти линейное приближение для DES нужно найти «хорошие» однораундовые линейные приближения и объединить их. Обратим внимание на S-блоки. У них 6 входных битов и 4 выходных. Входные биты можно объединить с помощью операции XOR 63 способами (26 - 1), а выходные биты – 15 способами. Теперь для каждого S-блока можно оценить вероятность того, что для случайно выбранного входа входная комбинация XOR равна некоторой выходной комбинации XOR. И т.д.

AES (Advanced Encryption Standard) – алгоритм шифрования, действующий в качестве государственного стандарта в США с 2001 года. В основу стандарта положен шифр Rijndael. Шифр Rijndael/AES (то есть рекомендуемый стандартом) характеризуется размером блока 128 бит, длиной ключа 128, 192 или 256 бит и количеством раундов 10, 12 или 14 в зависимости от длины ключа.

Основу Rijndael составляют так называемые линейно-подстановочные преобразования. В алгоритме широко используются табличные вычисления, причем все необходимые таблицы задаются константно, т. е. не зависят ни от ключа, ни от данных.

**Практическая часть**

В результате выполнения данной работы было создано оконное приложение, выполняющее чтение текста из файла, считывание ключа, преобразование текста в бинарный формат, его разбиение на блоки и шифрование/расшифрование с использованием алгоритма DES.

Стартовое окно приложения представлено на рисунке 2.1

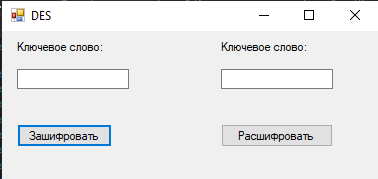


Рисунок 2.1 – Стартовое окно приложения

Результат зашифрования представлен на рисунке 2.2.

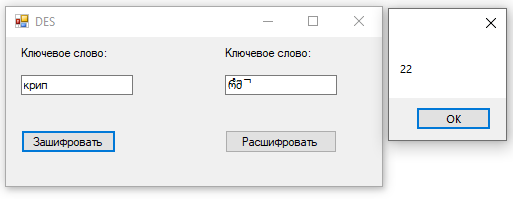


Рисунок 2.2 – Результат зашифрования исходного текста

Полученный шифротекст представлен на рисунке 2.3.

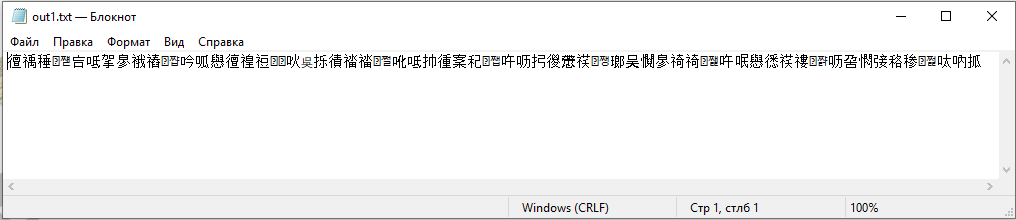


Рисунок 2.3 – Полученный шифротекст

Результат расшифрования представлен на рисунке 2.4.

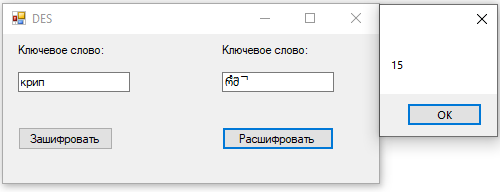


Рисунок 2.4 – Результат расшифрования шифротекста

Расшифрованный исходный текст представлен на рисунке 2.5.

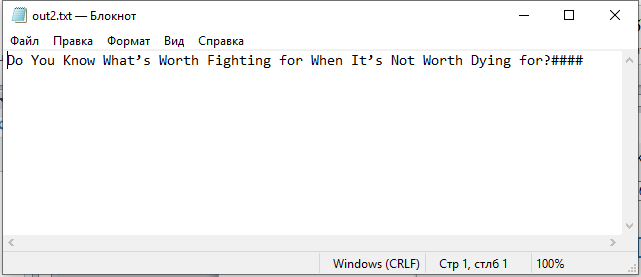


Рисунок 2.5 – Расшифрованный исходный текст

Функция для шифрования текста представлена в листинге 2.1.

private void buttonEncrypt\_Click(object sender, EventArgs e)

{

Stopwatch watch1 = new Stopwatch();

watch1.Start();

if (textBoxEncodeKeyWord.Text.Length > 0)

{

string s = "";

string key = textBoxEncodeKeyWord.Text;

StreamReader sr = new StreamReader("in.txt");

while (!sr.EndOfStream)

{

s += sr.ReadLine();

}

sr.Close();

s = StringToRightLength(s);

CutStringIntoBlocks(s);

key = CorrectKeyWord(key, s.Length / (2 \* Blocks.Length));

textBoxEncodeKeyWord.Text = key;

key = StringToBinaryFormat(key);

for (int j = 0; j < quantityOfRounds; j++)

{

for (int i = 0; i < Blocks.Length; i++)

Blocks[i] = EncodeDES\_One\_Round(Blocks[i], key);

key = KeyToNextRound(key);

}

key = KeyToPrevRound(key);

textBoxDecodeKeyWord.Text = StringFromBinaryToNormalFormat(key);

string result = "";

for (int i = 0; i < Blocks.Length; i++)

result += Blocks[i];

StreamWriter sw = new StreamWriter("out1.txt");

sw.WriteLine(StringFromBinaryToNormalFormat(result));

sw.Close();

watch1.Stop();

MessageBox.Show(watch1.ElapsedMilliseconds.ToString());

Process.Start("out1.txt");

}

else

MessageBox.Show("Введите ключевое слово!");

}

Листинг 2.1 – Функция для шифрования текста

Функция для дешифрования текста представлена в листинге 2.2.

private void buttonDecipher\_Click(object sender, EventArgs e)

{

Stopwatch watch1 = new Stopwatch();

watch1.Start();

if (textBoxDecodeKeyWord.Text.Length > 0)

{

string s = "";

string key = StringToBinaryFormat(textBoxDecodeKeyWord.Text);

StreamReader sr = new StreamReader("out1.txt");

while (!sr.EndOfStream)

{

s += sr.ReadLine();

}

sr.Close();

s = StringToBinaryFormat(s);

CutBinaryStringIntoBlocks(s);

for (int j = 0; j < quantityOfRounds; j++)

{

for (int i = 0; i < Blocks.Length; i++)

Blocks[i] = DecodeDES\_One\_Round(Blocks[i], key);

key = KeyToPrevRound(key);

}

key = KeyToNextRound(key);

textBoxEncodeKeyWord.Text = StringFromBinaryToNormalFormat(key);

string result = "";

for (int i = 0; i < Blocks.Length; i++)

result += Blocks[i];

StreamWriter sw = new StreamWriter("out2.txt");

sw.WriteLine(StringFromBinaryToNormalFormat(result));

sw.Close();

MessageBox.Show(watch1.ElapsedMilliseconds.ToString());

Process.Start("out2.txt");

}

else

MessageBox.Show("Введите ключевое слово!");

}

Листинг 2.2 – Функция для дешифрования текста

Функции для одного раунда шифрования и дешифрования представлены в листинге 2.3.

private string EncodeDES\_One\_Round(string input, string key)

{

string L = input.Substring(0, input.Length / 2);

string R = input.Substring(input.Length / 2, input.Length / 2);

return (R + XOR(L, f(R, key)));

}

//расшифровка DES один раунд

private string DecodeDES\_One\_Round(string input, string key)

{

string L = input.Substring(0, input.Length / 2);

string R = input.Substring(input.Length / 2, input.Length / 2);

return (XOR(f(L, key), R) + L);

}

Листинг 2.3 – Функции для одного раунда шифрования и дешифрования текста