Государственное образовательное учреждение
высшего профессионального образования

«Петербургский государственный университет путей сообщения»

Кафедра «Информационные и вычислительные системы»

Лабораторная работа № 3

По курсу «Методы и средства защиты информации»

Использование ассиметричных алгоритмов криптографического преобразования информации в приложениях Windows

Вариант № 6

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

Проверил:

САНКТ-ПЕТЕРБУРГ
2011

Листинг приложения

SERVER

```
unit Server;

interface

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

type
  ByteArray = array of byte;
  TMainForm = class(TForm)
    GroupBox1: TGroupBox;
    ePath: TEdit;
    btnSelectPath: TSpeedButton;
    GroupBox2: TGroupBox;
    eMessage: TMemo;
    GridPanel1: TGridPanel;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure btnSelectPathClick(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
    Key, ProviderDescriptor: Cardinal;
  public
    { Public declarations }
  end;

var
  MainForm: TMainForm;

implementation

uses Wcrypt2;

{$R *.dfm}

procedure TMainForm.Button1Click(Sender: TObject);
var
  KeyDescriptor: Cardinal;
  BufferSize: Integer;
  Buffer: ByteArray;
  f: file of byte;
begin
  if not CryptGenKey(ProviderDescriptor, AT_KEYEXCHANGE, $02000000,
@KeyDescriptor) then
  begin
    ShowMessage('Key generation failed!');
    Exit;
  end;
```

```
  Key:=KeyDescriptor;
  if not CryptExportKey(KeyDescriptor, 0, PUBLICKEYBLOB, 0, nil, @BufferSize)
then
  begin
    CryptDestroyKey(KeyDescriptor);
    ShowMessage('kernel panic!');
    Exit;
  end;
  SetLength(Buffer,BufferSize);
  if not CryptExportKey(KeyDescriptor, 0, PUBLICKEYBLOB, 0, @(Buffer[0]),
@BufferSize) then
  begin
    CryptDestroyKey(KeyDescriptor);
    ShowMessage('Public key creation failed!');
    Exit;
  end;
  AssignFile(f,ePath.Text+'key');
  Rewrite(f);
  BlockWrite(f,Buffer[0],BufferSize);
  CloseFile(f);
end;

procedure TMainForm.Button2Click(Sender: TObject);
var
  KeyDescriptor, HashDescriptor: Cardinal;
  f: file of byte;
  Buffer, Signature: ByteArray;
  BufferSize, NewSize: Integer;
  Text: AnsiString;
begin
  AssignFile(f,ePath.Text+'key');
  Reset(f);
  BufferSize:=FileSize(f);
  SetLength(Buffer,BufferSize);
  BlockRead(f,Buffer[0],BufferSize);
  CloseFile(f);
  if not
CryptImportKey(ProviderDescriptor,@(Buffer[0]),BufferSize,Key,0,@KeyDescripto
r) then
  begin
    ShowMessage('Public key creation failed!');
    Exit;
  end;
  CryptDestroyKey(Key);
  Key:=KeyDescriptor;
  if not CryptGenKey(ProviderDescriptor, AT_SIGNATURE, $02000000,
@KeyDescriptor) then
  begin
    ShowMessage('Signature key generation failed!');
    Exit;
  end;
  if not CryptExportKey(KeyDescriptor, 0, PUBLICKEYBLOB, 0, nil, @BufferSize)
then
  begin
    CryptDestroyKey(KeyDescriptor);
    ShowMessage(IntToStr(GetLastError));
```

```
    ShowMessage('kernel panic!');
    Exit;
  end;
  SetLength(Buffer,BufferSize);
  if not CryptExportKey(KeyDescriptor, 0, PUBLICKEYBLOB, 0, @(Buffer[0]),
@BufferSize) then
  begin
    CryptDestroyKey(KeyDescriptor);
    ShowMessage('Public key creation failed!');
    Exit;
  end;
  BufferSize:=Length(Buffer);
  NewSize:=BufferSize;
  if not CryptEncrypt(Key,0,true,0,nil,@NewSize,0) then
  begin
    ShowMessage('kernel panic!');
    Exit;
  end;
  SetLength(Buffer,NewSize);
  if not CryptEncrypt(Key,0,true,0,@(Buffer[0]),@BufferSize,NewSize) then
  begin
    ShowMessage('Encrypting failed!');
    Exit;
  end;
  AssignFile(f,ePath.Text+'key');
  Rewrite(f);
  BlockWrite(f,Buffer[0],NewSize);
  CloseFile(f);
  Text:=eMessage.Text;
  SetLength(Buffer,Length(ByteArray(Text)));
  SetLength(Signature,Length(ByteArray(Text)));
  Move(ByteArray(Text)[0],Buffer[0],Length(ByteArray(Text)));
  Move(ByteArray(Text)[0],Signature[0],Length(ByteArray(Text)));
  if not CryptCreateHash(ProviderDescriptor,CALG_MD5,0,0,@HashDescriptor)
then
  begin
    ShowMessage('Failed to create hash!');
    Exit;
  end;
  if not CryptHashData(HashDescriptor,@(Buffer[0]),Length(Buffer),0) then
  begin
    CryptDestroyHash(HashDescriptor);
    ShowMessage('Failed to make hash!');
    Exit;
  end;
  BufferSize:=Length(Buffer);
  NewSize:=BufferSize;
  if not CryptEncrypt(Key,0,true,0,nil,@NewSize,0) then
  begin
    ShowMessage('kernel panic!');
    Exit;
  end;
  SetLength(Buffer,NewSize);
  if not CryptEncrypt(Key,0,true,0,@(Buffer[0]),@BufferSize,NewSize) then
  begin
    ShowMessage('Encrypting failed!');
```

```
    Exit;
  end;
  BufferSize:=Length(Signature);
  NewSize:=BufferSize;
  if not CryptSignHash(HashDescriptor, AT_SIGNATURE, nil, 0, nil, @NewSize)
then
  begin
    ShowMessage('kernel panic!');
    Exit;
  end;
  SetLength(Signature,NewSize);
  if not CryptSignHash(HashDescriptor, AT_SIGNATURE, nil, 0, @(Signature[0]),
@NewSize) then
  begin
    ShowMessage('kernel panic!');
    Exit;
  end;
  BufferSize:=Length(Buffer);
  NewSize:=Length(Signature);
  AssignFile(f,ePath.Text+'message');
  Rewrite(f);
  BlockWrite(f,BufferSize,4);
  BlockWrite(f,Buffer[0],BufferSize);
  BlockWrite(f,NewSize,4);
  BlockWrite(f,Signature[0],NewSize);
  CloseFile(f);
end;

procedure TMainForm.FormCreate(Sender: TObject);
var
  ProviderName: Array [0..59] of WideChar;
begin
  ProviderName:='Microsoft Base Cryptographic Provider v1.0';
  if not
CryptAcquireContext(@ProviderDescriptor,nil,@(ProviderName[0]),PROV_RSA_FULL,
0) then
  begin
    ShowMessage('Provider not found!');
    Exit;
  end;
end;

procedure TMainForm.FormDestroy(Sender: TObject);
begin
  CryptReleaseContext(ProviderDescriptor,0);
end;

procedure TMainForm.btnSelectPathClick(Sender: TObject);
var
  Result: String;
begin
  Result:=ePath.Text;
  if SelectDirectory('Select shared directory', 'Desktop', Result,
      [sdNewUI, sdNewFolder]) then
  begin
    ePath.Text:=Result;
```

```
  end;
end;

end.
```

CLIENT

```
unit Client;

interface

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

type
  ByteArray = array of byte;
  TMainForm = class(TForm)
    GroupBox1: TGroupBox;
    btnSelectPath: TSpeedButton;
    ePath: TEdit;
    GroupBox2: TGroupBox;
    ePassword: TEdit;
    GroupBox3: TGroupBox;
    eMessage: TMemo;
    GridPanel1: TGridPanel;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure btnSelectPathClick(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
    Key, ProviderDescriptor: Cardinal;
  public
    { Public declarations }
  end;

var
  MainForm: TMainForm;

implementation

uses Wcrypt2;

{$R *.dfm}

procedure TMainForm.btnSelectPathClick(Sender: TObject);
var
  Result: String;
begin
  Result:=ePath.Text;
  if SelectDirectory('Select shared directory', 'Desktop', Result,
      [sdNewUI, sdNewFolder]) then
  begin
    ePath.Text:=Result;
```

```
  end;
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
  PublicKeyDescriptor: Cardinal;
  HashDescriptor, KeyDescriptor: Cardinal;
  f: file of byte;
  Buffer: ByteArray;
  BufferSize: Integer;
  Text: AnsiString;
begin
  AssignFile(f,ePath.Text+'key');
  Reset(f);
  BufferSize:=FileSize(f);
  SetLength(Buffer,BufferSize);
  BlockRead(f,Buffer[0],BufferSize);
  CloseFile(f);
  if not
CryptImportKey(ProviderDescriptor,@(Buffer[0]),BufferSize,0,0,@PublicKeyDescr
iptor) then
  begin
    ShowMessage('Public key creation failed!');
    Exit;
  end;
  if not CryptCreateHash(ProviderDescriptor,CALG_MD5,0,0,@HashDescriptor)
then
  begin
    CryptDestroyKey(PublicKeyDescriptor);
    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
    CryptDestroyKey(PublicKeyDescriptor);
    CryptDestroyHash(HashDescriptor);
    ShowMessage('Failed to make hash!');
    Exit;
  end;
  if not
CryptDeriveKey(ProviderDescriptor,CALG_RC2,HashDescriptor,CRYPT_EXPORTABLE,@K
eyDescriptor) then
  begin
    CryptDestroyKey(PublicKeyDescriptor);
    CryptDestroyHash(HashDescriptor);
    ShowMessage('Failed to create key!');
    Exit;
  end;
  CryptDestroyHash(HashDescriptor);
  if not CryptExportKey(KeyDescriptor, PublicKeyDescriptor, SIMPLEBLOB, 0,
nil, @BufferSize) then
  begin
    CryptDestroyKey(PublicKeyDescriptor);
```

```
    ShowMessage('kernel panic!');
    Exit;
  end;
  SetLength(Buffer,BufferSize);
  if not CryptExportKey(KeyDescriptor, PublicKeyDescriptor, SIMPLEBLOB, 0,
@(Buffer[0]), @BufferSize) then
  begin
    CryptDestroyKey(PublicKeyDescriptor);
    ShowMessage('Public key creation failed!');
    Exit;
  end;
  Key:=KeyDescriptor;
  AssignFile(f,ePath.Text+'key');
  Rewrite(f);
  BlockWrite(f,Buffer[0],BufferSize);
  CloseFile(f);
  CryptDestroyKey(PublicKeyDescriptor);
end;

procedure TMainForm.Button2Click(Sender: TObject);
var
  f: file of byte;
  Buffer, Signature: ByteArray;
  BufferSize, SignatureSize: Integer;
  Result: AnsiString;
  KeyDescriptor, HashDescriptor: Cardinal;
begin
  AssignFile(f,ePath.Text+'key');
  Reset(f);
  BufferSize:=FileSize(f);
  SetLength(Buffer,BufferSize);
  BlockRead(f,Buffer[0],BufferSize);
  CloseFile(f);
  if not CryptDecrypt(Key,0,true,0,@(Buffer[0]),@BufferSize) then
  begin
    ShowMessage(IntToStr(GetLastError));
    ShowMessage('Decrypting failed!');
    Exit;
  end;
  SetLength(Buffer, BufferSize);
  if not
CryptImportKey(ProviderDescriptor,@(Buffer[0]),BufferSize,0,0,@KeyDescriptor)
then
  begin
    ShowMessage('Public key creation failed!');
    Exit;
  end;
  AssignFile(f,ePath.Text+'message');
  Reset(f);
  BlockRead(f,BufferSize,4);
  SetLength(Buffer,BufferSize);
  BlockRead(f,Buffer[0],BufferSize);
  BlockRead(f,SignatureSize,4);
  SetLength(Signature,SignatureSize);
  BlockRead(f,Signature[0],SignatureSize);
  CloseFile(f);
```

```
  if not CryptDecrypt(Key,0,true,0,@(Buffer[0]),@BufferSize) then
  begin
    ShowMessage(IntToStr(GetLastError));
    ShowMessage('Dencrypting failed!');
    Exit;
  end;
  SetLength(Result,BufferSize);
  Move(Buffer[0],Result[1],BufferSize);
  eMessage.Text:=Result;
  SetLength(Buffer,BufferSize);
  if not CryptCreateHash(ProviderDescriptor,CALG_MD5,0,0,@HashDescriptor)
then
  begin
    ShowMessage('Failed to create hash!');
    Exit;
  end;
  if not CryptHashData(HashDescriptor,@(Buffer[0]),BufferSize,0) then
  begin
    CryptDestroyHash(HashDescriptor);
    ShowMessage('Failed to make hash!');
    Exit;
  end;
  if not
CryptVerifySignature(HashDescriptor,@(Signature[0]),SignatureSize,KeyDescript
or,nil,0) then
  begin
    MessageDlg('Signature failed!'#13'Error: '+IntToStr(GetLastError),
mtError, [mbOK], 0);
  end
  else
  begin
    MessageDlg('Signature success!', mtInformation, [mbOK], 0);
  end;
end;

procedure TMainForm.FormCreate(Sender: TObject);
var
  ProviderName: Array [0..59] of WideChar;
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;
end;

procedure TMainForm.FormDestroy(Sender: TObject);
begin
  CryptReleaseContext(ProviderDescriptor,0);
end;

end.
```

Краткие выводы