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

**Алгоритмы шифрования DES и ГОСТ**

**Введение**

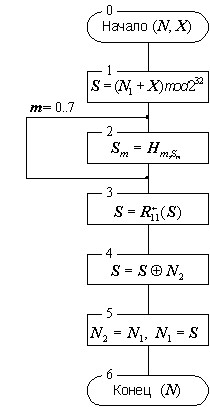
Практическая работа № 1 подразумевает реализацию алгоритмов симметричного шифрования DES и ГОСТ 28147.

Целью работы является создание программы шифрования, осуществляющей криптографическое преобразование введенного текста с помощью алгоритмов симметричного шифрования DES и ГОСТ 28147.

**Блок-схема алгоритма**

Схема DES

Структура одного раунда DES



ГОСТ 28147

**Результаты ввода данных и исполнения программы**

Исходный файл показан на рисунке 1.

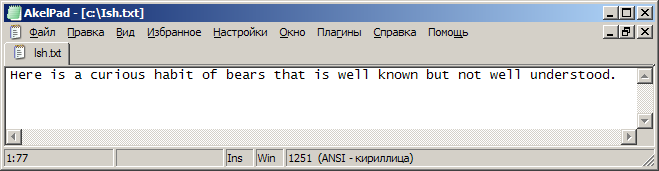


Рисунок 1 ‑ Исходный файл

Процесс шифрования и дешифрования показан на рисунке 2.

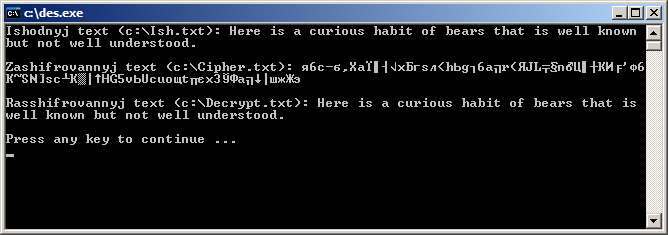


Рисунок 2 ‑ Результат работы программы

Шифрованный файл показан на рисунке 3.

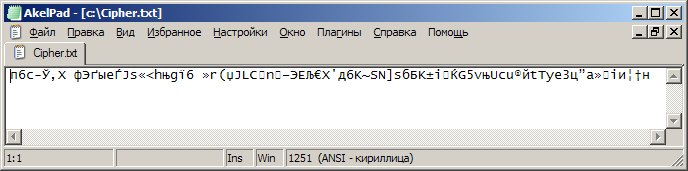


Рисунок 3 ‑ Шифрованный файл

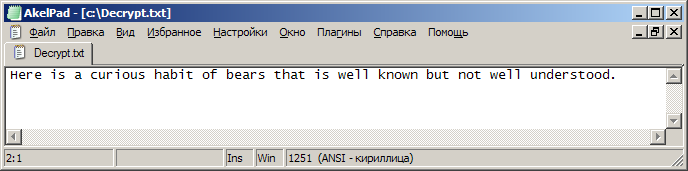


Рисунок 4 ‑ Дешифрованный файл

**Программный код**

*#include <stdio.h>*

*#include <string.h>*

*#include <conio.h>*

*#include "stdafx.h"*

*const int 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*

*};*

*void main()*

*{*

*Crypter\_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:\\Work\\text.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:\\Work\\text.txt): " << str << endl;*

*cout << "\nZashifrovannyj text (c:\\Work\\Encrypt.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:\\Work\\Encrypt.txt", "r");*

*ch = getc(fp);*

*str2[i++] = ch;*

*while (ch != EOF) {*

*ch = getc(fp);*

*str2[i++] = ch;*

*}*

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

*fclose(fp);*

*cout << "\nResult: (c:\\Work\\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();*

*}*

*class Crypter\_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 Crypter\_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 Crypter\_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 Crypter\_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 Crypter\_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 Crypter\_Des::xor\_oneE(int round) {*

*int i;*

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

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

*}*

*void Crypter\_Des::xor\_oneD(int round) {*

*int i;*

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

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

*}*

*void Crypter\_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 Crypter\_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 Crypter\_Des::xor\_two() {*

*int i;*

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

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

*}*

*}*

*void Crypter\_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 \*Crypter\_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 \*Crypter\_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 Crypter\_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];*

*}*

*}*

**Вывод**

Алгоритм DES широко применяется для защиты финансовой информации: так, модуль THALES (Racal) HSM RG7000 полностью поддерживает операции TripleDES для эмиссии и обработки кредитных карт VISA, EuroPay и проч.

В настоящее время DES используется (с ключом длины 56 бит) только для устаревших систем, чаще всего используют его более криптоустойчивый вид (3DES, DESX).

Основные проблемы стандарта **28147** связаны с неполнотой стандарта в части генерации ключей и таблиц замен. Считается, что у стандарта существуют «слабые» ключи и таблицы замен, но в стандарте не описываются критерии выбора и отсева «слабых».