

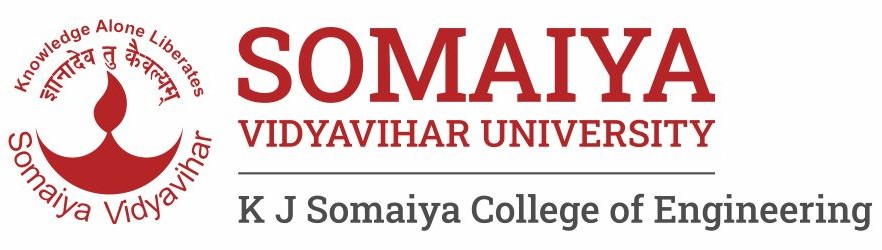
# KJ SOMAIYA COLLEGE OF ENGINEERING

**DIGITAL IMAGE PROCESSING**

A PROJECT REPORT ON

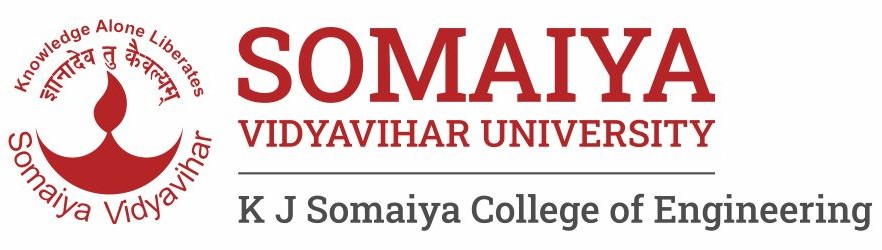
DATA HIDING USING STEGANOGRAPHY

**Vedant Kelkar - 1912052**



***INDEX***

|  |  |  |
| --- | --- | --- |
| **SR No.** | **CONTENTS** | **PageNo.** |
| **1** | **ABSTRACT** | 3 |
| **2** | **INTRODUCTION** | 3 |
| **3** | **ALGORITHM** | 4 |
| **4** | **CODE** | 6 |
| **5** | **OUTPUT** | 8 |



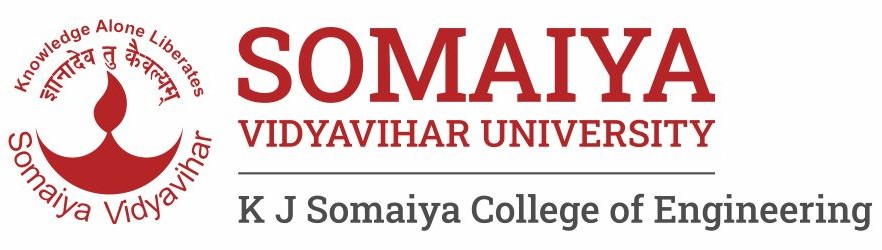
Steganography is the study of encoding hidden data in a suitable multimedia carrier, such as an image, audio, or video file. It is constantly concealed. The initial signal is less important in this message. Steganography can be used for a variety of purposes. Steganography's key goals are undetectability, resilience (resistance to various image processing methods and compression), and hidden data capacity. These are the major characteristics that distinguish it from other watermarking and cryptography systems. The main focus of this study is on a review of steganography in digital photographs, which includes the most important steganography approaches. In the world, data privacy and security are always a top priority. We need a dependable mechanism to encrypt the data so that it can safely reach its destination. Encryption is a basic but effective method of safeguarding our data while it is being transmitted to a destination. The proposed solution uses cutting-edge steganography and encryption technology. This research proposes a new method for concealing data in speech signals. Audio steganography is used to hide a ten-digit number within a speech signal, which is then encrypted with a unique key for added security.

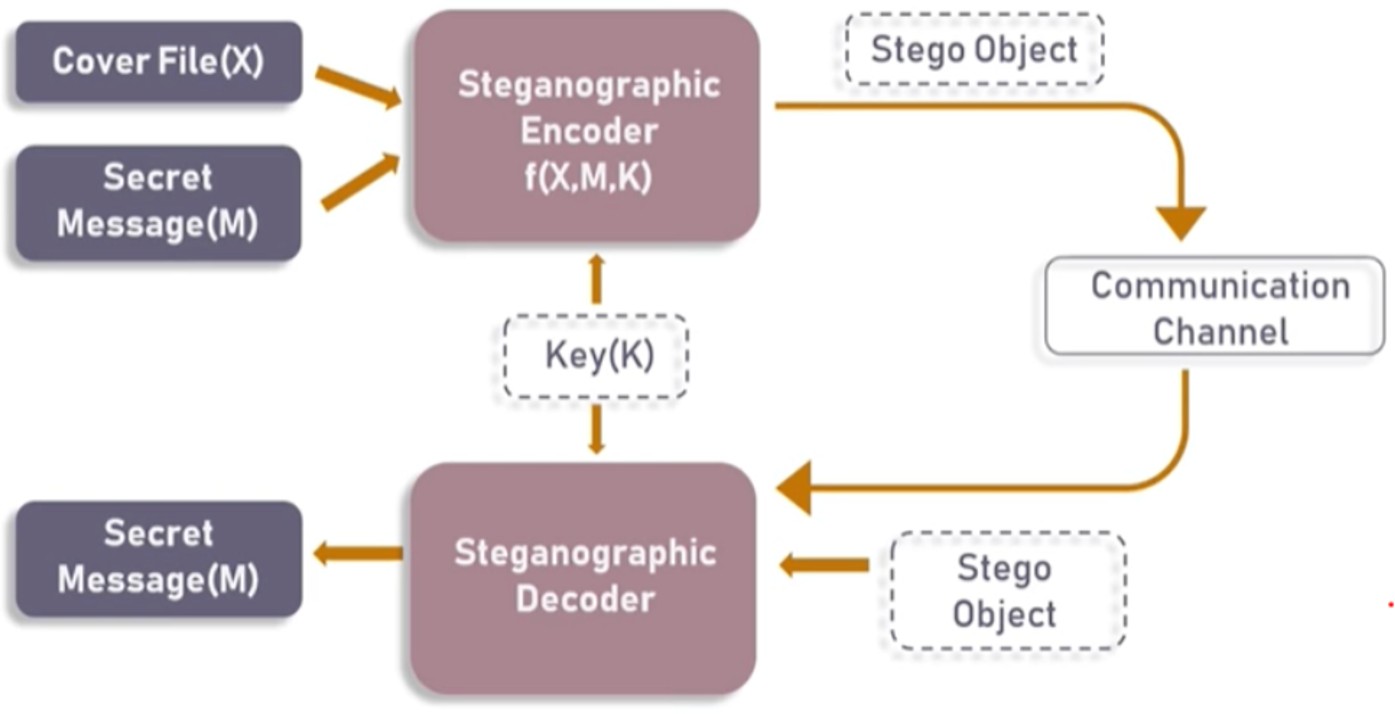
***Keywords***— **steganography, Adjacent Pixel Difference (APD), encryption, unique key, data hiding**

**Abstract:**

**Introduction:**

The art of hiding a message, image, or file within another message, image, or file is known as steganography. The message is encrypted using steganography. One of the most important requirements of data concealment is that the data be undetectable. Steganography offers numerous advantages and is extremely useful in digital image processing, making it ideal for a wide range of applications. In today's environment, the internet makes it quite easy to send massive volumes of data to distant regions of the globe. Long-distance communication, on the other hand, continues to be a concern. Steganography systems have been developed to address this issue of security and safety. Watermarking and cryptography aren't the same as steganography. Steganography's fundamental goal is to obscure the message's existence, making it impossible for an observer to figure out where it is. Cryptography approaches, on the other hand, tend to safeguard communications by transforming data into a format that an eavesdropper cannot comprehend. And in watermarking, the logo takes precedence over the data. Steganography (from the Greek words stego, "covered," and graphos, "to write") is a method of concealed communication that implies "covered writing." Jerome Cardan, an Italian mathematician, invented a secret writing system in 1550 that involved a paper mask with perforations. After that, remove the mask to fill in the vacant areas of the page, and the message will display as harmless text. We used the Adjacent Pixel Difference (APD) technique to study the challenges of determining the proper place in a given image for this research. The visual quality is compared and measured using the PSNR method on the Python platform. In Workflow, Algorithm, Code, Literature Survey, and Output, the project findings are discussed and compared to existing approaches.





**Algorithm:**

In this project, we will be focussing on image-based steganography, i.e hiding secret data in an image. But before diving a little deeper into it, let’s look at what an image comprises of.

1. **Pixels are the building blocks of an image.**
2. **Every pixel contains three values: (red, green, blue) also known as RGB values.**
3. **Every RGB value ranges from 0 to 255.**

Now, let’s look at how we can encode and decode data into our image.

## Part 1) Encoding :-

There are a lot of algorithms that can be used to encode data into the image, and in fact, you can also make one yourself. The one being used in this blog is easy to understand and implement, as well.

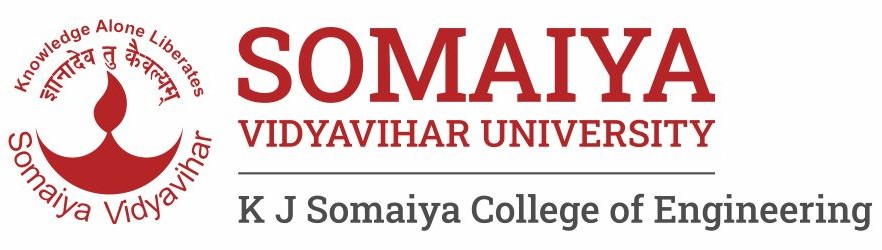
**Step 1:-** For each character in the data, its ASCII value is taken and converted into 8-bit binary [1]. The ASCII value of **H** is 72, whose binary equivalent is **01001000.**

**Step 2:-** 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**. Read the first three pixels. **(27, 64, 164), (248, 244, 194), (174, 246, 250)**

**Step 3:-** 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.**

***For example***, the first binary digit is 0 and the first RGB value is 27, it needs to be converted to even, which implies 26. Similarly, 64 gets converted to 63because the next binary digit is 1so the



RGB value should be made odd. So, the modified pixels are:

### (26, 63, 164), (248, 243, 194), (174, 246, 250)

**Step 4:-** The ninth value determines if more pixels should be read or not. If there is more data to be read, i.e. encoded or decoded, then the **ninth pixel changes to even**. Otherwise, if we want to stop reading pixels further, then make it odd.

Since we have to encode more data, the last value should be even. Similarly, i can be encoded in this image.

While making the pixel values **odd/even** by doing **+1 or -1,** you should take care of binary conditions. I.e., the pixel value should be more than or equal to 0 and less than or equal to 255. The new image will look like:

**[(26, 63, 164), (248, 243, 194), (174, 246, 250), (148, 95, 231),**

**(188, 155, 168), (70, 167, 126), (132, 173, 97), (112, 69, 206),**

**(254, 29, 213), (53, 153, 220), (246, 225, 229), (142, 82, 175)]**

## Part 2) Decoding :-

For decoding, we shall try to find how to reverse the previous algorithm that we used to encode data.

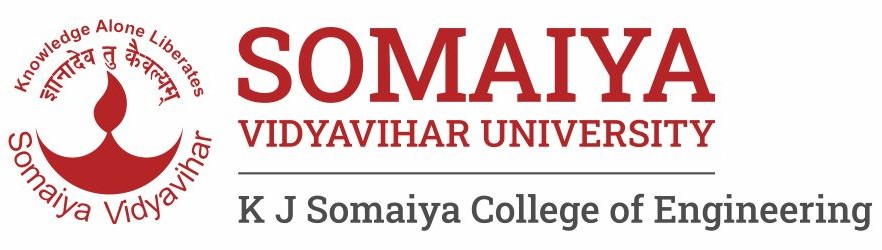
**Step 1:-** Again, three pixels are read at a time. The first 8 RGB values give us information about the secret data, and the ninth value tells us whether to move forward or not. Read the first three pixels. (27, 64, 164), (248, 244, 194), (174, 246, 250)

**Step 2:-** For the first eight values, if the value is odd, then the binary bit is 1, otherwise it is 0. Reading the first value: 26, which is even, therefore the binary bit is 0. Similarly, for 63, the binary bit is 1and for 164it is 0. This process continues until the eight RGB value.

**Step 3:-** The bits are concatenated to a string, and with every three pixels, we get a byte of secret data, which means one character. We finally get the binary value: 01001000 after concatenating all individual binary values. The final binary data corresponds to decimal value 72, and in ASCII, it represents the character H.

**Step 4:-** Now, if the ninth value is even then we keep reading pixels three at a time, or otherwise, we stop. Since the ninth value is even, we repeat the above steps. We stop when the ninth value encountered is odd.

As a result, we get our original message back which was Hii.



**Code:**

# Python program implementing Image Steganography

# PIL module is used to extract

# pixels of image and modify it

from PIL import Image

# Convert encoding data into 8-bit binary

# form using ASCII value of characters

def genData(data):

        # list of binary codes

        # of given data

        newd = []

        for i in data:

            newd.append(format(ord(i), '08b'))

        return newd

# Pixels are modified according to the

# 8-bit binary data and finally returned

def modPix(pix, data):

    datalist = genData(data)

    lendata = len(datalist)

    imdata = iter(pix)

    for i in range(lendata):

        # Extracting 3 pixels at a time

        pix = [value for value in imdata.\_\_next\_\_()[:3] +

                                imdata.\_\_next\_\_()[:3] +

                                imdata.\_\_next\_\_()[:3]]

        # Pixel value should be made

        # odd for 1 and even for 0

        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

        # Eighth pixel of every set tells

        # whether to stop ot read further.

        # 0 means keep reading; 1 means thec

        # message is over.

        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]

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

# Encode data into image

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()))

# Decode the data in the image

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]]

        # string of binary data

        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

# 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")

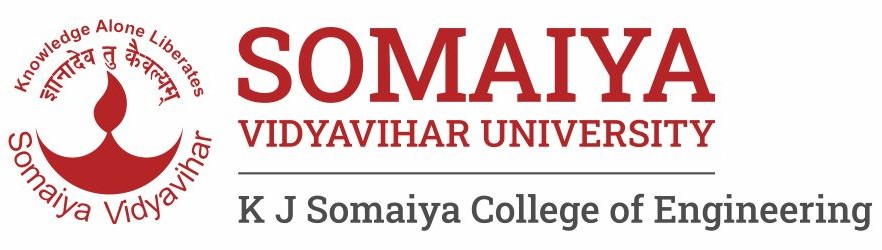
# Driver Code

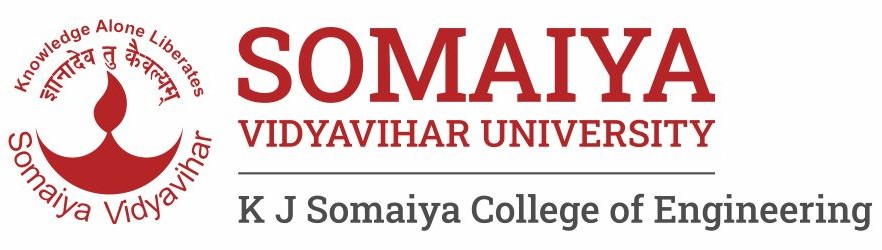
if \_\_name\_\_ == '\_\_main\_\_' :

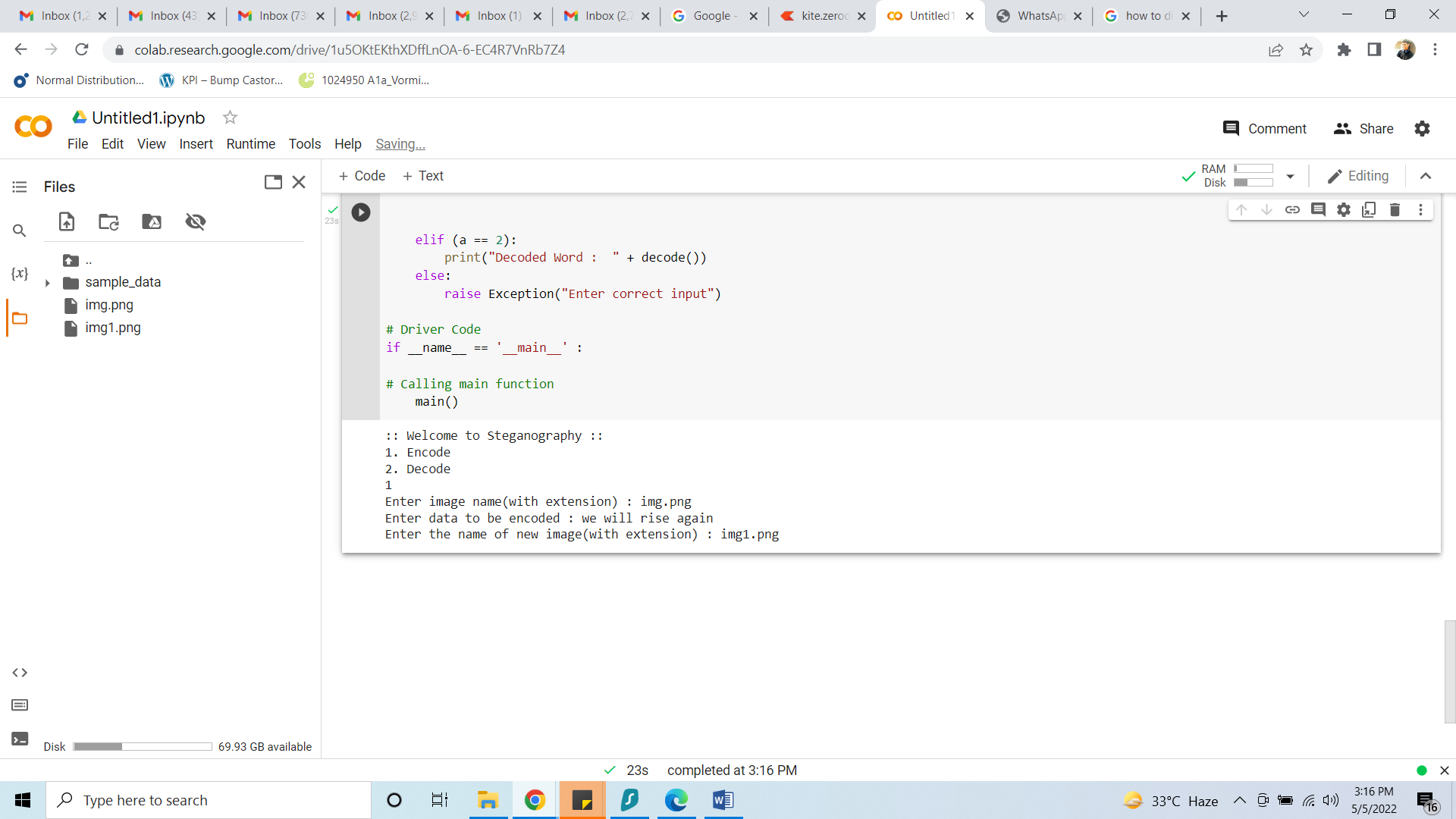
# Calling main function

    main()

**OUTPUT:**

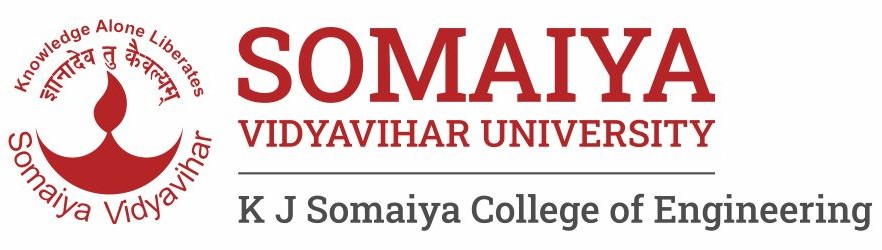


Encoding



Input Image

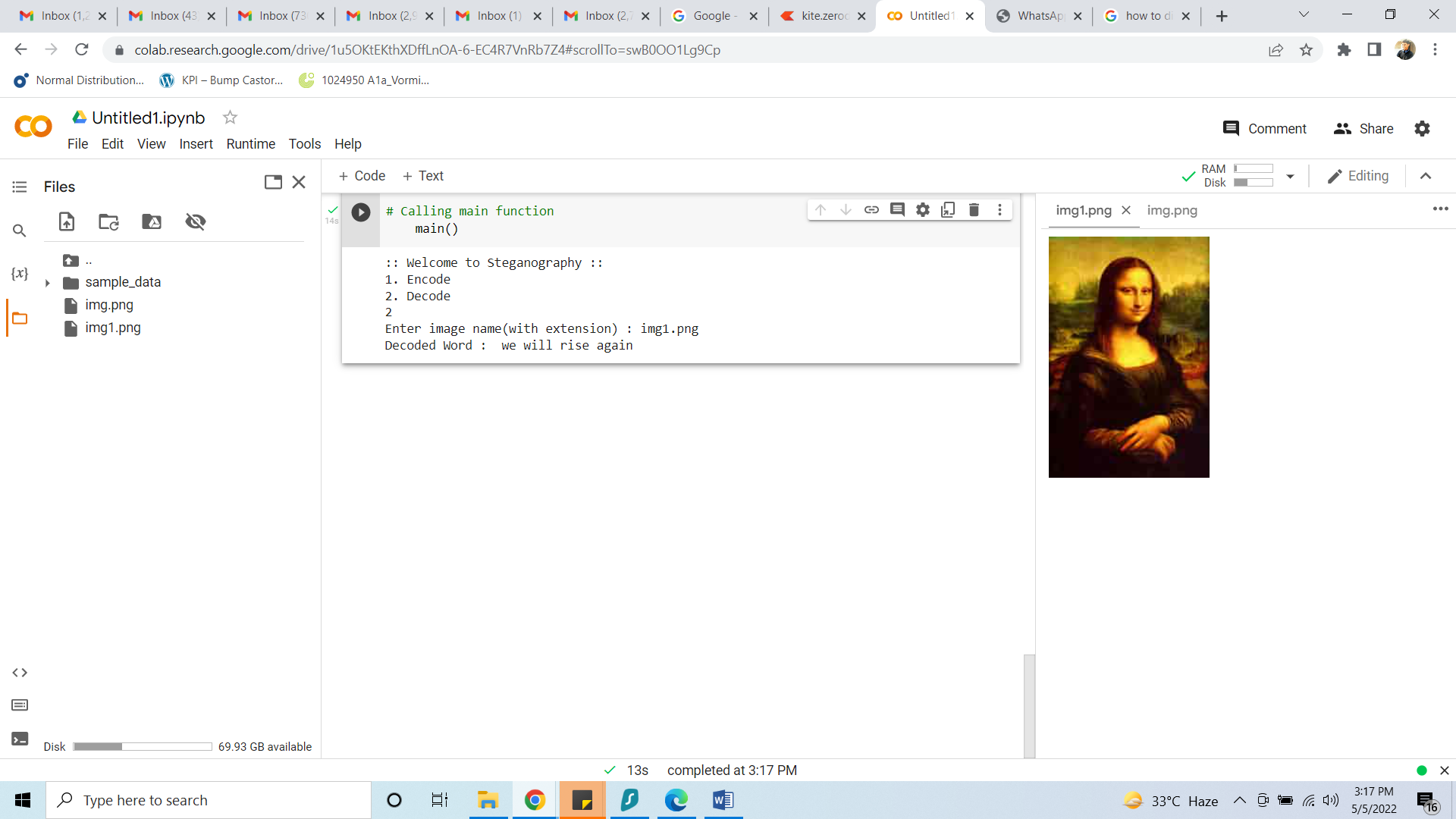




Updated Image (After Encoding)



DECODING



**Conclusion:**

* Here we have studied Steganography i.e. encoding hidden data into a multimedia carrier such as image, audio or video.
* We have encoded data into an image and then decoded it.The message is encrypted using steganography.
* This method is used to address this issue of security and safety.