Протокол самостоятельной работы

По Теории Защиты Информации

На тему: Сеть Фейстеля и шифр DES

Выполнил студент К17.1

Шишелов Владимир

**Сеть Фейстеля**

Сеть Фе́йстеля  — один из методов построения блочных шифров. Сеть представляет собой определённую многократно повторяющуюся структуру, называющуюся ячейкой Фейстеля. При переходе от одной ячейки к другой меняется ключ, причём выбор ключа зависит от конкретного алгоритма. Операции шифрования и расшифрования на каждом этапе очень просты, и при определённой доработке совпадают, требуя только обратного порядка используемых ключей. Шифрование при помощи данной конструкции легко реализуется как на программном уровне, так и на аппаратном, что обеспечивает широкие возможности применения. Большинство современных блочных шифров используют сеть Фейстеля в качестве основы.

Большинство из представленных выше алгоритмов реализованы с помощью сети Фейстеля: CAST-128, TEA, XTEA,  RC6, DES, ГОСТ 28147-89 и др..

**Шифрование**

Рассмотрим случай, когда мы хотим зашифровать некоторую информацию, представленную в двоичном виде в компьютерной памяти (например, файл), как последовательность нулей и единиц.

Вся информация разбивается на блоки фиксированной длины. В случае, если длина входного блока меньше, чем размер, который шифруется заданным алгоритмом, то блок удлиняется каким-либо способом. Как правило длина блока является степенью двойки, например: 64 бита, 128 бит. Далее будем рассматривать операции происходящие только с одним блоком, так как с другими в процессе шифрования выполняются те же самые операции.

Выбранный блок делится на два равных подблока — «левый» (L0) и «правый» (R0).

«Левый подблок» L0 видоизменяется функцией f(L0,K0) в зависимости от раундового ключа K0, после чего он складывается по модулю 2 с «правым подблоком» R0.

Результат сложения присваивается новому левому подблоку L1, который будет половиной входных данных для следующего раунда, а «левый подблок» L0 присваивается без изменений новому правому подблоку R1 (см. схему), который будет другой половиной.

После чего операция повторяется N-1 раз, при этом при переходе от одного этапа к другому меняются раундовые ключи (K0 на K1 и т. д.) по какому-либо математическому правилу, где N — количество раундов в заданном алгоритме.

**Расшифрование**

Расшифровка информации происходит так же, как и шифрование, с тем лишь исключением, что ключи идут в обратном порядке, то есть не от первого к N-ному, а от N-го к первому.

Алгоритмическое описание

1. блок открытого текста делится на 2 равные части https://lh4.googleusercontent.com/tQRWKoo0FhiXzp6NKp2riaKTX3Crz4zxbYNRGZIjpFXuH0yFT5QYlIa-kbODiaHdcyHiRLHuEG8gvp55ix5jwWJnpxsOtY__gXh8ouHYFFxgfvT18OGLUdLNWHpSoStSscsTp1g

2. в каждом раунде вычисляется (i  = 1..n  — номер раунда)

https://lh5.googleusercontent.com/1NPHllJS3o7boMuLnDO3Ip6Ssc9YaLdigdfiDNP3gWBjkOK006mrZI7POaQLGMimHtkY2zPfQj4MeGzqQNiL9cpIV0xuz9280v_RP22dpZQ-C3FprtJbw3QSSRNH3VFefrStzmU

https://lh4.googleusercontent.com/ilEU41ihtozNy47WA2e7EDLG2WRCWIEyD2Pf_7iCXOCTMzDAy17TG9wfesaQkOIUgGt54o07mM2i1HG5C6wUrVQnZnDEqfKpqjeYHHHUVHjeFybmVe-2bXoKJFEezyELGsTaD-4

где f — некоторая функция, а Ki − 1 — ключ i-го раунда.

Результатом выполнения n раундов является https://lh6.googleusercontent.com/YybOxXr1o1x391c415xq_pIBoJGXYnaO6BQ7wLCfG3AvzxU13iSurQyl7wiGY60x7pc_qMYqGULFwnK3F9O_fYIo44GTjjRpiSF1TzQOe9AzVGRgqFLmeTOcXgMBTgOF3djai9s. В n-ом раунде перестановка Ln и Rn не производится, что позволяет использовать ту же процедуру и для расшифрования, просто инвертировав порядок использования раундовой ключевой информации:

https://lh3.googleusercontent.com/bnKE-bYickJr_tL4XXCkdwwSJnc-dLVsZN1rwLFthco-fUpt1pV5q4T5rp08T8EiNxqURq1cz4L0E7xUjhfd4zfeuAfdnVTYQMixG2xjAV9Smehc2BAvl1-Vy_2IjnWQrexmX48

https://lh5.googleusercontent.com/mwcO1uNXCiCJ8J4puWsWDnL-X-2_URVNWH2vq7WPkig52a9BMd1mRAMvmHo2xwAYGMiQ8tzSD52-_Dx0Ckgy3ok36SKVMv_bNrDQ-QGqso2OTgrQLWYivtsDbbEzt1jymkshm2A

**Алгоритм DES**

DES (Data Encryption Standard) — симметричный алгоритм шифрования, разработанный фирмой IBM и утвержденный правительством США в 1977 году как официальный стандарт (FIPS 46-3). DES имеет блоки по 64 бита и 16 цикловую структуру сети Фейстеля, для шифрования использует ключ с длиной 56 бит.

**Схема шифрования алгоритма DES**

Исходный текст T — блок 64 бит.

Процесс шифрования состоит в начальной перестановке, 16 циклах шифрования и конечной перестановке.

Полученный после начальной перестановки (по таблице) 64-битовый блок IP(T) участвует в 16-циклах преобразования Фейстеля. Функция f играет роль шифрования.

Для вычисления функции f используются функция расширения Е, преобразование S и перестановка P.

Конечная перестановка IP − 1 действует на T16 и используется для восстановления позиции (по таблице). Она является обратной к перестановке IP.

**Алгоритм построения ключа**

В 16 циклах выполняются следующие операции:

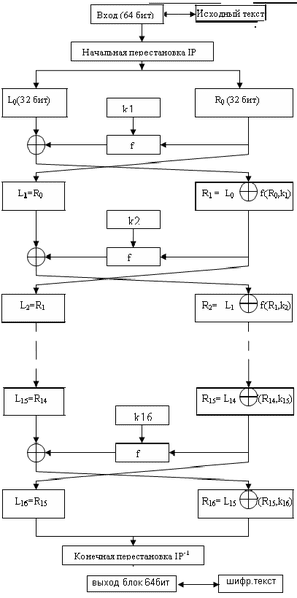
1. Ключ ki (56 бит) разбивается на два подключа kL, kR по 28 бит.

2. Каждые подключи прокручиваются влево на определенное число битов (по таблице). Данные ключи соединяются, к ним применяются некоторые действия и используются на итерации один раз.

3. Два подключа соединяются в 56 бит и процесс повторяется с п.1.

При расшифровании данных все действия выполняются в обратном порядке.

Схема шифрования алгоритм DES, выглядит следующим образом.

1. 

**Режимы использования DES**

DES может использоваться в четырёх режимах:

* Режим электронной кодовой книги (ECB — Electronic Code Book): обычное использование DES как блочного шифра. Шифруемый текст разбивается на блоки, при этом, каждый блок шифруется отдельно, не взаимодействуя с другими блоками.
* Режим сцепления блоков (СВС — Cipher Block Chaining). Каждый очередной блок Ci i>=1, перед зашифровыванием складывается по модулю 2 со следующим блоком открытого текста Mi + 1. Вектор C0 — начальный вектор, он меняется ежедневно и хранится в секрете.
* Режим обратной связи по шифротексту (англ. Cipher Feed Back). В режиме CFB используется блочная «гамма». Начальный вектор C0 является синхропосылкой и предназначен для того, чтобы разные наборы данных шифровались по-разному с использованием одного и того же секретного ключа. Синхропосылка посылается получателю в открытом виде вместе с зашифрованным файлом.
* Режим обратной связи по выходу (OFB — Output Feed Back). В режиме используется блочная гамма.

**Реализация Шифра на языке программирования С#.**

Так как алгоритм DES основан нк сети Фейстеля было принято решения обьеденить оба проекта в один.

private const int sizeOfBlock = 128; //в DES размер блока 64 бит, но поскольку в unicode символ в два раза длинее, то увеличим блок тоже в два раза

private const int sizeOfChar = 16; //размер одного символа (in Unicode 16 bit)

private const int shiftKey = 2; //сдвиг ключа

private const int quantityOfRounds = 16; //количество раундов

string[] Blocks; //сами блоки в двоичном формате

private string StringToRightLength(string input)

{

while (((input.Length \* sizeOfChar) % sizeOfBlock) != 0)

input += "#";

return input;

}

private void CutStringIntoBlocks(string input)

{

Blocks = new string[(input.Length \* sizeOfChar) / sizeOfBlock];

int lengthOfBlock = input.Length / Blocks.Length;

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

{

Blocks[i] = input.Substring(i \* lengthOfBlock, lengthOfBlock);

Blocks[i] = StringToBinaryFormat(Blocks[i]);

}

}

private void CutBinaryStringIntoBlocks(string input)

{

Blocks = new string[input.Length / sizeOfBlock];

int lengthOfBlock = input.Length / Blocks.Length;

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

Blocks[i] = input.Substring(i \* lengthOfBlock, lengthOfBlock);

}

private string StringToBinaryFormat(string input)

{

string output = "";

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

{

string char\_binary = Convert.ToString(input[i], 2);

while (char\_binary.Length < sizeOfChar)

char\_binary = "0" + char\_binary;

output += char\_binary;

}

return output;

}

private string CorrectKeyWord(string input, int lengthKey)

{

if (input.Length > lengthKey)

input = input.Substring(0, lengthKey);

else

while (input.Length < lengthKey)

input = "0" + input;

return input;

}

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)));

}

///Second round

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);

}

private string XOR(string s1, string s2)

{

string result = "";

for (int i = 0; i < s1.Length; i++)

{

bool a = Convert.ToBoolean(Convert.ToInt32(s1[i].ToString()));

bool b = Convert.ToBoolean(Convert.ToInt32(s2[i].ToString()));

if (a ^ b)

result += "1";

else

result += "0";

}

return result;

}

private string f(string s1, string s2)

{

return XOR(s1, s2);

}

private string KeyToNextRound(string key)

{

for (int i = 0; i < shiftKey; i++)

{

key = key[key.Length - 1] + key;

key = key.Remove(key.Length - 1);

}

return key;

}

private string KeyToPrevRound(string key)

{

for (int i = 0; i < shiftKey; i++)

{

key = key + key[0];

key = key.Remove(0, 1);

}

return key;

}

private string StringFromBinaryToNormalFormat(string input)

{

string output = "";

while (input.Length > 0)

{

string char\_binary = input.Substring(0, sizeOfChar);

input = input.Remove(0, sizeOfChar);

int a = 0;

int degree = char\_binary.Length - 1;

foreach (char c in char\_binary)

a += Convert.ToInt32(c.ToString()) \* (int)Math.Pow(2, degree--);

output += ((char)a).ToString();

}

return output;

}

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

{

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();

Process.Start("out1.txt");

}

else

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

}