LAMPIRAN

```
unit uMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls, ExtCtrls, ExtDlgs, DCPcrypt2, DCPrc4, DCPsha1,
  sSkinManager, Menus;

type
  TfrmMain = class(TForm)
    OpenPictureDialog1: TOpenPictureDialog;
    SavePictureDialog1: TSavePictureDialog;
    sSkinManager1: TsSkinManager;
    MainMenu1: TMainMenu;
    Action1: TMenuItem;
    Home1: TMenuItem;
    Encrypt1: TMenuItem;
    Decrypt1: TMenuItem;
    Exit1: TMenuItem;
    Label1: TLabel;
    Label2: TLabel;
    Page: TPageControl;
    TabSheet1: TTabSheet;
    Panel5: TPanel;
    Panel2: TPanel;
    imgEncode: TImage;
    OpEncode: TButton;
    NextEncode: TButton;
    ResetEncode: TButton;
    TabSheet2: TTabSheet;
    Panel6: TPanel;
    Panel4: TPanel;
    imgDecode: TImage;
    OpDecode: TButton;
    NextDecode: TButton;
    ResetDecode: TButton;
    TabSheet3: TTabSheet;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    btnEncode: TButton;
    BackEncode: TButton;
    moEncode: TMemo;
    TabSheet4: TTabSheet;
    Label3: TLabel;
    btnDecode: TButton;
    BackDecode: TButton;
    edDecodePass1: TEdit;
    TabSheet5: TTabSheet;
    Label4: TLabel;
```

```
    moDecode: TMemo;
    TabSheet6: TTabSheet;
    edEncodePass1: TEdit;
    edEncodePass2: TEdit;
    ResetEncode2: TButton;
    ResetDecode2: TButton;
    Label8: TLabel;
    Label9: TLabel;
    procedure btnEncodeClick(Sender: TObject);
    procedure btnDecodeClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure OpEncodeClick(Sender: TObject);
    procedure OpDecodeClick(Sender: TObject);
    procedure NextEncodeClick(Sender: TObject);
    procedure NextDecodeClick(Sender: TObject);
    procedure BackEncodeClick(Sender: TObject);
    procedure BackDecodeClick(Sender: TObject);
    procedure moEncodeChange(Sender: TObject);
    procedure edEncodePass1Change(Sender: TObject);
    procedure edEncodePass2Change(Sender: TObject);
    procedure Home1Click(Sender: TObject);
    procedure Encrypt1Click(Sender: TObject);
    procedure Decrypt1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure ResetEncodeClick(Sender: TObject);
    procedure ResetDecodeClick(Sender: TObject);
    procedure ResetEncode2Click(Sender: TObject);
    procedure ResetDecode2Click(Sender: TObject);
  private
    { Private declarations }
    function CarrierCheck(img: TBitmap): Boolean;
    function doDecrypt(Pass, Content: String): String;
    function doEncrypt(Pass, Content: String): String;
    procedure Encode;
    procedure Decode;
  public
    { Public declarations }
  end;

var
  frmMain: TfrmMain;
  Bmp: TBitmap;

implementation

{$R *.dfm}

function getByte(Number: Byte): String;
var i : integer;
    Bit : String[1];
```

```
    Return : string[8];
begin
  i := 0;
  Return := '00000000';
  if Number <> 0 then begin
    while Number <> 1 do begin
      Str(Number mod 2,Bit);
      Return[8-i] := Bit[1];
      Number := Trunc(Number/2);
      Inc(i);
    end;

    Str(Number,Bit);
    Return[8-i] := Bit[1];
  end;

  Result := Return;
end;

function getWord(Number: Word): String;
var i : integer;
    Bit : String[1];
    Return : string[16];
begin
  i := 0;
  Return := '0000000000000000';
  if Number <> 0 then begin
    while Number <> 1 do begin
      Str(Number mod 2,Bit);
      Return[16-i] := Bit[1];
      Number := Trunc(Number/2);
      Inc(i);
    end;

    Str(Number,Bit);
    Return[16-i] := Bit[1];
  end;

  Result := Return;
end;

function getDecimal(Bit: String): Integer;
var i : Integer;
    Return: Integer;
begin
  Return := 0;
  for i := 1 to Length(Bit) do begin
    if Bit[i] = '1' then Return := Return + Trunc(Exp((Length(Bit)-i)*ln(2)));
  end;

  Result := Return;
```

```
end;


procedure TfrmMain.Encode;
var i, j, k, x, LSBCount: Integer;
    Red, Green, Blue: Integer;
    strMark, bitMark, bitContent, tempContent: String;
    bitRed, bitGreen, bitBlue: String;
    bitBaris, bitKolom: String;
begin
    Bmp.Assign(imgEncode.Picture.Bitmap);
    Bmp.PixelFormat := pf24bit;
    if CarrierCheck(Bmp) then begin
      MessageDlg('File carrier sudah disisipi data.',mtWarning,[mbOK],0);
      Exit;
    end;

    LSBCount := (Bmp.Width * Bmp.Height) * 3;
    LSBCount := LSBCount - (Bmp.Width * 3);

    tempContent := edEncodePass1.Text + '#' + moEncode.Lines.Text;
    tempContent := doEncrypt(edEncodePass1.Text, tempContent);

    for i := 1 to Length(tempContent) do begin
      bitContent := bitContent + getByte(Ord(tempContent[i]));
    end;

    if Length(bitContent) > LSBCount then begin
      MessageDlg('Content yang akan diisikan terlalu besar.',mtWarning,[mbOK],0);
      Exit;
    end;

    Self.Cursor := crHourGlass;
    strMark := '&&&';
    bitMark := getByte(Ord(strMark[1])) +
               getByte(Ord(strMark[2])) +
               getByte(Ord(strMark[3]));

    j := 1;
    for i := 0 to 7 do begin
      Red := GetRValue(GetPixel(Bmp.Canvas.Handle,i,0));
      Green := GetGValue(GetPixel(Bmp.Canvas.Handle,i,0));
      Blue := GetBValue(GetPixel(Bmp.Canvas.Handle,i,0));

      bitRed := getByte(Red);
      bitGreen := getByte(Green);
      bitBlue := getByte(Blue);

      Delete(bitRed,8,1);
      Delete(bitGreen,8,1);
      Delete(bitBlue,8,1);
```

```
  bitBlue := bitBlue + bitMark[j];
  bitGreen := bitGreen + bitMark[j+1];
  bitRed := bitRed + bitMark[j+2];

  j := j + 3;

  Red := getDecimal(bitRed);
  Green := getDecimal(bitGreen);
  Blue := getDecimal(bitBlue);

  Bmp.Canvas.Pixels[i,0] := RGB(Red,Green,Blue);
end;

k := 1;

for i := 1 to Bmp.Height - 1 do begin
  for j := 0 to Bmp.Width - 1 do begin
    Red := GetRValue(GetPixel(Bmp.Canvas.Handle,j,i));
    Green := GetGValue(GetPixel(Bmp.Canvas.Handle,j,i));
    Blue := GetBValue(GetPixel(Bmp.Canvas.Handle,j,i));

    bitRed := getByte(Red);
    bitGreen := getByte(Green);
    bitBlue := getByte(Blue);

    Delete(bitRed,8,1);
    Delete(bitGreen,8,1);
    Delete(bitBlue,8,1);

    bitBlue := bitBlue + bitContent[k];
    bitGreen := bitGreen + bitContent[k+1];
    bitRed := bitRed + bitContent[k+2];

    k := k + 3;

    Red := getDecimal(bitRed);
    Green := getDecimal(bitGreen);
    Blue := getDecimal(bitBlue);

    Bmp.Canvas.Pixels[j,i] := RGB(Red,Green,Blue);
    if k > length(bitContent) then Break;
  end;

  if k > length(bitContent) then Break;
end;


k := 1;
bitBaris := getWord(i);
```

```
for x := 8 to 15 do begin


  Red := GetRValue(GetPixel(Bmp.Canvas.Handle,x,0));
  Green := GetGValue(GetPixel(Bmp.Canvas.Handle,x,0));
  Blue := GetBValue(GetPixel(Bmp.Canvas.Handle,x,0));

  bitRed := getByte(Red);
  bitGreen := getByte(Green);

  Delete(bitRed,8,1);
  Delete(bitGreen,8,1);

  bitGreen := bitGreen + bitBaris[k];
  bitRed := bitRed + bitBaris[k+1];

  k := k + 2;

  Red := getDecimal(bitRed);
  Green := getDecimal(bitGreen);

  Bmp.Canvas.Pixels[x,0] := RGB(Red,Green,Blue);
end;

k := 1;
bitKolom := getWord(j);


for x := 16 to 23 do begin


  Red := GetRValue(GetPixel(Bmp.Canvas.Handle,i,0));
  Green := GetGValue(GetPixel(Bmp.Canvas.Handle,i,0));
  Blue := GetBValue(GetPixel(Bmp.Canvas.Handle,i,0));

  bitRed := getByte(Red);
  bitGreen := getByte(Green);

  Delete(bitRed,8,1);
  Delete(bitGreen,8,1);

  bitGreen := bitGreen + bitKolom[k];
  bitRed := bitRed + bitKolom[k+1];

  k := k + 2;

  Red := getDecimal(bitRed);
  Green := getDecimal(bitGreen);

  Bmp.Canvas.Pixels[x,0] := RGB(Red,Green,Blue);
```

```
    end;

    Self.Cursor := crDefault;

      if SavePictureDialog1.Execute then begin
        SavePictureDialog1.DefaultExt := '*.bmp';
        Bmp.SaveToFile(SavePictureDialog1.FileName);
    end;
end;

procedure TfrmMain.Decode;
var i, j, k: Integer;
    Red, Green, Blue: Integer;
    bitContent, tempContent: String;
    bitRed, bitGreen, bitBlue: String;
    bitBaris, bitKolom: String;
    Baris, Kolom: Integer;
begin
    Bmp.Assign(imgDecode.Picture.Bitmap);
    Bmp.PixelFormat := pf24bit;
    if not CarrierCheck(Bmp) then begin
      MessageDlg('File gambar tidak mengandung pesan.',mtWarning,[mbOK],0);
      Exit;
    end;


    bitBaris := '';
    for k := 8 to 15 do begin
      Red := GetRValue(GetPixel(Bmp.Canvas.Handle,k,0));
      Green := GetGValue(GetPixel(Bmp.Canvas.Handle,k,0));
      bitBaris := bitBaris + getByte(Green)[8] + getByte(Red)[8];
    end;
    Baris := getDecimal(bitBaris);

    bitKolom := '';
    for k := 16 to 23 do begin
      Red := GetRValue(GetPixel(Bmp.Canvas.Handle,k,0));
      Green := GetGValue(GetPixel(Bmp.Canvas.Handle,k,0));
      bitKolom := bitKolom + getByte(Green)[8] + getByte(Red)[8];
    end;
    Kolom := getDecimal(bitKolom);

    tempContent := '';
    for i := 1 to Bmp.Height - 1 do begin
      for j := 0 to Bmp.Width - 1 do begin
        Red := GetRValue(GetPixel(Bmp.Canvas.Handle,j,i));
        Green := GetGValue(GetPixel(Bmp.Canvas.Handle,j,i));
        Blue := GetBValue(GetPixel(Bmp.Canvas.Handle,j,i));

        bitRed := getByte(Red);
        bitGreen := getByte(Green);
```

```
      bitBlue := getByte(Blue);

      tempContent := tempContent + Copy(bitBlue,8,1) + Copy(bitGreen,8,1) + Copy(bitRed,8,1);

      if (Baris = i) and (Kolom = j) then Break;
    end;

    if (Baris = i) and (Kolom = j) then Break;
  end;

  bitContent := '';
  k := 1;
  for i := 1 to Round(Length(tempContent)/8) do begin
    bitContent := bitContent + Chr(getDecimal(Copy(tempContent,k,8)));
    k := k + 8;
  end;

  bitContent := doDecrypt(edDecodePass1.Text,bitContent);
  if edDecodePass1.Text = Copy(bitContent,1,Length(edDecodePass1.Text)) then

    moDecode.Lines.Text := StringReplace(bitContent,edDecodePass1.Text+'#','',[rfReplaceAll])
  else MessageDlg('Password salah.',mtWarning,[mbOK],0);
end;

function TfrmMain.CarrierCheck(img: TBitmap): Boolean;
var i: Integer;
  Red, Green, Blue: Integer;
  strMark, bitLSB: String;
begin
  bitLSB := '';
  for i := 0 to 7 do begin
    Red := GetRValue(GetPixel(img.Canvas.Handle,i,0));
    Green := GetGValue(GetPixel(img.Canvas.Handle,i,0));
    Blue := GetBValue(GetPixel(img.Canvas.Handle,i,0));

    bitLSB := bitLSB + getByte(Blue)[8] + getByte(Green)[8] + getByte(Red)[8];
  end;

  strMark := Chr(getDecimal(Copy(bitLSB,1,8))) +
          Chr(getDecimal(Copy(bitLSB,9,8))) +
          Chr(getDecimal(Copy(bitLSB,17,8)));

  Result := strMark = '&&&';
end;


procedure TfrmMain.btnEncodeClick(Sender: TObject);
begin
  if edEncodePass1.Text <> edEncodePass2.Text then begin
    MessageDlg('Konfirmasi password salah',mtWarning,[mbOK],0);
    Exit;
```

```
    end;

    if imgEncode.Picture.Bitmap = nil then begin
      MessageDlg('File carrier belum dipilih',mtWarning,[mbOK],0);
      Exit;
    end;

    if Trim(moEncode.Lines.Text) = '' then begin
      MessageDlg('Content belum diisi',mtWarning,[mbOK],0);
      Exit;
    end;

    Encode;
end;

procedure TfrmMain.btnDecodeClick(Sender: TObject);
begin
    Decode;
    page.TabIndex:=5;
end;

procedure TfrmMain.FormCreate(Sender: TObject);
begin
    Bmp := TBitmap.Create;
end;

procedure TfrmMain.OpEncodeClick(Sender: TObject);
var bmp : TBitmap;
begin
    OpenPictureDialog1.Filter := 'Bitmaps (*.bmp)|*.bmp';
    OpenPictureDialog1.DefaultExt := '*.bmp';

    if OpenPictureDialog1.Execute then begin
      bmp := TBitmap.Create;
      bmp.LoadFromFile(OpenPictureDialog1.FileName);
      imgEncode.Picture.LoadFromFile(OpenPictureDialog1.FileName);
      FreeAndNil(bmp);
      NextEncode.Enabled :=true;
      ResetEncode.Enabled :=true;
    end;
end;

procedure TfrmMain.OpDecodeClick(Sender: TObject);
var bmp : TBitmap;
begin

    OpenPictureDialog1.Filter := 'Bitmaps (*.bmp)|*.bmp';
    OpenPictureDialog1.DefaultExt := '*.bmp';

    if OpenPictureDialog1.Execute then begin
      bmp := TBitmap.Create;
```

```
      bmp.LoadFromFile(OpenPictureDialog1.FileName);
      imgDecode.Picture.LoadFromFile(OpenPictureDialog1.FileName);
      FreeAndNil(bmp);
      NextDecode.Enabled:=true;
      ResetDecode.Enabled:=true;
    end;
end;

procedure TfrmMain.NextEncodeClick(Sender: TObject);
begin
  page.TabIndex := 2;
end;

procedure TfrmMain.NextDecodeClick(Sender: TObject);
begin
  page.TabIndex := 4;
end;

procedure TfrmMain.BackEncodeClick(Sender: TObject);
begin
  page.TabIndex := 1;
end;

procedure TfrmMain.BackDecodeClick(Sender: TObject);
begin
  page.TabIndex := 3;
end;

procedure TfrmMain.moEncodeChange(Sender: TObject);
begin
    if (edEncodePass1.Text ='') and (moEncode.Text ='') then
    begin
      btnEncode.Enabled := false;
    end
    else
    begin
      btnEncode.Enabled := true;
    end
end;

procedure TfrmMain.edEncodePass1Change(Sender: TObject);
begin
    if (edEncodePass1.Text ='') and (moEncode.Text ='') then
    begin
      btnEncode.Enabled := false;
    end
    else
    begin
      btnEncode.Enabled := true;
    end
end;
```

```
procedure TfrmMain.edEncodePass2Change(Sender: TObject);
begin
    if (edEncodePass1.Text ='') and (moEncode.Text ='') then
    begin
      btnEncode.Enabled := false;
    end
    else
    begin
      btnEncode.Enabled := true;
    end
end;

procedure TfrmMain.Home1Click(Sender: TObject);
begin
  page.tabindex := 0;
end;

procedure TfrmMain.Encrypt1Click(Sender: TObject);
begin
  page.tabindex := 1;
end;

procedure TfrmMain.Decrypt1Click(Sender: TObject);
begin
  page.tabindex := 3;
end;

procedure TfrmMain.Exit1Click(Sender: TObject);
begin
  Application.Terminate;
end;

procedure TfrmMain.FormShow(Sender: TObject);
begin
    page.TabIndex:=0;
end;

procedure TfrmMain.ResetEncodeClick(Sender: TObject);
begin
   imgEncode.Picture.Assign(nil);
   moEncode.Clear;
   edEncodePass1.Clear; edEncodePass2.Clear;
   ResetEncode.enabled :=false;
   NextEncode.enabled :=false;
end;

procedure TfrmMain.ResetDecodeClick(Sender: TObject);
begin
   imgDecode.Picture.Assign(nil);
   moDecode.Clear;
```

```
  edDecodePass1.Clear;
  NextDecode.enabled :=false;
  ResetDecode.enabled :=false;
end;

procedure TfrmMain.ResetEncode2Click(Sender: TObject);
begin
   moEncode.Clear;
   edEncodePass1.Clear; edEncodePass2.Clear;
end;

procedure TfrmMain.ResetDecode2Click(Sender: TObject);
begin
   moDecode.Clear;
   edDecodePass1.Clear;
end;

end.
```