

Image Steganography

Tanzir Hasan and Rohan Saha

Our Repo

<https://github.com/Tanzebruh/ImageSteganography.git>

If you want to follow along you will need pip. If you don't have it, just watch the screen, participate in the poll, and enjoy our presentation.

If you do have pip, do `pip install -r requirements.txt`



What is Steganography?

Steganography is the process of hiding information in plain sight to protect it from an unwanted third party.

Steganography can hide many different types of data including text, music, images and more in places no one would think to look at.

Our project deals with hiding messages in images by manipulating the least significant bit in their RGB values to hide messages in binary.

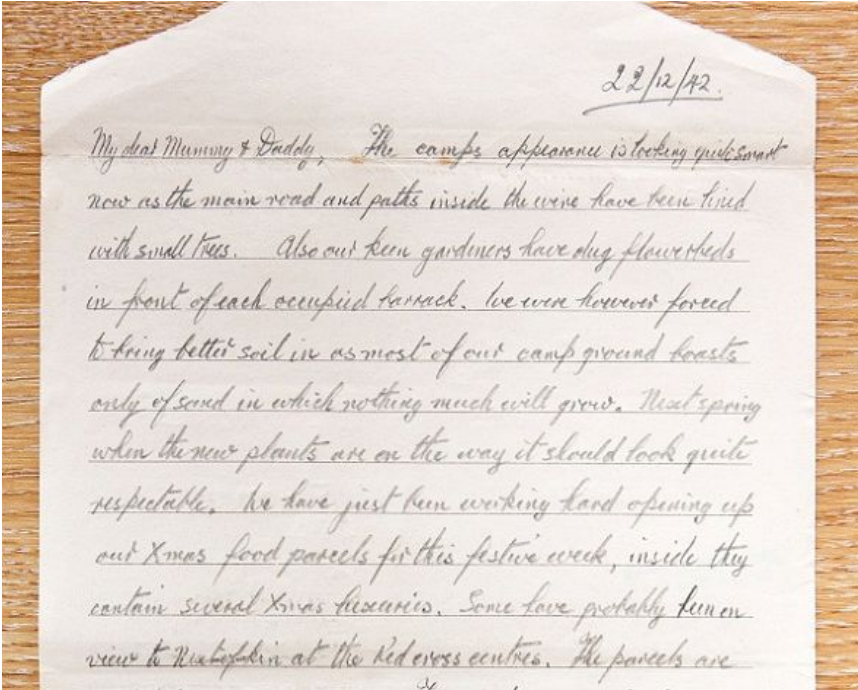


Examples of Steganography used to send messages

The first known form of steganography was used by the Greek tyrant Histiaeus. He had the head of a servant shaved and tattooed with a warning about an attack. The servant was sent out as a messenger once his hair grew back.

Captured British soldiers used steganography in their letters to their families so the messages couldn't be picked up by Nazi censors.

Italian members of al-Qaeda were put on trial for communicating through porn that had important blueprints encoded into them.

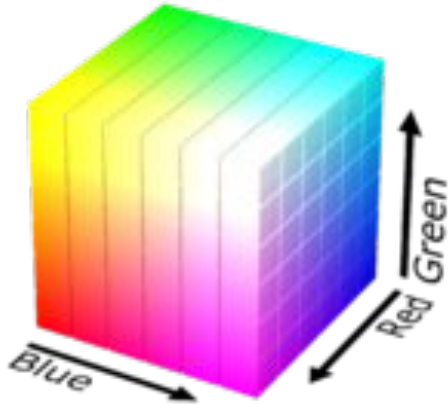


22/12/42.

My dear Mummy & Daddy, The camp appearance is looking quite smart now as the main road and paths inside the wire have been lined with small trees. Also our keen gardeners have dug flour beds in front of each occupied barrack. We were however forced to bring better soil in as most of our camp ground boasts only of sand in which nothing much will grow. Next spring when the new plants are on the way it should look quite respectable. We have just been working hard opening up our Xmas food parcels for this festive week, inside they contain several Xmas biscuits. Some have probably been en route to Wembley at the Red cross centres. The parcels are

W

How Image Steganography Works



- Images are made up pixels. Each pixel in an image is assigned an RGB value to indicate its color (ranging from 0-255). RGB values are denoted as such:

(Red-Value, Green-Value, Blue-Value)

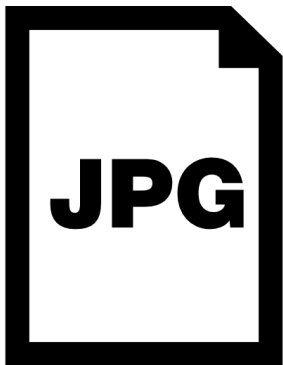
- Each of these values are made up of 8 bits. The first bit is considered to be the most significant bit because it changing its value would have the most significant impact on the final value. The reverse holds true for the least significant bit which is the last bit of any value (Red, Green, or Blue).

-By altering the LSBs of a consecutive set of pixels, we can encode a secret message without creating changes on the image noticeable to the naked eye.

Difference between .jpg and .png

Before we get into Steganography, you need to learn the difference between .jpg and .png.

Look in the chat, we'll have a poll. Thumbs up your favorite file extension, and we promise we won't judge you for it.



.jpg



PNGs are objectively better

.png



The .jpg file extension means that there is lossy conversion. The least significant bit is removed, so if you were to try and hide a message using that bit, it will be deleted. So, if you ever want to use this particular method of image steganography, you have to save it with the .png file extension.

Our Code: Encoding

1. Choose an RGB value to manipulate
2. Take in a message and add a stop sign
3. Turn the message into binary
4. Take in an image and turn it into a numpy array while changing the least significant bit to 0
5. Go from pixel to pixel changing that RGB value and adding the binary at that position.
6. Save the image as .png

<https://github.com/Tanzebruh/ImageSteganography.git>

```
def userInput(msg):  
    msg += "aspcv"  
    binary = ''  
    for i in msg:  
        nbinary = format(ord(i), 'b')  
        binary = binary + '0' * (8-len(nbinary)) + nbinary  
    return binary
```

```
for i in range(len(binary)):  
    h = i%height  
    w = i//height  
    if (numpy_array[w, h, color]%2 != 0):  
        numpy_array[w, h, color] = numpy_array[w, h, color] - 1  
        numpy_array[w, h, color] += int(binary[w + h])  
PIL_image = Image.fromarray(numpy_array.copy().astype(np.uint8))  
PIL_image.save(res_file)
```


Our Code: Decoding

1. Turn the image being decoded into a numpy array
2. Create an empty string variable for your binary
3. Turn the least significant bit into a string and add it to the binary string
4. Convert your binary string into a character every 8 bits and add the character to an answer string
5. If your program picks up the stop code, stop decoding
6. Print the answer

```
Dimage = Image.open(filename)
width, height = Dimage.size
numpy_array = np.array(Dimage)
binary = ""
answer = ""
found = False
for w in range(width):
    if (not found):
        for h in range(height):
            binary += str(numpy_array[w][h][color]%2)
            if (len(binary)==8):
                answer += chr(int(binary,2))
                binary = ""
            if ("aspcv" in answer):
                found = True
                break
print (answer[:len(answer)-6])
```

A Demonstration

First start encoding:

```
python stegoCode.py encode image_name text_file new_image_name color
```

```
python stegoCode.py encode green.jpg test.txt test.png 0
```

To decode enter this through the terminal:

```
python stegoCode.py decode image_name color
```

```
python stegoCode.py decode test.png 0
```

Before Encoding



This is green.jpg in our repo

We chose an image with only one color to show you how subtle the changes are

Encoded Image



This is demonstration.png from our repo

There is no visible difference between the two images

This is only a little of what you can do with Steganography. You can start inputting data at a certain pixel, have signs that signal that parts of the message are ciphered and so much more.

Example: When hiding a message inside an image you can start at the 3rd pixel instead of the first and have multiple stop switches. That way, only a person who knows where to start searching can find out what you hid.

Steganography is limited almost exclusively by creativity. We talked about hiding messages, but you can also hide malware.

Malicious Image Steganography

In 2016 hackers injected Visbot malware into an e-commerce platform named Magento.

The malware logged and encrypted user information, then sent it all back to the hackers in the form of a PNG file.

The entire process was incredibly hard to discover because no one thought to look for malicious images.

Another example of this would be having scripts that are automatically run by simply opening pdf or word files



Works Cited

<https://nakedsecurity.sophos.com/2013/05/15/and-the-winner-of-the-world-war-two-steganography-competition-is/>

<https://www.bleepingcomputer.com/news/security/github-hosted-malware-calculate-s-cobalt-strike-payload-from-imgur-pic/>

<https://towardsdatascience.com/hiding-data-in-an-image-image-steganography-using-python-e491b68b1372>

https://www.theregister.com/2003/05/12/alqaeda_said_to_be_using/

<https://www.bbc.com/news/world-24784756>

<https://www.bbntimes.com/technology/why-the-cybersecurity-industry-should-be-concerned-about-steganography>