Steganography

What is our GOAL for this CLASS?

In this class, we learned about steganography and how to hide data in image.

What did we ACHIEVE in the class TODAY?

- We Understood about Steganography
- We learned to hide data in image
- We learned to decrypt data from image

Which CONCEPTS/ CODING BLOCKS did we cover today?

- We used the libraries, smtplib,ssl
- We used key events to capture

Understanding concepts:

Steganography:
Steganography is the practice of hiding a secret message within (or on top of) something that is not secret. The message can be anything you want. Steganography involves embedding a secret piece of text inside of a picture or by hiding a secret message or script within a Word or Excel document.

How did we DO the activities?

1. Understand the boiler plate code:
    - from **PIL** import **image** The **Python Imaging Library Pillow** or **PIL** adds **image processing** capabilities to the Python interpreter.
    - Create a function to convert any type of data into binary, we can use this to convert secret data and pixel values in the encoding and decoding phase.
    - Create one function **genData()**, pass the argument **data**. This function will take the secret message and convert that message into **ASCII** codes using **ord()** function
    - Create new array **newd = []**
    - Using for loop check the index of all characters and convert into ASCII using **ord** and then and using **append()** method add into **newd** array and then return after conversion

```
from PIL import Image
def genData(data):

                # list of binary codes
                # of given data
                newd = []

                for i in data:
                        newd.append(format(ord(i), '08b'))
                return newd
```

2. Create the **main()** function

    - Take input from the user and save that in variable **a**
    - **If** user will press **a==1**, then do **encode**
    - **elif** user will press **a==2** then do **decode**
    - **If** user press anything else, then raise **exception**
    - Call the **main()** function

```
def main():
        a = int(input(":: Welcome to Steganography ::\n"
                                                "1. Encode\n2. Decode\n"))
        if (a == 1):
                encode()

        elif (a == 2):
                print("Decoded Word : " + decode())
        else:
                raise Exception("Enter correct input")

if __name__ == '__main__' :

        # Calling main function
        main()
```

3. Create function that takes the input image name and secret message from the user and calls **encode_enc ()** function to encode it
    - Create **img** variable and save input image in this variable, using **open()** method read the image using **"r"**
    - Create **data** variable and save hidden message in this variable
    - Check if length of data is zero then raise exception and print data is empty
    - Create **new_img_name** variable and that will save new image name
    - Call the **main()** function

```
def encode():
        img = input("Enter image name(with extension) : ")
        image = Image.open(img, 'r')

        data = input("Enter data to be encoded : ")
        if (len(data) == 0):
                raise ValueError('Data is empty')

        newimg = image.copy()
        encode_enc(newimg, data)

        new_img_name = input("Enter the name of new image(with extension) : ")
        newimg.save(new_img_name, str(new_img_name.split(".")[1].upper()))

        main()
```

4. Create a function to hide secret message into the image by altering the **LSB**
    - For each character in the data, its **ASCII** value is taken and converted into **8-bit binary**
    - Three pixels are read at a time having a total of **3*3=9 RGB** values. The first eight RGB values are used to store one character that is converted into an 8-bit binary. The corresponding RGB value and binary data are compared. If the binary digit is 1 then the RGB value is converted to odd and, otherwise, even.

```
def modPix(pix, data):

        datalist = genData(data)
        lendata = len(datalist)
        imdata = iter(pix)

        for i in range(lendata):

                pix = [value for value in imdata.__next__()[:3] +
                                                        imdata.__next__()[:3]
                                                        imdata.__next__()[:3]]
                for j in range(0, 8):
                        if (datalist[i][j] == '0' and pix[j]% 2 != 0):
                                pix[j] -= 1

                        elif (datalist[i][j] == '1' and pix[j] % 2 == 0):
                                if(pix[j] != 0):
                                        pix[j] -= 1
                                else:
                                        pix[j] += 1
                                # pix[j] -= 1


                if (i == lendata - 1):
                        if (pix[-1] % 2 == 0):
                                if(pix[-1] != 0):
                                        pix[-1] -= 1
                                else:
                                        pix[-1] += 1

                else:
                        if (pix[-1] % 2 != 0):
                                pix[-1] -= 1

                pix = tuple(pix)
                yield pix[0:3]
                yield pix[3:6]
                yield pix[6:9]
```

5. Create a function **encode_enc()**
    - Get the size of the **image()**
    - Save the coordinates
    - Using **if** condition check if more **data** is to be read
    - If **yes** then make it **even**
    - If **no** make it **odd**

```
def encode_enc(newimg, data):
        w = newimg.size[0]
        (x, y) = (0, 0)

        for pixel in modPix(newimg.getdata(), data):

                # Putting modified pixels in the new image
                newimg.putpixel((x, y), pixel)
                if (x == w - 1):
                        x = 0
                        y += 1
                else:
                        x += 1
```

6. Create a **function** to decode the hidden message from the **stego** image.
   - Create variable **img** that will ask the **stego** image
   - Using **open ()** and **" r "**
   - Create one empty variable **data**
   - The Python **iter()** function returns an **iterator** for the given object. The **iter()** function creates an object which can be iterated one element at a time.
   - **getdata()** returns the contents of this **image** as a sequence object containing pixel values.
   - It will take the one pixel value at time and then move to next three one by one
   - Check the **8 bits** , if it is divisible by 2 then return **binstr "0"**
   - If not then make it **"1"**
   - Convert **ascii** data into character again i.e readable form using **chr()**
   - And then return the **data**

```
def decode():
        img = input("Enter image name(with extension) : ")
        image = Image.open(img, 'r')

        data = ''
        imgdata = iter(image.getdata())

        while (True):
                pixels = [value for value in imgdata.__next__()[:3] +
                                                        imgdata.__next__()[:3] +
                                                        imgdata.__next__()[:3]]


                binstr = ''

                for i in pixels[:8]:
                        if (i % 2 == 0):
                                binstr += '0'
                        else:
                                binstr += '1'

                data += chr(int(binstr, 2))
                if (pixels[-1] % 2 != 0):

                        return data
```

What's NEXT?

In the next class we will learn about SQL

Expand Your Knowledge

To know more about click here