Lampiran Istilah-istilah

Istilah

AES : Advanced Encryption Standard. tipe kunci berupa symetric key

Bit : Bilangan biner yang memiliki nilai 0 dan 1

Block cipher : Penyandian blok

Byte : Kumpulan sejumlah bilangan biner yang terdiri dari delapan bit dan mencptakan tempatnya sendiri.

Cipher : Kode atau penyandian

Ciphertext : Data yang telah dienkripsi dan tidak dapat dimengerti, pesan input pada proses dekripsi, pesan output dari proses enkripsi.

Invers : Proses merubah dari keadaan yang sudah diubah ke data asal.

IV : Initialization vector

NB : Number of block, jumlah blok standar adalah 4 byte(yangterdiri dari satu word atau sebanding dengan 32 bit)

NIST : National Institute of Standards and Technology.

NK : Number of key, panjang key, dalam AES variasinya 128, 192, 256

NR : Number of round, banyaknya perputaran atau round

Plaintext : Pesan asli yang ingin diubah kedalam bentuk yang tidak dapat dibaca oleh orang lain yang tidak berhak .input untuk proses enkripsi, output untuk proses dekripsi.

Rijndael : Pencipta Algoritma AES Vincent **Rij**men, COSIC ,Belgium dan Joan **Dae**men ,Proton World, Belgium

S-box : Substitution box suatu fungsi pemetaan nonlinier yang biasanya digunakan pada algoritma penyandian simetrik khususnya pada algoritma block cipher.

Xor : Exclusive Or hasil penjumlahan bilangan biner yang hasilnya dimodulo dua.

Word : Kumpulan bit yang terdiri dari 32 bit aauyang terdiridari 4 byte yang menciptakan tempatnya sendiri.

Lampiran Kode Ascii

Kode ascii

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space
1	1	001	SOH (start of heading)	33	21	041	!	!
2	2	002	STX (start of text)	34	22	042	"	"
3	3	003	ETX (end of text)	35	23	043	#	#
4	4	004	EOT (end of transmission)	36	24	044	$	$
5	5	005	ENQ (enquiry)	37	25	045	%	%
6	6	006	ACK (acknowledge)	38	26	046	&	&
7	7	007	BEL (bell)	39	27	047	'	'
8	8	010	BS (backspace)	40	28	050	(	(
9	9	011	TAB (horizontal tab)	41	29	051	)	)
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*
11	B	013	VT (vertical tab)	43	2B	053	+	+
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,
13	D	015	CR (carriage return)	45	2D	055	-	-
14	E	016	SO (shift out)	46	2E	056	.	.
15	F	017	SI (shift in)	47	2F	057	/	/
16	10	020	DLE (data link escape)	48	30	060	0	0
17	11	021	DC1 (device control 1)	49	31	061	1	1
18	12	022	DC2 (device control 2)	50	32	062	2	2
19	13	023	DC3 (device control 3)	51	33	063	3	3
20	14	024	DC4 (device control 4)	52	34	064	4	4
21	15	025	NAK (negative acknowledge)	53	35	065	5	5
22	16	026	SYN (synchronous idle)	54	36	066	6	6
23	17	027	ETB (end of trans. block)	55	37	067	7	7
24	18	030	CAN (cancel)	56	38	070	8	8
25	19	031	EM (end of medium)	57	39	071	9	9
26	1A	032	SUB (substitute)	58	3A	072	:	:
27	1B	033	ESC (escape)	59	3B	073	;	;
28	1C	034	FS (file separator)	60	3C	074	<	<
29	1D	035	GS (group separator)	61	3D	075	=	=
30	1E	036	RS (record separator)	62	3E	076	>	>
31	1F	037	US (unit separator)	63	3F	077	?	?

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
64	40	100	@	@	96	60	140	`	`
65	41	101	A	A	97	61	141	a	a
66	42	102	B	B	98	62	142	b	b
67	43	103	C	C	99	63	143	c	c
68	44	104	D	D	100	64	144	d	d
69	45	105	E	E	101	65	145	e	e
70	46	106	F	F	102	66	146	f	f
71	47	107	G	G	103	67	147	g	g
72	48	110	H	H	104	68	150	h	h
73	49	111	I	I	105	69	151	i	i
74	4A	112	J	J	106	6A	152	j	j
75	4B	113	K	K	107	6B	153	k	k
76	4C	114	L	L	108	6C	154	l	l
77	4D	115	M	M	109	6D	155	m	m
78	4E	116	N	N	110	6E	156	n	n
79	4F	117	O	O	111	6F	157	o	o
80	50	120	P	P	112	70	160	p	p
81	51	121	Q	Q	113	71	161	q	q
82	52	122	R	R	114	72	162	r	r
83	53	123	S	S	115	73	163	s	s
84	54	124	T	T	116	74	164	t	t
85	55	125	U	U	117	75	165	u	u
86	56	126	V	V	118	76	166	v	v
87	57	127	W	W	119	77	167	w	w
88	58	130	X	X	120	78	170	x	x
89	59	131	Y	Y	121	79	171	y	y
90	5A	132	Z	Z	122	7A	172	z	z
91	5B	133	[	[	123	7B	173	{	{
92	5C	134	\	\	124	7C	174			|
93	5D	135	]	]	125	7D	175	}	}
94	5E	136	^	^	126	7E	176	~	~
95	5F	137	_	_	127	7F	177	&#127;	DEL

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	|	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	169	_	185	╣	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	█	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	_
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	░	192	└	208	╨	224	α	240	≡		

Lampiran Enkripsi AES 128, 192, 256

Enkripsi Proses Sesuai kunci

Proses enkripsi AES 128

Round	Function
–	Add Round Key(State)
0	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
1	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
2	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
3	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
4	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
5	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
6	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
7	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
8	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
9	Add Round Key(Shift Row(Byte Sub(State)))

Proses enkripsi AES 192

Round	Function
–	Add Round Key(State)
0	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
1	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
2	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
3	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
4	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
5	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
6	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
7	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
8	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
9	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
10	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
11	Add Round Key(Shift Row(Byte Sub(State)))

Proses enkripsi AES 256

Round	Function
–	Add Round Key(State)
0	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
1	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
2	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
3	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
4	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
5	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
6	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
7	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
8	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
9	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
10	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
11	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
12	Add Round Key(Mix Column(Shift Row(Byte Sub(State))))
13	Add Round Key(Shift Row(Byte Sub(State)))

Lampiran Dekripsi AES 128, 192, 256

Dekripsi Proses Sesuai kunci

Proses dekripsi AES 128

Round	Function
–	Add Round Key(State)
0	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
1	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
2	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
3	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
4	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
5	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
6	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
7	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
8	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
9	Add Round Key(Byte Sub(Shift Row(State)))

Proses dekripsi AES 192

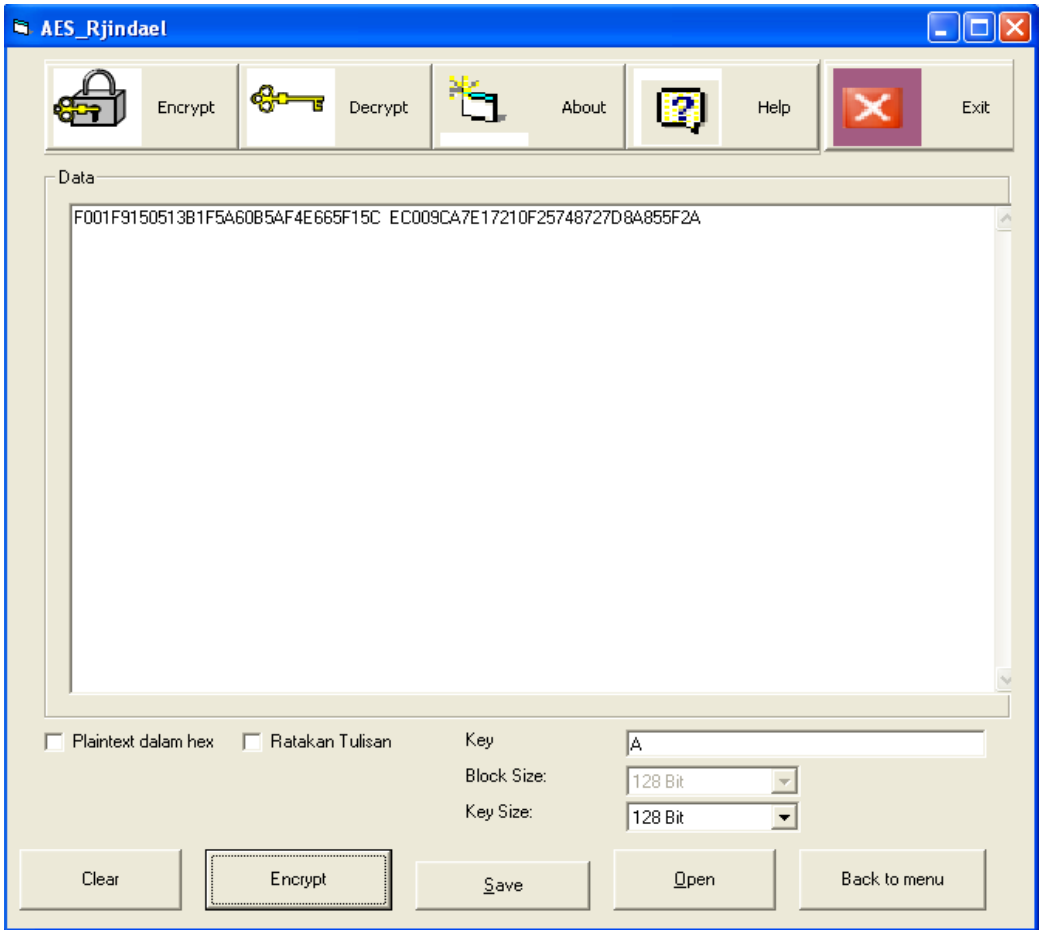
Round	Function
–	Add Round Key(State)
0	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
1	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
2	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
3	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
4	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
5	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
6	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
7	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
8	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
9	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
10	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
11	Add Round Key(Byte Sub(Shift Row(State)))

Proses dekripsi AES 256

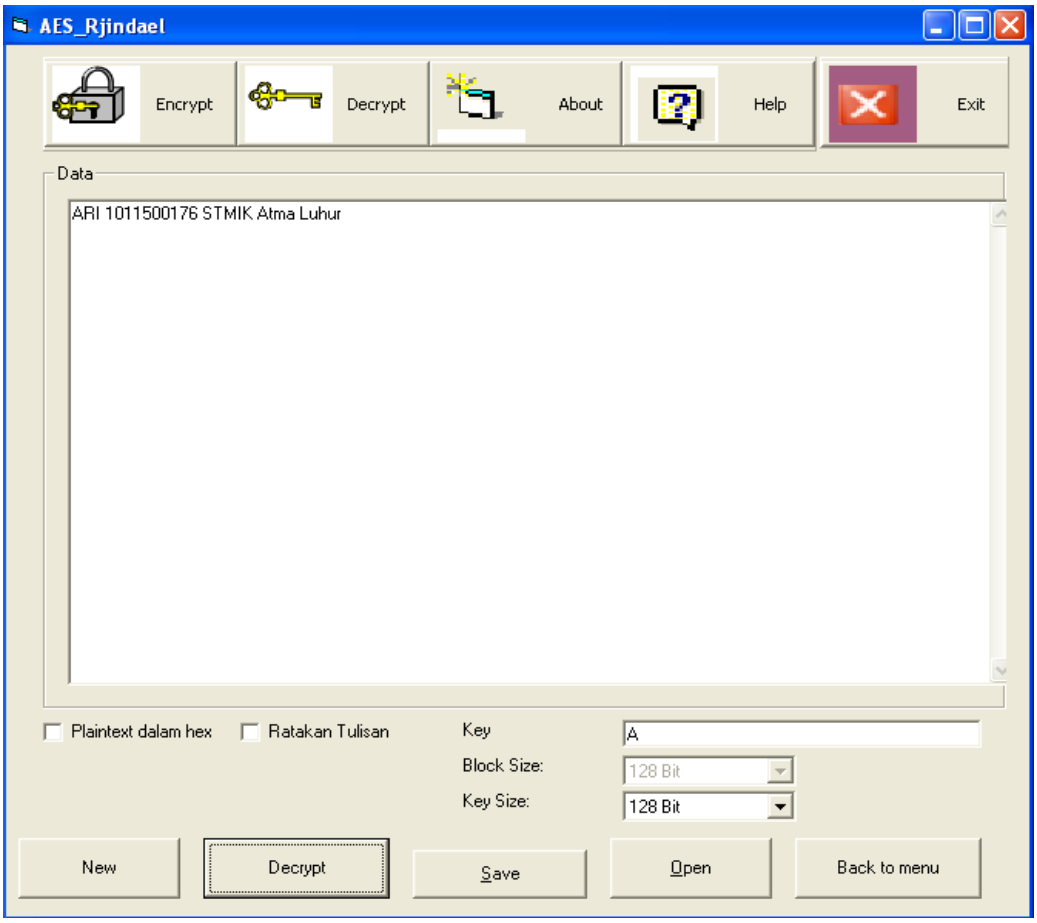
Round	Function
–	Add Round Key(State)
0	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
1	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
2	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
3	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
4	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
5	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
6	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
7	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
8	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
9	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
10	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
11	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
12	Mix Column(Add Round Key(Byte Sub(Shift Row(State))))
13	Add Round Key(Byte Sub(Shift Row(State)))


STMIK
ATMA LUHUR
PANGKAL PINANG

Lampiran Pengujian Modul Enkripsi



Lampiran Pengujian Modul Dekripsi



Lampiran Pengujian Modul About

APLIKASI PENGAMANAN SISTEM BASIS DATA DENGAN MENGGUNAKAN TEKNIK KRIPTOGRAFI

Algoritma AES-Rijndael

Block Size 128

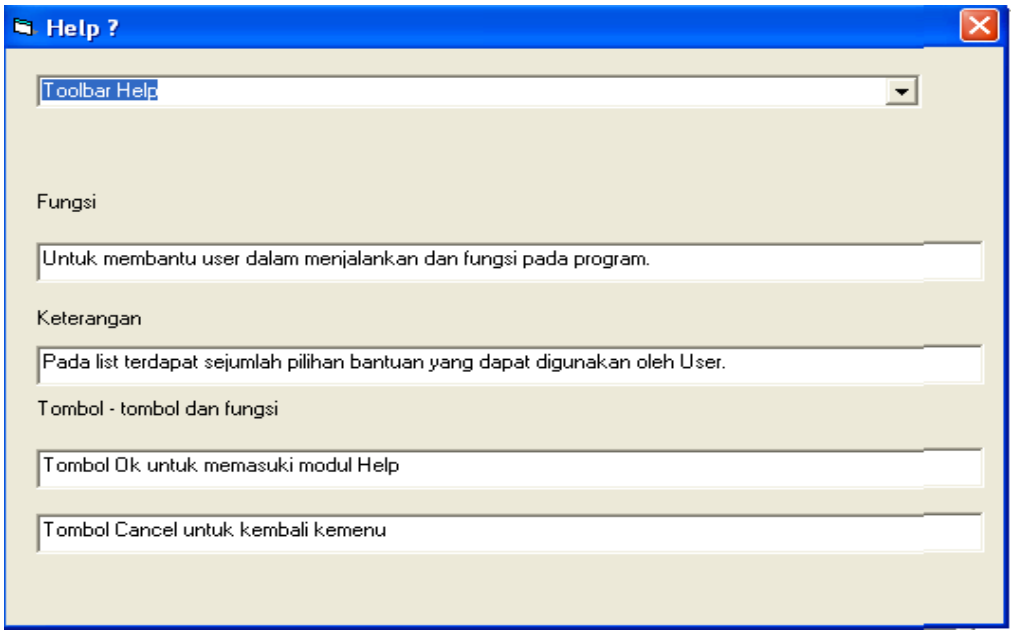
Key Size 128, 192, 256

By : ARI (1011500176)

STMIK Atma Luhur Pangkalpinang

Back to menu

Lampiran Pengujian Modul Help




STMIK
ATMA LUHUR
PANGKAL PINANG

Lampiran Coding

1. FrmMain

```
Option Explicit
        #Const SUPPORT_LEVEL = 0     'Default=0
        Private m_Rijndael As New CRijndael
Public Property Let Status(TheStatus As String)
   If Len(TheStatus) = 0 Then
      Me.Caption = App.Title
   Else
      Me.Caption = App.Title & " - " & TheStatus
   End If
   Me.Refresh
End Property
Private Sub cmdback_Click()
   Toolbar1.Enabled = True
   Call Form_Load
End Sub
Private Sub cmdEncrypt_Click()
   Dim pass()       As Byte
   Dim plaintext()   As Byte
   Dim ciphertext()  As Byte
   Dim KeyBits       As Long
   Dim BlockBits     As Long
   If Len(Text1.Text) = 0 Then
      MsgBox "Tidak ada Plaintext"
   Else
      If Len(txtPassword.Text) = 0 Then
         MsgBox "Tidak ada Kunci"
```

```
    Else
        KeyBits = cboKeySize.ItemData(cboKeySize.ListIndex)
        BlockBits = cboBlockSize.ItemData(cboBlockSize.ListIndex)
        pass = GetPassword
        Status = "Converting Text"
            If Check1.Value = 0 Then
           plaintext = StrConv(Text1.Text, vbFromUnicode)
        Else
           If HexDisplayRev(Text1.Text, plaintext) = 0 Then
              MsgBox "Text not Hex data"
              Status = ""
              Exit Sub
           End If
        End If
        Status = "Encrypting Data"
#If SUPPORT_LEVEL Then
        m_Rijndael.SetCipherKey pass, KeyBits, BlockBits
        m_Rijndael.ArrayEncrypt plaintext, ciphertext, 0, BlockBits
#Else
        m_Rijndael.SetCipherKey pass, KeyBits
        m_Rijndael.ArrayEncrypt plaintext, ciphertext, 0
#End If
        Status = "Converting Text"
        DisplayString Text1, HexDisplay(ciphertext, UBound(ciphertext) + 1, BlockBits \ 8)
        Status = ""
    End If
  End If
End Sub
Private Sub cmdDecrypt_Click()
  Dim pass()    As Byte
```

```
    Dim plaintext()   As Byte
    Dim ciphertext()  As Byte
    Dim KeyBits       As Long
    Dim BlockBits     As Long
    If Len(Text1.Text) = 0 Then
       MsgBox "Tidak ada Ciphertext"
    Else
       If Len(txtPassword.Text) = 0 Then
          MsgBox "Tidak ada Kunci"
       Else
          KeyBits = cboKeySize.ItemData(cboKeySize.ListIndex)
          BlockBits = cboBlockSize.ItemData(cboBlockSize.ListIndex)
          pass = GetPassword
          Status = "Converting Text"
          If HexDisplayRev(Text1.Text, ciphertext) = 0 Then
             MsgBox "Text not Hex data"
             Status = ""
             Exit Sub
          End If
          Status = "Decrypting Data"
#If SUPPORT_LEVEL Then
          m_Rijndael.SetCipherKey pass, KeyBits, BlockBits
          If m_Rijndael.ArrayDecrypt(plaintext, ciphertext, 0, BlockBits) <> 0 Then
             Status = ""
             Exit Sub
          End If
#Else
          m_Rijndael.SetCipherKey pass, KeyBits
          If m_Rijndael.ArrayDecrypt(plaintext, ciphertext, 0) <> 0 Then
             Status = ""
```

```
            Exit Sub
        End If
#End If
        Status = "Converting Text"
        If Check1.Value = 0 Then
            DisplayString Text1, StrConv(plaintext, vbUnicode)
        Else
            DisplayString Text1, HexDisplay(plaintext, UBound(plaintext) + 1, BlockBits \ 8)
        End If
        Status = ""
      End If
   End If
End Sub
Private Sub Encrypt()
   Cmdnc.Enabled = True
   cmdEncrypt.Enabled = True
   cmdSave.Enabled = True
   cmdOpen.Enabled = True
   cmdback.Enabled = True
   Cmdnc.Visible = True
   cmdEncrypt.Visible = True
   cmdSave.Visible = True
   cmdOpen.Visible = True
   Text1.Visible = True
   Label1.Visible = True
   Label2.Visible = True
   Label3.Visible = True
   txtPassword.Visible = True
   cboBlockSize.Visible = True
   cboKeySize.Visible = True
```

```
    chkTerminal.Visible = True
    Check1.Visible = True
    cmdback.Visible = True
End Sub
Private Sub Decrypt()
    Cmdnc.Enabled = True
    cmdDecrypt.Enabled = True
    cmdOpen.Enabled = True
    cmdback.Enabled = True
    cmdSave.Enabled = True
    Cmdnc.Visible = True
    cmdDecrypt.Visible = True
    cmdOpen.Visible = True
    Text1.Visible = True
    cmdSave.Visible = True
    Label1.Visible = True
    Label2.Visible = True
    Label3.Visible = True
    txtPassword.Visible = True
    cboBlockSize.Visible = True
    cboKeySize.Visible = True
    chkTerminal.Visible = True
    Check1.Visible = True
    cmdback.Visible = True
End Sub
Private Sub About()
    Unload FrmMain
    Load frmAbout
    frmAbout.Show
End Sub
```

```
Private Sub Help()
    Unload FrmMain
    Load FrmHelp
    FrmHelp.Show
End Sub
Private Sub Cmdnc_Click()
    If Cmdnc.Caption = "New" Then
        Text1.SetFocus
        Cmdnc.Caption = "Clear"
    ElseIf Cmdnc.Caption = "Clear" Then
        Text1.Text = ""
        Cmdnc.Caption = "New"
    End If
End Sub
Private Sub cmdOpen_Click()
Dim sFilename As String
    With CommonDialog1
    .DialogTitle = "Buka File - file yang telah Disimpan"
    .Filter = "Microsoft Word(*.doc|*.doc|Text Files(*.txt)|*.txt|All Files(*.*)|*.*|"
    .InitDir = App.Path
    .flags = cdlOFNFileMustExist Or cdlOFNHideReadOnly
    .ShowOpen
    sFilename = .FileName
    End With
    Open sFilename For Input As #1
    Text1.Text = StrConv(InputB$(LOF(1), 1), vbUnicode)
Close #1
    If Right(Text1.Text, 2) = vbCrLf Then _
    Text1.Text = Left(Text1.Text, Len(Text1.Text) - 2)
End Sub
```

```
Private Sub cmdSave_Click()

Dim sFilePath As String, Fn As Integer

   sFilePath = IIf(Right(App.Path, 1) = "\", App.Path, App.Path & "\")

fnExists:

   Fn = Fn + 1

   If FileExists(sFilePath & "crypt" & Fn & ".doc") Then GoTo fnExists

Open sFilePath & "crypt" & Fn & ".doc" For Output As #1

Print #1, Text1.Text: Close #1

   MsgBox "File Disimpan di ..." & vbCrLf & sFilePath & "Crypt" & Fn & ".doc", vbInformation, "Informasi
Penyimpananan"

End Sub

Private Sub Form_Load()

   Cmdnc.Enabled = False

   cmdEncrypt.Enabled = False

   cmdDecrypt.Enabled = False

   cmdOpen.Enabled = False

   cmdback.Enabled = False

   cmdSave.Enabled = False

   cmdSave.Visible = False

   Cmdnc.Visible = False

   cmdEncrypt.Visible = False

   cmdDecrypt.Visible = False

   cmdOpen.Visible = False

   Text1.Visible = False

   cmdback.Visible = False

   Label1.Visible = False

   Label2.Visible = False

   Label3.Visible = False

   txtPassword.Visible = False

   cboBlockSize.Visible = False
```

```
    cboKeySize.Visible = False
    chkTerminal.Visible = False
    Check1.Visible = False
    Toolbar1.Enabled = True
End Sub
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    If Button.Index = 1 Then
        If MsgBox("Apakah anda ingin mencoba button ini ?", vbOKCancel, "Informasi") = vbOK Then
            Toolbar1.Enabled = False
            Call Encrypt
        Else: Call Form_Load
        End If
    ElseIf Button.Index = 2 Then
        If MsgBox("Apakah anda ingin mencoba button ini ?", vbOKCancel, "Informasi") = vbOK Then
            Toolbar1.Enabled = False
            Call Decrypt
        Else: Call Form_Load
        End If
    ElseIf Button.Index = 3 Then
        If MsgBox("Apakah anda ingin mencoba button ini ?", vbOKCancel, "Informasi") = vbOK Then
        Toolbar1.Enabled = False
        Call About
        Else: Call Form_Load
        End If
    ElseIf Button.Index = 4 Then
        If MsgBox("Apakah anda ingin mencoba button ini ?", vbOKCancel, "Informasi") = vbOK Then
        Toolbar1.Enabled = False
        Call Help
        Else: Call Form_Load
        End If
```

```
    End If
End Sub
Private Sub DisplayString(TheTextBox As TextBox, ByVal TheString As String)
    If Len(TheString) < 65536 Then
        TheTextBox.Text = TheString
    Else
        MsgBox "Can not assign a String larger than 64k " & vbCrLf & _
            "to the Text property of a TextBox control." & vbCrLf & _
            "If you need to support Strings longer than 64k," & vbCrLf & _
            "you can use a RichTextBox control instead.", vbInformation
    End If
End Sub
Private Function HexDisplay(data() As Byte, n As Long, k As Long) As String
    Dim i As Long
    Dim j As Long
    Dim c As Long
    Dim data2() As Byte
    If LBound(data) = 0 Then
        ReDim data2(n * 4 - 1 + ((n - 1) \ k) * 4)
        j = 0
        For i = 0 To n - 1
            If i Mod k = 0 Then
                If i <> 0 Then
                    data2(j) = 32
                    data2(j + 2) = 32
                    j = j + 4
                End If
            End If
            c = data(i) \ 16&
            If c < 10 Then
```

```
            data2(j) = c + 48     ' "0"..."9"
        Else
            data2(j) = c + 55     ' "A"..."F"
        End If
        c = data(i) And 15&
        If c < 10 Then
            data2(j + 2) = c + 48 ' "0"..."9"
        Else
            data2(j + 2) = c + 55 ' "A"..."F"
        End If
        j = j + 4
    Next i
Debug.Assert j = UBound(data2) + 1
    HexDisplay = data2
  End If
End Function
Private Function HexDisplayRev(TheString As String, data() As Byte) As Long
  Dim i As Long
  Dim j As Long
  Dim c As Long
  Dim d As Long
  Dim n As Long
  Dim data2() As Byte
  n = 2 * Len(TheString)
  data2 = TheString
  ReDim data(n \ 4 - 1)
  d = 0
  i = 0
  j = 0
  Do While j < n
```

```
    c = data2(j)
    Select Case c
    Case 48 To 57    '"0" ... "9"
       If d = 0 Then   'high
          d = c
       Else            'low
          data(i) = (c - 48) Or ((d - 48) * 16&)
          i = i + 1
          d = 0
       End If
    Case 65 To 70   '"A" ... "F"
       If d = 0 Then   'high
          d = c - 7
       Else            'low
          data(i) = (c - 55) Or ((d - 48) * 16&)
          i = i + 1
          d = 0
       End If
    Case 97 To 102  '"a" ... "f"
       If d = 0 Then   'high
          d = c - 39
       Else            'low
          data(i) = (c - 87) Or ((d - 48) * 16&)
          i = i + 1
          d = 0
       End If
    End Select
    j = j + 2
Loop
n = i
```

```
    If n = 0 Then
        Erase data
    Else
        ReDim Preserve data(n - 1)
    End If
    HexDisplayRev = n
End Function
Private Function GetPassword() As Byte()
    Dim data() As Byte
    If Check1.Value = 0 Then
        data = StrConv(txtPassword.Text, vbFromUnicode)
        ReDim Preserve data(31)
    Else
        If HexDisplayRev(txtPassword.Text, data) <> (cboKeySize.ItemData(cboKeySize.ListIndex) \ 8) Then
            data = StrConv(txtPassword.Text, vbFromUnicode)
            ReDim Preserve data(31)
        End If
    End If
    GetPassword = data
End Function
Private Sub chkTerminal_Click()
    Static Text1FontName As String
    Static Text1FontBold As Boolean
    Static Text1FontSize As Long
    If chkTerminal.Value = 0 Then
        Text1.FontName = Text1FontName
        Text1.FontBold = Text1FontBold
        Text1.FontSize = Text1FontSize
    Else
        Text1FontName = Text1.FontName
```

```
        Text1FontBold = Text1.FontBold
        Text1FontSize = Text1.FontSize
        Text1.FontName = "Terminal"
    End If
End Sub
Private Sub Form_Initialize()
        cboBlockSize.AddItem "128 Bit"
        cboBlockSize.ItemData(cboBlockSize.NewIndex) = 128
    #If SUPPORT_LEVEL = 0 Then
        cboBlockSize.Enabled = False
    #Else
    #If SUPPORT_LEVEL = 2 Then
        cboBlockSize.AddItem "160 Bit"
        cboBlockSize.ItemData(cboBlockSize.NewIndex) = 160
        cmdSizeTest.Visible = True
    #End If
        cboBlockSize.AddItem "192 Bit"
        cboBlockSize.ItemData(cboBlockSize.NewIndex) = 192
    #If SUPPORT_LEVEL = 2 Then
        cboBlockSize.AddItem "224 Bit"
        cboBlockSize.ItemData(cboBlockSize.NewIndex) = 224
    #End If
        cboBlockSize.AddItem "256 Bit"
        cboBlockSize.ItemData(cboBlockSize.NewIndex) = 256
    #End If
        cboKeySize.AddItem "128 Bit"
        cboKeySize.ItemData(cboKeySize.NewIndex) = 128
    #If SUPPORT_LEVEL = 2 Then
        cboKeySize.AddItem "160 Bit"
        cboKeySize.ItemData(cboKeySize.NewIndex) = 160
```

```
    #End If
        cboKeySize.AddItem "192 Bit"
        cboKeySize.ItemData(cboKeySize.NewIndex) = 192
    #If SUPPORT_LEVEL = 2 Then
        cboKeySize.AddItem "224 Bit"
        cboKeySize.ItemData(cboKeySize.NewIndex) = 224
    #End If
        cboKeySize.AddItem "256 Bit"
        cboKeySize.ItemData(cboKeySize.NewIndex) = 256
        cboBlockSize.ListIndex = 0
        cboKeySize.ListIndex = 0
        txtPassword = ""
        Status = ""
End Sub
#If SUPPORT_LEVEL = 2 Then
Private Sub TestStuff(plaintext As String, passtext As String, ciphertext As String)
    Dim k As Long
    Dim p1() As Byte
    Dim c1() As Byte
    Dim cdata() As Byte
    Dim pdata() As Byte
    Dim pass() As Byte
    Dim Nk As Long
    Dim Nb As Long
    Dim n As Long
    k = HexDisplayRev(passtext, pass)
    Nk = k \ 4
    If Nk * 4 <> k Or Nk < 4 Or Nk > 8 Then Exit Sub
    n = HexDisplayRev(plaintext, pdata)
    Nb = n \ 4
```

```
    If Nb * 4 <> n Or Nb < 4 Or Nb > 8 Then Exit Sub

    If n <> HexDisplayRev(ciphertext, cdata) Then Exit Sub

    m_Rijndael.SetCipherKey pass, Nk * 32, Nb * 32

    m_Rijndael.ArrayEncrypt pdata, c1, 0, Nb * 32

    m_Rijndael.ArrayDecrypt p1, cdata, 0, Nb * 32

    Text1.Text = Text1.Text & vbCrLf & "ENCRYPT TEST  " & CStr(Nb * 4) & " byte block, " & CStr(Nk * 4) & " byte key" & vbCrLf

    Text1.Text = Text1.Text & "KEY:          " & passtext & IIf(UCase$(passtext) = HexDisplay(pass, Nk * 4, Nk * 4), " = ", "<>") & vbCrLf & String(14, 32) & HexDisplay(pass, Nk * 4, Nk * 4) & vbCrLf

    Text1.Text = Text1.Text & "PLAINTEXT:     " & plaintext & IIf(UCase$(plaintext) = HexDisplay(p1, Nb * 4, Nb * 4), " = ", "<>") & vbCrLf & String(14, 32) & HexDisplay(p1, Nb * 4, Nb * 4) & vbCrLf

    Text1.Text = Text1.Text & "CIPHERTEXT:    " & ciphertext & IIf(UCase$(ciphertext) = HexDisplay(c1, Nb * 4, Nb * 4), " = ", "<>") & vbCrLf & String(14, 32) & HexDisplay(c1, Nb * 4, Nb * 4) & vbCrLf

End Sub

Private Sub Toolbar2_ButtonClick(ByVal Button As MSComctlLib.Button)

    If MsgBox("Apakah anda ingin keluar dari program ini ?", vbOKCancel, "Informasi") = vbOK Then

      Unload Me

    Else: Call Form_Load

End If

End Sub
```

2. FrmHelp

```
Private Sub a()

    Txthelp1.Text = "Digunakan untuk menyandikan pesan yang anda ingin dirahasiakan."

    Txthelp2.Text = "Mengunakan AES,tombol  : New, Save, Proses(Encrypt), Open dan Back to menu."

    Txthelp3.Text = "Tombol Ok untuk memasuki modul Enkripsi"

    Txthelp4.Text = "Tombol Cancel untuk kembali kemenu"

End Sub

Private Sub b()

    Txthelp1.Text = "Digunakan untuk mengembalikan pesan yang dirahasiakan kepesan asli."

    Txthelp2.Text = " Mengunakan AES,tombol: New, Save, Proses(Decrypt), Open dan Back to menu."

    Txthelp3.Text = "Tombol Ok untuk memasuki modul Dekripsi"

    Txthelp4.Text = "Tombol Cancel untuk kembali kemenu"
```

```
End Sub
Private Sub c()
    Txthelp1.Text = "Untuk melihatketerangan singkat mengenai program dan pembuatnya."
    Txthelp2.Text = "copyright , Pangkalpinang 2013."
    Txthelp3.Text = "Tombol Ok untuk memasuki modul About"
    Txthelp4.Text = "Tombol Cancel untuk kembali kemenu"
End Sub
Private Sub d()
    Txthelp1.Text = "Untuk membantu user dalam menjalankan dan fungsi pada program."
    Txthelp2.Text = "Pada list terdapat sejumlah pilihan bantuan yang dapat digunakan oleh User."
    Txthelp3.Text = "Tombol Ok untuk memasuki modul Help"
    Txthelp4.Text = "Tombol Cancel untuk kembali kemenu"
End Sub
Private Sub e()
    Txthelp1.Text = "Digunakan untuk keluar dari program yang sedang berjalan."
    Txthelp3.Text = "Tombol Ok untuk keluar dari program"
    Txthelp4.Text = "Tombol Cancel untuk membatalkan bila tidak ingin keluar dari program"
End Sub
Private Sub f()
    Txthelp1.Text = "Digunakan untuk memulai input pesan atau untuk membersihkan layar."
    Txthelp3.Text = "New tombol untuk memulai baru"
    Txthelp4.Text = "Clear untuk membersihkan pesan"
End Sub
Private Sub g()
    Txthelp1.Text = "Digunakan untuk menyimpan data atau informasi yang anda inginkan."
    Txthelp2.Text = "Setiap save data akan disimpan dalam bentuk word(.doc) dengan nama CRYPT"
    Txthelp3.Text = "Tombol Ok untuk menyimpan data crypt ke-n"
    End Sub
Private Sub h()
    Txthelp1.Text = "Digunakan untuk membuka file yang telah anda simpan."
```

```
    Txthelp2.Text = "File yang dibuka adalah berbentuk word(.doc)tinggal mengetikan file yang dimaksud."
    Txthelp3.Text = "Tombol Ok untuk membuka file"
    Txthelp4.Text = "Tombol Cancel untuk kembali kemenu dan tidak jadi membuka file"
End Sub
Private Sub i()
    Txthelp1.Text = "Digunakan untuk Proses menyandikan pesan yang and ingin rahasiakan."
End Sub
Private Sub j()
    Txthelp1.Text = "Digunakan untuk proses mengembalikan pesan yang dirahasiakan kepesan asli."
End Sub
Private Sub k()
    Txthelp1.Text = "Digunakan untuk kembali ke menu Utama."
End Sub
Private Sub cmb1_Click()
    If cmb1.ListIndex = 1 Then
        Call a
    ElseIf cmb1.ListIndex = 2 Then
        Call b
    ElseIf cmb1.ListIndex = 3 Then
        Call c
    ElseIf cmb1.ListIndex = 4 Then
        Call d
    ElseIf cmb1.ListIndex = 5 Then
        Call e
    ElseIf cmb1.ListIndex = 6 Then
        Call f
    ElseIf cmb1.ListIndex = 7 Then
        Call g
    ElseIf cmb1.ListIndex = 8 Then
        Call h
```

```
    ElseIf cmb1.ListIndex = 9 Then
        Call i
    ElseIf cmb1.ListIndex = 10 Then
        Call j
    ElseIf cmb1.ListIndex = 11 Then
        Call k
    End If
End Sub
Private Sub Form_Load()
    cmb1.List(0) = "<Pilih Data Bantuan>"
    cmb1.List(1) = "Toolbar Encrypt"
    cmb1.List(2) = "Toolbar Decrypt"
    cmb1.List(3) = "Toolbar About"
    cmb1.List(4) = "Toolbar Help"
    cmb1.List(5) = "Toolbar Exit"
    cmb1.List(6) = "Button New"
    cmb1.List(7) = "Button Save"
    cmb1.List(8) = "Button Open"
    cmb1.List(9) = "Button Proses Encrypt"
    cmb1.List(10) = "Button Proses Decrypt"
    cmb1.List(11) = "Button Back to menu"
    cmb1.ListIndex = 0
End Sub
Private Sub Form_Unload(Cancel As Integer)
    Load FrmMain
    FrmMain.Show
    Unload frmAbout
End Sub
```

3. FrmAbout

```
Private Sub cmdmenu_Click()
```

```
    Load FrmMain
    FrmMain.Show
    Unload frmAbout
End Sub
```

4. MCommon (modules)

```
Option Explicit
Private Type OPENFILENAME
    lStructSize       As Long
    hwndOwner         As Long
    hInstance         As Long
    lpstrFilter       As String
    lpstrCustomFilter As String
    nMaxCustFilter    As Long
    nFilterIndex      As Long
    lpstrFile         As String
    nMaxFile          As Long
    lpstrFileTitle    As String
    nMaxFileTitle     As Long
    lpstrInitialDir   As String
    lpstrTitle        As String
    flags             As Long
    nFileOffset       As Integer
    nFileExtension    As Integer
    lpstrDefExt       As String
    lCustData         As Long
    lpfnHook          As Long
    lpTemplateName    As String
End Type
Private Type BrowseInfo
    hwndOwner         As Long
```

```
    pIDLRoot        As Long
    pszDisplayName  As Long
    lpszTitle       As Long
    ulFlags         As Long
    lpfnCallback    As Long
    lParam          As Long
    iImage          As Long
End Type
Private Const OFN_READONLY           As Long = &H1
Private Const OFN_OVERWRITEPROMPT    As Long = &H2
Private Const OFN_HIDEREADONLY       As Long = &H4
Private Const OFN_NOCHANGEDIR        As Long = &H8
Private Const OFN_SHOWHELP           As Long = &H10
Private Const OFN_ENABLEHOOK         As Long = &H20
Private Const OFN_ENABLETEMPLATE     As Long = &H40
Private Const OFN_ENABLETEMPLATEHANDLE As Long = &H80
Private Const OFN_NOVALIDATE         As Long = &H100
Private Const OFN_ALLOWMULTISELECT   As Long = &H200
Private Const OFN_EXTENSIONDIFFERENT  As Long = &H400
Private Const OFN_PATHMUSTEXIST      As Long = &H800
Private Const OFN_FILEMUSTEXIST      As Long = &H1000
Private Const OFN_CREATEPROMPT       As Long = &H2000
Private Const OFN_SHAREAWARE         As Long = &H4000
Private Const OFN_NOREADONLYRETURN   As Long = &H8000
Private Const OFN_NOTESTFILECREATE   As Long = &H10000
Private Const OFN_NONETWORKBUTTON    As Long = &H20000
Private Const OFN_NOLONGNAMES        As Long = &H40000
Private Const OFN_EXPLORER           As Long = &H80000
Private Const OFN_NODEREFERENCELINKS  As Long = &H100000
Private Const OFN_LONGNAMES          As Long = &H200000
```

```
Private Const OFN_SHAREFALLTHROUGH     As Long = 2

Private Const OFN_SHARENOWARN          As Long = 1

Private Const OFN_SHAREWARN            As Long = 0

Private Const BrowseForFolders         As Long = &H1

Private Const BrowseForComputers       As Long = &H1000

Private Const BrowseForPrinters        As Long = &H2000

Private Const BrowseForEverything      As Long = &H4000

Private Const CSIDL_BITBUCKET          As Long = 10

Private Const CSIDL_CONTROLS           As Long = 3

Private Const CSIDL_DESKTOP            As Long = 0

Private Const CSIDL_DRIVES             As Long = 17

Private Const CSIDL_FONTS              As Long = 20

Private Const CSIDL_NETHOOD            As Long = 18

Private Const CSIDL_NETWORK            As Long = 19

Private Const CSIDL_PERSONAL           As Long = 5

Private Const CSIDL_PRINTERS           As Long = 4

Private Const CSIDL_PROGRAMS           As Long = 2

Private Const CSIDL_RECENT             As Long = 8

Private Const CSIDL_SENDTO             As Long = 9

Private Const CSIDL_STARTMENU          As Long = 11

Private Const MAX_PATH                 As Long = 260

Private Declare Function GetOpenFileName Lib "comdlg32.dll" Alias "GetOpenFileNameA" (pOpenfilename
As OPENFILENAME) As Long

Private Declare Function GetSaveFileName Lib "comdlg32.dll" Alias "GetSaveFileNameA" (pOpenfilename
As OPENFILENAME) As Long

Private Declare Sub CoTaskMemFree Lib "ole32.dll" (ByVal hMem As Long)

Private Declare Function lstrcat Lib "kernel32" Alias "lstrcatA" (ByVal lpString1 As String, ByVal lpString2 As
String) As Long

Private Declare Function SHBrowseForFolder Lib "shell32" (lpBI As BrowseInfo) As Long

Private Declare Function SHGetPathFromIDList Lib "shell32" (ByVal pidList As Long, ByVal lpBuffer As
String) As Long
```

```
Private Declare Function SHGetSpecialFolderLocation Lib "shell32" (ByVal hwndOwner As Long, ByVal nFolder As Long, ListId As Long) As Long

Private Declare Function GetWindowsDirectory Lib "kernel32" Alias "GetWindowsDirectoryA" (ByVal lpBuffer As String, ByVal nSize As Long) As Long

Private Declare Function GetSystemDirectory Lib "kernel32" Alias "GetSystemDirectoryA" (ByVal lpBuffer As String, ByVal nSize As Long) As Long

Private Declare Function GetTempPath Lib "kernel32" Alias "GetTempPathA" (ByVal nBufferLength As Long, ByVal lpBuffer As String) As Long

Private Declare Function GetTempFileName Lib "kernel32" Alias "GetTempFileNameA" (ByVal lpszPath As String, ByVal lpPrefixString As String, ByVal wUnique As Long, ByVal lpTempFileName As String) As Long

Private Declare Function GetModuleHandle Lib "kernel32" Alias "GetModuleHandleA" (ByVal lpModuleName As String) As Long

Private Declare Function GetModuleFileName Lib "kernel32" Alias "GetModuleFileNameA" (ByVal hModule As Long, ByVal lpFileName As String, ByVal nSize As Long) As Long

Private Declare Function GetShortPathName Lib "kernel32" Alias "GetShortPathNameA" (ByVal lpszLongPath As String, ByVal lpszShortPath As String, ByVal cchBuffer As Long) As Long

Private Declare Function GetTickCount Lib "kernel32" () As Long

Public Function FileDialog(FormObject As Form, SaveDialog As Boolean, ByVal Title As String, ByVal Filter As String, Optional ByVal FileName As String, Optional ByVal Extention As String, Optional ByVal InitDir As String) As String

   Dim OFN   As OPENFILENAME

   Dim r     As Long

   If Len(FileName) > MAX_PATH Then Call MsgBox("Filename Length Overflow", vbExclamation, App.Title + " - FileDialog Function"): Exit Function

   FileName = FileName + String(MAX_PATH - Len(FileName), 0)

   With OFN

      .lStructSize = Len(OFN)

      .hwndOwner = 0 ' FormObject.hwnd

      .hInstance = App.hInstance

      .lpstrFilter = Replace(Filter, "|", vbNullChar)

      .lpstrFile = FileName

      .nMaxFile = MAX_PATH

      .lpstrFileTitle = Space$(MAX_PATH - 1)

      .nMaxFileTitle = MAX_PATH

      .lpstrInitialDir = InitDir
```

```
        .lpstrTitle = Title
        .flags = OFN_HIDEREADONLY Or OFN_OVERWRITEPROMPT Or OFN_CREATEPROMPT
        .lpstrDefExt = Extention
    End With
Dim L As Long
L = GetTickCount
    If SaveDialog Then r = GetSaveFileName(OFN) Else r = GetOpenFileName(OFN)
If GetTickCount - L < 20 Then
OFN.lpstrFile = ""
If SaveDialog Then r = GetSaveFileName(OFN) Else r = GetOpenFileName(OFN)
End If
    If r = 1 Then FileDialog = Left$(OFN.lpstrFile, InStr(1, OFN.lpstrFile + vbNullChar, vbNullChar) - 1)
End Function
Public Function BrowseFolders(FormObject As Form, sMessage As String) As String
    Dim b As BrowseInfo
    Dim r As Long
    Dim L As Long
    Dim f As String
    FormObject.Enabled = False
    With b
        .hwndOwner = FormObject.hWnd
        .lpszTitle = lstrcat(sMessage, "")
        .ulFlags = BrowseForFolders
    End With
    SHGetSpecialFolderLocation FormObject.hWnd, CSIDL_DRIVES, b.pIDLRoot
    r = SHBrowseForFolder(b)
    If r <> 0 Then     'A zero would mean cancel was pressed
        f = String(MAX_PATH, vbNullChar)
        SHGetPathFromIDList r, f
        CoTaskMemFree r
```

```
        L = InStr(1, f, vbNullChar) - 1
        If L < 0 Then L = 0
        f = Left(f, L)
        AddSlash f
    End If
    BrowseFolders = f
    FormObject.Enabled = True
End Function
Public Property Get WindowsDirectory() As String
    Static r As String
    If Len(r) = 0 Then
        Dim L As Long
        L = MAX_PATH
        r = String(L, 0)
        L = GetWindowsDirectory(r, L)
        If L > 0 Then
            r = Left$(r, L)
            AddSlash r
        Else
            r = ""
        End If
    End If
    WindowsDirectory = r
End Property
Public Property Get WindowsTempDirectory() As String
    Static m_WindowsTempDirectory As String
    If Len(m_WindowsTempDirectory) = 0 Then
        Dim Buffer As String
        Dim Length As Long
        Buffer = String(MAX_PATH, 0)
```

```
        Length = GetTempPath(MAX_PATH, Buffer)
        If Length > 0 Then
            m_WindowsTempDirectory = Left$(Buffer, Length)
            AddSlash m_WindowsTempDirectory
        End If
    End If
    WindowsTempDirectory = m_WindowsTempDirectory
End Property
Public Property Get WindowsSystemDirectory() As String
    Static m_WindowsSystemDirectory As String
    If Len(m_WindowsSystemDirectory) = 0 Then
        Dim Buffer As String
        Dim Length As Long
        Buffer = String(MAX_PATH, 0)
        Length = GetSystemDirectory(Buffer, MAX_PATH)
        If Length > 0 Then
            m_WindowsSystemDirectory = Left$(Buffer, Length)
            AddSlash m_WindowsSystemDirectory
        End If
    End If
    WindowsSystemDirectory = m_WindowsSystemDirectory
End Property
Public Property Get AppPath() As String
    Static m_AppPath As String 'Returns Program EXE File Name
    If Len(m_AppPath) = 0 Then
        Dim ret As Long
        Dim Length As Long
        Dim FilePath As String
        Dim FileHandle As Long
        FilePath = String(MAX_PATH, 0)
```

```
        FileHandle = GetModuleHandle(App.EXEName)

        ret = GetModuleFileName(FileHandle, FilePath, MAX_PATH)

        Length = InStr(1, FilePath, vbNullChar) - 1

        If Length > 0 Then m_AppPath = Left$(FilePath, Length)

    End If

    AppPath = m_AppPath

End Property

Public Property Get DefaultSettingsFile() As String

    Static m_DefaultSettingsFile As String

    If Len(m_DefaultSettingsFile) = 0 Then m_DefaultSettingsFile = FileTitleOnly(AppPath, True) & "Settings.Dat"

    DefaultSettingsFile = m_DefaultSettingsFile

End Property

Public Property Get DefaultLegendFile() As String

    Static m_DefaultLegendFile As String

    If Len(m_DefaultLegendFile) = 0 Then m_DefaultLegendFile = FileTitleOnly(AppPath, True) & "Legends.Txt"

    DefaultLegendFile = m_DefaultLegendFile

End Property

Public Function FileExists(FileName As String) As Boolean

    If Len(FileName) > 0 Then FileExists = (Len(Dir(FileName, vbNormal Or vbReadOnly Or vbHidden Or vbSystem Or vbArchive)) > 0)

End Function

Public Function DirectoryExists(ByVal Directory As String) As Boolean

    AddSlash Directory

    DirectoryExists = Len(Directory) > 0 And Len(Dir(Directory + "*.*", vbDirectory)) > 0

End Function

Public Function FileTitleOnly(FileName As String, Optional ReturnDirectory As Boolean) As String

    If ReturnDirectory Then

        FileTitleOnly = Left$(FileName, InStrRev(FileName, "\"))

    Else
```

```
        FileTitleOnly = Right$(FileName, Len(FileName) - InStrRev(FileName, "\"))

    End If

End Function

Public Sub AddSlash(Directory As String)

    If InStrRev(Directory, "\") <> Len(Directory) Then Directory = Directory + "\"

End Sub

Public Sub RemoveSlash(Directory As String)

    If Len(Directory) > 3 And InStrRev(Directory, "\") = Len(Directory) Then Directory = Left$(Directory, Len(Directory) - 1)

End Sub

Public Sub RidFile(FileName As String)

    If FileExists(FileName) Then

        SetAttr FileName, vbNormal

        Kill FileName

    End If

End Sub

Public Function GetShortName(ByVal FileName As String) As String

    Dim Buffer As String

    Dim Length As Long

    Buffer = String(MAX_PATH, 0)

    Length = GetShortPathName(FileName, Buffer, MAX_PATH)

    If Length > 0 Then GetShortName = Left$(Buffer, Length)

End Function

Public Function CreateTempFile(Optional ByVal Prefix As String, Optional Directory As String) As String

    Dim Buffer As String 'This code will CREATE a new temp file with a unique filename

    Dim Length As Long

    Buffer = String(MAX_PATH, 0)

    If Len(Prefix) = 0 Then Prefix = Left$(App.Title + "TMP", 3)

    If Not DirectoryExists(Directory) Then Directory = WindowsTempDirectory

    If GetTempFileName(Directory, Prefix, 0&, Buffer) = 0 Then Exit Function
```

```
    Length = InStr(1, Buffer, vbNullChar) - 1
    If Length > 0 Then CreateTempFile = Left$(Buffer, Length)
End Function
Public Function CreatePath(ByVal Path As String) As Boolean
    On Error GoTo Fail
    Dim i As Integer
    Dim s As String
    AddSlash Path
    Do
        i = InStr(i + 1, Path, "\")
        If i = 0 Then Exit Do
        s = Left$(Path, i - 1)
        If Not DirectoryExists(s) Then MkDir s
    Loop Until i = Len(Path)
    If DirectoryExists(Path) Then
        CreatePath = True
        Exit Function
    End If
Fail:
Call MsgBox(IIf(Err.Number = 0, "", "Error " + CStr(Err.Number) + ": " + Err.Description + vbCrLf) + "Could Not Create/Access Directory:" + vbCrLf + vbCrLf + Chr$(34) + Path + Chr$(34), vbExclamation, App.Title + " - CreatePath Function")
End Function
```

5. CRijndael (class modules)

```
Option Explicit
#Const SUPPORT_LEVEL = 0     'Default=0
#Const COMPILE_CONSTANTS = 0 'Default=0
Private Te0(255)     As Long
Private Te1(255)     As Long
Private Te2(255)     As Long
Private Te3(255)     As Long
```

```
Private Te4(255)      As Long
Private Td0(255)      As Long
Private Td1(255)      As Long
Private Td2(255)      As Long
Private Td3(255)      As Long
Private Td4(255)      As Long
#If SUPPORT_LEVEL Then
Private rco(28)       As Long
#Else
Private rco(9)        As Long
#End If
Private Nr            As Long
#If SUPPORT_LEVEL Then
Private fkey(119)     As Long
Private rkey(119)     As Long
#Else
Private fkey(59)      As Long
Private rkey(59)      As Long
#End If
Private Const MaxFileChunkSize As Long = 4000000
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (Destination As Any, Source As Any, ByVal Length As Long)
Private Sub CreateDecryptionKeys(Nb As Long)
   Dim i    As Long
   Dim j    As Long
   Dim k    As Long
   Dim s(3) As Byte
   i = 0
   j = Nb * Nr
   For k = 0 To Nr
```

```
        CopyMemory rkey(i), fkey(j), Nb * 4&
        i = i + Nb
        j = j - Nb
    Next k
    For i = Nb To Nb * Nr - 1
        CopyMemory s(0), rkey(i), 4&
        rkey(i) = Td0(Te4(s(0)) And &HFF&) Xor  Td1(Te4(s(1)) And &HFF&) Xor _
                Td2(Te4(s(2)) And &HFF&) Xor Td3(Te4(s(3)) And &HFF&)
    Next i
End Sub
#If SUPPORT_LEVEL Then
Public Function SetCipherKey(pass() As Byte, KeyBits As Long, BlockBits As Long) As Long
#Else
Public Function SetCipherKey(pass() As Byte, KeyBits As Long) As Long
#End If
    Dim i    As Long
    Dim j    As Long
    Dim s(3) As Byte
#If SUPPORT_LEVEL Then
    Select Case BlockBits
    Case 128
#End If
    '128 bit block size
    Select Case KeyBits
    Case 128
        i = 4
        CopyMemory fkey(0), pass(0), 4& * i
        For j = 0 To 9
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 4) Xor (Te4(s(0)) And &HFF000000)   Xor (Te4(s(3)) And &HFF0000) _
```

```
                    Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&) Xor rco(j)
        fkey(i + 1) = fkey(i - 3) Xor fkey(i)
        fkey(i + 2) = fkey(i - 2) Xor fkey(i + 1)
        fkey(i + 3) = fkey(i - 1) Xor fkey(i + 2)
        i = i + 4
      Next j
      Nr = 10
#If SUPPORT_LEVEL = 2 Then
   Case 160
      i = 5
      j = 0
      CopyMemory fkey(0), pass(0), 4& * i
      Do
        CopyMemory s(0), fkey(i - 1), 4&
        fkey(i) = fkey(i - 5) Xor (Te4(s(0)) And &HFF000000)  Xor (Te4(s(3)) And &HFF0000) _
                    Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&)  Xor rco(j)
        fkey(i + 1) = fkey(i - 4) Xor fkey(i)
        fkey(i + 2) = fkey(i - 3) Xor fkey(i + 1)
        If j = 8 Then Exit Do
        fkey(i + 3) = fkey(i - 2) Xor fkey(i + 2)
        fkey(i + 4) = fkey(i - 1) Xor fkey(i + 3)
        i = i + 5
        j = j + 1
      Loop
      Nr = 11
#End If
   Case 192
      i = 6
      j = 0
      CopyMemory fkey(0), pass(0), 4& * i
```

```
        Do
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 6) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&) Xor rco(j)
            fkey(i + 1) = fkey(i - 5) Xor fkey(i)
            fkey(i + 2) = fkey(i - 4) Xor fkey(i + 1)
            fkey(i + 3) = fkey(i - 3) Xor fkey(i + 2)
            If j = 7 Then Exit Do
            fkey(i + 4) = fkey(i - 2) Xor fkey(i + 3)
            fkey(i + 5) = fkey(i - 1) Xor fkey(i + 4)
            i = i + 6
            j = j + 1
        Loop
        Nr = 12
#If SUPPORT_LEVEL = 2 Then
    Case 224
        i = 7
        CopyMemory fkey(0), pass(0), 4& * i
        For j = 0 To 6
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 7) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&) Xor rco(j)
            fkey(i + 1) = fkey(i - 6) Xor fkey(i)
            fkey(i + 2) = fkey(i - 5) Xor fkey(i + 1)
            fkey(i + 3) = fkey(i - 4) Xor fkey(i + 2)
            CopyMemory s(0), fkey(i + 3), 4&
            fkey(i + 4) = fkey(i - 3) Xor (Te4(s(3)) And &HFF000000) Xor (Te4(s(2)) And &HFF0000) _
                            Xor (Te4(s(1)) And &HFF00&)  Xor (Te4(s(0)) And &HFF&)
            fkey(i + 5) = fkey(i - 2) Xor fkey(i + 4)
            fkey(i + 6) = fkey(i - 1) Xor fkey(i + 5)
```

```
        i = i + 7
      Next j
      Nr = 13
#End If
   Case 256
      i = 8
      j = 0
      CopyMemory fkey(0), pass(0), 4& * i
      Do
         CopyMemory s(0), fkey(i - 1), 4&
         fkey(i) = fkey(i - 8) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&) Xor rco(j)
         fkey(i + 1) = fkey(i - 7) Xor fkey(i)
         fkey(i + 2) = fkey(i - 6) Xor fkey(i + 1)
         fkey(i + 3) = fkey(i - 5) Xor fkey(i + 2)
         If j = 6 Then Exit Do
         CopyMemory s(0), fkey(i + 3), 4&
         fkey(i + 4) = fkey(i - 4) Xor (Te4(s(3)) And &HFF000000)  Xor (Te4(s(2)) And &HFF0000) _
                        Xor (Te4(s(1)) And &HFF00&)  Xor (Te4(s(0)) And &HFF&)
         fkey(i + 5) = fkey(i - 3) Xor fkey(i + 4)
         fkey(i + 6) = fkey(i - 2) Xor fkey(i + 5)
         fkey(i + 7) = fkey(i - 1) Xor fkey(i + 6)
         i = i + 8
         j = j + 1
      Loop
      Nr = 14
   Case Else
      Err.Raise 1, , "cRijndael.SetCipherKey - Illegal KeyBits Value"
      SetCipherKey = 1
      Exit Function
```

```
    End Select
    CreateDecryptionKeys 4
#If SUPPORT_LEVEL Then
#If SUPPORT_LEVEL = 2 Then
    Case 160
'160 bit block size
    Select Case KeyBits
    Case 128
        i = 4
        CopyMemory fkey(0), pass(0), 4& * i
        For j = 0 To 13
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 4) Xor (Te4(s(0)) And &HFF000000)  Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&)   Xor (Te4(s(1)) And &HFF&)  Xor rco(j)
            fkey(i + 1) = fkey(i - 3) Xor fkey(i)
            fkey(i + 2) = fkey(i - 2) Xor fkey(i + 1)
            fkey(i + 3) = fkey(i - 1) Xor fkey(i + 2)
            i = i + 4
        Next j
        Nr = 11
    Case 160
        i = 5
        CopyMemory fkey(0), pass(0), 4& * i
        For j = 0 To 10
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 5) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&)  Xor rco(j)
            fkey(i + 1) = fkey(i - 4) Xor fkey(i)
            fkey(i + 2) = fkey(i - 3) Xor fkey(i + 1)
            fkey(i + 3) = fkey(i - 2) Xor fkey(i + 2)
```

```
        fkey(i + 4) = fkey(i - 1) Xor fkey(i + 3)
        i = i + 5
    Next j
    Nr = 11
Case 192
    i = 6
    j = 0
    CopyMemory fkey(0), pass(0), 4& * i
    Do
        CopyMemory s(0), fkey(i - 1), 4&
        fkey(i) = fkey(i - 6) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                    Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&) Xor rco(j)
        fkey(i + 1) = fkey(i - 5) Xor fkey(i)
        fkey(i + 2) = fkey(i - 4) Xor fkey(i + 1)
        fkey(i + 3) = fkey(i - 3) Xor fkey(i + 2)
        fkey(i + 4) = fkey(i - 2) Xor fkey(i + 3)
        If j = 9 Then Exit Do
        fkey(i + 5) = fkey(i - 1) Xor fkey(i + 4)
        i = i + 6
        j = j + 1
    Loop
    Nr = 12
Case 224
    i = 7
    CopyMemory fkey(0), pass(0), 4& * i
    For j = 0 To 8
        CopyMemory s(0), fkey(i - 1), 4&
        fkey(i) = fkey(i - 7) Xor (Te4(s(0)) And &HFF000000)  Xor (Te4(s(3)) And &HFF0000) _
                    Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&) Xor rco(j)
        fkey(i + 1) = fkey(i - 6) Xor fkey(i)
```

```
        fkey(i + 2) = fkey(i - 5) Xor fkey(i + 1)
        fkey(i + 3) = fkey(i - 4) Xor fkey(i + 2)
        CopyMemory s(0), fkey(i + 3), 4&
        fkey(i + 4) = fkey(i - 3) Xor (Te4(s(3)) And &HFF000000) Xor (Te4(s(2)) And &HFF0000) _
                      Xor (Te4(s(1)) And &HFF00&)  Xor (Te4(s(0)) And &HFF&)
        fkey(i + 5) = fkey(i - 2) Xor fkey(i + 4)
        fkey(i + 6) = fkey(i - 1) Xor fkey(i + 5)
        i = i + 7
    Next j
    Nr = 13
Case 256
    i = 8
    j = 0
    CopyMemory fkey(0), pass(0), 4& * i
    Do
        CopyMemory s(0), fkey(i - 1), 4&
        fkey(i) = fkey(i - 8) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                  Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&) Xor rco(j)
        fkey(i + 1) = fkey(i - 7) Xor fkey(i)
        fkey(i + 2) = fkey(i - 6) Xor fkey(i + 1)
        If j = 8 Then Exit Do
        fkey(i + 3) = fkey(i - 5) Xor fkey(i + 2)
        CopyMemory s(0), fkey(i + 3), 4&
        fkey(i + 4) = fkey(i - 4) Xor (Te4(s(3)) And &HFF000000) Xor (Te4(s(2)) And &HFF0000) _
                      Xor (Te4(s(1)) And &HFF00&)  Xor (Te4(s(0)) And &HFF&)
        fkey(i + 5) = fkey(i - 3) Xor fkey(i + 4)
        fkey(i + 6) = fkey(i - 2) Xor fkey(i + 5)
        fkey(i + 7) = fkey(i - 1) Xor fkey(i + 6)
        i = i + 8
        j = j + 1
```

```
        Loop
        Nr = 14
    Case Else
        Err.Raise 1, , "cRijndael.SetCipherKey - Illegal KeyBits Value"
        SetCipherKey = 1
        Exit Function
    End Select
    CreateDecryptionKeys 5
#End If
    Case 192
    Select Case KeyBits
    Case 128
        i = 4
        j = 0
        CopyMemory fkey(0), pass(0), 4& * i
        Do
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 4) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&) Xor rco(j)
            fkey(i + 1) = fkey(i - 3) Xor fkey(i)
            If j = 18 Then Exit Do
            fkey(i + 2) = fkey(i - 2) Xor fkey(i + 1)
            fkey(i + 3) = fkey(i - 1) Xor fkey(i + 2)
            i = i + 4
            j = j + 1
        Loop
        Nr = 12
#If SUPPORT_LEVEL = 2 Then
    Case 160
        i = 5
```

```
        j = 0
        CopyMemory fkey(0), pass(0), 4& * i
        Do
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 5) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&)  Xor rco(j)
            fkey(i + 1) = fkey(i - 4) Xor fkey(i)
            fkey(i + 2) = fkey(i - 3) Xor fkey(i + 1)
            If j = 14 Then Exit Do
            fkey(i + 3) = fkey(i - 2) Xor fkey(i + 2)
            fkey(i + 4) = fkey(i - 1) Xor fkey(i + 3)
            i = i + 5
            j = j + 1
        Loop
        Nr = 12
#End If
    Case 192
        i = 6
        CopyMemory fkey(0), pass(0), 4& * i
        For j = 0 To 11
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 6) Xor (Te4(s(0)) And &HFF000000)  Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&)  Xor rco(j)
            fkey(i + 1) = fkey(i - 5) Xor fkey(i)
            fkey(i + 2) = fkey(i - 4) Xor fkey(i + 1)
            fkey(i + 3) = fkey(i - 3) Xor fkey(i + 2)
            fkey(i + 4) = fkey(i - 2) Xor fkey(i + 3)
            fkey(i + 5) = fkey(i - 1) Xor fkey(i + 4)
            i = i + 6
        Next j
```

```
        Nr = 12
#If SUPPORT_LEVEL = 2 Then
    Case 224 '(Nr+1)*Nb/Nk  (when to exit)
        i = 7
        CopyMemory fkey(0), pass(0), 4& * i
        For j = 0 To 10
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 7) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&)  Xor rco(j)
            fkey(i + 1) = fkey(i - 6) Xor fkey(i)
            fkey(i + 2) = fkey(i - 5) Xor fkey(i + 1)
            fkey(i + 3) = fkey(i - 4) Xor fkey(i + 2)
            CopyMemory s(0), fkey(i + 3), 4&
            fkey(i + 4) = fkey(i - 3) Xor (Te4(s(3)) And &HFF000000) _ Xor (Te4(s(2)) And &HFF0000) _
                            Xor (Te4(s(1)) And &HFF00&)  Xor (Te4(s(0)) And &HFF&)
            fkey(i + 5) = fkey(i - 2) Xor fkey(i + 4)
            fkey(i + 6) = fkey(i - 1) Xor fkey(i + 5)
            i = i + 7
        Next j
        Nr = 13
#End If
    Case 256
        i = 8
        j = 0
        CopyMemory fkey(0), pass(0), 4& * i
        Do
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 8) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&)Xor rco(j)
            fkey(i + 1) = fkey(i - 7) Xor fkey(i)
```

```
        If j = 10 Then Exit Do
        fkey(i + 2) = fkey(i - 6) Xor fkey(i + 1)
        fkey(i + 3) = fkey(i - 5) Xor fkey(i + 2)
        CopyMemory s(0), fkey(i + 3), 4&
        fkey(i + 4) = fkey(i - 4) Xor (Te4(s(3)) And &HFF000000)  Xor (Te4(s(2)) And &HFF0000) _
                        Xor (Te4(s(1)) And &HFF00&) Xor (Te4(s(0)) And &HFF&)
        fkey(i + 5) = fkey(i - 3) Xor fkey(i + 4)
        fkey(i + 6) = fkey(i - 2) Xor fkey(i + 5)
        fkey(i + 7) = fkey(i - 1) Xor fkey(i + 6)
        i = i + 8
        j = j + 1
      Loop
      Nr = 14
   Case Else
      Err.Raise 1, , "cRijndael.SetCipherKey - Illegal KeyBits Value"
      SetCipherKey = 1
      Exit Function
   End Select
   CreateDecryptionKeys 6
#If SUPPORT_LEVEL = 2 Then
   Case 224
'224 bit block size
   Select Case KeyBits
   Case 128
      i = 4
      j = 0
      CopyMemory fkey(0), pass(0), 4& * i
      Do
         CopyMemory s(0), fkey(i - 1), 4&
         fkey(i) = fkey(i - 4) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
```

```
                    Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&)Xor rco(j)
      fkey(i + 1) = fkey(i - 3) Xor fkey(i)
      If j = 23 Then Exit Do
      fkey(i + 2) = fkey(i - 2) Xor fkey(i + 1)
      fkey(i + 3) = fkey(i - 1) Xor fkey(i + 2)
      i = i + 4
      j = j + 1
   Loop
   Nr = 13
Case 160
   i = 5
   j = 0
   CopyMemory fkey(0), pass(0), 4& * i
   Do
      CopyMemory s(0), fkey(i - 1), 4&
      fkey(i) = fkey(i - 5) Xor (Te4(s(0)) And &HFF000000)Xor (Te4(s(3)) And &HFF0000) _
                    Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&)  Xor rco(j)
      fkey(i + 1) = fkey(i - 4) Xor fkey(i)
      fkey(i + 2) = fkey(i - 3) Xor fkey(i + 1)
      If j = 18 Then Exit Do
      fkey(i + 3) = fkey(i - 2) Xor fkey(i + 2)
      fkey(i + 4) = fkey(i - 1) Xor fkey(i + 3)
      i = i + 5
      j = j + 1
   Loop
   Nr = 13
Case 192
   i = 6
   j = 0
   CopyMemory fkey(0), pass(0), 4& * i
```

```
    Do
      CopyMemory s(0), fkey(i - 1), 4&
      fkey(i) = fkey(i - 6) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&) Xor rco(j)
      fkey(i + 1) = fkey(i - 5) Xor fkey(i)
      If j = 15 Then Exit Do
      fkey(i + 2) = fkey(i - 4) Xor fkey(i + 1)
      fkey(i + 3) = fkey(i - 3) Xor fkey(i + 2)
      fkey(i + 4) = fkey(i - 2) Xor fkey(i + 3)
      fkey(i + 5) = fkey(i - 1) Xor fkey(i + 4)
      i = i + 6
      j = j + 1
    Loop
    Nr = 13
  Case 224
    i = 7
    CopyMemory fkey(0), pass(0), 4& * i
    For j = 0 To 12
      CopyMemory s(0), fkey(i - 1), 4&
      fkey(i) = fkey(i - 7) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&) Xor rco(j)
      fkey(i + 1) = fkey(i - 6) Xor fkey(i)
      fkey(i + 2) = fkey(i - 5) Xor fkey(i + 1)
      fkey(i + 3) = fkey(i - 4) Xor fkey(i + 2)
      CopyMemory s(0), fkey(i + 3), 4&
      fkey(i + 4) = fkey(i - 3) Xor (Te4(s(3)) And &HFF000000) Xor (Te4(s(2)) And &HFF0000)_
                    Xor (Te4(s(1)) And &HFF00&) Xor (Te4(s(0)) And &HFF&)
      fkey(i + 5) = fkey(i - 2) Xor fkey(i + 4)
      fkey(i + 6) = fkey(i - 1) Xor fkey(i + 5)
      i = i + 7
```

```
    Next j
    Nr = 13
Case 256
    i = 8
    j = 0
    CopyMemory fkey(0), pass(0), 4& * i
    Do
        CopyMemory s(0), fkey(i - 1), 4&
        fkey(i) = fkey(i - 8) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&)Xor rco(j)
        If j = 12 Then Exit Do
        fkey(i + 1) = fkey(i - 7) Xor fkey(i)
        fkey(i + 2) = fkey(i - 6) Xor fkey(i + 1)
        fkey(i + 3) = fkey(i - 5) Xor fkey(i + 2)
        CopyMemory s(0), fkey(i + 3), 4&
        fkey(i + 4) = fkey(i - 4) Xor (Te4(s(3)) And &HFF000000)  Xor (Te4(s(2)) And &HFF0000) _
                            Xor (Te4(s(1)) And &HFF00&)  Xor (Te4(s(0)) And &HFF&)
        fkey(i + 5) = fkey(i - 3) Xor fkey(i + 4)
        fkey(i + 6) = fkey(i - 2) Xor fkey(i + 5)
        fkey(i + 7) = fkey(i - 1) Xor fkey(i + 6)
        i = i + 8
        j = j + 1
    Loop
    Nr = 14
Case Else
    Err.Raise 1, , "cRijndael.SetCipherKey - Illegal KeyBits Value"
    SetCipherKey = 1
    Exit Function
End Select
CreateDecryptionKeys 7
```

```
#End If
    Case 256
    '256 bit block size
    Select Case KeyBits
    Case 128
        i = 4
        CopyMemory fkey(0), pass(0), 4& * i
        For j = 0 To 28
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 4) Xor (Te4(s(0)) And &HFF000000)  Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&) Xor (Te4(s(1)) And &HFF&) Xor rco(j)
            fkey(i + 1) = fkey(i - 3) Xor fkey(i)
            fkey(i + 2) = fkey(i - 2) Xor fkey(i + 1)
            fkey(i + 3) = fkey(i - 1) Xor fkey(i + 2)
            i = i + 4
        Next j
        Nr = 14
#If SUPPORT_LEVEL = 2 Then
    Case 160
        i = 5
        CopyMemory fkey(0), pass(0), 4& * i
        For j = 0 To 22
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 5) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&) Xor rco(j)
            fkey(i + 1) = fkey(i - 4) Xor fkey(i)
            fkey(i + 2) = fkey(i - 3) Xor fkey(i + 1)
            fkey(i + 3) = fkey(i - 2) Xor fkey(i + 2)
            fkey(i + 4) = fkey(i - 1) Xor fkey(i + 3)
            i = i + 5
```

```
        Next j
        Nr = 14
#End If
    Case 192
        i = 6
        CopyMemory fkey(0), pass(0), 4& * i
        For j = 0 To 18
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 6) Xor (Te4(s(0)) And &HFF000000) Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&) Xor rco(j)
            fkey(i + 1) = fkey(i - 5) Xor fkey(i)
            fkey(i + 2) = fkey(i - 4) Xor fkey(i + 1)
            fkey(i + 3) = fkey(i - 3) Xor fkey(i + 2)
            fkey(i + 4) = fkey(i - 2) Xor fkey(i + 3)
            fkey(i + 5) = fkey(i - 1) Xor fkey(i + 4)
            i = i + 6
        Next j
        Nr = 14
#If SUPPORT_LEVEL = 2 Then
    Case 224
        i = 7
        j = 0
        CopyMemory fkey(0), pass(0), 4& * i
        Do
            CopyMemory s(0), fkey(i - 1), 4&
            fkey(i) = fkey(i - 7) Xor (Te4(s(0)) And &HFF000000)  Xor (Te4(s(3)) And &HFF0000) _
                        Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&) Xor rco(j)
            If j = 16 Then Exit Do
            fkey(i + 1) = fkey(i - 6) Xor fkey(i)
            fkey(i + 2) = fkey(i - 5) Xor fkey(i + 1)
```

```
        fkey(i + 3) = fkey(i - 4) Xor fkey(i + 2)
        CopyMemory s(0), fkey(i + 3), 4&
        fkey(i + 4) = fkey(i - 3) Xor (Te4(s(3)) And &HFF000000)  Xor (Te4(s(2)) And &HFF0000) _
                        Xor (Te4(s(1)) And &HFF00&)  Xor (Te4(s(0)) And &HFF&)
        fkey(i + 5) = fkey(i - 2) Xor fkey(i + 4)
        fkey(i + 6) = fkey(i - 1) Xor fkey(i + 5)
        i = i + 7
        j = j + 1
      Loop
      Nr = 14
#End If
   Case 256
      i = 8
      CopyMemory fkey(0), pass(0), 4& * i
      For j = 0 To 13
        CopyMemory s(0), fkey(i - 1), 4&
        fkey(i) = fkey(i - 8) Xor (Te4(s(0)) And &HFF000000)  Xor (Te4(s(3)) And &HFF0000) _
                    Xor (Te4(s(2)) And &HFF00&)  Xor (Te4(s(1)) And &HFF&)  Xor rco(j)
        fkey(i + 1) = fkey(i - 7) Xor fkey(i)
        fkey(i + 2) = fkey(i - 6) Xor fkey(i + 1)
        fkey(i + 3) = fkey(i - 5) Xor fkey(i + 2)
        CopyMemory s(0), fkey(i + 3), 4&
        fkey(i + 4) = fkey(i - 4) Xor (Te4(s(3)) And &HFF000000) Xor (Te4(s(2)) And &HFF0000) _
                        Xor (Te4(s(1)) And &HFF00&)  Xor (Te4(s(0)) And &HFF&)
        fkey(i + 5) = fkey(i - 3) Xor fkey(i + 4)
        fkey(i + 6) = fkey(i - 2) Xor fkey(i + 5)
        fkey(i + 7) = fkey(i - 1) Xor fkey(i + 6)
        i = i + 8
      Next j
      Nr = 14
```

```
    Case Else
        Err.Raise 1, , "cRijndael.SetCipherKey - Illegal KeyBits Value"
        SetCipherKey = 1
        Exit Function
    End Select
    CreateDecryptionKeys 8
    Case Else
        Err.Raise 1, , "cRijndael.SetCipherKey - Illegal BlockBits Value"
        SetCipherKey = 1
        Exit Function
    End Select
#End If
End Function
#If SUPPORT_LEVEL Then
Public Function SetCipherKeyString(PassPhrase As String, KeyBits As Long, BlockBits As Long) As Long
    Dim pass() As Byte
    pass = StrConv(PassPhrase, vbFromUnicode)
    ReDim Preserve pass(31)
    SetCipherKeyString = SetCipherKey(pass, KeyBits, BlockBits)
End Function
#Else
Public Function SetCipherKeyString(PassPhrase As String, KeyBits As Long) As Long
    Dim pass() As Byte
    pass = StrConv(PassPhrase, vbFromUnicode)
    ReDim Preserve pass(31)
    SetCipherKeyString = SetCipherKey(pass, KeyBits)
End Function
#End If
Public Sub BlockEncrypt(plaintext() As Byte, ciphertext() As Byte, p As Long, q As Long)
    Dim i     As Long
```

```
Dim k     As Long
Dim t0    As Long
Dim t1    As Long
Dim t2    As Long
Dim t3    As Long
Dim s(15) As Byte
CopyMemory t0, plaintext(p + 0), 4&
CopyMemory t1, plaintext(p + 4), 4&
CopyMemory t2, plaintext(p + 8), 4&
CopyMemory t3, plaintext(p + 12), 4&
t0 = t0 Xor fkey(0)
t1 = t1 Xor fkey(1)
t2 = t2 Xor fkey(2)
t3 = t3 Xor fkey(3)
k = 4
For i = 1 To Nr - 1 'Nr is number of rounds
   CopyMemory s(0), t0, 4&
   CopyMemory s(4), t1, 4&
   CopyMemory s(8), t2, 4&
   CopyMemory s(12), t3, 4&
   t0 = Te0(s(0)) Xor Te1(s(5)) Xor Te2(s(10)) Xor Te3(s(15)) Xor fkey(k + 0)
   t1 = Te0(s(4)) Xor Te1(s(9)) Xor Te2(s(14)) Xor Te3(s(3)) Xor fkey(k + 1)
   t2 = Te0(s(8)) Xor Te1(s(13)) Xor Te2(s(2)) Xor Te3(s(7)) Xor fkey(k + 2)
   t3 = Te0(s(12)) Xor Te1(s(1)) Xor Te2(s(6)) Xor Te3(s(11)) Xor fkey(k + 3)
   k = k + 4
Next i
CopyMemory s(0), t0, 4&
CopyMemory s(4), t1, 4&
CopyMemory s(8), t2, 4&
CopyMemory s(12), t3, 4&
```

```
    t0 = (Te4(s(0)) And &HFF&) Xor (Te4(s(5)) And &HFF00&) Xor (Te4(s(10)) And &HFF0000) Xor (Te4(s(15)) And &HFF000000) Xor fkey(k + 0)

    t1 = (Te4(s(4)) And &HFF&) Xor (Te4(s(9)) And &HFF00&) Xor (Te4(s(14)) And &HFF0000) Xor (Te4(s(3)) And &HFF000000) Xor fkey(k + 1)

    t2 = (Te4(s(8)) And &HFF&) Xor (Te4(s(13)) And &HFF00&) Xor (Te4(s(2)) And &HFF0000) Xor (Te4(s(7)) And &HFF000000) Xor fkey(k + 2)

    t3 = (Te4(s(12)) And &HFF&) Xor (Te4(s(1)) And &HFF00&) Xor (Te4(s(6)) And &HFF0000) Xor (Te4(s(11)) And &HFF000000) Xor fkey(k + 3)

    CopyMemory ciphertext(q + 0), t0, 4&

    CopyMemory ciphertext(q + 4), t1, 4&

    CopyMemory ciphertext(q + 8), t2, 4&

    CopyMemory ciphertext(q + 12), t3, 4&

End Sub

Public Sub BlockDecrypt(plaintext() As Byte, ciphertext() As Byte, p As Long, q As Long)

    Dim i     As Long

    Dim k     As Long

    Dim t0    As Long

    Dim t1    As Long

    Dim t2    As Long

    Dim t3    As Long

    Dim s(15) As Byte

    CopyMemory t0, ciphertext(q + 0), 4&

    CopyMemory t1, ciphertext(q + 4), 4&

    CopyMemory t2, ciphertext(q + 8), 4&

    CopyMemory t3, ciphertext(q + 12), 4&

    t0 = t0 Xor rkey(0)

    t1 = t1 Xor rkey(1)

    t2 = t2 Xor rkey(2)

    t3 = t3 Xor rkey(3)

    k = 4

    For i = 1 To Nr - 1 'Nr is number of rounds

        CopyMemory s(0), t0, 4&
```

```
        CopyMemory s(4), t1, 4&
        CopyMemory s(8), t2, 4&
        CopyMemory s(12), t3, 4&
        t0 = Td0(s(0)) Xor Td1(s(13)) Xor Td2(s(10)) Xor Td3(s(7)) Xor rkey(k + 0)
        t1 = Td0(s(4)) Xor Td1(s(1)) Xor Td2(s(14)) Xor Td3(s(11)) Xor rkey(k + 1)
        t2 = Td0(s(8)) Xor Td1(s(5)) Xor Td2(s(2)) Xor Td3(s(15)) Xor rkey(k + 2)
        t3 = Td0(s(12)) Xor Td1(s(9)) Xor Td2(s(6)) Xor Td3(s(3)) Xor rkey(k + 3)
        k = k + 4
    Next i
    CopyMemory s(0), t0, 4&
    CopyMemory s(4), t1, 4&
    CopyMemory s(8), t2, 4&
    CopyMemory s(12), t3, 4&
    t0 = (Td4(s(0)) And &HFF&) Xor (Td4(s(13)) And &HFF00&) Xor (Td4(s(10)) And &HFF0000) Xor (Td4(s(7)) And &HFF000000) Xor rkey(k + 0)
    t1 = (Td4(s(4)) And &HFF&) Xor (Td4(s(1)) And &HFF00&) Xor (Td4(s(14)) And &HFF0000) Xor (Td4(s(11)) And &HFF000000) Xor rkey(k + 1)
    t2 = (Td4(s(8)) And &HFF&) Xor (Td4(s(5)) And &HFF00&) Xor (Td4(s(2)) And &HFF0000) Xor (Td4(s(15)) And &HFF000000) Xor rkey(k + 2)
    t3 = (Td4(s(12)) And &HFF&) Xor (Td4(s(9)) And &HFF00&) Xor (Td4(s(6)) And &HFF0000) Xor (Td4(s(3)) And &HFF000000) Xor rkey(k + 3)
    CopyMemory plaintext(p + 0), t0, 4&
    CopyMemory plaintext(p + 4), t1, 4&
    CopyMemory plaintext(p + 8), t2, 4&
    CopyMemory plaintext(p + 12), t3, 4&
End Sub
#If SUPPORT_LEVEL Then
#If SUPPORT_LEVEL = 2 Then
Public Sub Block160Encrypt(plaintext() As Byte, ciphertext() As Byte, p As Long, q As Long)
    Dim i     As Long
    Dim k     As Long
    Dim t0    As Long
```

```
Dim t1    As Long
Dim t2    As Long
Dim t3    As Long
Dim t4    As Long
Dim s(19) As Byte
CopyMemory t0, plaintext(p + 0), 4&
CopyMemory t1, plaintext(p + 4), 4&
CopyMemory t2, plaintext(p + 8), 4&
CopyMemory t3, plaintext(p + 12), 4&
CopyMemory t4, plaintext(p + 16), 4&
t0 = t0 Xor fkey(0)
t1 = t1 Xor fkey(1)
t2 = t2 Xor fkey(2)
t3 = t3 Xor fkey(3)
t4 = t4 Xor fkey(4)
k = 5
For i = 1 To Nr - 1 'Nr is number of rounds
   CopyMemory s(0), t0, 4&
   CopyMemory s(4), t1, 4&
   CopyMemory s(8), t2, 4&
   CopyMemory s(12), t3, 4&
   CopyMemory s(16), t4, 4&
   t0 = Te0(s(0)) Xor Te1(s(5)) Xor Te2(s(10)) Xor Te3(s(15)) Xor fkey(k + 0)
   t1 = Te0(s(4)) Xor Te1(s(9)) Xor Te2(s(14)) Xor Te3(s(19)) Xor fkey(k + 1)
   t2 = Te0(s(8)) Xor Te1(s(13)) Xor Te2(s(18)) Xor Te3(s(3)) Xor fkey(k + 2)
   t3 = Te0(s(12)) Xor Te1(s(17)) Xor Te2(s(2)) Xor Te3(s(7)) Xor fkey(k + 3)
   t4 = Te0(s(16)) Xor Te1(s(1)) Xor Te2(s(6)) Xor Te3(s(11)) Xor fkey(k + 4)
   k = k + 5
Next i
CopyMemory s(0), t0, 4&
```

```
    CopyMemory s(4), t1, 4&

    CopyMemory s(8), t2, 4&

    CopyMemory s(12), t3, 4&

    CopyMemory s(16), t4, 4&

    t0 = (Te4(s(0)) And &HFF&) Xor (Te4(s(5)) And &HFF00&) Xor (Te4(s(10)) And &HFF0000) Xor (Te4(s(15)) And &HFF000000) Xor fkey(k + 0)

    t1 = (Te4(s(4)) And &HFF&) Xor (Te4(s(9)) And &HFF00&) Xor (Te4(s(14)) And &HFF0000) Xor (Te4(s(19)) And &HFF000000) Xor fkey(k + 1)

    t2 = (Te4(s(8)) And &HFF&) Xor (Te4(s(13)) And &HFF00&) Xor (Te4(s(18)) And &HFF0000) Xor (Te4(s(3)) And &HFF000000) Xor fkey(k + 2)

    t3 = (Te4(s(12)) And &HFF&) Xor (Te4(s(17)) And &HFF00&) Xor (Te4(s(2)) And &HFF0000) Xor (Te4(s(7)) And &HFF000000) Xor fkey(k + 3)

    t4 = (Te4(s(16)) And &HFF&) Xor (Te4(s(1)) And &HFF00&) Xor (Te4(s(6)) And &HFF0000) Xor (Te4(s(11)) And &HFF000000) Xor fkey(k + 4)

    CopyMemory ciphertext(q + 0), t0, 4&

    CopyMemory ciphertext(q + 4), t1, 4&

    CopyMemory ciphertext(q + 8), t2, 4&

    CopyMemory ciphertext(q + 12), t3, 4&

    CopyMemory ciphertext(q + 16), t4, 4&

End Sub

Public Sub Block160Decrypt(plaintext() As Byte, ciphertext() As Byte, p As Long, q As Long)

    Dim i     As Long

    Dim k     As Long

    Dim t0    As Long

    Dim t1    As Long

    Dim t2    As Long

    Dim t3    As Long

    Dim t4    As Long

    Dim s(19) As Byte

    CopyMemory t0, ciphertext(q + 0), 4&

    CopyMemory t1, ciphertext(q + 4), 4&

    CopyMemory t2, ciphertext(q + 8), 4&
```

```
    CopyMemory t3, ciphertext(q + 12), 4&
    CopyMemory t4, ciphertext(q + 16), 4&
    t0 = t0 Xor rkey(0)
    t1 = t1 Xor rkey(1)
    t2 = t2 Xor rkey(2)
    t3 = t3 Xor rkey(3)
    t4 = t4 Xor rkey(4)
    k = 5
    For i = 1 To Nr - 1 'Nr is number of rounds
        CopyMemory s(0), t0, 4&
        CopyMemory s(4), t1, 4&
        CopyMemory s(8), t2, 4&
        CopyMemory s(12), t3, 4&
        CopyMemory s(16), t4, 4&
        t0 = Td0(s(0)) Xor Td1(s(17)) Xor Td2(s(14)) Xor Td3(s(11)) Xor rkey(k + 0)
        t1 = Td0(s(4)) Xor Td1(s(1)) Xor Td2(s(18)) Xor Td3(s(15)) Xor rkey(k + 1)
        t2 = Td0(s(8)) Xor Td1(s(5)) Xor Td2(s(2)) Xor Td3(s(19)) Xor rkey(k + 2)
        t3 = Td0(s(12)) Xor Td1(s(9)) Xor Td2(s(6)) Xor Td3(s(3)) Xor rkey(k + 3)
        t4 = Td0(s(16)) Xor Td1(s(13)) Xor Td2(s(10)) Xor Td3(s(7)) Xor rkey(k + 4)
        k = k + 5
    Next i
    CopyMemory s(0), t0, 4&
    CopyMemory s(4), t1, 4&
    CopyMemory s(8), t2, 4&
    CopyMemory s(12), t3, 4&
    CopyMemory s(16), t4, 4&
    t0 = (Td4(s(0)) And &HFF&) Xor (Td4(s(17)) And &HFF00&) Xor (Td4(s(14)) And &HFF0000) Xor
(Td4(s(11)) And &HFF000000) Xor rkey(k + 0)
    t1 = (Td4(s(4)) And &HFF&) Xor (Td4(s(1)) And &HFF00&) Xor (Td4(s(18)) And &HFF0000) Xor
(Td4(s(15)) And &HFF000000) Xor rkey(k + 1)
```

```
    t2 = (Td4(s(8)) And &HFF&) Xor (Td4(s(5)) And &HFF00&) Xor (Td4(s(2)) And &HFF0000) Xor (Td4(s(19)) And &HFF000000) Xor rkey(k + 2)

    t3 = (Td4(s(12)) And &HFF&) Xor (Td4(s(9)) And &HFF00&) Xor (Td4(s(6)) And &HFF0000) Xor (Td4(s(3)) And &HFF000000) Xor rkey(k + 3)

    t4 = (Td4(s(16)) And &HFF&) Xor (Td4(s(13)) And &HFF00&) Xor (Td4(s(10)) And &HFF0000) Xor (Td4(s(7)) And &HFF000000) Xor rkey(k + 4)

    CopyMemory plaintext(p + 0), t0, 4&

    CopyMemory plaintext(p + 4), t1, 4&

    CopyMemory plaintext(p + 8), t2, 4&

    CopyMemory plaintext(p + 12), t3, 4&

    CopyMemory plaintext(p + 16), t4, 4&

End Sub

#End If

Public Sub Block192Encrypt(plaintext() As Byte, ciphertext() As Byte, p As Long, q As Long)

    Dim i     As Long

    Dim k     As Long

    Dim t0    As Long

    Dim t1    As Long

    Dim t2    As Long

    Dim t3    As Long

    Dim t4    As Long

    Dim t5    As Long

    Dim s(23) As Byte

    CopyMemory t0, plaintext(p + 0), 4&

    CopyMemory t1, plaintext(p + 4), 4&

    CopyMemory t2, plaintext(p + 8), 4&

    CopyMemory t3, plaintext(p + 12), 4&

    CopyMemory t4, plaintext(p + 16), 4&

    CopyMemory t5, plaintext(p + 20), 4&

    t0 = t0 Xor fkey(0)

    t1 = t1 Xor fkey(1)
```

```
    t2 = t2 Xor fkey(2)

    t3 = t3 Xor fkey(3)

    t4 = t4 Xor fkey(4)

    t5 = t5 Xor fkey(5)

    k = 6

    For i = 1 To Nr - 1 'Nr is number of rounds

        CopyMemory s(0), t0, 4&

        CopyMemory s(4), t1, 4&

        CopyMemory s(8), t2, 4&

        CopyMemory s(12), t3, 4&

        CopyMemory s(16), t4, 4&

        CopyMemory s(20), t5, 4&

        t0 = Te0(s(0)) Xor Te1(s(5)) Xor Te2(s(10)) Xor Te3(s(15)) Xor fkey(k + 0)

        t1 = Te0(s(4)) Xor Te1(s(9)) Xor Te2(s(14)) Xor Te3(s(19)) Xor fkey(k + 1)

        t2 = Te0(s(8)) Xor Te1(s(13)) Xor Te2(s(18)) Xor Te3(s(23)) Xor fkey(k + 2)

        t3 = Te0(s(12)) Xor Te1(s(17)) Xor Te2(s(22)) Xor Te3(s(3)) Xor fkey(k + 3)

        t4 = Te0(s(16)) Xor Te1(s(21)) Xor Te2(s(2)) Xor Te3(s(7)) Xor fkey(k + 4)

        t5 = Te0(s(20)) Xor Te1(s(1)) Xor Te2(s(6)) Xor Te3(s(11)) Xor fkey(k + 5)

        k = k + 6

    Next i

    CopyMemory s(0), t0, 4&

    CopyMemory s(4), t1, 4&

    CopyMemory s(8), t2, 4&

    CopyMemory s(12), t3, 4&

    CopyMemory s(16), t4, 4&

    CopyMemory s(20), t5, 4&

    t0 = (Te4(s(0)) And &HFF&) Xor (Te4(s(5)) And &HFF00&) Xor (Te4(s(10)) And &HFF0000) Xor
(Te4(s(15)) And &HFF000000) Xor fkey(k + 0)

    t1 = (Te4(s(4)) And &HFF&) Xor (Te4(s(9)) And &HFF00&) Xor (Te4(s(14)) And &HFF0000) Xor
(Te4(s(19)) And &HFF000000) Xor fkey(k + 1)
```

```
    t2 = (Te4(s(8)) And &HFF&) Xor (Te4(s(13)) And &HFF00&) Xor (Te4(s(18)) And &HFF0000) Xor (Te4(s(23)) And &HFF000000) Xor fkey(k + 2)

    t3 = (Te4(s(12)) And &HFF&) Xor (Te4(s(17)) And &HFF00&) Xor (Te4(s(22)) And &HFF0000) Xor (Te4(s(3)) And &HFF000000) Xor fkey(k + 3)

    t4 = (Te4(s(16)) And &HFF&) Xor (Te4(s(21)) And &HFF00&) Xor (Te4(s(2)) And &HFF0000) Xor (Te4(s(7)) And &HFF000000) Xor fkey(k + 4)

    t5 = (Te4(s(20)) And &HFF&) Xor (Te4(s(1)) And &HFF00&) Xor (Te4(s(6)) And &HFF0000) Xor (Te4(s(11)) And &HFF000000) Xor fkey(k + 5)

    CopyMemory ciphertext(q + 0), t0, 4&

    CopyMemory ciphertext(q + 4), t1, 4&

    CopyMemory ciphertext(q + 8), t2, 4&

    CopyMemory ciphertext(q + 12), t3, 4&

    CopyMemory ciphertext(q + 16), t4, 4&

    CopyMemory ciphertext(q + 20), t5, 4&

End Sub

Public Sub Block192Decrypt(plaintext() As Byte, ciphertext() As Byte, p As Long, q As Long)

    Dim i     As Long

    Dim k     As Long

    Dim t0    As Long

    Dim t1    As Long

    Dim t2    As Long

    Dim t3    As Long

    Dim t4    As Long

    Dim t5    As Long

    Dim s(23) As Byte

    CopyMemory t0, ciphertext(q + 0), 4&

    CopyMemory t1, ciphertext(q + 4), 4&

    CopyMemory t2, ciphertext(q + 8), 4&

    CopyMemory t3, ciphertext(q + 12), 4&

    CopyMemory t4, ciphertext(q + 16), 4&

    CopyMemory t5, ciphertext(q + 20), 4&

    t0 = t0 Xor rkey(0)
```

```
    t1 = t1 Xor rkey(1)
    t2 = t2 Xor rkey(2)
    t3 = t3 Xor rkey(3)
    t4 = t4 Xor rkey(4)
    t5 = t5 Xor rkey(5)
    k = 6
    For i = 1 To Nr - 1 'Nr is number of rounds
        CopyMemory s(0), t0, 4&
        CopyMemory s(4), t1, 4&
        CopyMemory s(8), t2, 4&
        CopyMemory s(12), t3, 4&
        CopyMemory s(16), t4, 4&
        CopyMemory s(20), t5, 4&
        t0 = Td0(s(0)) Xor Td1(s(21)) Xor Td2(s(18)) Xor Td3(s(15)) Xor rkey(k + 0)
        t1 = Td0(s(4)) Xor Td1(s(1)) Xor Td2(s(22)) Xor Td3(s(19)) Xor rkey(k + 1)
        t2 = Td0(s(8)) Xor Td1(s(5)) Xor Td2(s(2)) Xor Td3(s(23)) Xor rkey(k + 2)
        t3 = Td0(s(12)) Xor Td1(s(9)) Xor Td2(s(6)) Xor Td3(s(3)) Xor rkey(k + 3)
        t4 = Td0(s(16)) Xor Td1(s(13)) Xor Td2(s(10)) Xor Td3(s(7)) Xor rkey(k + 4)
        t5 = Td0(s(20)) Xor Td1(s(17)) Xor Td2(s(14)) Xor Td3(s(11)) Xor rkey(k + 5)
        k = k + 6
    Next i
    CopyMemory s(0), t0, 4&
    CopyMemory s(4), t1, 4&
    CopyMemory s(8), t2, 4&
    CopyMemory s(12), t3, 4&
    CopyMemory s(16), t4, 4&
    CopyMemory s(20), t5, 4&
    t0 = (Td4(s(0)) And &HFF&) Xor (Td4(s(21)) And &HFF00&) Xor (Td4(s(18)) And &HFF0000) Xor
(Td4(s(15)) And &HFF000000) Xor rkey(k + 0)
    t1 = (Td4(s(4)) And &HFF&) Xor (Td4(s(1)) And &HFF00&) Xor (Td4(s(22)) And &HFF0000) Xor
(Td4(s(19)) And &HFF000000) Xor rkey(k + 1)
```

```
    t2 = (Td4(s(8)) And &HFF&) Xor (Td4(s(5)) And &HFF00&) Xor (Td4(s(2)) And &HFF0000) Xor (Td4(s(23)) And &HFF000000) Xor rkey(k + 2)

    t3 = (Td4(s(12)) And &HFF&) Xor (Td4(s(9)) And &HFF00&) Xor (Td4(s(6)) And &HFF0000) Xor (Td4(s(3)) And &HFF000000) Xor rkey(k + 3)

    t4 = (Td4(s(16)) And &HFF&) Xor (Td4(s(13)) And &HFF00&) Xor (Td4(s(10)) And &HFF0000) Xor (Td4(s(7)) And &HFF000000) Xor rkey(k + 4)

    t5 = (Td4(s(20)) And &HFF&) Xor (Td4(s(17)) And &HFF00&) Xor (Td4(s(14)) And &HFF0000) Xor (Td4(s(11)) And &HFF000000) Xor rkey(k + 5)

    CopyMemory plaintext(p + 0), t0, 4&

    CopyMemory plaintext(p + 4), t1, 4&

    CopyMemory plaintext(p + 8), t2, 4&

    CopyMemory plaintext(p + 12), t3, 4&

    CopyMemory plaintext(p + 16), t4, 4&

    CopyMemory plaintext(p + 20), t5, 4&

End Sub

#If SUPPORT_LEVEL = 2 Then

Public Sub Block224Encrypt(plaintext() As Byte, ciphertext() As Byte, p As Long, q As Long)

    Dim i     As Long

    Dim k     As Long

    Dim t0    As Long

    Dim t1    As Long

    Dim t2    As Long

    Dim t3    As Long

    Dim t4    As Long

    Dim t5    As Long

    Dim t6    As Long

    Dim s(27) As Byte

    CopyMemory t0, plaintext(p + 0), 4&

    CopyMemory t1, plaintext(p + 4), 4&

    CopyMemory t2, plaintext(p + 8), 4&

    CopyMemory t3, plaintext(p + 12), 4&

    CopyMemory t4, plaintext(p + 16), 4&
```

```
CopyMemory t5, plaintext(p + 20), 4&
CopyMemory t6, plaintext(p + 24), 4&
t0 = t0 Xor fkey(0)
t1 = t1 Xor fkey(1)
t2 = t2 Xor fkey(2)
t3 = t3 Xor fkey(3)
t4 = t4 Xor fkey(4)
t5 = t5 Xor fkey(5)
t6 = t6 Xor fkey(6)
k = 7
For i = 1 To Nr - 1 'Nr is number of rounds
    CopyMemory s(0), t0, 4&
    CopyMemory s(4), t1, 4&
    CopyMemory s(8), t2, 4&
    CopyMemory s(12), t3, 4&
    CopyMemory s(16), t4, 4&
    CopyMemory s(20), t5, 4&
    CopyMemory s(24), t6, 4&
    t0 = Te0(s(0)) Xor Te1(s(5)) Xor Te2(s(10)) Xor Te3(s(19)) Xor fkey(k + 0)
    t1 = Te0(s(4)) Xor Te1(s(9)) Xor Te2(s(14)) Xor Te3(s(23)) Xor fkey(k + 1)
    t2 = Te0(s(8)) Xor Te1(s(13)) Xor Te2(s(18)) Xor Te3(s(27)) Xor fkey(k + 2)
    t3 = Te0(s(12)) Xor Te1(s(17)) Xor Te2(s(22)) Xor Te3(s(3)) Xor fkey(k + 3)
    t4 = Te0(s(16)) Xor Te1(s(21)) Xor Te2(s(26)) Xor Te3(s(7)) Xor fkey(k + 4)
    t5 = Te0(s(20)) Xor Te1(s(25)) Xor Te2(s(2)) Xor Te3(s(11)) Xor fkey(k + 5)
    t6 = Te0(s(24)) Xor Te1(s(1)) Xor Te2(s(6)) Xor Te3(s(15)) Xor fkey(k + 6)
    k = k + 7
Next i
'Final round
CopyMemory s(0), t0, 4&
CopyMemory s(4), t1, 4&
```

```
    CopyMemory s(8), t2, 4&

    CopyMemory s(12), t3, 4&

    CopyMemory s(16), t4, 4&

    CopyMemory s(20), t5, 4&

    CopyMemory s(24), t6, 4&

    t0 = (Te4(s(0)) And &HFF&) Xor (Te4(s(5)) And &HFF00&) Xor (Te4(s(10)) And &HFF0000) Xor (Te4(s(19)) And &HFF000000) Xor fkey(k + 0)

    t1 = (Te4(s(4)) And &HFF&) Xor (Te4(s(9)) And &HFF00&) Xor (Te4(s(14)) And &HFF0000) Xor (Te4(s(23)) And &HFF000000) Xor fkey(k + 1)

    t2 = (Te4(s(8)) And &HFF&) Xor (Te4(s(13)) And &HFF00&) Xor (Te4(s(18)) And &HFF0000) Xor (Te4(s(27)) And &HFF000000) Xor fkey(k + 2)

    t3 = (Te4(s(12)) And &HFF&) Xor (Te4(s(17)) And &HFF00&) Xor (Te4(s(22)) And &HFF0000) Xor (Te4(s(3)) And &HFF000000) Xor fkey(k + 3)

    t4 = (Te4(s(16)) And &HFF&) Xor (Te4(s(21)) And &HFF00&) Xor (Te4(s(26)) And &HFF0000) Xor (Te4(s(7)) And &HFF000000) Xor fkey(k + 4)

    t5 = (Te4(s(20)) And &HFF&) Xor (Te4(s(25)) And &HFF00&) Xor (Te4(s(2)) And &HFF0000) Xor (Te4(s(11)) And &HFF000000) Xor fkey(k + 5)

    t6 = (Te4(s(24)) And &HFF&) Xor (Te4(s(1)) And &HFF00&) Xor (Te4(s(6)) And &HFF0000) Xor (Te4(s(15)) And &HFF000000) Xor fkey(k + 6)

    CopyMemory ciphertext(q + 0), t0, 4&

    CopyMemory ciphertext(q + 4), t1, 4&

    CopyMemory ciphertext(q + 8), t2, 4&

    CopyMemory ciphertext(q + 12), t3, 4&

    CopyMemory ciphertext(q + 16), t4, 4&

    CopyMemory ciphertext(q + 20), t5, 4&

    CopyMemory ciphertext(q + 24), t6, 4&

End Sub

Public Sub Block224Decrypt(plaintext() As Byte, ciphertext() As Byte, p As Long, q As Long)

    Dim i     As Long

    Dim k     As Long

    Dim t0    As Long

    Dim t1    As Long

    Dim t2    As Long
```

```
Dim t3    As Long
Dim t4    As Long
Dim t5    As Long
Dim t6    As Long
Dim s(27) As Byte
CopyMemory t0, ciphertext(q + 0), 4&
CopyMemory t1, ciphertext(q + 4), 4&
CopyMemory t2, ciphertext(q + 8), 4&
CopyMemory t3, ciphertext(q + 12), 4&
CopyMemory t4, ciphertext(q + 16), 4&
CopyMemory t5, ciphertext(q + 20), 4&
CopyMemory t6, ciphertext(q + 24), 4&
t0 = t0 Xor rkey(0)
t1 = t1 Xor rkey(1)
t2 = t2 Xor rkey(2)
t3 = t3 Xor rkey(3)
t4 = t4 Xor rkey(4)
t5 = t5 Xor rkey(5)
t6 = t6 Xor rkey(6)
k = 7
For i = 1 To Nr - 1 'Nr is number of rounds
   CopyMemory s(0), t0, 4&
   CopyMemory s(4), t1, 4&
   CopyMemory s(8), t2, 4&
   CopyMemory s(12), t3, 4&
   CopyMemory s(16), t4, 4&
   CopyMemory s(20), t5, 4&
   CopyMemory s(24), t6, 4&
   t0 = Td0(s(0)) Xor Td1(s(25)) Xor Td2(s(22)) Xor Td3(s(15)) Xor rkey(k + 0)
   t1 = Td0(s(4)) Xor Td1(s(1)) Xor Td2(s(26)) Xor Td3(s(19)) Xor rkey(k + 1)
```

```
    t2 = Td0(s(8)) Xor Td1(s(5)) Xor Td2(s(2)) Xor Td3(s(23)) Xor rkey(k + 2)

    t3 = Td0(s(12)) Xor Td1(s(9)) Xor Td2(s(6)) Xor Td3(s(27)) Xor rkey(k + 3)

    t4 = Td0(s(16)) Xor Td1(s(13)) Xor Td2(s(10)) Xor Td3(s(3)) Xor rkey(k + 4)

    t5 = Td0(s(20)) Xor Td1(s(17)) Xor Td2(s(14)) Xor Td3(s(7)) Xor rkey(k + 5)

    t6 = Td0(s(24)) Xor Td1(s(21)) Xor Td2(s(18)) Xor Td3(s(11)) Xor rkey(k + 6)

    k = k + 7

  Next i

  'Final round

  CopyMemory s(0), t0, 4&

  CopyMemory s(4), t1, 4&

  CopyMemory s(8), t2, 4&

  CopyMemory s(12), t3, 4&

  CopyMemory s(16), t4, 4&

  CopyMemory s(20), t5, 4&

  CopyMemory s(24), t6, 4&

  t0 = (Td4(s(0)) And &HFF&) Xor (Td4(s(25)) And &HFF00&) Xor (Td4(s(22)) And &HFF0000) Xor (Td4(s(15)) And &HFF000000) Xor rkey(k + 0)

  t1 = (Td4(s(4)) And &HFF&) Xor (Td4(s(1)) And &HFF00&) Xor (Td4(s(26)) And &HFF0000) Xor (Td4(s(19)) And &HFF000000) Xor rkey(k + 1)

  t2 = (Td4(s(8)) And &HFF&) Xor (Td4(s(5)) And &HFF00&) Xor (Td4(s(2)) And &HFF0000) Xor (Td4(s(23)) And &HFF000000) Xor rkey(k + 2)

  t3 = (Td4(s(12)) And &HFF&) Xor (Td4(s(9)) And &HFF00&) Xor (Td4(s(6)) And &HFF0000) Xor (Td4(s(27)) And &HFF000000) Xor rkey(k + 3)

  t4 = (Td4(s(16)) And &HFF&) Xor (Td4(s(13)) And &HFF00&) Xor (Td4(s(10)) And &HFF0000) Xor (Td4(s(3)) And &HFF000000) Xor rkey(k + 4)

  t5 = (Td4(s(20)) And &HFF&) Xor (Td4(s(17)) And &HFF00&) Xor (Td4(s(14)) And &HFF0000) Xor (Td4(s(7)) And &HFF000000) Xor rkey(k + 5)

  t6 = (Td4(s(24)) And &HFF&) Xor (Td4(s(21)) And &HFF00&) Xor (Td4(s(18)) And &HFF0000) Xor (Td4(s(11)) And &HFF000000) Xor rkey(k + 6)

  CopyMemory plaintext(p + 0), t0, 4&

  CopyMemory plaintext(p + 4), t1, 4&

  CopyMemory plaintext(p + 8), t2, 4&

  CopyMemory plaintext(p + 12), t3, 4&
```

```
    CopyMemory plaintext(p + 16), t4, 4&
    CopyMemory plaintext(p + 20), t5, 4&
    CopyMemory plaintext(p + 24), t6, 4&
End Sub
#End If
Public Sub Block256Encrypt(plaintext() As Byte, ciphertext() As Byte, p As Long, q As Long)
    Dim i     As Long
    Dim k     As Long
    Dim t0    As Long
    Dim t1    As Long
    Dim t2    As Long
    Dim t3    As Long
    Dim t4    As Long
    Dim t5    As Long
    Dim t6    As Long
    Dim t7    As Long
    Dim s(31) As Byte
    CopyMemory t0, plaintext(p + 0), 4&
    CopyMemory t1, plaintext(p + 4), 4&
    CopyMemory t2, plaintext(p + 8), 4&
    CopyMemory t3, plaintext(p + 12), 4&
    CopyMemory t4, plaintext(p + 16), 4&
    CopyMemory t5, plaintext(p + 20), 4&
    CopyMemory t6, plaintext(p + 24), 4&
    CopyMemory t7, plaintext(p + 28), 4&
    t0 = t0 Xor fkey(0)
    t1 = t1 Xor fkey(1)
    t2 = t2 Xor fkey(2)
    t3 = t3 Xor fkey(3)
    t4 = t4 Xor fkey(4)
```

```
t5 = t5 Xor fkey(5)
t6 = t6 Xor fkey(6)
t7 = t7 Xor fkey(7)
k = 8
For i = 1 To Nr - 1 'Nr is number of rounds
    CopyMemory s(0), t0, 4&
    CopyMemory s(4), t1, 4&
    CopyMemory s(8), t2, 4&
    CopyMemory s(12), t3, 4&
    CopyMemory s(16), t4, 4&
    CopyMemory s(20), t5, 4&
    CopyMemory s(24), t6, 4&
    CopyMemory s(28), t7, 4&
    t0 = Te0(s(0)) Xor Te1(s(5)) Xor Te2(s(14)) Xor Te3(s(19)) Xor fkey(k + 0)
    t1 = Te0(s(4)) Xor Te1(s(9)) Xor Te2(s(18)) Xor Te3(s(23)) Xor fkey(k + 1)
    t2 = Te0(s(8)) Xor Te1(s(13)) Xor Te2(s(22)) Xor Te3(s(27)) Xor fkey(k + 2)
    t3 = Te0(s(12)) Xor Te1(s(17)) Xor Te2(s(26)) Xor Te3(s(31)) Xor fkey(k + 3)
    t4 = Te0(s(16)) Xor Te1(s(21)) Xor Te2(s(30)) Xor Te3(s(3)) Xor fkey(k + 4)
    t5 = Te0(s(20)) Xor Te1(s(25)) Xor Te2(s(2)) Xor Te3(s(7)) Xor fkey(k + 5)
    t6 = Te0(s(24)) Xor Te1(s(29)) Xor Te2(s(6)) Xor Te3(s(11)) Xor fkey(k + 6)
    t7 = Te0(s(28)) Xor Te1(s(1)) Xor Te2(s(10)) Xor Te3(s(15)) Xor fkey(k + 7)
    k = k + 8
Next i
'Final round
CopyMemory s(0), t0, 4&
CopyMemory s(4), t1, 4&
CopyMemory s(8), t2, 4&
CopyMemory s(12), t3, 4&
CopyMemory s(16), t4, 4&
CopyMemory s(20), t5, 4&
```

```
    CopyMemory s(24), t6, 4&

    CopyMemory s(28), t7, 4&

    t0 = (Te4(s(0)) And &HFF&) Xor (Te4(s(5)) And &HFF00&) Xor (Te4(s(14)) And &HFF0000) Xor (Te4(s(19)) And &HFF000000) Xor fkey(k + 0)

    t1 = (Te4(s(4)) And &HFF&) Xor (Te4(s(9)) And &HFF00&) Xor (Te4(s(18)) And &HFF0000) Xor (Te4(s(23)) And &HFF000000) Xor fkey(k + 1)

    t2 = (Te4(s(8)) And &HFF&) Xor (Te4(s(13)) And &HFF00&) Xor (Te4(s(22)) And &HFF0000) Xor (Te4(s(27)) And &HFF000000) Xor fkey(k + 2)

    t3 = (Te4(s(12)) And &HFF&) Xor (Te4(s(17)) And &HFF00&) Xor (Te4(s(26)) And &HFF0000) Xor (Te4(s(31)) And &HFF000000) Xor fkey(k + 3)

    t4 = (Te4(s(16)) And &HFF&) Xor (Te4(s(21)) And &HFF00&) Xor (Te4(s(30)) And &HFF0000) Xor (Te4(s(3)) And &HFF000000) Xor fkey(k + 4)

    t5 = (Te4(s(20)) And &HFF&) Xor (Te4(s(25)) And &HFF00&) Xor (Te4(s(2)) And &HFF0000) Xor (Te4(s(7)) And &HFF000000) Xor fkey(k + 5)

    t6 = (Te4(s(24)) And &HFF&) Xor (Te4(s(29)) And &HFF00&) Xor (Te4(s(6)) And &HFF0000) Xor (Te4(s(11)) And &HFF000000) Xor fkey(k + 6)

    t7 = (Te4(s(28)) And &HFF&) Xor (Te4(s(1)) And &HFF00&) Xor (Te4(s(10)) And &HFF0000) Xor (Te4(s(15)) And &HFF000000) Xor fkey(k + 7)

    CopyMemory ciphertext(q + 0), t0, 4&

    CopyMemory ciphertext(q + 4), t1, 4&

    CopyMemory ciphertext(q + 8), t2, 4&

    CopyMemory ciphertext(q + 12), t3, 4&

    CopyMemory ciphertext(q + 16), t4, 4&

    CopyMemory ciphertext(q + 20), t5, 4&

    CopyMemory ciphertext(q + 24), t6, 4&

    CopyMemory ciphertext(q + 28), t7, 4&

End Sub

Public Sub Block256Decrypt(plaintext() As Byte, ciphertext() As Byte, p As Long, q As Long)

    Dim i     As Long

    Dim k     As Long

    Dim t0    As Long

    Dim t1    As Long

    Dim t2    As Long

    Dim t3    As Long
```

```
Dim t4    As Long
Dim t5    As Long
Dim t6    As Long
Dim t7    As Long
Dim s(31) As Byte
CopyMemory t0, ciphertext(q + 0), 4&
CopyMemory t1, ciphertext(q + 4), 4&
CopyMemory t2, ciphertext(q + 8), 4&
CopyMemory t3, ciphertext(q + 12), 4&
CopyMemory t4, ciphertext(q + 16), 4&
CopyMemory t5, ciphertext(q + 20), 4&
CopyMemory t6, ciphertext(q + 24), 4&
CopyMemory t7, ciphertext(q + 28), 4&
t0 = t0 Xor rkey(0)
t1 = t1 Xor rkey(1)
t2 = t2 Xor rkey(2)
t3 = t3 Xor rkey(3)
t4 = t4 Xor rkey(4)
t5 = t5 Xor rkey(5)
t6 = t6 Xor rkey(6)
t7 = t7 Xor rkey(7)
k = 8
For i = 1 To Nr - 1 'Nr is number of rounds
   CopyMemory s(0), t0, 4&
   CopyMemory s(4), t1, 4&
   CopyMemory s(8), t2, 4&
   CopyMemory s(12), t3, 4&
   CopyMemory s(16), t4, 4&
   CopyMemory s(20), t5, 4&
   CopyMemory s(24), t6, 4&
```

```
    CopyMemory s(28), t7, 4&
    t0 = Td0(s(0)) Xor Td1(s(29)) Xor Td2(s(22)) Xor Td3(s(19)) Xor rkey(k + 0)
    t1 = Td0(s(4)) Xor Td1(s(1)) Xor Td2(s(26)) Xor Td3(s(23)) Xor rkey(k + 1)
    t2 = Td0(s(8)) Xor Td1(s(5)) Xor Td2(s(30)) Xor Td3(s(27)) Xor rkey(k + 2)
    t3 = Td0(s(12)) Xor Td1(s(9)) Xor Td2(s(2)) Xor Td3(s(31)) Xor rkey(k + 3)
    t4 = Td0(s(16)) Xor Td1(s(13)) Xor Td2(s(6)) Xor Td3(s(3)) Xor rkey(k + 4)
    t5 = Td0(s(20)) Xor Td1(s(17)) Xor Td2(s(10)) Xor Td3(s(7)) Xor rkey(k + 5)
    t6 = Td0(s(24)) Xor Td1(s(21)) Xor Td2(s(14)) Xor Td3(s(11)) Xor rkey(k + 6)
    t7 = Td0(s(28)) Xor Td1(s(25)) Xor Td2(s(18)) Xor Td3(s(15)) Xor rkey(k + 7)
    k = k + 8
  Next i
  'Final round
  CopyMemory s(0), t0, 4&
  CopyMemory s(4), t1, 4&
  CopyMemory s(8), t2, 4&
  CopyMemory s(12), t3, 4&
  CopyMemory s(16), t4, 4&
  CopyMemory s(20), t5, 4&
  CopyMemory s(24), t6, 4&
  CopyMemory s(28), t7, 4&
  t0 = (Td4(s(0)) And &HFF&) Xor (Td4(s(29)) And &HFF00&) Xor (Td4(s(22)) And &HFF0000) Xor (Td4(s(19)) And &HFF000000) Xor rkey(k + 0)
  t1 = (Td4(s(4)) And &HFF&) Xor (Td4(s(1)) And &HFF00&) Xor (Td4(s(26)) And &HFF0000) Xor (Td4(s(23)) And &HFF000000) Xor rkey(k + 1)
  t2 = (Td4(s(8)) And &HFF&) Xor (Td4(s(5)) And &HFF00&) Xor (Td4(s(30)) And &HFF0000) Xor (Td4(s(27)) And &HFF000000) Xor rkey(k + 2)
  t3 = (Td4(s(12)) And &HFF&) Xor (Td4(s(9)) And &HFF00&) Xor (Td4(s(2)) And &HFF0000) Xor (Td4(s(31)) And &HFF000000) Xor rkey(k + 3)
  t4 = (Td4(s(16)) And &HFF&) Xor (Td4(s(13)) And &HFF00&) Xor (Td4(s(6)) And &HFF0000) Xor (Td4(s(3)) And &HFF000000) Xor rkey(k + 4)
  t5 = (Td4(s(20)) And &HFF&) Xor (Td4(s(17)) And &HFF00&) Xor (Td4(s(10)) And &HFF0000) Xor (Td4(s(7)) And &HFF000000) Xor rkey(k + 5)
```

```
    t6 = (Td4(s(24)) And &HFF&) Xor (Td4(s(21)) And &HFF00&) Xor (Td4(s(14)) And &HFF0000) Xor (Td4(s(11)) And &HFF000000) Xor rkey(k + 6)

    t7 = (Td4(s(28)) And &HFF&) Xor (Td4(s(25)) And &HFF00&) Xor (Td4(s(18)) And &HFF0000) Xor (Td4(s(15)) And &HFF000000) Xor rkey(k + 7)

    CopyMemory plaintext(p + 0), t0, 4&

    CopyMemory plaintext(p + 4), t1, 4&

    CopyMemory plaintext(p + 8), t2, 4&

    CopyMemory plaintext(p + 12), t3, 4&

    CopyMemory plaintext(p + 16), t4, 4&

    CopyMemory plaintext(p + 20), t5, 4&

    CopyMemory plaintext(p + 24), t6, 4&

    CopyMemory plaintext(p + 28), t7, 4&

End Sub

#End If

#If SUPPORT_LEVEL Then

Public Function ArrayEncrypt(plaintext() As Byte, ciphertext() As Byte, appendsize As Long, BlockBits As Long) As Long

#Else

Public Function ArrayEncrypt(plaintext() As Byte, ciphertext() As Byte, appendsize As Long) As Long

#End If

    Dim i          As Long

    Dim m          As Long

    Dim n          As Long

#If SUPPORT_LEVEL = 0 Then

    Const BlockSize As Long = 16 'bytes

#Else

    Dim BlockSize   As Long

    Select Case BlockBits

    Case 128:  BlockSize = 16

    Case 192:  BlockSize = 24

    Case 256:  BlockSize = 32
```

```
#If SUPPORT_LEVEL = 2 Then
    Case 160:  BlockSize = 20
    Case 224:  BlockSize = 28
#End If
    Case Else: Err.Raise 1, , "cRijndael.ArrayEncrypt - Illegal BlockBits value"
    End Select
#End If
    If LBound(plaintext) <> 0 Then Err.Raise 1, , "cRijndael.ArrayEncrypt - plaintext must be zero based array"
    n = UBound(plaintext) + 1
    If appendsize = 0 Then
#If SUPPORT_LEVEL Then
        m = ((n + BlockSize - 1) \ BlockSize) * BlockSize
#Else
        m = (n + BlockSize - 1) And &HFFFFFFF0  'BlockSize=16 specific
#End If
        ReDim ciphertext(m - 1)
    Else
#If SUPPORT_LEVEL Then
        m = ((n + BlockSize) \ BlockSize) * BlockSize
#Else
        m = (n + BlockSize) And &HFFFFFFF0      'BlockSize=16 specific
#End If
        ReDim ciphertext(m - 1)
        ciphertext(m - 1) = n Mod BlockSize
    End If
#If SUPPORT_LEVEL Then
    Select Case BlockBits
    Case 128
#End If
    For i = 0 To n - BlockSize Step BlockSize
```

```
        BlockEncrypt plaintext, ciphertext, i, i
    Next i
#If SUPPORT_LEVEL Then
    Case 192
    For i = 0 To n - BlockSize Step BlockSize
        Block192Encrypt plaintext, ciphertext, i, i
    Next i
    Case 256
    For i = 0 To n - BlockSize Step BlockSize
        Block256Encrypt plaintext, ciphertext, i, i
    Next i
#If SUPPORT_LEVEL = 2 Then
    Case 160
    For i = 0 To n - BlockSize Step BlockSize
        Block160Encrypt plaintext, ciphertext, i, i
    Next i
    Case 224
    For i = 0 To n - BlockSize Step BlockSize
        Block224Encrypt plaintext, ciphertext, i, i
    Next i
#End If
    End Select
#End If
    If (n Mod BlockSize) <> 0 Then CopyMemory ciphertext(i), plaintext(i), n Mod BlockSize
#If SUPPORT_LEVEL Then
    Select Case BlockBits
    Case 128
#End If
    If i <> m Then BlockEncrypt ciphertext, ciphertext, i, i
#If SUPPORT_LEVEL Then
```

```
    Case 192
    If i <> m Then Block192Encrypt ciphertext, ciphertext, i, i
    Case 256
    If i <> m Then Block256Encrypt ciphertext, ciphertext, i, i
#If SUPPORT_LEVEL = 2 Then
    Case 160
    If i <> m Then Block160Encrypt ciphertext, ciphertext, i, i
    Case 224
    If i <> m Then Block224Encrypt ciphertext, ciphertext, i, i
#End If
    End Select
#End If
End Function
#If SUPPORT_LEVEL Then
Public Function ArrayDecrypt(plaintext() As Byte, ciphertext() As Byte, appendsize As Long, BlockBits As Long) As Long
#Else
Public Function ArrayDecrypt(plaintext() As Byte, ciphertext() As Byte, appendsize As Long) As Long
#End If
    Dim i          As Long
    Dim m          As Long
    Dim n          As Long
#If SUPPORT_LEVEL = 0 Then
    Const BlockSize As Long = 16 'bytes
#Else
    Dim BlockSize   As Long
    Select Case BlockBits
    Case 128:  BlockSize = 16
    Case 192:  BlockSize = 24
    Case 256:  BlockSize = 32
```

```
#If SUPPORT_LEVEL = 2 Then
    Case 160:  BlockSize = 20
    Case 224:  BlockSize = 28
#End If
    Case Else: Err.Raise 1, , "cRijndael.ArrayDecrypt - Illegal BlockBits value"
    End Select
#End If
    If LBound(ciphertext) <> 0 Then Err.Raise 1, , " ArrayDecrypt - ciphertext must be zero based array"
    n = UBound(ciphertext) + 1
    If ((n Mod BlockSize) = 0) Then
        ReDim plaintext(n - 1)
#If SUPPORT_LEVEL Then
        Select Case BlockBits
        Case 128
#End If
        For i = 0 To n - BlockSize Step BlockSize
            BlockDecrypt plaintext, ciphertext, i, i
        Next i
#If SUPPORT_LEVEL Then
        Case 192
        For i = 0 To n - BlockSize Step BlockSize
            Block192Decrypt plaintext, ciphertext, i, i
        Next i
        Case 256
        For i = 0 To n - BlockSize Step BlockSize
            Block256Decrypt plaintext, ciphertext, i, i
        Next i
#If SUPPORT_LEVEL = 2 Then
        Case 160
        For i = 0 To n - BlockSize Step BlockSize
```

```
            Block160Decrypt plaintext, ciphertext, i, i
        Next i
        Case 224
        For i = 0 To n - BlockSize Step BlockSize
            Block224Decrypt plaintext, ciphertext, i, i
        Next i
#End If
        End Select
#End If
        If appendsize Then
            If plaintext(n - 1) < BlockSize Then
                n = n - BlockSize + plaintext(n - 1)
                If n > 0 Then ReDim Preserve plaintext(n - 1)
            Else
                MsgBox "warning - incorrect length field"
                ArrayDecrypt = 1
            End If
        End If
    Else
        MsgBox "ciphertext size not a multiple of block size"
        ArrayDecrypt = -1
    End If
End Function
#If SUPPORT_LEVEL Then
Public Function FileEncrypt(PlaintextFileName As String, CiphertextFileName As String, BlockBits As Long) As Long
#Else
Public Function FileEncrypt(PlaintextFileName As String, CiphertextFileName As String) As Long
#End If
    Dim FileNum     As Integer
```

```
    Dim FileNum2    As Integer
    Dim i           As Long
    Dim m           As Long 'ciphertext file size
    Dim n           As Long 'plaintext file size
    Dim data()      As Byte
#If SUPPORT_LEVEL = 0 Then
    Const BlockSize As Long = 16 'bytes
    Const MaxBlocks As Long = MaxFileChunkSize \ BlockSize
#Else
    Dim BlockSize   As Long
    Dim MaxBlocks   As Long
    Select Case BlockBits
    Case 128:  BlockSize = 16
    Case 192:  BlockSize = 24
    Case 256:  BlockSize = 32
#If SUPPORT_LEVEL = 2 Then
    Case 160:  BlockSize = 20
    Case 224:  BlockSize = 28
#End If
    Case Else: Err.Raise 1, , "cRijndael.FileEncrypt - Illegal BlockBits value"
    End Select
    MaxBlocks = MaxFileChunkSize \ BlockSize
#End If
    n = FileLen(PlaintextFileName)
#If SUPPORT_LEVEL Then
    m = ((n + BlockSize) \ BlockSize) * BlockSize
#Else
    m = (n + BlockSize) And (-BlockSize) 'BlockSize=16 specific
#End If
    FileNum = FreeFile
```

```
    Open PlaintextFileName For Binary Access Read As FileNum
    FileNum2 = FreeFile
    Open CiphertextFileName For Binary Access Write As FileNum2
    'For large files, encrypt in pieces no larger than MaxFileChunkSize
    If m > MaxBlocks * BlockSize Then
        ReDim data(MaxBlocks * BlockSize - 1)
        Do
            Get #FileNum, , data
#If SUPPORT_LEVEL Then
            Select Case BlockBits
            Case 128
#End If
            For i = 0 To (MaxBlocks - 1) * BlockSize Step BlockSize
                BlockEncrypt data, data, i, i
            Next i
#If SUPPORT_LEVEL Then
            Case 192
            For i = 0 To (MaxBlocks - 1) * BlockSize Step BlockSize
                Block192Encrypt data, data, i, i
            Next i
            Case 256
            For i = 0 To (MaxBlocks - 1) * BlockSize Step BlockSize
                Block256Encrypt data, data, i, i
            Next i
#If SUPPORT_LEVEL = 2 Then
            Case 160
            For i = 0 To (MaxBlocks - 1) * BlockSize Step BlockSize
                Block160Encrypt data, data, i, i
            Next i
            Case 224
```

```
            For i = 0 To (MaxBlocks - 1) * BlockSize Step BlockSize
                Block224Encrypt data, data, i, i
            Next i
#End If
            End Select
#End If
            Put #FileNum2, , data
            m = m - MaxBlocks * BlockSize
        Loop While m > MaxBlocks * BlockSize
    End If
    ReDim data(m - 1)
    Get #FileNum, , data
    data(m - 1) = n Mod BlockSize
#If SUPPORT_LEVEL Then
    Select Case BlockBits
    Case 128
#End If
    For i = 0 To m - BlockSize Step BlockSize
        BlockEncrypt data, data, i, i
    Next i
#If SUPPORT_LEVEL Then
    Case 192
    For i = 0 To m - BlockSize Step BlockSize
        Block192Encrypt data, data, i, i
    Next i
    Case 256
    For i = 0 To m - BlockSize Step BlockSize
        Block256Encrypt data, data, i, i
    Next i
#If SUPPORT_LEVEL = 2 Then
```

```
    Case 160
    For i = 0 To m - BlockSize Step BlockSize
        Block160Encrypt data, data, i, i
    Next i
    Case 224
    For i = 0 To m - BlockSize Step BlockSize
        Block224Encrypt data, data, i, i
    Next i
#End If
    End Select
#End If
    Put FileNum2, , data
    Close FileNum
    Close FileNum2
End Function
#If SUPPORT_LEVEL Then
Public Function FileDecrypt(PlaintextFileName As String, CiphertextFileName As String, BlockBits As Long) As Long
#Else
Public Function FileDecrypt(PlaintextFileName As String, CiphertextFileName As String) As Long
#End If
    Dim FileNum     As Integer
    Dim FileNum2    As Integer
    Dim i           As Long
    Dim m           As Long 'ciphertext file size
    Dim n           As Long 'plaintext file size
    Dim data()      As Byte
#If SUPPORT_LEVEL = 0 Then
    Const BlockSize As Long = 16 'bytes
    Const MaxBlocks As Long = MaxFileChunkSize \ BlockSize
```

```
#Else
    Dim BlockSize   As Long
    Dim MaxBlocks   As Long
    Select Case BlockBits
    Case 128:  BlockSize = 16
    Case 192:  BlockSize = 24
    Case 256:  BlockSize = 32
#If SUPPORT_LEVEL = 2 Then
    Case 160:  BlockSize = 20
    Case 224:  BlockSize = 28
#End If
    Case Else: Err.Raise 1, , "cRijndael.FileDecrypt - Illegal BlockBits value"
    End Select
    MaxBlocks = MaxFileChunkSize \ BlockSize
#End If
    m = FileLen(CiphertextFileName)
#If SUPPORT_LEVEL Then
    If (m = 0) Or ((m Mod BlockSize) <> 0) Then
#Else
    If (m = 0) Or ((m And (BlockSize - 1)) <> 0) Then 'BlockSize=16 specific
#End If
        MsgBox "File Size Error - ciphertext file not a multiple of block size"
        FileDecrypt = 1
    Else
        FileNum = FreeFile
        Open CiphertextFileName For Binary Access Read As FileNum
        FileNum2 = FreeFile
        Open PlaintextFileName For Binary Access Write As FileNum2
        If m > MaxBlocks * BlockSize Then
            ReDim data(MaxBlocks * BlockSize - 1)
```

```
        Do
            Get #FileNum, , data
#If SUPPORT_LEVEL Then
            Select Case BlockBits
            Case 128
#End If
            For i = 0 To (MaxBlocks - 1) * BlockSize Step BlockSize
               BlockDecrypt data, data, i, i
            Next i
#If SUPPORT_LEVEL Then
            Case 192
            For i = 0 To (MaxBlocks - 1) * BlockSize Step BlockSize
               Block192Decrypt data, data, i, i
            Next i
            Case 256
            For i = 0 To (MaxBlocks - 1) * BlockSize Step BlockSize
               Block256Decrypt data, data, i, i
            Next i
#If SUPPORT_LEVEL = 2 Then
            Case 160
            For i = 0 To (MaxBlocks - 1) * BlockSize Step BlockSize
               Block160Decrypt data, data, i, i
            Next i
            Case 224
            For i = 0 To (MaxBlocks - 1) * BlockSize Step BlockSize
               Block224Decrypt data, data, i, i
            Next i
#End If
            End Select
#End If
```

```
            Put #FileNum2, , data
            m = m - MaxBlocks * BlockSize
        Loop While m > MaxBlocks * BlockSize
    End If
    'Decrypt the last piece of the file
    ReDim data(m - 1)
    Get #FileNum, , data
#If SUPPORT_LEVEL Then
    Select Case BlockBits
    Case 128
#End If
    For i = 0 To m - BlockSize Step BlockSize
        BlockDecrypt data, data, i, i
    Next i
#If SUPPORT_LEVEL Then
    Case 192
    For i = 0 To m - BlockSize Step BlockSize
        Block192Decrypt data, data, i, i
    Next i
    Case 256
    For i = 0 To m - BlockSize Step BlockSize
        Block256Decrypt data, data, i, i
    Next i
#If SUPPORT_LEVEL = 2 Then
    Case 160
    For i = 0 To m - BlockSize Step BlockSize
        Block160Decrypt data, data, i, i
    Next i
    Case 224
    For i = 0 To m - BlockSize Step BlockSize
```

```
            Block224Decrypt data, data, i, i
        Next i
#End If
        End Select
#End If
        'Recover length field
        If data(m - 1) < BlockSize Then
            n = m - BlockSize + CLng(data(m - 1))
        Else
#If SUPPORT_LEVEL Then
            MsgBox "warning - incorrect in decrypted file." & vbCrLf & "Wrong key, keysize, or blocksize?"
#Else
            MsgBox "warning - incorrect in decrypted file." & vbCrLf & "Wrong key or keysize?"
#End If
            n = m
        End If
        If n > 0 Then
            ReDim Preserve data(n - 1)
            Put FileNum2, , data
        End If
        Close FileNum
        Close FileNum2
    End If
End Function
Private Sub Class_Initialize()
#If COMPILE_CONSTANTS = 0 Then
    Dim i         As Long
    Dim y         As Byte
    Dim s(7)      As Byte
    Dim ib        As Byte
```

```
Dim ptab(255)  As Byte
Dim ltab(255)  As Byte
ltab(0) = 0
ltab(1) = 0
ltab(3) = 1
ptab(0) = 1
ptab(1) = 3
For i = 2 To 255
If (ptab(i - 1) And &H80) Then
      ptab(i) = ptab(i - 1) Xor ((ptab(i - 1) And 127) * 2) Xor &H1B
   Else
      ptab(i) = ptab(i - 1) Xor (ptab(i - 1) * 2)
   End If
   ltab(ptab(i)) = i
Next i
Te4(0) = &H63636363
Td4(&H63) = 0
For i = 1 To 255
   y = ptab(255 - ltab(i))
   ib = y
   If ib And &H80 Then ib = (ib And 127) * 2 Or 1 Else ib = ib * 2
   y = y Xor ib
   If ib And &H80 Then ib = (ib And 127) * 2 Or 1 Else ib = ib * 2
   y = y Xor ib
   If ib And &H80 Then ib = (ib And 127) * 2 Or 1 Else ib = ib * 2
   y = y Xor ib
   If ib And &H80 Then ib = (ib And 127) * 2 Or 1 Else ib = ib * 2
   y = y Xor ib Xor &H63
   s(0) = y:          s(1) = s(0):          s(2) = s(0):          s(3) = s(0)
   CopyMemory Te4(i), s(0), 4&
```

```
    s(0) = I:           s(1) = s(0):          s(2) = s(0):          s(3) = s(0)
    CopyMemory Td4(y), s(0), 4&
  Next i
  y = 1
  For i = 0 To UBound(rco)
    rco(i) = y
    If (y And &H80) Then 'y = Xtime(y)
      y = ((y And 127) * 2) Xor &H1B
    Else
      y = y * 2
    End If
  Next i
  For i = 0 To 255
    y = Te4(i) And &HFF&
    's(3) = y Xor Xtime(y)
'   s(0) = Xtime(y)
    If (y And &H80) Then
      s(0) = ((y And 127) * 2) Xor &H1B
      s(3) = y Xor s(0)
    Else
      s(0) = y * 2
      s(3) = y Xor s(0)
    End If
    s(2) = y
    s(1) = y
    CopyMemory s(4), s(0), 4&
    CopyMemory Te0(i), s(0), 4&
    CopyMemory Te1(i), s(3), 4&
    CopyMemory Te2(i), s(2), 4&
    CopyMemory Te3(i), s(1), 4&
```

```
    y = Td4(i) And &HFF&
    If y = 0 Then 'x.y= AntiLog(Log(x) + Log(y))
       s(3) = 0
       s(2) = 0
       s(1) = 0
       s(0) = 0
    Else
       s(3) = ptab((CLng(ltab(&HB)) + CLng(ltab(y))) Mod 255)
       s(2) = ptab((CLng(ltab(&HD)) + CLng(ltab(y))) Mod 255)
       s(1) = ptab((CLng(ltab(&H9)) + CLng(ltab(y))) Mod 255)
       s(0) = ptab((CLng(ltab(&HE)) + CLng(ltab(y))) Mod 255)
    End If
    CopyMemory s(4), s(0), 4&
    CopyMemory Td0(i), s(0), 4&
    CopyMemory Td1(i), s(3), 4&
    CopyMemory Td2(i), s(2), 4&
    CopyMemory Td3(i), s(1), 4&
  Next i
End Sub
```


STMIK
ATMA LUHUR
PANGKAL PINANG

Lampiran Kartu Bimbingan

STMIK ATMA LUHUR

KARTU BIMBINGAN

NIM : 1011500176

NAMA : ARI

DOSEN PEMBIMBENG : BAMBANG ADIWINOTO, M.Kom

JUDUL TUGAS AKHIR (TA) : APLIKASI PENGAMANAN SISTEM BASIS DATA DENGAN MENGGUNAKAN TEKNIK KRIPTOGRAFI

NO	TANGGAL	MATERI	PARAF DOSEN
1	08-06-2015	Judul	
2	15-06-2015	Bab 1	
3	22-06-2015	Bab 2	
4	29-06-2015	Metode	
5	06-07-2015	Flowchart	
6	13-07-2015	Analisa	
7	20-07-2015	Kelengkapan	
8	27-07-2015	Bab 4	
9	31-08-2015	Kesimpulan	
10	07-09-2015	Revisi	
11			
12			
13			
14			

Mahasiswa di atas telah melakukan bimbingan dengan jumlah materi yang telah mencukupi untuk di sidangkan

Pangkalpinang, 7-9-2015

Mahasiswa

(ARI)

Dosen Pembimbing

( )