Steganografija

Projekat za predmet Naučno izračunavanje

Steganografija

nastaje kao ideja skrivanja informacije koja se prenosi unutar nekog medija ili skupa podataka, kao što to danas najčešće podrazumeva neke multimedijalne datoteke (slike, audio ili video datoteke)

Prvi put se pojavlja krajem 15. veka. Reč 'steganogragija' znači sakriveno pisanje. Ideja takvog pisanja je postojala jos 440. godine p. n. e.

-pisanje na kurirovom telu
-nevidljivo mastilo
-mikrofotografije i mikrotekst
-nulta šifra

Projekat

- svaki piksel slike predstavljen je kao skup od tri 8-bitna broja
- bitove najmanje težine slike prenosioca menjamo bitovima najveće tezine skrivene poruke ili slike
- korišćena je biblioteka Pillow i Image modul koji predstavlja alat za manipulaciju sa slikama

Slika u slici

```
def encrypt(image_holder, image_secret):
    #If image size (with and hight given as a tuple) is bigger then holder image, than we want to exit
    h_weight, h_hight = get_size(image_holder)
    s_weight, s_hight = get_size(image_secret)
    if s_weight > h_weight or s_weight > h_hight:
        raise ValueError('Secret image is bigger than holder image!')

    #We can see RGB values for all pixcels with command: #print np.asarray(image_holder)

    #Pixel maps for both images
    image_holder_pixels = image_holder.load()
    image_secret_pixels = image_secret.load()

    #Output image will be the same as holder, and then we are going to change it's pixels
    #importing part of pixels of secret image
    output_image = copy_image(image_holder)
    output_image_pixels = output_image.load()

    for i in range(h_weight):
        for j in range(h_hight):
            rgb_binary_holder = Steganography_image.get_rgb(image_holder_pixels[i,j])

            #Check if the secret image is finished because it is smaller than holder image
            if(s_weight > i and s_hight > j):
                rgb_binary_secret = Steganography_image.get_rgb(image_secret_pixels[i,j])
            else:
                rgb_binary_secret = Steganography_image.get_rgb((0, 0, 0))

            #Now we have rgb binary values for holder and secret image
            #Then we want to place secret image into holder one into last 4 bits
            r = rgb_binary_holder[0][:4] + rgb_binary_secret[0][:4]
            g = rgb_binary_holder[1][:4] + rgb_binary_secret[1][:4]
            b = rgb_binary_holder[2][:4] + rgb_binary_secret[2][:4]
            output_image_pixels[i, j] = (int(r, 2), int(g, 2), int(b, 2))

    image_holder.close()
    image_secret.close()
    return output_image
```

Slika u slici

```
def decrypt(image):
    #Pixel map for image
    image_pixels = image.load()

    weight, hight = get_size(image)
    size = image.size

    #Output image will be the same as holder, and then we are going to change it's pixels
    #importing part of pixels of secret image only. After that we will crop extra pixels
    output_image = copy_image(image)  #Image.new(image.mode, size, (0, 0, 0))
    output_image_pixels = output_image.load()

    #Variables for the size of secret image
    k = 0
    l = 0

    #Making rgb map and checking if all pixels are zero then there is no hidden picture
    for i in range(weight):
        for j in range(hight):
            rgb_binary = Steganography_image.get_rgb(image_pixels[i,j])

            #Now we have rgb binary values
            #Then we want to get secret image from holder image by taking last 4 bits
            r = rgb_binary[0][4:] + '0000'
            g = rgb_binary[1][4:] + '0000'
            b = rgb_binary[2][4:] + '0000'
            output_image_pixels[i, j] = (int(r, 2), int(g, 2), int(b, 2))

            #If all parts are zero, then there is no secret picture merged on that pixel.
            #We are seting size of secret picture, last pixel that is not zero is end of the secret image
            #so we are saving that pixel indexes in k and l.
            #Last number 255 is always 255, that is made in code above output_image.load()
            if output_image_pixels[i, j] == (0, 0, 0, 255):
                pass
            else:
                k = i
                l = j

    output_weight = k + 1
    output_hight = l + 1
    #Crop picture from left upper corner to size that we found (size of secret image)
    output_image = output_image.crop((0, 0, output_weight, output_hight))

    image.close()
    return output_image
```

Tekst u slici

```
def encrypt(image_name, message):
    image = Image.open(image_name + '.png')
    image_pixels = image.load()

    #Generating binary representation of whole text
    message_bin = []
    message_len = len(message)
    for i in range(message_len):
        message_bin.append(Steganography_text.get_binary_text(message[i]))

    #Output image with secret text is the same as forwarded image at first
    output_image = copy_image(image)
    output_image_pixels = output_image.load()
    weight, hight = get_size(output_image)

    #k is counter for number of letters in message
    k = 0
    for i in range(weight):
        for j in range(hight):
            rgb_binary = Steganography_image.get_rgb(image_pixels[i,j])

            #If message is not finished take binary representation of letter,
            #else zero binary pixel.
            #Representation of the text made like this is getting into image
            if(message_len > 0):
                rgb_binary_secret =  message_bin[k]
            else:
                rgb_binary_secret = '00000000'

            #In one pixel of image we are putting one letter in parts (3, 3 and 2 bits)
            #at the end of each rgb part of the image, creating new rgb presentation
            r = rgb_binary[0][:5] + rgb_binary_secret[:3]
            g = rgb_binary[1][:5] + rgb_binary_secret[3:6]
            b = rgb_binary[2][:6] + rgb_binary_secret[6:8]
            output_image_pixels[i, j] = (int(r, 2), int(g, 2), int(b, 2))

            k += 1
            message_len -= 1

            output_image_pixels[i, j] = (int(r, 2), int(g, 2), int(b, 2))

    return output_image
```

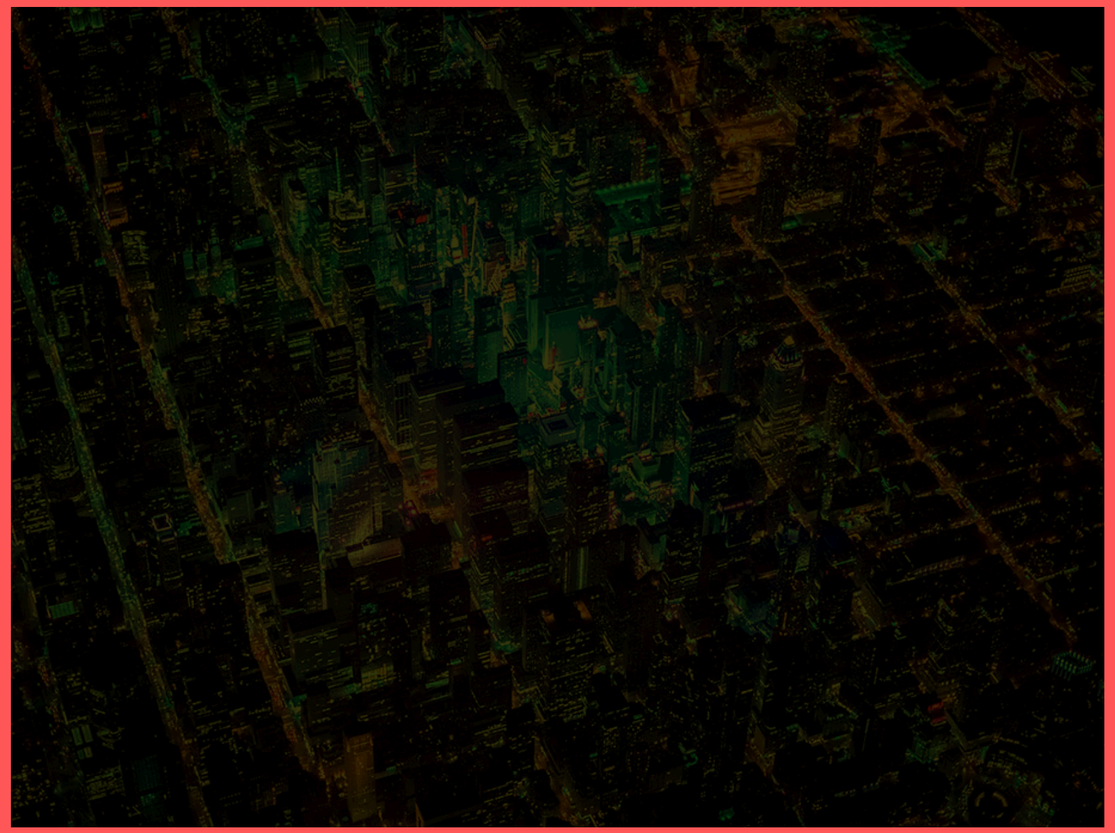
Tekst u slici

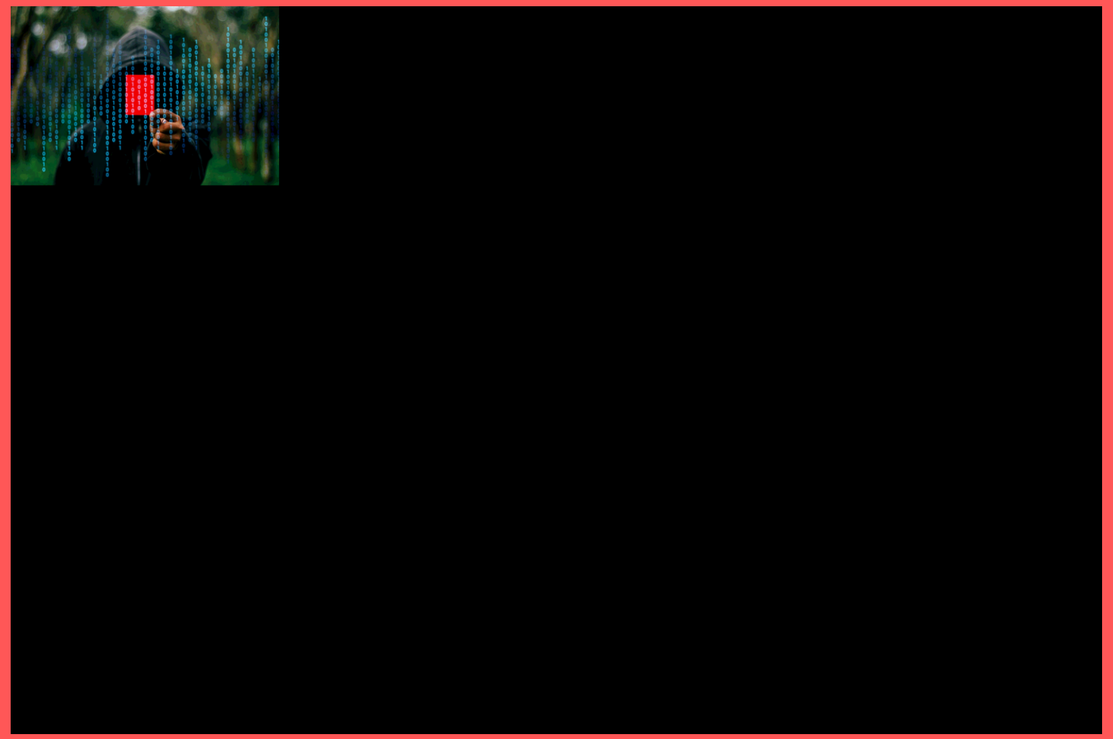
```
def decrypt(image):
    #Pixel map for image
    image = Image.open(image + '.png')
    image_pixels = image.load()

    weight, hight = get_size(image)
    size = image.size

    #Output is the message
    message = ""

    #If, after manipulating, pixel is zero then there is no secret message into that pixel
    for i in range(weight):
        for j in range(hight):
            rgb_binary = Steganography_image.get_rgb(image_pixels[i,j])

            #Now we want to take part of the pixels that will maybe be letter from message
            m = rgb_binary[0][5:] + rgb_binary[1][5:] + rgb_binary[2][6:]

            #If new pixel is zero, then there is no secret message
            if m != '00000000':
                message += Steganography_text.get_ascii(m)

    return message
```

Rezultat - slika nosač

Rezultat - tajna slika

Neuspešni pokušaji





Hvala na pažnji