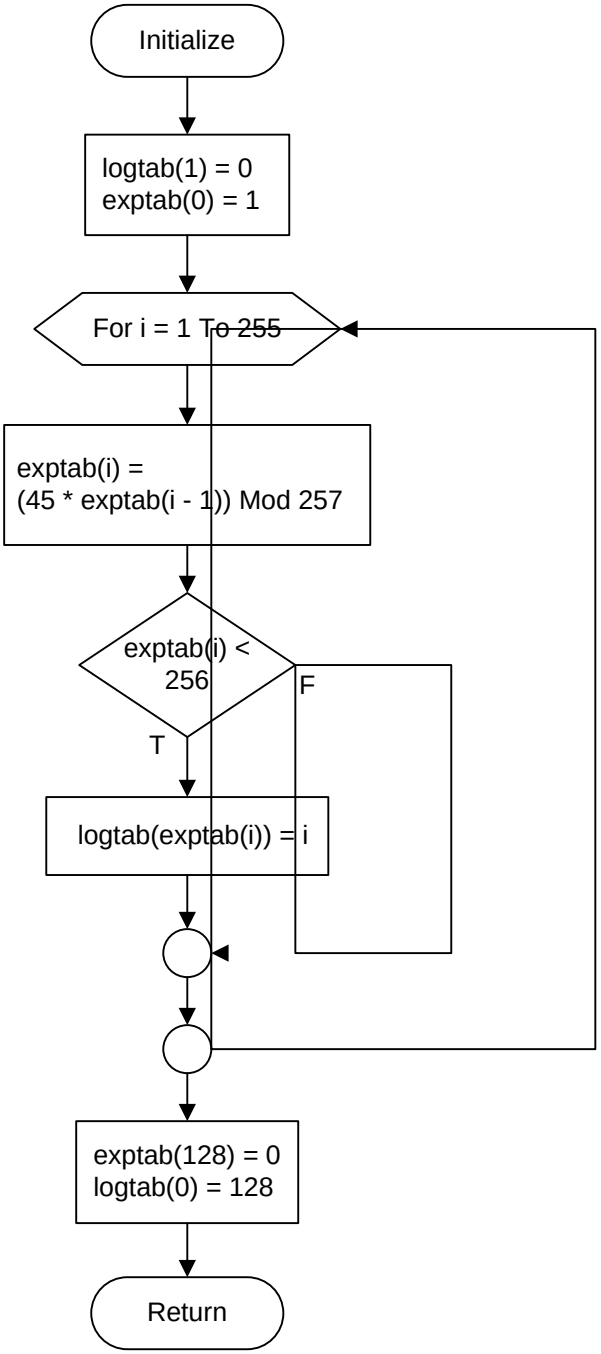
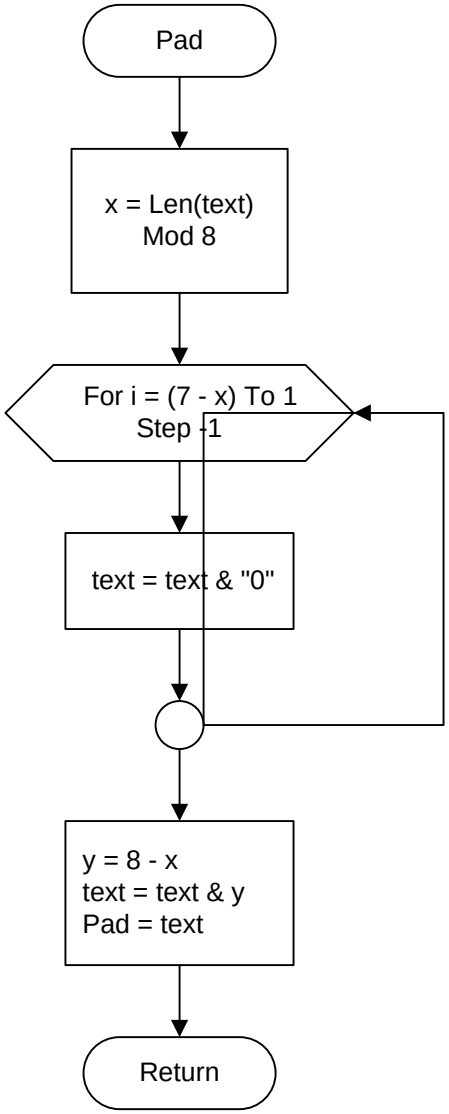
LAMPIRAN A

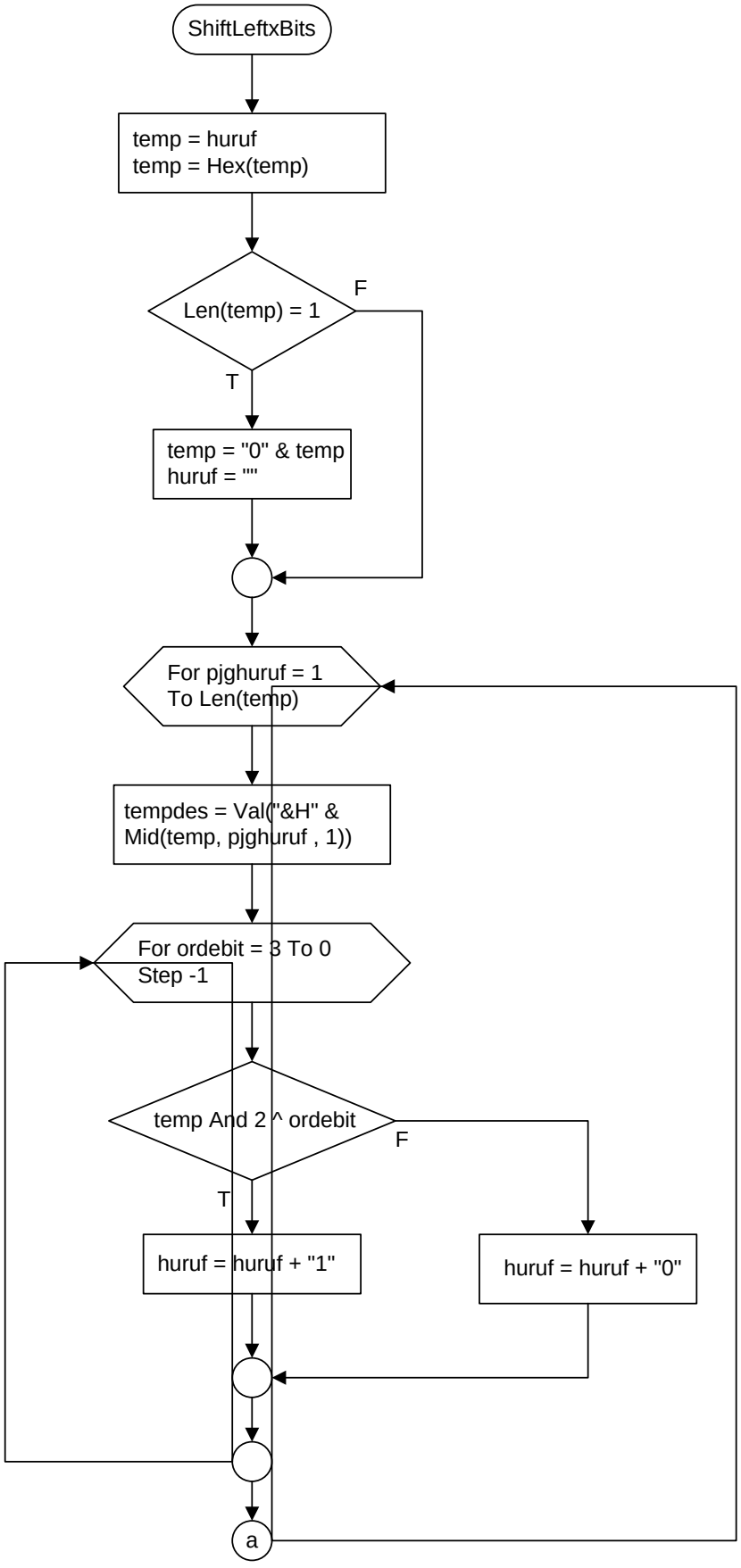
FLOWCHART

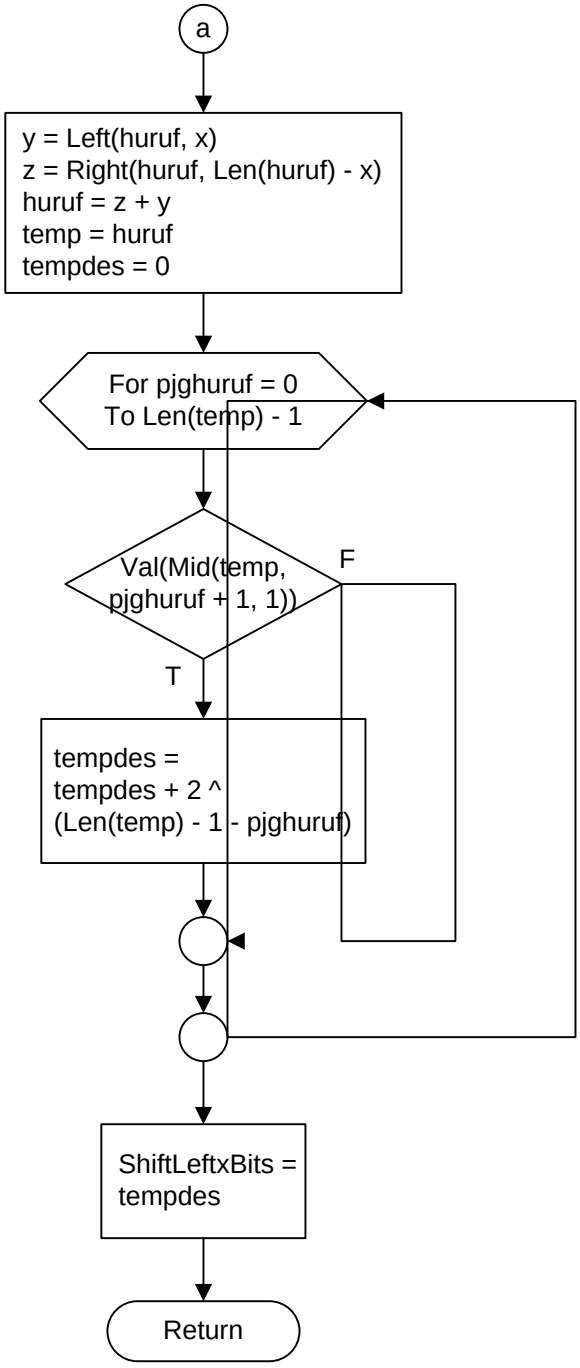


Flowchart fungsi Initialize

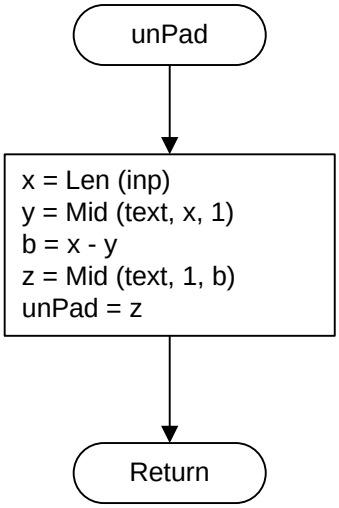


Flowchart fungsi Pad

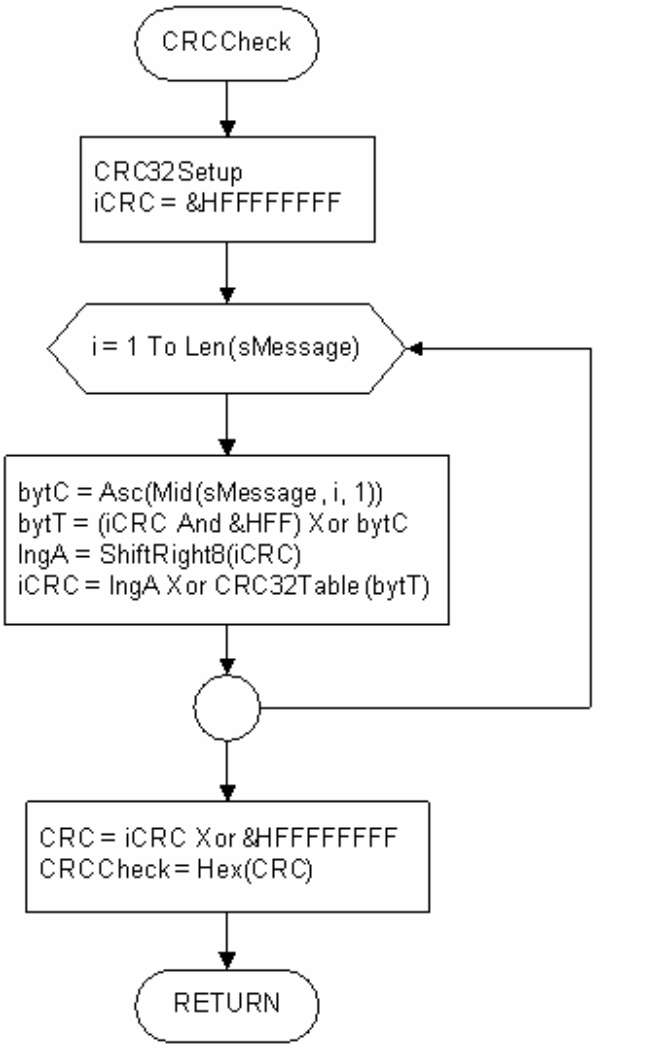

ShiftLeftxBits
temp = huruf
temp = Hex(temp)
Len(temp) = 1
F
T
temp = "0" & temp
huruf = ""
For pjghuruf = 1
To Len(temp)
tempdes = Val("&H" &
Mid(temp, pjghuruf , 1))
For ordebit = 3 To 0
Step -1
temp And 2 ^ ordebit
F
T
huruf = huruf + "1"
huruf = huruf + "0"
a



Flowchart fungsi ShiftLeftxBits



Flowchart fungsi unPad



Flowchart fungsi CRCCheck

LAMPIRAN B

SOURCE CODE

FORM UTAMA

```
Private Sub hapus1()
rtbplain = ""
End Sub

Private Sub hapus2()
rtbcipher = ""
End Sub

Private Sub simpan1()
cdbdialog.Filter = "Text File|*.txt"
cdbdialog.DialogTitle = "Save File"
cdbdialog.ShowSave
simpan = cdbdialog.FileName
If simpan = "" Then
   Exit Sub
End If
   hasil = rtbplain.text
Open simpan For Output As #1
   Print #1, hasil
Close #1
End Sub

Private Sub simpan2()
cdbdialog.DialogTitle = "Save File"
cdbdialog.Filter = "Text File|*.txt"
cdbdialog.ShowSave
simpan = cdbdialog.FileName
If simpan = "" Then
   Exit Sub
End If
   hasil = rtbcipher.text
Open simpan For Output As #1
Print #1, hasil
Close #1
End Sub

Private Sub buka1()
cdbdialog.Filter = "Text File|*.txt"
cdbdialog.FileName = ""
cdbdialog.DialogTitle = "Open File"
cdbdialog.ShowOpen
If cdbdialog.FileName = "" Then
   Exit Sub
End If
```

```
    rtbplain.FileName = cdbdialog.FileName
    x = rtbplain.text
    pjg = Len(x)
    temp = Mid(x, pjg - 1, 2)
    For i = 1 To 2
    p = Mid(temp, i, 1)
    h = Hex(Asc(p))
    c = c & h
    Next i
    If c = "DA" Then
    hasil = Mid(x, 1, pjg - 2)
    Else
    hasil = x
    End If
txttime.text = ""
rtbcipher = ""
rtbplain.text = hasil
End Sub

Private Sub buka2()
cdbdialog.Filter = "Text File|*.txt"
cdbdialog.FileName = ""
cdbdialog.DialogTitle = "Save File"
cdbdialog.ShowOpen
If cdbdialog.FileName = "" Then
    Exit Sub
End If
    rtbcipher.FileName = cdbdialog.FileName
    x = rtbcipher.text
    pjg = Len(x)
    temp = Mid(x, pjg - 1, 2)
    For i = 1 To 2
    p = Hex(Asc(Mid(temp, i, 1)))
    c = c & p
    Next i
    If c = "DA" Then
    hasil = Mid(x, 1, pjg - 2)
    Else
    hasil = x
    End If
txttime.text = ""
rtbplain = ""
rtbcipher.text = hasil
End Sub

Private Sub cmdenkrip_Click()
Dim pteks, kunci As String, round
```

```
cmdenkrip.Enabled = False
cmddekrip.Enabled = False

pteks = rtbplain.text
kunci = txtkey.text
tipe = cbtipe.text

If rtbplain.text = "" Then
   s = MsgBox("Please insert text into PLAINTEXT box!", vbExclamation,
"ERROR")
   rtbplain.SetFocus
   GoTo bawah
End If

vad = CRCCheck(CStr(pteks))

Select Case tipe
   Case "SAFER K-64":
      If Not Len(kunci) = 8 Then
      s = MsgBox("Please insert keyword in 8 character!", vbExclamation,
"ERROR")
      txtkey.SetFocus
      GoTo bawah
      End If

      If txtround.text = "" Then
      s = MsgBox("Please insert round number of " & tipe & "!", vbExclamation,
"ERROR")
      txtround.SetFocus
      GoTo bawah
      End If

      For z = 1 To Len(txtround.text)
      If Asc(Mid(txtround.text, z, 1)) < 48 Or Asc(Mid(txtround.text, z, 1)) > 57
Then
         s = MsgBox("Please insert round number of " & tipe & " only in
number!", vbExclamation, "ERROR")
         txtround.SetFocus
         GoTo bawah
         Exit For
      End If
      Next z

      MainForm.Caption = MainForm.Caption & " - " & "Working ..."
      round = CLng(txtround.text)
```

```
        tawal = Timer

        cipher = EnkripsiSK64(CStr(kunci), CStr(pteks), CLng(round))

        takhir = Timer
        txttime.text = takhir - tawal

    Case "SAFER K-128":
        If Len(kunci) > 16 Or kunci = "" Then
        s = MsgBox("Please insert keyword in maximum 16 character!",
vbExclamation, "ERROR")
        txtkey.SetFocus
        GoTo bawah
        End If

        If Not Len(kunci) = 16 Then
        x = 16 - Len(kunci)
        For i = x - 1 To 1 Step -1
        kunci = kunci & "0"
        Next i
        kunci = kunci & Hex(x)
        End If

        If txtround.text = "" Then
        s = MsgBox("Please insert round number of " & tipe & "!", vbExclamation,
"ERROR")
        txtround.SetFocus
        GoTo bawah
        End If

        For z = 1 To Len(txtround.text)
        If Asc(Mid(txtround.text, z, 1)) < 48 Or Asc(Mid(txtround.text, z, 1)) > 57
Then
            s = MsgBox("Please insert round number of " & tipe & " only in
number!", vbExclamation, "ERROR")
            txtround.SetFocus
            GoTo bawah
            Exit For
        End If
        Next z

        MainForm.Caption = MainForm.Caption & " - " & "Working ..."
        round = CLng(txtround.text)

        tawal = Timer

        cipher = EnkripsiSK128(CStr(kunci), CStr(pteks), CLng(round))
```

```
        takhir = Timer
        txttime.text = takhir - tawal

    Case "":
        s = MsgBox("Please select type of encryption!", vbExclamation, "ERROR")
        cbtipe.SetFocus
        GoTo bawah

    Case Else:
        s = MsgBox("Please select the correct type of encryption, SAFER K-64 or
SAFER K-128!", vbExclamation, "ERROR")
        cbtipe.SetFocus
        GoTo bawah
End Select

    vcipher = cipher & vad
    On Error Resume Next
    rtbcipher.text = vcipher

bawah:
If tipe = "SAFER K-64" Or tipe = "SAFER K-128" Then
    MainForm.Caption = tipe
Else
    MainForm.Caption = "SAFER"
End If
cmdenkrip.Enabled = True
cmddekrip.Enabled = True
End Sub

Private Sub cmddekrip_Click()
Dim cteks, plain As String, round As Integer

cmdenkrip.Enabled = False
cmddekrip.Enabled = False

cteks = rtbcipher.text
kunci = txtkey.text
tipe = cbtipe.text

If rtbcipher.text = "" Then
    s = MsgBox("Please insert text into CIPHERTEXT box!", vbExclamation,
"ERROR")
    rtbcipher.SetFocus
    GoTo bawah
End If
```

```
vad1 = Mid(cteks, Len(cteks) - 7, 8)
cteks2 = Mid(cteks, 1, Len(cteks) - 8)

Select Case tipe
   Case "SAFER K-64":
      If Not Len(kunci) = 8 Then
      s = MsgBox("Please insert keyword in 8 character!", vbExclamation,
"ERROR")
      txtkey.SetFocus
      GoTo bawah
      End If

      If txtround.text = "" Then
      s = MsgBox("Please insert round number of " & tipe & "!", vbExclamation,
"ERROR")
      txtround.SetFocus
      GoTo bawah
      End If

      For z = 1 To Len(txtround.text)
      If Asc(Mid(txtround.text, z, 1)) < 48 Or Asc(Mid(txtround.text, z, 1)) > 57
Then
         s = MsgBox("Please insert round number of " & tipe & " only in
number!", vbExclamation, "ERROR")
         txtround.SetFocus
         GoTo bawah
         Exit For
      End If
      Next z

      MainForm.Caption = MainForm.Caption & " - " & "Working ..."
      round = CLng(txtround.text)

      tawal = Timer

      plain = DekripsiSK64(CStr(kunci), CStr(cteks2), CLng(round))

      takhir = Timer
      txttime.text = takhir - tawal

   Case "SAFER K-128":
      If Len(kunci) > 16 Or kunci = "" Then
      s = MsgBox("Please insert keyword in maximum 16 character !",
vbExclamation, "ERROR")
      txtkey.SetFocus
      GoTo bawah
      End If
```

```
        If Not Len(kunci) = 16 Then
        x = 16 - Len(kunci)
        For i = x - 1 To 1 Step -1
        kunci = kunci & "0"
        Next i
        kunci = kunci & Hex(x)
        End If

        If txtround.text = "" Then
        s = MsgBox("Please insert round number of " & tipe & "!", vbExclamation,
"ERROR")
        txtround.SetFocus
        GoTo bawah
        End If

        For z = 1 To Len(txtround.text)
        If Asc(Mid(txtround.text, z, 1)) < 48 Or Asc(Mid(txtround.text, z, 1)) > 57
Then
            s = MsgBox("Please insert round number of " & tipe & " only in
number!", vbExclamation, "ERROR")
            txtround.SetFocus
            GoTo bawah
            Exit For
        End If
        Next z

        MainForm.Caption = MainForm.Caption & " - " & "Working ..."
        round = CLng(txtround.text)

        tawal = Timer

        plain = DekripsiSK128(CStr(kunci), CStr(cteks2), CLng(round))

        takhir = Timer
        txttime.text = takhir - tawal

    Case "":
        s = MsgBox("Please select type of decryption!", vbExclamation, "ERROR")
        cbtipe.SetFocus
        GoTo bawah

    Case Else:
        s = MsgBox("Please select the correct type of decryption, SAFER K-64 or
SAFER K-128!", vbExclamation, "ERROR")
        cbtipe.SetFocus
        GoTo bawah
```

```
End Select

   vad2 = CRCCheck(CStr(plain))
   If Not vad1 = vad2 Then
      temp = MsgBox("Data is not valid  ", vbInformation, "Information")
   End If
   rtbplain.text = plain

bawah:
If tipe = "SAFER K-64" Or tipe = "SAFER K-128" Then
MainForm.Caption = tipe
Else
MainForm.Caption = "SAFER"
End If
cmdenkrip.Enabled = True
cmddekrip.Enabled = True

End Sub

Private Sub cbtipe_click()
Dim tipe As String
tipe = cbtipe.text
If tipe = "SAFER K-64" Then
   txtkey.ToolTipText = "in 8 character"
   MainForm.Caption = "SAFER K-64"
Else
   txtkey.ToolTipText = "maximum 16 character"
   MainForm.Caption = "SAFER K-128"
End If
End Sub

Private Sub mnabout_Click()
   Load AboutForm
   AboutForm.Show
End Sub

Private Sub mnbuka1_Click()
   Call buka1
End Sub

Private Sub mnbuka2_Click()
   Call buka2
End Sub

Private Sub mnexit_Click()
   End
End Sub
```

```
Private Sub mnhapus1_Click()
   Call hapus1
End Sub

Private Sub mnhapus2_Click()
   Call hapus2
End Sub

Private Sub mnreadme_Click()
   Load ReadMeForm
   ReadMeForm.Show
End Sub

Private Sub mnreset_Click()
   Call hapus1
   Call hapus2
   cbtipe = ""
   txtkey = ""
   txtround = ""
End Sub

Private Sub mnsimpan1_Click()
   Call simpan1
End Sub

Private Sub mnsimpan2_Click()
   Call simpan2
End Sub
```

FORM ABOUT

```
Private Declare Function ShellExecute Lib "shell32.dll" _
   Alias "ShellExecuteA" (ByVal hWnd As Long, ByVal _
   lpOperation As String, ByVal lpFile As String, ByVal _
   lpParameters As String, ByVal lpDirectory As String, _
   ByVal nShowCmd As Long) As Long

Private Sub cmdOK_Click()
    Unload AboutForm
End Sub

Private Sub Label8_Click()
   'Membuka hyperlink ke website
   ShellExecute hWnd, "open", "http://www.macphisto.net/", _
   vbNullString, vbNullString, conSwNormal
End Sub

Private Sub Label6_Click()
   'link ke e-mail
   ShellExecute hWnd, "open", "mailto:wwalkon@yahoo.com?Subject=About
SAFER", _
   vbNullString, vbNullString, conSwNormal
End Sub
```

FORM READ ME

```
Private Sub cmdOK_Click()
    Unload ReadMeForm
End Sub
```

MODUL SAFER K-64

```
Public logtab(255) As Integer, exptab(255) As Integer
Private CRC32Table(255) As Long

Public Function EnkripsiSK64(ByVal kunci As String, ByVal pteks As String,
ByVal round As Integer) As String
Dim k(1 To 8) As Byte, kx(1 To 8, 1 To 8) As Byte
Dim a(1 To 8) As Integer, b(1 To 8) As Integer

Initialize

pteks2 = CStr(Pad(pteks))
For i = 1 To Len(kunci)
k(i) = Asc(Mid(kunci, i, 1))
kx(1, i) = k(i)
kx(2, i) = ShiftLeftxBits((kx(1, i)), 3)
kx(3, i) = ShiftLeftxBits((kx(2, i)), 3)
kx(4, i) = ShiftLeftxBits((kx(3, i)), 3)
kx(5, i) = ShiftLeftxBits((kx(4, i)), 3)
kx(6, i) = ShiftLeftxBits((kx(5, i)), 3)
kx(7, i) = ShiftLeftxBits((kx(6, i)), 3)
kx(8, i) = ShiftLeftxBits((kx(7, i)), 3)
Next i

'Awal enkripsi terhadap plaintext

For j = 1 To Len(pteks2) Step 8
   For m = 1 To 8
   a(m) = Asc(Mid(pteks2, m + j - 1, 1))
   Next m

   'Awal iterasi enkripsi

   For n = 1 To round

   If 2 * n - 1 = 1 Then
   a(1) = a(1) Xor k(1)
   a(2) = (a(2) + k(2)) Mod 256
   a(3) = (a(3) + k(3)) Mod 256
   a(4) = a(4) Xor k(4)
   a(5) = a(5) Xor k(5)
   a(6) = (a(6) + k(6)) Mod 256
   a(7) = (a(7) + k(7)) Mod 256
   a(8) = a(8) Xor k(8)
   Else
```

```
   a(1) = a(1) Xor ((kx((2 * n - 2) Mod 8 + 1, (2 * n - 2) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 1) Mod 256))) Mod 256)
   a(2) = (a(2) + (kx((2 * n - 2) Mod 8 + 1, (2 * n - 1) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 2) Mod 256))) Mod 256) Mod 256
   a(3) = (a(3) + (kx((2 * n - 2) Mod 8 + 1, (2 * n - 0) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 3) Mod 256))) Mod 256) Mod 256
   a(4) = a(4) Xor ((kx((2 * n - 2) Mod 8 + 1, (2 * n + 1) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 4) Mod 256))) Mod 256)
   a(5) = a(5) Xor ((kx((2 * n - 2) Mod 8 + 1, (2 * n + 2) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 5) Mod 256))) Mod 256)
   a(6) = (a(6) + (kx((2 * n - 2) Mod 8 + 1, (2 * n + 3) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 6) Mod 256))) Mod 256) Mod 256
   a(7) = (a(7) + (kx((2 * n - 2) Mod 8 + 1, (2 * n + 4) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 7) Mod 256))) Mod 256) Mod 256
   a(8) = a(8) Xor ((kx((2 * n - 2) Mod 8 + 1, (2 * n + 5) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 8) Mod 256))) Mod 256)
   End If

   b(1) = exptab(a(1))
   b(2) = logtab(a(2))
   b(3) = logtab(a(3))
   b(4) = exptab(a(4))
   b(5) = exptab(a(5))
   b(6) = logtab(a(6))
   b(7) = logtab(a(7))
   b(8) = exptab(a(8))

   b(1) = (b(1) + (kx((2 * n - 1) Mod 8 + 1, (2 * n - 1) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 1) Mod 256))) Mod 256) Mod 256
   b(2) = b(2) Xor ((kx((2 * n - 1) Mod 8 + 1, (2 * n - 0) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 2) Mod 256))) Mod 256)
   b(3) = b(3) Xor ((kx((2 * n - 1) Mod 8 + 1, (2 * n + 1) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 3) Mod 256))) Mod 256)
   b(4) = (b(4) + (kx((2 * n - 1) Mod 8 + 1, (2 * n + 2) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 4) Mod 256))) Mod 256) Mod 256
   b(5) = (b(5) + (kx((2 * n - 1) Mod 8 + 1, (2 * n + 3) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 5) Mod 256))) Mod 256) Mod 256
   b(6) = b(6) Xor ((kx((2 * n - 1) Mod 8 + 1, (2 * n + 4) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 6) Mod 256))) Mod 256)
   b(7) = b(7) Xor ((kx((2 * n - 1) Mod 8 + 1, (2 * n + 5) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 7) Mod 256))) Mod 256)
   b(8) = (b(8) + (kx((2 * n - 1) Mod 8 + 1, (2 * n + 6) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 8) Mod 256))) Mod 256) Mod 256

      For p = 1 To 8 Step 2
      a(p) = (2 * b(p) + b(p + 1)) Mod 256
      a(p + 1) = (b(p) + b(p + 1)) Mod 256
```

```
        Next p
        For p = 1 To 4 Step 2
        b(p) = (2 * a(2 * p - 1) + a(2 * p + 1)) Mod 256
        b(p + 1) = (a(2 * p - 1) + a(2 * p + 1)) Mod 256
        Next p
        For p = 5 To 8 Step 2
        b(p) = (2 * a(2 * p - 8) + a(2 * p - 6)) Mod 256
        b(p + 1) = (a(2 * p - 8) + a(2 * p - 6)) Mod 256
        Next p
        For p = 1 To 4 Step 2
        a(p) = (2 * b(2 * p - 1) + b(2 * p + 1)) Mod 256
        a(p + 1) = (b(2 * p - 1) + b(2 * p + 1)) Mod 256
        Next p
        For p = 5 To 8 Step 2
        a(p) = (2 * b(2 * p - 8) + b(2 * p - 6)) Mod 256
        a(p + 1) = (b(2 * p - 8) + b(2 * p - 6)) Mod 256
        Next p

    Next n

    'Akhir iterasi enkripsi
    'Proses akhir untuk memperoleh cryptogram

    a(1) = a(1) Xor ((kx((2 * round) Mod 8 + 1, (2 * round) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 1) Mod 256))) Mod 256)
    a(2) = (a(2) + (kx((2 * round) Mod 8 + 1, (2 * round + 1) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 2) Mod 256))) Mod 256) Mod 256
    a(3) = (a(3) + (kx((2 * round) Mod 8 + 1, (2 * round + 2) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 3) Mod 256))) Mod 256) Mod 256
    a(4) = a(4) Xor ((kx((2 * round) Mod 8 + 1, (2 * round + 3) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 4) Mod 256))) Mod 256)
    a(5) = a(5) Xor ((kx((2 * round) Mod 8 + 1, (2 * round + 4) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 5) Mod 256))) Mod 256)
    a(6) = (a(6) + (kx((2 * round) Mod 8 + 1, (2 * round + 5) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 6) Mod 256))) Mod 256) Mod 256
    a(7) = (a(7) + (kx((2 * round) Mod 8 + 1, (2 * round + 6) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 7) Mod 256))) Mod 256) Mod 256
    a(8) = a(8) Xor ((kx((2 * round) Mod 8 + 1, (2 * round + 7) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 8) Mod 256))) Mod 256)

'Output proses enkripsi

    For r = 1 To 8
    Mid(pteks2, r + j - 1, 1) = Chr(a(r))
    Next r

Next j
```

```
EnkripsiSK64 = pteks2

End Function


Public Sub Initialize()
logtab(1) = 0
exptab(0) = 1
For i = 1 To 255
exptab(i) = (45 * exptab(i - 1)) Mod 257
   If exptab(i) < 256 Then
   logtab(exptab(i)) = i
   End If
Next i
exptab(128) = 0
logtab(0) = 128
End Sub


Public Function Pad(text As String) As String
x = Len(text) Mod 8
For i = (7 - x) To 1 Step -1
   text = text & "0"
Next i
y = 8 - x
text = text & y
   Pad = text
End Function


Public Function ShiftLeftxBits(huruf As String, x As Integer) As String
Dim temp As String
Dim pjghuruf As Integer, ordebit As Integer
Dim tempdes As Byte

   temp = huruf
   temp = Hex(temp)
   If Len(temp) = 1 Then temp = "0" & temp
   huruf = ""

   ' Konversi ke biner

   For pjghuruf = 1 To Len(temp)
      tempdes = Val("&H" & Mid(temp, pjghuruf, 1))
      For ordebit = 3 To 0 Step -1
         If tempdes And 2 ^ ordebit Then
```

```
            huruf = huruf + "1"
         Else
            huruf = huruf + "0"
         End If
      Next ordebit
   Next pjghuruf

      y = Left(huruf, x)
      z = Right(huruf, Len(huruf) - x)
      huruf = z + y

   temp = huruf

    ' Konversi ke desimal

   tempdes = 0
   For pjghuruf = 0 To Len(temp) - 1
      If Val(Mid(temp, pjghuruf + 1, 1)) Then
      tempdes = tempdes + 2 ^ (Len(temp) - 1 - pjghuruf)
      End If
   Next pjghuruf

   ShiftLeftxBits = tempdes
End Function


Public Function DekripsiSK64(ByVal kunci As String, ByVal cteks As String,
ByVal round As Integer) As String
Dim k(1 To 8) As Byte, kx(1 To 8, 1 To 8) As Byte
Dim a(1 To 8) As Integer, b(1 To 8) As Integer

Initialize

For i = 1 To Len(kunci)
k(i) = Asc(Mid(kunci, i, 1))
kx(1, i) = k(i)
kx(2, i) = ShiftLeftxBits((kx(1, i)), 3)
kx(3, i) = ShiftLeftxBits((kx(2, i)), 3)
kx(4, i) = ShiftLeftxBits((kx(3, i)), 3)
kx(5, i) = ShiftLeftxBits((kx(4, i)), 3)
kx(6, i) = ShiftLeftxBits((kx(5, i)), 3)
kx(7, i) = ShiftLeftxBits((kx(6, i)), 3)
kx(8, i) = ShiftLeftxBits((kx(7, i)), 3)
Next i

'Awal dekripsi terhadap ciphertext
```

```
For j = 1 To Len(cteks) Step 8
   For m = 1 To 8
   If Not (Mid(cteks, m + j - 1, 1)) = "" Then
   a(m) = Asc(Mid(cteks, m + j - 1, 1))
   End If
   Next m

   'Proses awal untuk memperoleh input iterasi

   a(1) = a(1) Xor ((kx((2 * round) Mod 8 + 1, (2 * round) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 1) Mod 256))) Mod 256)
   a(2) = (a(2) - (kx((2 * round) Mod 8 + 1, (2 * round + 1) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 2) Mod 256))) Mod 256 + 256) Mod 256
   a(3) = (a(3) - (kx((2 * round) Mod 8 + 1, (2 * round + 2) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 3) Mod 256))) Mod 256 + 256) Mod 256
   a(4) = a(4) Xor ((kx((2 * round) Mod 8 + 1, (2 * round + 3) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 4) Mod 256))) Mod 256)
   a(5) = a(5) Xor ((kx((2 * round) Mod 8 + 1, (2 * round + 4) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 5) Mod 256))) Mod 256)
   a(6) = (a(6) - (kx((2 * round) Mod 8 + 1, (2 * round + 5) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 6) Mod 256))) Mod 256 + 256) Mod 256
   a(7) = (a(7) - (kx((2 * round) Mod 8 + 1, (2 * round + 6) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 7) Mod 256))) Mod 256 + 256) Mod 256
   a(8) = a(8) Xor ((kx((2 * round) Mod 8 + 1, (2 * round + 7) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 8) Mod 256))) Mod 256)

   'Memulai iterasi dekripsi

   For n = round To 1 Step -1

      For p = 1 To 8 Step 2
      b(p) = (a(p) - a(p + 1) + 256) Mod 256
      b(p + 1) = (-a(p) + 2 * a(p + 1) + 256) Mod 256
      Next p
      h = 0
      For p = 1 To 8 Step 2
      a(p) = (b(p - h) - b(p + 4 - h) + 256) Mod 256
      a(p + 1) = (-b(p - h) + 2 * b(p + 4 - h) + 256) Mod 256
      h = h + 1
      Next p
      t = 0
      For p = 1 To 8 Step 2
      b(p) = (a(p - t) - a(p + 4 - t) + 256) Mod 256
      b(p + 1) = (-a(p - t) + 2 * a(p + 4 - t) + 256) Mod 256
      t = t + 1
      Next p
```

```
   b(1) = (b(1) - (kx((2 * n - 1) Mod 8 + 1, (2 * n - 1) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 1) Mod 256))) Mod 256 + 256) Mod 256
   b(2) = b(2) Xor ((kx((2 * n - 1) Mod 8 + 1, (2 * n - 0) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 2) Mod 256))) Mod 256)
   b(3) = b(3) Xor ((kx((2 * n - 1) Mod 8 + 1, (2 * n + 1) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 3) Mod 256))) Mod 256)
   b(4) = (b(4) - (kx((2 * n - 1) Mod 8 + 1, (2 * n + 2) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 4) Mod 256))) Mod 256 + 256) Mod 256
   b(5) = (b(5) - (kx((2 * n - 1) Mod 8 + 1, (2 * n + 3) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 5) Mod 256))) Mod 256 + 256) Mod 256
   b(6) = b(6) Xor ((kx((2 * n - 1) Mod 8 + 1, (2 * n + 4) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 6) Mod 256))) Mod 256)
   b(7) = b(7) Xor ((kx((2 * n - 1) Mod 8 + 1, (2 * n + 5) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 7) Mod 256))) Mod 256)
   b(8) = (b(8) - (kx((2 * n - 1) Mod 8 + 1, (2 * n + 6) Mod 8 + 1) +
exptab(exptab((9 * 2 * n + 8) Mod 256))) Mod 256 + 256) Mod 256

   a(1) = logtab(b(1))
   a(2) = exptab(b(2))
   a(3) = exptab(b(3))
   a(4) = logtab(b(4))
   a(5) = logtab(b(5))
   a(6) = exptab(b(6))
   a(7) = exptab(b(7))
   a(8) = logtab(b(8))

   If 2 * n - 1 = 1 Then
   a(1) = a(1) Xor k(1)
   a(2) = (a(2) - k(2) + 256) Mod 256
   a(3) = (a(3) - k(3) + 256) Mod 256
   a(4) = a(4) Xor k(4)
   a(5) = a(5) Xor k(5)
   a(6) = (a(6) - k(6) + 256) Mod 256
   a(7) = (a(7) - k(7) + 256) Mod 256
   a(8) = a(8) Xor k(8)
   Else
   a(1) = a(1) Xor ((kx((2 * n - 2) Mod 8 + 1, (2 * n - 2) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 1) Mod 256))) Mod 256)
   a(2) = (a(2) - (kx((2 * n - 2) Mod 8 + 1, (2 * n - 1) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 2) Mod 256))) Mod 256 + 256) Mod 256
   a(3) = (a(3) - (kx((2 * n - 2) Mod 8 + 1, (2 * n - 0) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 3) Mod 256))) Mod 256 + 256) Mod 256
   a(4) = a(4) Xor ((kx((2 * n - 2) Mod 8 + 1, (2 * n + 1) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 4) Mod 256))) Mod 256)
   a(5) = a(5) Xor ((kx((2 * n - 2) Mod 8 + 1, (2 * n + 2) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 5) Mod 256))) Mod 256)
```

```
   a(6) = (a(6) - (kx((2 * n - 2) Mod 8 + 1, (2 * n + 3) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 6) Mod 256))) Mod 256 + 256) Mod 256
   a(7) = (a(7) - (kx((2 * n - 2) Mod 8 + 1, (2 * n + 4) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 7) Mod 256))) Mod 256 + 256) Mod 256
   a(8) = a(8) Xor ((kx((2 * n - 2) Mod 8 + 1, (2 * n + 5) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 8) Mod 256))) Mod 256)
   End If

   Next n

   'Iterasi dekripsi selesai
   'Output proses dekripsi

   For r = 1 To 8
   If Not (Mid(cteks, r + j - 1, 1)) = "" Then
   Mid(cteks, r + j - 1, 1) = Chr(a(r))
   End If
   Next r

Next j

DekripsiSK64 = unPad(CStr(cteks))

End Function


Public Function unPad(inp As String) As String
On Error GoTo ErrunPad
   x = Len(inp)
   y = Mid(inp, x, 1)
   b = CLng(x - y)
   z = CStr(Mid(inp, 1, b))
   unPad = z
Exit Function

ErrunPad:
   unPad = inp
End Function


Public Function CRCCheck(sMessage As String) As String
   Dim iCRC As Long
   Dim bytT As Byte
   Dim bytC As Byte
   Dim lngA As Long

   Call CRC32Setup
```

```
    iCRC = &HFFFFFFFF
    For i = 1 To Len(sMessage)
        bytC = Asc(Mid(sMessage, i, 1))
        bytT = (iCRC And &HFF) Xor bytC
        lngA = ShiftRight8(iCRC)
        iCRC = lngA Xor CRC32Table(bytT)
    Next

    CRC = iCRC Xor &HFFFFFFFF
    CRCCheck = Hex(CRC)

End Function


Public Function ShiftRight8(x As Long) As Long
    Dim iNew As Long
    iNew = (x And &H7FFFFFFF) \ 256
    If (x And &H80000000) <> 0 Then
        iNew = iNew Or &H800000
    End If
    ShiftRight8 = iNew
End Function


Public Function CRC32Setup()

    Static bDone As Boolean

    Dim vntA As Variant
    Dim i As Integer, iOffset As Integer
    Dim nLen As Integer

    If bDone Then
        Exit Function
    End If

    iOffset = 0
    nLen = 32
    vntA = Array( _
        &H0, &H77073096, &HEE0E612C, &H990951BA, _
        &H76DC419, &H706AF48F, &HE963A535, &H9E6495A3, _
        &HEDB8832, &H79DCB8A4, &HE0D5E91E, &H97D2D988, _
        &H9B64C2B, &H7EB17CBD, &HE7B82D07, &H90BF1D91, _
        &H1DB71064, &H6AB020F2, &HF3B97148, &H84BE41DE, _
        &H1ADAD47D, &H6DDDE4EB, &HF4D4B551, &H83D385C7, _
        &H136C9856, &H646BA8C0, &HFD62F97A, &H8A65C9EC, _
```

```
   &H14015C4F, &H63066CD9, &HFA0F3D63, &H8D080DF5)

For i = iOffset To iOffset + nLen - 1
   CRC32Table(i) = vntA(i - iOffset)
Next
iOffset = iOffset + nLen

vntA = Array( _
   &H3B6E20C8, &H4C69105E, &HD56041E4, &HA2677172, _
   &H3C03E4D1, &H4B04D447, &HD20D85FD, &HA50AB56B, _
   &H35B5A8FA, &H42B2986C, &HDBBBC9D6, &HACBCF940, _
   &H32D86CE3, &H45DF5C75, &HDCD60DCF, &HABD13D59, _
   &H26D930AC, &H51DE003A, &HC8D75180, &HBFD06116, _
   &H21B4F4B5, &H56B3C423, &HCFBA9599, &HB8BDA50F, _
   &H2802B89E, &H5F058808, &HC60CD9B2, &HB10BE924, _
   &H2F6F7C87, &H58684C11, &HC1611DAB, &HB6662D3D)

For i = iOffset To iOffset + nLen - 1
   CRC32Table(i) = vntA(i - iOffset)
Next
iOffset = iOffset + nLen

vntA = Array( _
   &H76DC4190, &H1DB7106, &H98D220BC, &HEFD5102A, _
   &H71B18589, &H6B6B51F, &H9FBFE4A5, &HE8B8D433, _
   &H7807C9A2, &HF00F934, &H9609A88E, &HE10E9818, _
   &H7F6A0DBB, &H86D3D2D, &H91646C97, &HE6635C01, _
   &H6B6B51F4, &H1C6C6162, &H856530D8, &HF262004E, _
   &H6C0695ED, &H1B01A57B, &H8208F4C1, &HF50FC457, _
   &H65B0D9C6, &H12B7E950, &H8BBEB8EA, &HFCB9887C, _
   &H62DD1DDF, &H15DA2D49, &H8CD37CF3, &HFBD44C65)

For i = iOffset To iOffset + nLen - 1
   CRC32Table(i) = vntA(i - iOffset)
Next
iOffset = iOffset + nLen

vntA = Array( _
   &H4DB26158, &H3AB551CE, &HA3BC0074, &HD4BB30E2, _
   &H4ADFA541, &H3DD895D7, &HA4D1C46D, &HD3D6F4FB, _
   &H4369E96A, &H346ED9FC, &HAD678846, &HDA60B8D0, _
   &H44042D73, &H33031DE5, &HAA0A4C5F, &HDD0D7CC9, _
   &H5005713C, &H270241AA, &HBE0B1010, &HC90C2086, _
   &H5768B525, &H206F85B3, &HB966D409, &HCE61E49F, _
   &H5EDEF90E, &H29D9C998, &HB0D09822, &HC7D7A8B4, _
   &H59B33D17, &H2EB40D81, &HB7BD5C3B, &HC0BA6CAD)
```

```
For i = iOffset To iOffset + nLen - 1
   CRC32Table(i) = vntA(i - iOffset)
Next
iOffset = iOffset + nLen

vntA = Array( _
   &HEDB88320, &H9ABFB3B6, &H3B6E20C, &H74B1D29A, _
   &HEAD54739, &H9DD277AF, &H4DB2615, &H73DC1683, _
   &HE3630B12, &H94643B84, &HD6D6A3E, &H7A6A5AA8, _
   &HE40ECF0B, &H9309FF9D, &HA00AE27, &H7D079EB1, _
   &HF00F9344, &H8708A3D2, &H1E01F268, &H6906C2FE, _
   &HF762575D, &H806567CB, &H196C3671, &H6E6B06E7, _
   &HFED41B76, &H89D32BE0, &H10DA7A5A, &H67DD4ACC, _
   &HF9B9DF6F, &H8EBEEFF9, &H17B7BE43, &H60B08ED5)

For i = iOffset To iOffset + nLen - 1
   CRC32Table(i) = vntA(i - iOffset)
Next
iOffset = iOffset + nLen

vntA = Array( _
   &HD6D6A3E8, &HA1D1937E, &H38D8C2C4, &H4FDFF252, _
   &HD1BB67F1, &HA6BC5767, &H3FB506DD, &H48B2364B, _
   &HD80D2BDA, &HAF0A1B4C, &H36034AF6, &H41047A60, _
   &HDF60EFC3, &HA867DF55, &H316E8EEF, &H4669BE79, _
   &HCB61B38C, &HBC66831A, &H256FD2A0, &H5268E236, _
   &HCC0C7795, &HBB0B4703, &H220216B9, &H5505262F, _
   &HC5BA3BBE, &HB2BD0B28, &H2BB45A92, &H5CB36A04, _
   &HC2D7FFA7, &HB5D0CF31, &H2CD99E8B, &H5BDEAE1D)

For i = iOffset To iOffset + nLen - 1
   CRC32Table(i) = vntA(i - iOffset)
Next
iOffset = iOffset + nLen

vntA = Array( _
   &H9B64C2B0, &HEC63F226, &H756AA39C, &H26D930A, _
   &H9C0906A9, &HEB0E363F, &H72076785, &H5005713, _
   &H95BF4A82, &HE2B87A14, &H7BB12BAE, &HCB61B38, _
   &H92D28E9B, &HE5D5BE0D, &H7CDCEFB7, &HBDBDF21, _
   &H86D3D2D4, &HF1D4E242, &H68DDB3F8, &H1FDA836E, _
   &H81BE16CD, &HF6B9265B, &H6FB077E1, &H18B74777, _
   &H88085AE6, &HFF0F6A70, &H66063BCA, &H11010B5C, _
   &H8F659EFF, &HF862AE69, &H616BFFD3, &H166CCF45)

For i = iOffset To iOffset + nLen - 1
   CRC32Table(i) = vntA(i - iOffset)
```

```
    Next
    iOffset = iOffset + nLen

    vntA = Array( _
       &HA00AE278, &HD70DD2EE, &H4E048354, &H3903B3C2, _
       &HA7672661, &HD06016F7, &H4969474D, &H3E6E77DB, _
       &HAED16A4A, &HD9D65ADC, &H40DF0B66, &H37D83BF0, _
       &HA9BCAE53, &HDEBB9EC5, &H47B2CF7F, &H30B5FFE9, _
       &HBDBDF21C, &HCABAC28A, &H53B39330, &H24B4A3A6, _
       &HBAD03605, &HCDD70693, &H54DE5729, &H23D967BF, _
       &HB3667A2E, &HC4614AB8, &H5D681B02, &H2A6F2B94, _
       &HB40BBE37, &HC30C8EA1, &H5A05DF1B, &H2D02EF8D)

    For i = iOffset To iOffset + nLen - 1
       CRC32Table(i) = vntA(i - iOffset)
    Next
    iOffset = iOffset + nLen

    bDone = True

End Function
```

MODUL SAFER K-128

```
Public Function EnkripsiSK128(ByVal kunci As String, ByVal pteks As String,
ByVal round As Integer) As String
Dim ka(1 To 8) As Byte, kb(1 To 8) As Byte, kax(1 To 4, 1 To 8) As Byte,
kbx(1 To 4, 1 To 8) As Byte
Dim a(1 To 8) As Integer, b(1 To 8) As Integer

Initialize

pteks2 = CStr(Pad(pteks))

For i = 1 To Len(kunci) / 2
ka(i) = Asc(Mid(kunci, i, 1))
kb(i) = Asc(Mid(kunci, 8 + i, 1))
Next i

For i = 1 To 8
kbx(1, i) = kb(i)
kax(1, i) = ShiftLeftxBits((ka(i)), 3)
kbx(2, i) = ShiftLeftxBits((kbx(1, i)), 6)
kax(2, i) = ShiftLeftxBits((kax(1, i)), 6)
```

```
kbx(3, i) = ShiftLeftxBits((kbx(2, i)), 6)
kax(3, i) = ShiftLeftxBits((kax(2, i)), 6)
kbx(4, i) = ShiftLeftxBits((kbx(3, i)), 6)
kax(4, i) = ShiftLeftxBits((kax(3, i)), 6)
Next i

'Awal enkripsi terhadap plaintext

For j = 1 To Len(pteks2) Step 8
   For m = 1 To 8
   a(m) = Asc(Mid(pteks2, m + j - 1, 1))
   Next m

   'Awal iterasi enkripsi

   For n = 1 To round

   If 2 * n - 1 = 1 Then
   a(1) = a(1) Xor kb(1)
   a(2) = (a(2) + kb(2)) Mod 256
   a(3) = (a(3) + kb(3)) Mod 256
   a(4) = a(4) Xor kb(4)
   a(5) = a(5) Xor kb(5)
   a(6) = (a(6) + kb(6)) Mod 256
   a(7) = (a(7) + kb(7)) Mod 256
   a(8) = a(8) Xor kb(8)

   Else
   a(1) = a(1) Xor ((kbx((n - 1) Mod 4 + 1, (2 * n - 2) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 1) Mod 256))) Mod 256)
   a(2) = (a(2) + (kbx((n - 1) Mod 4 + 1, (2 * n - 1) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 2) Mod 256))) Mod 256) Mod 256
   a(3) = (a(3) + (kbx((n - 1) Mod 4 + 1, (2 * n - 0) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 3) Mod 256))) Mod 256) Mod 256
   a(4) = a(4) Xor ((kbx((n - 1) Mod 4 + 1, (2 * n + 1) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 4) Mod 256))) Mod 256)
   a(5) = a(5) Xor ((kbx((n - 1) Mod 4 + 1, (2 * n + 2) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 5) Mod 256))) Mod 256)
   a(6) = (a(6) + (kbx((n - 1) Mod 4 + 1, (2 * n + 3) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 6) Mod 256))) Mod 256) Mod 256
   a(7) = (a(7) + (kbx((n - 1) Mod 4 + 1, (2 * n + 4) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 7) Mod 256))) Mod 256) Mod 256
   a(8) = a(8) Xor ((kbx((n - 1) Mod 4 + 1, (2 * n + 5) Mod 8 + 1) +
exptab(exptab((9 * (2 * n - 1) + 8) Mod 256))) Mod 256)
   End If

   b(1) = exptab(a(1))
```

```
    b(2) = logtab(a(2))
    b(3) = logtab(a(3))
    b(4) = exptab(a(4))
    b(5) = exptab(a(5))
    b(6) = logtab(a(6))
    b(7) = logtab(a(7))
    b(8) = exptab(a(8))

    b(1) = (b(1) + (kax((n - 1) Mod 4 + 1, (2 * n - 1) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 1) Mod 256))) Mod 256) Mod 256
    b(2) = b(2) Xor ((kax((n - 1) Mod 4 + 1, (2 * n - 0) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 2) Mod 256))) Mod 256)
    b(3) = b(3) Xor ((kax((n - 1) Mod 4 + 1, (2 * n + 1) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 3) Mod 256))) Mod 256)
    b(4) = (b(4) + (kax((n - 1) Mod 4 + 1, (2 * n + 2) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 4) Mod 256))) Mod 256) Mod 256
    b(5) = (b(5) + (kax((n - 1) Mod 4 + 1, (2 * n + 3) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 5) Mod 256))) Mod 256) Mod 256
    b(6) = b(6) Xor ((kax((n - 1) Mod 4 + 1, (2 * n + 4) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 6) Mod 256))) Mod 256)
    b(7) = b(7) Xor ((kax((n - 1) Mod 4 + 1, (2 * n + 5) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 7) Mod 256))) Mod 256)
    b(8) = (b(8) + (kax((n - 1) Mod 4 + 1, (2 * n + 6) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 8) Mod 256))) Mod 256) Mod 256

        For p = 1 To 8 Step 2
        a(p) = (2 * b(p) + b(p + 1)) Mod 256
        a(p + 1) = (b(p) + b(p + 1)) Mod 256
        Next p
        For p = 1 To 4 Step 2
        b(p) = (2 * a(2 * p - 1) + a(2 * p + 1)) Mod 256
        b(p + 1) = (a(2 * p - 1) + a(2 * p + 1)) Mod 256
        Next p
        For p = 5 To 8 Step 2
        b(p) = (2 * a(2 * p - 8) + a(2 * p - 6)) Mod 256
        b(p + 1) = (a(2 * p - 8) + a(2 * p - 6)) Mod 256
        Next p
        For p = 1 To 4 Step 2
        a(p) = (2 * b(2 * p - 1) + b(2 * p + 1)) Mod 256
        a(p + 1) = (b(2 * p - 1) + b(2 * p + 1)) Mod 256
        Next p
        For p = 5 To 8 Step 2
        a(p) = (2 * b(2 * p - 8) + b(2 * p - 6)) Mod 256
        a(p + 1) = (b(2 * p - 8) + b(2 * p - 6)) Mod 256
        Next p

    Next n
```

```
    'Akhir iterasi enkripsi
    'Proses akhir untuk memperoleh cryptogram

    a(1) = a(1) Xor ((kbx(round Mod 4 + 1, (2 * round) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 1) Mod 256))) Mod 256)
    a(2) = (a(2) + (kbx(round Mod 4 + 1, (2 * round + 1) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 2) Mod 256))) Mod 256) Mod 256
    a(3) = (a(3) + (kbx(round Mod 4 + 1, (2 * round + 2) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 3) Mod 256))) Mod 256) Mod 256
    a(4) = a(4) Xor ((kbx(round Mod 4 + 1, (2 * round + 3) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 4) Mod 256))) Mod 256)
    a(5) = a(5) Xor ((kbx(round Mod 4 + 1, (2 * round + 4) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 5) Mod 256))) Mod 256)
    a(6) = (a(6) + (kbx(round Mod 4 + 1, (2 * round + 5) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 6) Mod 256))) Mod 256) Mod 256
    a(7) = (a(7) + (kbx(round Mod 4 + 1, (2 * round + 6) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 7) Mod 256))) Mod 256) Mod 256
    a(8) = a(8) Xor ((kbx(round Mod 4 + 1, (2 * round + 7) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 8) Mod 256))) Mod 256)

    'Output proses enkripsi

    For r = 1 To 8
    Mid(pteks2, r + j - 1, 1) = Chr(a(r))
    Next r

Next j

EnkripsiSK128 = pteks2

End Function


Public Function DekripsiSK128(ByVal kunci As String, ByVal cteks As String,
ByVal round As Integer) As String
Dim ka(1 To 8) As Byte, kax(1 To 4, 1 To 8) As Byte, kb(1 To 8) As Byte,
kbx(1 To 4, 1 To 8) As Byte
Dim a(1 To 8) As Integer, b(1 To 8) As Integer

Initialize

For i = 1 To Len(kunci) / 2
ka(i) = Asc(Mid(kunci, i, 1))
kb(i) = Asc(Mid(kunci, 8 + i, 1))
Next i
```

```
For i = 1 To 8
kbx(1, i) = kb(i)
kax(1, i) = ShiftLeftxBits((ka(i)), 3)
kbx(2, i) = ShiftLeftxBits((kbx(1, i)), 6)
kax(2, i) = ShiftLeftxBits((kax(1, i)), 6)
kbx(3, i) = ShiftLeftxBits((kbx(2, i)), 6)
kax(3, i) = ShiftLeftxBits((kax(2, i)), 6)
kbx(4, i) = ShiftLeftxBits((kbx(3, i)), 6)
kax(4, i) = ShiftLeftxBits((kax(3, i)), 6)
Next i

'Awal dekripsi terhadap ciphertext

For j = 1 To Len(cteks) Step 8
   For m = 1 To 8
   If Not (Mid(cteks, m + j - 1, 1)) = "" Then
   a(m) = Asc(Mid(cteks, m + j - 1, 1))
   End If
   Next m

   'Proses awal untuk memperoleh input iterasi

   a(1) = a(1) Xor ((kbx(round Mod 4 + 1, (2 * round) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 1) Mod 256))) Mod 256)
   a(2) = (a(2) - (kbx(round Mod 4 + 1, (2 * round + 1) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 2) Mod 256))) Mod 256 + 256) Mod 256
   a(3) = (a(3) - (kbx(round Mod 4 + 1, (2 * round + 2) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 3) Mod 256))) Mod 256 + 256) Mod 256
   a(4) = a(4) Xor ((kbx(round Mod 4 + 1, (2 * round + 3) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 4) Mod 256))) Mod 256)
   a(5) = a(5) Xor ((kbx(round Mod 4 + 1, (2 * round + 4) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 5) Mod 256))) Mod 256)
   a(6) = (a(6) - (kbx(round Mod 4 + 1, (2 * round + 5) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 6) Mod 256))) Mod 256 + 256) Mod 256
   a(7) = (a(7) - (kbx(round Mod 4 + 1, (2 * round + 6) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 7) Mod 256))) Mod 256 + 256) Mod 256
   a(8) = a(8) Xor ((kbx(round Mod 4 + 1, (2 * round + 7) Mod 8 + 1) +
exptab(exptab((9 * (2 * round + 1) + 8) Mod 256))) Mod 256)

   'Memulai iterasi dekripsi

   For n = round To 1 Step -1

      For p = 1 To 8 Step 2
      b(p) = (a(p) - a(p + 1) + 256) Mod 256
      b(p + 1) = (-a(p) + 2 * a(p + 1) + 256) Mod 256
      Next p
```

```
h = 0
For p = 1 To 8 Step 2
a(p) = (b(p - h) - b(p + 4 - h) + 256) Mod 256
a(p + 1) = (-b(p - h) + 2 * b(p + 4 - h) + 256) Mod 256
h = h + 1
Next p
t = 0
For p = 1 To 8 Step 2
b(p) = (a(p - t) - a(p + 4 - t) + 256) Mod 256
b(p + 1) = (-a(p - t) + 2 * a(p + 4 - t) + 256) Mod 256
t = t + 1
Next p
```

b(1) = (b(1) - (kax((n - 1) Mod 4 + 1, (2 * n - 1) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 1) Mod 256))) Mod 256 + 256) Mod 256
b(2) = b(2) Xor ((kax((n - 1) Mod 4 + 1, (2 * n - 0) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 2) Mod 256))) Mod 256)
b(3) = b(3) Xor ((kax((n - 1) Mod 4 + 1, (2 * n + 1) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 3) Mod 256))) Mod 256)
b(4) = (b(4) - (kax((n - 1) Mod 4 + 1, (2 * n + 2) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 4) Mod 256))) Mod 256 + 256) Mod 256
b(5) = (b(5) - (kax((n - 1) Mod 4 + 1, (2 * n + 3) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 5) Mod 256))) Mod 256 + 256) Mod 256
b(6) = b(6) Xor ((kax((n - 1) Mod 4 + 1, (2 * n + 4) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 6) Mod 256))) Mod 256)
b(7) = b(7) Xor ((kax((n - 1) Mod 4 + 1, (2 * n + 5) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 7) Mod 256))) Mod 256)
b(8) = (b(8) - (kax((n - 1) Mod 4 + 1, (2 * n + 6) Mod 8 + 1) + exptab(exptab((9 * 2 * n + 8) Mod 256))) Mod 256 + 256) Mod 256

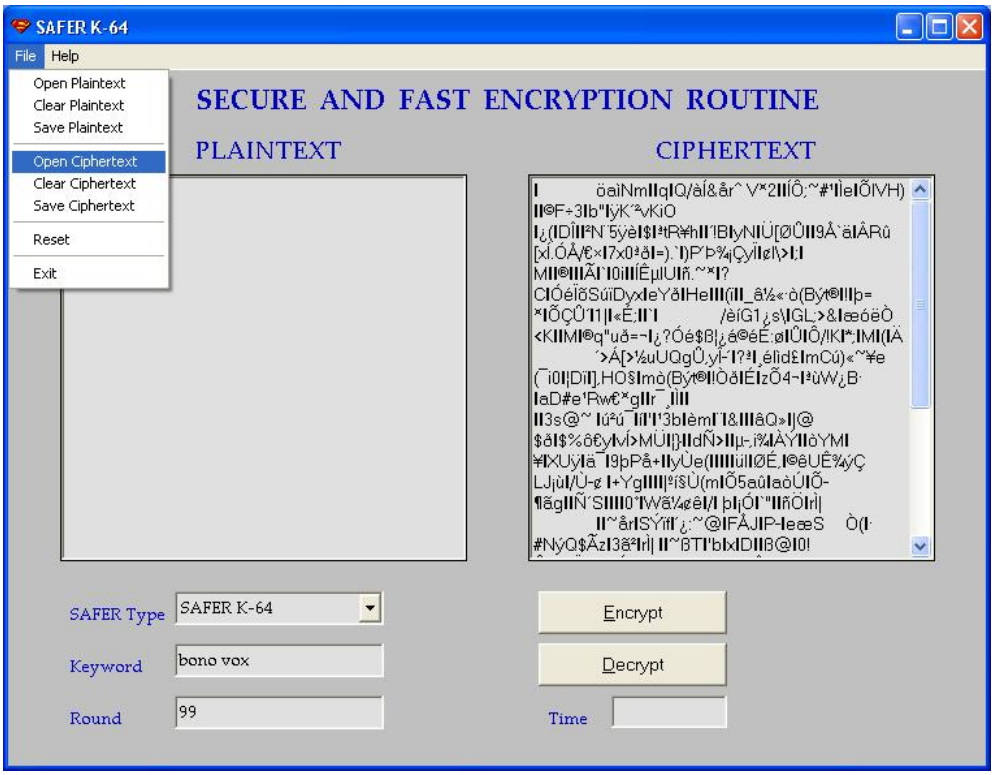
```
a(1) = logtab(b(1))
a(2) = exptab(b(2))
a(3) = exptab(b(3))
a(4) = logtab(b(4))
a(5) = logtab(b(5))
a(6) = exptab(b(6))
a(7) = exptab(b(7))
a(8) = logtab(b(8))

If 2 * n - 1 = 1 Then
a(1) = a(1) Xor kb(1)
a(2) = (a(2) - kb(2) + 256) Mod 256
a(3) = (a(3) - kb(3) + 256) Mod 256
a(4) = a(4) Xor kb(4)
a(5) = a(5) Xor kb(5)
a(6) = (a(6) - kb(6) + 256) Mod 256
a(7) = (a(7) - kb(7) + 256) Mod 256
```

```
    a(8) = a(8) Xor kb(8)
    Else
    a(1) = a(1) Xor ((kbx((n - 1) Mod 4 + 1, (2 * n - 2) Mod 8 + 1) + exptab(exptab((9 * (2 * n - 1) + 1) Mod 256))) Mod 256)
    a(2) = (a(2) - (kbx((n - 1) Mod 4 + 1, (2 * n - 1) Mod 8 + 1) + exptab(exptab((9 * (2 * n - 1) + 2) Mod 256))) Mod 256 + 256) Mod 256
    a(3) = (a(3) - (kbx((n - 1) Mod 4 + 1, (2 * n - 0) Mod 8 + 1) + exptab(exptab((9 * (2 * n - 1) + 3) Mod 256))) Mod 256 + 256) Mod 256
    a(4) = a(4) Xor ((kbx((n - 1) Mod 4 + 1, (2 * n + 1) Mod 8 + 1) + exptab(exptab((9 * (2 * n - 1) + 4) Mod 256))) Mod 256)
    a(5) = a(5) Xor ((kbx((n - 1) Mod 4 + 1, (2 * n + 2) Mod 8 + 1) + exptab(exptab((9 * (2 * n - 1) + 5) Mod 256))) Mod 256)
    a(6) = (a(6) - (kbx((n - 1) Mod 4 + 1, (2 * n + 3) Mod 8 + 1) + exptab(exptab((9 * (2 * n - 1) + 6) Mod 256))) Mod 256 + 256) Mod 256
    a(7) = (a(7) - (kbx((n - 1) Mod 4 + 1, (2 * n + 4) Mod 8 + 1) + exptab(exptab((9 * (2 * n - 1) + 7) Mod 256))) Mod 256 + 256) Mod 256
    a(8) = a(8) Xor ((kbx((n - 1) Mod 4 + 1, (2 * n + 5) Mod 8 + 1) + exptab(exptab((9 * (2 * n - 1) + 8) Mod 256))) Mod 256)
    End If

    Next n

    'Iterasi dekripsi selesai
    'Output proses dekripsi

    For r = 1 To 8
    If Not Mid(cteks, r + j - 1, 1) = "" Then
    Mid(cteks, r + j - 1, 1) = Chr(a(r))
    End If
    Next r

Next j

DekripsiSK128 = unPad(CStr(cteks))

End Function
```

LAMPIRAN C

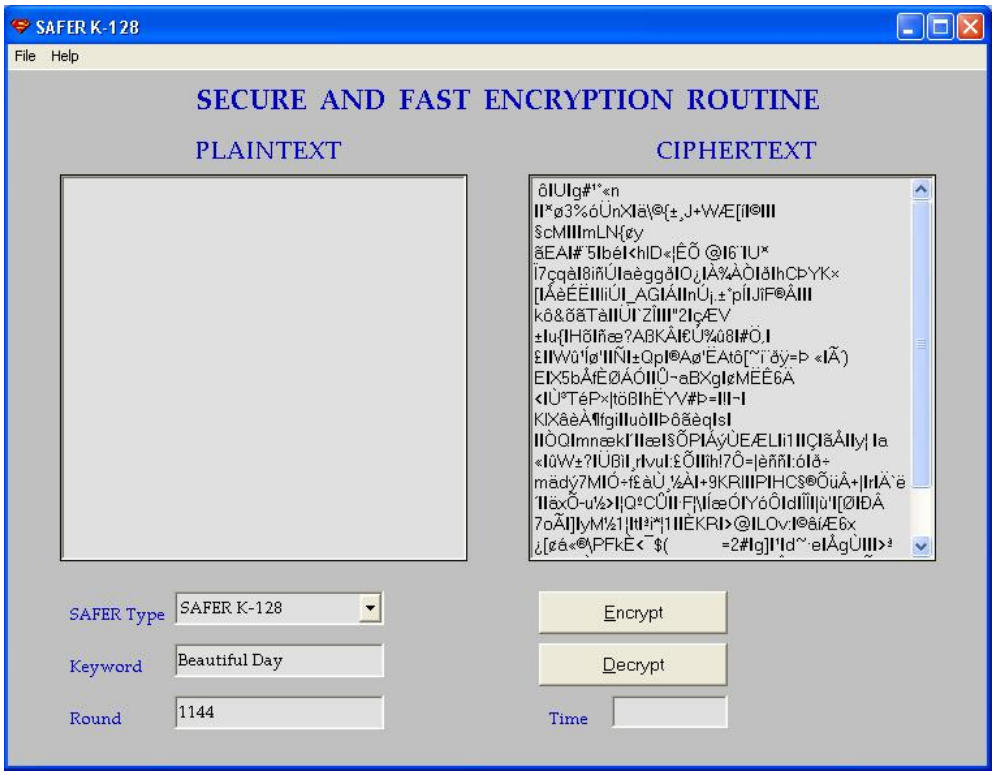
TAMPILAN PROGRAM

Enkripsi SAFER K-64



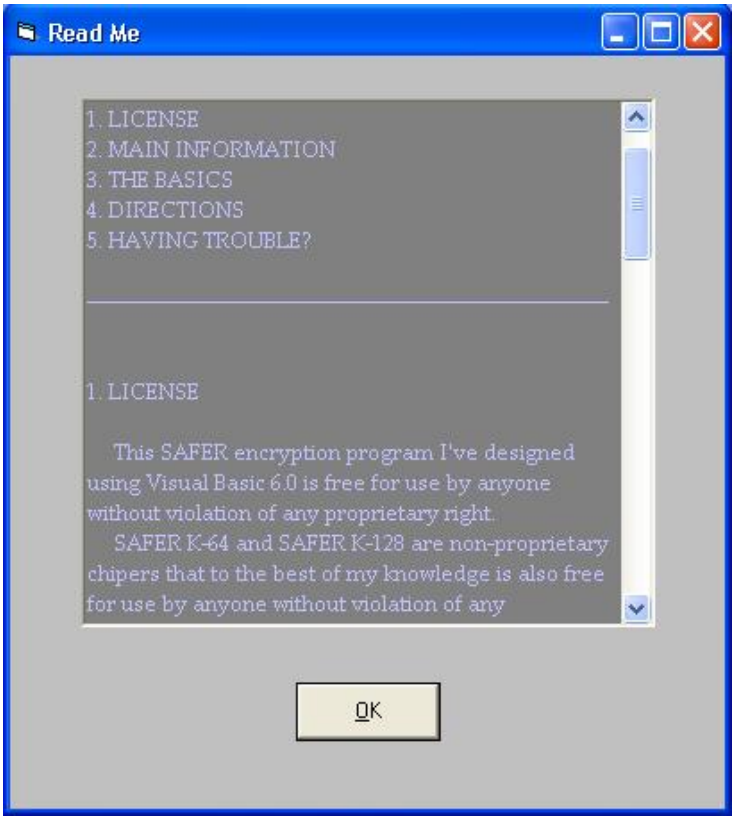
Dekripsi SAFER K-64

Enkripsi SAFER K-128



Dekripsi SAFER K-128

Form About



Form Read Me