19BIT0378
Paras Dang
ISM Project Review

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Under the Guidance of Dr. Sumaiya Thaseen I Associate Professor, SITE School of Information Technology and Engineering 2021-2022

TOPIC – IMAGE ENCRYPTION AND DECRYPTION

## LITERATURE REVIEWS

Base Paper
Using Gaussian Elimination

A complex project research involving the conversion of image into matrix form, using a mathematical concept to encrypt and decrypt it, to begin with, the plain-image is converted into blocks of single bytes and then the block is replaced as the value of key. Further, the control parameters of Gaussian Elimination method are selected by the user. The experimental result shows that the proposed algorithm can successfully encrypt/decrypt the images, and the algorithm has good encryption effect.

The issue in the literature has been addressed in the reference paper:
We've also talked about consensus algorithms that contain weaknesses, which has led the way for new consensus algorithms

This algorithm is irreversible without knowing the decryption key, which prevents brute force attack.

Next Reviews

RGB PIXEL Transposition and Shuffling

This paper proposed how to modify and restructure RGB values of step-by-step photo, which proved to be really effective in terms of security analysis. Additional exchanges for RGB values in an image file after the RGB component increase the image protection against all possible current attacks.

Key Cryptography CRDDBT Using – Relative Displacement (RDC) and Dynamic Base Transformation

This paper focused on a new encryption system outside of pre-defined key. The input unit unit is divided into several parts, each component encrypted using a different algorithm. Three different algorithms have been used to encrypt a character unit separated by a base. At higher levels of security, the key is taken both keys set separately. An important feature of this algorithm is that, part of the character unit is changed using modification of the base, the second part of the character unit is disabled by exchanging space with a growing number of repetitions, and in the remaining objects, they perform simple tasks. Therefore, this algorithm was a complex combination without adding any complex arithmetic.

## Blowfish Algorithm In Matlab

This paper proposed encryption and image resolution using a private key block cipher called 64-bits Blowfish designed to expand safety and performance improvement. This algorithm is is used as a flexible key size of up to 448 bits. It is renting Feistel network that duplicates 16 easy tasks. The blowfish algorithm is safer from unauthorized attacks and works faster than existing popular algorithms. The proposed algorithm is designed and visualized in use MATLAB. So when the number of cycles increases then the blowfish algorithm becomes powerful. Since Blowfish have no known weak points of defense to date it can be considered as the good algorithm for standard writing.

## Image Encryption / Decryption Using Advanced Encryption Standard

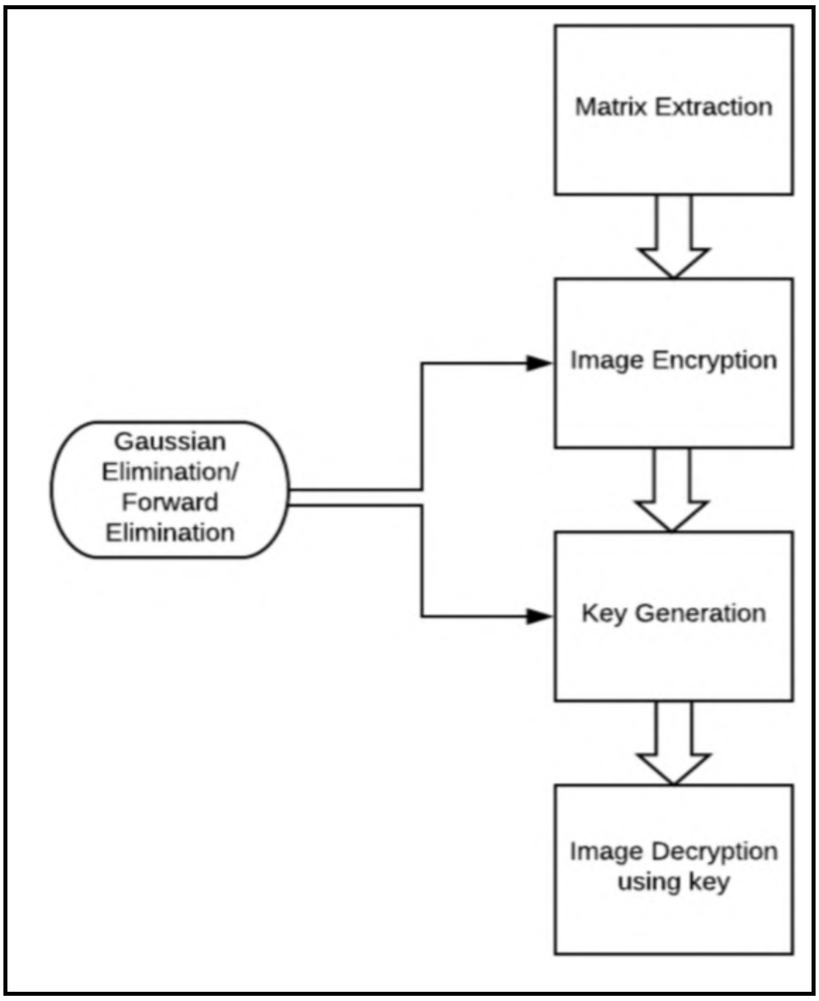
This paper has used text and image encryption and decryption using AES. The data features depend on its types. Therefore the same encryption method cannot be used
all kinds of data. If Images have a large data size and have real-time problems then they are the same method cannot be used to protect images and text from unauthorized access. A few changes to the AES approach can be used to protect image and text.

| Paper | Techniques | Advantages | Issues |
|---|---|---|---|
| Using Gaussian Elimination | Forward Elimination is used for encrypting images and LDU factorization is used for key | 1)Simplicity<br>2)Ratio<br>3)Robust<br>4)Best | Bad efficiency Decryption takes 4 time more time than encryption |

| | | | |
|---|---|---|---|
| | generation and for decrypting images, key is multiplied with encrypted image to get back original image. | | |
| RGB PIXEL Transposition and Shuffling | The algorithm ultimately makes it possible for encryption and decryption of the images based on the RGB pixel. | Really effective in terms of security analysis. | Complex System |
| Blowfish Algorithm In Matlab | This algorithm is is used as a flexible key size of up to 448 bits. | The blowfish algorithm is safer from unauthorized attacks and works faster than existing popular algorithms. | Since Blowfish have no known weak points of defense to date it can be considered as the good algorithm for standard writing. |
| Key Cryptography CRDDBT Using – Relative Displacement (RDC) and Dynamic Base Transformation | An important feature of this algorithm is that, part of the character unit is changed using modification of the base, the second part of the character unit is disabled by exchanging space with a growing number | This algorithm was a complex combination without adding any complex arithmetic. | The salient feature of this algorithm is that, a part of string is manipulated using base conversion, second part of string is deformed by interchanging position and increasing number of |

|  |  |  |  |
|---|---|---|---|
|  | of repetitions, and in the remaining objects, they perform simple tasks. |  | repetitions. time taken for encryption time taken for encryption. |

**SYSTEM ARCHITECTURE**


Matrix Extraction
Image Encryption
Gaussian Elimination/ Forward Elimination
Key Generation
Image Decryption using key

**Step 1:** Average of RGB values is taken to extract matrix for each image.

**Step 2:** Image Encryption is performed using Gaussian Elimination/Forward Elimination, where image matrix is converted to upper triangular matrix. Upper triangular matrix is converted back to RGB values, this gives encrypted image also known as ciphered image. Encrypted image is used for transmission.

**Step 3:** Key is generated alongside encryption of image which is further used for decryption. Key matrix stores all the operations performed on image matrix while converting it into upper triangular matrix.

**Step 4:** Ciphered image is converted back to original image using key.

## ALGORITHMS

### GAUSSIAN ELIMINATION

Introduction: In linear algebra, Gaussian elimination (also known as row reduction) is an algorithm for solving systems of linear equations. It is usually understood as a sequence of operations performed on the corresponding matrix of coefficients. This method can also be used to find the rank of a matrix, to calculate the determinant of a matrix, and to calculate the inverse of an invertible square matrix.

To perform row reduction on a matrix, one uses a sequence of elementary row operations to modify the matrix until the lower left-hand corner of the matrix is filled with zeros, as much as possible. There are three types of elementary row operations:

- Swapping two rows,
- Multiplying a row by a nonzero number,
- Adding a multiple of one row to another row.

The process of row reduction makes use of elementary row operations, and can be divided into two parts. The first part (sometimes called forward elimination) reduces a given system to row echelon form, from which one can tell whether there are no solutions, a unique solution, or infinitely many solutions. The second part (sometimes called back substitution) continues to use row operations until the solution is found; in other words, it puts the matrix into reduced row echelon form. Another point of view, which turns out to be very useful to analyze the algorithm, is that row reduction produces a matrix decomposition of the original matrix. The elementary row operations may be viewed as the multiplication on the left of the original matrix by elementary matrices. Alternatively, a sequence of elementary operations that reduces a single row may be viewed as multiplication by a Frobenius matrix. Then the first part of the algorithm computes an LU decomposition, while the second part writes the original matrix as the product of a uniquely determined invertible matrix and a uniquely determined reduced row echelon matrix.

Algorithm:

1. Partial pivoting: Find the kth pivot by swapping rows, to move the entry with the largestabsolute value to the pivot position. This imparts computational stability to the algorithm.
2. For each row below the pivot, calculate the factor f which makes the kth entry zero, and forevery element in the row subtract the fth multiple of corresponding element in kth row.
3. Repeat above steps for each unknown. We will be left with an upper triangular matrix

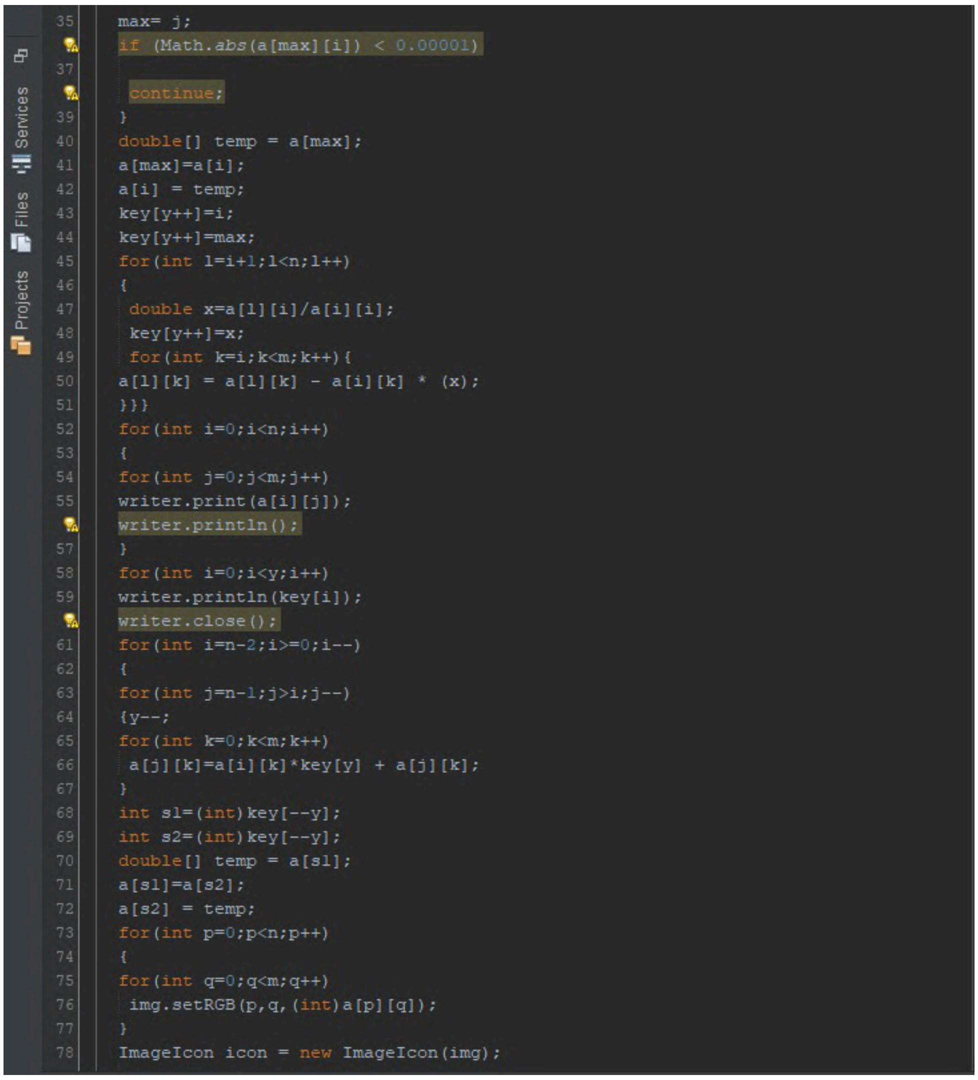
## IMPLEMENTATION

- Each image is actually a matrix consisting of RGB and alpha values.
- The first step of the project involves extracting the matrix from the image.
- The average of RGB values is then taken so as to get one image matrix.
- Gaussian Elimination is then applied on the image matrix to get an upper-triangular matrix which is the encrypted image matrix and the decryption key is generated. The algorithm is majorly about performing a sequence of operations on the rows of the matrix. What we wouldlike to keep in mind while performing these operations is that we want to convert the matrixinto an upper triangular matrix. The operations can be:
  - Swapping two rows
  - Multiplying a row by a non-zero scalar
  - Adding to one row a multiple of another
- This upper triangular matrix is our encrypted image matrix.
- The encrypted image matrix is converted back into the image, image is executed, andencryption process is completed.
- Key is generated alongside encryption of image. Information regarding image encryption isstored in the key matrix.

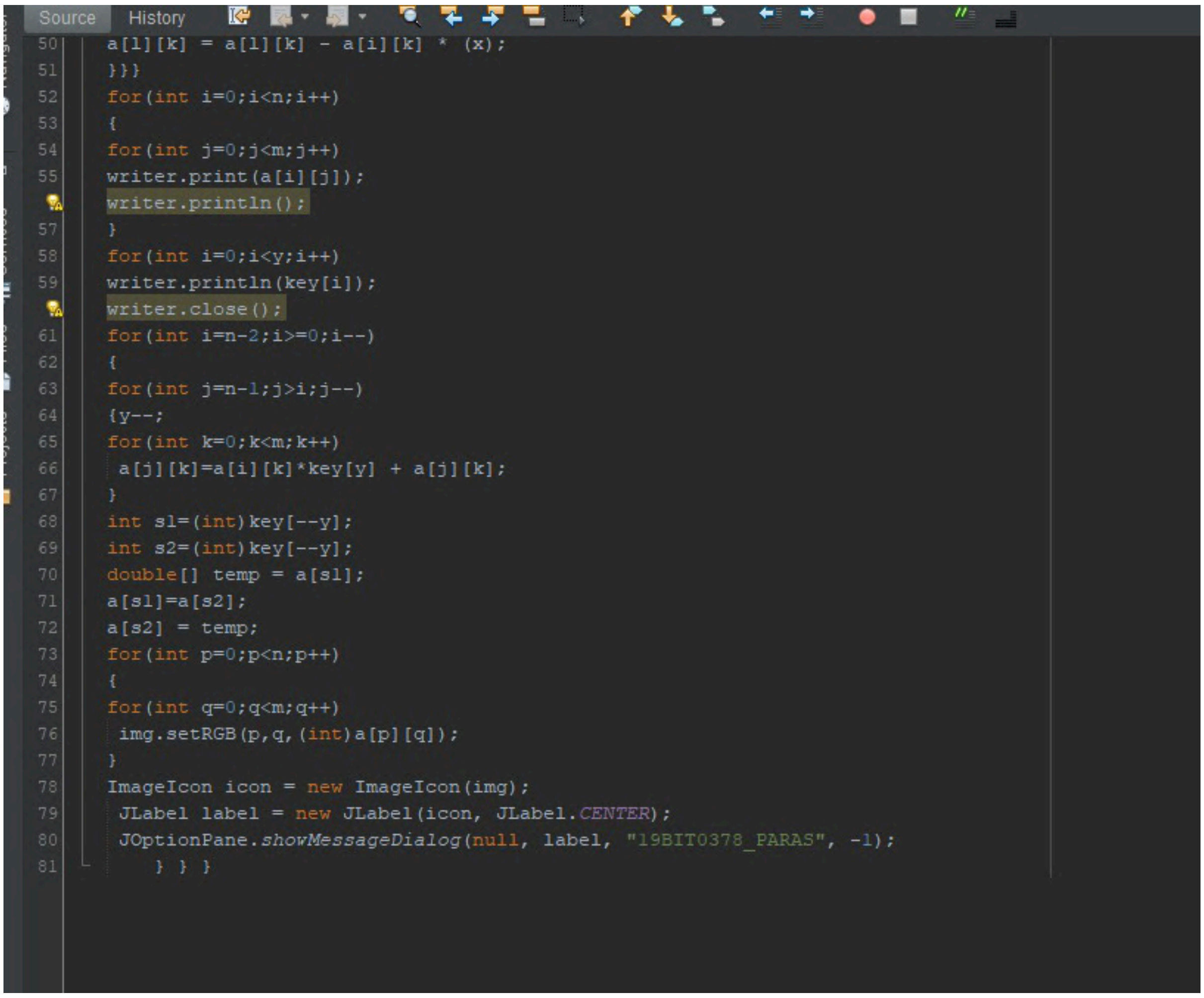
- Key matrix consists of, for each row: value of indices of the row with which this row isexchanged and the multiplier associated with this row.
- Ciphered image and key are transmitted to the receiver’s side.
- Then the program waits for decryption signals.
- As soon as the user approves decryption, the encrypted image matrix and the decryption key are multiplied so as to get back the original image matrix. In this step, all the operations are reversed with the help pf decryption key provided.
- The original matrix is converted back into image, image is executed, and decryption process completes.
- Since algorithm used is irreversible, brute force attack is prevented.
- Attacker cannot break the ciphered image as it is totally different from original image and algorithm used is reversible only if decryption key is known.
- Further, key size is very large and hence brute force attack is prevented.

CODE:

```
package ism_project;
import java.io.*;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;
public class Nis_project {
 public static void main(String args[])throws IOException{
 BufferedImage img = null;
 File f = null;
 try{
 f=new File("C:\\Users\\paras\\Downloads\\167-1677865_facebook-button-
image-facebook-small-icon-png-clipart.png");
 img = ImageIO.read(f);
 }catch(IOException e){
 System.out.println(e);
 }

 int n= img.getWidth();
 int m = img.getHeight();
 double a[][]=new double[n][m];
 for(int i=0;i<n;i++)
 {
 for(int j=0;j<m;j++)
 a[i][j]=img.getRGB(i,j);
 }
int keyLength= (n-1)*n/2 + n*2;
double key[]=new double[keyLength];
int y=0;
PrintWriter writer = new PrintWriter("D:\\key2.txt", "UTF-8");
for(int i=0;i<n-1;i++)
{
int max=i;
for (int j = i; j < n; j++){

if (Math.abs(a[j][i]) > a[max][i] )
max= j;
if (Math.abs(a[max][i]) < 0.00001)

 continue;
}
double[] temp = a[max];
a[max]=a[i];
a[i] = temp;
key[y++]=i;
key[y++]=max;
for(int l=i+1;l<n;l++)
{
 double x=a[l][i]/a[i][i];
 key[y++]=x;
 for(int k=i;k<m;k++){
a[l][k] = a[l][k] - a[i][k] * (x);
}}}
for(int i=0;i<n;i++)
{
for(int j=0;j<m;j++)
writer.print(a[i][j]);
writer.println();
}
for(int i=0;i<y;i++)
writer.println(key[i]);
writer.close();
for(int i=n-2;i>=0;i--)
{
for(int j=n-1;j>i;j--)
{y--;
for(int k=0;k<m;k++)
 a[j][k]=a[i][k]*key[y] + a[j][k];
}
int s1=(int)key[--y];
int s2=(int)key[--y];
double[] temp = a[s1];
a[s1]=a[s2];
a[s2] = temp;
for(int p=0;p<n;p++)
{
for(int q=0;q<m;q++)
 img.setRGB(p,q,(int)a[p][q]);
}
ImageIcon icon = new ImageIcon(img);
 JLabel label = new JLabel(icon, JLabel.CENTER);
 JOptionPane.showMessageDialog(null, label, "19BIT0378_PARAS", -1);
    } } }
```

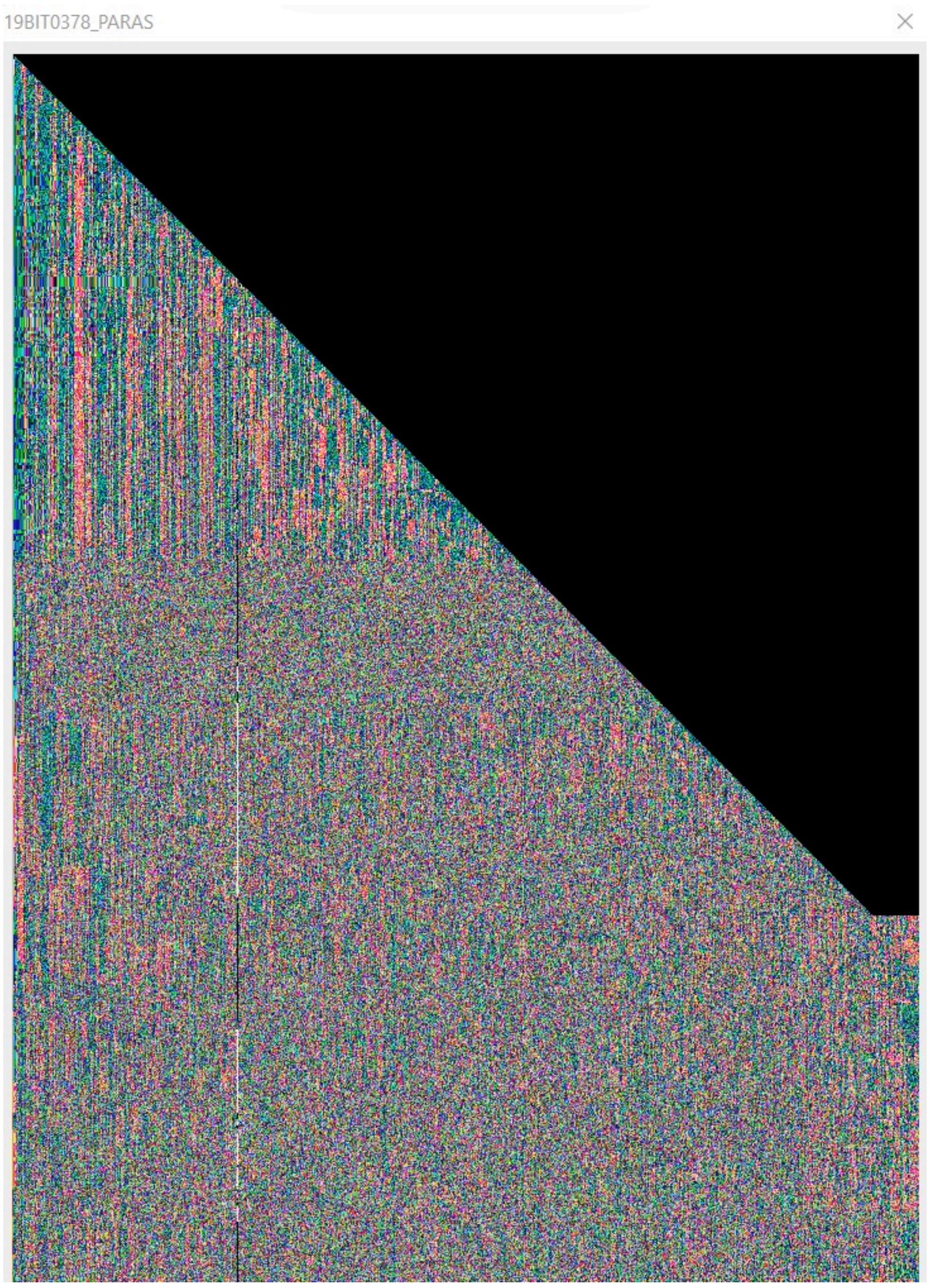


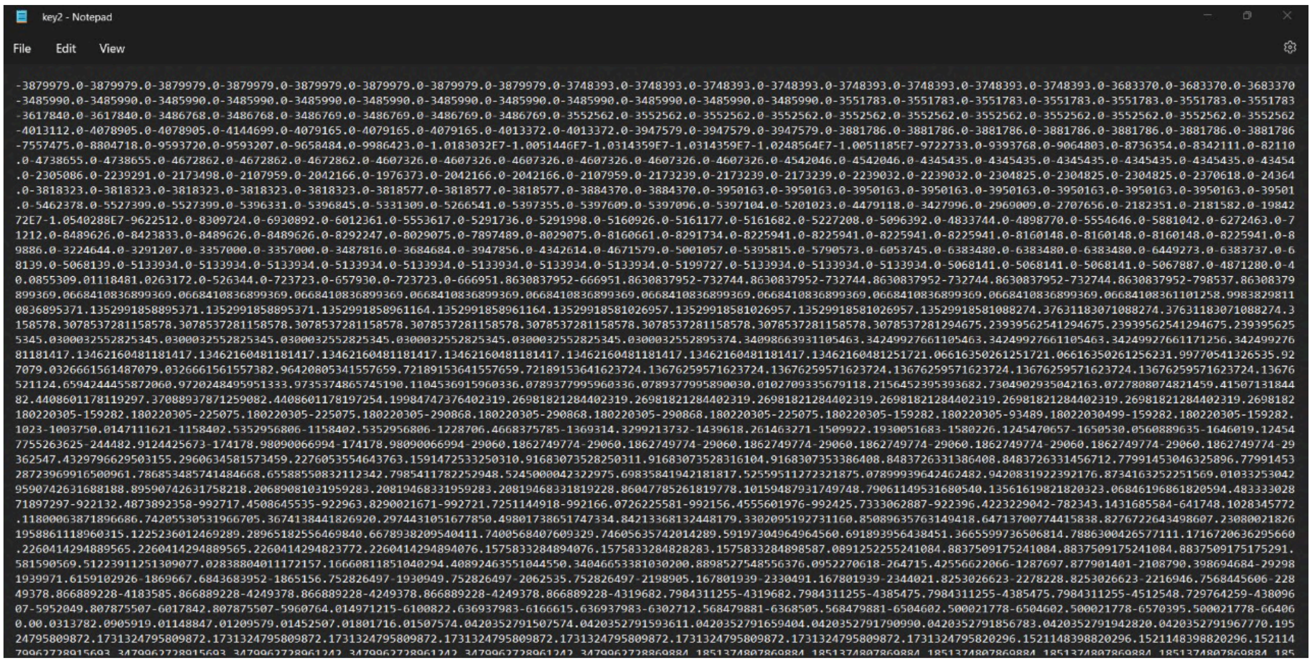
```java
max= j;
if (Math.abs(a[max][i]) < 0.00001)

 continue;
}
double[] temp = a[max];
a[max]=a[i];
a[i] = temp;
key[y++]=i;
key[y++]=max;
for(int l=i+1;l<n;l++)
{
 double x=a[l][i]/a[i][i];
 key[y++]=x;
 for(int k=i;k<m;k++){
a[l][k] = a[l][k] - a[i][k] * (x);
}}}
for(int i=0;i<n;i++)
{
for(int j=0;j<m;j++)
writer.print(a[i][j]);
writer.println();
}
for(int i=0;i<y;i++)
writer.println(key[i]);
writer.close();
for(int i=n-2;i>=0;i--)
{
for(int j=n-1;j>i;j--)
{y--;
for(int k=0;k<m;k++)
 a[j][k]=a[i][k]*key[y] + a[j][k];
}
int s1=(int)key[--y];
int s2=(int)key[--y];
double[] temp = a[s1];
a[s1]=a[s2];
a[s2] = temp;
for(int p=0;p<n;p++)
{
for(int q=0;q<m;q++)
 img.setRGB(p,q,(int)a[p][q]);
}
ImageIcon icon = new ImageIcon(img);
```



```
a[l][k] = a[l][k] - a[i][k] * (x);
}}}
for(int i=0;i<n;i++)
{
for(int j=0;j<m;j++)
writer.print(a[i][j]);
writer.println();
}
for(int i=0;i<y;i++)
writer.println(key[i]);
writer.close();
for(int i=n-2;i>=0;i--)
{
for(int j=n-1;j>i;j--)
{y--;
for(int k=0;k<m;k++)
 a[j][k]=a[i][k]*key[y] + a[j][k];
}
int s1=(int)key[--y];
int s2=(int)key[--y];
double[] temp = a[s1];
a[s1]=a[s2];
a[s2] = temp;
for(int p=0;p<n;p++)
{
for(int q=0;q<m;q++)
 img.setRGB(p,q,(int)a[p][q]);
}
ImageIcon icon = new ImageIcon(img);
 JLabel label = new JLabel(icon, JLabel.CENTER);
 JOptionPane.showMessageDialog(null, label, "19BIT0378_PARAS", -1);
    } } }
```

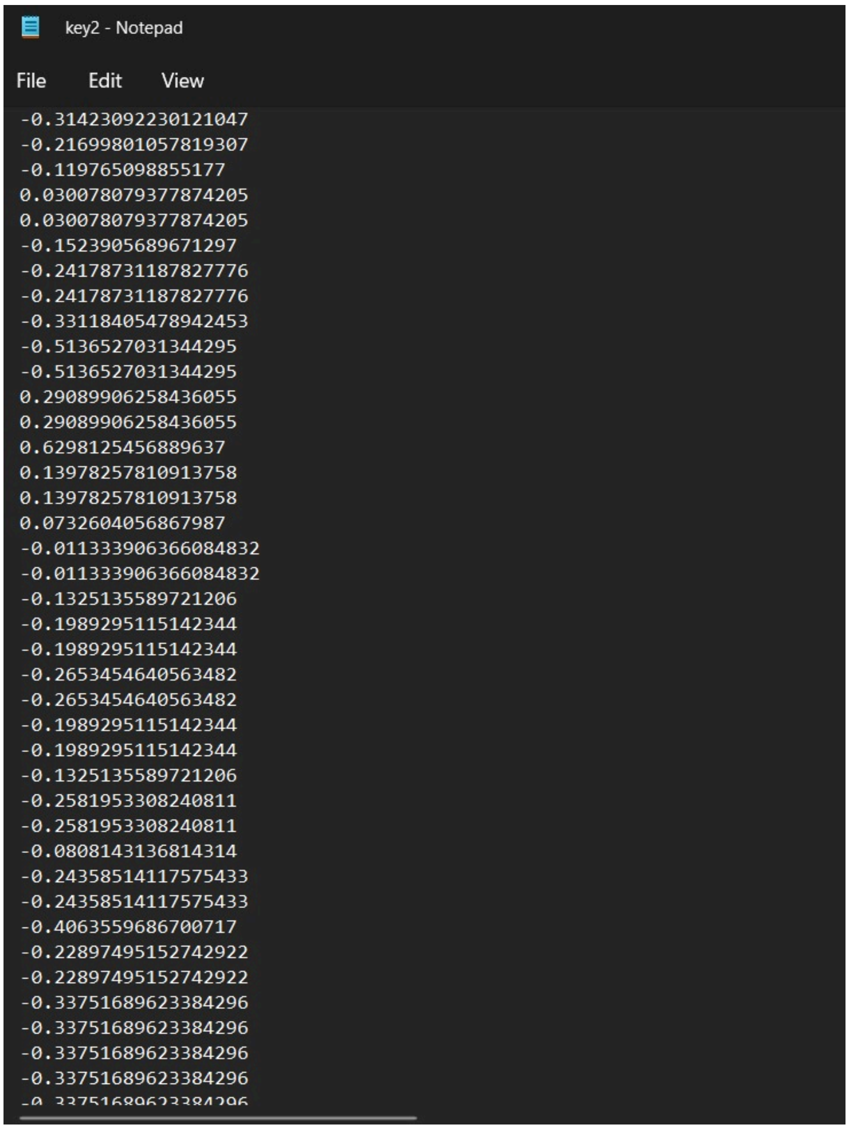
19BIT0378_PARAS




key2 - Notepad

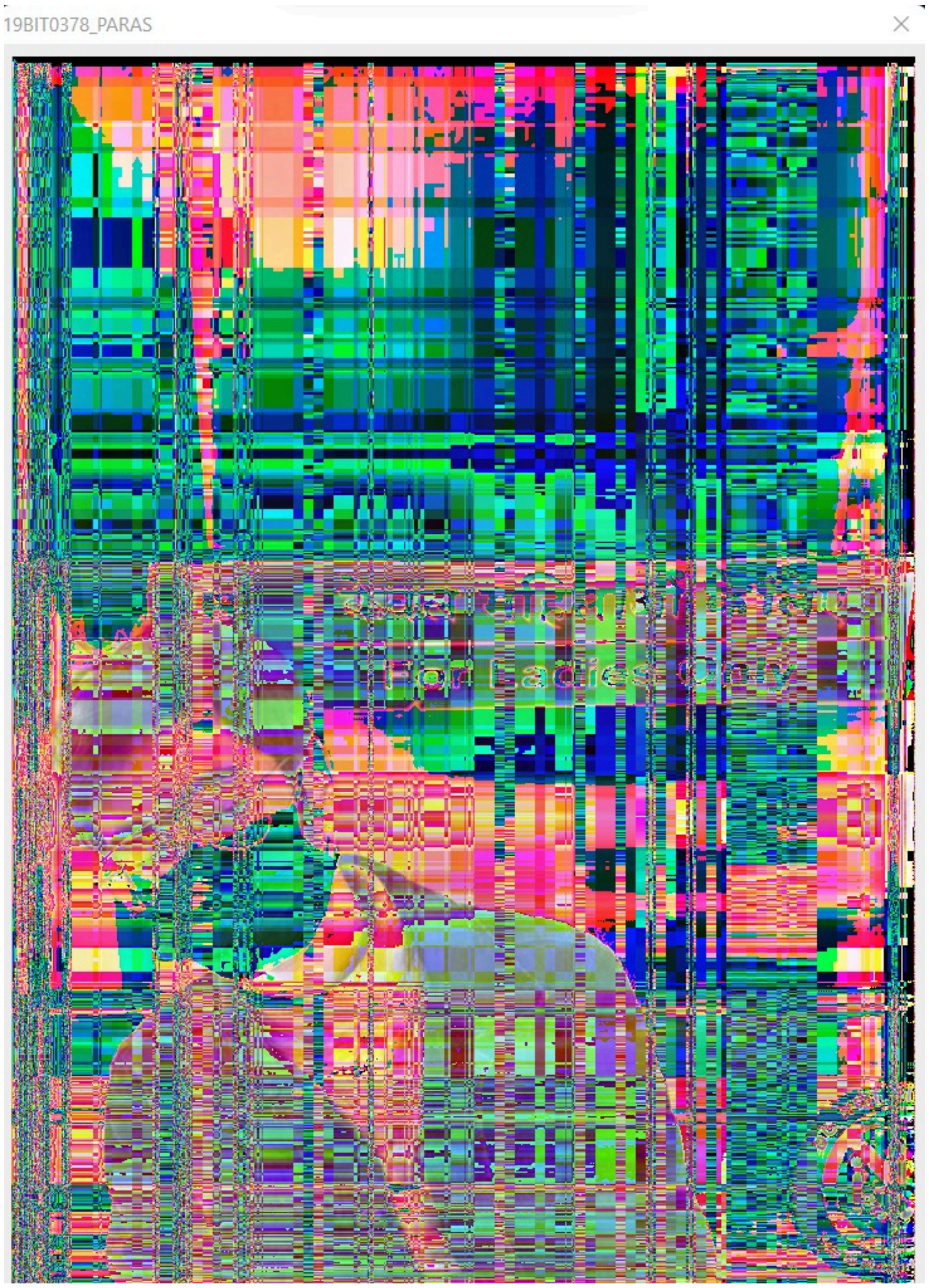
File Edit View

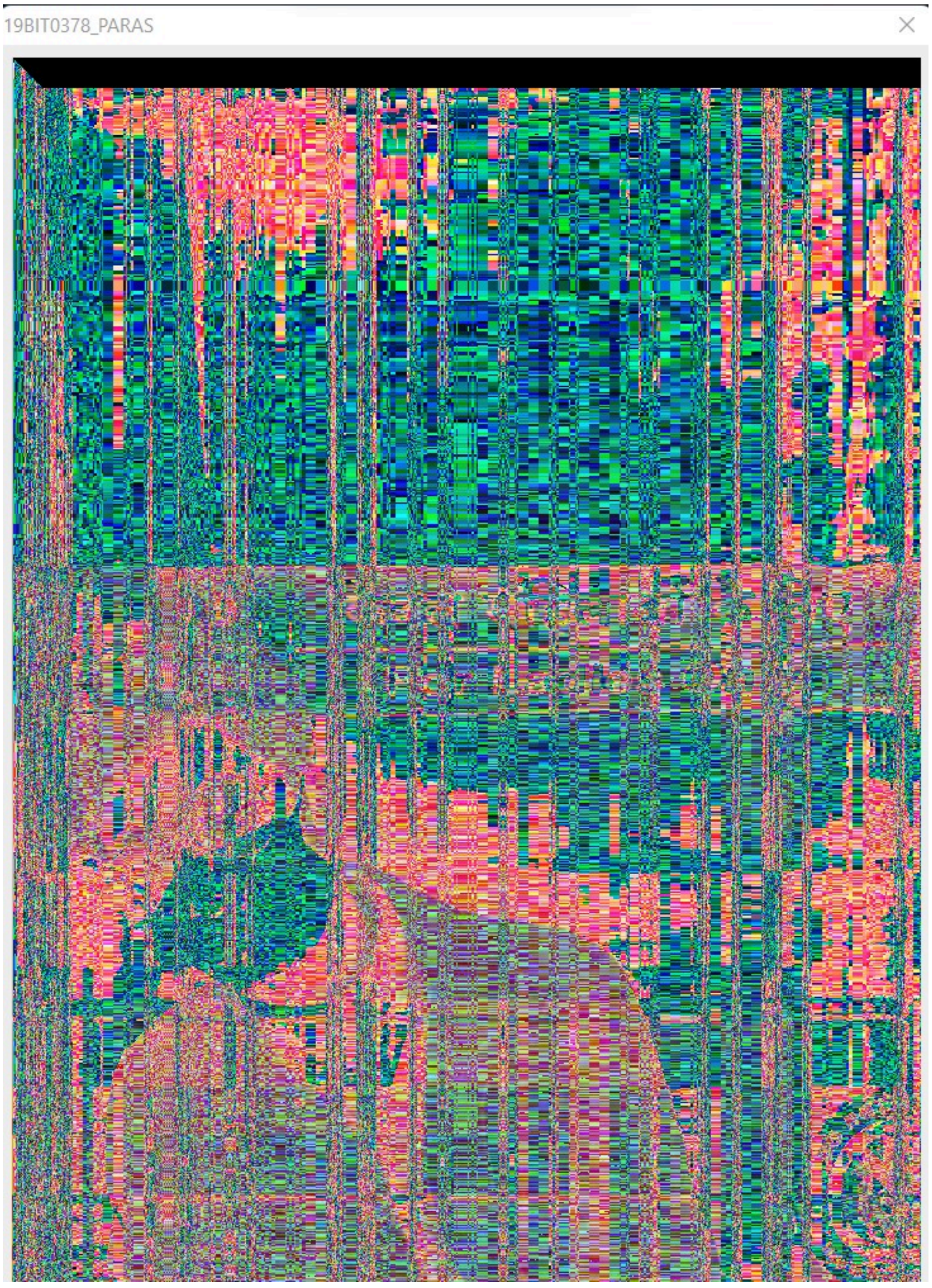
```
-3879979.0-3879979.0-3879979.0-3879979.0-3879979.0-3879979.0-3879979.0-3879979.0-3748393.0-3748393.0-3748393.0-3748393.0-3748393.0-3748393.0-3748393.0-3748393.0-3683370.0-3683370.0-3683370
-3485990.0-3485990.0-3485990.0-3485990.0-3485990.0-3485990.0-3485990.0-3485990.0-3485990.0-3485990.0-3485990.0-3485990.0-3551783.0-3551783.0-3551783.0-3551783.0-3551783.0-3551783.0-3551783
-3617840.0-3617840.0-3486768.0-3486768.0-3486769.0-3486769.0-3486769.0-3486769.0-3552562.0-3552562.0-3552562.0-3552562.0-3552562.0-3552562.0-3552562.0-3552562.0-3552562.0-3552562.0-3552562
-4013112.0-4078905.0-4078905.0-4144699.0-4079165.0-4079165.0-4079165.0-4013372.0-4013372.0-3947579.0-3947579.0-3947579.0-3881786.0-3881786.0-3881786.0-3881786.0-3881786.0-3881786.0-3881786
-7557475.0-8804718.0-9593720.0-9593207.0-9658484.0-9986423.0-1.0183032E7-1.0051446E7-1.0314359E7-1.0314359E7-1.0248564E7-1.0051185E7-9722733.0-9393768.0-9064803.0-8736354.0-8342111.0-82110
.0-4738655.0-4738655.0-4672862.0-4672862.0-4672862.0-4607326.0-4607326.0-4607326.0-4607326.0-4607326.0-4607326.0-4542046.0-4542046.0-4345435.0-4345435.0-4345435.0-4345435.0-4345435.0-43454
.0-2305086.0-2239291.0-2173498.0-2107959.0-2042166.0-1976373.0-2042166.0-2042166.0-2107959.0-2173239.0-2173239.0-2173239.0-2239032.0-2239032.0-2304825.0-2304825.0-2304825.0-2370618.0-24364
.0-3818323.0-3818323.0-3818323.0-3818323.0-3818323.0-3818577.0-3818577.0-3818577.0-3884370.0-3884370.0-3950163.0-3950163.0-3950163.0-3950163.0-3950163.0-3950163.0-3950163.0-39501
.0-5462378.0-5527399.0-5527399.0-5396331.0-5396845.0-5331309.0-5266541.0-5397355.0-5397609.0-5397096.0-5397104.0-5201023.0-4479118.0-3427996.0-2969009.0-2707656.0-2182351.0-2181582.0-19842
72E7-1.0540288E7-9622512.0-8309724.0-6930892.0-6012361.0-5553617.0-5291736.0-5291998.0-5160926.0-5161177.0-5161682.0-5227208.0-5096392.0-4833744.0-4898770.0-5554646.0-5881042.0-6272463.0-7
1212.0-8489626.0-8423833.0-8489626.0-8489626.0-8292247.0-8029075.0-7897489.0-8029075.0-8160661.0-8291734.0-8225941.0-8225941.0-8225941.0-8225941.0-8160148.0-8160148.0-8160148.0-8225941.0-8
9886.0-3224644.0-3291207.0-3357000.0-3357000.0-3487816.0-3684684.0-3947856.0-4342614.0-4671579.0-5001057.0-5395815.0-5790573.0-6053745.0-6383480.0-6383480.0-6383480.0-6449273.0-6383737.0-6
8139.0-5068139.0-5133934.0-5133934.0-5133934.0-5133934.0-5133934.0-5133934.0-5133934.0-5133934.0-5199727.0-5133934.0-5133934.0-5133934.0-5068141.0-5068141.0-5068141.0-5067887.0-4871280.0-4
0.0855309.01118481.0263172.0-526344.0-723723.0-657930.0-723723.0-666951.8630837952-666951.8630837952-732744.8630837952-732744.8630837952-732744.8630837952-732744.8630837952-798537.86308379
899369.0668410836899369.0668410836899369.0668410836899369.0668410836899369.0668410836899369.0668410836899369.0668410836899369.0668410836899369.0668410836899369.06684108361101258.9983829811
0836895371.1352991858895371.1352991858895371.1352991858961164.1352991858961164.13529918581026957.13529918581026957.13529918581026957.13529918581088274.37631183071088274.37631183071088274.3
158578.3078537281158578.3078537281158578.3078537281158578.3078537281158578.3078537281158578.3078537281158578.3078537281158578.3078537281294675.2393956254129 4675.2393956254129 4675.239395625
5345.0300032552825345.0300032552825345.0300032552825345.0300032552825345.0300032552825345.0300032552895374.34098663931105463.34249927661105463.34249927661105463.34249927661171256.342499276
81181417.13462160481181417.13462160481181417.13462160481181417.13462160481181417.13462160481181417.13462160481181417.13462160481251721.06616350261251721.06616350261256231.99770541326535.92
7079.0326661561487079.0326661561557382.96420805341557659.72189153641557659.72189153641623724.13676259571623724.13676259571623724.13676259571623724.13676259571623724.13676259571623724.13676
521124.6594244455872060.9720248495951333.9735374865745190.1104536915960336.0789377995960336.0789377995890030.0102709335679118.2156452395393682.7304902935042163.0727808074821459.41507131844
82.4408601178119297.37088937871259082.4408601178197254.19984747376402319.26981821284402319.26981821284402319.26981821284402319.26981821284402319.26981821284402319.26981821284402319.2698182
180220305-159282.180220305-225075.180220305-225075.180220305-290868.180220305-290868.180220305-290868.180220305-225075.180220305-159282.180220305-93489.18022030499-159282.180220305-159282.
1023-1003750.014711621-1158402.5352956806-1158402.5352956806-1228706.4668375785-1369314.3299213732-1439618.261463271-1509922.1930051683-1580226.1245470657-1650530.0560889635-1646019.12454
7755263625-244482.9124425673-174178.98090066994-174178.98090066994-29060.1862749774-29060.1862749774-29060.1862749774-29060.1862749774-29060.1862749774-29060.1862749774-29060.1862749774-29
362547.4329796629503155.2960634581573459.2276053554643763.1591472533250310.91683073528250311.91683073528316104.9168307353386408.8483726331386408.8483726331456712.77991453046325896.77991453
28723969916500961.786853485741484668.65588550832112342.79854117822529 48.52450000423229 75.69835841942181817.52559511272321875.07899939642462482.94208319223921 76.87341632522515 69.01033253042
9590742631688188.89590742631758218.20689081031959283.20819468331959283.20819468331819228.86047785261819778.10159487931749748.79061149531680540.13561619821820323.06846196861820594.483333028
71897297-922132.4873892358-992717.4508645535-922963.8290021671-992721.7251144918-992166.0726225581-992156.4555601976-992425.7333062887-922396.4223229042-782343.1431685584-641748.1028345772
.11800063871896686.74205530531966705.3674138441826920.2974431051677850.49801738651747334.84213368132448179.3302095192731160.85089635763149418.64713700774415838.8276722643498607.23080021826
1958861118960315.1225236012469289.28965182556469840.6678938209540411.7400568407609329.74605635742014289.59197304964964560.691893956438451.3665599736506814.7886300426577111.17167206362956 60
.2260414294889565.2260414294889565.2260414294823772.2260414294894076.1575833284894076.1575833284828283.1575833284898587.0891252255241084.8837509175241084.8837509175241084.8837509175175291.
581590569.51223911251309077.02838804011172157.16660811851040294.40892463551044550.3404665338103020 0.8898527548556376.0952270618-264715.42556622066-1287697.877901401-2108790.398694684-29298
1939971.6159102926-1869667.6843683952-1865156.752826497-1930949.752826497-2062535.752826497-2198905.167801939-2330491.167801939-2344021.8253026623-2278228.8253026623-2216946.7568445606-228
49378.866889228-4183585.866889228-4249378.866889228-4249378.866889228-4249378.866889228-4319682.7984311255-4319682.7984311255-4385475.7984311255-4385475.7984311255-4512548.729764259-438096
07-5952049.807875507-6017842.807875507-5960764.014971215-6100822.636937983-6166615.636937983-6302712.568479881-6368505.568479881-6504602.500021778-6504602.500021778-6570395.500021778-66406
0.00.0313782.0905919.01148847.01209579.01452507.01801716.01507574.0420352791507574.0420352791593611.0420352791659404.0420352791790990.0420352791856783.0420352791942820.0420352791967770.195
24795809872.1731324795809872.1731324795809872.1731324795809872.1731324795809872.1731324795809872.1731324795809872.1731324795809872.1731324795820296.1521148398820296.1521148398820296.152114
79962728915693.34799627728915693.34799627728961242.34799627728961242.34799627728869884.1851374807869884.1851374807869884.1851374807869884.1851374807869884.1851374807869884.185
```


key2 - Notepad

File Edit View

```
-0.31423092230121047
-0.21699801057819307
-0.119765098855177
0.030078079377874205
0.030078079377874205
-0.1523905689671297
-0.24178731187827776
-0.24178731187827776
-0.33118405478942453
-0.5136527031344295
-0.5136527031344295
0.29089906258436055
0.29089906258436055
0.6298125456889637
0.13978257810913758
0.13978257810913758
0.0732604056867987
-0.011333906366084832
-0.011333906366084832
-0.1325135589721206
-0.1989295115142344
-0.1989295115142344
-0.2653454640563482
-0.2653454640563482
-0.1989295115142344
-0.1989295115142344
-0.1325135589721206
-0.2581953308240811
-0.2581953308240811
-0.0808143136814314
-0.24358514117575433
-0.24358514117575433
-0.4063559686700717
-0.22897495152742922
-0.22897495152742922
-0.33751689623384296
-0.33751689623384296
-0.33751689623384296
-0.33751689623384296
-0.33751689623384296
```


19BIT0378_PARAS


19BIT0378_PARAS

## IMAGE ENCRYPTION AND DECRYPTION RSA

### INTRODUCTION

The RSA is an cryptographic algorithm which is use to encrypt and decrypt the data. Here we are Implementing RSA Algorithm on an IMAGE to encrypt and decrypt using two keys, Private key and Public Key.

## ABSTRACT

The modified RSA algorithm is suitable for RGB image encryption. The test result shows that the proposed method can encrypt / remove various encryption successfully images, and the algorithm has a good encryption effect. Cipher image enhanced by this method will they are quite different from the original image. This method provides better security and will qualify for secure transfer of images via the Internet.

## IMPLEMENTATION:

1.Read the input RGB color image, S

2. First choose the two distinct prime numbers p and q.

3. Calculate the value, $n = pq$.

4. Compute $\phi(n) = \phi(p)\phi(q) = (p - 1)(q - 1) = n - (p + q - 1)$, where $\phi$ is Euler's totient function.

5. Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$; i.e., e and $\phi(n)$ are co-prime.e is the released as the public key.

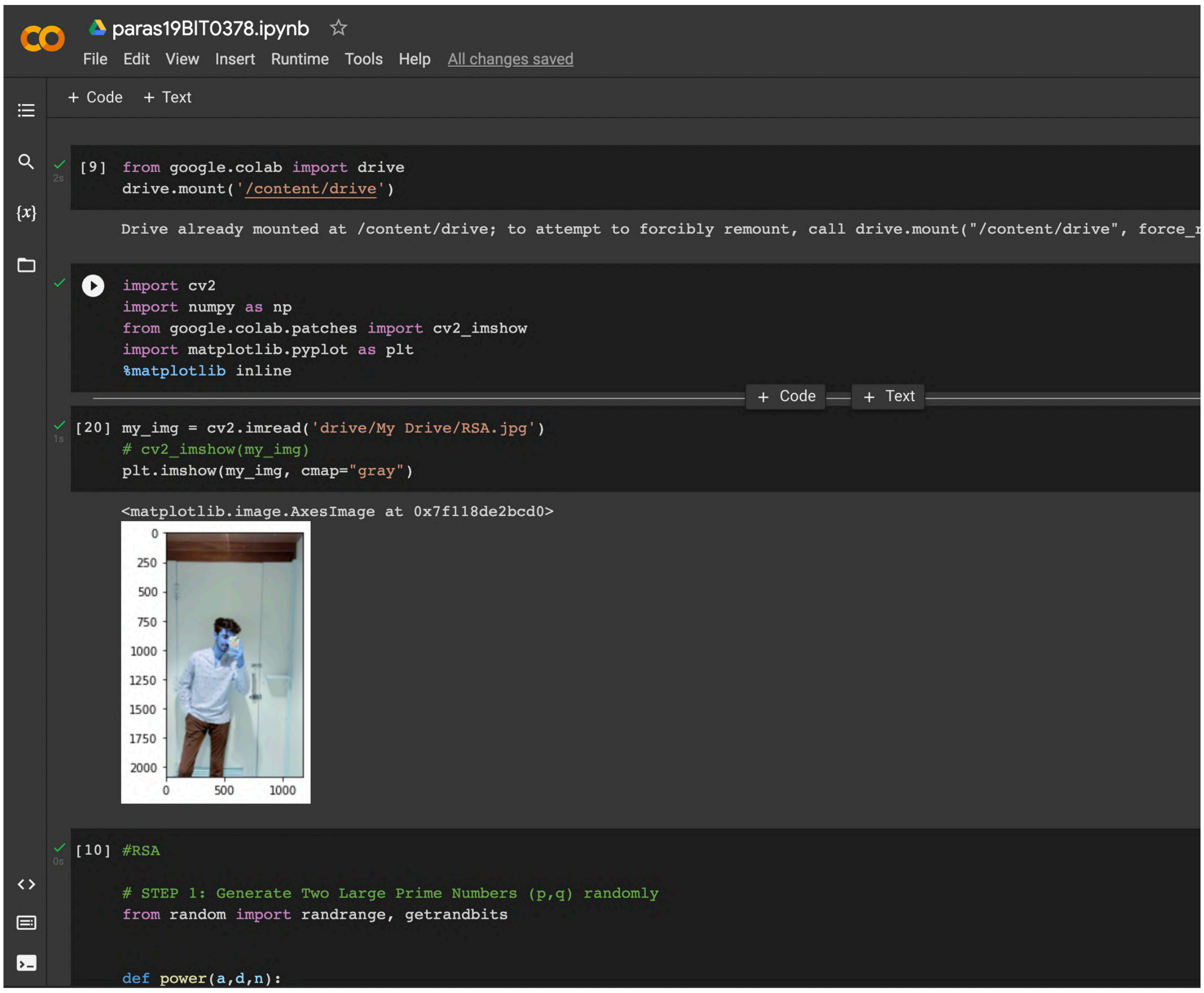
6. Determine d as $d \equiv e -1 \pmod{\phi(n)}$; i.e., d is the modular multiplicative inverse of e (modulo $\phi(n)$).Solve the d given $d \cdot e \equiv 1 \pmod{\phi(n)}$.

7. Obtain the encrypted image, $C = Se \bmod \phi(n)$.

8. $C = C \bmod 256$, as gray level values of an image lie in the range [0,255].

9. Recover decrypted image, $S = Cd \bmod 256$.

10. The original input (recovered) image, $R = S \bmod \phi(n)$.



paras19BIT0378.ipynb

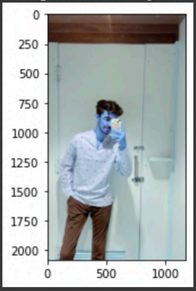
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

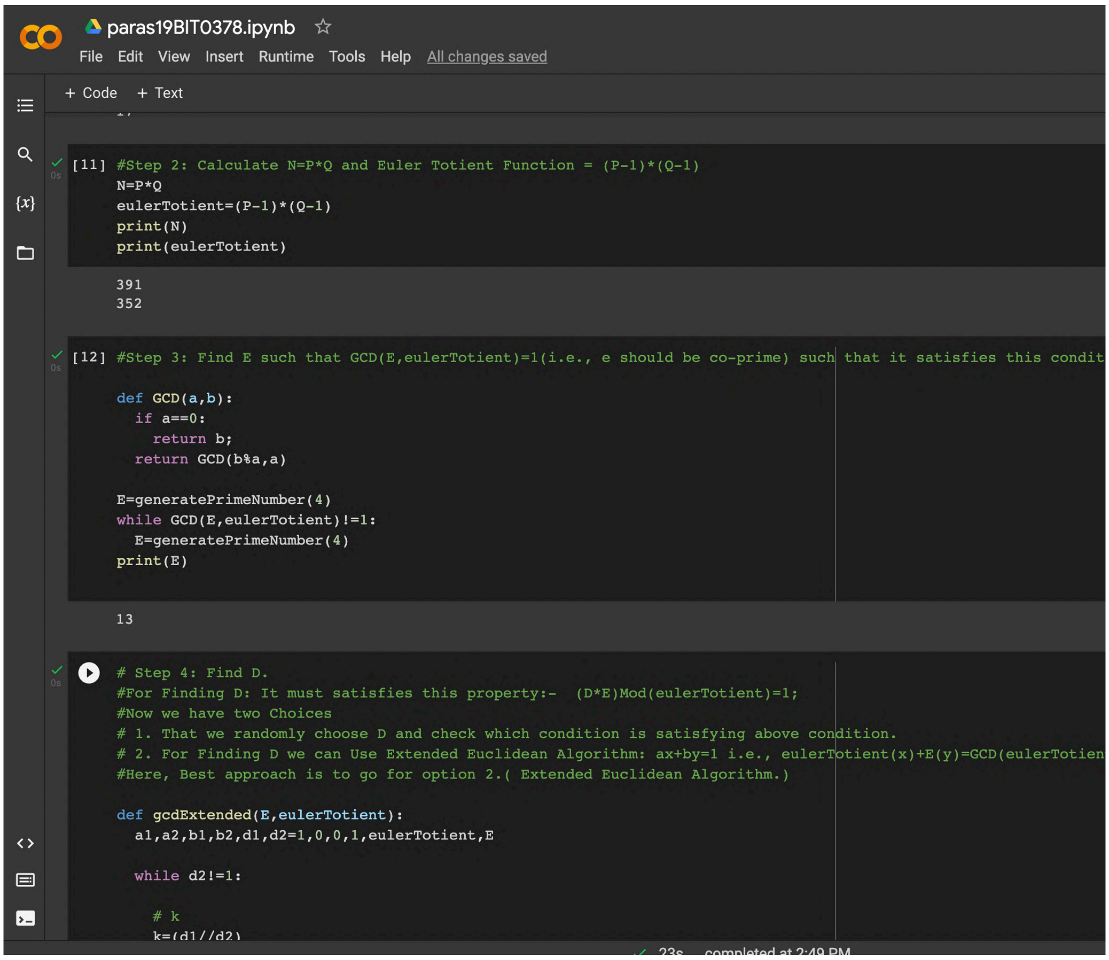
```
[9] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_r
```

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
%matplotlib inline
```

```
[20] my_img = cv2.imread('drive/My Drive/RSA.jpg')
     # cv2_imshow(my_img)
     plt.imshow(my_img, cmap="gray")

     <matplotlib.image.AxesImage at 0x7f118de2bcd0>
```



```
[10] #RSA

     # STEP 1: Generate Two Large Prime Numbers (p,q) randomly
     from random import randrange, getrandbits


     def power(a,d,n):
```



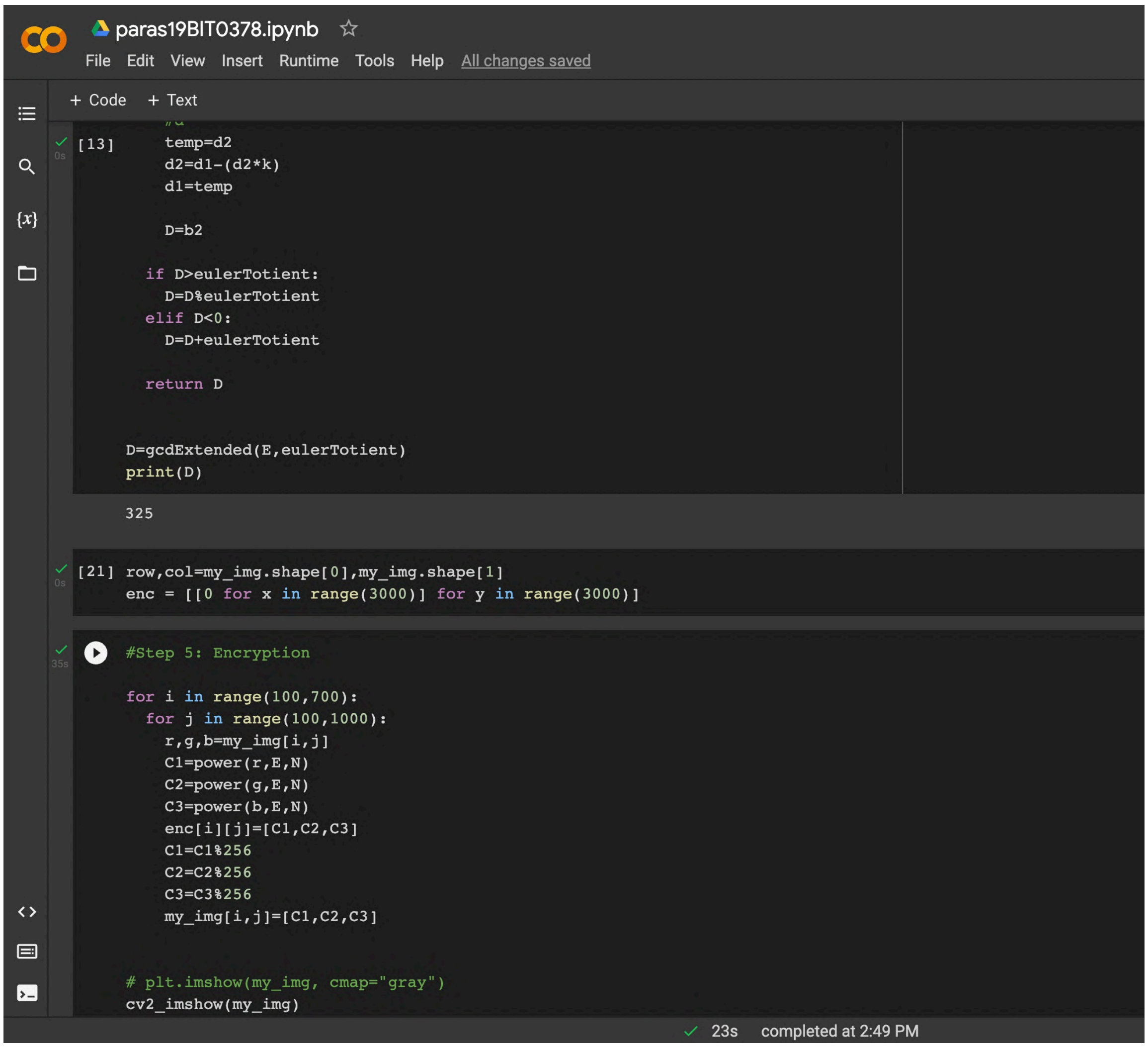
```
[11] #Step 2: Calculate N=P*Q and Euler Totient Function = (P-1)*(Q-1)
     N=P*Q
     eulerTotient=(P-1)*(Q-1)
     print(N)
     print(eulerTotient)
```

```
391
352
```

```
[12] #Step 3: Find E such that GCD(E,eulerTotient)=1(i.e., e should be co-prime) such that it satisfies this condit

     def GCD(a,b):
       if a==0:
         return b;
       return GCD(b%a,a)

     E=generatePrimeNumber(4)
     while GCD(E,eulerTotient)!=1:
       E=generatePrimeNumber(4)
     print(E)
```

```
13
```

```
     # Step 4: Find D.
     #For Finding D: It must satisfies this property:-  (D*E)Mod(eulerTotient)=1;
     #Now we have two Choices
     # 1. That we randomly choose D and check which condition is satisfying above condition.
     # 2. For Finding D we can Use Extended Euclidean Algorithm: ax+by=1 i.e., eulerTotient(x)+E(y)=GCD(eulerTotien
     #Here, Best approach is to go for option 2.( Extended Euclidean Algorithm.)

     def gcdExtended(E,eulerTotient):
       a1,a2,b1,b2,d1,d2=1,0,0,1,eulerTotient,E

       while d2!=1:

         # k
         k=(d1//d2)
```



paras19BIT0378.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

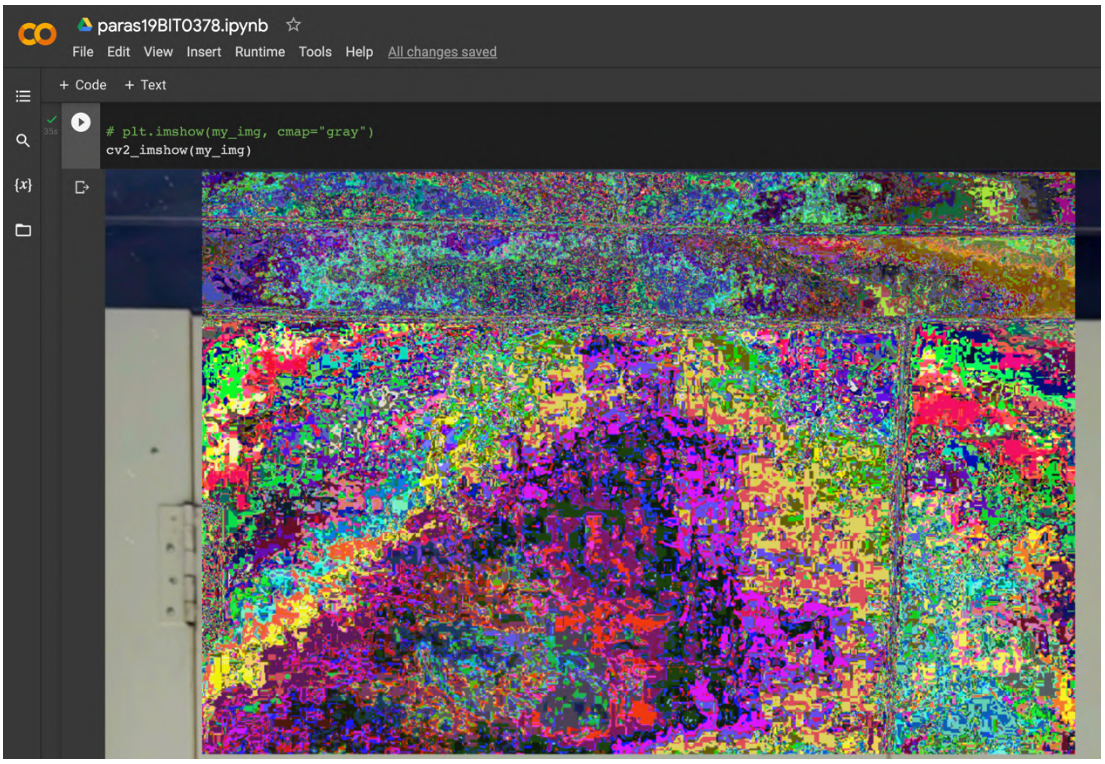
```
[13]
        temp=d2
        d2=d1-(d2*k)
        d1=temp

        D=b2

      if D>eulerTotient:
        D=D%eulerTotient
      elif D<0:
        D=D+eulerTotient

      return D


    D=gcdExtended(E,eulerTotient)
    print(D)
```

```
325
```

```
[21] row,col=my_img.shape[0],my_img.shape[1]
     enc = [[0 for x in range(3000)] for y in range(3000)]
```

```
#Step 5: Encryption

for i in range(100,700):
  for j in range(100,1000):
    r,g,b=my_img[i,j]
    C1=power(r,E,N)
    C2=power(g,E,N)
    C3=power(b,E,N)
    enc[i][j]=[C1,C2,C3]
    C1=C1%256
    C2=C2%256
    C3=C3%256
    my_img[i,j]=[C1,C2,C3]


# plt.imshow(my_img, cmap="gray")
cv2_imshow(my_img)
```

✓ 23s completed at 2:49 PM

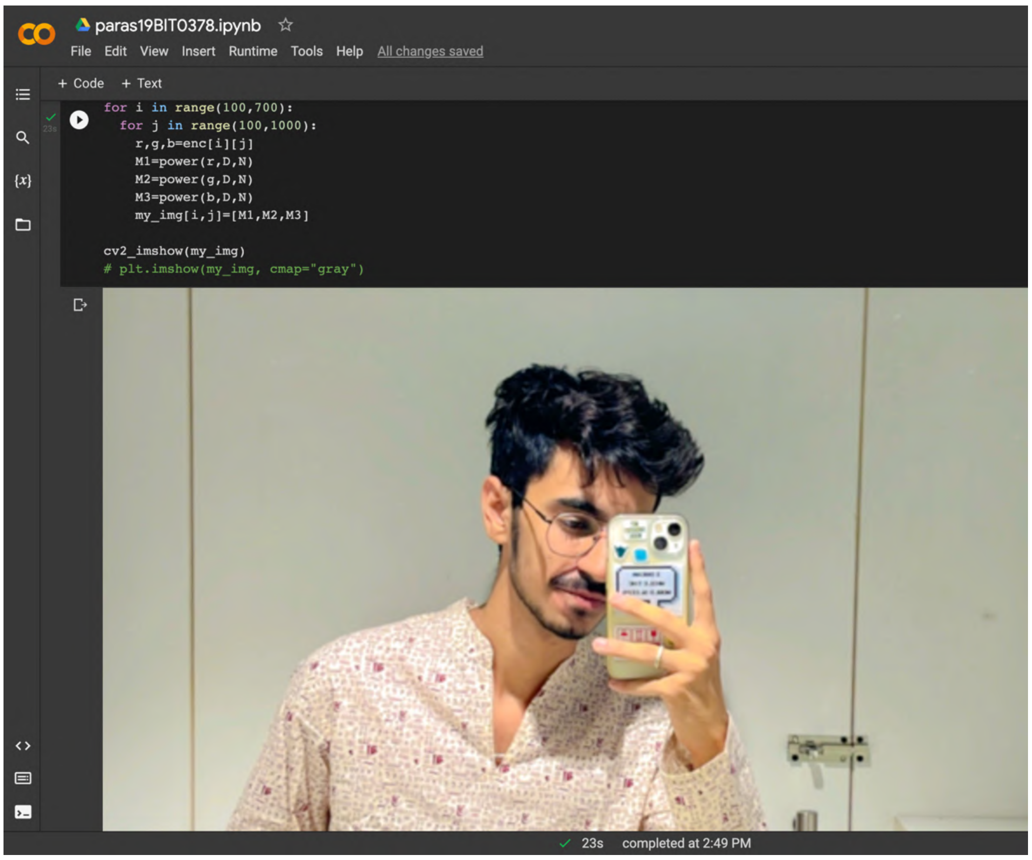


paras19BIT0378.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
# plt.imshow(my_img, cmap="gray")
cv2_imshow(my_img)
```



```
for i in range(100,700):
  for j in range(100,1000):
    r,g,b=enc[i][j]
    M1=power(r,D,N)
    M2=power(g,D,N)
    M3=power(b,D,N)
    my_img[i,j]=[M1,M2,M3]

cv2_imshow(my_img)
# plt.imshow(my_img, cmap="gray")
```

Now We Decrypted the Image

## ADVANTAGES OF RSA

Here, the image encryption and decryption approaches are highly securable and with less computational time.

The results of the simulation prove that this approach is an efficient cryptosystem for image encryption, and it is suitable for secured transmission of images over the Internet.

## ADVANTAGES OF GAUSSIAN ELIMINATION

Simplicity: It is quite simple to implement and cheaper as well.

Ratio: Encryption takes 1⁄4 th time the decryption process takes.

Robust: The encrypted image is hard to hack to obtain the original image as algorithm used is irreversible, thus preventing Brute force attack.

Pixels obtained after Gaussian elimination is fully distorted, so the key is needed for decryption to obtain original image because it’s hard to crack the encrypted form.

We have reduced time taken by encryption smartly by updating row. Exchanges use another matrix. So basically, it’s a Space for time trade-off.

## Conclusion

Due to image powerful attribute such as vast data capacity, data redundancy and great correlation among pixels of image, algorithms used for text encryption can’t be used specifically to images. These algorithms need much time when used with image. However, this requirement is not necessaryfor image data. Due to the characteristic of human perception, a decrypted image containing small distortion is usually acceptable. Therefore, we have tested Gaussian Elimination Row Exchangemethod and RSA to encrypt and decrypt the image as this cater to above issues. Further, these algorithms have their various advantages .A complex project involving the conversion of image into matrix form, using a mathematical concept to encrypt and decrypt it, was instrumental in giving us a thorough understanding of how the concepts of ADA and Linear Algebra together can actually be implemented in the real world and also RSA algorithm is modified for colour image encryption. By the end of the project, we have gained valuable skills including a grounding of how to interact with the operating system, file handling in java,optimizing algorithms, calculating the efficiencies, and learning how to form and manipulate images.

## G-drive link to the video

https://drive.google.com/file/d/1_oSYzrwVUolu7z0nafcLVSHRGRpQ_MJ7/view?usp=sharing

## References

https://en.wikipedia.org/wiki/Encryption

Nehal Kandele, Shrikant Tiwari “A New Combined Symmetric Key Cryptography CRDDBT Using - Relative Displacement (RDC) and Dynamic Base Transformation (DBTC)”, International Journal of Engineering Research & Technology, Vol.2 - Issue 10 (October - 2013)( 2278-0181)

Quist-Aphetsi Keste,“ Image Encryption based on the RGB PIXEL Transposition and Shuffling” I.J.Computer Network and Information Security, 2013,7, 43-50,2013.

Reji Mathews, Amnesh Goel, Prachur Saxena & Ved Prakash Mishra, "Image Encryption Based on Explosive Inter-pixel Displacement of the RGB Attributes of a PIXEL", Proceedings of the World Congress on Engineering and Computer Science 2011 Vol I WCECS 2011, October 19-21, 2011, San Francisco, USA. ISBN: 978-988-18210-9-6.

J. Katz, Y. Lindell, "Introduction to Modern Cryptography," 2nd edition, Boca Raton : CRC Press/Taylor & Francis, pages 583, 2015.

Yarmolik, V. N.& S. N Demidenko, “ Generation and Application of Pseudo Random Sequences for Random Testing “ , John Wiley &Suns, 1988.

Jackie Nicholas," The inverse of a n _ n matrix", Mathematics Learning Centre University of Sydney,2010.

Ozturk, I.Sogukpinar, "Analysis and comparison of image encryption algorithm," Journal of transactions on engineering, computing and technology December, vol. 3, pp.38, 2004.

William Stalling, “ Cryptography and Network Security Principle and Practices, Fourth Edition”, Prentice Hall, November 16, 2005

Komal D Patel , Sonal Belani, Image Encryption Using Different Techniques: A Review International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, Volume 1, Issue 1, November 2011.

Ahmad Abusukhon and Mohammad Talib, “A Novel Network Security Algorithm Based on Private Key Encryption”, IEEE 2012.

Anal Paul, Nibaran Das and Agyan Kumar Prusty, “An Advanced Gray Image Encryption Scheme by Using Discrete Logarithm with Logistic and HEH64 Chaotic Functions”, IEEE 2012.

Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, Vol 21, No. 2, February 1978, p. 120-26.

Hiral Rathod, Mahendra Singh Sisodia, Sanjay Kumar Sharma, Design and Implementation of Image Encryption Algorithm by using Block Based Symmetric Transformation Algorithm (Hyper Image Encryption Algorithm), ISSN 2249-6343 International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 1, Issue 3.