Государственное образовательное учреждение
высшего профессионального образования

«Петербургский государственный университет путей сообщения»

Кафедра «Информационные и вычислительные системы»

Лабораторная работа № 2

По курсу «Методы и средства защиты информации»

Изучение криптографических возможностей ОС Windows. Симметричное шифрование

Вариант № 6

Скачано с сайта http://ivc.clan.su

Выполнил:
студент группы ПВТ-711
Круглов В. А.

Проверил:

САНКТ-ПЕТЕРБУРГ
2011

Листинг приложения

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Wcrypt2, ExtCtrls;

type
  ByteArray = array of byte;
  TMainForm = class(TForm)
    GroupBox2: TGroupBox;
    Panel1: TPanel;
    Label5: TLabel;
    GenerateKeys: TButton;
    ePassword: TEdit;
    GridPanel4: TGridPanel;
    eSource: TMemo;
    eCrypt: TMemo;
    eResult: TMemo;
    Splitter1: TSplitter;
    GridPanel3: TGridPanel;
    Providers: TGroupBox;
    PovidersList: TListBox;
    RefreshProviders: TButton;
    GroupBox1: TGroupBox;
    GridPanel1: TGridPanel;
    Label1: TLabel;
    eProviderName: TEdit;
    Label2: TLabel;
    eProviderVersion: TEdit;
    Label3: TLabel;
    eProviderType: TEdit;
    Label4: TLabel;
    eProviderAlgorithms: TListBox;
    procedure RefreshProvidersClick(Sender: TObject);
    procedure PovidersListClick(Sender: TObject);
    procedure GenerateKeysClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

procedure TMainForm.GenerateKeysClick(Sender: TObject);
var
  ProviderDescriptor, KeyDescriptor, HashDescriptor: Cardinal;
```

```
  ProviderName: Array [0..59] of WideChar;
  Buffer: ByteArray;
  BufferSize, NewSize, i: Integer;
  Text, Result: AnsiString;
begin
  ProviderName:='Microsoft Base Cryptographic Provider v1.0';
  if not
CryptAcquireContext(@ProviderDescriptor,nil,@(ProviderName[0]),PROV_RSA_FULL,
CRYPT_VERIFYCONTEXT) then
  begin
    ShowMessage('Provider not found!');
    Exit;
  end;
  if not CryptCreateHash(ProviderDescriptor,CALG_MD5,0,0,@HashDescriptor)
then
  begin
    CryptReleaseContext(ProviderDescriptor,0);
    ShowMessage('Failed to create hash!');
    Exit;
  end;
  Text:=ePassword.Text;
  SetLength(Buffer,Length(ByteArray(Text)));
  Move(ByteArray(Text)[0],Buffer[0],Length(ByteArray(Text)));
  if not CryptHashData(HashDescriptor,@(Buffer[0]),Length(Buffer),0) then
  begin
    CryptReleaseContext(ProviderDescriptor,0);
    CryptDestroyHash(HashDescriptor);
    ShowMessage('Failed to make hash!');
    Exit;
  end;
  if not
CryptDeriveKey(ProviderDescriptor,CALG_RC2,HashDescriptor,CRYPT_EXPORTABLE,@K
eyDescriptor) then
  begin
    CryptReleaseContext(ProviderDescriptor,0);
    CryptDestroyHash(HashDescriptor);
    ShowMessage('Failed to create key!');
    Exit;
  end;
  CryptDestroyHash(HashDescriptor);
  Text:=eSource.Text;
  SetLength(Buffer,Length(ByteArray(Text)));
  Move(ByteArray(Text)[0],Buffer[0],Length(ByteArray(Text)));
  BufferSize:=Length(Buffer);
  NewSize:=BufferSize;
  if not CryptEncrypt(KeyDescriptor,0,true,0,nil,@NewSize,0) then
  begin
    CryptReleaseContext(ProviderDescriptor,0);
    CryptDestroyKey(KeyDescriptor);
    ShowMessage('kernel panic!');
    Exit;
  end;
  if NewSize>BufferSize then SetLength(Buffer,NewSize);
  if not
CryptEncrypt(KeyDescriptor,0,true,0,@(Buffer[0]),@BufferSize,NewSize) then
  begin
```

```
    CryptReleaseContext(ProviderDescriptor,0);
    CryptDestroyKey(KeyDescriptor);
    ShowMessage('Encrypting failed!');
    Exit;
  end;
  SetLength(Result,3*NewSize);
  for i := 0 to NewSize do
  begin
    Result[3*i+1]:=AnsiChar((IntToHex(Buffer[i],2))[1]);
    Result[3*i+2]:=AnsiChar((IntToHex(Buffer[i],2))[2]);
    Result[3*i+3]:=' ';
  end;
  eCrypt.Text:=Result;
  if not CryptDecrypt(KeyDescriptor,0,true,0,@(Buffer[0]),@NewSize) then
  begin
    CryptReleaseContext(ProviderDescriptor,0);
    CryptDestroyKey(KeyDescriptor);
    ShowMessage('Dencrypting failed!');
    Exit;
  end;
  SetLength(Result,NewSize);
  Move(Buffer[0],Result[1],NewSize);
  eResult.Text:=Result;
  CryptDestroyKey(KeyDescriptor);
  CryptReleaseContext(ProviderDescriptor,0);
end;

procedure TMainForm.PovidersListClick(Sender: TObject);
var
  ProviderDescriptor: Cardinal;
  ProviderName: Array of WideChar;
  ProviderVersion: Array of Byte;
  ProviderType: Array of Byte;
  ProviderAlgorithm: Array of Byte;
  BufferSize: Integer;
  Buffer: String;
  Algorithm: AnsiString;
  i: Integer;
begin
  eProviderName.Text:=PovidersList.Items.Strings[PovidersList.ItemIndex];
  SetLength(ProviderName,Length(eProviderName.Text)+2);
  for i := 0 to Length(eProviderName.Text)-1 do
    ProviderName[i]:=(String(eProviderName.Text))[i+1];
  if not
CryptAcquireContext(@ProviderDescriptor,nil,@(ProviderName[0]),PROV_RSA_FULL,
CRYPT_VERIFYCONTEXT) then
  begin
    ShowMessage('Provider not found!');
    Exit;
  end;
  BufferSize:=4;
  SetLength(ProviderVersion, BufferSize);
  if not
CryptGetProvParam(ProviderDescriptor,PP_VERSION,@(ProviderVersion[0]),@Buffer
Size,0) then
  begin
```

```
    CryptReleaseContext(ProviderDescriptor,0);
    eProviderVersion.Text:='--- ERROR ---';
    Exit;
  end;
  Buffer:='';
  for i := 0 to BufferSize-1 do
    Buffer:=Buffer+IntToStr(ProviderVersion[i])+' ';
  eProviderVersion.Text:=Buffer;
  BufferSize:=4;
  SetLength(ProviderType, BufferSize);
  if not
CryptGetProvParam(ProviderDescriptor,PP_IMPTYPE,@(ProviderType[0]),@BufferSiz
e,0) then
  begin
    CryptReleaseContext(ProviderDescriptor,0);
    eProviderType.Text:='--- ERROR ---';
    Exit;
  end;
  Buffer:='';
  for i := 0 to BufferSize-1 do
    Buffer:=Buffer+IntToStr(ProviderType[i])+' ';
  eProviderType.Text:=Buffer;
  eProviderAlgorithms.Items.Clear;
  if not
CryptGetProvParam(ProviderDescriptor,PP_ENUMALGS,nil,@BufferSize,CRYPT_FIRST)
then
  begin
    CryptReleaseContext(ProviderDescriptor,0);
    eProviderAlgorithms.Items.Add('--- ERROR ---');
    Exit;
  end;
  SetLength(ProviderAlgorithm, BufferSize);
  while
CryptGetProvParam(ProviderDescriptor,PP_ENUMALGS,@(ProviderAlgorithm[0]),@Buf
ferSize,0) do
  begin
    SetLength(Algorithm,Length(ProviderAlgorithm)-12);

Move(ProviderAlgorithm[12],ByteArray(Algorithm)[0],Length(ProviderAlgorithm)-
11);
    eProviderAlgorithms.Items.Add(Algorithm);
  end;
  CryptReleaseContext(ProviderDescriptor,0);
end;

procedure TMainForm.RefreshProvidersClick(Sender: TObject);
var
  ProviderType: Integer;
  ProviderName: Array of WideChar;
  BufferSize: Integer;
  i: Cardinal;
begin
  i:=0;
  PovidersList.Items.Clear;
  while CryptEnumProviders(i,nil,0,@ProviderType,nil,@BufferSize) do
  begin
```

```
    SetLength(ProviderName, BufferSize);
    if not
CryptEnumProviders(i,nil,0,@ProviderType,@(ProviderName[0]),@BufferSize) then
    begin
      PovidersList.Items.Add('--- ERROR ---');
      Break;
    end;
    PovidersList.Items.Add(WideString(ProviderName));
    Inc(i);
  end;
end;

end.
```

Краткие выводы