Visual Cryptography for File Protection

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science

by

Kulvir Singh – 19BCE2074

Rishabh Johri – 19BDS0021

Garvit Arora – 19BCE0422

Under the guidance of

Prof. Joshva Devadas

SCOPE

VIT, Vellore.

November, 2020

Acknowledgement

We would like to use this opportunity to thank everybody who has helped us throughout this Cyber Security Project on Visual Cryptography for file protection. We would explicitly like to thank our Professor, Mr. Joshva Devdas T for his unlimited support and guidance all along the course of this project work without which the project work would not be a success. All that we have done would not be possible without the supervision and support and we would not forget to thank them.

Signature of Students

Kulvir Singh

Rishabh Johri

Garvit Arora

Contents

Abstract and Problem Statement

In today's world, cyber security is the basic need for every person or entity to protect its files and digital data from unwanted users accessing it. We have derived inspiration from various research papers on Visual Cryptography and related techniques to create an improved and efficient code.

The project uses the basic concept of visual cryptography. The simplest version of the visual secret sharing problem assumes that the message consists of a collection of black and white pixels and each pixel is handled separately.

Our main motive is that we will be using a better and much simpler algorithm, that is, Rubik's Cube Principle based image encryption which includes an RBG color scheme. *This will enable us to encrypt images in a colored pixels format instead of using outdated black and white scheme*.

The ***objective*** of project to produce the encrypted and decrypted version of the image that is taken as the input from the user. The encrypted image would be having a color scheme of red, blue and green. The code will decrypt the encrypted image and return the same image as the input by the user.

Through the project we will be addressing the problem of encrypting and decrypting an image file by using Rubik's Cube Principle based image encryption which includes a RBG color scheme(red, blue, green).

Through the project we will be implementing the Rubik's Cube Algorithm which is nothing but the various possible combinations of a Rubik's Cube and technique's related to solve the randomly jumbled cube. Encryption: In a similar way, we need to encrypt the image file by randomly arranging the bits and add a color scheme of red, blue and green to display the encrypted form of the image file. Decryption: For decrypting the image we need to solve the randomly ordered bits to the original order which will be possible by using the Rubik's Cube algorithm.

Related Work

1. ***A Secure Image Encryption Algorithm Based on Rubik's Cube Principle by Khaled Loukhaoukha, Jean-Yves Chouinard, and Abdellah Berdai***

https://www.researchgate.net/publication/257946264_A_Secure_Image_Encryption_Algorithm_Based_on_Rubik's_Cube_Principle

Study of the Rubik's Cube Principle for image encryption of black and white images has been mention. The study does not implement the algorithm which would work of colored images hence making it irrelevant at this time.

2. ***Secure Communication Based on Rubik's Cube Algorithm and Chaotic Baker Map by Abitha K.A and Pradeep K Bharathan***

https://pdf.sciencedirectassets.com/282073

This paper deals with the implementation of Chaotic Key a Chaotic Baker Map for the Rubik's Cube Algorithm This technique increased the complexity of the code used up more memory as it produced an extra image before finally getting the fully encrypted image. Also, this paper only caters to black and white pixels.

3. ***Multistage Image Encryption Using Rubik's Cube for Secured Image Transmission byT. Gomathi, B.L. Shiva kumar***

http://www.ijarcs.info/index.php/Ijarcs/article/view/2544

This paper gives us the algorithm more multistage image encryption using Rubik's Cube. The image was being divided into different stages/parts which was then being encrypted. This technique can be made more secure by jumbling the partial image and then encrypting.

4. ***Rubik's cube principle based image encryption implementation on mobile devices by Valeriu Manuel Ionescu and Adrian-Viorel Diaconu***

https://ieeexplore.ieee.org/document/7301247

The application of Rubik's Cube Algorithm in various applications is observed and especially in mobile services. The algorithm needs to updated from time to time to match the high performance of the today's mobile services.

5. Visual Cryptography by Moni Naor and Adi Shamir

https://link.springer.com/chapter/10.1007/BFb0053419

A new type of crypto system which can decode images without any specific cryptographic technique is discussed. The algorithm is old and newer and simple techniques can be implemented as of today.

Other Links and References

https://docs.python.org/3.9/library/index.html

https://docs.python.org/3.9/using/index.html

https://pypi.org/project/Pillow/

https://pillow.readthedocs.io/en/stable/reference/Image.html

https://www.w3schools.com/python/numpy_intro.asp

https://www.programiz.com/python-programming/file-operation

https://github.com/python-pillow/Pillow/issues/2601

https://www.tutorialspoint.com/working-with-images-in-python

https://www.geeksforgeeks.org/python-bitwise-operators/#:~:text=In%20Python%2C%20bitwise%20operators%20are,perform%20bitwise%20calculations%20on%20integers.&text=Bitwise%20AND%20operator%3A%20Returns%201,bit%20is%201%20else%200.

Methodology of the proposed System

Modules :

- Key Generation
- Encryption
- Decryption
- Image Processing
- File Writing

We have also incorporated certain user-defined functions which can be considered as Sub-Modules :

- Up Circular Shift
- Down Circular Shift
- Bit Rotation

Key Generation

The image to be encrypted is taken as input from the user. This image is then processed as pixels. These pixels are stored into a multi-dimensional array. The number of rows of this array is stored into a variable say m and the number of columns is stored into another variable n.

A variable alpha is set to 8 and then two vectors Kr and Kc are declared to represent the keys of rows and columns respectively. Kr vector is filled with m random integers ranging from 0 to ((2^8)-1) and Kc is filled with n random integers ranging from 0 to ((2^8)-1) using the randint(a,b) function in python.

Image Processing

The user is required to store their image to be encrypted in the project folder. Using Image class from PIL library in python, we extract and process the image stored by the user. Image.open() function is used to fetch the input image. It is then converted to pixels by using the image.load() function. Red, blue and green matrices are then initialized to get the respective values from the pixel matrix by looping through the pixel matrix.

After encryption the altered red, blue and green matrices are compiled and set into the pixel matrix. The pixel matrix is then passed to the image.save() function to convert it back to image format.

File Writing

Using open() function from python's default library we create a keys.txt file. After key generation we copy the data stored in vector Kr and Kc into the file by using the write() function.

Up Circular Shift

This is a user defined function that performs up circular shift on an array that is send through the function. It uses the numpy library and numpy.roll() function of python. It shifts the data at a particular index of the array to a new position according to number of shifts required to be performed. The number of shifts is also taken in as a parameter. It shifts the data vertically upwards in the column.

Down Circular Shift

This is a user defined function that performs down circular shift on an array that is send through the function. It uses the numpy library and numpy.roll() function of python. It shifts the data at a particular index of the array to a new position according to number of shifts required to be performed. The number of shifts is also taken in as a parameter. It shifts the data vertically downwards in the column.

Bit Rotation

This function is used to reverse the bits that is entered as parameter to this function. It uses basic string reversal technique which is starting from the end of the string and reaching to first bit there by getting the bits of the string in reverse order.

Encryption

Following are the steps for encryption

First, we need to generate two vectors Kr and Kc of length M and N and they are initialized with random values. Now determine the total number of iterations and then implement the algorithm and the code as given below.

(1) Generate randomly two vectors K_R and K_C of length M and N, respectively. Element $K_R(i)$ and $K_C(j)$ Each take a random value of the set $\mathcal{A} = \{0, 1, 2, \ldots, 2^\alpha - 1\}$. Note that both K_R and K_C must not have constant values.

(2) Determine the number of iterations, $\text{ITER}_{\max}$, and initialize the counter ITER at 0.

(3) Increment the counter by one: ITER = ITER + 1.

(4) For each row i of image I_o,

(a) compute the sum of all elements in the row i, this sum is denoted by $\alpha(i)$

$$\alpha(i) = \sum_{j=1}^{N} \mathrm{I}_o(i, j), \quad i = 1, 2, \ldots, M, \tag{1}$$

(b) compute modulo 2 of $\alpha(i)$, denoted by $M_{\alpha(i)}$,

(c) row i is left, or right, circular-shifted by $K_R(i)$ positions (image pixels are moved $K_R(i)$ positions to the left or right direction, and the first pixel moves in last pixel.), according to the following:

$$\begin{aligned} &\text{if } M_{\alpha(i)} = 0 \longrightarrow \text{right circular shift} \\ &\quad \text{else} \longrightarrow \text{left circular shift.} \end{aligned} \tag{2}$$

(5) For each column j of image I_o,

(a) compute the sum of all elements in the column j, this sum is denoted by $\beta(j)$,

$$\beta(j) = \sum_{i=1}^{M} \mathrm{I}_o(i, j), \quad j = 1, 2, \ldots, N, \tag{3}$$

(b) compute modulo 2 of $\beta(j)$, denoted by $M_{\beta(j)}$.

(c) column j is down, or up, circular-shifted by $K_C(i)$ positions, according to the following:

$$\begin{aligned} &\text{if } M_{\beta(j)} = 0 \longrightarrow \text{up circular shift} \\ &\quad \text{else} \longrightarrow \text{down circular shift.} \end{aligned} \tag{4}$$

Steps 4 and 5 above will create a scrambled image, denoted by I_{SCR}.

(6) Using vector K_C, the bitwise XOR operator is applied to each row of scrambled image I_{SCR} using the following expressions:

$$\begin{aligned} \mathrm{I}_1(2i-1, j) &= \mathrm{I}_{\text{SCR}}(2i-1, j) \oplus K_C(j), \\ \mathrm{I}_1(2i, j) &= \mathrm{I}_{\text{SCR}}(2i, j) \oplus \text{rot}\,180(K_C(j)), \end{aligned} \tag{5}$$

where $\oplus$ and rot $180(K_C)$ represent the bitwise XOR operator and the flipping of vector K_C from left to right, respectively.

(7) Using vector K_R, the bitwise XOR operator is applied to each column of image I_1 using the following formulas:

$$\begin{aligned} \mathrm{I}_{\text{ENC}}(i, 2j-1) &= \mathrm{I}_1(i, 2j-1) \oplus K_R(j), \\ \mathrm{I}_{\text{ENC}}(i, 2j) &= \mathrm{I}_1(i, 2j) \oplus \text{rot}\,180(K_R(j)). \end{aligned} \tag{6}$$

with rot $180(K_R)$ indicating the left to right flip of vector K_R.

(8) If ITER = $\text{ITER}_{\max}$, then encrypted image I_{ENC} is created and encryption process is done; otherwise, the algorithm branches to step 3.

Vectors K_R, K_C and the max iteration number $\text{ITER}_{\max}$ are considered as secret keys in the proposed encryption algorithm. However, to obtain a fast encryption algorithm it is preferable to set $\text{ITER}_{\max} = 1$ (single iteration). Conversely, if $\text{ITER}_{\text{MAX}} > 1$, then the algorithm is more secure because the key space is larger than for $\text{ITER}_{\text{MAX}} = 1$. Nevertheless, in the simulations presented in Section 3, the number of iterations $\text{ITER}_{\max}$ was set to one.

```python
for i in range(m):
 rTotalSum = sum(r[i]) #sum of each array present in r[][]
 gTotalSum = sum(g[i])
 bTotalSum = sum(b[i])
 #modulo of sum of each r,g,b
 rModulus = rTotalSum % 2
 gModulus = gTotalSum % 2
 bModulus = bTotalSum % 2

 if(rModulus==0):
  #right circular shift according to Kr
  r[i] = numpy.roll(r[i],Kr[i])
 else:
  #left circular shit according to Kr
  r[i] = numpy.roll(r[i],-Kr[i])
 if(gModulus==0):
  g[i] = numpy.roll(g[i],Kr[i])
 else:
  g[i] = numpy.roll(g[i],-Kr[i])
 if(bModulus==0):
  b[i] = numpy.roll(b[i],Kr[i])
 else:
  b[i] = numpy.roll(b[i],-Kr[i])
# For each column
for i in range(n):
```

```python
for i in range(n):
 rTotalSum = 0
 gTotalSum = 0
 bTotalSum = 0
 for j in range(m):
  rTotalSum += r[j][i]
  gTotalSum += g[j][i]
  bTotalSum += b[j][i]
 rModulus = rTotalSum % 2
 gModulus = gTotalSum % 2
 bModulus = bTotalSum % 2
 if (rModulus == 0):
  upshift(r, i, Kc[i])
 else:
  downshift(r, i, Kc[i])
 if (gModulus == 0):
  upshift(g, i, Kc[i])
 else:
  downshift(g, i, Kc[i])
 if (bModulus == 0):
  upshift(b, i, Kc[i])
 else:
  downshift(b, i, Kc[i])
```

```
# For each row
for i in range(m):
 for j in range(n):
  if (i % 2 == 1):
   r[i][j] = r[i][j] ^ Kc[j]
   g[i] [j] = g[i][j] ^ Kc[j]
   b[i][j] = b[i][j] ^ Kc[j]
  else:
   r[i][j] = r[i][j] ^ rotate180(Kc[j])
   g[i][j] = g[i][j] ^ rotate180(Kc[j])
   b[i][j] = b[i][j] ^ rotate180(Kc[j])

# For each column
for j in range(n):
 for i in range(m):
  if (j % 2 == 0):
   r[i][j] = r[i][j] ^ Kr[i]
   g[i][j] = g[i][j] ^ Kr[i]
   b[i][j] = b[i][j] ^ Kr[i]
  else:
   r[i][j] = r[i][j] ^ rotate180(Kr[i])
   g[i][j] = g[i][j] ^ rotate180(Kr[i])
   b[i][j] = b[i][j] ^ rotate180(Kr[i])
```

Decryption

Load the image and convert it into matrix containing the RGB values of the pixels using im.load(). Extract the RBG Values into 3 separate lists using nested for loop : outer for traversing pixels and inner for accessing the R,G,B value. Declare m and n as the number of pixels in row and columns respectively. INPUT the values of Kr and Kc. Follow the algorithm as given below:

(3) The bitwise XOR operation is applied on vector K_R and each column of the encrypted image I_{ENC} as follows:

$$\mathrm{I}_1(i,2j-1) = \mathrm{I}_{\mathrm{ENC}}(i,2j-1) \oplus K_R(j),$$
$$\mathrm{I}_1(i,2j) = \mathrm{I}_{\mathrm{ENC}}(i,2j) \oplus \mathrm{rot}\,180(K_R(j)), \tag{7}$$

(4) Then, using the K_C vector, the bitwise XOR operator is applied to each row of image I_1:

$$\mathrm{I}_{\mathrm{SCR}}(2i-1,j) = \mathrm{I}_1(2i-1,j) \oplus K_C(j),$$
$$\mathrm{I}_{\mathrm{SCR}}(2i,j) = \mathrm{I}_1(2i,j) \oplus \mathrm{rot}\,180(K_C(j)). \tag{8}$$

(5) For each column j of the scrambled image I_{SCR},

(a) compute the sum of all elements in that column j, denoted as $\beta_{SCR}(j)$:

$$\beta_{\mathrm{SCR}}(j) = \sum_{i=1}^{M} \mathrm{I}_{\mathrm{SCR}}(i,j), \quad j = 1,2,\ldots,N, \tag{9}$$

(b) compute modulo 2 of $\beta_{SCR}(j)$, denoted by $M_{\beta_{SCR}(j)}$,

(c) column j is down, or up, circular-shifted by $K_C(i)$ positions according to the following:

$$\text{if } M_{\beta_{\mathrm{SCR}}(j)} = 0 \longrightarrow \text{up circular shift}$$
$$\text{else} \longrightarrow \text{down circular shift}. \tag{10}$$

(6) For each row i of scrambled image I_{SCR},

(a) compute the sum of all elements in row i, this sum is denoted by $\alpha_{SCR}(i)$:

$$\alpha_{\mathrm{SCR}}(i) = \sum_{j=1}^{N} \mathrm{I}_{\mathrm{SCR}}(i,j), \quad i = 1,2,\ldots,M, \tag{11}$$

(b) compute modulo 2 of $\alpha_{SCR}(j)$, denoted by $M_{\alpha_{SCR}(j)}$,

(c) row i is then left, or right, circular-shifted by $K_R(i)$ according to the following:

$$\text{if } M_{\alpha_{\mathrm{SCR}}(j)} = 0 \longrightarrow \text{right circular shift}$$
$$\text{else} \longrightarrow \text{left circular shift}. \tag{12}$$

(7) If ITER = ITER_{max}, then image I_{ENC} is decrypted and the decryption process is done; otherwise, the algorithm branches back to step 2.

```python
    # For each column
    for j in range(n):
        for i in range(m):
            if (j % 2 == 0):
                r[i][j] = r[i][j] ^ Kr[i]
                g[i][j] = g[i][j] ^ Kr[i]
                b[i][j] = b[i][j] ^ Kr[i]
            else:
                r[i][j] = r[i][j] ^ rotate180(Kr[i])
                g[i][j] = g[i][j] ^ rotate180(Kr[i])
                b[i][j] = b[i][j] ^ rotate180(Kr[i])
    # For each row
    for i in range(m):
        for j in range(n):
            if (i % 2 == 1):
                r[i][j] = r[i][j] ^ Kc[j]
                g[i][j] = g[i][j] ^ Kc[j]
                b[i][j] = b[i][j] ^ Kc[j]
            else:
                r[i][j] = r[i][j] ^ rotate180(Kc[j])
                g[i][j] = g[i][j] ^ rotate180(Kc[j])
                b[i][j] = b[i][j] ^ rotate180(Kc[j])
    # For each column
    for i in range(n):
        rTotalSum = 0
```

DECRYPT.py

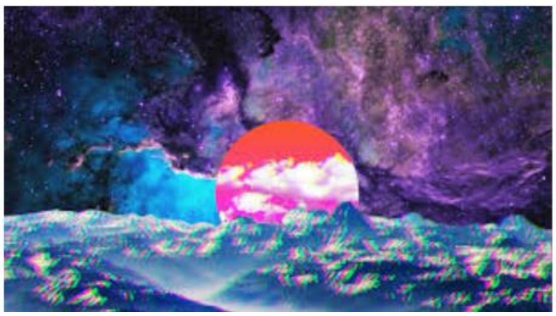
```python
        rTotalSum = 0
        gTotalSum = 0
        bTotalSum = 0
        for j in range(m):
            rTotalSum += r[j][i]
            gTotalSum += g[j][i]
            bTotalSum += b[j][i]
        rModulus = rTotalSum % 2
        gModulus = gTotalSum % 2
        bModulus = bTotalSum % 2
        if (rModulus == 0):
            downshift(r, i, Kc[i])
        else:
            upshift(r, i, Kc[i])
        if (gModulus == 0):
            downshift(g, i, Kc[i])
        else:
            upshift(g, i, Kc[i])
        if (bModulus == 0):
            downshift(b, i, Kc[i])
        else:
            upshift(b, i, Kc[i])
    # For each row
    for i in range(m):
        rTotalSum = sum(r[i])
```

```
# For each row
for i in range(m):
    rTotalSum = sum(r[i])
    gTotalSum = sum(g[i])
    bTotalSum = sum(b[i])
    rModulus = rTotalSum % 2
    gModulus = gTotalSum % 2
    bModulus = bTotalSum % 2
    if (rModulus == 0):
        r[i] = numpy.roll(r[i],-Kr[i])
    else:
        r[i] = numpy.roll(r[i], Kr[i])
    if (gModulus == 0):
        g[i] = numpy.roll(g[i],-Kr[i])
    else:
        g[i] = numpy.roll(g[i], Kr[i])
    if (bModulus == 0):
        b[i] = numpy.roll(b[i],-Kr[i])
    else:
        b[i] = numpy.roll(b[i], Kr[i])
```

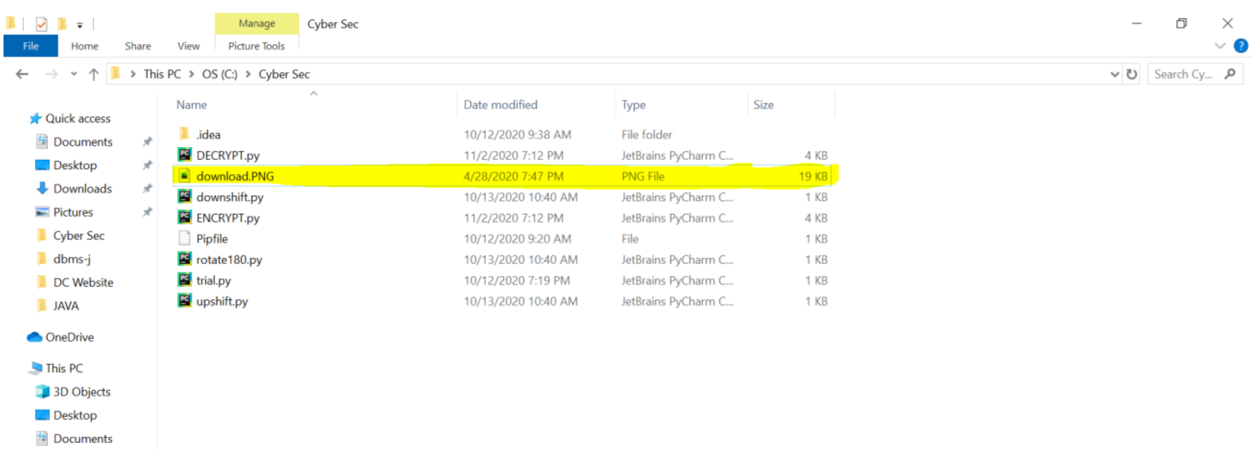
Results and Analysis

Working of the project is shown below using snapshots of each step.

The following image is encrypted. The image file is under the name of ‘download.PNG’.



The image to be encrypted is to be saved in the project directory. The image file is named ‘download.PNG’ and is highlighted below.

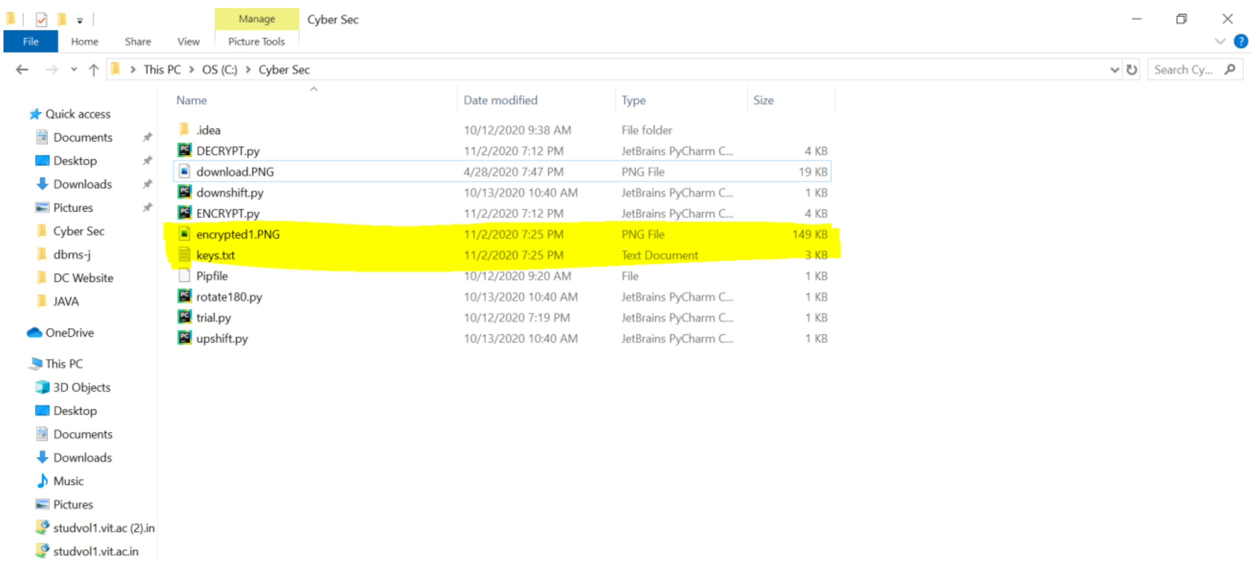


On running the encrypt.py file in the terminal, the keys are displayed and the encrypted image along with the keys.txt file is created

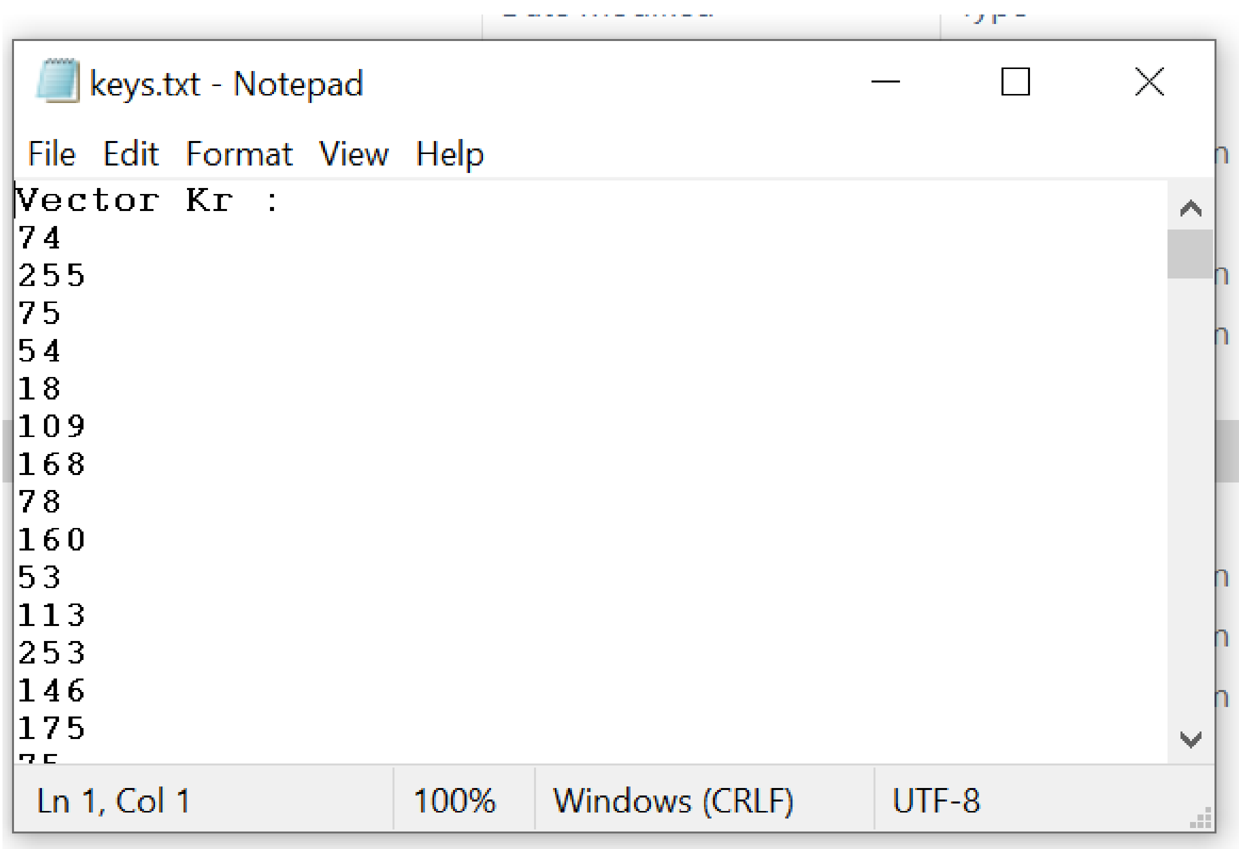
```
Terminal: Local (2)
Microsoft Windows [Version 10.0.18362.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

(Cyber Sec) C:\Cyber Sec>python encrypt.py
Vector Kr :  [74, 255, 75, 54, 18, 109, 168, 78, 160, 53, 113, 253, 146, 175, 75, 22, 190, 37, 227, 143, 184, 102, 94, 39, 117, 250, 57, 174, 229, 140, 175, 18, 229, 156, 214, 2
11, 57, 139, 244, 96, 171, 11, 60, 151, 251, 158, 70, 140, 209, 29, 116, 94, 55, 218, 69, 251, 219, 142, 77, 241, 46, 231, 109, 168, 54, 19, 7, 221, 83, 246, 83, 56, 50, 188, 14
6, 223, 85, 104, 251, 23, 192, 121, 208, 80, 87, 135, 179, 61, 139, 41, 174, 182, 41, 144, 37, 29, 3, 238, 217, 81, 225, 241, 43, 218, 23, 177, 161, 83, 253, 14, 254, 63, 146, 9
3, 206, 28, 28, 27, 172, 56, 237, 213, 79, 140, 83, 103, 35, 164, 87, 106, 160, 90, 79, 94, 127, 4, 78, 147, 249, 99, 143, 20, 109, 15, 149, 114, 32, 58, 22, 189, 90, 246, 213,
16, 172, 254, 146, 92, 120, 69, 180, 196, 243, 72, 76, 243, 86, 226, 240, 32, 202, 85, 34, 188, 99, 29, 83, 137, 66, 81, 150, 49, 159, 13, 251, 103, 29, 28, 169, 98, 161, 225, 1
40, 179, 1, 7, 56, 199, 118, 137, 67, 6, 127, 154, 235, 32, 11, 190, 222, 137, 156, 84, 70, 137, 109, 22, 144, 5, 174, 76, 221, 198, 8, 116, 220, 14, 34, 114, 206, 74, 142, 7, 2
10, 144, 188, 191, 48, 50, 26, 74, 40, 114, 253, 157, 139, 198, 236, 182, 108, 154, 76, 160, 141, 250, 194, 143, 234, 176, 108, 106, 15, 243, 69, 226, 7, 122, 214, 18, 172, 184,
 82, 128, 27, 169, 92, 27, 162, 130, 150, 136, 158, 13, 137, 206, 232, 250, 28, 83, 143, 112, 66, 81, 135, 92, 165, 186, 245, 217, 156, 233]
Vector Kc :  [119, 78, 117, 25, 116, 95, 222, 94, 170, 53, 248, 219, 149, 100, 44, 179, 214, 4, 103, 249, 199, 177, 146, 209, 181, 68, 7, 62, 19, 144, 64, 231, 12, 29, 59, 158,
112, 231, 60, 229, 66, 234, 247, 51, 169, 69, 149, 164, 57, 34, 193, 183, 196, 209, 90, 46, 50, 181, 194, 41, 30, 242, 250, 91, 204, 231, 176, 162, 177, 238, 224, 251, 119, 103,
 214, 51, 70, 70, 54, 55, 64, 17, 222, 84, 13, 80, 87, 3, 37, 87, 176, 95, 142, 3, 169, 218, 226, 35, 6, 116, 231, 36, 81, 193, 220, 76, 129, 77, 1, 162, 109, 207, 51, 90, 104,
10, 42, 135, 191, 191, 138, 200, 251, 140, 168, 36, 166, 122, 251, 168, 70, 180, 234, 162, 216, 76, 28, 250, 122, 212, 214, 239, 185, 211, 146, 197, 230, 117, 121, 184, 231, 49,
 135, 183, 24, 193, 210, 24, 27, 180, 112, 136, 118, 94, 74, 73, 225, 91]
Success
```

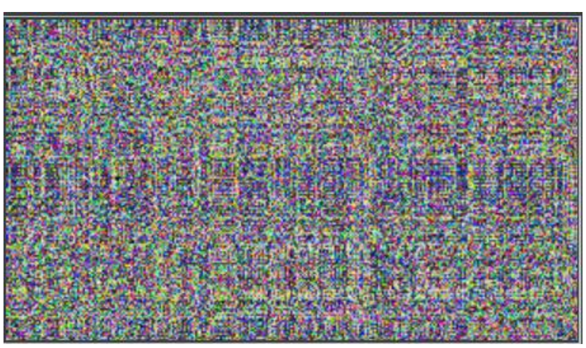
The encrypted image is generated in the encrypted1.png file and the keys are stored in the keys.txt file as highlighted below.



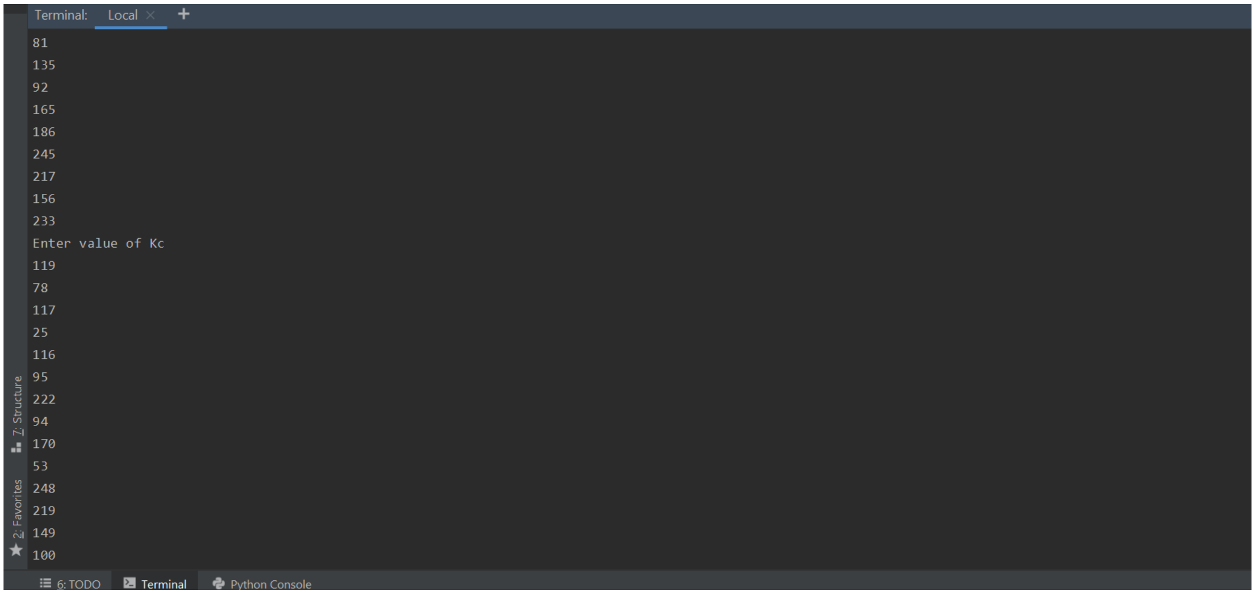
The keys.txt file containing the keys required for decryption.

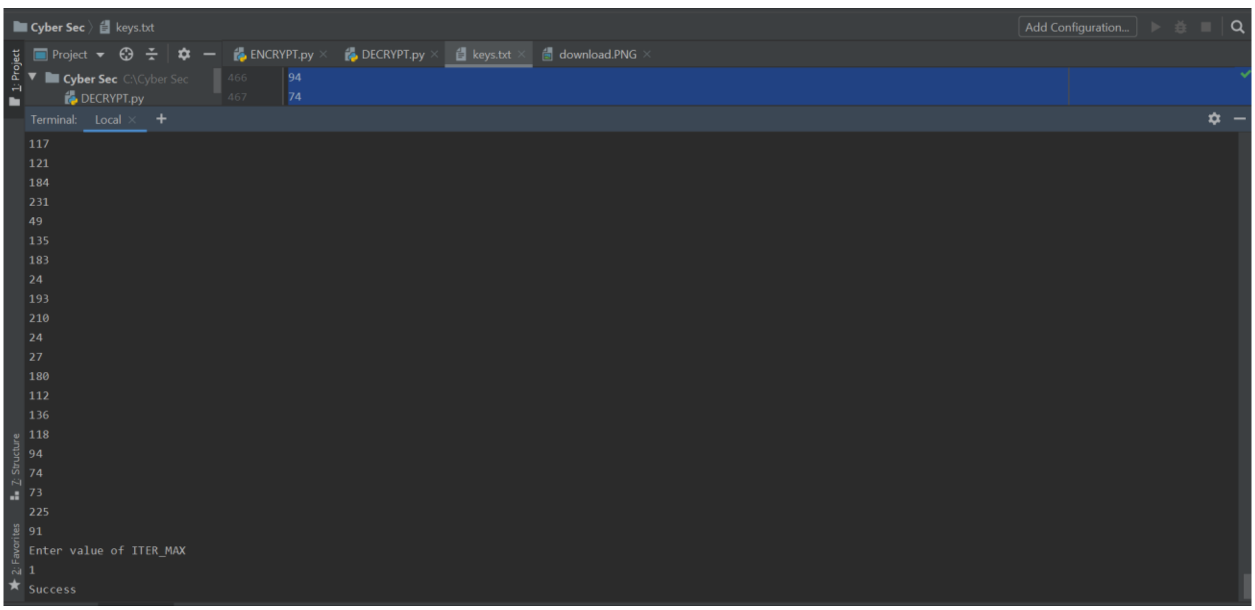


The ‘download.PNG’ is encrypted successfully and the encrypted image(‘encrypted1.PNG’) is shown below.

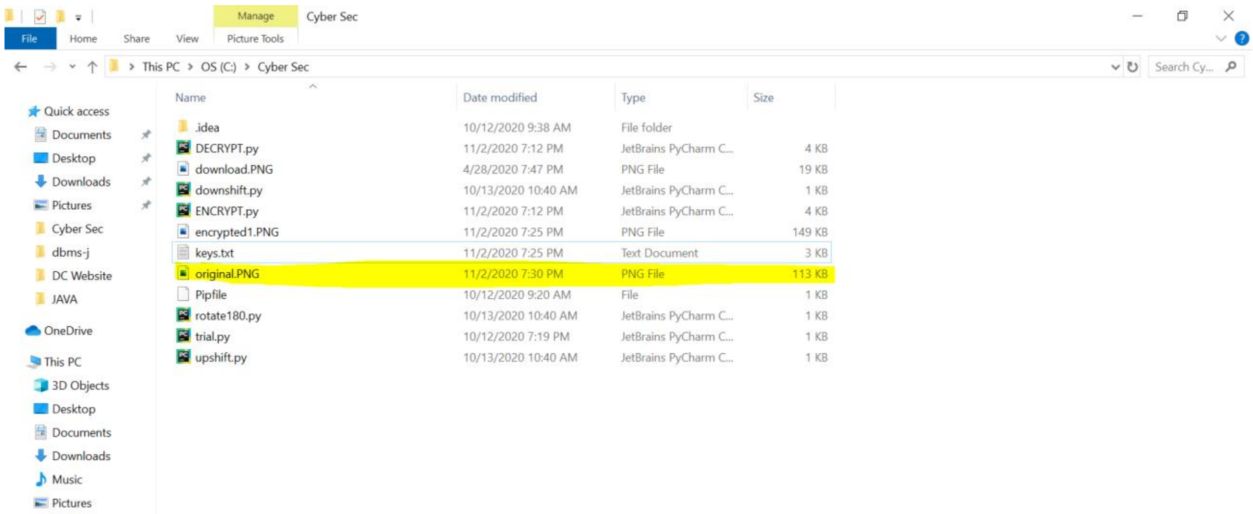


On running the decrypt.py file in terminal to decrypt the encrypted image, we first need to copy the keys from the keys.txt file and enter it into the terminal as shown below.

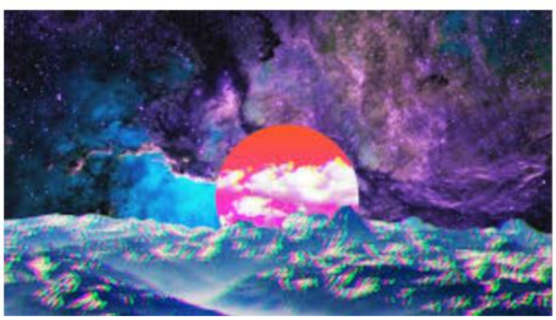




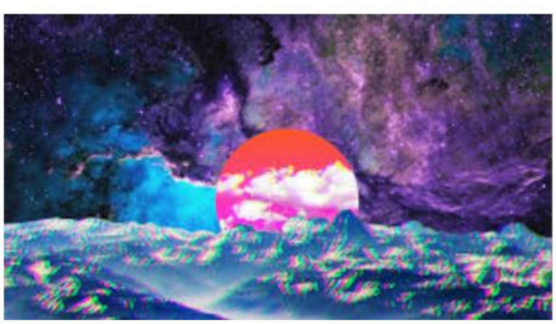
After entering all the keys, the code gets executed which generates the decrypted image which is stored under the same folder with the name ‘original.PNG’



The original.PNG is the same as the download.PNG image hence decryption is performed successfully.

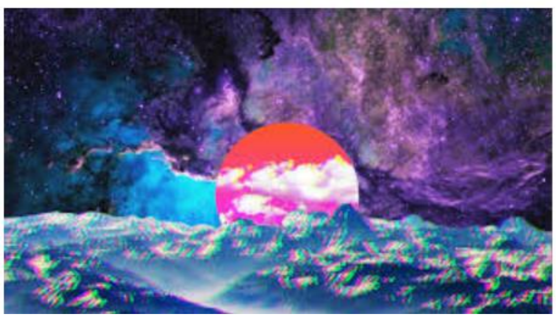


To sum up:



Download.PNG ==== User input

Encrypted1.PNG======encrypted form of user input



Original.PNG====== decrypted image

Conclusion and Future Work

Our project successfully encrypts an image, generates the keys which are further used for decryption. On decryption the user can see the initial image as it is. The image on encryption or decryption does not get corrupted and hence there is no loss of data.

This project can be extended to protect files of other formats such word, excel, ppt etc. by converting the file to image format and then using the project to encrypt and decrypt the same. In future we can write code to convert a non-image file into an image and protect it using our project.

Plagiarism Report

Originality report

COURSE NAME
Cybersecurityprj

STUDENT NAME
KULVIR SINGH 19BCE2074

FILE NAME
KULVIR SINGH 19BCE2074 - Project

REPORT CREATED
Nov 6, 2020

Summary

Flagged passages	2	2%
Cited/quoted passages	2	2%
Web matches		
springer.com	2	2%
semanticscholar.org	1	1%
researchgate.net	1	0.5%

1 of 4 passages
Student passage QUOTED

Submitted in partial fulfillment of the requirements for the degree of

Top web match

Dissertation **Submitted in Partial Fulfillment of the Requirements for the Degree of** Doctor of Philosophy (Ph.D) In Forest ecology Faculty of Natural Resources Tarbiat Modares University Effect of...

(PDF) Dissertation Submitted in Partial Fulfillment of the ... https://www.researchgate.net/publication/342735836_Dissertation_Submitted_in_Partial_Fulfillment_of_the_Requirements_for_the_Degree_of_Doctor_of_Philosophy_

2 of 4 passages
Student passage FLAGGED

The simplest version of the visual secret sharing problem assumes that the message consists of a collection of black and white pixels and each pixel is handled separately.

Top web match

2 The Model **The simplest version of the visual secret sharing problem assumes that the message consists of a collection of black and white pixels and each pixel is handled separately** 2. Each original...

Visual cryptography https://link.springer.com/content/pdf/10.1007/BFb0053419.pdf

3 of 4 passages
Student passage QUOTED

4. Rubik's **cube principle based image encryption implementation on mobile devices** by Valeriu **Manuel Ionescu and Adrian-Viorel Diaconu**

Top web match

article(Ionescu2015RubiksCP, title=(Rubik's **cube principle based image encryption** algorithm **implementation on mobile devices**), author=(Valeriu **Manuel Ionescu and Adrian-Viorel Diaconu**), journal=(2015...

Rubik's cube principle based image encryption algorithm ... https://www.semanticscholar.org/paper/Rubik's-cube-principle-based-image-encryption-on-Ionescu-Diaconu/b640996d70d4b6b85e2f1eb010e7f3ca8711bd0c

4 of 4 passages
Student passage FLAGGED

A new type of crypto system **which can decode images without any** specific **cryptographic** technique is discussed. The algorithm is old and newer...

Top web match

Abstract. In this paper we consider **a new type of** cryptographic scheme, **which can decode** concealed **images without any cryptographic** computations . The scheme is perfectly secure and very easy to...

Visual cryptography https://link.springer.com/content/pdf/10.1007/BFb0053419.pdf

---

Plagiarism Detector https://plagiarismdetector.net/

Plagiarism Detector

Date: November, 02 2020

PLAGIARISM SCAN REPORT

5%	95%	1000	5942
Plagiarised	Unique	Words	Characters

Exclude Url : None

Content Checked For Plagiarism

Abstract/Problem Statement In today's world, cyber security is the basic need for every person or entity to protect its files and digital data from unwanted users accessing it. We have derived inspiration from various research papers on Visual Cryptography and related techniques to create an improved and efficient code. The project uses the basic concept of visual cryptography. The simplest version of the visual secret sharing problem assumes that the message consists of a collection of black and white pixels and each pixel is handled separately. Our main motive is that we will be using a better and much simpler algorithm, that is, Rubik's Cube Principle based Image encryption which includes an RBG color scheme. This will enable us to encrypt images in a colored pixels format instead of using outdated black and white scheme. The objective of project to produce the encrypted and decrypted version of the image that is taken as the input from the user. The encrypted image would be having a color scheme of red, blue and green. The code will decrypt the encrypted image and return the same image as the input by the user. Methodology of the proposed System Modules : • KeyGeneration • Encryption • Decryption • ImageProcessing • File Writing We have also incorporated certain user-defined functions which can be considered as Sub-Modules : • Up Circular Shift • Down Circular Shift • Bit Rotation Key Generation The image to be encrypted is taken as input from the user. This image is then processed as pixels. These pixels are stored into a multi-dimensional array. The number of rows of this array is stored into a variable say m and the number of columns is stored into another variable n. A variable alpha is set to 8 and then two vectors Kr and Kc are declared to represent the keys of rows and columns respectively. Kr vector is filled with m random integers ranging from 0 to ((2^8)-1) and Kc is filled with n random integers ranging from 0 to ((2^8)-1) using the randint(a,b) function in python. Image Processing The user is required to store their image to be encrypted in the project folder. Using Image class from PIL library in python, we extract and process the image stored by the user. Image.open() function is used to fetch the input image. It is then converted to pixels by using the image.load() function. Red, blue and green matrices are then initialized to get the respective values from the pixel matrix by looping through the pixel matrix. After encryption the altered red, blue and green matrices are compiled and set into the pixel matrix. The pixel matrix is then passed to the image.save() function to convert it back to image format. File Writing Using open() function from python's default library we create a keys.txt file. After key generation we copy the data stored in vector Kr and Kc into the file by using the write() function. Up Circular Shift This is a user defined function that performs up circular shift on an array that is send through the function. It uses the numpy library and numpy.roll() function of python. It shifts the data at a particular index of the array to a new position according to number of shifts required to be performed. The number of shifts is also taken in as a parameter. It shifts the data vertically upwards in the column. Down Circular Shift This is a user defined function that performs down circular shift on an array that is send through the function. It uses the numpy library and numpy.roll() function of python. It shifts the data at a particular index of the array to a new position according to number of shifts required to be performed. The number of shifts is also taken in as a parameter. It shifts the data vertically downwards in the column. Bit Rotation This function is used to reverse the bits that is entered as parameter to this function. It uses basic string reversal technique which is starting from the end of the string and reaching to first bit there by getting the bits of the string in reverse order. Results and Analysis Working of the project is shown below using snapshots of each step. The following image is encrypted. The image file is under the name of 'download.PNG'. The image to be encrypted is to be saved in the project directory. The image file is named 'download.PNG' and is highlighted below. On running the encrypt.py file in the terminal, the keys are displayed and the encrypted image along with the keys.txt file is created The encrypted image is generated in the encrypted1.png file and the keys are stored in the keys.txt file as highlighted below. The keys.txt file containing the keys required for decryption. The 'download.PNG' is encrypted successfully and the encrypted image('encrypted1.PNG') is shown below. On running the decrypt.py file in terminal to decrypt the encrypted image, we first need to copy the keys from the keys.txt file and enter it into the terminal as shown below. After entering all the keys, the code gets executed which generates the decrypted image which is stored under the same folder with the name 'original.PNG' The original.PNG is the same as the download.PNG image hence decryption is performed successfully. To sum up: Download.PNG --- User input Encrypted1.PNG---encrypted form of user input Original.PNG---decrypted image Conclusion and Future Work Our project successfully encrypts an image, generates the keys which are further used for decryption. On decryption the user can see the initial image as it is. The image on encryption or decryption does not get corrupted and hence there is no loss of data. This project can be extended to protect files of other formats such word, excel, ppt etc. by converting the file to image format and then using the project to encrypt and decrypt the same. In future we can write code to convert a non-image file into an image and protect it using our project.

5% Plagiarised

The simplest version of the visual secret sharing problem assumes that the message consists of a collection of black-and-white pixels and each pixel is handled ...

https://books.google.com/books?id=lVa-BAAAQBAJ

Plagiarism Detector

Grammarly https://www.grammarly.com/plagiarism-checker grammarly Report: final rev csec

Final revcsec

by Kulvir Singh

General metrics

8,375	1,385	110	5 min 32 sec	10 min 39 sec
characters	words	sentences	reading time	speaking time

Score

77

This text scores better than 77% of all texts checked by Grammarly

Writing Issues

96	35	61
Issues left	Critical	Advanced

Plagiarism

3 %

4 sources

3% of your text matches 4 sources on the web or in archives of academic publications