Министерство образования республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Индивидуальная работа №1

по дисциплине «Методы защиты информации»

Студент 4 курса

Группы № 493551

Рыбак Артем Владимирович

Минск, 2018

**Содержание**

**[Введение 3](#_Toc475368072)**

**[Описание алгоритма 5](#_Toc475368073)**

**[Практическая реализация 10](#_Toc475368074)**

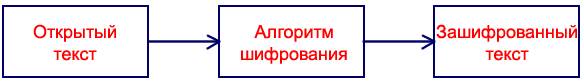
**[Список использованных источников 11](#_Toc475368075)**

**[Приложение 12](#_Toc475368076)**

**Введение**

Алгоритм шифрования это математический процесс преобразования информации в строку данных, которые выглядят как случайные.

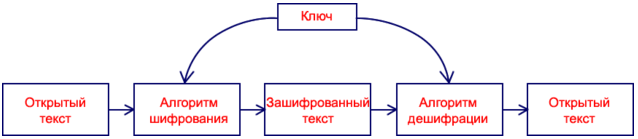
Исходные данные часто называют открытым или простым текстом, хотя для процесса шифрования не имеет значения, что представляет собой информация реальный текст или данные другого рода. Аналогично, зашифрованная информация называется зашифрованным текстом, но как правило, она мало напоминает текст. На рисунке процесс шифрования представлен в виде простой блок-схемы. Открытый текст загружается в механизм шифрования, который может быть даже механическим устройством наподобие машины Enigma, применявшейся во время второй мировой войны. В настоящее время почти все шифровальные машины это компьютерные программы. Шифровальный механизм создает зашифрованный текст.



Чтобы создать защищенный каталог, при попытке доступа к которому открывается диалоговое окно, мы воспользовались самым простым методом аутентификации, который обеспечивает сервер Apache. Этот метод шифрует пароли перед их сохранением. Мы создали пользователя с паролем password. Этот пароль был зашифрован и сохранен в виде строки aWDuA3X3H.mc2. Как видите, открытый и зашифрованный текст внешне не похожи друг на друга.

Показанный метод шифрования не является обратимым. Многие пароли сохраняются с помощью однонаправленного алгоритма шифрования. Для проверки корректности вводимого пароля расшифровывать сохраненный пароль не потребуется. Вместо этого вводимый пароль шифруется, и результат сравнивается с сохраненной версией.

Многие, но все процессы шифрования могут быть обратимыми. Обратный процесс называют дешифрацией. На рисунке ниже показан двунаправленный процесс шифрования.



История криптографии насчитывает почти 4000 лет, но наибольшего развития эта наука достигла в период второй мировой войны. С тех пор развитие криптографии повторяет развитие компьютерных сетей сначала криптография использовалась только военными и финансовыми организациями, с семидесятых годов прошлого века криптография стала шире использоваться в коммерческих компаниях, а в девяностых годах, прошлого века, криптография стала применяться практически повсеместно. За последние несколько лет криптография прошла путь от концепции, с которой обычные люди сталкиваются только в фильмах о второй мировой и в шпионских триллерах, до технологии, о которой каждый день можно прочитать в газетах, и которая применяется при каждом приобретении чего-нибудь через веб.

Существует великое множество различных алгоритмов шифрования. Некоторые, например, DES, используют секретный, или закрытый ключ. Другие, например, RSA, используют открытый и отдельный закрытый ключи.

Шифрование с закрытым ключом.

Шифрование с закрытым ключом основано на том, что доступ к ключу имеет только авторизованный персонал. Этот ключ должен держаться в секрете. Если ключ попадет в нехорошие руки, посторонний сможет получить несанкционированный доступ к зашифрованной информации. Как показано на рисунке, и отправитель, который шифрует сообщение, и получатель, который дешифрует сообщение, владеют одним и тем же ключом.

Наиболее широко используемым алгоритмом с закрытым ключом является стандарт Data Encryption Standard (DES). Этот алгоритм, разработанный компанией IBM в семидесятых годах прошлого века, принят в качестве американского стандарта для коммерческих и несекретных правительственных коммуникаций. Современные скорости вычислений на порядок превышают скорости вычислений в семидесятых годах, поэтому алгоритм DES считается устаревшим как минимум с 1998 года.

Другие известные системы шифрования с закрытым ключом это RC2, RC4, RC5, тройной DES (triple DES) и IDEA. Тройной DES-алгоритм обеспечивает достаточную степень защиты. Этот алгоритм использует тот же метод шифрования, что и DES, но применяет его трижды, используя при этом до трех разных ключей. Открытый текст шифруется с использованием первого ключа, дешифруется при помощи второго ключа, а затем шифруется с применением третьего ключа.

Явный недостаток алгоритмов с закрытым ключом состоит в том, что для отправки кому-то защищенного сообщения необходимо располагать безлопастным способом передачи этому лицу закрытого ключа. А если у вас есть безлопастный метод передачи ключа, то почему не воспользоваться этим же методом для передачи сообщений?

К счастью, в 1976 году произошел прорыв, когда Дифи и Хелман опубликовали первый алгоритм шифрования с открытым ключом.

**Описание алгоритма**

Одной из наиболее известных криптографических систем с закрытым ключом является *DES ‑ Data Encryption Standard*. Эта система первой получила статус государственного стандарта в области шифрования данных. Она разработана специалистами фирмы *IBM* и вступила в действие в США 1977 году. *Алгоритм* *DES* широко использовался при хранении и передаче данных между различными вычислительными системами; в почтовых системах, в электронных системах чертежей и при электронном обмене *коммерческой информацией*. Стандарт *DES* реализовывался как программно, так и аппаратно. Предприятиями разных стран был налажен массовый выпуск цифровых устройств, использующих *DES* для шифрования данных. Все устройства проходили обязательную сертификацию на соответствие стандарту.

Несмотря на то, что уже некоторое время эта система не имеет статуса государственного стандарта, она по-прежнему широко применяется и заслуживает внимания при изучении блочных шифров с закрытым ключом.

*Длина ключа* в алгоритме *DES* составляет 56 *бит*. Именно с этим фактом связана основная полемика относительно способности *DES* противостоять различным атакам. Как известно, любой блочный *шифр* с закрытым ключом можно взломать, перебрав все возможные комбинации ключей. При длине ключа 56 *бит* возможны 256 разных ключей. Если *компьютер* перебирает за одну секунду 1 000 000 ключей (что примерно равно 220), то на перебор всех 256 ключей потребуется 236 секунд или чуть более двух тысяч лет, что, конечно, является неприемлемым для злоумышленников.

Однако возможны более дорогие и быстрые вычислительные системы, чем *персональный компьютер*. Например, если иметь возможность объединить для проведения параллельных вычислений миллион процессоров, то максимальное время подбора ключа сокращается примерно до 18 часов. Это время не слишком велико, и криптоаналитик, оснащенный подобной дорогой техникой, вполне может выполнить вскрытие данных, зашифрованных *DES* за приемлемое для себя время.

Вместе с этим можно отметить, что систему *DES* вполне можно использовать в небольших и средних приложениях для шифрования данных, имеющих небольшую ценность. Для шифрования данных государственной важности или имеющих значительную коммерческую *стоимость* система *DES* в настоящее время, конечно, не должна использоваться. В 2001 году после специально объявленного конкурса в США был принят новый стандарт на блочный *шифр*, названный *AES (Advanced Encryption Standard)*, в основу которого был положен *шифр Rijndael*, разработанный бельгийскими специалистами.

Основные параметры *DES*: размер блока 64 бита, *длина ключа* 56 *бит*, количество раундов ‑ 16. *DES* является классической сетью Фейштеля с двумя ветвями. *Алгоритм* преобразует за несколько раундов 64-битный *входной* *блок данных* в 64-битный выходной блок. Стандарт *DES* построен на комбинированном использовании перестановки, замены и гаммирования. Шифруемые данные должны быть представлены в двоичном виде.

***Шифрование***

Общая структура *DES* представлена на рис. 1. Процесс шифрования каждого 64-битового блока исходных данных можно разделить на три этапа:

1. начальная подготовка блока данных;
2. 16 раундов «основного цикла»;
3. конечная обработка блока данных.

На первом этапе выполняется *начальная перестановка* 64-битного исходного блока текста, во время которой биты определенным образом переупорядочиваются.

На следующем (основном) этапе блок делится на две части (ветви) по 32 бита каждая. Правая *ветвь* преобразуется с использованием некоторой функции *F* и соответствующего *частичного ключа*, получаемого из основного ключа шифрования по специальному алгоритму преобразования ключей. Затем производится *обмен данными* между левой и правой ветвями блока. Это повторяется в цикле 16 раз.

Наконец, на третьем этапе выполняется *перестановка* результата, полученного после шестнадцати шагов основного *цикла*. Эта *перестановка* обратна начальной перестановке.

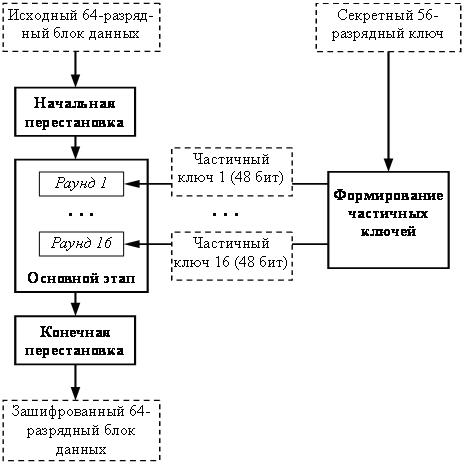


Рисунок 1 ‑ Общая схема DES

Рассмотрим более подробно все этапы криптографического преобразования по стандарту *DES*.

На первом этапе 64-разрядный блок исходных данных подвергается начальной перестановке. В литературе эта операция иногда называется «забеливание» ‑ *whitening*. При начальной перестановке биты блока данных определенным образом переупорядочиваются. Эта операция придает некоторую «хаотичность» исходному сообщению, снижая возможность использования криптоанализа статистическими методами.

Одновременно с начальной перестановкой блока данных выполняется начальная *перестановка* 56 *бит* ключа. Из рис. 2 видно, что в каждом из раундов используется соответствующий 48-битный частичный *ключ* Ki. Ключи Ki получаются по определенному алгоритму, используя каждый из битов начального ключа по нескольку раз. В каждом раунде 56-битный *ключ* делится на две 28-битовые половинки. Затем половинки сдвигаются влево на один или два бита в зависимости от номера раунда. После сдвига определенным образом выбирается 48 из 56 битов. Так как при этом не только выбирается *подмножество* битов, но и изменяется их порядок, то эта операция называется «*перестановка* со сжатием». Ее результатом является набор из 48 битов. В среднем каждый *бит* исходного 56-битного ключа используется в 14 из 16 подключей, хотя не все биты используются одинаковое количество раз.

Далее выполняется основной цикл преобразования, организованный по сети Фейштеля и состоящий из 16 одинаковых раундов. При этом в каждом раунде (рис. 2) получается промежуточное 64-битное *значение*, которое затем обрабатывается в следующем раунде.

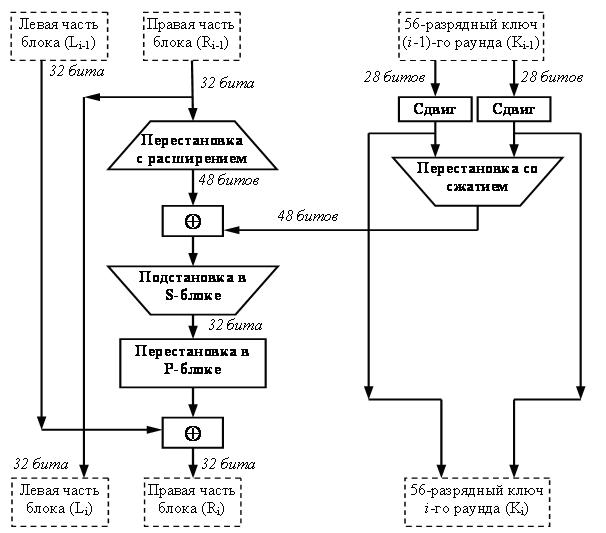


Рисунок 2 ‑ Структура одного раунда DES

Левая и правая ветви каждого промежуточного значения обрабатываются как отдельные 32-битные значения, обозначенные L и R.

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»). Чем сильнее проявляется лавинный эффект при использовании того или иного алгоритма шифрования, тем лучше.

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-битовых подблоков, каждый из которых по своей таблице замен заменяется четырьмя битами. *Подстановка* с помощью S-блоков является одним из важнейших этапом *DES*. Таблицы замен для этой *операции* специально спроектированы специалистами так, чтобы обеспечивать максимальную *безопасность*. В результате выполнения этого этапа получаются восемь 4-битовых блоков, которые вновь объединяются в единое 32-битовое *значение*.

Далее полученное 32-битовое *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.

После шестнадцати раундов шифрования выполняется конечная *перестановка* результата. Эта *перестановка* инверсна (обратна) начальной перестановке.

После выполнения всех указанных шагов *блок данных* считается полностью зашифрованным и можно переходить к шифрованию следующего блока исходного сообщения.

***Расшифрование***

Как известно, *криптографическая система* должна позволять не только зашифровать, но и расшифровать сообщения. Можно было бы ожидать, что процесс расшифрования по *DES* сильно запутан. Однако разработчики так подобрали различные компоненты стандарта, чтобы для зашифрования и расшифрования использовался один и тот же *алгоритм*. При расшифровании на *вход алгоритма* подается зашифрованный текст. Единственное отличие состоит в обратном порядке использования частичных ключей Ki. K16 используется на первом раунде, K1 на последнем раунде.

После последнего раунда процесса расшифрования две половины выхода меняются местами так, чтобы вход заключительной перестановки был составлен из R16 и L16. Выходом этой стадии является незашифрованный текст.

**Практическая реализация**

Исходный файл показан на рисунке 3.

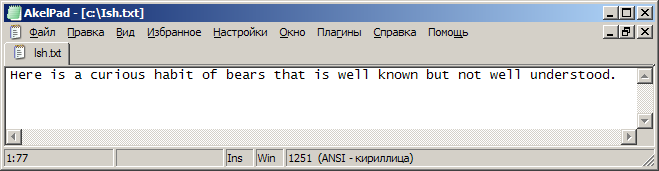


Рисунок 3 ‑ Исходный файл

Процесс шифрования и дешифрования показан на рисунке 4.

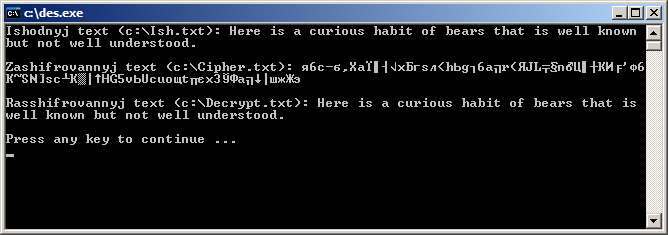


Рисунок 4 ‑ Результат работы программы

Шифрованный файл показан на рисунке 5.

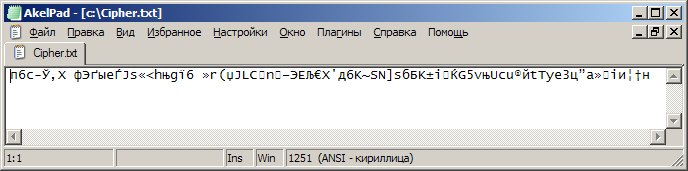


Рисунок 5 ‑ Шифрованный файл

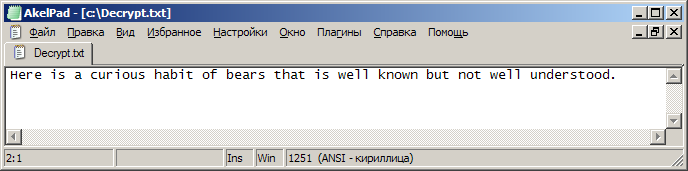


Рисунок 6 ‑ Дешифрованный файл

**Список использованных источников**

1. Харин Ю.С., Берник В.И., Матвеев Г.В., Агиевич С.В. Математические и компьютерные основы криптологии. - Минск: Новое Знание, 2003.
2. Брассар Ж. Современная криптология. - М.: издательско-полиграфическая фирма ПОЛИМЕД, 1999.
3. Блейхут Р. Теория и практика кодов, кодирующих ошибки. - М.: Мир, 1986.
4. Виноградов В.В. Основы теории чисел. - М.: Наука, 1981.
5. Венбо М. Современная критпография. Теория и практика. М.: Издательский дом «Вильямс», 2005.
6. Молдовян H.A., Молдовян А.А. Введение в криптосистемы с открытым ключом: Проблематика криптографии; элементы теории чисел; двухключевые криптосистемы и др.: Учебное пособие для вузов.- СПб.: Издательство БВХ-Петербург, 2005.
7. Кнут Д. Искусство программирования для ЭВМ: В 3-х томах. Получисленные методы. Перевод с английского. - М.: Мир. Т.2. 724с.
8. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. - М.: Радио и связь, 1999.
9. Шнайер Б., Фергюссон H., Практическая криптография. - М.: компьютерное издательство «Диалектика», 2005.

**Приложение**

*#include <stdio.h>*

*#include <fstream.h>*

*#include <string.h>*

*#include <conio.h>*

*#include <iostream.h>*

*const*

*N = 100000;*

*int key[64]={*

*0,0,0,1,0,0,1,1,*

*0,0,1,1,0,1,0,0,*

*0,1,0,1,0,1,1,1,*

*0,1,1,1,1,0,0,1,*

*1,0,0,1,1,0,1,1,*

*1,0,1,1,1,1,0,0,*

*1,1,0,1,1,1,1,1,*

*1,1,1,1,0,0,0,1*

*};*

*class Des{*

*public:*

*int keyi[16][48],total[64],left[32],right[32],ck[28],dk[28],expansion[48],z[48],xor1[48],sub[32],p[32],xor2[32],temp[64],*

*pc1[56],ip[64],inv[8][8];*

*char final[N];*

*void IP();*

*void PermChoice1();*

*void PermChoice2();*

*void Expansion();*

*void inverse();*

*void xor\_two();*

*void xor\_oneE(int);*

*void xor\_oneD(int);*

*void substitution();*

*void permutation();*

*void keygen();*

*char \* Encrypt(char \*);*

*char \* Decrypt(char \*);*

*};*

*void Des::IP(){*

*int k = 58, i;*

*for(i=0; i < 32; i++){*

*ip[i] = total[k-1];*

*if (k - 8 > 0)*

*k= k - 8;*

*else*

*k= k + 58;*

*}*

*k = 57;*

*for(i=32; i < 64; i++){*

*ip[i] = total[k-1];*

*if(k-8 > 0)*

*k = k - 8;*

*else*

*k = k + 58;*

*}*

*}*

*void Des::PermChoice1(){*

*int k = 57, i;*

*for(i = 0; i < 28; i++){*

*pc1[i]=key[k-1];*

*if(k-8 > 0)*

*k=k-8;*

*else*

*k=k+57;*

*}*

*k = 63;*

*for(i = 28; i < 52; i++){*

*pc1[i] = key[k-1];*

*if(k-8 > 0)*

*k=k-8;*

*else*

*k=k+55;*

*}*

*k = 28;*

*for(i=52; i < 56; i++){*

*pc1[i] = key[k-1];*

*k= k - 8;*

*}*

*}*

*void Des::Expansion(){*

*int exp[8][6], i, j, k;*

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

*for(j = 0; j < 6; j++){*

*if ((j != 0)||(j != 5)){*

*k = 4 \* i + j;*

*exp[i][j] = right[k-1];*

*}*

*if(j == 0){*

*k = 4 \* i;*

*exp[i][j] = right[k-1];*

*}*

*if(j == 5){*

*k= 4\*i + j;*

*exp[i][j] = right[k-1];*

*}*

*}*

*}*

*exp[0][0] = right[31];*

*exp[7][5] = right[0];*

*k = 0;*

*for(i = 0; i < 8; i++)*

*for(j = 0; j < 6; j++)*

*expansion[k++] = exp[i][j];*

*}*

*void Des::PermChoice2(){*

*int per[56], i, k;*

*for(i = 0; i < 28; i++)*

*per[i]=ck[i];*

*for(k = 0, i = 28; i < 56; i++)*

*per[i] = dk[k++];*

*z[0] = per[13]; z[1] = per[16]; z[2] = per[10]; z[3] = per[23]; z[4] = per[0]; z[5] = per[4]; z[6] = per[2]; z[7] = per[27];*

*z[8] = per[14]; z[9] = per[5]; z[10] = per[20]; z[11] = per[9]; z[12] = per[22]; z[13] = per[18]; z[14] = per[11]; z[15] = per[3];*

*z[16] = per[25]; z[17] = per[7]; z[18] = per[15]; z[19] = per[6]; z[20] = per[26]; z[21] = per[19]; z[22] = per[12]; z[23] = per[1];*

*z[24] = per[40]; z[25] = per[51]; z[26] = per[30]; z[27] = per[36]; z[28] = per[46]; z[29] = per[54]; z[30] = per[29]; z[31] = per[39];*

*z[32] = per[50]; z[33] = per[46]; z[34] = per[32]; z[35] = per[47]; z[36] = per[43]; z[37] = per[48]; z[38] = per[38]; z[39] = per[55];*

*z[40] = per[33]; z[41] = per[52]; z[42] = per[45]; z[43] = per[41]; z[44] = per[49]; z[45] = per[35]; z[46] = per[28]; z[47] = per[31];*

*}*

*void Des::xor\_oneE(int round){*

*int i;*

*for(i = 0; i < 48; i++)*

*xor1[i] = expansion[i]^keyi[round-1][i];*

*}*

*void Des::xor\_oneD(int round){*

*int i;*

*for(i = 0; i < 48; i++)*

*xor1[i] = expansion[i]^keyi[16-round][i];*

*}*

*void Des::substitution(){*

*int s1[4][16]={14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,*

*0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,*

*4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,*

*15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13*

*};*

*int s2[4][16]={15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,*

*3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,*

*0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,*

*13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9*

*};*

*int s3[4][16]={10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,*

*13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1,*

*13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7,*

*1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12*

*};*

*int s4[4][16]={7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,*

*13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9,*

*10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4,*

*3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14*

*};*

*int s5[4][16]={2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,*

*14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6,*

*4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14,*

*11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3*

*};*

*int s6[4][16]={12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,*

*10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8,*

*9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6,*

*4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13*

*};*

*int s7[4][16]={4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,*

*13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6,*

*1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2,*

*6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12*

*};*

*int s8[4][16]={13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,*

*1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,*

*7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,*

*2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11*

*};*

*int a[8][6], k = 0, i, j, p, q, count = 0, g = 0, v;*

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

*for(j = 0; j < 6; j++){*

*a[i][j] = xor1[k++];*

*}*

*}*

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

*p = 1; q = 0;*

*k = (a[i][0]\*2) + (a[i][5]\*1);*

*j = 4;*

*while(j > 0){*

*q = q + (a[i][j]\*p);*

*p = p\*2;*

*j--;*

*}*

*count = i + 1;*

*switch(count){*

*case 1: v = s1[k][q]; break;*

*case 2: v = s2[k][q]; break;*

*case 3: v = s3[k][q]; break;*

*case 4: v = s4[k][q]; break;*

*case 5: v = s5[k][q]; break;*

*case 6: v = s6[k][q]; break;*

*case 7: v = s7[k][q]; break;*

*case 8: v = s8[k][q]; break;*

*}*

*int d, i = 3, a[4];*

*while(v > 0){*

*d = v%2;*

*a[i--] = d;*

*v = v/2;*

*}*

*while(i >= 0){*

*a[i--] = 0;*

*}*

*for(i = 0; i < 4; i++)*

*sub[g++] = a[i];*

*}*

*}*

*void Des::permutation(){*

*p[0] = sub[15]; p[1] = sub[6]; p[2] = sub[19]; p[3] = sub[20]; p[4] = sub[28]; p[5] = sub[11]; p[6] = sub[27]; p[7] = sub[16];*

*p[8] = sub[0]; p[9] = sub[14]; p[10]= sub[22]; p[11]= sub[25]; p[12]= sub[4]; p[13]= sub[17]; p[14]= sub[30]; p[15]= sub[9];*

*p[16]= sub[1]; p[17]= sub[7]; p[18]= sub[23]; p[19]= sub[13]; p[20]= sub[31]; p[21]= sub[26]; p[22]= sub[2]; p[23]= sub[8];*

*p[24]= sub[18]; p[25]= sub[12]; p[26]= sub[29]; p[27]= sub[5]; p[28]= sub[21]; p[29]= sub[10]; p[30]= sub[3]; p[31]= sub[24];*

*}*

*void Des::xor\_two(){*

*int i;*

*for(i = 0; i < 32; i++){*

*xor2[i] = left[i]^p[i];*

*}*

*}*

*void Des::inverse(){*

*int p = 40, q = 8, k1, k2, i, j;*

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

*k1 = p; k2 = q;*

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

*if(j%2 == 0){*

*inv[i][j] = temp[k1-1];*

*k1 = k1 + 8;*

*}*

*else if(j%2 != 0){*

*inv[i][j] = temp[k2-1];*

*k2 = k2 + 8;*

*}*

*}*

*p = p - 1;*

*q = q - 1;*

*}*

*}*

*char \*Des::Encrypt(char \*Text1){*

*int i, a1, j, nB, m, iB, k, K, B[8], n, t, d, round;*

*char \*Text = new char[N];*

*strcpy(Text, Text1);*

*i = strlen(Text);*

*int mc = 0;*

*a1 = i%8;*

*if (a1 != 0)*

*for(j = 0; j < 8-a1; j++, i++)*

*Text[i] = ' ';*

*Text[i] = '\0';*

*keygen();*

*for(iB = 0, nB = 0, m = 0; m < (strlen(Text)/8); m++){*

*for(iB = 0, i = 0; i < 8; i++, nB++){*

*n = (int)Text[nB];*

*for(K = 7; n >= 1; K--){*

*B[K] = n%2;*

*n/= 2;*

*}*

*for(; K >= 0; K--)*

*B[K] = 0;*

*for(K = 0; K < 8; K++, iB++)*

*total[iB] = B[K];*

*}*

*IP();*

*for(i = 0; i < 64; i++)*

*total[i] = ip[i];*

*for(i = 0; i < 32; i++)*

*left[i] = total[i];*

*for(; i < 64; i++)*

*right[i-32] = total[i];*

*for(round = 1; round <= 16; round++){*

*Expansion();*

*xor\_oneE(round);*

*substitution();*

*permutation();*

*xor\_two();*

*for(i = 0; i < 32; i++)*

*left[i] = right[i];*

*for(i = 0; i < 32; i++)*

*right[i] = xor2[i];*

*}*

*for(i = 0; i < 32; i++)*

*temp[i] = right[i];*

*for(; i < 64; i++)*

*temp[i] = left[i-32];*

*inverse();*

*k = 128;*

*d = 0;*

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

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

*d = d + inv[i][j] \* k;*

*k= k/2;*

*}*

*final[mc++] = (char)d;*

*k = 128;*

*d = 0;*

*}*

*}*

*final[mc] = '\0';*

*return(final);*

*}*

*char \* Des::Decrypt(char \*Text1){*

*int i, a1, j, nB, m, iB, k, K, B[8], n, t, d, round;*

*char \*Text = new char[N];*

*unsigned char ch;*

*strcpy(Text, Text1);*

*i = strlen(Text);*

*keygen();*

*int mc = 0;*

*for(iB = 0, nB = 0, m = 0; m < (strlen(Text)/8); m++){*

*for(iB = 0, i = 0; i < 8; i++, nB++){*

*ch = Text[nB];*

*n = (int)ch;*

*for(K = 7; n >= 1; K--){*

*B[K] = n%2;*

*n/= 2;*

*}*

*for(; K >= 0; K--) B[K] = 0;*

*for(K = 0; K < 8; K++, iB++) total[iB]=B[K];*

*}*

*IP();*

*for(i = 0; i < 64; i++) total[i] = ip[i];*

*for(i = 0; i < 32; i++) left[i] = total[i];*

*for(; i < 64; i++) right[i-32] = total[i];*

*for(round = 1; round <= 16; round++){*

*Expansion();*

*xor\_oneD(round);*

*substitution();*

*permutation();*

*xor\_two();*

*for(i = 0; i < 32; i++) left[i]=right[i];*

*for(i = 0; i < 32; i++) right[i]=xor2[i];*

*}*

*for(i = 0; i < 32; i++) temp[i] = right[i];*

*for(; i < 64; i++) temp[i]=left[i-32];*

*inverse();*

*k = 128;*

*d = 0;*

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

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

*d = d + inv[i][j] \* k;*

*k = k/2;*

*}*

*final[mc++] = (char)d;*

*k = 128;*

*d = 0;*

*}*

*}*

*final[mc] = '\0';*

*char \*final1 = new char[N];*

*for(i = 0, j = strlen(Text); i < strlen(Text); i++, j++)*

*final1[i] = final[j];*

*final1[i] = '\0';*

*return(final);*

*}*

*void main()*

*{*

*Des d1,d2;*

*char \*str = new char[N];*

*char \*str1 = new char[N];*

*char \*str2 = new char[N];*

*int i = 0;*

*char ch;*

*FILE \*fp;*

*fp = fopen("c:\\Ish.txt", "r");*

*ch = getc(fp);*

*str[i++] = ch;*

*while (ch != EOF) {*

*ch = getc(fp);*

*str[i++] = ch;*

*}*

*str[i] = '\0';*

*fclose(fp);*

*str1 = d1.Encrypt(str);*

*cout<<"Ishodnyj text (c:\\Ish.txt): "<< str<< endl;*

*cout<<"\nZashifrovannyj text (c:\\Cipher.txt): "<< str1 << endl;*

*fp = fopen("c:\\Cipher.txt", "w");*

*i = 0;*

*while (str1[i] != '\0'){*

*putc(str1[i], fp);*

*i++;*

*}*

*putc('\n', fp);*

*fclose(fp);*

*i = 0;*

*fp = fopen("c:\\Cipher.txt", "r");*

*ch = getc(fp);*

*str2[i++] = ch;*

*while (ch != EOF) {*

*ch = getc(fp);*

*str2[i++] = ch;*

*}*

*str2[i] = '\0';*

*fclose(fp);*

*cout<<"\nRasshifrovannyj text (c:\\Decrypt.txt): "<< d2.Decrypt(str2)<< endl;*

*str1 = d2.Decrypt(str2);*

*fp = fopen("c:\\Decrypt.txt", "w");*

*i = 0;*

*while (str1[i] != '\0'){*

*putc(str1[i], fp);*

*i++;*

*}*

*putc('\n', fp);*

*fclose(fp);*

*cout << "\nPress any key to continue ..." << endl;*

*getch();*

*}*

*void Des::keygen(){*

*PermChoice1();*

*int i, j, k = 0;*

*for(i = 0; i < 28; i++) ck[i] = pc1[i];*

*for(i = 28; i < 56; i++){*

*dk[k] = pc1[i];*

*k++;*

*}*

*int noshift = 0, round;*

*for(round = 1; round <= 16; round++){*

*if(round == 1||round == 2||round == 9||round == 16)*

*noshift = 1;*

*else*

*noshift = 2;*

*while(noshift > 0){*

*int t;*

*t = ck[0];*

*for(i = 0; i < 28; i++)*

*ck[i] = ck[i+1];*

*ck[27] = t;*

*t = dk[0];*

*for(i = 0; i < 28; i++)*

*dk[i] = dk[i+1];*

*dk[27] = t;*

*noshift--;*

*}*

*PermChoice2();*

*for(i = 0; i < 48; i++)*

*keyi[round-1][i] = z[i];*

*}*

*}*