Republic of Iraq
Ministry Of Higher Education
And Scientific Research
University Of Baghdad
College Of Science
Department Of Computer Science

Information Hiding In Images Using Randomly Selected Points

A project is submitted to the department of computer science, college of science, University of Baghdad in partial fulfillment of the requirements for the degree of B.Sc in computer science

Prepared By:
Husam Namir
Omar Fitian

Supervised By:
Dunia Fadheel saffo

May-2010

Index

Chapter one

1.1 Introduction to Steganography

In this age of universal electronic connectivity, of viruses and hackers, of electronic eavesdropping and electronic fraud, there is indeed no time at which security of information does not matter. The explosive growth of computer systems and their interconnections via networks has increased the dependency on the information stored and communication using these systems. This has led to a heightened awareness of the need to protect the data transmitted.
Thus the field of cryptography has got more attention nowadays. More and more complex techniques for encrypting the information are proposed every now and then. Some advanced encryption algorithms like RSA, DES, AES etc. which are extremely hard to crack have been developed. But, as usual, when a small step is made to improve the security, more work is done in the opposite direction by the hackers to break it. Thus they are able to attack most of these algorithms and that too, successfully. Even complex algorithms like RSA are no exception to this.
So, to deceive the hackers, people have started to follow a technique called 'Steganography'. It is not an entirely new technique and has been in the practice from ancient times. In this method, the data is hidden behind unsuspecting objects like images, audio, video etc. so that people cannot even recognize that there is a second message behind the object. Images are commonly used in this technique.

1.2 Background

Steganography is the science of hiding information within other information. For example, a watermark "hides" an image on a piece of paper. If you look at most paper currency at a low angle or if you hold it up to a bright light, you can see a ghostly image in the paper. When you look at the currency straight on in normal light, you cannot see the image. Because this example is so easy to understand, steganography is often called "watermarking."

Another example is a correspondence where the last letter in each word spells out a secret message. Composing this sort of message can be fun. The more constraints you place on the correspondence, the more challenging it is to write something that satisfies the constraints and still contains the hidden message. For example, try writing a poem where the last letter of each word spells out the message. Or try writing a sonnet, which has particular rules for the poem's meter and rhyme.

Today steganography is often used to hide copyright information in an image, movie, or audio file. The information is carefully encrypted and hidden so you cannot easily find it. Later, if I think you have stolen my movie file, I can pull the hidden copyright information out of it to prove it is mine not yours.

1.3 Easter Eggs

For example one developer who uses Easter Eggs in his software as additional copyright protection. An Easter Egg is a hidden feature in the program that displays a secret form or game when activated.

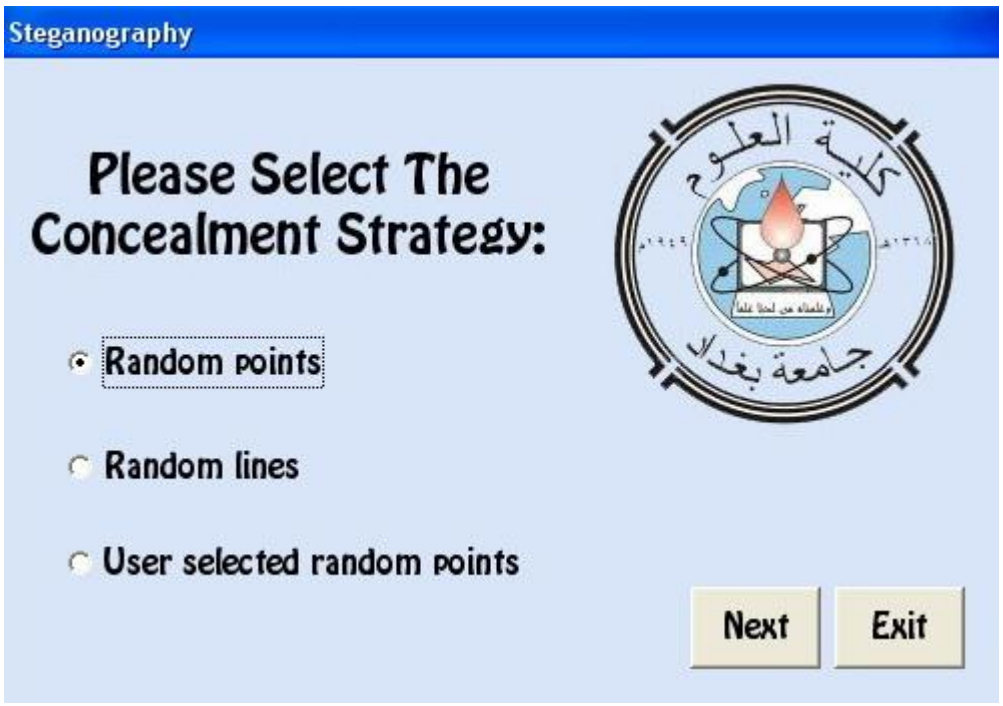
Anyway this particular developer hides copyright information in Easter Eggs. On one occasion, someone else stole his software and claimed to have written it. The true author produced the Easter egg listing his name and the creator and the impostor was forced to admit the truth.

1.4 STEGANOGRAPHY IN IMAGES

In essence, image steganography is about exploiting the limited powers of the human visual system. Within reason, any plain text, ciphertext, other images, or anything that can be embedded in a bit stream can be hidden in an image. The common methods followed for hiding data in images are the 'Least Significant Bit (LSB) Insertion' technique in which the LSB of the pixel values are replaced with the data to be encoded in binary form, the 'Masking Technique' in which the original bits are masked with data bits and the 'Filtering Technique' in which certain transformations are done on the image to hide data. The last two techniques hide data by marking an image in a manner similar to paper watermarks But, there are some drawbacks with these methods which hinders their use.

Chapter TWO

Discussion:



Our project consists of three options that control the distribution of the decrypted message in the image,
Those options are:
(Random points, random lines, user specified random points)
And all those options depend on the Rnd Function.
In this section we'll explain in details the project and those three options…

2.1 Random points

For each bit in the message, this program picks a random pixel in the image and a random red, green, or blue component in that pixel. It then sets the least significant bit in that component to the bit value it is encoding.

For example, suppose the chosen pixel is green so its red, green, and blue components in binary are 0000, 1111, 0000. Now suppose the program wants to hide the bit value 1 in this pixel's red component. The new pixel value has components 0001, 1111, 0000.

For another example, suppose the program wants to store the bit value 0 in the same pixel's blue component. The least significant bit in that component is already 0 so there is no change to the color.

These changes are so small they are almost impossible to detect. In a photographic image, you will not be able to tell the difference just by looking.

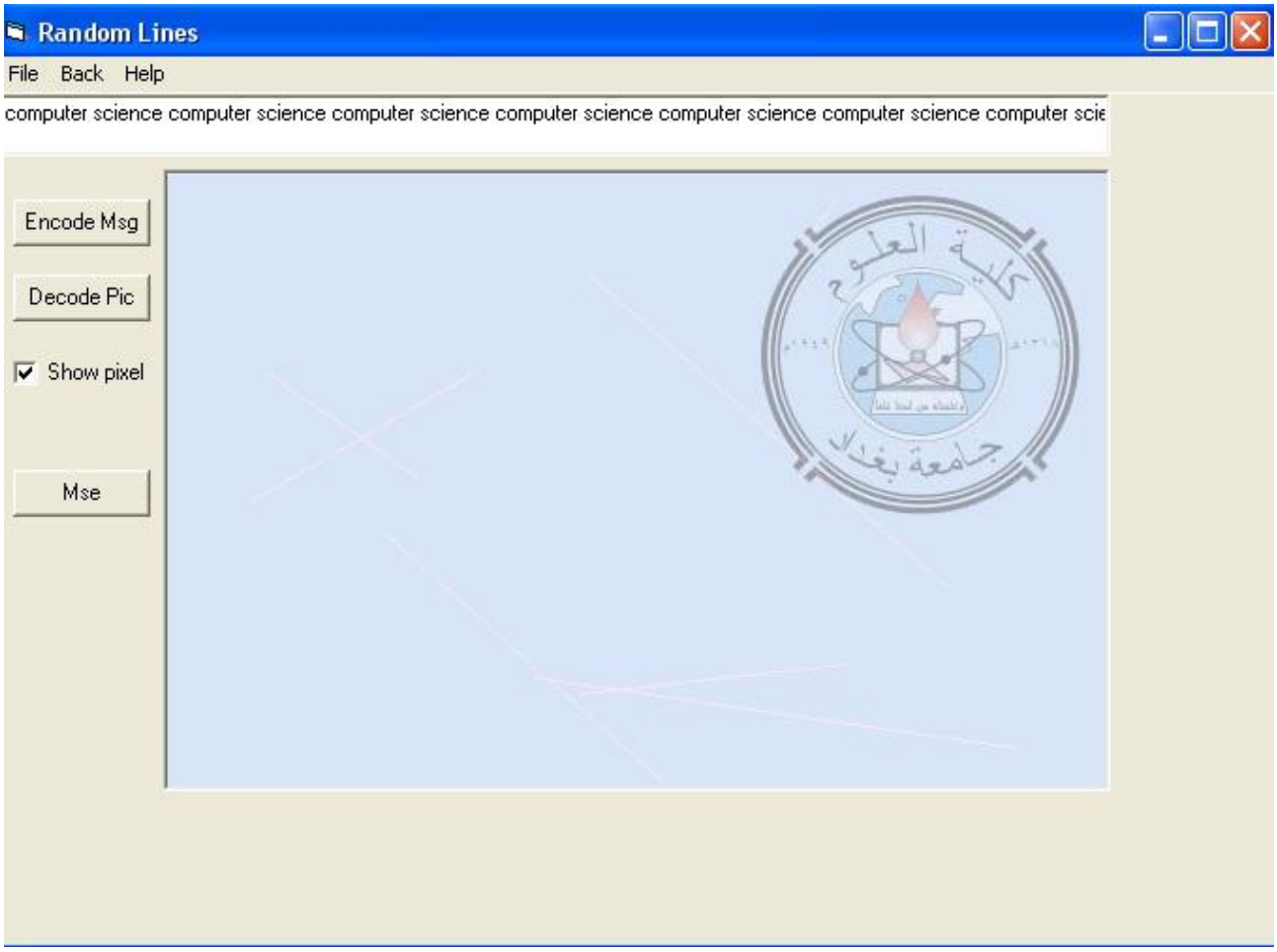
This program handles a lot of little details. For example, before it stores its message it stores the length of the message. That allows the program to know how many characters are encoded when it tries to decode them.

The program uses a zero to initialize the random number generator Rnd. When you decode the message, the program initializes the random number generator using the same value so it produces the same series of pseudo-random numbers. The program also needs to ensure that each message bit maps to a different pixel color component. If two message bits mapped to the same component, the second would hide the value of the first.

The program also provides an option to show the bits it is encoding and decoding. If you check the Show Pixels box, the program makes the pixels it is encoding red. It still sets the least significant bits to encode the messages so you can recover the hidden text even with the red dots in the image.

Similarly if you check the Show Pixels box, the program makes the pixels it is decoding black.

2.2 Random Lines



The idea is the same as the previous option the only difference is that the each time the program generates two random points and by using the DDA algorithm the program uses the points of the lines generated by the DDA between the two selected random points to encrypt the message and sets the least significant bit to the bit value it is encoding.

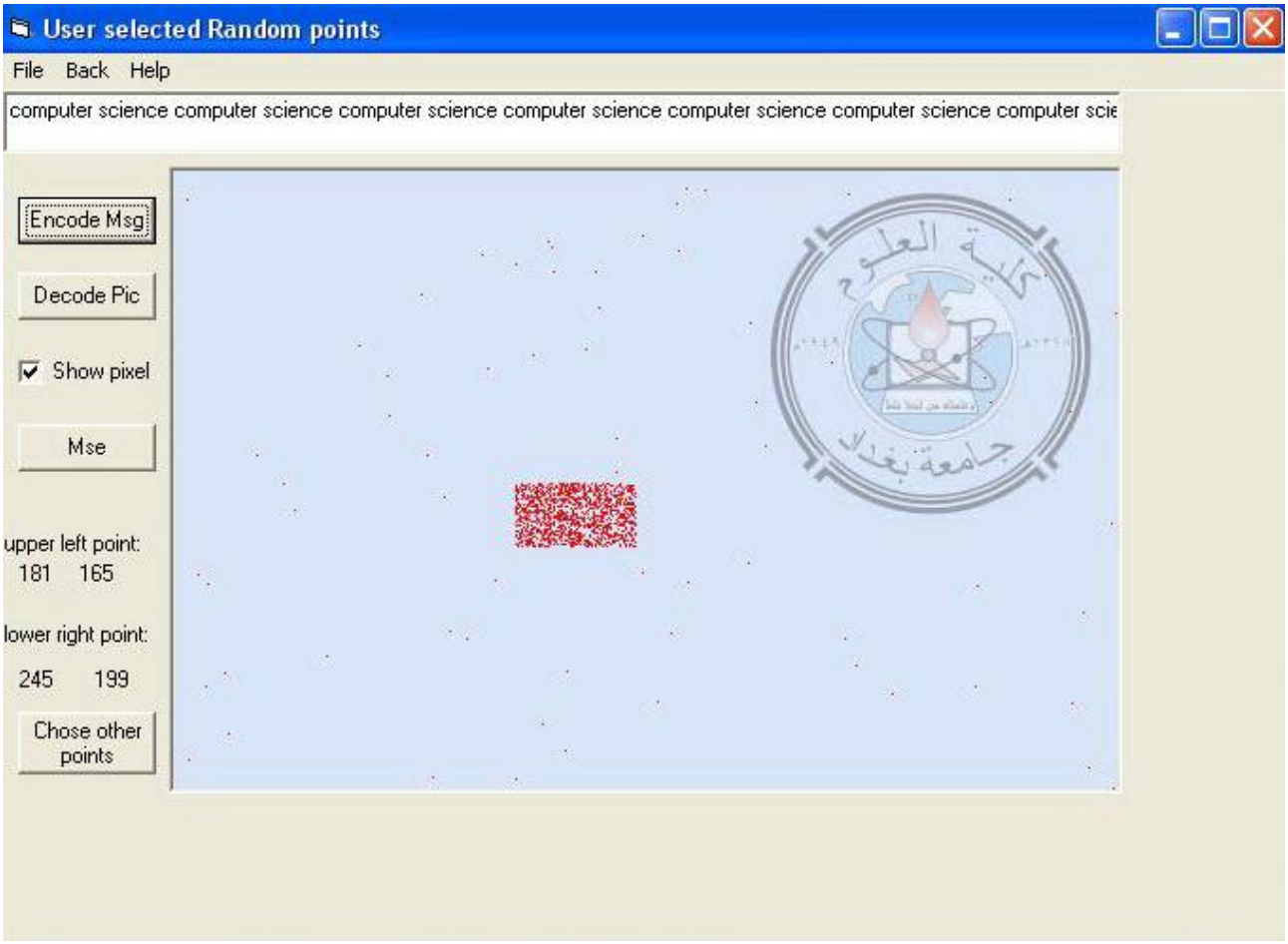
The program also uses a zero to initialize the random number generator Rnd. When you decode the message, the program initializes the random number generator using the same value so it produces the same series of pseudo-random numbers.

This option uses only the green component of each selected point**.**

The program also provides an option to show the bits it is encoding and decoding. If you check the Show Pixels box, the program makes the pixels it is encoding red. It still sets the least significant bits to encode the messages so you can recover the hidden text even with the red dots in the image.

Similarly if you check the Show Pixels box, the program makes the pixels it is decoding black.

2.3 user specified Random points



This option is a lot like the first one,
The only difference is that it will not use the whole image, the user can specify a square or a rectangle by selecting the upper left point and the lower right point and the message will be encrypted in the selected area ,
This option uses the green component of each selected point**.**
The program saves the message length and also the two points selected by the user

Of course the message length and those two points will by saved in all the image (Not only in the selected area)
Because in the decode process we fist decode the length of the message and the two points and after that we can know the selected area where the message is encrypted For that reason we cannot encrypt the message length and the two points in that specified area.

2.4 Additional information

All those options have the capability to calculate the Mean square Error (MSE)
Only after you encrypt a message in the image first, the MSE basically calculate the summation of the difference between each pixel of the original image and the new image that contain the encrypted message,
Note that:
The value of the MSE would be very high if you select the option (Show pixels)

Chapter Three

3.1 Fragile Protection

Note that this technique is relatively fragile. It depends on the least significant bits in an image. Those are the bits most likely to modify by a lossy image compression algorithm such as JPEG. If you encode a message in one of these images, convert it into a JPEG image, and try to decode the JPEG version, you will probably get nothing but garbage.

The bits may also be modified if you view the image with a color resolution other than the one in which the image was made. For example, suppose you make an image with 24-bit color but then view it on another computer with only 8-bit color. The system will modify the colors so it can display the picture on the 8-bit color system. This will almost certainly mess up the hidden data.

These issues are problems with this kind of steganography. To hide the message, the program makes the smallest modifications to the image possible. The basic idea is to make the message look like a tiny bit of noise in the image. However, the smallest bits of noise are most likely to be modified if the image is altered in any way.If you replace Visual Basic's Rnd function with a cryptographically secure random pseudo-number generator, you will get a much more secure system. If you encrypt your message before you embed it in the image, it will be even harder to break.

3.2 DRAWBACKS IN THE CURRENT TECHNIQUES

- Extremely liable to attacks like Image Manipulation techniques where the pixels will be scanned for a possible relation which will be used to trace out the actual characters.
- Only 24 bit images are suitable and 8 bit images are to be used at great risk.
- Extreme Care needs to be taken in the selection of the cover image, so that changes to the data will not be visible in the stego-image.
- Commonly known images, such as famous paintings must be avoided

Chapter Four

Suggestions:

Our method only hides the data in the image with out any encryption.

- One way to improve this method is to encrypt the data (using any encryption methods such as RSA, DES, AES etc) before hiding it ,
  And in the encoding process we first encode the message then we decrypt it to get the original message, the only downside is that some encryption methods adds additional random characters to the plain text causing the cipher text to increase in size thus causing potential problems in hiding large messages in small images where we want to minimize the number of characters as possible.

- Another suggestion is to use a key to initialize the RND function so that the sequence of the random location changes depends on the selected key.

  Ex: Randomize key

- Our last suggestion is that after hiding a message in an image we encrypt the image it self (add some control changes in the pixel) so that we are able to restore the original pixels and the stored message in that image, those changes may be either randomly (same as the hiding method) or in some arbitrary order .

Chapter Five

Data Structure:

Although the randomize function almost guarantees that there are no repeated values in the random no generated by the Rnd function,
But when we use the Int function (ex: x=Int(rnd*picture1.width))
Several different float values will produce the same integer values.
This problem requires that we keep track of the generated locations so that we do not user the same location more than one.
For this purpose we user Collection data type.

A Collection object is an ordered set of items that can be referred to as a unit.
A collection can be created the same way other objects are created. For example:

```
Dim X As New Collection
```

Once a collection is created, members can be added using the Add method and removed using the Remove method.

The best feature of this data type is that if we add to identical values an error will occur, we've taken advantage of this feature by doing the following:

```
On Error Resume Next
   Do
      ' Pick a position.
      r = Int(Rnd * wid)
      c = Int(Rnd * hgt)

      ' See if the position is unused.
      position_code = "(" & r & "," & c & ")"
      used_positions.Add position_code, position_code
      If Err.Number = 0 Then Exit Do
      Err.Clear
   Loop
```

First we generate a random location (r,c), then we add the location to the collection
After that we check if there are no error we exit the loop and we use the selected location, else clear the error and chose another location and so on....

Chapter Six

Results:

In this chapter we present several images before and after we hide a message in it
The message is "computer science computer science computer science"


www.DesktopCollector.com


www.DesktopCollector.com


www.DesktopCollector.com


www.DesktopCollector.com


www.DesktopCollector.com


www.DesktopCollector.com

Chapter Seven

Random points

```
Option Explicit
Dim pixel()
Dim indi As Integer
Dim indj As Integer

Private Sub aa_Click()
End
End Sub

Private Sub aaaa_Click()
Me.Enabled = False
about.Show

End Sub

Private Sub bb_Click()
Unload Me
Form0.Show
End Sub

Private Sub Command1_Click()
Dim msg_len As Integer
Dim wid As Integer
Dim hgt As Integer
Dim i As Integer
Dim msg As String
Dim used_positions As Collection
Dim flag As Boolean
Dim Value As Byte
Dim value1 As Integer
Dim img_size As Double

Screen.MousePointer = vbHourglass

If Check1.Value = 1 Then flag = True Else flag = False
wid = Picture1.ScaleWidth
hgt = Picture1.ScaleHeight
img_size = Picture1.Width * Picture1.Height
msg = Left(Text1.Text, 16000)
msg_len = Len(msg)
ReDim pixel(Picture1.Width, Picture1.Height)
'check if there are enough pixels in the selected image
If img_size < (msg_len * 8 + 14) Then MsgBox "The Image is too small for this message": Exit Sub

 'store the image in the matrix pixel
   For indi = 0 To Picture1.Width - 1
      For indj = 0 To Picture1.Height - 1
      pixel(indi, indj) = Picture1.Point(indi, indj)
      Next
    Next
```

```
    ' Initialize the random number generator.
    Rnd -1
    Randomize 0
    'generate random locations for the msg and the msg length

    Set used_positions = New Collection
    'encode the msg length
value1 = msg_len
Encodelen value1, used_positions, wid, hgt, flag

  'encode the msg
   For i = 1 To msg_len
    EncodeByte Asc(Mid(msg, i, 1)), used_positions, wid, hgt, flag
    Next
Picture1.Picture = Picture1.Image

Command3.Enabled = True
Screen.MousePointer = vbDefault

End Sub

Private Sub Command2_Click()
Dim msglen As Integer
Dim wid As Integer
Dim hgt As Integer
Dim i As Integer
Dim msg As String
Dim used_positions As Collection
Dim flag As Boolean
Dim ch As String

Screen.MousePointer = vbHourglass

If Check1.Value = 1 Then flag = True Else flag = False
wid = Picture1.ScaleWidth
hgt = Picture1.ScaleHeight

 ' Initialize the random number generator.
    Rnd -1
    Randomize 0
    'generate random locations for the msg and the msg length

    Set used_positions = New Collection

    'decode the msg length
    msglen = Decodelen(used_positions, wid, hgt, flag)
 For i = 1 To msglen
 ch = DecodeByte(used_positions, wid, hgt, flag)

 msg = msg & Chr(ch)


  Next
 Text1.Text = msg
```

```
 Screen.MousePointer = vbDefault
End Sub

Private Sub Command3_Click()
Dim mse
Screen.MousePointer = vbHourglass
'calculate the mean square error
For indi = 0 To Picture1.Width - 1
       For indj = 0 To Picture1.Height - 1
       mse = mse + Abs(pixel(indi, indj) - Picture1.Point(indi, indj))
       Next
     Next
Label2.Caption = mse

Screen.MousePointer = vbDefault
End Sub

Private Sub Form_Load()
cd1.Flags = &H2
End Sub

Private Sub hh_Click()
Me.Enabled = False
help.Show

End Sub

Private Sub o_Click()
cd1.InitDir = App.Path
cd1.ShowOpen
If cd1.FileName = "" Then Exit Sub
Picture1.Picture = LoadPicture(cd1.FileName)
Text1.Width = Picture1.Width + Picture1.Left
Command1.Enabled = True
Command2.Enabled = True
cd1.FileName = ""
End Sub

Private Sub UnRGB(ByVal color As OLE_COLOR, ByRef r As Byte, ByRef g As Byte, ByRef b
As Byte)
   r = color And &HFF&
   g = (color And &HFF00&) \ &H100&
   b = (color And &HFF0000) \ &H10000
End Sub

Private Sub ss_Click()
cd1.ShowSave
If cd1.FileName = "" Then Exit Sub
SavePicture Picture1.Picture, cd1.FileName
cd1.FileName = ""
End Sub

' Pick an unused (r, c, pixel) combination.
```

```
Private Sub PickPosition(ByVal used_positions As Collection, ByVal wid As Integer, ByVal hgt As
Integer, ByRef r As Integer, ByRef c As Integer, ByRef pixel As Integer)
Dim position_code As String

    On Error Resume Next
    Do
        ' Pick a position.
        r = Int(Rnd * wid)
        c = Int(Rnd * hgt)
        pixel = Int(Rnd * 3)

        ' See if the position is unused.
        position_code = "(" & r & "," & c & "," & pixel & ")"
        used_positions.Add position_code, position_code
        If Err.Number = 0 Then Exit Do
        Err.Clear
    Loop
End Sub

Private Function DecodeByte(ByVal used_positions As Collection, ByVal wid As Integer, ByVal
hgt As Integer, ByVal show_pixels As Boolean) As Byte
Dim Value As Integer
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 8
        ' Pick a random pixel and RGB component.
        PickPosition used_positions, wid, hgt, r, c, pixel

        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb

        ' Get the stored value.
        Select Case pixel
            Case 0
                color_mask = (clrr And &H1)
            Case 1
                color_mask = (clrg And &H1)
            Case 2
                color_mask = (clrb And &H1)
        End Select

        If color_mask Then
            Value = Value Or byte_mask
        End If
```

```
        If show_pixels Then
            Picture1.PSet (r, c), RGB(0, 0, 0)

        End If

        byte_mask = byte_mask * 2
    Next i

    DecodeByte = CByte(Value)
End Function


Private Sub EncodeByte(ByVal Value As Byte, ByVal used_positions As Collection, ByVal wid As
Integer, ByVal hgt As Integer, ByVal show_pixels As Boolean)
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 8
        ' Pick a random pixel and RGB component.
        PickPosition used_positions, wid, hgt, r, c, pixel

        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb
        If show_pixels Then
            clrr = 255
            clrg = clrg And &H1
            clrb = clrb And &H1
        End If

        ' Get the value we must store.
        If Value And byte_mask Then
            color_mask = 1
        Else
            color_mask = 0
        End If

        ' Update the color.
        Select Case pixel
            Case 0
                clrr = (clrr And &HFE) Or color_mask
            Case 1
                clrg = (clrg And &HFE) Or color_mask
            Case 2
                clrb = (clrb And &HFE) Or color_mask
        End Select
```

```
        ' Set the pixel's color.
        Picture1.PSet (r, c), RGB(clrr, clrg, clrb)

        byte_mask = byte_mask * 2
    Next i
End Sub


Private Sub Encodelen(ByVal Value As Integer, ByVal used_positions As Collection, ByVal wid As
Integer, ByVal hgt As Integer, ByVal show_pixels As Boolean)
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 14
        ' Pick a random pixel and RGB component.
        PickPosition used_positions, wid, hgt, r, c, pixel

        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb
        If show_pixels Then
            clrr = 255
            clrg = clrg And &H1
            clrb = clrb And &H1
        End If

        ' Get the value we must store.
        If Value And byte_mask Then
            color_mask = 1
        Else
            color_mask = 0
        End If

        ' Update the color.
        Select Case pixel
            Case 0
                clrr = (clrr And &HFE) Or color_mask
            Case 1
                clrg = (clrg And &HFE) Or color_mask
            Case 2
                clrb = (clrb And &HFE) Or color_mask
        End Select

        ' Set the pixel's color.
        Picture1.PSet (r, c), RGB(clrr, clrg, clrb)

        byte_mask = byte_mask * 2
```

```
    Next i
End Sub


Private Function Decodelen(ByVal used_positions As Collection, ByVal wid As Integer, ByVal hgt
As Integer, ByVal show_pixels As Boolean) As Integer
Dim Value As Integer
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 14
        ' Pick a random pixel and RGB component.
        PickPosition used_positions, wid, hgt, r, c, pixel

        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb

        ' Get the stored value.
        Select Case pixel
            Case 0
                color_mask = (clrr And &H1)
            Case 1
                color_mask = (clrg And &H1)
            Case 2
                color_mask = (clrb And &H1)
        End Select

        If color_mask Then
            Value = Value Or byte_mask
        End If

        If show_pixels Then
            Picture1.PSet (r, c), RGB(0, 0, 0)

        End If

        byte_mask = byte_mask * 2
    Next i

    Decodelen = Value
End Function
```

Random lines

```
Option Explicit
Dim pixel()
Dim indi As Integer
Dim indj As Integer
Dim index As Integer

Private Sub aaa_Click()
Me.Enabled = False
about.Show
End Sub

Private Sub bb_Click()
Unload Me
Form0.Show
End Sub

Private Sub Command1_Click()
Dim img_size As Double
Dim j As Integer
Dim wid As Integer
Dim hgt As Integer
Dim msg As String
Dim pos_no As Integer
Dim col As Collection
Set col = New Collection
Dim x1 As Integer
Dim x2 As Integer
Dim y1 As Integer
Dim y2 As Integer
Dim line_len As Integer
Dim i As Integer
Dim flag As Boolean
Dim msg_len As Integer
Screen.MousePointer = vbHourglass
On Error Resume Next
If Check1.Value = 1 Then flag = True Else flag = False
Rnd -1
Randomize 0
msg_len = Len(Text1.Text)
img_size = Picture1.Width * Picture1.Height
'check if there are enough pixels in the selected image
If img_size < (msg_len * 8 + 14) Then MsgBox "The Image is too small for this message": Exit Sub
ReDim pixel(Picture1.Width, Picture1.Height)
'store the image in the matrix pixel
   For indi = 0 To Picture1.Width - 1
      For indj = 0 To Picture1.Height - 1
      pixel(indi, indj) = Picture1.Point(indi, indj)
      Next
    Next
```

```
msg = Left(Text1.Text, 16000)
wid = Picture1.ScaleWidth
hgt = Picture1.ScaleHeight
Set col = New Collection
'get the no of pixels required to encode the msg & the lenght of the msg
  pos_no = (Len(msg) * 8) + 14
j = 0
'generate location for the msg and length and store it in array mx2 and my2
Dim mx2() As Integer
Dim my2() As Integer
 While (j < pos_no)
 Dim mx() As Integer
    Dim my() As Integer
start:
Err.Clear
     'pick two points of the line
     PickPosition col, wid, hgt, x1, y1
     PickPosition col, wid, hgt, x2, y2
              dda mx, my, x1, y1, x2, y2
             ' If (dda_len(x1, y1, x2, y2) > 10) Then Exit Do

     For i = 0 To UBound(mx)
       If Err.Number <> 0 Then GoTo start
        If (check_pixel(col, mx(i), my(i)) = 1) And (j < pos_no) Then
         ReDim Preserve mx2(j + 1) As Integer
         ReDim Preserve my2(j + 1) As Integer
         mx2(j) = mx(i)
          my2(j) = my(i)
         j = j + 1
         'Picture1.PSet (mx(i), my(i)), vbBlack

        End If
    Next
 Wend
 index = 0
 'encode the msg length
 Encodelen Len(msg), mx2, my2, flag
'encode the msg
For i = 1 To Len(msg)
Call EncodeByte(Asc(Mid(msg, i, 1)), mx2, my2, flag)
Next
Command3.Enabled = True
Picture1.Picture = Picture1.Image
Screen.MousePointer = vbDefault

End Sub

Private Sub Command2_Click()
Dim ch As Byte
Dim msg_len As Byte
Dim j As Integer
Dim wid As Integer
Dim hgt As Integer
Dim msg As String
```

```
Dim pos_no As Integer
Dim col As Collection
Set col = New Collection
Dim x1 As Integer
Dim x2 As Integer
Dim y1 As Integer
Dim y2 As Integer
Dim line_len As Integer
Dim i As Integer
Dim flag As Boolean
Screen.MousePointer = vbHourglass
On Error Resume Next
If Check1.Value = 1 Then flag = True Else flag = False
Rnd -1
Randomize 0
j = 0
wid = Picture1.ScaleWidth
hgt = Picture1.ScaleHeight
Set col = New Collection
Dim mx2() As Integer
Dim my2() As Integer
Dim mx() As Integer
Dim my() As Integer
st1:
Err.Clear
    'pick two points of the line
     PickPosition col, wid, hgt, x1, y1
     PickPosition col, wid, hgt, x2, y2
        dda mx, my, x1, y1, x2, y2
     For i = 0 To UBound(mx)
       If Err.Number <> 0 Then GoTo st1
        If (check_pixel(col, mx(i), my(i)) = 1) Then
         ReDim Preserve mx2(j + 1) As Integer
         ReDim Preserve my2(j + 1) As Integer
         mx2(j) = mx(i)
          my2(j) = my(i)
         j = j + 1
          End If
    Next
index = 0
'decode the msg length
msg_len = Decodelen(mx2(), my2(), flag)
pos_no = ((msg_len * 8) + 14)
'generate the location for the msg
For i = 0 To UBound(mx)
        If (check_pixel(col, mx(i), my(i)) = 1) Then
         ReDim Preserve mx2(j + 1) As Integer
         ReDim Preserve my2(j + 1) As Integer
         mx2(j) = mx(i)
          my2(j) = my(i)
         j = j + 1
          End If
    Next
 While (j < pos_no)
```

```
    Dim mxx() As Integer
    Dim myy() As Integer
st2:
    Err.Clear
    'pick two points of the line
     PickPosition col, wid, hgt, x1, y1
     PickPosition col, wid, hgt, x2, y2
          dda mxx, myy, x1, y1, x2, y2
     For i = 0 To UBound(mxx)
       If Err.Number <> 0 Then GoTo st2
        If (check_pixel(col, mxx(i), myy(i)) = 1) And (j < pos_no) Then
         ReDim Preserve mx2(j + 1) As Integer
         ReDim Preserve my2(j + 1) As Integer
         mx2(j) = mxx(i)
          my2(j) = myy(i)
         j = j + 1


        End If
    Next
 Wend
 'decode the msg
 index = 14
 For i = 1 To msg_len
 ch = DecodeByte(mx2(), my2(), flag)
 msg = msg & Chr(ch)
 Next
Text1.Text = msg
Screen.MousePointer = vbDefault

End Sub

Private Sub Command3_Click()
Dim mse
Screen.MousePointer = vbHourglass
'calculate the mean square error
For indi = 0 To Picture1.Width - 1
        For indj = 0 To Picture1.Height - 1
        mse = mse + Abs(pixel(indi, indj) - Picture1.Point(indi, indj))
        Next
      Next
Label1.Caption = mse

Screen.MousePointer = vbDefault
End Sub

Private Sub ee_Click()
End
End Sub

Private Sub Form_Load()
cd1.Flags = &H2
End Sub
```

```
Private Sub hh_Click()
Me.Enabled = False
help.Show
End Sub

Private Sub o_Click()
cd1.InitDir = App.Path
cd1.ShowOpen
If cd1.FileName = "" Then Exit Sub
Picture1.Picture = LoadPicture(cd1.FileName)
Text1.Width = Picture1.Width + Picture1.Left
Command1.Enabled = True
Command2.Enabled = True
cd1.FileName = ""
End Sub

Private Sub UnRGB(ByVal color As OLE_COLOR, ByRef r As Byte, ByRef g As Byte, ByRef b
As Byte)
   r = color And &HFF&
   g = (color And &HFF00&) \ &H100&
   b = (color And &HFF0000) \ &H10000
End Sub

Private Sub ss_Click()
cd1.ShowSave
If cd1.FileName = "" Then Exit Sub
SavePicture Picture1.Picture, cd1.FileName
cd1.FileName = ""

End Sub

' Pick an unused (r, c, pixel) combination.
Private Sub PickPosition(ByVal used_positions As Collection, ByVal wid As Integer, ByVal hgt As
Integer, ByRef r As Integer, ByRef c As Integer)
Dim position_code As String

   On Error Resume Next
   Do
      ' Pick a position.
      r = Int(Rnd * wid)
      c = Int(Rnd * hgt)


      ' See if the position is unused.
      position_code = "(" & r & "," & c & ")"
      used_positions.Add position_code, position_code
      If Err.Number = 0 Then Exit Do
      Err.Clear
   Loop
End Sub

Private Sub dda(mx() As Integer, my() As Integer, x0 As Integer, y0 As Integer, x1 As Integer, y1
As Integer)
Dim dx As Integer
```

```
Dim dy As Integer
Dim X As Single
Dim Y As Single
Dim m As Single
Dim i As Integer
i = 0
If (x0 > x1) Or (y0 > y1) Then
Call swap(x0, x1)
Call swap(y0, y1)
End If
dx = x1 - x0
dy = y1 - y0
X = x0
Y = y0
If dx <> 0 Then m = dy / dx

If (dx = 0) Then 'VERTICAL LINE
Y = y0
While (Y <= y1)
ReDim Preserve mx(i + 1)
ReDim Preserve my(i + 1)
mx(i) = x0
my(i) = Y
i = i + 1
Y = Y + 1
Wend
ElseIf (dy = 0) Then 'HORIZONTAL LINE
While X <= x1
ReDim Preserve mx(i + 1)
ReDim Preserve my(i + 1)
mx(i) = X
my(i) = y0
i = i + 1
X = X + 1
Wend
ElseIf Abs(m) <= 1 Then
While X <= x1
ReDim Preserve mx(i + 1)
ReDim Preserve my(i + 1)
mx(i) = X
my(i) = Round(Y)
i = i + 1
Y = Y + m
X = X + 1
Wend
ElseIf Abs(m) > 1 Then
Y = y0
X = x0
While Y <= y1
ReDim Preserve mx(i + 1)
ReDim Preserve my(i + 1)
mx(i) = Round(X)
my(i) = Y
i = i + 1
```

```
X = X + (1 / m)
Y = Y + 1
Wend
End If

End Sub
Private Sub swap(X As Integer, Y As Integer)
Dim t As Integer
t = X
X = Y
Y = t
End Sub

Private Function check_pixel(ByVal used_positions As Collection, X As Integer, Y As Integer) As
Integer
Dim position_code As String

    On Error Resume Next



        ' See if the position is unused.
        position_code = "(" & X & "," & Y & ")"
        used_positions.Add position_code, position_code
        If Err.Number = 0 Then check_pixel = 1 Else check_pixel = 0
        Err.Clear

End Function

Private Sub EncodeByte(ByVal Value As Byte, mx() As Integer, my() As Integer, ByVal
show_pixels As Boolean)
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 8
        r = mx(index)
        c = my(index)
        index = index + 1

        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb
        If show_pixels Then
            clrr = 255

        End If
```

```
        ' Get the value we must store.
        If Value And byte_mask Then
            color_mask = 1
        Else
            color_mask = 0
        End If


                clrg = (clrg And &HFE) Or color_mask


        ' Set the pixel's color.
        Picture1.PSet (r, c), RGB(clrr, clrg, clrb)

        byte_mask = byte_mask * 2
    Next i
End Sub

Private Function DecodeByte(mx() As Integer, my() As Integer, ByVal show_pixels As Boolean) As
Byte
Dim Value As Integer
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 8

        r = mx(index)
        c = my(index)
        index = index + 1


        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb

        ' Get the stored value.

                color_mask = (clrg And &H1)


        If color_mask Then
            Value = Value Or byte_mask
        End If

        If show_pixels Then
            Picture1.PSet (r, c), vbBlack
```

```
        End If

        byte_mask = byte_mask * 2
    Next i

    DecodeByte = CByte(Value)
End Function

Private Sub Encodelen(ByVal Value As Integer, mx() As Integer, my() As Integer, ByVal
show_pixels As Boolean)
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 14
        r = mx(index)
        c = my(index)
        index = index + 1

        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb
        If show_pixels Then
            clrr = 255

        End If

        ' Get the value we must store.
        If Value And byte_mask Then
            color_mask = 1
        Else
            color_mask = 0
        End If


              clrg = (clrg And &HFE) Or color_mask


        ' Set the pixel's color.
        Picture1.PSet (r, c), RGB(clrr, clrg, clrb)

        byte_mask = byte_mask * 2
    Next i
End Sub
Private Function Decodelen(mx() As Integer, my() As Integer, ByVal show_pixels As Boolean) As
Byte
Dim Value As Integer
Dim i As Integer
```

```
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 14

        r = mx(index)
        c = my(index)
        index = index + 1


        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb

        ' Get the stored value.

              color_mask = (clrg And &H1)


        If color_mask Then
            Value = Value Or byte_mask
        End If

        If show_pixels Then
            Picture1.PSet (r, c), vbBlack

        End If

        byte_mask = byte_mask * 2
    Next i

    Decodelen = Value
End Function
'programed by Husam namir & Omar Fitian
```

User selected random points

```
Option Explicit
Dim pixel()
Dim indi As Integer
Dim indj As Integer

Private Sub aaaaaaa_Click()
Me.Enabled = False
about.Show
End Sub

Private Sub bb_Click()
Unload Me
Form0.Show
End Sub

Private Sub Command1_Click()
Dim msg_len As Integer
Dim wid As Integer
Dim hgt As Integer
Dim i As Integer
Dim msg As String
Dim used_positions As Collection
Dim flag As Boolean
Dim Value As Byte
Dim value1 As Integer
Dim img_size
Dim x1 As Integer
Dim y1 As Integer
Dim x2 As Integer
Dim y2 As Integer


If Label5 = "" Then MsgBox "You Must Select The Area You Wish to Encode in to First": Exit Sub
x1 = Label1
y1 = Label3
x2 = Label4
y2 = Label5
If (x1 > x2) Or (y1 > y2) Then
MsgBox "Select another two points"
Label1 = ""
Label3 = ""
Label4 = ""
Label5 = ""
Exit Sub
End If


Screen.MousePointer = vbHourglass
If Check1.Value = 1 Then flag = True Else flag = False
wid = Picture1.ScaleWidth
hgt = Picture1.ScaleHeight
```

```
img_size = (Abs(x1 - x2)) * (Abs(y1 - y2))
msg = Left(Text1.Text, 16000)
msg_len = Len(msg)
ReDim pixel(Picture1.Width, Picture1.Height)
'check if there are enough pixels in the selected image
If img_size < (msg_len * 8) Then MsgBox "The Selected area  is too small for this message": Exit
Sub

 'store the image in the matrix pixel
    For indi = 0 To Picture1.Width - 1
        For indj = 0 To Picture1.Height - 1
        pixel(indi, indj) = Picture1.Point(indi, indj)
        Next
      Next


    ' Initialize the random number generator.
    Rnd -1
    Randomize 0
    'generate random locations for the msg and the msg length

    Set used_positions = New Collection
    'encode the msg length
value1 = msg_len
Encodelen value1, used_positions, wid, hgt, flag
  'encode the two points
Encodelen x1, used_positions, wid, hgt, flag
Encodelen y1, used_positions, wid, hgt, flag
Encodelen x2, used_positions, wid, hgt, flag
Encodelen y2, used_positions, wid, hgt, flag

  'encode the msg
  wid = Abs(x1 - x2)
  hgt = Abs(y1 - y2)
   For i = 1 To msg_len
    EncodeByte Asc(Mid(msg, i, 1)), used_positions, wid, hgt, flag, x1, y1
    Next
Picture1.Picture = Picture1.Image

Command3.Enabled = True
Screen.MousePointer = vbDefault

End Sub

Private Sub Command2_Click()
Dim msglen As Integer
Dim wid As Integer
Dim hgt As Integer
Dim wid2 As Integer
Dim hgt2 As Integer
Dim i As Integer
Dim msg As String
Dim used_positions As Collection
Dim flag As Boolean
```

```
Dim ch As String
Dim x1 As Integer
Dim y1 As Integer
Dim x2 As Integer
Dim y2 As Integer
Screen.MousePointer = vbHourglass

If Check1.Value = 1 Then flag = True Else flag = False
wid = Picture1.ScaleWidth
hgt = Picture1.ScaleHeight

 ' Initialize the random number generator.
    Rnd -1
    Randomize 0
    'generate random locations for the msg and the msg length

    Set used_positions = New Collection

    'decode the msg length
    msglen = Decodelen(used_positions, wid, hgt, flag)
    x1 = Decodelen(used_positions, wid, hgt, flag)
   y1 = Decodelen(used_positions, wid, hgt, flag)
    x2 = Decodelen(used_positions, wid, hgt, flag)
    y2 = Decodelen(used_positions, wid, hgt, flag)
 wid2 = Abs(x1 - x2)
  hgt2 = Abs(y1 - y2)
 For i = 1 To msglen
 ch = DecodeByte(used_positions, wid2, hgt2, flag, x1, y1)

 msg = msg & Chr(ch)


  Next
 Text1.Text = msg
 Screen.MousePointer = vbDefault
End Sub

Private Sub Command3_Click()
Dim mse
Screen.MousePointer = vbHourglass
'calculate the mean square error
For indi = 0 To Picture1.Width - 1
        For indj = 0 To Picture1.Height - 1
        mse = mse + Abs(pixel(indi, indj) - Picture1.Point(indi, indj))
        Next
      Next
Label2.Caption = mse

Screen.MousePointer = vbDefault
End Sub

Private Sub Command4_Click()
Label1 = ""
Label3 = ""
```

```
Label4 = ""
Label5 = ""
End Sub

Private Sub ee_Click()
End
End Sub

Private Sub Form_Load()
cd1.Flags = &H2
End Sub

Private Sub hh_Click()
Me.Enabled = False
help.Show
End Sub

Private Sub o_Click()
cd1.InitDir = App.Path
cd1.ShowOpen
If cd1.FileName = "" Then Exit Sub
Picture1.Picture = LoadPicture(cd1.FileName)
Text1.Width = Picture1.Width + Picture1.Left
Command1.Enabled = True
Command2.Enabled = True
cd1.FileName = ""
End Sub

Private Sub UnRGB(ByVal color As OLE_COLOR, ByRef r As Byte, ByRef g As Byte, ByRef b
As Byte)
   r = color And &HFF&
   g = (color And &HFF00&) \ &H100&
   b = (color And &HFF0000) \ &H10000
End Sub

Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

If Button = 1 Then
If Label1 = "" Then
Label1 = X: Label3 = Y
ElseIf Label4 = "" Then
Label4 = X: Label5 = Y
End If
End If
End Sub

Private Sub ss_Click()
cd1.ShowSave
If cd1.FileName = "" Then Exit Sub
SavePicture Picture1.Picture, cd1.FileName
cd1.FileName = ""
End Sub

' Pick an unused (r, c, pixel) combination.
```

```
Private Sub PickPosition(ByVal used_positions As Collection, ByVal wid As Integer, ByVal hgt As Integer, ByRef r As Integer, ByRef c As Integer, ByRef pixel As Integer)
Dim position_code As String

    On Error Resume Next
    Do
        ' Pick a position.
        r = Int(Rnd * wid)
        c = Int(Rnd * hgt)
        pixel = Int(Rnd * 3)

        ' See if the position is unused.
        position_code = "(" & r & "," & c & "," & pixel & ")"
        used_positions.Add position_code, position_code
        If Err.Number = 0 Then Exit Do
        Err.Clear
    Loop
End Sub
Private Sub PickPosition2(ByVal used_positions As Collection, ByVal wid As Integer, ByVal hgt As Integer, ByRef r As Integer, ByRef c As Integer, ByRef pixel As Integer, x1 As Integer, y1 As Integer)
Dim position_code As String

    On Error Resume Next
    Do
        ' Pick a position.
        r = Int(Rnd * wid) + x1
        c = Int(Rnd * hgt) + y1
        pixel = Int(Rnd * 3)

        ' See if the position is unused.
        position_code = "(" & r & "," & c & "," & pixel & ")"
        used_positions.Add position_code, position_code
        If Err.Number = 0 Then Exit Do
        Err.Clear
    Loop
End Sub

Private Function DecodeByte(ByVal used_positions As Collection, ByVal wid As Integer, ByVal hgt As Integer, ByVal show_pixels As Boolean, x1 As Integer, y1 As Integer) As Byte
Dim Value As Integer
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 8
        ' Pick a random pixel and RGB component.
```

```
        PickPosition2 used_positions, wid, hgt, r, c, pixel, x1, y1

        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb

        ' Get the stored value.
        Select Case pixel
            Case 0
                color_mask = (clrr And &H1)
            Case 1
                color_mask = (clrg And &H1)
            Case 2
                color_mask = (clrb And &H1)
        End Select

        If color_mask Then
            Value = Value Or byte_mask
        End If

        If show_pixels Then
            Picture1.PSet (r, c), RGB(0, 0, 0)

        End If

        byte_mask = byte_mask * 2
    Next i

    DecodeByte = CByte(Value)
End Function


Private Sub EncodeByte(ByVal Value As Byte, ByVal used_positions As Collection, ByVal wid As
Integer, ByVal hgt As Integer, ByVal show_pixels As Boolean, x1 As Integer, y1 As Integer)
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 8
        ' Pick a random pixel and RGB component.
        PickPosition2 used_positions, wid, hgt, r, c, pixel, x1, y1

        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb
        If show_pixels Then
            clrr = 255
            clrg = clrg And &H1
            clrb = clrb And &H1
```

```
        End If

        ' Get the value we must store.
        If Value And byte_mask Then
            color_mask = 1
        Else
            color_mask = 0
        End If

        ' Update the color.
        Select Case pixel
            Case 0
                clrr = (clrr And &HFE) Or color_mask
            Case 1
                clrg = (clrg And &HFE) Or color_mask
            Case 2
                clrb = (clrb And &HFE) Or color_mask
        End Select

        ' Set the pixel's color.
        Picture1.PSet (r, c), RGB(clrr, clrg, clrb)

        byte_mask = byte_mask * 2
    Next i
End Sub


Private Sub Encodelen(ByVal Value As Integer, ByVal used_positions As Collection, ByVal wid As
Integer, ByVal hgt As Integer, ByVal show_pixels As Boolean)
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 14
        ' Pick a random pixel and RGB component.
        PickPosition used_positions, wid, hgt, r, c, pixel

        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb
        If show_pixels Then
            clrr = 255
            clrg = clrg And &H1
            clrb = clrb And &H1
        End If

        ' Get the value we must store.
        If Value And byte_mask Then
```

```
            color_mask = 1
        Else
            color_mask = 0
        End If

        ' Update the color.
        Select Case pixel
            Case 0
                clrr = (clrr And &HFE) Or color_mask
            Case 1
                clrg = (clrg And &HFE) Or color_mask
            Case 2
                clrb = (clrb And &HFE) Or color_mask
        End Select

        ' Set the pixel's color.
        Picture1.PSet (r, c), RGB(clrr, clrg, clrb)

        byte_mask = byte_mask * 2
    Next i
End Sub



Private Function Decodelen(ByVal used_positions As Collection, ByVal wid As Integer, ByVal hgt
As Integer, ByVal show_pixels As Boolean) As Integer
Dim Value As Integer
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 14
        ' Pick a random pixel and RGB component.
        PickPosition used_positions, wid, hgt, r, c, pixel

        ' Get the pixel's color components.
        UnRGB Picture1.Point(r, c), clrr, clrg, clrb

        ' Get the stored value.
        Select Case pixel
            Case 0
                color_mask = (clrr And &H1)
            Case 1
                color_mask = (clrg And &H1)
            Case 2
                color_mask = (clrb And &H1)
        End Select
```

```
        If color_mask Then
            Value = Value Or byte_mask
        End If

        If show_pixels Then
            Picture1.PSet (r, c), RGB(0, 0, 0)

        End If

        byte_mask = byte_mask * 2
    Next i

    Decodelen = Value
End Function

'programed by Husam namir & Omar Fitian
```

REFERENCES

- *"Steganography"* by Markus Kuhn, Steganography Mailing List.
- "*Network and Internetwork Security*" by William Stallings, Addison-Wesley.
- *"Steganography"* by Dorian A. Flowers Xavier, University of Louisiana
- "*Privacy on the net - steganography*", Michael Berkowitz.