



SOURCE CODE PROGRAM

FORM MAIN (Frmmain)

```
Private Sub mnuabout_Click()  
FrmAbout.Left = Frmmain.Left  
FrmAbout.Top = Frmmain.Top  
FrmAbout.Show  
Unload Frmmain  
End Sub
```

```
Private Sub mnudekrip_Click()  
FrmDecrypt.Left = Frmmain.Left  
FrmDecrypt.Top = Frmmain.Top  
FrmDecrypt.Show  
Unload Frmmain  
End Sub
```

```
Private Sub mnuekstraksi_Click()  
frm_Ekstraksi.Left = Frmmain.Left  
frm_Ekstraksi.Top = Frmmain.Top  
frm_Ekstraksi.Show  
Unload Frmmain  
End Sub
```

```
Private Sub mnuenkrip_Click()  
FrmEncrypt.Left = Frmmain.Left  
FrmEncrypt.Top = Frmmain.Top  
FrmEncrypt.Show  
Unload Frmmain  
End Sub
```

```
Private Sub mnuexit_Click()  
End  
End Sub
```

```
Private Sub mnukunci_Click()  
FrmSign.Left = Frmmain.Left  
FrmSign.Top = Frmmain.Top  
FrmSign.Show  
Unload Frmmain  
End Sub
```

```
Private Sub mnuotentikasi_Click()  
FrmOtentikasi.Left = Frmmain.Left  
FrmOtentikasi.Top = Frmmain.Top  
FrmOtentikasi.Show  
Unload Frmmain  
End Sub
```

```
Private Sub mnuttd_Click()  
Frm_ttd_file.Left = Frmmain.Left  
Frm_ttd_file.Top = Frmmain.Top  
Frm_ttd_file.Show  
Unload Frmmain  
End Sub
```

FORM ENKRIPSI (FrmEncrypt)

Event UpdateProgress()

Private Sub CmdBrowseChiper_Click()

 CommonDialog.Filter = ""

 CommonDialog.ShowOpen

 If CommonDialog.FileName <> "" Then

 TxtFileInput2 = CommonDialog.FileName

 End If

 CommonDialog.FileName = ""

End Sub

Private Sub CmdBrowseOutput_Click()

 CommonDialog.Filter = ""

 CommonDialog.ShowSave

 If CommonDialog.FileName <> "" Then

 TxtFileOutput = CommonDialog.FileName

 End If

 CommonDialog.FileName = ""

End Sub

Private Sub CmdEncrypt_Click()

 Dim i As Long

 Dim J As Long

 Dim Key(15) As Integer

 Dim HndFile As Integer

 Dim IsiFile() As Byte

 Dim valInput(7) As Byte

 Dim valOutput(7) As Byte

 Dim JumlahBlok As Long

 Dim JumlahByteBlokTerakhir As Integer

 Dim objNewDes As clsNewDes

 Set objNewDes = New clsNewDes

 Dim tmpString As String

 Dim strExt As String

 If TxtFileInput2 = "" Or TxtFileOutput = "" Then

 s = MsgBox("Masukkan Output File atau Chiper File !!!", vbExclamation, "ERROR")

 TxtFileInput1.SetFocus

 TxtFileOutput.SetFocus

 GoTo bawah

 End If

 If Not Len(TxtKey.Text) = 15 Then

 s = MsgBox("Masukkan Kunci (Key) sebanyak 15 karakter !!!", vbExclamation, "ERROR")

 TxtKey.SetFocus

 GoTo bawah

 End If

 ' baca key

 For i = 0 To 14

 Key(i) = Asc(Mid(TxtKey.Text, i + 1, 1))

 Next

 ' Baca file

 HndFile = FreeFile

 Open TxtFileInput2 For Binary As #HndFile

 ReDim IsiFile(LOF(HndFile) - 1)

 Get #HndFile, , IsiFile

```

' menghitung jumlah blok
JumlahBlok = Int(LOF(HndFile) / 8)
JumlahByteBlokTerakhir = LOF(HndFile) Mod 8
Close #HndFile
tmpString = ""
tmpString = CStr(IsiFile)

'Memberi extension output program
If Right(TxtFileOutput, 4) <> ".nwd" Then
    TxtFileOutput = TxtFileOutput & ".nwd"
End If

tawal = Timer

Open TxtFileOutput For Binary As #HndFile
    Dim tmpByte As Byte
    tmpint = 8 - JumlahByteBlokTerakhir
    tmpByte = Asc(tmpint)
    Put #HndFile, , tmpByte
    Put #HndFile, , CRCCheck(tmpString)

'menyipkan informasi extension file
For i = Len(TxtFileInput2) To 1 Step -1
    If Mid(TxtFileInput2, i, 1) = "." Then
        Exit For ' i keluar
    End If
Next
If i = 1 Then
    strExt = ""
Else
    strExt = Mid(TxtFileInput2, i, Len(TxtFileInput2) - i + 1)
End If

While Len(strExt) < 8
    strExt = strExt & " "
Wend

Put #HndFile, , strExt

'enkrip
For i = 1 To JumlahBlok
    RaiseEvent UpdateProgress
    ProgressBar1.value = 100 * i / JumlahBlok
    For J = 0 To 7
        valInput(J) = IsiFile((i - 1) * 8 + J)
    Next
    objNewDes.Encrypt valInput, Key, valOutput
    Put #HndFile, , valOutput
Next

takhir = Timer
Time.Text = takhir - tawal

' proses padding
If JumlahByteBlokTerakhir > 0 Then
    For J = 0 To JumlahByteBlokTerakhir - 1
        valInput(J) = IsiFile((JumlahBlok) * 8 + J)
    Next

```

```

        objNewDes.Encrypt valInput, Key, valOutput
        Put #HndFile, , valOutput
    End If

    Close #HndFile
    MsgBox ("File telah Dienkrip !!! ")

bawah:
End Sub

Private Sub mnuabout1_Click()
    FrmAbout.Left = FrmEncrypt.Left
    FrmAbout.Top = FrmEncrypt.Top
    FrmAbout.Show
    Unload Me
End Sub

Private Sub mnuexit_Click()
    End
End Sub

Private Sub mnudekrip_Click()
    FrmDecrypt.Left = FrmEncrypt.Left
    FrmDecrypt.Top = FrmEncrypt.Top
    FrmDecrypt.Show
    Unload Me
End Sub

Private Sub mnuekstraksi_Click()
    frm_Ekstraksi.Left = FrmEncrypt.Left
    frm_Ekstraksi.Top = FrmEncrypt.Top
    frm_Ekstraksi.Show
    Unload Me
End Sub

Private Sub mnukunci_Click()
    frm_Ekstraksi.Left = FrmEncrypt.Left
    frm_Ekstraksi.Top = FrmEncrypt.Top
    FrmSign.Show
    Unload Me
End Sub

Private Sub mnuotentikasi_Click()
    FrmOtentikasi.Left = FrmEncrypt.Left
    FrmOtentikasi.Top = FrmEncrypt.Top
    FrmOtentikasi.Show
    Unload Me
End Sub

Private Sub mnuttd_Click()
    Frm_ttd_file.Left = FrmEncrypt.Left
    Frm_ttd_file.Top = FrmEncrypt.Top
    Frm_ttd_file.Show
    Unload Me
End Sub

```

FORM DEKRIPSI (FrmDecrypt)

Event UpdateProgress()

```
Private Sub CmdBrowseencrypt_Click()  
CommonDialog.Filter = "Newdes Encryption (*.nwd) | *.nwd"  
CommonDialog.ShowOpen  
If CommonDialog.FileName <> "" Then  
    TxtFileInput1 = CommonDialog.FileName  
End If  
CommonDialog.FileName = ""  
End Sub
```

```
Private Sub CmdBrowseOutput_Click()  
CommonDialog.Filter = ""  
CommonDialog.ShowSave  
If CommonDialog.FileName <> "" Then  
    TxtFileOutput = CommonDialog.FileName  
End If  
CommonDialog.FileName = ""  
End Sub
```

```
Private Sub CmdDecrypt_Click()  
    Dim tmpCrc As String  
    Dim i As Long  
    Dim J As Long  
    Dim Key(15) As Integer  
    Dim HndFile As Integer  
    Dim IsiFile() As Byte  
    Dim valInput(7) As Byte  
    Dim valOutput() As Byte  
    Dim JumlahBlok As Long  
    Dim JumlahByteBlokTerakhir As Integer  
    Dim objNewDes As clsNewDes  
    Set objNewDes = New clsNewDes  
    Dim StrCrc As String  
    Dim strTmp As String  
  
    If TxtFileInput1 = "" Or TxtFileOutput = "" Then  
        s = MsgBox("Masukkan Output File atau Encrypted File !!!!", vbExclamation, "ERROR")  
        TxtFileInput1.SetFocus  
        TxtFileOutput.SetFocus  
        GoTo bawah  
    End If  
  
    If Not Len(TxtKey.Text) = 15 Then  
        s = MsgBox("Masukkan Kunci (Key) sebanyak 15 karakter !!!!", vbExclamation, "ERROR")  
        TxtKey.SetFocus  
        GoTo bawah  
    End If
```

```
    ReDim valOutput(7)  
    ' convert txtkey  
    For i = 0 To 14  
        Key(i) = Asc(Mid(TxtKey.Text, i + 1, 1))  
    Next  
    ' Baca file  
    HndFile = FreeFile  
    Open TxtFileInput1 For Binary As #HndFile
```

```

ReDim IsiFile(LOF(HndFile) - 1)
Get #HndFile, , IsiFile

' menghitung jumlah blok
JumlahBlok = Int((LOF(HndFile) - (1 + 8 + 8)) / 8)
JumlahBytePadding = Int(Left(Chr(IsiFile(0)), 1))

StrCrc = ""
For i = 1 To 8
    StrCrc = StrCrc & Chr(IsiFile(i))
Next

'Extension File
strExt = ""
For i = 1 To 8
    strExt = strExt & Chr(IsiFile(i + 8))
Next
Close #HndFile

strExt2 = ""
For i = 1 To 8
    If Mid(strExt, i, 1) <> " " Then
        strExt2 = strExt2 & Mid(strExt, i, 1)
    End If
Next

If Right(TxtFileOutput, Len(strExt2)) <> strExt2 Then
    TxtFileOutput = TxtFileOutput & strExt2
End If
tawal = Timer

' dekrip
Open TxtFileOutput For Binary As #HndFile
For i = 1 To JumlahBlok
    RaiseEvent UpdateProgress
    ProgressBar1.value = 100 * i / JumlahBlok
    For J = 0 To 7
        valInput(J) = IsiFile((i - 1) * 8 + J + 9 + 8)
    Next
    objNewDes.Decrypt valInput, Key, valOutput
    If JumlahBytePadding <> 8 And i = JumlahBlok Then
        ReDim Preserve valOutput(7 - JumlahBytePadding)
    End If
    Put #HndFile, , valOutput
Next
Close #HndFile

takhir = Timer
Time.Text = takhir - tawal

'CRC Check
Open TxtFileOutput For Binary As #HndFile
ReDim IsiFile(LOF(HndFile) - 1)
Get #HndFile, , IsiFile
strTmp = ""
strTmp = CStr(IsiFile)
Close #HndFile
If CRCCheck(strTmp) <> StrCrc Then
    MsgBox "Data not valid."
End If

```

```
MsgBox ("File telah Didekrip !!!!!")  
bawah:  
End Sub
```

```
Private Sub mnuabout1_Click()  
FrmAbout.Left = FrmDecrypt.Left  
FrmAbout.Top = FrmDecrypt.Top  
FrmAbout.Show  
Unload Me  
End Sub
```

```
Private Sub mnuekstraksi_Click()  
frm_Ekstraksi.Left = FrmDecrypt.Left  
frm_Ekstraksi.Top = FrmDecrypt.Top  
frm_Ekstraksi.Show  
Unload Me  
End Sub
```

```
Private Sub mnuenkrip_Click()  
FrmEncrypt.Left = FrmDecrypt.Left  
FrmEncrypt.Top = FrmDecrypt.Top  
FrmEncrypt.Show  
Unload Me  
End Sub
```

```
Private Sub mnuexit_Click()  
End  
End Sub
```

```
Private Sub mnukunci_Click()  
FrmSign.Left = FrmDecrypt.Left  
FrmSign.Top = FrmDecrypt.Top  
FrmSign.Show  
Unload Me  
End Sub
```

```
Private Sub mnuotentikasi_Click()  
FrmOtentikasi.Left = FrmDecrypt.Left  
FrmOtentikasi.Top = FrmDecrypt.Top  
FrmOtentikasi.Show  
Unload Me  
End Sub
```

```
Private Sub mnuttd_Click()  
Frm_ttd_file.Left = FrmDecrypt.Left  
Frm_ttd_file.Top = FrmDecrypt.Top  
Frm_ttd_file.Show  
Unload Me  
End Sub
```


MODULE NEWDES (clsNewdes.cls)

Option Explicit

Private f(255) As Integer

Private Sub Inisialisasi()

f(0) = 32	f(53) = 168	f(106) = 34	f(159) = 104
f(1) = 137	f(54) = 250	f(107) = 206	f(160) = 91
f(2) = 239	f(55) = 235	f(108) = 35	f(161) = 136
f(3) = 188	f(56) = 54	f(109) = 69	f(162) = 31
f(4) = 102	f(57) = 141	f(110) = 231	f(163) = 42
f(5) = 125	f(58) = 195	f(111) = 246	f(164) = 243
f(6) = 221	f(59) = 247	f(112) = 29	f(165) = 66
f(7) = 72	f(60) = 240	f(113) = 109	f(166) = 126
f(8) = 212	f(61) = 63	f(114) = 74	f(167) = 135
f(9) = 68	f(62) = 148	f(115) = 71	f(168) = 30
f(10) = 81	f(63) = 2	f(116) = 176	f(169) = 26
f(11) = 37	f(64) = 224	f(117) = 6	f(170) = 87
f(12) = 86	f(65) = 169	f(118) = 60	f(171) = 186
f(13) = 237	f(66) = 214	f(119) = 145	f(172) = 182
f(14) = 147	f(67) = 180	f(120) = 65	f(173) = 154
f(15) = 149	f(68) = 62	f(121) = 13	f(174) = 242
f(16) = 70	f(69) = 22	f(122) = 77	f(175) = 123
f(17) = 229	f(70) = 117	f(123) = 151	f(176) = 82
f(18) = 17	f(71) = 108	f(124) = 12	f(177) = 166
f(19) = 124	f(72) = 19	f(125) = 127	f(178) = 208
f(20) = 115	f(73) = 172	f(126) = 95	f(179) = 39
f(21) = 207	f(74) = 161	f(127) = 199	f(180) = 152
f(22) = 33	f(75) = 159	f(128) = 57	f(181) = 190
f(23) = 20	f(76) = 160	f(129) = 101	f(182) = 113
f(24) = 122	f(77) = 47	f(130) = 5	f(183) = 205
f(25) = 143	f(78) = 43	f(131) = 232	f(184) = 114
f(26) = 25	f(79) = 171	f(132) = 150	f(185) = 105
f(27) = 215	f(80) = 194	f(133) = 210	f(186) = 225
f(28) = 51	f(81) = 175	f(134) = 129	f(187) = 84
f(29) = 183	f(82) = 178	f(135) = 24	f(188) = 73
f(30) = 138	f(83) = 56	f(136) = 181	f(189) = 163
f(31) = 142	f(84) = 196	f(137) = 10	f(190) = 99
f(32) = 146	f(85) = 112	f(138) = 121	f(191) = 111
f(33) = 211	f(86) = 23	f(139) = 187	f(192) = 204
f(34) = 110	f(87) = 220	f(140) = 48	f(193) = 61
f(35) = 173	f(88) = 89	f(141) = 193	f(194) = 200
f(36) = 1	f(89) = 21	f(142) = 139	f(195) = 217
f(37) = 228	f(90) = 164	f(143) = 252	f(196) = 170
f(38) = 189	f(91) = 130	f(144) = 219	f(197) = 15
f(39) = 14	f(92) = 157	f(145) = 64	f(198) = 198
f(40) = 103	f(93) = 8	f(146) = 88	f(199) = 28
f(41) = 78	f(94) = 85	f(147) = 233	f(200) = 192
f(42) = 162	f(95) = 251	f(148) = 96	f(201) = 254
f(43) = 36	f(96) = 216	f(149) = 128	f(202) = 134
f(44) = 253	f(97) = 44	f(150) = 80	f(203) = 234
f(45) = 167	f(98) = 94	f(151) = 53	f(204) = 222
f(46) = 116	f(99) = 179	f(152) = 191	f(205) = 7
f(47) = 255	f(100) = 226	f(153) = 144	f(206) = 236
f(48) = 158	f(101) = 38	f(154) = 218	f(207) = 248
f(49) = 45	f(102) = 90	f(155) = 11	f(208) = 201
f(50) = 185	f(103) = 119	f(156) = 106	f(209) = 41
f(51) = 50	f(104) = 40	f(157) = 132	f(210) = 177
f(52) = 98	f(105) = 202	f(158) = 155	f(211) = 156

f(212) = 92	f(224) = 133	f(236) = 4	f(248) = 118
f(213) = 131	f(225) = 174	f(237) = 107	f(249) = 49
f(214) = 67	f(226) = 75	f(238) = 140	f(250) = 79
f(215) = 249	f(227) = 18	f(239) = 52	f(251) = 230
f(216) = 245	f(228) = 93	f(240) = 58	f(252) = 223
f(217) = 184	f(229) = 209	f(241) = 55	f(253) = 165
f(218) = 203	f(230) = 100	f(242) = 3	f(254) = 153
f(219) = 9	f(231) = 120	f(243) = 244	f(255) = 59
f(220) = 241	f(232) = 76	f(244) = 97	End Sub
f(221) = 0	f(233) = 213	f(245) = 197	
f(222) = 27	f(234) = 16	f(246) = 238	
f(223) = 46	f(235) = 83	f(247) = 227	

```
Public Sub Encrypt(ByRef valInput() As Byte, ByRef Key() As Integer, ByRef valOutput() As Byte)
```

```
    Dim B0 As Integer
    Dim B1 As Integer
    Dim B2 As Integer
    Dim B3 As Integer
    Dim B4 As Integer
    Dim B5 As Integer
    Dim B6 As Integer
    Dim B7 As Integer
    Dim i As Integer
    Dim ex As Integer
    Dim IsTrue As Boolean
```

```
    Inisialisasi
```

```
    B0 = valInput(0)
    B1 = valInput(1)
    B2 = valInput(2)
    B3 = valInput(3)
    B4 = valInput(4)
    B5 = valInput(5)
    B6 = valInput(6)
    B7 = valInput(7)
    i = 0
    ex = 0
    IsTrue = True
```

```
    While (IsTrue = True)
```

```
        B4 = B4 Xor f(B0 Xor Key(i) Xor ex)
        i = i + 1
        If (i = 15) Then
            i = 0
            ex = Key(7)
        End If
```

```
        B5 = B5 Xor f(B1 Xor Key(i) Xor ex)
        i = i + 1
        If (i = 15) Then
            i = 0
            ex = Key(8)
        End If
```

```
        B6 = B6 Xor f(B2 Xor Key(i) Xor ex)
        i = i + 1
        If (i = 15) Then
            i = 0
            ex = Key(9)
```

```

End If

B7 = B7 Xor f(B3 Xor Key(i) Xor ex)
i = i + 1
If (i = 15) Then
    ' keluar dari sub
    valOutput(0) = B0
    valOutput(1) = B1
    valOutput(2) = B2
    valOutput(3) = B3
    valOutput(4) = B4
    valOutput(5) = B5
    valOutput(6) = B6
    valOutput(7) = B7
    IsTrue = False
    Exit Sub
End If

B1 = B1 Xor f(B4 Xor Key(i) Xor ex)
i = i + 1
B2 = B2 Xor f(B4 Xor B5)
B3 = B3 Xor f(B6 Xor Key(i) Xor ex)
i = i + 1
B0 = B0 Xor f(B7 Xor Key(i) Xor ex)
i = i + 1
Wend
End Sub
Public Sub Decrypt(ByRef valInput() As Byte, ByRef Key() As Integer, ByRef valOutput() As Byte)
    Dim B0 As Integer
    Dim B1 As Integer
    Dim B2 As Integer
    Dim B3 As Integer
    Dim B4 As Integer
    Dim B5 As Integer
    Dim B6 As Integer
    Dim B7 As Integer
    Dim i As Integer
    Dim ex As Integer
    Dim IsTrue As Boolean

    Inisialisasi

    B0 = valInput(0)
    B1 = valInput(1)
    B2 = valInput(2)
    B3 = valInput(3)
    B4 = valInput(4)
    B5 = valInput(5)
    B6 = valInput(6)
    B7 = valInput(7)
    i = 14
    ex = Key(9)
    IsTrue = True

    While (IsTrue = True)
        B7 = B7 Xor f(B3 Xor Key(i) Xor ex)
        i = i - 1
        If (i < 0) Then
            i = 14
            ex = Key(8)

```

```

End If

B6 = B6 Xor f(B2 Xor Key(i) Xor ex)
i = i - 1
If (i < 0) Then
i = 14
ex = Key(7)
End If

B5 = B5 Xor f(B1 Xor Key(i) Xor ex)
i = i - 1
If (i < 0) Then
i = 14
ex = 0
End If

B4 = B4 Xor f(B0 Xor Key(i) Xor ex)
i = i - 1
If (i < 0) Then
    valOutput(0) = B0
    valOutput(1) = B1
    valOutput(2) = B2
    valOutput(3) = B3
    valOutput(4) = B4
    valOutput(5) = B5
    valOutput(6) = B6
    valOutput(7) = B7
    IsTrue = False
    Exit Sub
    GoTo selesaisub
End If

B0 = B0 Xor f(B7 Xor Key(i) Xor ex)
i = i - 1

B3 = B3 Xor f(B6 Xor Key(i) Xor ex)
i = i - 1

B2 = B2 Xor f(B4 Xor B5)

B1 = B1 Xor f(B4 Xor Key(i) Xor ex)
i = i - 1
selesaisub:
Wend
End Sub

```

MODULE CRC32 (CRC32.bas)

```

Private CRC32Table(255) As Long
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 = &HFFFFFFF
    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 &HFFFFFFF
    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, &HDBB9C9D6, &HACBCF940, _
        &H32D86CE3, &H45DF5C75, &HDCD60DCF, &HABD13D59, _
        &H26D930AC, &H51DE003A, &HC8D75180, &HBFDD06116, _
        &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, &HFBDD44C65)

```

```

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, &H8EBEFFF9, &H17B7BE43, &H60B08ED5)
For i = iOffset To iOffset + nLen - 1
CRC32Table(i) = vntA(i - iOffset)
Next
iOffset = iOffset + nLen
vntA = Array( _
&HD6D6A3E8, &HA1D1937E, &H38D8C2C4, &H4FDF252, _
&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

```

FORM BANGKITKAN KUNCI (FrmSign)

```

Private Sub CmdRandom_Click(Index As Integer)
Dim p As String, g As String
Dim privat As String, publik As String

If Index = 0 Then
' index = 0 : random digit
txtDigit = (Int(Rnd * 99) + 1)
ElseIf Index = 1 Then
' index = 1 : random digit
txtP = EG_BangkitkanP(txtDigit)
ElseIf Index = 2 Then
' index = 2 : bangkitkan kunci
If LI_IsPrime(txtP) = True Then
Old_P_For_GXY = txtP
EG_BikinKunci txtDigit, txtP, g, privat, publik
txtG = g
txtPrivat = privat
txtPublik = publik
End If
End If
End Sub

Private Sub cmdSimpanKunci_Click()
Dim tmpStr As String, tmpStr2 As String
Dim IsiFile1 As String, IsiFile2 As String
Dim NamaFile1 As String, NamaFile2 As String
Dim HndFile As Integer
Dim tmpBoolean As Boolean
Dim MD5Digest As MD5
Set MD5Digest = New MD5

NamaFile1 = ""
NamaFile2 = ""
If txtP <> "" And txtG <> "" And _
txtPrivat <> "" And txtPublik <> "" Then

' bentuk string fileprivat & periksa file
tmpStr = "Sign Privat" & vbCrLf & txtP & vbCrLf & txtG & vbCrLf & txtPrivat
tmpStr2 = MD5Digest.DigestStrToHexStr(tmpStr)
tmpStr2 = tmpStr2 & vbCrLf & tmpStr
IsiFile1 = tmpStr2
tmpBoolean = False
Do
CommonDialog1.FileName = ""
CommonDialog1.Filter = "Kunci Privat (*.pri) | *.pri"
CommonDialog1.ShowSave

```

```

tmpBoolean = True
If CommonDialog1.FileName <> "" Then
    If FileExist(CommonDialog1.FileName) = True Then
        res = MsgBox("File is exist. Replace?", vbYesNo + vbCritical, "File is exist ... ")
        If res <> vbYes Then tmpBoolean = False
    End If
End If
Loop Until tmpBoolean = True
If CommonDialog1.FileName <> "" Then
    NamaFile1 = CommonDialog1.FileName
    If Right(NamaFile1, 4) <> ".pri" Then _
        NamaFile1 = NamaFile1 & ".pri"
End If

If NamaFile1 <> "" Then

    ' bentuk string filepublik
    tmpStr = "Sign Publik" & vbCrLf & txtP & vbCrLf & txtG & vbCrLf & txtPublik
    tmpStr2 = MD5Digest.DigestStrToHexStr(tmpStr)
    tmpStr2 = tmpStr2 & vbCrLf & tmpStr
    IsiFile2 = tmpStr2
    tmpBoolean = False
    Do
        CommonDialog1.FileName = ""
        CommonDialog1.Filter = "Kunci Publik (*.pub) | *.pub"
        CommonDialog1.ShowSave
        tmpBoolean = True
        If CommonDialog1.FileName <> "" Then
            If FileExist(CommonDialog1.FileName) = True Then
                res = MsgBox("File is exist. Replace?", vbYesNo + vbCritical, "File is exist ... ")
                If res <> vbYes Then tmpBoolean = False
            End If
        End If
    Loop Until tmpBoolean = True
    If CommonDialog1.FileName <> "" Then
        NamaFile2 = CommonDialog1.FileName
        If Right(NamaFile2, 4) <> ".pub" Then _
            NamaFile2 = NamaFile2 & ".pub"
        End If
    End If
End If

If NamaFile1 <> "" And NamaFile2 <> "" Then
    Close All
    HndFile = FreeFile
    Open NamaFile1 For Output As #HndFile
    Print #HndFile, IsiFile1
    Close #HndFile
    Open NamaFile2 For Output As #HndFile
    Print #HndFile, IsiFile2
    Close #HndFile
    MsgBox "File Kunci sudah disimpan.", vbOKOnly + vbInformation, "Save..."
End If
End If
End Sub

Private Sub Form_Load()
    APP_INIT
End Sub

Private Sub mnuabout1_Click()

```



```

FrmAbout.Left = FrmSign.Left
FrmAbout.Top = FrmSign.Top
FrmAbout.Show
Unload Me
End Sub

```

```

Private Sub mnudekrip_Click()
FrmDecrypt.Left = FrmSign.Left
FrmDecrypt.Top = FrmSign.Top
FrmDecrypt.Show
Unload Me
End Sub

```

```

Private Sub mnuekstraksi_Click()
frm_Ekstraksi.Left = FrmSign.Left
frm_Ekstraksi.Top = FrmSign.Top
frm_Ekstraksi.Show
Unload FrmSign
End Sub

```

```

Private Sub mnuenkrip_Click()
FrmEncrypt.Left = FrmSign.Left
FrmEncrypt.Top = FrmSign.Top
FrmEncrypt.Show
Unload Me
End Sub

```

```

Private Sub mnuotentikasi_Click()
FrmOtentikasi.Left = FrmSign.Left
FrmOtentikasi.Top = FrmSign.Top
FrmOtentikasi.Show
Unload FrmSign
End Sub

```

```

Private Sub mnutttd_Click()
Frm_ttd_file.Left = FrmSign.Left
Frm_ttd_file.Top = FrmSign.Top
Frm_ttd_file.Show
Unload FrmSign
End Sub

```

FORM TANDA TANGAN FILE (frm_ttd_file)

```

Private Sub cmdBeritTD_BrowseFile1_Click()
txtBeritTD_ValidFile = 0
CommonDialog1.FileName = ""
CommonDialog1.Filter = "All Files (*.*) | *.*"
CommonDialog1.ShowOpen
If CommonDialog1.FileName <> "" Then
txtBeritTD_File1 = CommonDialog1.FileName
txtBeritTD_ValidFile = 1
End If
End Sub
Private Sub cmdBeritTD_BrowseFile2_Click()
txtBeritTD_ValidPrivat = 0
CommonDialog1.FileName = ""
CommonDialog1.Filter = "Kunci Privat (*.pri) | *.pri"
CommonDialog1.ShowOpen

```

```

If CommonDialog1.FileName <> "" Then
    txtBeritTD_File2 = CommonDialog1.FileName
    txtBeritTD_ValidPrivat = 1
End If
End Sub
Private Sub cmdBeritTD_OK_Click()

    Dim NamaFileInput As String
    Dim NamaFileOutput As String
    Dim NamaFilePrivat As String
    Dim tmpStr As String * 512
    Dim valFile As String, valP As String, valG As String, valX As String, valA As String, valB As
String

    ' minta file output
    CommonDialog1.FileName = ""
    CommonDialog1.Filter = " El Gamal Digital Signature (*.ELg) | *.ELg"
    CommonDialog1.ShowSave
    If CommonDialog1.FileName = "" Then GoTo lblSelesai
    NamaFileOutput = CommonDialog1.FileName
    If Right(NamaFileOutput, 4) <> ".ELg" Then NamaFileOutput = NamaFileOutput & ".ELg"
    If FileExist(NamaFileOutput) = True Then
        res = MsgBox("File is Exist. Replace?", vbOKCancel + vbCritical, "Exist...")
        If res = vbCancel Then GoTo lblSelesai
    End If

    NamaFileInput = txtBeritTD_File1
    NamaFilePrivat = txtBeritTD_File2

    If txtBeritTD_ValidFile = 1 And txtBeritTD_ValidPrivat = 1 Then
        ' memberikan tanda tangan
        HndFile = FreeFile
        Open NamaFilePrivat For Input As #1
        Line Input #1, tmpStr1
        Line Input #1, tmpStr2
        Line Input #1, valP
        Line Input #1, valG
        Line Input #1, valX
        Close HndFile
        FileCopy NamaFileInput, NamaFileOutput
        EG_Signing NamaFileOutput, valP, valG, valX, valA, valB
        Open NamaFileOutput For Append As #HndFile
        Print #HndFile, vbCrLf & vbCrLf & "---BEGIN---"
        Print #HndFile, LI_DecToHex(valA)
        Print #HndFile, LI_DecToHex(valB)
        Print #HndFile, "---END---"
        Close #HndFile

        MsgBox ("File sudah ditandatangani !! ")

    End If
lblSelesai:
End Sub

Private Sub mnuabout1_Click()
    FrmAbout.Left = Frm_ttd_file.Left
    FrmAbout.Top = Frm_ttd_file.Top
    FrmAbout.Show
    Unload Me
End Sub

```

```
Private Sub mnuexit_Click()
End
End Sub
```

```
Private Sub mnudekrip_Click()
Frmmain.Left = Frm_ttd_file.Left
Frmmain.Top = Frm_ttd_file.Top
FrmDecrypt.Show
Unload Frm_ttd_file
End Sub
```

```
Private Sub mnuekstraksi_Click()
frm_Ekstraksi.Left = Frm_ttd_file.Left
frm_Ekstraksi.Top = Frm_ttd_file.Top
frm_Ekstraksi.Show
Unload Frm_ttd_file
End Sub
```

```
Private Sub mnuenkrip_Click()
FrmEncrypt.Left = Frm_ttd_file.Left
FrmEncrypt.Top = Frm_ttd_file.Top
FrmEncrypt.Show
Unload Frm_ttd_file
End Sub
```

```
Private Sub mnukunci_Click()
FrmSign.Left = Frm_ttd_file.Left
FrmSign.Top = Frm_ttd_file.Top
FrmSign.Show
Unload Me
End Sub
```

```
Private Sub mnuotentikasi_Click()
FrmOtentikasi.Left = Frm_ttd_file.Left
FrmOtentikasi.Top = Frm_ttd_file.Top
FrmOtentikasi.Show
Unload Frm_ttd_file
End Sub
```

FORM EKSTRAKSI TANDA TANGAN (Frm_Ekstrasi)

```
Private Sub cmdEkstraksiDoc_Browse_Click()
CommonDialog1.FileName = ""
CommonDialog1.Filter = "eL Gamal Digital Signature (*.ELg) | *.ELg"
CommonDialog1.ShowOpen
If CommonDialog1.FileName <> "" Then txtEkstraksiDok_File = CommonDialog1.FileName
End Sub
```

```
Private Sub cmdEkstraksiDoc_Ekstrak_Click()
Dim tmpStr As String
Dim tmpLong As Long
Dim posDS As Long
Dim isFoundDS As Boolean
Dim NamaFileEkstrak As String
CommonDialog1.FileName = ""
CommonDialog1.Filter = "All Files (*.*) | *.*"
CommonDialog1.ShowOpen
If CommonDialog1.FileName <> "" Then
```

```

    NamaFileEkstrak = CommonDialog1.FileName
    If FileExist(NamaFileEkstrak) = True Then
        res = MsgBox("File sudah ada. Timpa?", vbOKCancel + vbCritical, "File..")
        If res = vbCancel Then GoTo lblSelesai
    End If
Else
    GoTo lblSelesai
End If

tmpStr = ReadFileText(txtEkstraksiDok_File)
isFoundDS = False
tmpLong = 0
Do
    tmpLong = InStr(tmpLong + 1, tmpStr, vbCrLf & vbCrLf & "---BEGIN---")
    If tmpLong <> 0 Then
        isFoundDS = True
        posDS = tmpLong
    End If
Loop Until tmpLong = 0

If posDS = 0 Then
    MsgBox "Data Tanda Tangan tidak valid."
Else
    HndFile = FreeFile
    If FileExist(NamaFileEkstrak) = True Then Kill (NamaFileEkstrak)
    Open NamaFileEkstrak For Binary As #HndFile
    tmpStr = Left(tmpStr, posDS - 1)
    Put #HndFile, , tmpStr
    Close #HndFile
    MsgBox ("Dokumen sudah diekstrak.")
End

End If
lblSelesai:
End Sub

Private Sub mnuabout1_Click()
    FrmAbout.Left = frm_Ekstraksi.Left
    FrmAbout.Top = frm_Ekstraksi.Top
    FrmAbout.Show
    Unload Me
End Sub

Private Sub mnudekrip_Click()
    FrmDecrypt.Left = frm_Ekstraksi.Left
    FrmDecrypt.Top = frm_Ekstraksi.Top
    FrmDecrypt.Show
    Unload frm_Ekstraksi
End Sub

Private Sub mnuenkrip_Click()
    FrmEncrypt.Left = frm_Ekstraksi.Left
    FrmEncrypt.Top = frm_Ekstraksi.Top
    FrmEncrypt.Show
    Unload frm_Ekstraksi
End Sub

Private Sub mnuexit_Click()
    End
End Sub

```

```

Private Sub mnukunci_Click()
FrmSign.Left = frm_Ekstraksi.Left
FrmSign.Top = frm_Ekstraksi.Top
FrmSign.Show
Unload frm_Ekstraksi
End Sub

Private Sub mnuotentikasi_Click()
FrmOtentikasi.Left = frm_Ekstraksi.Left
FrmOtentikasi.Top = frm_Ekstraksi.Top
FrmOtentikasi.Show
Unload frm_Ekstraksi
End Sub

Private Sub mnuttdd_Click()
Frm_ttd_file.Left = frm_Ekstraksi.Left
Frm_ttd_file.Top = frm_Ekstraksi.Top
Frm_ttd_file.Show
Unload frm_Ekstraksi
End Sub

```

FORM OTENTIKASI (FrmOtentikasi)

```

Private Sub cmdUjiTTD_BrowseFile1_Click()
    CommonDialog1.FileName = ""
    CommonDialog1.Filter = "ElGamal (*.ELg) | *.ELg"
    CommonDialog1.ShowOpen
    If CommonDialog1.FileName <> "" Then txtUjiTTD_File1 = CommonDialog1.FileName
End Sub

Private Sub cmdUjiTTD_BrowseFile2_Click()
    CommonDialog1.FileName = ""
    CommonDialog1.Filter = "Kunci Publik (*.pub) | *.pub"
    CommonDialog1.ShowOpen
    If CommonDialog1.FileName <> "" Then txtUjiTTD_File2 = CommonDialog1.FileName
End Sub

Private Sub cmdUjiTTD_Uji_Click()

    Dim tmpStr As String
    Dim tmpLong As Long
    Dim posDS As Long
    Dim isFoundDS As Boolean
    Dim tmpTTD As String, tmpDok As String

    Dim valA As String, valB As String, valP As String, valG As String, valY As String

    tmpStr = ReadFileText(txtUjiTTD_File1)
    isFoundDS = False
    tmpLong = 0
    Do
        tmpLong = InStr(tmpLong + 1, tmpStr, vbCrLf & vbCrLf & "---BEGIN---")
        If tmpLong <> 0 Then
            isFoundDS = True
            posDS = tmpLong
        End If
    Loop Until tmpLong = 0

```

```

If posDS = 0 Then
    MsgBox "Data Tanda Tangan tidak valid."
    GoTo lblSelesai
Else
    tmpDok = Left(tmpStr, posDS - 1)
    tmpTTD = Mid(tmpStr, posDS + 4, Len(tmpStr) - posDS)

    ' print signature ke file temp
    If FileExist(APP_PATH & "\temp.tmp") Then Kill (APP_PATH & "\temp.tmp")
    HndFile = FreeFile
    Open APP_PATH & "\temp.tmp" For Binary As #HndFile
    Put #HndFile, , tmpTTD
    Close #HndFile
    Open APP_PATH & "\temp.tmp" For Input As #HndFile
    Line Input #1, tmpStr
    Line Input #1, valA
    Line Input #1, valB
    Close #HndFile

    ' print tmpdok ke temp
    If FileExist(APP_PATH & "\temp.tmp") Then Kill (APP_PATH & "\temp.tmp")
    HndFile = FreeFile
    Open APP_PATH & "\temp.tmp" For Binary As #HndFile
    Put #HndFile, , tmpDok
    Close #HndFile

    ' baca kunci publik
    Open txtUjiTTD_File2 For Input As HndFile
    Line Input #1, tmpStr1
    Line Input #1, tmpStr2
    Line Input #1, valP
    Line Input #1, valG
    Line Input #1, valY
    Close HndFile

    If EG_Verify(APP_PATH & "\temp.tmp", valP, valG, valY, LI_HexToDec(valA),
LI_HexToDec(valB)) Then
        MsgBox "File Otentik."
    Else
        MsgBox "File tidak otentik."
    End If
End If
'End If
lblSelesai:
End Sub

Private Sub mnuabout1_Click()
    FrmAbout.Left = FrmOtentikasi.Left
    FrmAbout.Top = FrmOtentikasi.Top
    FrmAbout.Show
    Unload Me
End Sub

Private Sub mnudekrip_Click()
    FrmDecrypt.Left = FrmOtentikasi.Left
    FrmDecrypt.Top = FrmOtentikasi.Top
    FrmDecrypt.Show
    Unload FrmOtentikasi
End Sub

```

```

Private Sub mnuekstraksi_Click()
frm_Ekstraksi.Left = FrmOtentikasi.Left
frm_Ekstraksi.Top = FrmOtentikasi.Top
frm_Ekstraksi.Show
Unload FrmOtentikasi
End Sub

```

```

Private Sub mnuenkrip_Click()
FrmEncrypt.Left = FrmOtentikasi.Left
FrmEncrypt.Top = FrmOtentikasi.Top
FrmEncrypt.Show
Unload FrmOtentikasi
End Sub

```

```

Private Sub mnuexit_Click()
End
End Sub

```

```

Private Sub mnukunci_Click()
FrmSign.Left = FrmOtentikasi.Left
FrmSign.Top = FrmOtentikasi.Top
FrmSign.Show
Unload FrmOtentikasi
End Sub

```

```

Private Sub mnuttd_Click()
Frm_ttd_file.Left = FrmOtentikasi.Left
Frm_ttd_file.Top = FrmOtentikasi.Top
Frm_ttd_file.Show
Unload FrmOtentikasi
End Sub

```

MODULE ELGAMAL (Elgamal.bas)

```

Function EG_BangkitkanP(numdigit As Integer) As String
' hitung p
Randomize Timer
EG_BangkitkanP = LI_PrimeRandom(numdigit)
End Function
Public Sub EG_BikinKunci( _
valdigit As Integer, _
valP As String, _
ByRef valG As String, _
ByRef valX As String, _
ByRef valY As String)

Dim p As String, g As String, x As String, p_min_1 As String
Dim k As String, k_invers As String
Dim i As Long
Dim strTmp$

p = valP
' hitung p-1
p_min_1 = LI_Subt(p, "1")

' hitung g <= p-2
Randomize Timer
Do
intTmp = Int(Rnd * Len(p))

```

```

    If intTmp = 0 Then intTmp = 1
    g = ""
    For i = 1 To intTmp
        g = g & Int(Rnd * 10)
        If i = 1 And g = "0" Then g = "5"
    Next
    Loop Until LI_Compare(g, p_min_1) = -1

' hitung x <= p-2
Randomize Timer
Do
    intTmp = Int(Rnd * Len(p))
    If intTmp = 0 Then intTmp = 1
    x = ""
    For i = 1 To intTmp
        x = x & Int(Rnd * 10)
        If i = 1 And x = "0" Then x = "5"
    Next
    Loop Until LI_Compare(x, p_min_1) = -1

' hitung y = g^x mod p
y = LI_ModPwr(g, x, p)

' return value
valG = g
valX = x
valY = y
End Sub

Public Sub EG_Signing( _
    valFile As String, _
    valP As String, _
    valG As String, _
    valX As String, _
    ByRef valA As String, _
    ByRef valB As String)

    Dim tmpStr As String
    Dim p As String, g As String, x As String, p_min_1 As String
    Dim a_sig As String, b_sig As String
    Dim k As String, k_invers As String
    Dim i As Long
    Dim HashMD5 As String

    Dim MD5Digest As MD5
    Set MD5Digest = New MD5

    p = valP
    g = valG
    x = valX

    ' p_min_1
    p_min_1 = LI_Subt(p, "1")

    ' hitung k <= p-2 dan gcd(k,p-1) = 1
    Randomize Timer
    Do
        intTmp = Int(Rnd * Len(p))
        If intTmp = 0 Then intTmp = 1
        k = ""

```



```

    For i = 1 To intTmp
        k = k & Int(Rnd * 10)
        If i = 1 And k = "0" Then k = "5"
    Next
    Loop Until (LI_Compare(k, p_min_1) = -1) And (LI_Gcd(k, p_min_1) = 1)

    ' hitung a_sig = g^k mod p
    a_sig = LI_ModPwr(g, k, p)
    valA = a_sig

    ' hitung k invers
    k_invers = LI_ExtendEuclidean_GetInvers(k, p_min_1)

    ' Hash pesan
    HashMD5 = UCase(MD5Digest.DigestFileToHexStr(valFile))

    ' hitung b = k_inv { h(m) - privat.a_sig } mod (p-1)
    tmpStr = LI_Mult(valX, a_sig)
    'tmpStr = LI_Subt(Left(LI_HexToDec(HashMD5), 2), tmpStr)
    tmpStr = LI_Subt(LI_HexToDec(HashMD5), tmpStr)
    tmpStr = LI_Mult(k_invers, tmpStr)
    b_sig = LI_ModPwr(tmpStr, "1", p_min_1)
    valB = b_sig
End Sub

Public Function EG_Verify( _
    valFile As String, _
    valP As String, _
    valG As String, _
    valY As String, _
    valA As String, _
    valB As String) As Boolean

    Dim tmpStr As String
    Dim Val1 As String, Val2 As String
    Dim p As String, g As String, x As String, p_min_1 As String
    Dim a_sig As String, b_sig As String
    Dim k As String, k_invers As String
    Dim i As Long
    Dim HashMD5 As String
    Dim PanjangBlokProsesMD5 As String

    Dim MD5Digest As MD5
    Set MD5Digest = New MD5

    HashMD5 = UCase(MD5Digest.DigestFileToHexStr(valFile))

    ' val1 = (y^a x a^b) mod p
    ' val1 = [ (y^a mod p) (a^b mod p) ] mod p
    tmpStr = LI_ModPwr(valY, valA, valP)
    Val1 = LI_ModPwr(valA, valB, valP)
    Val1 = LI_Mult(tmpStr, Val1)
    Val1 = LI_ModPwr(Val1, "1", valP)

    Val2 = LI_ModPwr(valG, LI_HexToDec(HashMD5), valP)

    If Val1 = Val2 Then
        EG_Verify = True
    Else
        EG_Verify = False
    End If
End Function

```

```
End If
End Function
```

MODULE LARGE INTEGER (LargeInteger.bas)

```
Option Base 1
```

```
Const LI_TMP = 99
Const LI_TMP2 = 98
```

```
Const LI0 = 0, LI1 = 1, LI2 = 2, LI3 = 3, LI4 = 4
```

```
' Inisialisasi
Sub LI_Init()
    InitBigNum 100, ""
    It = 10000 ' maxindex
End Sub
```

```
' Large Integer and String
```

```
Sub LI_StrtoLI(p As Long, s As String)
    ' return p
    Readst p, s
End Sub
```

```
Function LI_LtoStr(p As Long) As String
    Printn p, g$, "", 0
    LI_LtoStr = DataPF$
End Function
```

```
' comparison
```

```
Function LI_Compare(p As String, q As String)
    LI_StrtoLI LI0, p
    LI_StrtoLI LI1, q
    LI_Compare = Cmp(LI0, LI1)
End Function
```

```
' perhitungan aritmatika
```

```
Function LI_Add(p As String, q As String) As String
    LI_StrtoLI LI0, p
    LI_StrtoLI LI1, q
    Add LI0, LI1
    LI_Add = LI_LtoStr(LI0)
End Function
```

```
Function LI_Subt(p As String, q As String) As String
    LI_StrtoLI LI0, p
    LI_StrtoLI LI1, q
    Subt LI0, LI1
    LI_Subt = LI_LtoStr(LI0)
End Function
```

```
Function LI_Mult(p As String, q As String) As String
    LI_StrtoLI LI0, p
    LI_StrtoLI LI1, q
    Mult LI0, LI1, LI_TMP
    LI_Mult = LI_LtoStr(LI0)
End Function
```

```

Function LI_Div(p As String, q As String) As String
    LI_StrtoLI LI0, p
    LI_StrtoLI LI1, q
    Divd LI0, LI1, LI_TMP
    LI_Div = LI_LtoStr(LI_TMP)
End Function
' modulo

Function LI_ModPwr(p0 As String, p1 As String, p2 As String) As String
    ' presult = (p0 ^ p1) mod p2
    LI_StrtoLI LI0, p0
    LI_StrtoLI LI1, p1
    LI_StrtoLI LI2, p2
    Modpwr LI0, LI1, LI2
    LI_ModPwr = LI_LtoStr(LI0)
End Function

' lain2

Function LI_PrimeRandom(numdigit As Integer) As String
    Dim NotPrimeList(7) As String
    NotPrimeList(1) = "2047"
    NotPrimeList(2) = "1373653"
    NotPrimeList(3) = "25326001"
    NotPrimeList(4) = "3215031751"
    NotPrimeList(5) = "2152302898747"
    NotPrimeList(6) = "3474749660383"
    NotPrimeList(7) = "341550071728321"
    Randomize Timer
    Do
        isContinue = True
        strTmp$ = ""
        Do
            tmpint = Int(Rnd * 10)
            If (Len(strTmp$) = (numdigit - 1)) And (tmpint Mod 2 = 0) Then
            ElseIf Len(strTmp$) = 0 And tmpint = 0 Then
            Else
                strTmp$ = strTmp$ & tmpint
            End If
        Loop Until Len(strTmp$) >= numdigit
        strTmp$ = Left(strTmp$, numdigit)
        For i = 1 To 7
            If strTmp$ = NotPrimeList(i) Then isContinue = False
        Next
        If isContinue = True Then
            LI_StrtoLI LI0, strTmp$
            If IsPPrm(LI0, LI_TMP, 8) Then isPrime = True
        End If
        Loop Until isPrime = True
        LI_PrimeRandom = LI_LtoStr(LI0)
    End Function

Function LI_IsPrime(p As String) As Boolean
    Dim NotPrimeList(7) As String
    NotPrimeList(1) = "2047"
    NotPrimeList(2) = "1373653"
    NotPrimeList(3) = "25326001"
    NotPrimeList(4) = "3215031751"
    NotPrimeList(5) = "2152302898747"

```

```

NotPrimeList(6) = "3474749660383"
NotPrimeList(7) = "341550071728321"
isContinue = True
isPrime = False

For i = 1 To 7
    If strTmp$ = NotPrimeList(i) Then
        isPrime = False
        isContinue = False
        Exit For
    End If
Next

If isContinue = True Then
    LI_StrtoLI LI0, p
    If IsPPrm(LI0, LI_TMP, 8) Then isPrime = True
End If
LI_IsPrime = isPrime
End Function

Function LI_Gcd(p As String, q As String) As String
    LI_StrtoLI LI0, p
    LI_StrtoLI LI1, q
    Euclid LI0, LI1, LI2
    LI_Gcd = LI_LItoStr(LI2)
End Function

Function LI_ExtendEuclidean_GetInvers(p As String, q As String) As String
    LI_StrtoLI LI0, p
    LI_StrtoLI LI1, q
    Euclid LI0, LI1, LI2
    LI_ExtendEuclidean_GetInvers = LI_LItoStr(LI0)
End Function

Function LI_DecToHex(valP As String) As String
    Dim tmpStr As String
    Dim tmpResult As String
    Dim p As String
    p = valP
    Do
        tmpStr = LI_ModPwr(p, "1", 16)
        If tmpStr = "0" Or tmpStr = "1" Or tmpStr = "2" Or tmpStr = "3" Or tmpStr = "4" _
            Or tmpStr = "5" Or tmpStr = "6" Or tmpStr = "7" Or tmpStr = "8" _
            Or tmpStr = "9" Then
            tmpResult = tmpStr & tmpResult
        ElseIf tmpStr = "10" Then
            tmpResult = "A" & tmpResult
        ElseIf tmpStr = "11" Then
            tmpResult = "B" & tmpResult
        ElseIf tmpStr = "12" Then
            tmpResult = "C" & tmpResult
        ElseIf tmpStr = "13" Then
            tmpResult = "D" & tmpResult
        ElseIf tmpStr = "14" Then
            tmpResult = "E" & tmpResult
        ElseIf tmpStr = "15" Then
            tmpResult = "F" & tmpResult
        End If
        p = LI_Div(p, "16")
    Loop Until p = "0"
    LI_DecToHex = tmpResult

```

```

End Function
Function LI_HexToDec(valP As String) As String
    Dim tmpStr As String
    Dim tmpPangkat As String
    Dim tmpKali As String
    Dim tmpResult As String
    Dim p As String
    p = valP
    p = UCase(p)
    tmpResult = "0"
    For i = 1 To Len(p)
        tmpPangkat = LI_ModPwr("16", (i - 1), "0")
        tmpStr = Mid(p, Len(p) - i + 1, 1)
        If tmpStr = "A" Then
            tmpStr = "10"
        ElseIf tmpStr = "B" Then
            tmpStr = "11"
        ElseIf tmpStr = "C" Then
            tmpStr = "12"
        ElseIf tmpStr = "D" Then
            tmpStr = "13"
        ElseIf tmpStr = "E" Then
            tmpStr = "14"
        ElseIf tmpStr = "F" Then
            tmpStr = "15"
        End If
        tmpKali = LI_Mult(tmpStr, tmpPangkat)
        tmpResult = LI_Add(tmpResult, tmpKali)
    Next
    LI_HexToDec = tmpResult
End Function

```

MODULE BigNumVB (BigNumVB.bas)

```

DefInt A-B, D-Z
DefLng C
' *****
'Subject: Include file for large-integer arithmetic Basic library.
'Author : Sjoerd.J.Schaper - vspickelen@zonnet.nl
'Date : 08-05-2005
'Code : Visual Basic-callable FreeBasic 0.14b compiled dll.
'`[...] there is a bound to the number of symbols which the computer
'can observe at one moment. If he wishes to observe more, he must use
'successive operations.' Alan Turing, On computable numbers [...], 1936.
'
' *****
'
' Global constants
' *****
Global Const ui = 171, uj = 5119 ' number array n(ui, uj) upper bounds
Global Const LB = 15, MB = 32768, MH = 16384 ' binary storage base
Global Const M1 = 32767, M2 = 1073709056 ' masks MB - 1, (MB - 1)* MB
Global Const L10 = 0.221461873 ' 1/ 10log(MB)
Global Const LD = 8, MD = 100000000 ' decimal in/out base, MY < MB
Global Const MY = 10000&, Fx = 2147483629 ' fixed limit
Global Const t0 = -1, t1 = -2 ' pointers to swap-columns
Global Const t2 = -3, t3 = -4
'Maximal number length = uj / L10, approximately 9 * uj / 2.
'
' *****
'
' Initialization and assignment

```

```

' *****
'The LargeInt class is initialized with an 'Init(size%, name$)' call at
'module level. You have to declare constant indices for referencing the
'large integers you're planning to use ('CONST p = 0, q = 1, ..'), so that
'each number is associated with its own distinct pointer. These pointers
'are passed to the procedures. Don't call with same pointers ('Squ p, p'),
'and beware of duplicate references, for they will crash your program.
,

'In the following prototypes,
'the letters p, q, r, d, m denote 32-bit pointers to large integers;
'variables a, c, k, sw are 32-bit signed integers;
'g is a number string; f is any string.
,

Declare Function InitBigNum Lib "BigNumVB" Alias "Init@8" (ByVal k As Long, _
ByRef f As String) As Integer
'Initialize the number-arrays and open the primetable.
'To be called with dummy k and f, return value = -1.
,

Declare Sub Letf Lib "BigNumVB" Alias "Lef@8" (ByVal p As Long, _
ByVal c As Long)
'Assign long integer c to large integer p.
,

Declare Sub Pwr10 Lib "BigNumVB" Alias "Pwr10@8" (ByVal p As Long, _
ByVal k As Long)
'Assign 10^k to large integer p.
,

Declare Sub Pwr2 Lib "BigNumVB" Alias "Pwr2@8" (ByVal p As Long, _
ByVal k As Long)
'Assign 2^k to large integer p.
,

Declare Sub Rndf Lib "BigNumVB" Alias "Rndf@8" (ByVal p As Long, _
ByVal k As Long)
'Assign to p a random positive number with length Abs(k).
,

Declare Sub Term Lib "BigNumVB" Alias "Term@0" ()
'Close the primetable.
,

' *****
'
' Conversion functions
' *****
Declare Function DecLSB Lib "BigNumVB" Alias "DecLSB@4" (ByVal p As Long) _
As Integer
'Return the four rightmost digits of large integer p.
,

Declare Function Decf Lib "BigNumVB" Alias "Decf@4" (ByVal p As Long) _
As Integer
'Convert large integer p to base MD in array ln(),
'the bytelength of p is returned. Call before CnvSt.
,

Declare Sub CnvSt Lib "BigNumVB" Alias "CnvSt@4" (ByRef g As String)
'Convert base MD number to string. First call k = Decf(p),
'create stringbuffer g = STRING$(k, "0") and pass to CnvSt.
,

Declare Sub Ratdec Lib "BigNumVB" Alias "Ratdec@12" (ByRef g As String, _
ByVal p As Long, ByVal q As Long)
'Return the decimal expansion of rational p / q in [0, 1) in string g.
,

Declare Sub Readst Lib "BigNumVB" Alias "Readst@8" (ByVal p As Long, _
ByRef g As String)
'Convert decimal number string g to large integer p.

```

```

'Except for a prefixed minus sign, there's no check
'on non-digit characters in the string.
'
' *****
'
'           Obtaining output
' *****
'Use the printfunctions in "PrintFun.bas" or write your own.
'
' *****
'
'           Basic arithmetic functions
' *****
Declare Sub Add Lib "BigNumVB" Alias "Add@8" (ByVal p As Long, _
ByVal q As Long)
'Add p and q, the result is in p.
'
Declare Sub Chs Lib "BigNumVB" Alias "Chs@4" (ByVal p As Long)
'Change the sign of number p.
'
Declare Sub Dcr Lib "BigNumVB" Alias "Dcr@8" (ByVal p As Long, _
ByVal a As Long)
'Decrease large integer p with small positive a.
'
Declare Sub Divd Lib "BigNumVB" Alias "Divd@12" (ByVal p As Long, _
ByVal d As Long, ByVal q As Long)
'Divide p by d. The quotient is in q, the remainder in p.
'
Declare Sub Inc Lib "BigNumVB" Alias "Inc@8" (ByVal p As Long, _
ByVal a As Long)
'Increase large integer p with small positive a.
'
Declare Sub Mult Lib "BigNumVB" Alias "Mult@12" (ByVal p As Long, _
ByVal q As Long, ByVal r As Long)
'Multiply p and q. The result is in p by default,
' 'Swp p, r' puts it in r.
'
Declare Sub Squ Lib "BigNumVB" Alias "Squ@8" (ByVal p As Long, _
ByVal q As Long)
'Return the square of p in q.
'
Declare Sub Subt Lib "BigNumVB" Alias "Subt@8" (ByVal p As Long, _
ByVal q As Long)
'Subtract q from p, the result is in p.
'To retain p, use 'Subt q, p: Chs q'.
'
' *****
'
'           Comparison and copying
' *****
Declare Function Cmp Lib "BigNumVB" Alias "Cmp@8" (ByVal p As Long, _
ByVal q As Long) As Integer
'Compare returns -1 if p < q, 0 if p = q, or 1 if p > q.
'
Declare Function Isf Lib "BigNumVB" Alias "Isf@8" (ByVal p As Long, _
ByVal a As Long) As Integer
'Boolean: check if large integer p equals one-word value a.
'
Declare Function Ismin1 Lib "BigNumVB" Alias "Ismin1@8" (ByVal p As Long, _
ByVal m As Long) As Integer
'Boolean: check if p = m - 1
'
Declare Sub Copyf Lib "BigNumVB" Alias "Copyf@8" (ByVal p As Long, _

```

```

ByVal q As Long)
'Copy number p to q.
'

Declare Sub Swp Lib "BigNumVB" Alias "Swp@8" (ByVal p As Long, _
ByVal q As Long)
'Exchange numbers p and q.
'

' *****
'
'           Bit manipulation
' *****

Declare Sub Andf Lib "BigNumVB" Alias "Andf@8" (ByVal p As Long, _
ByVal q As Long)
'Bitwise logical p AND q, the result is in p.
'

Declare Sub Lsft Lib "BigNumVB" Alias "Lsft@8" (ByVal p As Long, _
ByVal k As Long)
'Large integer p is shifted left by k bits.
'

Declare Sub Rsft Lib "BigNumVB" Alias "Rsft@8" (ByVal p As Long, _
ByVal k As Long)
'Large integer p is shifted right by k bits.
'

' *****
'
'           Modular arithmetic functions
' *****

Declare Sub Modbal Lib "BigNumVB" Alias "Modbal@8" (ByVal p As Long, _
ByVal m As Long)
'Balanced residue p modulo m:  $Abs(p) \leq m \setminus 2$ .
'

Declare Sub Moddiv Lib "BigNumVB" Alias "Moddiv@8" (ByVal p As Long, _
ByVal m As Long)
'Compute positive residue p modulo m.
'

Declare Sub Modmlt Lib "BigNumVB" Alias "Modmlt@12" (ByVal p As Long, _
ByVal q As Long, ByVal m As Long)
'Multiply p and q, then reduce modulo m. The result is in p.
'

Declare Sub Modpwr Lib "BigNumVB" Alias "Modpwr@12" (ByVal p As Long, _
ByVal q As Long, ByVal m As Long)
'Modular exponentiation: base p, exponent q, modulus m,
'the result is in p. This doubles as power function:
'first put 'Letf m, 0' to switch off modulo reduction.
'

Declare Sub Modsqu Lib "BigNumVB" Alias "Modsqu@8" (ByVal p As Long, _
ByVal m As Long)
'Square p, then reduce modulo m.
'

' *****
'
'           Number theoretic functions
' *****

Declare Sub Euclid Lib "BigNumVB" Alias "Euclid@12" (ByVal p As Long, _
ByVal q As Long, ByVal d As Long)
'The extended Euclidean algorithm computes
' $p^{-1} \bmod q$  in p,  $q / \gcd$  in q and  $\gcd(p, q)$  in d.
'

Declare Sub Fctl Lib "BigNumVB" Alias "Fctl@8" (ByVal p As Long, _
ByVal a As Long)
'Computes factorial  $a(a-1)\cdots 2\cdot 1$  in large integer p.
'

Declare Function IsPPrm Lib "BigNumVB" Alias "IsPPrm@12" (ByVal p As Long, _

```



```

ByVal d As Long, ByVal k As Long) As Integer
'Boolean: check if p is probably prime with the Miller-Rabin test.
'k is the number of witnesses, d returns a small divisor, if any.
'
Declare Function IsSqr Lib "BigNumVB" Alias "IsSqr@8" (ByVal p As Long, _
ByVal q As Long) As Integer
'Boolean: check if p is square. If true, the root is in q.
'
Declare Sub Isqrt Lib "BigNumVB" Alias "Isqrt@8" (ByVal p As Long, _
ByVal q As Long)
'Integer square root of p with Heron's algorithm. Result is in q.
'
Declare Function Kronec Lib "BigNumVB" Alias "Kronec@12" (ByVal p As Long, _
ByVal q As Long, ByVal d As Long) As Integer
'Kronecker's quadratic residuosity symbol (p/q) = 0, 1, or -1.
'Returns an odd gcd(p, q) in d, and 2 if it's even.
'
Declare Function Nxtprm Lib "BigNumVB" Alias "Nxtprm@4" (ByVal sw As Long) _
As Long
'Loop through primetable PrimFlgs.bin. Initialize with sw = 0
'and get the next prime with sw <> 0 for each successive call.
'
' *****
'               Direct access
' *****
Declare Function Gete Lib "BigNumVB" Alias "Gete@8" (ByVal p As Long, _
ByVal k As Long) As Integer
'Return array element k of number p. This is a base MB digit.
'
Declare Function Getl Lib "BigNumVB" Alias "Getl@4" (ByVal p As Long) _
As Integer
'Return the length of number p measured in array elements.
'
Declare Function Gets Lib "BigNumVB" Alias "Gets@4" (ByVal p As Long) _
As Integer
'Return the sign of number p, Gets = -1 if p < 0, 1 otherwise.
'
Declare Sub Sete Lib "BigNumVB" Alias "Sete@12" (ByVal p As Long, _
ByVal k As Long, ByVal a As Long)
'Set binary array element k of number p to Abs(a).
'
Declare Sub Setl Lib "BigNumVB" Alias "Setl@8" (ByVal p As Long, _
ByVal a As Long)
'Set the length of number array p to Abs(a).
'
Declare Sub Sets Lib "BigNumVB" Alias "Sets@8" (ByVal p As Long, _
ByVal a As Long)
'Set the sign of number p to Sgn(a + 0.5).
'
' *****
'               Internal functions
' *****
Declare Sub CpMem Lib "Kernel32" Alias "RtlMoveMemory" (dest As Any, _
source As Any, ByVal numBytes As Long)
'To emulate QB Cvl() and Mkl$() functions.
'
Declare Function ErrMsg Lib "User32" Alias "MessageBoxA" _
(ByVal hwnd As Long, ByVal lpText As String, ByVal lpCaption As String, _
ByVal wType As Long) As Integer
'To display an error message.

```

```

',
Declare Function Hp2 Lib "BigNumVB" Alias "Hp2@4" (ByVal p As Long) _
    As Integer
'Return 2^ (highest set bit in the leftmost word of p).
',
Declare Sub Lftj Lib "BigNumVB" Alias "Lftj@8" (ByVal p As Long, _
    ByVal k As Long)
'Left-justify large integer p on from array element k.
',
Declare Function PrimCeil Lib "BigNumVB" Alias "PrimCeil@0" () As Long
'Return the upper limit of primetable PrimFlgs.bin.

```

MODULE FILE (File.bas)

```

Public Function FileExist(asPath As String) As Boolean
    On Error GoTo lblerr
    HndFile = FreeFile
    Open asPath For Input As #HndFile
    Close #HndFile
    GoTo lblnoterr
    '
    'Checks for an existing File,
    'returns True or False
    'examples:
    'If FindFile(Text1.Text) Then Label1 = "YES"
    'If Not FindFile(Text1.Text) Then Label1 = "NO"

lblerr:
    Close #HndFile
    FileExist = False
    GoTo Selesai
lblnoterr:
    FileExist = True
Selesai:
End Function
Public Function TrimPath(ByVal asPath As String) As String

    If Len(asPath) = 0 Then Exit Function
    Dim x As Integer

    Do
        x = InStr(asPath, "\")
        If x = 0 Then Exit Do
        asPath = Right(asPath, Len(asPath) - x)
    Loop
    TrimPath = asPath
End Function

Public Function ReadFileText>NamaFile As String) As String
    Dim tmpStr As String * 512, tmpStr2 As String
    Dim HndFile As Integer

    tmpStr2 = ""
    HndFile = FreeFile
    Open>NamaFile For Binary As #HndFile
    While Len(tmpStr2) < FileLen>NamaFile)
        Get #HndFile, , tmpStr
        tmpStr2 = tmpStr2 & tmpStr
    Wend
    Close #HndFile

```

```

    ReadFileText = tmpStr2
End Function

```

MODULE OTHER (Other.bas)

```

Public APP_PATH As String

```

```

Public Sub APP_INIT()

```

```

    APP_PATH = CurDir
    LI_Init
End Sub

```

MODULE PRINTFUN (PrintFun.bas)

```

'Author : Sjoerd.J.Schaper - vspickelen@zonnet.nl

```

```

'Date : 03-16-2005

```

```

'Code : Visual Basic for Windows 5.0

```

```

Option Explicit

```

```

Public Const dbCrLf = vbCrLf + vbCrLf

```

```

Public DataPF As String

```

```

Public WrkD As String

```

```

Public Key As Boolean

```

```

Public Lognr As Integer

```

```

Public tim As Double

```

```

'Numerical <-> string conversion

```

```

Function Cvl(g As String) As Long

```

```

    Dim k As Integer

```

```

    k = Len(g)

```

```

    If k < 4 Then g = g + String$(4 - k, Chr$(0))

```

```

    CpMem Cvl, ByVal g, 4

```

```

End Function

```

```

Function Mkl(c As Long) As String

```

```

    Mkl = String$(4, Chr$(0))

```

```

    CpMem ByVal Mkl, c, 4

```

```

End Function

```

```

'Printfunctions

```

```

Sub Printf(ByVal f As String, ByVal g As String, _

```

```

    ByVal h As String, ByVal sw As Integer)

```

```

    Dim s As String

```

```

    Select Case sw

```

```

        Case 0

```

```

            DataPF = f & g & h

```

```

        Case 1

```

```

            DataPF = DataPF & f & g & h

```

```

        Case 2

```

```

            DataPF = DataPF & f & g & h & vbCrLf

```

```

        Case Else

```

```

            s = "[" & Len(g) & "]"

```

```

            DataPF = DataPF & f & g & h & s & vbCrLf

```

```

    End Select

```

```

End Sub

```

```

Sub Printn(ByVal p As Long, ByVal f As String, _

```

```

    ByVal h As String, ByVal sw As Integer)

```

```

    Dim k As Integer, g As String

```

```

    k = Decf(p)
    g = String$(k, "0")
    CnvSt g
    Printf f, g, h, sw
End Sub

Sub Printr(ByVal p As Long, ByVal q As Long, _
    ByVal r As Long, ByVal f As String, ByVal h As String)
Dim k As Integer, lx As Integer, g As String
    lx = Getl(r)
    Divd p, q, r: k = Decf(r)
    g = String$(k, "0"): CnvSt g
    If Isf(p, 0) Then
        Printf f, g, "", 0
    Else
        Printf f, g, "", 0
        g = String$(lx, "0")
        Ratdec g, p, q
        If Len(h) = 0 Then
            h = "[" & lx & "]"
        End If
        Printf ". ", g, h, 1
    End If
End Sub

Sub Prints(ByVal f As String, ByVal sw As Integer)
    Select Case sw
    Case 0
        DataPF = f
    Case 1
        DataPF = DataPF & f
    Case Else
        DataPF = DataPF & f & vbCrLf
    End Select
End Sub

```

MODULE MD5 (clsMD5.cls)

Option Explicit

```

' /*****
' * Copyright (C) 2000 by Robert Hubley.
' * All rights reserved.
' *
' * This software is provided ``AS IS" and any express or implied
' * warranties, including, but not limited to, the implied warranties of
' * merchantability and fitness for a particular purpose, are disclaimed.
' * In no event shall the authors be liable for any direct, indirect,
' * incidental, special, exemplary, or consequential damages (including, but
' * not limited to, procurement of substitute goods or services; loss of use,
' * data, or profits; or business interruption) however caused and on any
' * theory of liability, whether in contract, strict liability, or tort
' * (including negligence or otherwise) arising in any way out of the use of
' * this software, even if advised of the possibility of such damage.
' *
' *****/
'
' CLASS: MD5
'
' DESCRIPTION:

```

This is a class which encapsulates a set of MD5 Message Digest functions. MD5 algorithm produces a 128 bit digital fingerprint (signature) from an dataset of arbitrary length. For details see RFC 1321 (summarized below). This implementation is derived from the RSA Data Security, Inc. MD5 Message-Digest algorithm reference implementation (originally written in C)

AUTHOR:

Robert M. Hubley 12/1999

NOTES:

Network Working Group R. Rivest
Request for Comments: 1321 MIT Laboratory for Computer Science
and RSA Data Security, Inc.
April 1992

The MD5 Message-Digest Algorithm

Summary

This document describes the MD5 message-digest algorithm. The algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.

The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly.

The MD5 algorithm is an extension of the MD4 message-digest algorithm [1,2]. MD5 is slightly slower than MD4, but is more "conservative" in design. MD5 was designed because it was felt that MD4 was perhaps being adopted for use more quickly than justified by the existing critical review; because MD4 was designed to be exceptionally fast, it is "at the edge" in terms of risking successful cryptanalytic attack. MD5 backs off a bit, giving up a little in speed for a much greater likelihood of ultimate security. It incorporates some suggestions made by various reviewers, and contains additional optimizations. The MD5 algorithm is being placed in the public domain for review and possible adoption as a standard.

RFC Author:

Ronald L. Rivest
Massachusetts Institute of Technology
Laboratory for Computer Science
NE43 -324545 Technology Square
Cambridge, MA 02139-1986
Phone: (617) 253-5880
EMail: Rivest@theory.lcs.mit.edu

CHANGE HISTORY:

```

'
' 0.1.0 RMH 1999/12/29 Original version
'=
'= Class Constants
'=
Private Const OFFSET_4 = 4294967296#
Private Const MAXINT_4 = 2147483647

Private Const S11 = 7
Private Const S12 = 12
Private Const S13 = 17
Private Const S14 = 22
Private Const S21 = 5
Private Const S22 = 9
Private Const S23 = 14
Private Const S24 = 20
Private Const S31 = 4
Private Const S32 = 11
Private Const S33 = 16
Private Const S34 = 23
Private Const S41 = 6
Private Const S42 = 10
Private Const S43 = 15
Private Const S44 = 21

'=
'= Class Variables
'=
Private State(4) As Long
Private ByteCounter As Long
Private ByteBuffer(63) As Byte

'=
'= Class Properties
'=
Property Get RegisterA() As String
    RegisterA = State(1)
End Property

Property Get RegisterB() As String
    RegisterB = State(2)
End Property

Property Get RegisterC() As String
    RegisterC = State(3)
End Property

Property Get RegisterD() As String
    RegisterD = State(4)
End Property

'=
'= Class Functions
'=
'
' Function to quickly digest a file into a hex string

```

```

',
Public Function DigestFileToHexStr(FileName As String) As String
    Dim HndFile As Integer
    HndFile = FreeFile
    Open FileName For Binary Access Read As #HndFile
    MD5Init
    Do While Not EOF(HndFile)
        Get #HndFile, , ByteBuffer
        If Loc(HndFile) < LOF(HndFile) Then
            ByteCounter = ByteCounter + 64
            MD5Transform ByteBuffer
        End If
    Loop
    ByteCounter = ByteCounter + (LOF(HndFile) Mod 64)
    Close #HndFile
    MD5Final
    DigestFileToHexStr = GetValues
End Function

',
' Function to digest a text string and output the result as a string
' of hexadecimal characters.
',
Public Function DigestStrToHexStr(SourceString As String) As String
    MD5Init
    MD5Update Len(SourceString), StringToArray(SourceString)
    MD5Final
    DigestStrToHexStr = GetValues
End Function

',
' A utility function which converts a string into an array of
' bytes.
',
Private Function StringToArray(InString As String) As Byte()
    Dim i As Integer
    Dim bytBuffer() As Byte
    ReDim bytBuffer(Len(InString))
    For i = 0 To Len(InString) - 1
        bytBuffer(i) = Asc(Mid(InString, i + 1, 1))
    Next i
    StringToArray = bytBuffer
End Function

',
' Concatenate the four state vaules into one string
',
Public Function GetValues() As String
    GetValues = LongToString(State(1)) & LongToString(State(2)) & LongToString(State(3)) &
LongToString(State(4))
End Function

',
' Convert a Long to a Hex string
',
Private Function LongToString(Num As Long) As String
    Dim a As Byte
    Dim b As Byte
    Dim c As Byte
    Dim d As Byte

```

```

a = Num And &HFF&
If a < 16 Then
    LongToString = "0" & Hex(a)
Else
    LongToString = Hex(a)
End If

b = (Num And &HFF00&) \ 256
If b < 16 Then
    LongToString = LongToString & "0" & Hex(b)
Else
    LongToString = LongToString & Hex(b)
End If

c = (Num And &HFF0000) \ 65536
If c < 16 Then
    LongToString = LongToString & "0" & Hex(c)
Else
    LongToString = LongToString & Hex(c)
End If

If Num < 0 Then
    d = ((Num And &H7F000000) \ 16777216) Or &H80&
Else
    d = (Num And &HFF000000) \ 16777216
End If

If d < 16 Then
    LongToString = LongToString & "0" & Hex(d)
Else
    LongToString = LongToString & Hex(d)
End If

End Function

'
' Initialize the class
' This must be called before a digest calculation is started
'
Public Sub MD5Init()
    ByteCounter = 0
    State(1) = UnsignedToLong(1732584193#)
    State(2) = UnsignedToLong(4023233417#)
    State(3) = UnsignedToLong(2562383102#)
    State(4) = UnsignedToLong(271733878#)
End Sub

'
' MD5 Final
'
Public Sub MD5Final()
    Dim dblBits As Double

    Dim padding(72) As Byte
    Dim lngBytesBuffered As Long

    padding(0) = &H80

    dblBits = ByteCounter * 8

```



```

' Pad out
lngBytesBuffered = ByteCounter Mod 64
If lngBytesBuffered <= 56 Then
    MD5Update 56 - lngBytesBuffered, padding
Else
    MD5Update 120 - ByteCounter, padding
End If

padding(0) = UnsignedToLong(dblBits) And &HFF&
padding(1) = UnsignedToLong(dblBits) \ 256 And &HFF&
padding(2) = UnsignedToLong(dblBits) \ 65536 And &HFF&
padding(3) = UnsignedToLong(dblBits) \ 16777216 And &HFF&
padding(4) = 0
padding(5) = 0
padding(6) = 0
padding(7) = 0

MD5Update 8, padding
End Sub

' Break up input stream into 64 byte chunks
,
Public Sub MD5Update(InputLen As Long, InputBuffer() As Byte)
    Dim II As Integer
    Dim i As Integer
    Dim J As Integer
    Dim k As Integer
    Dim lngBufferedBytes As Long
    Dim lngBufferRemaining As Long
    Dim lngRem As Long

    lngBufferedBytes = ByteCounter Mod 64
    lngBufferRemaining = 64 - lngBufferedBytes
    ByteCounter = ByteCounter + InputLen
    ' Use up old buffer results first
    If InputLen >= lngBufferRemaining Then
        For II = 0 To lngBufferRemaining - 1
            ByteBuffer(lngBufferedBytes + II) = InputBuffer(II)
        Next II
        MD5Transform ByteBuffer

        lngRem = (InputLen) Mod 64
        ' The transfer is a multiple of 64 lets do some transformations
        For i = lngBufferRemaining To InputLen - II - lngRem Step 64
            For J = 0 To 63
                ByteBuffer(J) = InputBuffer(i + J)
            Next J
            MD5Transform ByteBuffer
        Next i
        lngBufferedBytes = 0
    Else
        i = 0
    End If

    ' Buffer any remaining input
    For k = 0 To InputLen - i - 1
        ByteBuffer(lngBufferedBytes + k) = InputBuffer(i + k)
    Next k

```

End Sub

' MD5 Transform

,

Private Sub MD5Transform(Buffer() As Byte)

Dim x(16) As Long

Dim a As Long

Dim b As Long

Dim c As Long

Dim d As Long

a = State(1)

b = State(2)

c = State(3)

d = State(4)

Decode 64, x, Buffer

' Round 1

FF a, b, c, d, x(0), S11, -680876936

FF d, a, b, c, x(1), S12, -389564586

FF c, d, a, b, x(2), S13, 606105819

FF b, c, d, a, x(3), S14, -1044525330

FF a, b, c, d, x(4), S11, -176418897

FF d, a, b, c, x(5), S12, 1200080426

FF c, d, a, b, x(6), S13, -1473231341

FF b, c, d, a, x(7), S14, -45705983

FF a, b, c, d, x(8), S11, 1770035416

FF d, a, b, c, x(9), S12, -1958414417

FF c, d, a, b, x(10), S13, -42063

FF b, c, d, a, x(11), S14, -1990404162

FF a, b, c, d, x(12), S11, 1804603682

FF d, a, b, c, x(13), S12, -40341101

FF c, d, a, b, x(14), S13, -1502002290

FF b, c, d, a, x(15), S14, 1236535329

' Round 2

GG a, b, c, d, x(1), S21, -165796510

GG d, a, b, c, x(6), S22, -1069501632

GG c, d, a, b, x(11), S23, 643717713

GG b, c, d, a, x(0), S24, -373897302

GG a, b, c, d, x(5), S21, -701558691

GG d, a, b, c, x(10), S22, 38016083

GG c, d, a, b, x(15), S23, -660478335

GG b, c, d, a, x(4), S24, -405537848

GG a, b, c, d, x(9), S21, 568446438

GG d, a, b, c, x(14), S22, -1019803690

GG c, d, a, b, x(3), S23, -187363961

GG b, c, d, a, x(8), S24, 1163531501

GG a, b, c, d, x(13), S21, -1444681467

GG d, a, b, c, x(2), S22, -51403784

GG c, d, a, b, x(7), S23, 1735328473

GG b, c, d, a, x(12), S24, -1926607734

' Round 3

HH a, b, c, d, x(5), S31, -378558

HH d, a, b, c, x(8), S32, -2022574463

HH c, d, a, b, x(11), S33, 1839030562

HH b, c, d, a, x(14), S34, -35309556

HH a, b, c, d, x(1), S31, -1530992060

```

HH d, a, b, c, x(4), S32, 1272893353
HH c, d, a, b, x(7), S33, -155497632
HH b, c, d, a, x(10), S34, -1094730640
HH a, b, c, d, x(13), S31, 681279174
HH d, a, b, c, x(0), S32, -358537222
HH c, d, a, b, x(3), S33, -722521979
HH b, c, d, a, x(6), S34, 76029189
HH a, b, c, d, x(9), S31, -640364487
HH d, a, b, c, x(12), S32, -421815835
HH c, d, a, b, x(15), S33, 530742520
HH b, c, d, a, x(2), S34, -995338651

```

```

' Round 4

```

```

II a, b, c, d, x(0), S41, -198630844
II d, a, b, c, x(7), S42, 1126891415
II c, d, a, b, x(14), S43, -1416354905
II b, c, d, a, x(5), S44, -57434055
II a, b, c, d, x(12), S41, 1700485571
II d, a, b, c, x(3), S42, -1894986606
II c, d, a, b, x(10), S43, -1051523
II b, c, d, a, x(1), S44, -2054922799
II a, b, c, d, x(8), S41, 1873313359
II d, a, b, c, x(15), S42, -30611744
II c, d, a, b, x(6), S43, -1560198380
II b, c, d, a, x(13), S44, 1309151649
II a, b, c, d, x(4), S41, -145523070
II d, a, b, c, x(11), S42, -1120210379
II c, d, a, b, x(2), S43, 718787259
II b, c, d, a, x(9), S44, -343485551

```

```

State(1) = LongOverflowAdd(State(1), a)
State(2) = LongOverflowAdd(State(2), b)
State(3) = LongOverflowAdd(State(3), c)
State(4) = LongOverflowAdd(State(4), d)

```

```

' /* Zeroize sensitive information.
' MD5_memset ((POINTER)x, 0, sizeof (x));

```

```

End Sub

```

```

Private Sub Decode(Length As Integer, OutputBuffer() As Long, InputBuffer() As Byte)

```

```

    Dim intDblIndex As Integer
    Dim intByteIndex As Integer
    Dim dblSum As Double

```

```

    intDblIndex = 0

```

```

    For intByteIndex = 0 To Length - 1 Step 4

```

```

        dblSum = InputBuffer(intByteIndex) + _
            InputBuffer(intByteIndex + 1) * 256# + _
            InputBuffer(intByteIndex + 2) * 65536# + _
            InputBuffer(intByteIndex + 3) * 16777216#

```

```

        OutputBuffer(intDblIndex) = UnsignedToLong(dblSum)

```

```

        intDblIndex = intDblIndex + 1

```

```

    Next intByteIndex

```

```

End Sub

```

```

'

```

```

' FF, GG, HH, and II transformations for rounds 1, 2, 3, and 4.

```

```

' Rotation is separate from addition to prevent recomputation.

```

```

Private Function FF(a As Long, _
    b As Long, _
    c As Long, _
    d As Long, _
    x As Long, _
    s As Long, _
    ac As Long) As Long
    a = LongOverflowAdd4(a, (b And c) Or (Not (b) And d), x, ac)
    a = LongLeftRotate(a, s)
    a = LongOverflowAdd(a, b)
End Function

```

```

Private Function GG(a As Long, _
    b As Long, _
    c As Long, _
    d As Long, _
    x As Long, _
    s As Long, _
    ac As Long) As Long
    a = LongOverflowAdd4(a, (b And d) Or (c And Not (d)), x, ac)
    a = LongLeftRotate(a, s)
    a = LongOverflowAdd(a, b)
End Function

```

```

Private Function HH(a As Long, _
    b As Long, _
    c As Long, _
    d As Long, _
    x As Long, _
    s As Long, _
    ac As Long) As Long
    a = LongOverflowAdd4(a, b Xor c Xor d, x, ac)
    a = LongLeftRotate(a, s)
    a = LongOverflowAdd(a, b)
End Function

```

```

Private Function II(a As Long, _
    b As Long, _
    c As Long, _
    d As Long, _
    x As Long, _
    s As Long, _
    ac As Long) As Long
    a = LongOverflowAdd4(a, c Xor (b Or Not (d)), x, ac)
    a = LongLeftRotate(a, s)
    a = LongOverflowAdd(a, b)
End Function

```

```

'
' Rotate a long to the right
'

```

```

Function LongLeftRotate(value As Long, bits As Long) As Long
    Dim lngSign As Long
    Dim lngI As Long
    bits = bits Mod 32
    If bits = 0 Then LongLeftRotate = value: Exit Function
    For lngI = 1 To bits
        lngSign = value And &HC0000000
        value = (value And &H3FFFFFFF) * 2
    
```

```

        value = value Or ((lngSign < 0) And 1) Or (CBool(lngSign And _
            &H40000000) And &H80000000)
    Next
    LongLeftRotate = value
End Function

' Function to add two unsigned numbers together as in C.
' Overflows are ignored!
'
Private Function LongOverflowAdd(Val1 As Long, Val2 As Long) As Long
    Dim lngHighWord As Long
    Dim lngLowWord As Long
    Dim lngOverflow As Long

    lngLowWord = (Val1 And &HFFFF&) + (Val2 And &HFFFF&)
    lngOverflow = lngLowWord \ 65536
    lngHighWord = (((Val1 And &HFFFF0000) \ 65536) + ((Val2 And &HFFFF0000) \ 65536) +
lngOverflow) And &HFFFF&
    LongOverflowAdd = UnsignedToLong((lngHighWord * 65536#) + (lngLowWord And &HFFFF&))
End Function

'
' Function to add two unsigned numbers together as in C.
' Overflows are ignored!
'
Private Function LongOverflowAdd4(Val1 As Long, Val2 As Long, val3 As Long, val4 As Long) As
Long
    Dim lngHighWord As Long
    Dim lngLowWord As Long
    Dim lngOverflow As Long

    lngLowWord = (Val1 And &HFFFF&) + (Val2 And &HFFFF&) + (val3 And &HFFFF&) + (val4
And &HFFFF&)
    lngOverflow = lngLowWord \ 65536
    lngHighWord = (((Val1 And &HFFFF0000) \ 65536) + _
        ((Val2 And &HFFFF0000) \ 65536) + _
        ((val3 And &HFFFF0000) \ 65536) + _
        ((val4 And &HFFFF0000) \ 65536) + _
        lngOverflow) And &HFFFF&
    LongOverflowAdd4 = UnsignedToLong((lngHighWord * 65536#) + (lngLowWord And
&HFFFF&))
End Function

' Convert an unsigned double into a long
'
Private Function UnsignedToLong(value As Double) As Long
    If value < 0 Or value >= OFFSET_4 Then Error 6 ' Overflow
    If value <= MAXINT_4 Then
        UnsignedToLong = value
    Else
        UnsignedToLong = value - OFFSET_4
    End If
End Function

' Convert a long to an unsigned Double
'
Private Function LongToUnsigned(value As Long) As Double
    If value < 0 Then
        LongToUnsigned = value + OFFSET_4
    Else

```

```

        LongToUnsigned = value
    End If
End Function

```

FORM ABOUT (FrmAbout)

```

Private Sub mnuabout_Click()
    FrmAbout.Left = FrmAbout.Left
    FrmAbout.Top = FrmAbout.Top
    FrmAbout.Show
    Unload Me
End Sub

```

```

Private Sub mnudekrip_Click()
    FrmDecrypt.Left = FrmAbout.Left
    FrmDecrypt.Top = FrmAbout.Top
    FrmDecrypt.Show
    Unload Me
End Sub

```

```

Private Sub mnuekstraksi_Click()
    frm_Ekstraksi.Left = FrmAbout.Left
    frm_Ekstraksi.Top = FrmAbout.Top
    frm_Ekstraksi.Show
    Unload Me
End Sub

```

```

Private Sub mnuenkrip_Click()
    FrmEncrypt.Left = FrmAbout.Left
    FrmEncrypt.Top = FrmAbout.Top
    FrmEncrypt.Show
    Unload Me
End Sub

```

```

Private Sub mnuexit_Click()
    End
End Sub

```

```

Private Sub mnukunci_Click()
    FrmSign.Left = FrmAbout.Left
    FrmSign.Top = FrmAbout.Top
    FrmSign.Show
    Unload Me
End Sub

```

```

Private Sub mnuotentikasi_Click()
    FrmOtentikasi.Left = FrmAbout.Left
    FrmOtentikasi.Top = FrmAbout.Top
    FrmOtentikasi.Show
    Unload Me
End Sub

```

```

Private Sub mnutttd_Click()
    Frm_ttd_file.Left = FrmAbout.Left
    Frm_ttd_file.Top = FrmAbout.Top
    Frm_ttd_file.Show
    Unload Me
End Sub

```