Lampiran A
Listing Program

Listing UMain.pas

```
unit UMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Menus, ComCtrls, Buttons, ShellAPI;

type
  TMain = class(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    Label1: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    btBrowse: TButton;
    btGen: TButton;
    GroupBox1: TGroupBox;
    Label8: TLabel;
    Label9: TLabel;
    PrivD: TEdit;
    PrivN: TEdit;
    GroupBox2: TGroupBox;
    Label10: TLabel;
    Label11: TLabel;
    PubE: TEdit;
    PubN: TEdit;
    edStart: TEdit;
    edDuration: TEdit;
    edEnd: TEdit;
    edPath: TEdit;
    btEncrypt: TButton;
    RPlain: TRichEdit;
    edPass: TEdit;
    TabSheet2: TTabSheet;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    edPath1: TEdit;
    btBrowse1: TButton;
    edPass1: TEdit;
    edStart1: TEdit;
    edEnd1: TEdit;
    edDuration1: TEdit;
    GroupBox3: TGroupBox;
    Label18: TLabel;
    Label19: TLabel;
    PubE1: TEdit;
    PubN1: TEdit;
    btDecrypt: TButton;
    RCipher: TMemo;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    About1: TMenuItem;
    OpenDlgDec: TOpenDialog;
    OpenDlgEnc: TOpenDialog;
    SaveDlgEnc: TSaveDialog;
    SaveDlgDec: TSaveDialog;
    SaveDlgV: TSaveDialog;
    OpenDlgV: TOpenDialog;
    TabSheet3: TTabSheet;
    CreateP: TMemo;
    btSimpan: TButton;
    SaveDlgP: TSaveDialog;
    BtnViewCipherFiles: TBitBtn;
    BitBtn1: TBitBtn;
    procedure btSimpanClick(Sender: TObject);
    procedure btBrowseClick(Sender: TObject);
    procedure btBrowse1Click(Sender: TObject);
    procedure btGenClick(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure btEncryptClick(Sender: TObject);
    procedure btDecryptClick(Sender: TObject);
    procedure BtnViewCiphrFilesClick(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure TabSheet1Show(Sender: TObject);
    procedure TabSheet2Show(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure TabSheet3Show(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Main: TMain;
```

```
implementation

uses Camellia, RSATools, Tools, UAbout;

{$R *.dfm}

procedure TMain.btSimpanClick(Sender: TObject);
begin
if SaveDlgP.Execute then
   begin
     CreateP.Lines.SaveToFile(SaveDlgP.FileName);
   end;
end;

procedure TMain.btBrowseClick(Sender: TObject);
begin
if OpenDlgEnc.Execute then
   begin
     edPath.Text:=OpenDlgEnc.FileName;
     RPlain.Clear;
     RPlain.Lines.LoadFromFile(OpenDlgEnc.FileName);
   end;
end;

procedure TMain.btBrowse1Click(Sender: TObject);
begin
if OpenDlgDec.Execute then
   begin
     edPath1.Text:=OpenDlgDec.FileName;
     RCipher.Clear;
     RCipher.Lines.LoadFromFile(OpenDlgDec.FileName);
   end;
end;

procedure TMain.btGenClick(Sender: TObject);
var p : Longint;  //random prime
    q : Longint;  //second random prime that not equal to p
    n : Longint;  //p * q
    pi : Longint; //(p - 1)(q - 1)
    e : Longint;  //e that relatively prime to pi but less than pi
    d : Longint;  //d that d*e congruent to 1 mod pi
    i1 : Longint; //counter

    c : Longint;
    temp2 : array of Longint; // temp dynamic array handler that hand selection
of e
    temp3 : Longint;
    temp4 : Longint;
    temp5 : Longint; // temp handler
    temp6 : Longint; // temp handler 2
label lagi,ulang;

begin
lagi:
    p := RdmPrime;
    q := RdmPrime;

    //Trap handler if p = q
    If p = q Then
       GoTo lagi;

    n := p * q;

PrivN.Text := IntToStr(n);
PubN.Text := IntToStr(n);
    pi := (p - 1) * (q - 1);

//search for e
c := pi - 1;
SetLength(temp2,c);

For e := 2 To (pi - 1) do
   begin
    temp6 := gcd(pi, e);
    If temp6 = 1 Then
       begin
        temp2[c] := e;
        c := c - 1;
       end;
   end;

//random selection of e
ulang:
    Randomize;
    temp3 := Round((pi - 1) * (Random(10000)/10000));
    temp4 := temp2[temp3];
    If (temp4 = 0) Or (temp4 = Null) Then
       GoTo ulang;


//select e that is prime
For i1 := 2 To Round(Sqrt(temp4)) do
  begin
    temp5 := temp4 Mod i1;
    If temp5 = 0 Then
```

```
      GoTo ulang;
  end;

PubE.Text := IntToStr(temp4);

//determine d such that d*e congruent 1 mod pi and d > 0, d > e
d := Euclid(pi, temp4);
If d < temp4 Then
   GoTo ulang;

PrivD.Text := IntToStr(d);
end;

procedure TMain.About1Click(Sender: TObject);
begin
frmAbout.ShowModal;
end;

procedure TMain.btEncryptClick(Sender: TObject);
var slama:String;
    Source,Dest:File;
    valid:TextFile;
    Buffer: array[0..15] of byte;
    temp:Byte128;
    Read:Integer;
    awal,akhir,lama:TDateTime;
    i,d,n:Integer;
    s,x:longint;
begin
if edPass.Text = '' then
   MessageDlg('Kata Kunci Belum Diisi',mtError,mbOKCancel,0)
  else
    if RPlain.Text = '' then
       MessageDlg('Document masih kosong',mtError,mbOKCancel,0)
      else
       if PrivD.Text = '' then
          MessageDlg('Klik dahulu Generate Key',mtError,mbOKCancel,0)
         else
           begin
           ShowMessage('Masukan Nama File Ciphertext');
           if SaveDlgEnc.Execute then
              begin
                awal:=Time;
                edStart.Text:=TimeToStr(awal);
                AssignFile(Source,edPath.Text);
                AssignFile(Dest,SaveDlgEnc.FileName);
                Reset(source,1);
                ReWrite(Dest,1);
          repeat
            FillChar(Buffer,sizeof(Buffer),0);
            FillChar(temp,sizeof(temp),0);
            BlockRead(Source,Buffer,sizeof(Buffer),read);
            if read <> 0 then
              begin
               move(Buffer,temp,read);
               Encrypt(temp,edPass.Text,temp);
               move(temp,Buffer,sizeof(temp));
               BlockWrite(Dest,Buffer,SizeOf(Buffer));
              end;
          until read <> sizeof(Buffer);
          CloseFile(Source);
          CloseFile(Dest);
          akhir:=Time;
          edEnd.Text:=TimeToStr(akhir);
          lama:=(akhir-awal)*86000;
          str(lama:12:8,slama);
          edDuration.Text:=slama;
        end;
      if Application.MessageBox('Masukan Nama File Validasi dan Digital
Signature','Pesan',MB_OK) = IDOK then
        begin
          if SaveDlgV.Execute then
             begin
               AssignFile(Source,SaveDlgEnc.FileName);
               Reset(Source,1);
               repeat
                 BlockRead(Source,Buffer,sizeof(Buffer),read);
                 for i:=0 to 15 do
                   s := s xor buffer[i];
               until read <> sizeof(buffer);
               CloseFile(Source);
               AssignFile(valid,SaveDlgV.FileName);
               Rewrite(valid);
               d:=StrToInt(PrivD.Text);
               n:=StrToInt(PrivN.Text);
               x:=pangkatmod(s,d,n);
               write(valid,inttostr(x));
               CloseFile(valid);
             end;
        end;
      end;
end;

procedure TMain.btDecryptClick(Sender: TObject);
var slama,temp1:String;
    Source,Dest:File;
    valid:TextFile;
    Buffer: array[0..15] of byte;
    temp:Byte128;
    Read:Integer;
    awal,akhir,lama:TDateTime;
    i,e,n:Integer;
    s,p:Longint;
begin
if edPass1.Text = '' then
   MessageDlg('Kata Kunci Belum Diisi',mtError,mbOKCancel,0)
  else
    if RCipher.Text = '' then
       MessageDlg('Document masih kosong',mtError,mbOKCancel,0)
      else
        if (PubE1.Text = '') or (PubN1.Text = '') then
            MessageDlg('Public Key Belum Di Isi',mtError,mbOKCancel,0)
           else
            begin
              MessageDlg('Buka File Validasi',mtInformation,mbOKCancel,0);
              if OpenDlgV.Execute then
                 begin
                   e:=StrToInt(PubE1.Text);
                   n:=StrToInt(PubN1.Text);
                   AssignFile(valid,OpenDlgV.FileName);
                   Reset(valid);
                   Readln(valid,temp1);
                   CloseFile(valid);
                   AssignFile(Source,edPath1.Text);
                   Reset(Source,1);
                   repeat
                     BlockRead(Source,Buffer,sizeof(Buffer),read);
                     for i:=0 to 15 do
                         s := s xor buffer[i];
                   until read <> sizeof(buffer);
                   CloseFile(Source);
                   p:=pangkatmod(StrToInt(temp1),e,n);
                   if p <> s then
                      begin
                        MessageDlg('File      Chipertext      tidak      valid      proses
berhenti',mtError,mbOKCancel,0);
                        exit;
                      end
                     else
                      if SaveDlgDec.Execute then
                         begin
                           awal:=Time;
                           edStart1.Text:=TimeToStr(awal);
                           AssignFile(Source,edPath1.Text);
```

```
                    AssignFile(Dest,SaveDlgDec.FileName);
                    Reset(Source,1);
                    Rewrite(Dest,1);
                    repeat
                      FillChar(Buffer,sizeof(Buffer),0);
                      FillChar(temp,sizeof(temp),0);
                      BlockRead(Source,Buffer,sizeof(Buffer),read);
                      if read <> 0 then
                        begin
                         move(Buffer,temp,read);
                         Decrypt(temp,edPass.Text,temp);
                         move(temp,Buffer,sizeof(temp));
                         BlockWrite(Dest,Buffer,read);
                        end;
                    until read <> sizeof(Buffer);
                    CloseFile(Source);
                    CloseFile(Dest);
                    akhir:=Time;
                    edEnd1.Text:=TimeToStr(akhir);
                    lama:=(akhir-awal)*86000;
                    str(lama:12:8,slama);
                    edDuration1.Text:=slama;
                  end;
          end;
     end;
end;

procedure TMain.BtnViewCiphrFilesClick(Sender: TObject);
begin
  if FileExists(SaveDlgEnc.FileName) then
    ShellExecute(Handle,            nil,            PChar('NotePad.exe'),
PChar(SaveDlgEnc.FileName), nil, sw_ShowNormal);
end;

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

procedure TMain.TabSheet1Show(Sender: TObject);
begin
OpenDlgEnc.FileName:='';
SaveDlgEnc.FileName:='';
SaveDlgV.FileName:='';
edPath.Clear;
edStart.Clear;
edDuration.Clear;
edEnd.Clear;
```

```
edPass.Clear;
PrivD.Clear;
PrivN.Clear;
PubE.Clear;
PubN.Clear;
RPlain.Clear;
end;

procedure TMain.TabSheet2Show(Sender: TObject);
begin
OpenDlgDec.FileName:='';
SaveDlgDec.FileName:='';
OpenDlgV.FileName:='';
edPath1.Clear;
edPass1.Clear;
edStart1.Clear;
edDuration1.Clear;
edEnd1.Clear;
PubE1.Clear;
PubN1.Clear;
RCipher.Clear;
end;

procedure TMain.BitBtn1Click(Sender: TObject);
begin
  if FileExists(SaveDlgDec.FileName) then
    ShellExecute(Handle,          nil,          PChar('NotePad.exe'),
PChar(SaveDlgDec.FileName), nil, sw_ShowNormal);
end;

procedure TMain.TabSheet3Show(Sender: TObject);
begin
SaveDlgP.FileName:='';
CreateP.Clear;
end;

end.
```

Listing Tools.pas

```
unit Tools;

interface

uses
  Sysutils;

type
{$IFDEF VER120}
  dword= longword;
{$ELSE}
  dword= longint;
{$ENDIF}

  Word64 = array[0..1] of DWord;
  Word128 = array[0..3] of Dword;
  Byte64 = array[0..7] of Byte;
  Byte128 = array[0..15] of Byte;
  Byte256 = array[0..31] of Byte;
function LRot16(X: word; c: integer): word; assembler;
function RRot16(X: word; c: integer): word; assembler;
function LRot32(X: dword; c: integer): dword; assembler;
function RRot32(X: dword; c: integer): dword; assembler;
function SwapDWord(X: DWord): DWord; register; assembler;
procedure XorBlock128(const I1: Byte128; const I2: Byte128; var O1:
Byte128);
procedure XorBlock64(const I1: Byte64; const I2: Byte64; var O1: Byte64);
procedure IncBlock(P: PByteArray; Len: integer);
procedure SwapHalf(var x: Byte128);
procedure ByteDword(const x: PByteArray; var y: Word128);
procedure DWordByte(const x: Word128; var y: PByteArray);
function RotBlock128(const x: Byte128; const n: integer): Byte128;
procedure split128(const x: Byte128; var L: Byte64; var R: Byte64);
procedure merge128(const L: Byte64; const R: Byte64; var z: Byte128);

implementation

function LRot16(X: word; c: integer): word; assembler;
asm
  mov ecx,&c
  mov ax,&X
  rol ax,cl
  mov &Result,ax
end;

function RRot16(X: word; c: integer): word; assembler;
```

```
asm
  mov ecx,&c
  mov ax,&X
  ror ax,cl
  mov &Result,ax
end;

function LRot32(X: dword; c: integer): dword; register; assembler;
asm
  mov ecx, edx
  rol eax, cl
end;

function RRot32(X: dword; c: integer): dword; register; assembler;
asm
  mov ecx, edx
  ror eax, cl
end;

function SwapDWord(X: DWord): DWord; register; assembler;
asm
  xchg al,ah
  rol  eax,16
  xchg al,ah
end;

procedure XorBlock128(const I1: Byte128; const I2: Byte128; var O1:
Byte128);
var
  i: integer;
begin
  for i:= 0 to 15 do
    O1[i]:= I1[i] xor I2[i];
end;

procedure XorBlock64(const I1: Byte64; const I2: Byte64; var O1: Byte64);
var
  i: integer;
begin
  for i:= 0 to 7 do
    O1[i]:= I1[i] xor I2[i];
end;

procedure IncBlock(P: PByteArray; Len: integer);
begin
  Inc(P[Len-1]);
  if (P[Len-1]= 0) and (Len> 1) then
    IncBlock(P,Len-1);
```

```
end;

procedure SwapHalf(var x: Byte128);
var t:byte;
    i:integer;
begin
for i:=0 to 7 do
 begin
  t:=x[i];
  x[i]:=x[8+i];
  x[8+i]:=t;
 end;
end;

procedure ByteDword(const x: PByteArray; var y: Word128);
var i: Integer;
begin
  for i:=0 to 3 do
    begin
      y[i]:= x[i*4+0];
      y[i]:= y[i] shl 8;
      y[i]:= y[i] + x[i*4+1];
      y[i]:= y[i] shl 8;
      y[i]:= y[i] + x[i*4+2];
      y[i]:= y[i] shl 8;
      y[i]:= y[i] + x[i*4+3];
    end;
end;

procedure DWordByte(const x: Word128; var y: PByteArray);
var i: integer;
    rr,ll: word;
begin
  for i:=0 to 3 do
    begin
      rr:=(x[i] shl 16) shr 16;
      ll:=(x[i] shr 16);
      y[i*4+1]:=(ll shl 8) shr 8;
      y[i*4+0]:=(ll shr 8);
      y[i*4+3]:=(rr shl 8) shr 8;
      y[i*4+2]:=(rr shr 8);
    end;
end;

function pangkat(const x: integer; const y: integer): integer;
var res,i : integer;
begin
if y = 0 then
```

```
   result:=1
  else
   begin
    res:=1;
    for i:=1 to y do
      res:=res * x;
    result:=res;
   end;
end;

function RotBlock128(const x: Byte128; const n: integer): Byte128;
var y:Byte128;
    m:array[0..127] of integer;
    i,j,l,t:integer;
begin
FillChar(m,sizeof(m),0);
for i:=0 to 15 do
  begin
    l:=x[i];
    j:=7;
    while l>0 do
      begin
        m[8*i+j]:= l mod 2;
        l := l div 2;
        dec(j);
      end;
  end;
for i:= 1 to n do
  begin
    t:=m[0];
    for j:=0 to 126 do
        m[j]:=m[j+1];
    m[127]:=t;
  end;
for i:=0 to 15 do
  begin
    l:=0;
    for j:=0 to 7 do
      l:=l+pangkat(2,7-j)*m[i*8+j];
    y[i]:=l;
  end;
move(y,Result,sizeof(y));
end;

procedure split128(const x: Byte128; var L: Byte64; var R: Byte64);
var i:integer;
    Le,Ri: Byte64;
begin
```

```
  for i:=0 to 7 do
    begin
      Le[i]:=x[i];
      Ri[i]:=x[i+8];
    end;
move(Le,L,sizeof(Le));
move(Ri,R,sizeof(Ri));
end;

procedure merge128(const L: Byte64; const R: Byte64; var z: Byte128);
var i:integer;
begin
  for i:=0 to 7 do
    begin
      z[i]:=L[i];
      z[i+8]:=R[i];
    end;
end;

end.
```

*** Program Pembangkit Kunci**

```
procedure  Ekeygen(const n: integer; const key: Byte256);
var
    i: integer;
    f,ktemp:Byte128;
    d:Byte64;
begin
// key burn

FillChar(KL,sizeof(KL),0);
FillChar(KR,sizeof(KR),0);
FillChar(KA,sizeof(KA),0);
FillChar(KB,sizeof(KB),0);
FillChar(K,sizeof(K),0);
FillChar(K1,sizeof(K1),0);
FillChar(Kw,sizeof(Kw),0);

if n = 128 then
  begin
    for i:= 0 to 15 do
      KL[i]:=key[i];
    for i:=0 to 15 do
      KR[i]:=0;
  end
 else

if n = 192 then
  begin
    for i:= 0 to 15 do
      KL[i]:=key[i];
    for i:=0 to 7 do
      begin
        KR[i]:=key[i+16];
        KR[i+8]:=key[i+16];
      end;
  end

 else
if n = 256 then
  begin
    for i:= 0 to 15 do
      begin
        KL[i]:=key[i];
        KR[i]:=key[i+16];
      end;
  end;
```

```
// preschedule
XorBlock128(KL,KR,f);
Feistel(f,Sigma1,f);
Feistel(f,Sigma2,f);
XorBlock128(f,KL,f);
Feistel(f,Sigma3,f);
Feistel(f,Sigma4,f);
move(f,KA,sizeof(f));
XorBlock128(f,KR,f);
Feistel(f,Sigma5,f);
Feistel(f,Sigma6,f);
move(f,KB,sizeof(f));

if n = 128 then
   begin

     // Kw schedule
     split128(KL,Kw[1],Kw[2]);
     ktemp:=RotBlock128(KA,111);
     split128(ktemp,Kw[3],Kw[4]);

     // k schedule
     split128(KA,k[1],k[2]);
     ktemp:=RotBlock128(KL,15);     split128(ktemp,k[3],k[4]);
     ktemp:=RotBlock128(KA,15);     split128(ktemp,k[5],k[6]);
     ktemp:=RotBlock128(KL,45);     split128(ktemp,k[7],k[8]);
     ktemp:=RotBlock128(KA,45);     split128(ktemp,k[9],d);
     ktemp:=RotBlock128(KL,60);     split128(ktemp,d,k[10]);
     ktemp:=RotBlock128(KA,60);     split128(ktemp,k[11],k[12]);
     ktemp:=RotBlock128(KL,94);     split128(ktemp,k[13],k[14]);
     ktemp:=RotBlock128(KA,94);     split128(ktemp,k[15],k[16]);
     ktemp:=RotBlock128(KL,111);    split128(ktemp,k[17],k[18]);

     // K1 schedule
     ktemp:=RotBlock128(KA,30);     split128(ktemp,k1[1],k1[2]);
     ktemp:=RotBlock128(KL,77);     split128(ktemp,k1[3],k1[4]);
   end
  else

   begin
     // Kw schedule
     split128(KL,Kw[1],Kw[2]);
     ktemp:=RotBlock128(KB,111);
     split128(ktemp,Kw[3],Kw[4]);
```

```
    // k schedule
    split128(KB,k[1],k[2]);
    ktemp:=RotBlock128(KR,15);     split128(ktemp,k[3],k[4]);
    ktemp:=RotBlock128(KA,15);     split128(ktemp,k[5],k[6]);
    ktemp:=RotBlock128(KB,30);     split128(ktemp,k[7],k[8]);
    ktemp:=RotBlock128(KL,45);     split128(ktemp,k[9],k[10]);
    ktemp:=RotBlock128(KA,45);     split128(ktemp,k[11],k[12]);
    ktemp:=RotBlock128(KR,60);     split128(ktemp,k[13],k[14]);
    ktemp:=RotBlock128(KB,60);     split128(ktemp,k[15],k[16]);
    ktemp:=RotBlock128(KL,77);     split128(ktemp,k[17],k[18]);
    ktemp:=RotBlock128(KR,94);     split128(ktemp,k[19],k[20]);
    ktemp:=RotBlock128(KA,94);     split128(ktemp,k[21],k[22]);
    ktemp:=RotBlock128(KL,111);    split128(ktemp,k[23],k[24]);

    // K1 schedule
    ktemp:=RotBlock128(KR,30);     split128(ktemp,k1[1],k1[2]);
    ktemp:=RotBlock128(KL,60);     split128(ktemp,k1[3],k1[4]);
    ktemp:=RotBlock128(KA,77);     split128(ktemp,k1[5],k1[6]);
  end;
end;
```

*** Program Enkripsi**

```
procedure Encrypt(const p: Byte128; const key: string; var c: Byte128);
var i,round,keylen:integer;
    kn: Byte256;
    ktmp,ptmp: Byte128;
    N:array[0..24] of Byte128;
begin
  //key schedule

  keylen:=length(key);
  if keylen <= 16 then
     begin
       keylen:=128;
       round:=18;
     end
    else
  if keylen <= 24 then
     begin
       keylen:=192;
       round:=24;
     end
    else
     begin
       keylen:=256;
       round:=24;
     end;

  FillChar(kn,sizeof(kn),0);
  for i:=1 to length(key) do
      kn[i-1]:=ord(key[i]);
  move(p,ptmp,sizeof(p));
  Ekeygen(keylen,kn);

  // ecnrypt schedule
  // prewhitening
  FillChar(N,sizeof(N),0);
  merge128(kw[1],kw[2],ktmp);
  XorBlock128(ptmp,ktmp,N[0]);
  // round
  if keylen = 128 then
    begin
      for i:=1 to round do
        begin
          if (i = 6) or (i = 12) then
              FLayer(N[i-1],k[i],k1[2*(i div 6)-1],k1[(2*(i div 6))],N[i])
             else
              Feistel(N[i-1],k[i],N[i]);
        end;
    end
   else
    begin
      for i:=1 to round do
        begin
          if (i = 6) or (i = 12) or (i = 18) then
             FLayer(N[i-1],k[i],k1[2*(i div 6)-1],k1[2*(i div 6)+1],N[i])
            else
             Feistel(N[i-1],k[i],N[i]);
        end;
    end;
  // post whitening
  if keylen = 128 then
     begin
       merge128(Kw[3],Kw[4],ktmp);
       SwapHalf(N[18]);
       XorBlock128(N[18],ktmp,c);
     end
    else
     begin
       merge128(Kw[3],Kw[4],ktmp);
       SwapHalf(N[24]);
       XorBlock128(N[24],ktmp,c);
     end;
end;
```

*** Program Dekripsi**

```
procedure Decrypt(const c: Byte128; const key: string; var p: Byte128);
var i,round,keylen:integer;
    kn: Byte256;
    ktmp,ptmp: Byte128;
    N:array[0..24] of Byte128;
begin
  //key schedule

  keylen:=length(key);
  if keylen <= 16 then
     begin
       keylen:=128;
       round:=18;
     end
    else
  if keylen <= 24 then
     begin
       keylen:=192;
       round:=24;
     end
    else
     begin
       keylen:=256;
       round:=24;
     end;

  FillChar(kn,sizeof(kn),0);
  for i:=1 to length(key) do
      kn[i-1]:=ord(key[i]);
  move(c,ptmp,sizeof(c));
  Ekeygen(keylen,kn);

  // ecnrypt schedule
  // prewhitening
  FillChar(N,sizeof(N),0);
  merge128(kw[3],kw[4],ktmp);
  XorBlock128(ptmp,ktmp,N[round]);
  SwapHalf(N[round]);
  // round
  if keylen = 128 then
    begin
      for i:=round downto 1 do
        begin
          if (i = 7) or (i = 13) then
```

```
            FLayerInv(N[i],k[i],k1[(2*(i-1) div 6)],k1[(2*(i-1) div 6)-1],N[i-1])
           else
            FeistelInv(N[i],k[i],N[i-1]);
       end;
   end
  else
   begin
     for i:=round downto 1 do
       begin
         if (i = 7) or (i = 13) or (i = 19) then
            FLayerInv(N[i],k[i],k1[(2*(i-1) div 6)],k1[(2*(i-1) div 6)-1],N[i-1])
           else
            FeistelInv(N[i],k[i],N[i-1]);
       end;
   end;
 // post whitening

      merge128(Kw[1],Kw[2],ktmp);
      XorBlock128(N[0],ktmp,p);

end;

end.
```

Lampiran B
Subtitution Box

Subtitution Box 1

```
cast_sbox1: array[0..255]of DWord= (
   $30FB40D4, $9FA0FF0B, $6BECCD2F, $3F258C7A,
   $1E213F2F, $9C004DD3, $6003E540, $CF9FC949,
   $BFD4AF27, $88BBBDB5, $E2034090, $98D09675,
   $6E63A0E0, $15C361D2, $C2E7661D, $22D4FF8E,
   $28683B6F, $C07FD059, $FF2379C8, $775F50E2,
   $43C340D3, $DF2F8656, $887CA41A, $A2D2BD2D,
   $A1C9E0D6, $346C4819, $61B76D87, $22540F2F,
   $2ABE32E1, $AA54166B, $22568E3A, $A2D341D0,
   $66DB40C8, $A784392F, $004DFF2F, $2DB9D2DE,
   $97943FAC, $4A97C1D8, $527644B7, $B5F437A7,
   $B82CBAEF, $D751D159, $6FF7F0ED, $5A097A1F,
   $827B68D0, $90ECF52E, $22B0C054, $BC8E5935,
   $4B6D2F7F, $50BB64A2, $D2664910, $BEE5812D,
   $B7332290, $E93B159F, $B48EE411, $4BFF345D,
   $FD45C240, $AD31973F, $C4F6D02E, $55FC8165,
   $D5B1CAAD, $A1AC2DAE, $A2D4B76D, $C19B0C50,
   $882240F2, $0C6E4F38, $A4E4BFD7, $4F5BA272,
   $564C1D2F, $C59C5319, $B949E354, $B04669FE,
   $B1B6AB8A, $C71358DD, $6385C545, $110F935D,
   $57538AD5, $6A390493, $E63D37E0, $2A54F6B3,
   $3A787D5F, $6276A0B5, $19A6FCDF, $7A42206A,
   $29F9D4D5, $F61B1891, $BB72275E, $AA508167,
   $38901091, $C6B505EB, $84C7CB8C, $2AD75A0F,
   $874A1427, $A2D1936B, $2AD286AF, $AA56D291,
   $D7894360, $425C750D, $93B39E26, $187184C9,
   $6C00B32D, $73E2BB14, $A0BEBC3C, $54623779,
   $64459EAB, $3F328B82, $7718CF82, $59A2CEA6,
   $04EE002E, $89FE78E6, $3FAB0950, $325FF6C2,
   $81383F05, $6963C5C8, $76CB5AD6, $D49974C9,
   $CA180DCF, $380782D5, $C7FA5CF6, $8AC31511,
   $35E79E13, $47DA91D0, $F40F9086, $A7E2419E,
   $31366241, $051EF495, $AA573B04, $4A805D8D,
   $548300D0, $00322A3C, $BF64CDDF, $BA57A68E,
   $75C6372B, $50AFD341, $A7C13275, $915A0BF5,
   $6B54BFAB, $2B0B1426, $AB4CC9D7, $449CCD82,
   $F7FBF265, $AB85C5F3, $1B55DB94, $AAD4E324,
   $CFA4BD3F, $2DEAA3E2, $9E204D02, $C8BD25AC,
   $EADF55B3, $D5BD9E98, $E31231B2, $2AD5AD6C,
   $954329DE, $ADBE4528, $D8710F69, $AA51C90F,
   $AA786BF6, $22513F1E, $AA51A79B, $2AD344CC,
   $7B5A41F0, $D37CFBAD, $1B069505, $41ECE491,
   $B4C332E6, $032268D4, $C9600ACC, $CE387E6D,
   $BF6BB16C, $6A70FB78, $0D03D9C9, $D4DF39DE,
   $E01063DA, $4736F464, $5AD328D8, $B347CC96,
   $75BB0FC3, $98511BFB, $4FFBCC35, $B58BCF6A,
 $E11F0ABC, $BFC5FE4A, $A70AEC10, $AC39570A,
 $3F04442F, $6188B153, $E0397A2E, $5727CB79,
 $9CEB418F, $1CACD68D, $2AD37C96, $0175CB9D,
 $C69DFF09, $C75B65F0, $D9DB40D8, $EC0E7779,
 $4744EAD4, $B11C3274, $DD24CB9E, $7E1C54BD,
 $F01144F9, $D2240EB1, $9675B3FD, $A3AC3755,
 $D47C27AF, $51C85F4D, $56907596, $A5BB15E6,
 $580304F0, $CA042CF1, $011A37EA, $8DBFAADB,
 $35BA3E4A, $3526FFA0, $C37B4D09, $BC306ED9,
 $98A52666, $5648F725, $FF5E569D, $0CED63D0,
 $7C63B2CF, $700B45E1, $D5EA50F1, $85A92872,
 $AF1FBDA7, $D4234870, $A7870BF3, $2D3B4D79,
 $42E04198, $0CD0EDE7, $26470DB8, $F881814C,
 $474D6AD7, $7C0C5E5C, $D1231959, $381B7298,
 $F5D2F4DB, $AB838653, $6E2F1E23, $83719C9E,
 $BD91E046, $9A56456E, $DC39200C, $20C8C571,
 $962BDA1C, $E1E696FF, $B141AB08, $7CCA89B9,
 $1A69E783, $02CC4843, $A2F7C579, $429EF47D,
 $427B169C, $5AC9F049, $DD8F0F00, $5C8165BF
);
```

Subtitution Box 2

```
cast_sbox2: array[0..255] of DWord = (
 $1F201094, $EF0BA75B, $69E3CF7E, $393F4380,
 $FE61CF7A, $EEC5207A, $55889C94, $72FC0651,
 $ADA7EF79, $4E1D7235, $D55A63CE, $DE0436BA,
 $99C430EF, $5F0C0794, $18DCDB7D, $A1D6EFF3,
 $A0B52F7B, $59E83605, $EE15B094, $E9FFD909,
 $DC440086, $EF944459, $BA83CCB3, $E0C3CDFB,
 $D1DA4181, $3B092AB1, $F997F1C1, $A5E6CF7B,
 $01420DDB, $E4E7EF5B, $25A1FF41, $E180F806,
 $1FC41080, $179BEE7A, $D37AC6A9, $FE5830A4,
 $98DE8B7F, $77E83F4E, $79929269, $24FA9F7B,
 $E113C85B, $ACC40083, $D7503525, $F7EA615F,
 $62143154, $0D554B63, $5D681121, $C866C359,
 $3D63CF73, $CEE234C0, $D4D87E87, $5C672B21,
 $071F6181, $39F7627F, $361E3084, $E4EB573B,
 $602F64A4, $D63ACD9C, $1BBC4635, $9E81032D,
 $2701F50C, $99847AB4, $A0E3DF79, $BA6CF38C,
 $10843094, $2537A95E, $F46F6FFE, $A1FF3B1F,
 $208CFB6A, $8F458C74, $D9E0A227, $4EC73A34,
 $FC884F69, $3E4DE8DF, $EF0E0088, $3559648D,
 $8A45388C, $1D804366, $721D9BFD, $A58684BB,
 $E8256333, $844E8212, $128D8098, $FED33FB4,
 $CE280AE1, $27E19BA5, $D5A6C252, $E49754BD,
 $C5D655DD, $EB667064, $77840B4D, $A1B6A801,
 $84DB26A9, $E0B56714, $21F043B7, $E5D05860,
 $54F03084, $066FF472, $A31AA153, $DADC4755,
 $B5625DBF, $68561BE6, $83CA6B94, $2D6ED23B,
 $ECCF01DB, $A6D3D0BA, $B6803D5C, $AF77A709,
 $33B4A34C, $397BC8D6, $5EE22B95, $5F0E5304,
 $81ED6F61, $20E74364, $B45E1378, $DE18639B,
 $881CA122, $B96726D1, $8049A7E8, $22B7DA7B,
 $5E552D25, $5272D237, $79D2951C, $C60D894C,
 $488CB402, $1BA4FE5B, $A4B09F6B, $1CA815CF,
 $A20C3005, $8871DF63, $B9DE2FCB, $0CC6C9E9,
 $0BEEFF53, $E3214517, $B4542835, $9F63293C,
 $EE41E729, $6E1D2D7C, $50045286, $1E6685F3,
 $F33401C6, $30A22C95, $31A70850, $60930F13,
 $73F98417, $A1269859, $EC645C44, $52C877A9,
 $CDFF33A6, $A02B1741, $7CBAD9A2, $2180036F,
 $50D99C08, $CB3F4861, $C26BD765, $64A3F6AB,
 $80342676, $25A75E7B, $E4E6D1FC, $20C710E6,
 $CDF0B680, $17844D3B, $31EEF84D, $7E0824E4,
 $2CCB49EB, $846A3BAE, $8FF77888, $EE5D60F6,
 $7AF75673, $2FDD5CDB, $A11631C1, $30F66F43,
 $B3FAEC54, $157FD7FA, $EF8579CC, $D152DE58,
 $DB2FFD5E, $8F32CE19, $306AF97A, $02F03EF8,
 $99319AD5, $C242FA0F, $A7E3EBB0, $C68E4906,
 $B8DA230C, $80823028, $DCDEF3C8, $D35FB171,
 $088A1BC8, $BEC0C560, $61A3C9E8, $BCA8F54D,
 $C72FEFFA, $22822E99, $82C570B4, $D8D94E89,
 $8B1C34BC, $301E16E6, $273BE979, $B0FFEAA6,
 $61D9B8C6, $00B24869, $B7FFCE3F, $08DC283B,
 $43DAF65A, $F7E19798, $7619B72F, $8F1C9BA4,
 $DC8637A0, $16A7D3B1, $9FC393B7, $A7136EEB,
 $C6BCC63E, $1A513742, $EF6828BC, $520365D6,
 $2D6A77AB, $3527ED4B, $821FD216, $095C6E2E,
 $DB92F2FB, $5EEA29CB, $145892F5, $91584F7F,
 $5483697B, $2667A8CC, $85196048, $8C4BACEA,
 $833860D4, $0D23E0F9, $6C387E8A, $0AE6D249,
 $B284600C, $D835731D, $DCB1C647, $AC4C56EA,
 $3EBD81B3, $230EABB0, $6438BC87, $F0B5B1FA,
 $8F5EA2B3, $FC184642, $0A036B7A, $4FB089BD,
 $649DA589, $A345415E, $5C038323, $3E5D3BB9,
 $43D79572, $7E6DD07C, $06DFDF1E, $6C6CC4EF,
 $7160A539, $73BFBE70, $83877605, $4523ECF1
);
```

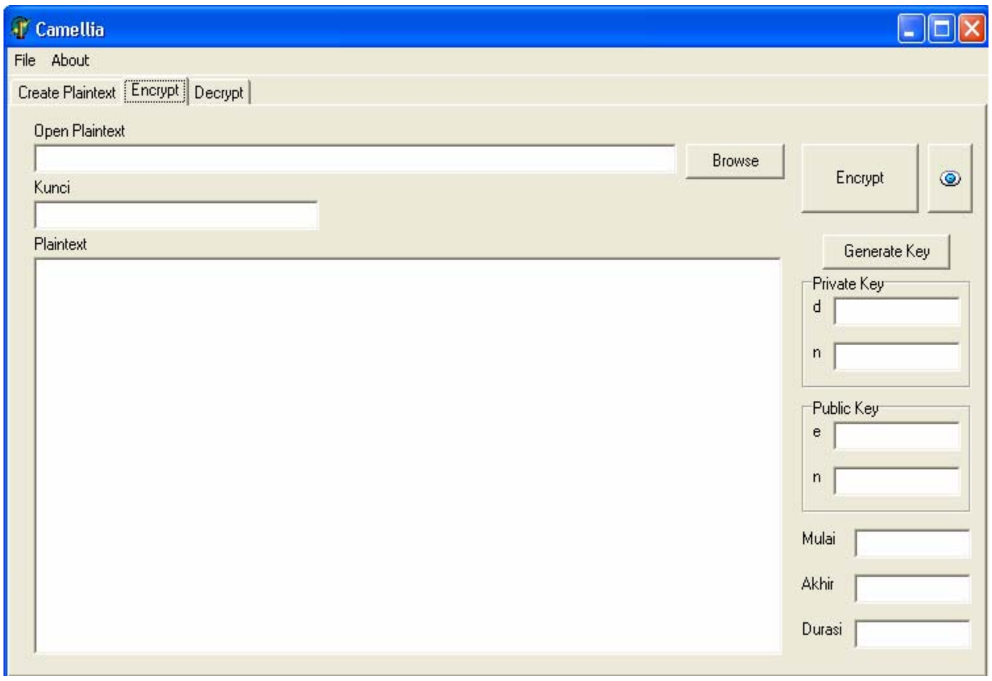
Subtitution Box 3

```
cast_sbox3: array[0..255] of DWord = (
 $8DEFC240, $25FA5D9F, $EB903DBF, $E810C907,
 $47607FFF, $369FE44B, $8C1FC644, $AECECA90,
 $BEB1F9BF, $EEFBCAEA, $E8CF1950, $51DF07AE,
 $920E8806, $F0AD0548, $E13C8D83, $927010D5,
 $11107D9F, $07647DB9, $B2E3E4D4, $3D4F285E,
 $B9AFA820, $FADE82E0, $A067268B, $8272792E,
 $553FB2C0, $489AE22B, $D4EF9794, $125E3FBC,
 $21FFFCEE, $825B1BFD, $9255C5ED, $1257A240,
 $4E1A8302, $BAE07FFF, $528246E7, $8E57140E,
 $3373F7BF, $8C9F8188, $A6FC4EE8, $C982B5A5,
 $A8C01DB7, $579FC264, $67094F31, $F2BD3F5F,
 $40FFF7C1, $1FB78DFC, $8E6BD2C1, $437BE59B,
 $99B03DBF, $B5DBC64B, $638DC0E6, $55819D99,
 $A197C81C, $4A012D6E, $C5884A28, $CCC36F71,
 $B843C213, $6C0743F1, $8309893C, $0FEDDD5F,
 $2F7FE850, $D7C07F7E, $02507FBF, $5AFB9A04,
 $A747D2D0, $1651192E, $AF70BF3E, $58C31380,
 $5F98302E, $727CC3C4, $0A0FB402, $0F7FEF82,
 $8C96FDAD, $5D2C2AAE, $8EE99A49, $50DA88B8,
 $8427F4A0, $1EAC5790, $796FB449, $8252DC15,
 $EFBD7D9B, $A672597D, $ADA840D8, $45F54504,
 $FA5D7403, $E83EC305, $4F91751A, $925669C2,
 $23EFE941, $A903F12E, $60270DF2, $0276E4B6,
 $94FD6574, $927985B2, $8276DBCB, $02778176,
 $F8AF918D, $4E48F79E, $8F616DDF, $E29D840E,
 $842F7D83, $340CE5C8, $96BBB682, $93B4B148,
 $EF303CAB, $984FAF28, $779FAF9B, $92DC560D,
 $224D1E20, $8437AA88, $7D29DC96, $2756D3DC,
 $8B907CEE, $B51FD240, $E7C07CE3, $E566B4A1,
 $C3E9615E, $3CF8209D, $6094D1E3, $CD9CA341,
 $5C76460E, $00EA983B, $D4D67881, $FD47572C,
 $F76CEDD9, $BDA8229C, $127DADAA, $438A074E,
 $1F97C090, $081BDB8A, $93A07EBE, $B938CA15,
 $97B03CFF, $3DC2C0F8, $8D1AB2EC, $64380E51,
 $68CC7BFB, $D90F2788, $12490181, $5DE5FFD4,
 $DD7EF86A, $76A2E214, $B9A40368, $925D958F,
 $4B39FFFA, $BA39AEE9, $A4FFD30B, $FAF7933B,
 $6D498623, $193CBCFA, $27627545, $825CF47A,
 $61BD8BA0, $D11E42D1, $CEAD04F4, $127EA392,
 $10428DB7, $8272A972, $9270C4A8, $127DE50B,
 $285BA1C8, $3C62F44F, $35C0EAA5, $E805D231,
 $428929FB, $B4FCDF82, $4FB66A53, $0E7DC15B,
 $1F081FAB, $108618AE, $FCFD086D, $F9FF2889,
 $694BCC11, $236A5CAE, $12DECA4D, $2C3F8CC5,
 $D2D02DFE, $F8EF5896, $E4CF52DA, $95155B67,
 $494A488C, $B9B6A80C, $5C8F82BC, $89D36B45,
 $3A609437, $EC00C9A9, $44715253, $0A874B49,
 $D773BC40, $7C34671C, $02717EF6, $4FEB5536,
 $A2D02FFF, $D2BF60C4, $D43F03C0, $50B4EF6D,
 $07478CD1, $006E1888, $A2E53F55, $B9E6D4BC,
 $A2048016, $97573833, $D7207D67, $DE0F8F3D,
 $72F87B33, $ABCC4F33, $7688C55D, $7B00A6B0,
 $947B0001, $570075D2, $F9BB88F8, $8942019E,
 $4264A5FF, $856302E0, $72DBD92B, $EE971B69,
 $6EA22FDE, $5F08AE2B, $AF7A616D, $E5C98767,
 $CF1FEBD2, $61EFC8C2, $F1AC2571, $CC8239C2,
 $67214CB8, $B1E583D1, $B7DC3E62, $7F10BDCE,
 $F90A5C38, $0FF0443D, $606E6DC6, $60543A49,
 $5727C148, $2BE98A1D, $8AB41738, $20E1BE24,
 $AF96DA0F, $68458425, $99833BE5, $600D457D,
 $282F9350, $8334B362, $D91D1120, $2B6D8DA0,
 $642B1E31, $9C305A00, $52BCE688, $1B03588A,
 $F7BAEFD5, $4142ED9C, $A4315C11, $83323EC5,
 $DFEF4636, $A133C501, $E9D3531C, $EE353783
);
```

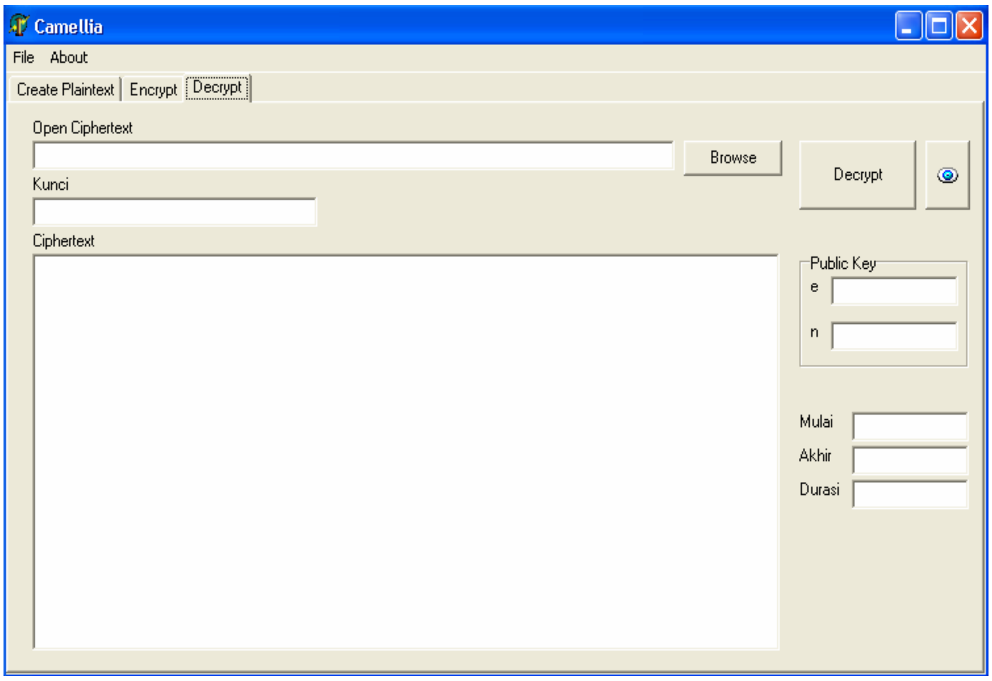
Subtitution Box 4

```
cast_sbox4: array[0..255] of DWord = (
 $9DB30420, $1FB6E9DE, $A7BE7BEF, $D273A298,
 $4A4F7BDB, $64AD8C57, $85510443, $FA020ED1,
 $7E287AFF, $E60FB663, $095F35A1, $79EBF120,
 $FD059D43, $6497B7B1, $F3641F63, $241E4ADF,
 $28147F5F, $4FA2B8CD, $C9430040, $0CC32220,
 $FDD30B30, $C0A5374F, $1D2D00D9, $24147B15,
 $EE4D111A, $0FCA5167, $71FF904C, $2D195FFE,
 $1A05645F, $0C13FEFE, $081B08CA, $05170121,
 $80530100, $E83E5EFE, $AC9AF4F8, $7FE72701,
 $D2B8EE5F, $06DF4261, $BB9E9B8A, $7293EA25,
 $CE84FFDF, $F5718801, $3DD64B04, $A26F263B,
 $7ED48400, $547EEBE6, $446D4CA0, $6CF3D6F5,
 $2649ABDF, $AEA0C7F5, $36338CC1, $503F7E93,
 $D3772061, $11B638E1, $72500E03, $F80EB2BB,
 $ABE0502E, $EC8D77DE, $57971E81, $E14F6746,
 $C9335400, $6920318F, $081DBB99, $FFC304A5,
 $4D351805, $7F3D5CE3, $A6C866C6, $5D5BCCA9,
 $DAEC6FEA, $9F926F91, $9F46222F, $3991467D,
 $A5BF6D8E, $1143C44F, $43958302, $D0214EEB,
 $022083B8, $3FB6180C, $18F8931E, $281658E6,
 $26486E3E, $8BD78A70, $7477E4C1, $B506E07C,
 $F32D0A25, $79098B02, $E4EABB81, $28123B23,
 $69DEAD38, $1574CA16, $DF871B62, $211C40B7,
 $A51A9EF9, $0014377B, $041E8AC8, $09114003,
 $BD59E4D2, $E3D156D5, $4FE876D5, $2F91A340,
 $557BE8DE, $00EAE4A7, $0CE5C2EC, $4DB4BBA6,
 $E756BDFF, $DD3369AC, $EC17B035, $06572327,
 $99AFC8B0, $56C8C391, $6B65811C, $5E146119,
 $6E85CB75, $BE07C002, $C2325577, $893FF4EC,
 $5BBFC92D, $D0EC3B25, $B7801AB7, $8D6D3B24,
 $20C763EF, $C366A5FC, $9C382880, $0ACE3205,
 $AAC9548A, $ECA1D7C7, $041AFA32, $1D16625A,
 $6701902C, $9B757A54, $31D477F7, $9126B031,
 $36CC6FDB, $C70B8B46, $D9E66A48, $56E55A79,
 $026A4CEB, $52437EFF, $2F8F76B4, $0DF980A5,
 $8674CDE3, $EDDA04EB, $17A9BE04, $2C18F4DF,
 $B7747F9D, $AB2AF7B4, $EFC34D20, $2E096B7C,
 $1741A254, $E5B6A035, $213D42F6, $2C1C7C26,
 $61C2F50F, $6552DAF9, $D2C231F8, $25130F69,
 $D8167FA2, $0418F2C8, $001A96A6, $0D1526AB,
 $63315C21, $5E0A72EC, $49BAFEFD, $187908D9,
 $8D0DBD86, $311170A7, $3E9B640C, $CC3E10D7,
 $D5CAD3B6, $0CAEC388, $F73001E1, $6C728AFF,
 $71EAE2A1, $1F9AF36E, $CFCBD12F, $C1DE8417,
 $AC07BE6B, $CB44A1D8, $8B9B0F56, $013988C3,
 $B1C52FCA, $B4BE31CD, $D8782806, $12A3A4E2,
 $6F7DE532, $58FD7EB6, $D01EE900, $24ADFFC2,
 $F4990FC5, $9711AAC5, $001D7B95, $82E5E7D2,
 $109873F6, $00613096, $C32D9521, $ADA121FF,
 $29908415, $7FBB977F, $AF9EB3DB, $29C9ED2A,
 $5CE2A465, $A730F32C, $D0AA3FE8, $8A5CC091,
 $D49E2CE7, $0CE454A9, $D60ACD86, $015F1919,
 $77079103, $DEA03AF6, $78A8565E, $DEE356DF,
 $21F05CBE, $8B75E387, $B3C50651, $B8A5C3EF,
 $D8EEB6D2, $E523BE77, $C2154529, $2F69EFDF,
 $AFE67AFB, $F470C4B2, $F3E0EB5B, $D6CC9876,
 $39E4460C, $1FDA8538, $1987832F, $CA007367,
 $A99144F8, $296B299E, $492FC295, $9266BEAB,
 $B5676E69, $9BD3DDDA, $DF7E052F, $DB25701C,
 $1B5E51EE, $F65324E6, $6AFCE36C, $0316CC04,
 $8644213E, $B7DC59D0, $7965291F, $CCD6FD43,
 $41823979, $932BCDF6, $B657C34D, $4EDFD282,
 $7AE5290C, $3CB9536B, $851E20FE, $9833557E,
 $13ECF0B0, $D3FFB372, $3F85C5C1, $0AEF7ED2
);
```

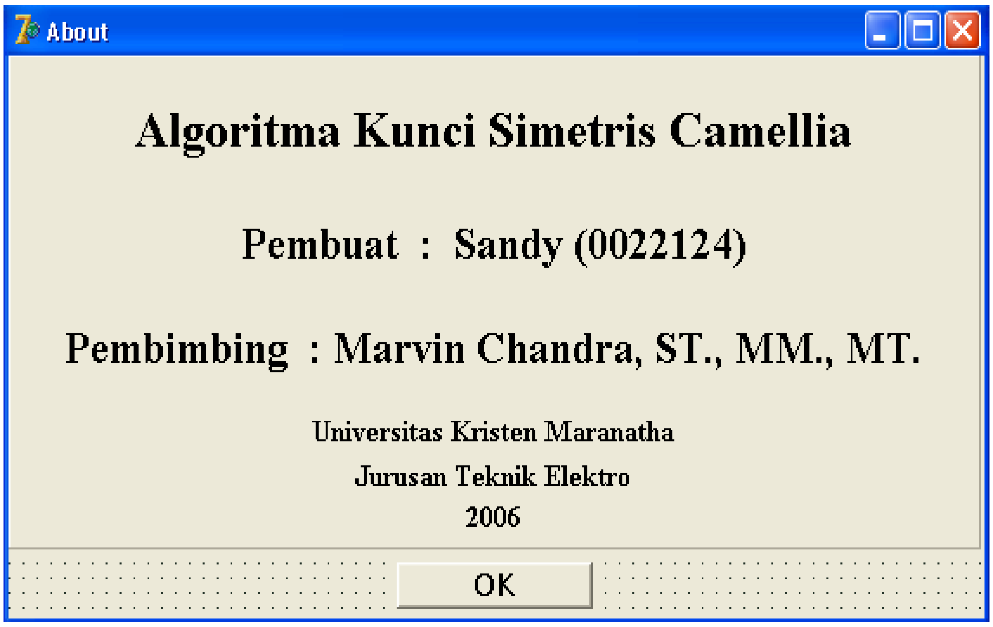
Lampiran C
Gambar Tampilan



Gambar Tampilan Program Enkripsi



Gambar Tampilan Program Dekripsi


About

Algoritma Kunci Simetris Camellia

Pembuat : Sandy (0022124)

Pembimbing : Marvin Chandra, ST., MM., MT.

Universitas Kristen Maranatha
Jurusan Teknik Elektro
2006

Gambar Tampilan About Box