LAMPIRAN A

LISTING PROGRAM

EinsteinProject-FormKeyGen (Code)

```
Dim currNumber As Long
Private Sub CmdGenerate_Click()
   currNumber = Val(TxtSmallest.Text)
   TmrGenerator.Enabled = True
   CmdGenerate.Enabled = False
End Sub
Private Sub Form_Load()
   LstKeys.Clear
   SetNumber TxtSmallest, True
End Sub
Private Sub LstKeys_DblClick()
   FormNMain.TxtEKey.Text = LstKeys.List(LstKeys.ListIndex)
End Sub
Private Sub TmrGenerator_Timer()
   If LstKeys.ListCount < 25 Then
      If IsValidKey(currNumber) Then
         LstKeys.AddItem CStr(currNumber)
      End If
      currNumber = currNumber + 1
   Else
      TmrGenerator.Enabled = False
      CmdGenerate.Enabled = True
   End If
End Sub
Private Sub TxtSmallest_Change()
   If TxtSmallest.Text <> "0" Then
      CmdGenerate.Enabled = True
   Else
      CmdGenerate.Enabled = False
   End If
End Sub
```

EinsteinProject-FormNMain (Code)

```
Dim FDlg As New ClsFileDialog
Dim SHApp As Object
Private Sub Execute(ByVal execName As String, Optional ByVal aParam As
Variant, _
   Optional ByVal execDirectory As String)
   SHApp.ShellExecute execName, aParam, vbNullString, vbNullString, 0
End Sub
Private Sub refreshList()
Dim i As Long
```

```
Dim anItem As ListItem
   i = 0
   LVLogs.ListItems.Clear
   While i < projLogCount
      Set anItem = LVLogs.ListItems.Add
      With projLog(i)
         anItem.Text = .projectID
         anItem.SubItems(1) = .sourceFilename
         anItem.SubItems(2) = .saveAsFilename
         anItem.SubItems(3) = .encryptTime
      End With
      i = i + 1
   Wend
   Set anItem = Nothing
End Sub
Private Sub CmdCompare_Click()
   Execute App.Path & "\File Compare.exe", PPDResult.ItemValue(0)
End Sub
Private Sub CmdDBrowse_Click()
   FDlg.filter = "Einstein Encrypted File (*.eef)|*.eef"
   FDlg.Filename = TxtDFilename.Text
   FDlg.ShowOpen
   If FDlg.Filename <> "" And FDlg.Filename <> TxtDFilename.Text Then
      TxtDFilename.Text = FDlg.Filename
   End If
End Sub
Private Sub CmdDecrypt_Click()
   If TxtDFilename.Text = "" Then
      MsgBox "Please specify filename to be decrypt!", vbOKOnly + vbCritical,
"Decryption Error"
   Else
      If TxtDKey.Text <> "" Then
         If IsValidKey(Val(TxtDKey.Text)) Then
            pKey = Val(TxtDKey.Text)
            If FormProcess.DecryptFile(TxtDFilename.Text, Me) Then
               PPDResult.ItemValue(1) = Trim(Str(FileLen(App.Path &
"\DResult.tmp"))) & " bytes"
            Else
               MsgBox "Invalid key.  The key is not encryption key", vbOKOnly +
vbCritical, "Decryption Error"
            End If
         Else
            MsgBox "Invalid key.  The key must have two prime factor",
vbOKOnly + vbCritical, "Decryption Error"
         End If
      Else
```

```
        MsgBox "Please specify encryption key", vbOKOnly + vbCritical,
"Decryption Error"
      End If
   End If
End Sub
Private Sub CmdDNew_Click()
   TxtDFilename.Text = ""
   TxtDKey.Text = ""
   PPDResult.ItemValue(0) = ""
   PPDResult.ItemValue(1) = ""
   PPDResult.ItemGroupValue(2, 0) = "0"
   PPDResult.ItemGroupValue(2, 1) = "0"
   PPDResult.ItemGroupValue(2, 2) = "0"
   FDlg.DeleteFile App.Path & "\DResult.tmp"
End Sub
Private Sub CmdDSave_Click()
Dim arrAttr() As String
Dim anAttr As Long
Dim i As Long
   If PPDResult.ItemValue(0) <> "" Then
      FDlg.CopyFile App.Path & "\DResult.tmp", PPDResult.ItemValue(0), True
      If StrComp(PPDResult.ItemValue(3), "True", vbTextCompare) = 0 Then
         arrAttr = Split(PPSProp.ItemValue(3), "+", , vbTextCompare)
         anAttr = 0
         i = LBound(arrAttr)
         While i <= UBound(arrAttr)
            If StrComp(arrAttr(i), "Normal", vbTextCompare) = 0 Then
               anAttr = 0
            Else
               If StrComp(arrAttr(i), "Archive", vbTextCompare) = 0 Then
                  anAttr = anAttr + vbArchive
               ElseIf StrComp(arrAttr(i), "Hidden", vbTextCompare) = 0 Then
                  anAttr = anAttr + vbHidden
               ElseIf StrComp(arrAttr(i), "Read-only", vbTextCompare) = 0 Then
                  anAttr = anAttr + vbReadOnly
               ElseIf StrComp(arrAttr(i), "System", vbTextCompare) = 0 Then
                  anAttr = anAttr + vbSystem
               End If
            End If
            i = i + 1
         Wend
         SetAttr PPDResult.ItemValue(0), anAttr
      End If
   Else
      MsgBox "Please specify the Save As Filename property!", vbOKOnly +
vbCritical, "Saving Error"
   End If
```

```
End Sub
Private Sub CmdEBrowse_Click()
Dim strAttr As String
   FDlg.filter = "All Files(*.*)|*.*"
   FDlg.Filename = TxtEFilename.Text
   FDlg.ShowOpen
   If FDlg.Filename <> "" Then
      TxtEFilename.Text = FDlg.Filename
      PPESource.ItemValue(0) = FDlg.FileTitle
      PPESource.ItemValue(1) = Trim(Str(FileLen(FDlg.Filename))) & " bytes"
      PPESource.ItemValue(2) = FDlg.FileType
      strAttr = ""
      If GetAttr(FDlg.Filename) And vbNormal Then
         strAttr = "Normal"
      Else
         If GetAttr(FDlg.Filename) And vbArchive Then
            strAttr = "Archive"
         End If
         If GetAttr(FDlg.Filename) And vbHidden Then
            If strAttr <> "" Then
               strAttr = "Hidden"
            Else
               strAttr = "+Hidden"
            End If
         End If
         If GetAttr(FDlg.Filename) And vbReadOnly Then
            If strAttr <> "" Then
               strAttr = "Read-only"
            Else
               strAttr = "+Read-only"
            End If
         End If
         If GetAttr(FDlg.Filename) And vbSystem Then
            If strAttr <> "" Then
               strAttr = "System"
            Else
               strAttr = "+System"
            End If
         End If
      End If
      PPESource.ItemValue(3) = strAttr
   End If
End Sub
Private Sub CmdEncrypt_Click()
Dim fTmp As Long
   If TxtEText.Text <> "" Then
      fTmp = FreeFile
```

```
      Open App.Path & "\Untitled.txt" For Output As #fTmp
      Print #fTmp, TxtEText.Text
      Close (fTmp)
   End If
   If TxtEFilename.Text = "" And TxtEText.Text = "" Then
      MsgBox "Please specify filename or text to be encrypted", vbOKOnly +
vbCritical, "Encryption Error"
   Else
      If TxtEKey.Text = "" Then
         MsgBox "Please specify encryption key", vbOKOnly + vbCritical,
"Encryption Error"
      Else
         If IsValidKey(Val(TxtEKey.Text)) Then
            Me.MousePointer = 11
            pKey = Val(TxtEKey.Text)
            If TxtEText.Text = "" Then
               FormProcess.EncryptFile TxtEFilename.Text, Me
            Else
               FormProcess.EncryptFile App.Path & "\Untitled.txt", Me
            End If
            PPEResult.ItemValue(2) = Trim(Str(FileLen(App.Path &
"\EResult.tmp"))) & " bytes"
            Me.MousePointer = 0
         Else
            MsgBox "Invalid key.  The key must have two prime factor",
vbOKOnly + vbCritical, "Encryption Error"
         End If
      End If
   End If
End Sub
Private Sub CmdENew_Click()
   TxtEFilename.Text = ""
   TxtEText.Text = ""
   TxtEKey.Text = ""
   PPEResult.ItemValue(0) = getNewID
   PPEResult.ItemValue(1) = ""
   PPEResult.ItemValue(2) = ""
   PPEResult.ItemGroupValue(3, 0) = "0"
   PPEResult.ItemGroupValue(3, 1) = "0"
   PPEResult.ItemGroupValue(3, 2) = "0"
   FDlg.DeleteFile App.Path & "\EResult.tmp"
End Sub
Private Sub CmdESave_Click()
   If PPEResult.ItemValue(1) <> "" Then
      If Len(PPEResult.ItemValue(1)) > 4 Then
         If StrComp(Right(PPEResult.ItemValue(1), 4), ".eef", vbTextCompare)
<> 0 Then
```

```
            PPEResult.ItemValue(1) = PPEResult.ItemValue(1) & ".eef"
          End If
        End If
        FDlg.CopyFile App.Path & "\EResult.tmp", PPEResult.ItemValue(1), True
        addLog PPEResult.ItemValue(0), PPESource.ItemValue(0),
PPEResult.ItemValue(1), PPEResult.GroupValueByName("Time")
        refreshList
    Else
        MsgBox "Please specify the Save As Filename property!", vbOKOnly +
vbCritical, "Saving Error"
        PPEResult.SelectedItem = 1
    End If
End Sub
Private Sub CmdGenerate_Click()
    FormKeyGen.Show vbModal, Me
End Sub
Private Sub Form_Initialize()
    InitCommonControls
End Sub
Private Sub Form_Load()
Dim arrList(0 To 1) As String
    Set SHApp = CreateObject("Shell.Application")
    arrList(0) = "True"
    arrList(1) = "False"
    FDlg.hWndOwner = hWnd
    PPESource.AddItem "Original Filename", "", vtText
    PPESource.ItemReadOnly(0) = True
    PPESource.AddItem "Size", "", vtText
    PPESource.ItemReadOnly(1) = True
    PPESource.AddItem "Type", "", vtText
    PPESource.ItemReadOnly(2) = True
    PPESource.AddItem "Attribute", "", vtText
    PPESource.ItemReadOnly(3) = True
    PPSProp.AddItem "Original Filename", "", vtText
    PPSProp.ItemReadOnly(0) = True
    PPSProp.AddItem "Size", "", vtText
    PPSProp.ItemReadOnly(1) = True
    PPSProp.AddItem "Type", "", vtText
    PPSProp.ItemReadOnly(2) = True
    PPSProp.AddItem "Attribute", "", vtText
    PPSProp.ItemReadOnly(3) = True
    PPEResult.AddItem "Encryption ID", "", vtText
    PPEResult.ItemReadOnly(0) = True
    PPEResult.AddItem "Save As Filename", "", vtBrowseForSave
    PPEResult.ItemBrowseFilter(1) = "Einstein Encrypted File (*.eef)|*.eef"
    PPEResult.AddItem "Result Size", "", vtText
    PPEResult.ItemReadOnly(2) = True
```

```
    PPEResult.AddGroup "Time"
    PPEResult.GroupSeparator(3) = ":"
    PPEResult.AddGroupMember 3, "Hour", "", vtText
    PPEResult.AddGroupMember 3, "Minute", "", vtText
    PPEResult.AddGroupMember 3, "Second", "", vtText
    PPEResult.ItemGroupReadOnly(3, 0) = True
    PPEResult.ItemGroupReadOnly(3, 1) = True
    PPEResult.ItemGroupReadOnly(3, 2) = True
    PPDResult.AddItem "Save As Filename", "", vtBrowseForSave
    PPDResult.ItemBrowseFilter(0) = "All Files(*.*)|*.*"
    PPDResult.AddItem "Result Size", "", vtText
    PPDResult.ItemReadOnly(1) = True
    PPDResult.AddGroup "Time"
    PPDResult.GroupSeparator(2) = ":"
    PPDResult.AddGroupMember 2, "Hour", "", vtText
    PPDResult.AddGroupMember 2, "Minute", "", vtText
    PPDResult.AddGroupMember 2, "Second", "", vtText
    PPDResult.ItemGroupReadOnly(2, 0) = True
    PPDResult.ItemGroupReadOnly(2, 1) = True
    PPDResult.ItemGroupReadOnly(2, 2) = True
    PPDResult.AddItem "Set Attributes", GetSetting("Einstein Project",
"AppSetting", "SetAttr", "True"), vtList, arrList
    SetNumber TxtEKey, True
    SetNumber TxtDKey, True
    loadLogFile
    CmdENew_Click
    CmdDNew_Click
    SSTab1.Tab = 0
    refreshList
End Sub
Private Sub Form_Unload(Cancel As Integer)
    Set SHApp = Nothing
    SaveSetting "Einstein Project", "AppSetting", "SetAttr",
PPDResult.ItemValue(3)
    saveLogFile
End Sub
Private Sub TxtDFilename_Change()
Dim tmpHandle As Long
    If TxtDFilename.Text <> "" Then
        tmpHandle = FreeFile
        Open TxtDFilename.Text For Binary As #tmpHandle
        PPSProp.ItemValue(0) = getOriginalFilename(tmpHandle)
        PPSProp.ItemValue(1) = getOriginalFileSize(tmpHandle)
        PPSProp.ItemValue(2) = getOriginalFileType(tmpHandle)
        PPSProp.ItemValue(3) = getOriginalFileAttribute(tmpHandle)
        Close (tmpHandle)
```

```
      PPDResult.ItemBrowseFilter(0) = PPSProp.ItemValue(2) & " Files|*." &
getFileExtension(PPSProp.ItemValue(0)) & "|All Files(*.*)|*.*"
   Else
      PPSProp.ItemValue(0) = ""
      PPSProp.ItemValue(1) = ""
      PPSProp.ItemValue(2) = ""
      PPSProp.ItemValue(3) = ""
      PPDResult.ItemBrowseFilter(0) = "All Files(*.*)|*.*"
   End If
End Sub
Private Sub TxtEText_Change()
Dim strAttr As String
   If TxtEText.Text <> "" Then
      TxtEFilename.Text = ""
      PPESource.ItemValue(0) = "Untitled.txt"
      PPESource.ItemValue(1) = Trim(Str(Len(TxtEText.Text))) & " bytes"
      PPESource.ItemValue(2) = "Text"
      PPESource.ItemValue(3) = "Normal"
   Else
      TxtEFilename.Text = FDlg.Filename
      If FDlg.Filename <> "" Then
         PPESource.ItemValue(0) = FDlg.FileTitle
         PPESource.ItemValue(1) = Trim(Str(FileLen(FDlg.Filename))) & " bytes"
         PPESource.ItemValue(2) = FDlg.FileType
         strAttr = ""
         If GetAttr(FDlg.Filename) And vbNormal Then
            strAttr = "Normal"
         Else
            If GetAttr(FDlg.Filename) And vbArchive Then
               strAttr = "Archive"
            End If
            If GetAttr(FDlg.Filename) And vbHidden Then
               If strAttr <> "" Then
                  strAttr = "Hidden"
               Else
                  strAttr = "+Hidden"
               End If
            End If
            If GetAttr(FDlg.Filename) And vbReadOnly Then
               If strAttr <> "" Then
                  strAttr = "Read-only"
               Else
                  strAttr = "+Read-only"
               End If
            End If
            If GetAttr(FDlg.Filename) And vbSystem Then
               If strAttr <> "" Then
```

```
                strAttr = "System"
              Else
                strAttr = "+System"
              End If
            End If
          End If
          PPESource.ItemValue(3) = strAttr
        Else
          PPESource.ItemValue(0) = ""
          PPESource.ItemValue(1) = ""
          PPESource.ItemValue(2) = ""
          PPESource.ItemValue(3) = ""
        End If
    End If
End Sub
```

EinsteinProject-FormProcess (Code)

```
Private Declare Function GetTickCount Lib "kernel32" () As Long
Dim fHandle As Long
Dim fTmpHandle As Long
Dim fBuffer As Byte
Dim lstRandom As Long
Dim encResult As Double
Dim byteSign1 As Long, byteSign2 As Long, byteData As Long
Dim bytesEncrypted(0 To 2) As Long
Dim validKey As Long
Dim lastTime As Long
Dim elapsedTime As Long
Dim isEncrypt As Boolean
Private Sub printTime()
Dim newTime As Long
    newTime = (GetTickCount \ 1000) - lastTime
    If newTime <> elapsedTime Then
        elapsedTime = newTime
        LbTime.Caption = Right("00" & Trim(Str(elapsedTime \ 3600)), 2) & ":" &
Right("00" & Trim(Str((elapsedTime Mod 3600) \ 60)), 2) & ":" & Right("00" &
Trim(Str(elapsedTime Mod 60)), 2)
        If isEncrypt Then
            FormNMain.PPEResult.ItemGroupValue(3, 0) = Right("00" &
Trim(Str(elapsedTime \ 3600)), 2)
            FormNMain.PPEResult.ItemGroupValue(3, 1) = Right("00" &
Trim(Str((elapsedTime Mod 3600) \ 60)), 2)
            FormNMain.PPEResult.ItemGroupValue(3, 2) = Right("00" &
Trim(Str(elapsedTime Mod 60)), 2)
```

```
        Else
            FormNMain.PPDResult.ItemGroupValue(2, 0) = Right("00" &
Trim(Str(elapsedTime \ 3600)), 2)
            FormNMain.PPDResult.ItemGroupValue(2, 1) = Right("00" &
Trim(Str((elapsedTime Mod 3600) \ 60)), 2)
            FormNMain.PPDResult.ItemGroupValue(2, 2) = Right("00" &
Trim(Str(elapsedTime Mod 60)), 2)
        End If
    End If
End Sub
Public Sub EncryptFile(ByVal Filename As String, Owner As Form)
    isEncrypt = True
    Label1.Caption = "Please wait, encrypting..."
    fHandle = FreeFile
    Open Filename For Binary As #fHandle
    fTmpHandle = FreeFile
    Open App.Path + "\EResult.tmp" For Binary As #fTmpHandle
    PrintFilename originalFilename, fTmpHandle
    SplitValue pKey, bytesEncrypted(0), bytesEncrypted(1), bytesEncrypted(2)
    Put #fTmpHandle, , bytesEncrypted
    strResult = strResult + Trim(Chr$(byteSign1)) + Trim(Chr$(byteSign2)) +
Trim(Chr$(byteData))
    lstRandom = pKey
    lastTime = GetTickCount \ 1000
    elapsedTime = -1
    TmrEncrypt.Enabled = True
    Me.MousePointer = 11
    Me.Show vbModal, Owner
End Sub
Public Function DecryptFile(ByVal Filename As String, Owner As Form) As
Boolean
    isEncrypt = False
    Label1.Caption = "Please wait, decrypting..."
    fHandle = FreeFile
    Open Filename For Binary As #fHandle
    originalFilename = getOriginalFilename(fHandle)
    getOriginalFileSize fHandle
    getOriginalFileType fHandle
    getOriginalFileAttribute fHandle
    Get #fHandle, , bytesEncrypted
    validKey = CombineValue(bytesEncrypted(0), bytesEncrypted(1),
bytesEncrypted(2))
    DecryptFile = True
    If IsValidKey(validKey) Then
        If validKey <> pKey Then
            DecryptFile = False
            Close (fHandle)
```

```
            Exit Function
        End If
    Else
        DecryptFile = False
        Close (fHandle)
        Exit Function
    End If
    fTmpHandle = FreeFile
    Open App.Path + "\DResult.tmp" For Binary As #fTmpHandle
    lstRandom = validKey
    pKey = validKey
    lastTime = GetTickCount \ 1000
    elapsedTime = -1
    TmrDecrypt.Enabled = True
    Me.MousePointer = 11
    Me.Show vbModal, Owner
End Function
Private Sub TmrDecrypt_Timer()
Dim ecrData As Double
Dim plainData As Double
Dim plainBuffer As Byte
Dim i As Long
    printTime
    i = 0
    While Not EOF(fHandle) And i < 5
        Get #fHandle, , bytesEncrypted
        If Not EOF(fHandle) Then
            ecrData = CombineValue(bytesEncrypted(0), bytesEncrypted(1),
bytesEncrypted(2))
            lstRandom = NextRandom(lstRandom)
            ecrData = XORValueEx(ecrData, lstRandom)
            plainData = DecryptChar(ecrData)
            If plainData Mod 256 >= 0 Then
                plainBuffer = plainData Mod 256
                Put #fTmpHandle, , plainBuffer
            Else
                MsgBox "Invalid data!", vbOKOnly + vbExclamation, "Einstein
Encryption"
                Close (fHandle)
                Close (fTmpHandle)
                TmrDecrypt.Enabled = False
                Me.MousePointer = 0
                Unload Me
            End If
        End If
        i = i + 1
    Wend
```

```
    If EOF(fHandle) Then
        Close (fHandle)
        Close (fTmpHandle)
        TmrDecrypt.Enabled = False
        Me.MousePointer = 0
        Unload Me
    End If
End Sub
Private Sub TmrEncrypt_Timer()
Dim i As Long
    printTime
    i = 0
    While Not EOF(fHandle) And i < 5
        Get #fHandle, , fBuffer
        If Not EOF(fHandle) Then
            encResult = EncryptChar(fBuffer)
            If encResult > -1 Then
                lstRandom = NextRandom(lstRandom)
                encResult = XORValue(encResult, lstRandom)
                SplitValue encResult, bytesEncrypted(0), bytesEncrypted(1),
bytesEncrypted(2)
                Put #fTmpHandle, , bytesEncrypted
            Else
                Close (fHandle)
                Close (fTmpHandle)
                TmrEncrypt.Enabled = False
                Me.MousePointer = 0
                Unload Me
            End If
        End If
        i = i + 1
    Wend
    If EOF(fHandle) Then
        Close (fHandle)
        Close (fTmpHandle)
        TmrEncrypt.Enabled = False
        Me.MousePointer = 0
        Unload Me
    End If
End Sub
Private Sub PrintFilename(ByVal printedFilename As String, ByVal fHandle As
Long)
Dim i As Long, strPrinted As String
    strPrinted = "Original Filename:" & FormNMain.PPESource.ItemValue(0) &
";"
    i = 1
    While i <= Len(strPrinted)
```

```
      Put #fHandle, , AscB(Mid(strPrinted, i, 1))
      i = i + 1
   Wend
   strPrinted = "File size:" & FormNMain.PPESource.ItemValue(1) & ";"
   i = 1
   While i <= Len(strPrinted)
      Put #fHandle, , AscB(Mid(strPrinted, i, 1))
      i = i + 1
   Wend
   strPrinted = "File Type:" & FormNMain.PPESource.ItemValue(2) & ";"
   i = 1
   While i <= Len(strPrinted)
      Put #fHandle, , AscB(Mid(strPrinted, i, 1))
      i = i + 1
   Wend
   strPrinted = "File Attribute:" & FormNMain.PPESource.ItemValue(3) & ";"
   i = 1
   While i <= Len(strPrinted)
      Put #fHandle, , AscB(Mid(strPrinted, i, 1))
      i = i + 1
   Wend
End Sub
```

EinsteinProject-EinsteinModule (Code)

```
Option Explicit
Public Const GWL_STYLE = (-16)
Public Const ES_NUMBER = &H2000&
Public Enum ReadFileMode
   rfmText = 0
   rfmBinary = 1
End Enum
Public Type ProjectLogType
   projectID As String
   sourceFilename As String
   saveAsFilename As String
   encryptTime As String
End Type
Private Declare Function GetTickCount Lib "kernel32" () As Long
Public Declare Sub InitCommonControls Lib "ComCtl32.dll" ()
Public Declare Function GetWindowLong Lib "user32" Alias
"GetWindowLongA" _
   (ByVal hWnd As Long, ByVal nIndex As Long) As Long
Public Declare Function SetWindowLong Lib "user32" Alias
"SetWindowLongA" _
```

```
   (ByVal hWnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long)
As Long
Public AFactor As Long, BFactor As Long, pKey As Long
Public strResult As String
Public originalFilename As String
Public projLog() As ProjectLogType
Public projLogCount As Long
Public Function IsValidKey(ByVal aKey As Long) As Boolean
Dim i As Long
   IsValidKey = False
   i = 1
   While i <= aKey And Not IsValidKey
      If IsPrimeNumber(i) Then
         If aKey Mod i = 0 Then
            If IsPrimeNumber(aKey / i) Then
               If i > aKey / i Then
                  AFactor = i
                  BFactor = aKey / i
               Else
                  BFactor = i
                  AFactor = aKey / i
               End If
               IsValidKey = True
               Exit Function
            End If
         End If
      End If
      i = i + 1
   Wend
End Function
Public Function IsPrimeNumber(ByVal aValue As Long) As Boolean
Dim Divider As Integer
   IsPrimeNumber = True
   If aValue < 0 Then
      aValue = aValue * -1
   End If
   Divider = 2
   While IsPrimeNumber And Divider < aValue
      If aValue Mod Divider = 0 Then
         IsPrimeNumber = False
      End If
      Divider = Divider + 1
   Wend
End Function
Public Function DecToBinary(ByVal aValue As Long, Optional ByVal lBinary
As Long) As String
   DecToBinary = ""
```

```
    While aValue > 0
        DecToBinary = Trim(Str(aValue Mod 2)) + DecToBinary
        aValue = aValue \ 2
    Wend
    While Len(DecToBinary) < lBinary
        DecToBinary = "0" + DecToBinary
    Wend
End Function
Public Function BinaryToDec(ByVal ABinaryValue As String) As Long
Dim i As Integer
    BinaryToDec = 0
    i = 0
    While i < Len(ABinaryValue)
        If Mid(ABinaryValue, Len(ABinaryValue) - i, 1) = "1" Then
            BinaryToDec = BinaryToDec + (2 ^ i)
        End If
        i = i + 1
    Wend
End Function
Public Function XORValue(ByVal aValue As Long, ByVal modValue As Long)
As Long
Dim aValueBin As String, modValueBin As String, resultBin As String
Dim i As Long
    i = 1
    aValueBin = DecToBinary(aValue, 24)
    modValueBin = DecToBinary(modValue, 24)
    resultBin = ""
    While i <= Len(aValueBin)
        If (Mid(aValueBin, i, 1) = "0" And Mid(modValueBin, i, 1) = "0") Or _
            (Mid(aValueBin, i, 1) = "1" And Mid(modValueBin, i, 1) = "1") Then
            resultBin = resultBin + "0"
        Else
            resultBin = resultBin + "1"
        End If
        i = i + 1
    Wend
    XORValue = BinaryToDec(resultBin)
End Function
Public Function XORValueEx(ByVal aValue As Long, ByVal modValue As
Long) As Long
Dim aValueBin As String, modValueBin As String, resultBin As String
Dim i As Long
    i = 1
    aValueBin = DecToBinary(aValue, 24)
    modValueBin = DecToBinary(modValue, 24)
    resultBin = ""
    While i <= Len(aValueBin)
```

```
        If Mid(aValueBin, i, 1) = "0" Then
           resultBin = resultBin + Mid(modValueBin, i, 1)
        Else
           If Mid(modValueBin, i, 1) = "0" Then
              resultBin = resultBin + "1"
           Else
              resultBin = resultBin + "0"
           End If
        End If
        i = i + 1
    Wend
    XORValueEx = BinaryToDec(resultBin)
End Function
Public Function EncryptChar(ByVal strChar As Byte) As Double
    On Error GoTo ecrError
    EncryptChar = (pKey * AFactor / BFactor) + (pKey * strChar * (AFactor +
BFactor))
    Exit Function
ecrError:
    EncryptChar = -1
    MsgBox "Key overflow", vbOKOnly + vbCritical, "Einstein Encryption"
End Function
Public Function DecryptChar(ByVal encChar As Double) As Double
    DecryptChar = (encChar - (pKey * AFactor / BFactor)) / (pKey * (AFactor +
BFactor))
End Function
Public Sub SplitValue(ByVal srcValue As Double, ByRef signValue1 As Long,
ByRef signValue2 As Long, ByRef dataValue As Long)
    dataValue = srcValue Mod 256
    signValue1 = srcValue \ 256
    signValue2 = signValue1 Mod 256
    signValue1 = signValue1 \ 256
End Sub
Public Function CombineValue(ByVal signValue1 As Long, ByVal signValue2
As Long, ByVal dataValue As Long) As Double
    CombineValue = (signValue1 * (256 ^ 2)) + (signValue2 * 256) + dataValue
End Function
Public Function NextRandom(ByVal rSeed As Long) As Long
    'If rSeed <= 0 Then rSeed = 13
    NextRandom = (rSeed * 214013 + 2531011) Mod 1023
End Function
Public Sub SetNumber(NumberText As TextBox, Flag As Boolean)
Dim curstyle As Long
Dim newstyle As Long
    curstyle = GetWindowLong(NumberText.hWnd, GWL_STYLE)
    If Flag Then
      curstyle = curstyle Or ES_NUMBER
```

```
    Else
      curstyle = curstyle And (Not ES_NUMBER)
    End If
    newstyle = SetWindowLong(NumberText.hWnd, GWL_STYLE, curstyle)
    NumberText.Refresh
End Sub
Public Function getOriginalFilename(ByVal fHandle As Long) As String
Dim fBuffer As Byte
Dim arrOrig() As String
Dim origStr As String
    origStr = ""
    Get #fHandle, , fBuffer
    While Chr$(fBuffer) <> ";"
       origStr = origStr + Chr$(fBuffer)
       Get #fHandle, , fBuffer
    Wend
    arrOrig = Split(origStr, ":", , vbTextCompare)
    getOriginalFilename = arrOrig(1)
End Function
Public Function getOriginalFileSize(ByVal fHandle As Long) As String
Dim fBuffer As Byte
Dim arrOrig() As String
Dim origStr As String
    origStr = ""
    Get #fHandle, , fBuffer
    While Chr$(fBuffer) <> ";"
       origStr = origStr + Chr$(fBuffer)
       Get #fHandle, , fBuffer
    Wend
    arrOrig = Split(origStr, ":", , vbTextCompare)
    getOriginalFileSize = arrOrig(1)
End Function
Public Function getOriginalFileType(ByVal fHandle As Long) As String
Dim fBuffer As Byte
Dim arrOrig() As String
Dim origStr As String
    origStr = ""
    Get #fHandle, , fBuffer
    While Chr$(fBuffer) <> ";"
       origStr = origStr + Chr$(fBuffer)
       Get #fHandle, , fBuffer
    Wend
    arrOrig = Split(origStr, ":", , vbTextCompare)
    getOriginalFileType = arrOrig(1)
End Function
Public Function getOriginalFileAttribute(ByVal fHandle As Long) As String
Dim fBuffer As Byte
```

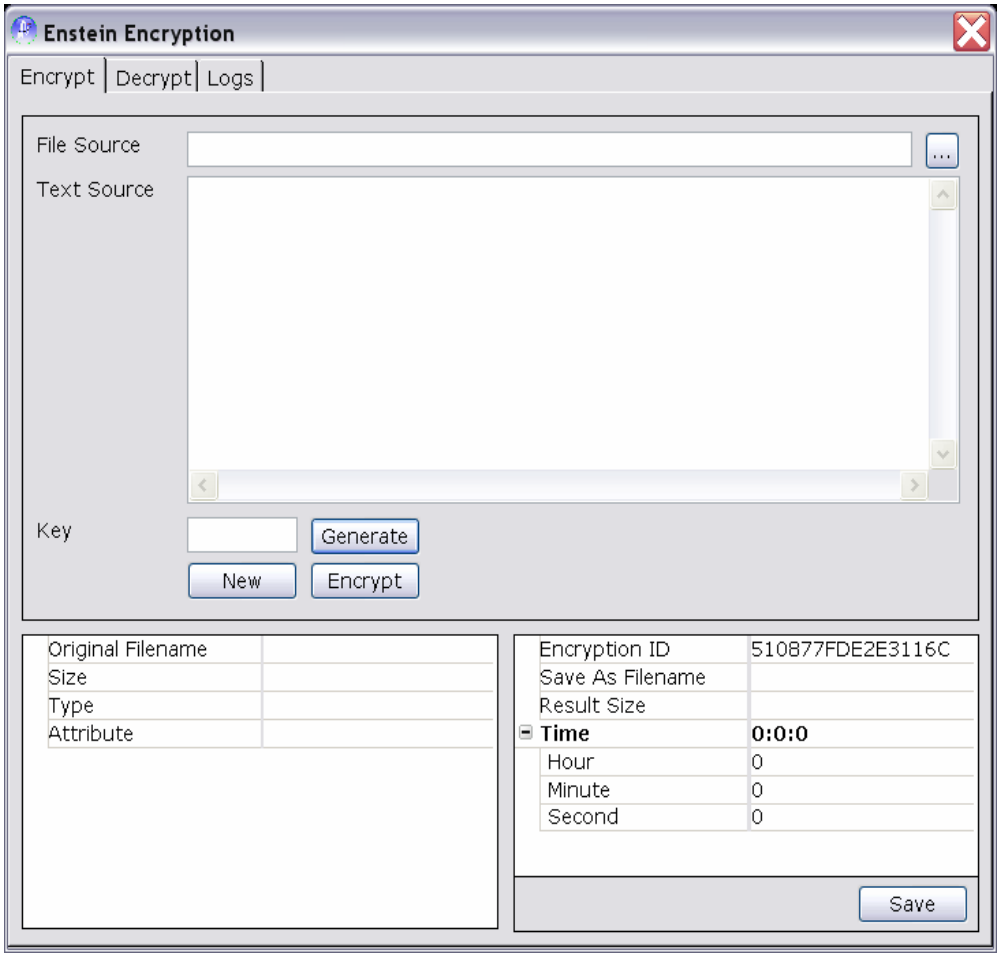
```
Dim arrOrig() As String
Dim origStr As String
   origStr = ""
   Get #fHandle, , fBuffer
   While Chr$(fBuffer) <> ";"
      origStr = origStr + Chr$(fBuffer)
      Get #fHandle, , fBuffer
   Wend
   arrOrig = Split(origStr, ":", , vbTextCompare)
   getOriginalFileAttribute = arrOrig(1)
End Function
Public Sub loadLogFile()
Dim fHandle As Long
Dim fBuffer As String
Dim arrBuffer() As String
   If Dir(App.Path & "\Log.txt") <> "" Then
      fHandle = FreeFile
      projLogCount = 0
      Open App.Path & "\Log.txt" For Input As #fHandle
      While Not EOF(fHandle)
         Line Input #fHandle, fBuffer
         arrBuffer = Split(fBuffer, ";", , vbTextCompare)
         projLogCount = projLogCount + 1
         ReDim Preserve projLog(0 To projLogCount - 1) As ProjectLogType
         projLog(projLogCount - 1).projectID = arrBuffer(0)
         projLog(projLogCount - 1).sourceFilename = arrBuffer(1)
         projLog(projLogCount - 1).saveAsFilename = arrBuffer(2)
         projLog(projLogCount - 1).encryptTime = arrBuffer(3)
      Wend
      Close (fHandle)
   Else
      projLogCount = 0
   End If
End Sub
Public Sub saveLogFile()
Dim i As Long
Dim fHandle As Long
   fHandle = FreeFile
   Open App.Path & "\Log.txt" For Output As #fHandle
   i = 0
   While i < projLogCount
      With projLog(i)
         Print #fHandle, .projectID & ";" & .sourceFilename & ";" &
.saveAsFilename & ";" & .encryptTime
      End With
      i = i + 1
   Wend
```

```
    Close (fHandle)
End Sub
Public Sub addLog(ByVal projID As String, ByVal sourceName As String,
ByVal saveAsName As String, ByVal projTime As String)
    projLogCount = projLogCount + 1
    ReDim Preserve projLog(0 To projLogCount - 1) As ProjectLogType
    With projLog(projLogCount - 1)
        .projectID = projID
        .sourceFilename = sourceName
        .saveAsFilename = saveAsName
        .encryptTime = projTime
    End With
End Sub
Public Function HexGenerator(Optional ByVal hexLength As Long = 16) As
String
Dim tCount As Double
Dim aValue As Long
    HexGenerator = ""
    While Len(HexGenerator) < hexLength
        tCount = GetTickCount
        aValue = (tCount * Rnd) Mod 16
        HexGenerator = HexGenerator & UCase(Trim(Hex$(aValue)))
    Wend
End Function
Public Function getNewID() As String
Dim aHexNumber As String
    aHexNumber = HexGenerator
    While checkProjID(aHexNumber)
        aHexNumber = HexGenerator
    Wend
    getNewID = aHexNumber
End Function
Public Function checkProjID(ByVal aProjID As String) As Boolean
Dim i As Long
Dim find As Boolean
    i = 0
    find = False
    While i < projLogCount And Not find
        If StrComp(aProjID, projLog(i).projectID, vbTextCompare) = 0 Then
            find = True
        End If
        i = i + 1
    Wend
    checkProjID = find
End Function
Public Function getFileExtension(ByVal aFilename As String) As String
Dim arrFilename() As String
```

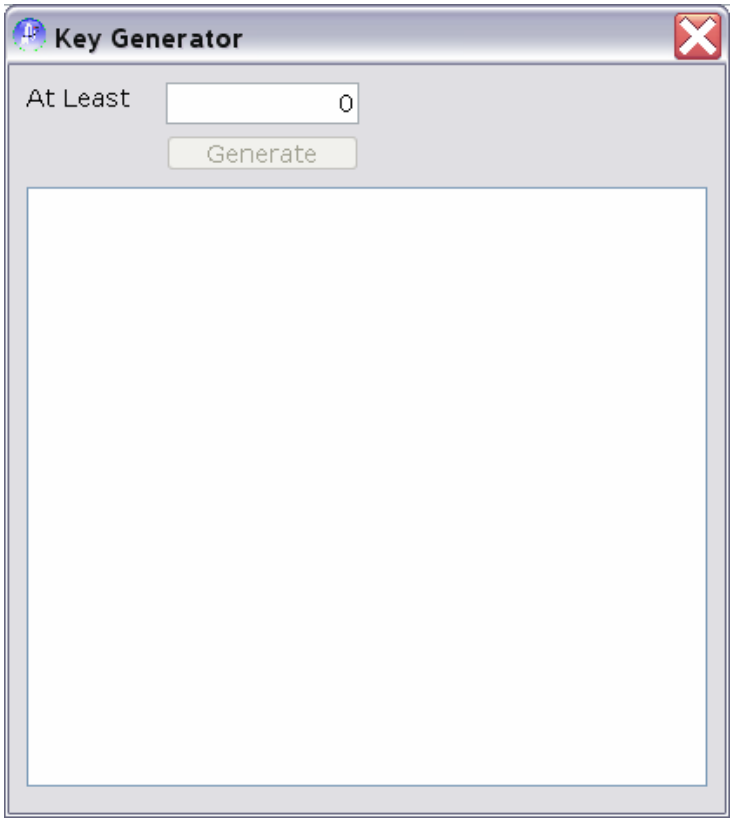
```
    arrFilename = Split(aFilename, "." , , vbTextCompare)
    getFileExtension = ""
    If IsArray(arrFilename) Then
       getFileExtension = arrFilename(UBound(arrFilename))
    End If
End Function
```

LAMPIRAN B

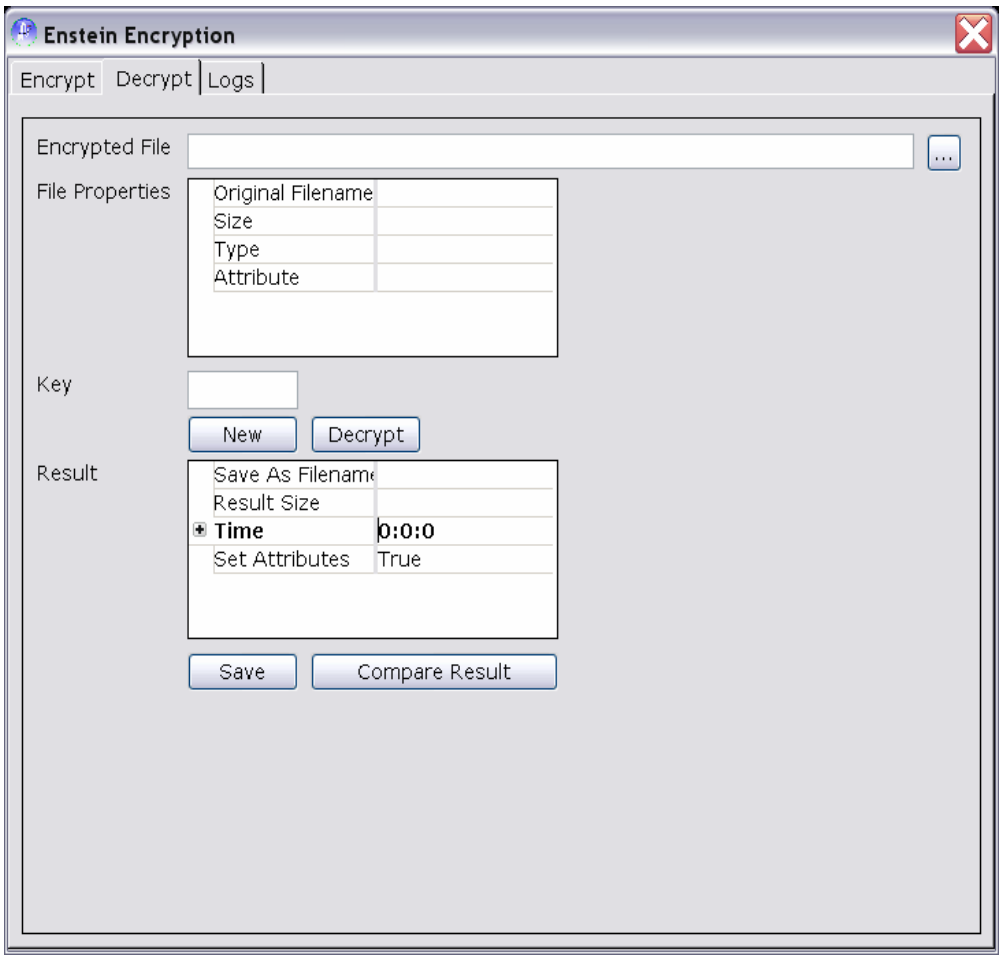
GAMBAR dan TAMPILAN PROGRAM



Tampilan program enkripsi



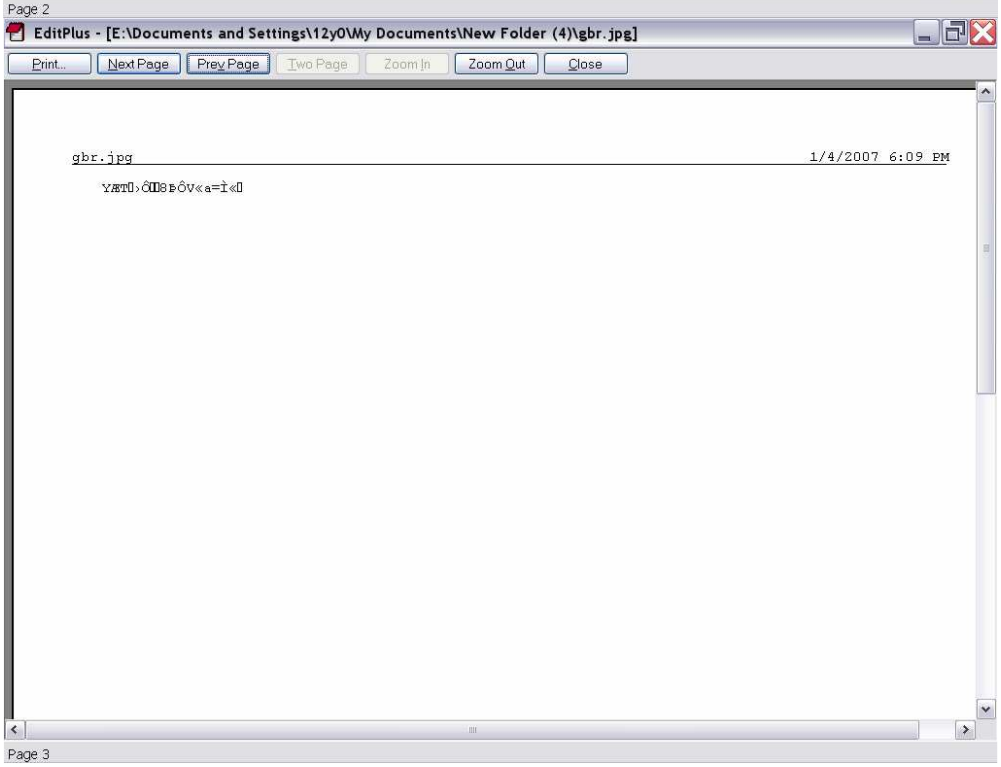
Tampilan pembangkit kunci (*key generator*)



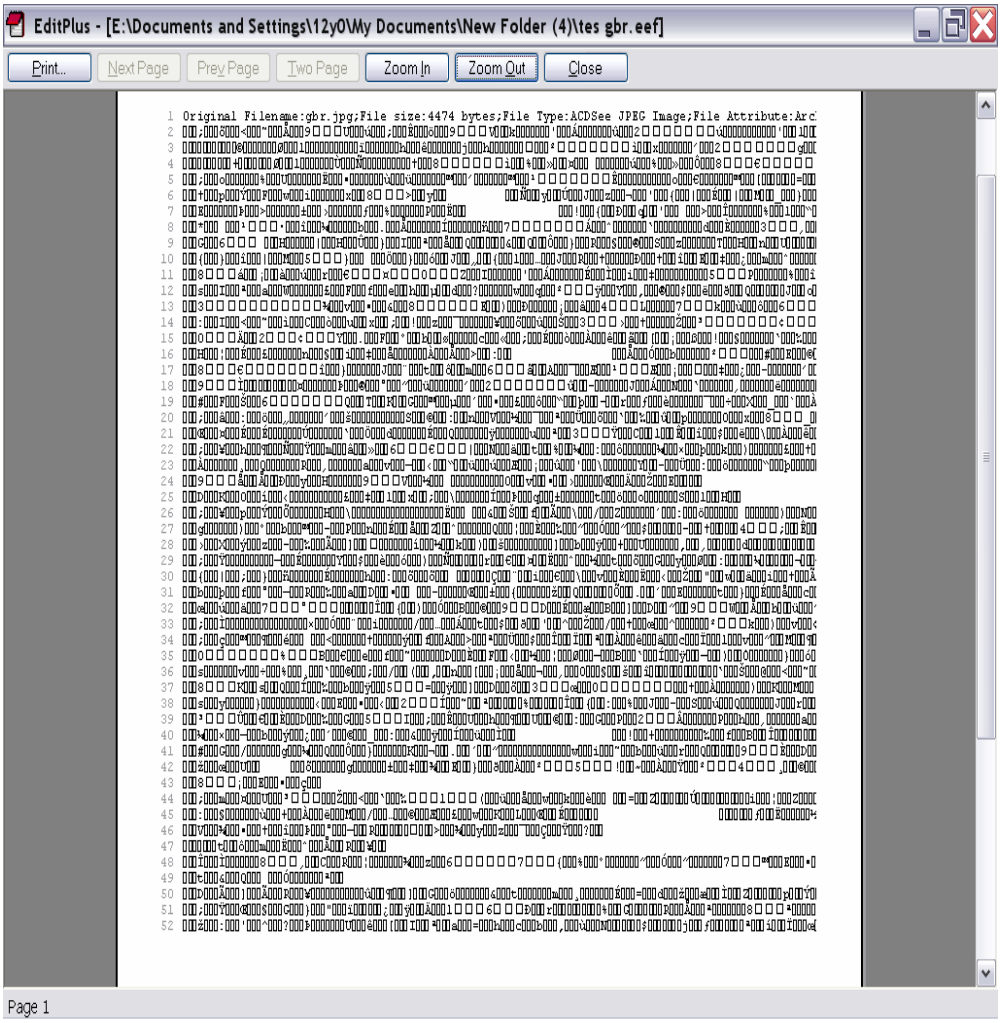
Tampilan program dekripsi

Plaintext gambar *file* ”Gbr.jpg”


EditPlus - [E:\Documents and Settings\12y0\My Documents\New Folder (4)\gbr.jpg]
Print...
Next Page
Prev Page
Two Page
Zoom In
Zoom Out
Close
gbr.jpg
1/4/2007 6:07 PM
Page 2


EditPlus - [E:\Documents and Settings\12y0\My Documents\New Folder (4)\gbr.jpg]
Print...
Next Page
Prev Page
Two Page
Zoom In
Zoom Out
Close
gbr.jpg
1/4/2007 6:09 PM
Page 3

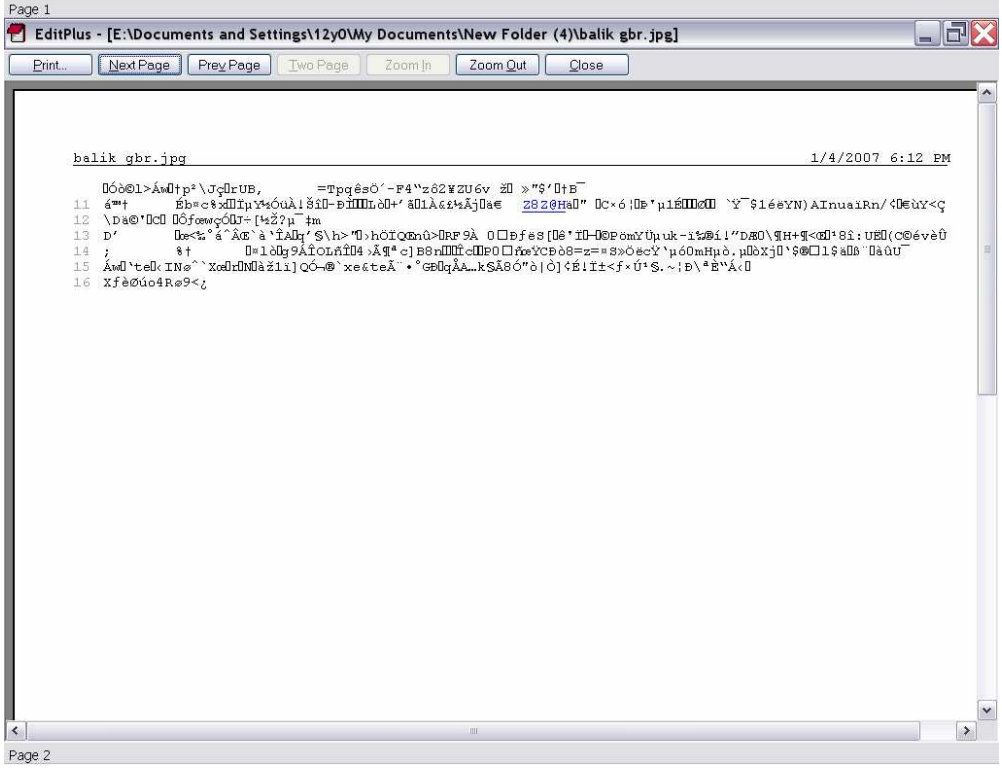
Tampilan kode ASCII *file* ”Gbr.jpg” dengan EditPlus



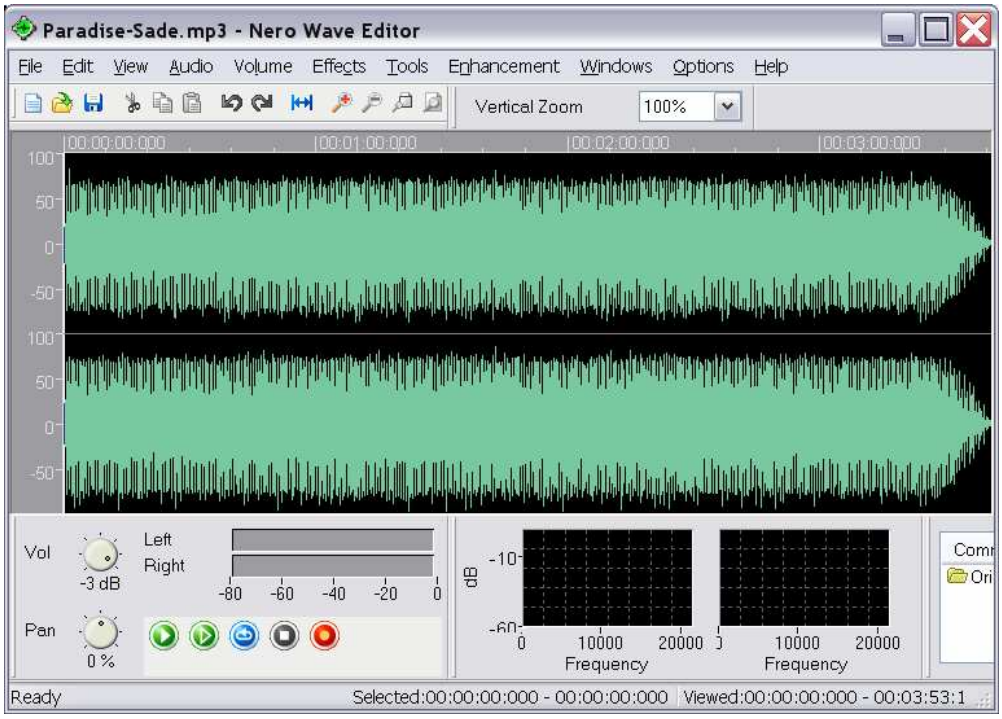
Tampilan *ciphertext file* ”tes gbr.eef” dengan EditPlus

Tampilan gambar *ciphertext file* ”tes gbr.eef” yang telah didekripsi kembali menjadi *file* gambar ”balik gbr.jpg”

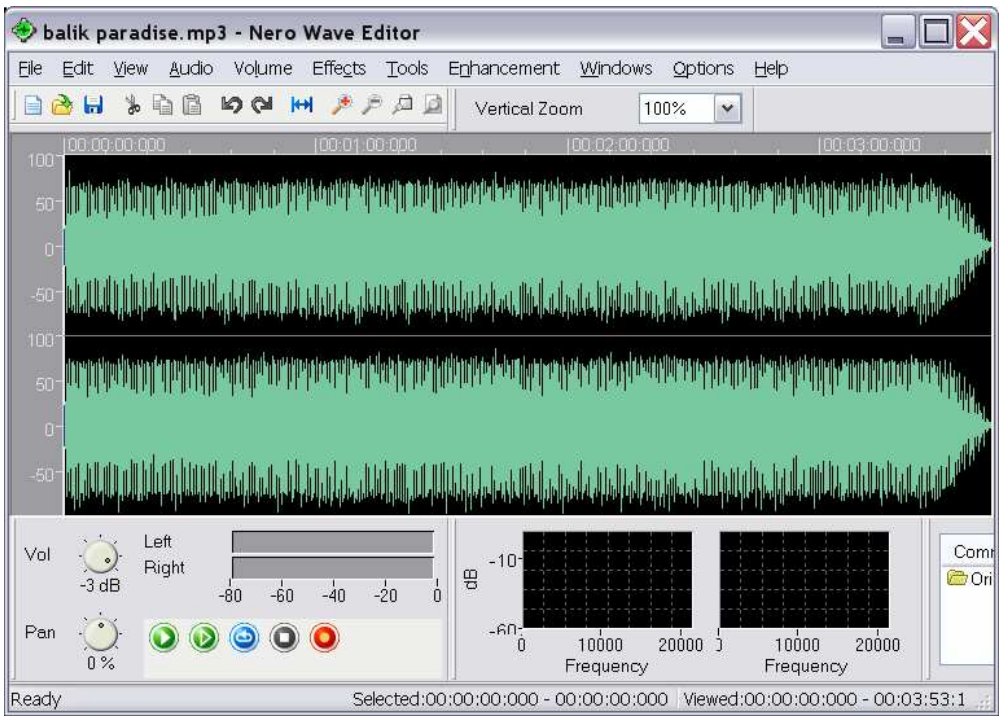

EditPlus - [E:\Documents and Settings\12y0\My Documents\New Folder (4)\balik gbr.jpg]
Print...
Next Page
Prev Page
Two Page
Zoom In
Zoom Out
Close
balik gbr.jpg
1/4/2007 6:02 PM
Page 1


EditPlus - [E:\Documents and Settings\12y0\My Documents\New Folder (4)\balik gbr.jpg]
Print...
Next Page
Prev Page
Two Page
Zoom In
Zoom Out
Close
balik gbr.jpg
1/4/2007 6:12 PM
Page 2

Tampilan dengan EditPlus *file* ”balik gbr.jpg” hasil dekripsi *ciphertext file* ”tes gbr.eef”



Tampilan *plaintext file* ”Paradise-Sade.mp3” dengan Nero Wave Editor 6



Tampilan dengan Nero Wave Editor 6 *file* “balik paradise.mp3” hasil dekripsi *ciphertext file* “tes paradise.eef”