International Journal of Advance Research in Science and Engineering
Vol. No.6, Special Issue (01), September 2017, BVCNSCS 2017
www.ijarse.com
IJARSE
ISSN 2319 - 8354

Encryption and Decryption Of Image using Triangular Matrix in Hill Cipher Algorithm

Mrs. G Santhi Priya
Lecturer in Mathematics
Bhavan's Vivekananda College of Science, Humanities &Commerce, Sainikpuri, Secunderabad. Telangana

Mrs. K Padma Priya
Lecturer in Computer Science
Bhavan's Vivekananda College of Science, Humanities &Commerce, Sainikpuri, Secunderabad, Telangana

Abstract: The Hill cipher Algorithm is one of the Symmetric Key Algorithms in Data Encryption. It uses a pair of key matrices, a key on one side in Encryption and its inverse on the other side. Sometimes finding either an invertible matrix or finding the inverse of a matrix may be complex in calculations. In this paper to encrypt and decrypt a given image it is proposed to use a particular type of triangular matrix with all elements in Z_n, as key matrices, whose inverse can be found using properties of Modular Arithmetic algorithms.

Key words: Hill Cipher, Residue Matrices, Congruence modulo operator (mod), Multiplicative and additive inverses in Z_n, Inverse of Triangular Matrix, Pixel Matrix.

Introdction: In Hills Algorithm to encrypt a given plain text we use a key matrix A with entries in Z_m. In this Paper it is proposed to use a particular type of triangular matrix A where A^{-1} can be found in a simple way using modular arithmetic.

Upper Triangular Matrix:

Consider an upper triangular Matrix :

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots\ldots\ldots & a_{1n} \\ 0 & a_{22} & a_{23} & \ldots\ldots\ldots & a_{2n} \\ 0 & 0 & a_{33} & \ldots\ldots\ldots & a_{3n} \\ . & . & . & . & . \\ 0 & 0 & 0 & 0 & a_{nn} \end{pmatrix} = \begin{pmatrix} a_1 & a_1 & a_1 & \ldots\ldots\ldots & a_1 \\ 0 & a_2 & a_2 & \ldots\ldots\ldots & a_2 \\ 0 & 0 & a_3 & \ldots\ldots\ldots & a_3 \\ . & . & . & . & . \\ 0 & 0 & 0 & 0 & a_n \end{pmatrix}$$

Then A^{-1} will be also an upper triangular matrix can be found using properties of Modular arithmetic.

Pixels Location and Matrix

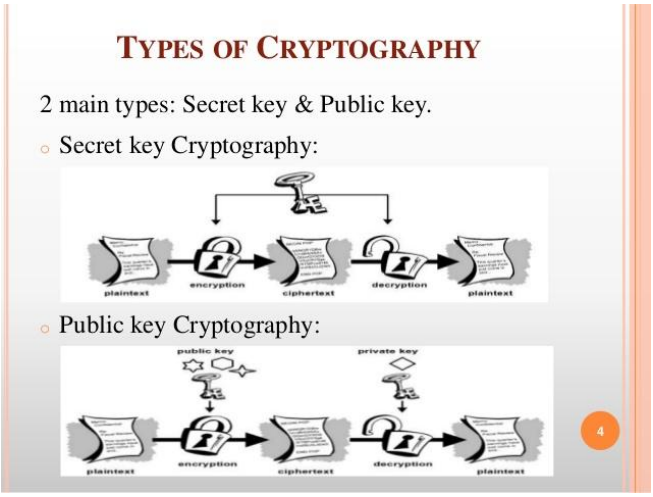
In many image processing applications, the XY location of the pixels has to be done with a one dimensional pixel array by identifying the column and row that any given pixel is located in. In programming with pixels, every pixel is considered as living in a two dimensional world, but continue to access the data in one. **Objective**: By converting a given Image into a matrix containing location of pixels and using the matrix as cipher we propose to encrypt a given Image.

I INTRODUCTION

Nowadays Internet is used for faster transmission of huge amount of important and valuable data.
Extension of the current Internet and providing connection, communication, and inter-networking between devices and physical objects, or "Things," is a growing trend is referred to as the "Internet *of Things".*

Information security demands for more importantance in data storage and transmission.
Cryptography is considered [1] to be a branch of Mathematics and Computer Science that is affiliated with Information theory, Computer Security and Engineering.

International Journal of Advance Research in Science and Engineering
Vol. No.6, Special Issue (01), September 2017, BVCNSCS 2017
www.ijarse.com
IJARSE
ISSN 2319 - 8354



A cryptographic algorithm, known as a CIPHER is the function used for Encryption and Decryption of Data.
The Hill cipher Algorithm [4] is one of the Symmetric Key Algorithms which have several advantages in Data Encryption. It uses a pair of key matrices, a key on one side in Encryption and its inverse on the other side.
Images are widely used in different –different processes. Therefore, the security of image data from unauthorized uses is important.
Image encryption [5] acting an important role in the field of information hiding. Image encryption method prepares information unreadable. Hence, no hacker or eavesdropper and others, are accessible to original message or any other type of transmitted information using public networks such as Internet.

Why do we require Image Encryption?

Image encryption decryption has applications in internet communication, multimedia systems, medical imaging, telemedicine, military communication, etc. Many image content encryption algorithms have been proposed. To ensure data secure from various attacks and for the integrity of data we have to encrypt the data before it is transmitted or stored. Government, military, financial institution, hospitals and private business deals with confidential images about their patient (in Hospitals), geographical areas (in research), enemy positions (in defense), product, financial status. Majority of this information is now collected and stored on electronic computers and transmitted across network to other computer. If these confidential images about enemy positions, patient and geographical areas fall into the wrong hands, then such a breach of security could lead to declination of war, wrong treatment etc. Protecting confidential images is an ethical and legal requirement.

To safely and securely transfer images for the following uses.

*1)*Military
*2)*Health Care
*3)*Picture messaging on Cell Phones
*4)*Privacy
*5)*Government Documents etc.,

II EXISTING SYSTEM

There are traditional image encryption techniques [3] like DES, Triple- DES and IDEA.

Limitations:
Requires huge data size
Lengthy computational time
High computing power
Not suitable for practical image encryption and for online communications.

III PROPOSED SYSTEM

Role Of Matrix As Cipher In Encryption And Decryption

Residue Matrices

Cryptography uses residue matrices in which all elements are in $\mathbf{Z_n}$. All operations on residue matrices are carried out the same as for the integer matrices except that the operations are done in modular arithmetic. One interesting result is that a residue matrix has a multiplicative inverse if the determinant of the matrix has a multiplicative inverse in $\mathbf{Z_n}$. Therefore, a residue matrix has a multiplicative inverse if gcd
(det(**A**), *n*) = 1.

Congruence: Two matrices are [3] congruent modulo *n*, written as **A** ≡ **B** (mod *n*), if they have the same number of rows and columns and all related elements are congruent modulo *n*.
In other words, **A** ≡ **B** (mod *n*) if *aij*≡ *bij*(mod *n*) for all *i*'s and *j* 's.

Upper Triangular Matrix as Key Matrix.

Consider an upper triangular Matrix :

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots\ldots\ldots & a_{1n} \\ 0 & a_{22} & a_{23} & \ldots\ldots\ldots & a_{2n} \\ 0 & 0 & a_{33} & \ldots\ldots\ldots & a_{3n} \\ . & . & . & . & . \\ 0 & 0 & 0 & 0 & a_{nn} \end{pmatrix} =$$

$$\begin{pmatrix} a_1 & a_1 & a_1 & \ldots\ldots\ldots & a_1 \\ 0 & a_2 & a_2 & \ldots\ldots\ldots & a_2 \\ 0 & 0 & a_3 & \ldots\ldots\ldots & a_3 \\ . & . & . & . & . \\ 0 & 0 & 0 & 0 & a_n \end{pmatrix}$$

i.e. $a_{11} = a_{12} = a_{13} =$ --------- $= a_{1n} = a_1$

$a_{22} = a_{23} =$ ---------- $= a_{2n} = a_2$

International Journal of Advance Research in Science and Engineering
Vol. No.6, Special Issue (01), September 2017, BVCNSCS 2017
www.ijarse.com
IJARSE
ISSN 2319 - 8354

a_{33} = ---------- = a_{3n} = a_3

---

a_{nn} = a_n

Case I: If j≠i+1 , then b_{ij} =0

Case II: If j=i+1, then b_{ij} = Additive inverse(mod m) of $b_{(i+1)j}$

Then A^{-1} will be also an upper triangular matrix given by:

$$A^{-1} = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & \dots & & b_{1n} \\ 0 & b_{22} & b_{23} & b_{24} & b_{24} & \dots & & b_{2n} \\ 0 & 0 & b_{33} & b_{34} & b_{34} & \dots & & b_{3n} \\ 0 & 0 & 0 & b_{44} & b_{45} & \dots & & b_{4n} \\ 0 & 0 & 0 & 0 & b_{55} & \dots & & b_{5n} \\ 0 & 0 & 0 & 0 & 0 & \dots & & . \\ . & . & . & . & . & \dots & & . \\ . & . & . & . & . & \dots & & . \\ 0. & 0 & .0 & 0 & 0 & \dots & b_{(n-1)(n-1)} & b_{(n-1)n} \\ .0 & 0 & 0 & . & . & \dots & & b_{nn} \end{pmatrix}$$

$$\begin{pmatrix} a_{11}^{-1} & add.inv.of b_{22} & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & a_{22}^{-1} & add.inv.of b_{33} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & a_{33}^{-1} & add.inv.of b_{44} & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & a_{44}^{-1} & add.inv.of b_{55} & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{55}^{-1} & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & . & . \\ . & . & . & . & . & \dots & . & . \\ . & . & . & . & . & \dots & . & . \\ 0. & 0 & .0 & 0 & 0 & \dots & a_{(n-1)(n-1)^{-1}} & add.inv.of b_{nn} \\ .0 & 0 & 0 & . & . & \dots & 0 & a_{nn}^{-1} \end{pmatrix}$$

Where:

1) For i=j, The diagonal elements are in such a way that:

$b_{11} = a_{11}^{-1}$ (Multiplicative inverse Mod m)

$b_{22} = a_{22}^{-1}$ (Multiplicative inverse Mod m)

……

……

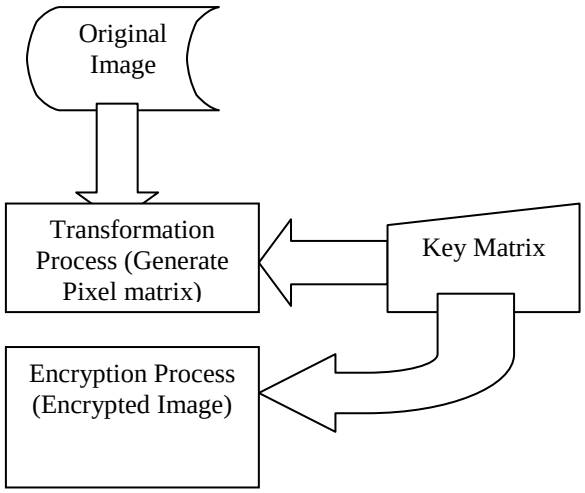
i.e.

For i=j $b_{ij} = a_{ij}^{-1}$ (Multiplicative inverse Mod m)

2) For i≠j we have two cases…

Requirements of Image Encryption

In order to get the pixels of the original image.[6]
Create a strong encryption image such that it cannot be hacked easily. Key is needed to obtain original image because it's hard to crack that encrypted form.
Faster encryption time such that encrypted image is transferred faster to the person.
Perfection in the original image we derive after decrypting it.



Original Image **Encrypted Image**

International Journal of Advance Research in Science and Engineering
Vol. No.6, Special Issue (01), September 2017, BVCNSCS 2017
www.ijarse.com
IJARSE
ISSN 2319 - 8354

Encryption

Encrypted Image **Original Image**

Decryption

Algorithm and Example of Image Encryption

Step1: **consider an Image to be encrypted.**

Write Image pixel matrix.

$$P=\begin{pmatrix} 0 & 0 & 244 & 254 & 211 \\ 0 & 230 & 222 & 255 & 242 \\ 255 & 178 & 193 & 231 & 244 \\ 0 & 154 & 196 & 255 & 242 \\ 0 & 142 & 217 & 254 & 230 \end{pmatrix}_{5*5}$$

Consider the key 2 4 3 1 5 is sent as private key.
The key is converted to matrix.

Step 2:- **Since P is 5*5 matrix consider our Key matrix to be a square matrix with it's Order 5*5(where 5= number of columns of matrix).**

$$\text{Let } A=\begin{pmatrix} 2 & 2 & 2 & 2 & 2 \\ 0 & 4 & 4 & 4 & 4 \\ 0 & 0 & 3 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

Where the key elements are the principal diagonal elements of above matrix.

Step 3: **Find the product AP i.e.**

$$AP=\begin{pmatrix} 510 & 1408 & 2144 & 2498 & 2338 \\ 1020 & 2816 & 3312 & 3980 & 3832 \\ 765 & 1422 & 1818 & 2220 & 2148 \\ 0 & 296 & 413 & 509 & 472 \\ 0 & 710 & 1085 & 1270 & 1150 \end{pmatrix}$$

This has to be converted to a matrix with elements to be in Z_{257}. I e the above matrix is equivalent to a matrix in which each element is the remainder of it's corresponding element in AP when divided by 257.

Therefore AP=

$$\begin{pmatrix} 253 & 125 & 88 & 185 & 25 \\ 249 & 246 & 228 & 125 & 234 \\ 251 & 137 & 19 & 164 & 92 \\ 0 & 39 & 156 & 252 & 215 \\ 0 & 453 & 57 & 242 & 122 \end{pmatrix}$$

I e.510=253(mod 257), 1408=125(mod 257) etc.

Step 4: **Now consider AP as pixel matrix and obtain the image related which is the encrypted Message to be sent as secret image, along with key.**

Algorithm and Example for Image Decryption

Step 1:

a) The received image has to be converted to it's pixel matrix, that should be

International Journal of Advance Research in Science and Engineering
Vol. No.6, Special Issue (01), September 2017, BVCNSCS 2017
www.ijarse.com
IJARSE
ISSN 2319 - 8354

AP mentioned before.

b) And the key should be considered as triangular matrix A mentioned before.

Step 2: Finding A^{-1}

Important Note: Instead of traditional method of finding A^{-1} , we are finding A^{-1} with the help of modular arithmetic..

$$A=\begin{pmatrix} 2 & 2 & 2 & 2 & 2 \\ 0 & 4 & 4 & 4 & 4 \\ 0 & 0 & 3 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

Then

$$\begin{pmatrix} 129 & 64 & 0 & 0 & 0 \\ 0 & 193 & 171 & 0 & 0 \\ 0 & 0 & 86 & 256 & 0 \\ 0 & 0 & 0 & 1 & 154 \\ 0 & 0 & 0 & 0 & 103 \end{pmatrix}$$

Where (1) the principal diagonal elements are Obtained first as follows:

129= multiplicative inverse of 2 is Z_{257}.

1). 2 * 129 = 258 = 1(mod 257)

4 * 193 = 772 =1(mod 257)

3 * 86 = 258 =1(mod 257)

1 * 1 = 1(mod 257)

5 * 103 = 515 =1(mod 257)

3) 64 = additive inverse of 193

i.e. 193 + 64= 257 = 0---→64=257-193.

And 171 + 86 = 257 = 0 --→ 86= 257 – 171.

256=257-1

154 = 257 – 103.

And remaining all elements are zeros.

Step 3: then multiply A^{-1} and (AP) .

$A^{-1}AP = P$ pixel matrix of original image.

4) This (p) has to be converted to a matrix with elements to be in Z_{257}. i. e, each element in the AP matrix(Original matrix) is the remainder of it's corresponding element in P when divided by 257.

Algorithms Implementation using Java Programming Language

// convert Sunset original image to pixel matrix

```
import java.awt.image.Raster;
import java.io.IOException;
import javax.imageio.ImageIO;
public class JavaImage
{

 public static void main(String ar[])
 {
  try
  {
   // the line that reads the image file
   BufferedImage image = ImageIO.read(new
File("images.jpg"));

   // work with the image here...
  Raster raster=image.getData();
int w=raster.getWidth(),h=raster.getHeight();
System.out.println ("width="+w+"\theight="+h);

  int pixels[][]=new int[w][h];
  for (int x=0;x<w;x++)
  {
    for(int y=0;y<h;y++)
    {
      pixels[x][y]=raster.getSample(x,y,0);
System.out.print(pixels[x][y]+"\t");
    }
System.out.println();
  }
}

  catch (IOException e)
  {
   // log the exception
   // re-throw if desired
  }
 }
}
```

Output:

Original image:

Converted pixel matrix:

```
C:\>java  JavaImageIOTest
width=13        height=10
156     181     210     237     244     221     179     130     128     122
175     184     206     224     236     219     174     142     130     126
197     200     207     192     209     231     188     138     130     124
212     222     220     190     186     211     190     131     128     124
223     235     242     230     192     179     181     138     129     126
223     242     254     255     224     205     204     144     145     144
227     248     255     255     248     243     221     148     164     166
233     246     255     255     249     231     196     152     153     154
225     244     253     252     229     201     171     132     129     130
215     231     237     234     215     190     159     120     114     111
196     210     219     219     207     185     150     108     99      97
171     186     202     210     208     186     143     98      94      90
151     167     189     202     206     187     139     94      89      84
```

// pixel matrix is converted to encrypted pixel matrix

```
import  java.awt.image.BufferedImage;
import java.io.File;
import java.lang.*;
public class Encryption
```

```
{
public static void main(String ar[])
{

//image in pixel matrix of order m*n

int[][] p = {{156,181,210,  237,  244,  221,  179,
130,  128,  122},

{175,  184,  206,  224,  236,  219,  174,  142,
130,  126},

{197,  200 ,  207 ,  192 ,  209 ,  231 ,  188 ,  138 ,
130 ,  124},

{212 ,  222  , 220  , 190  , 186  , 211 ,  190  , 131  ,
128  , 124},

{223 ,  235 ,  242 ,  230  , 192 ,  179 ,  181  , 138 ,
129  , 126},

{223  , 242  , 254 ,  255  , 224 ,  205 ,  204  , 144  ,
145  , 144},

{227 ,  248 ,  255 ,  255 ,  248 ,  243  , 221  , 148 ,
164 ,  166},

{233 ,  246  , 255 ,  255 ,  249 ,  231  , 196 ,  152 ,
153 ,  154},

{225 ,  244 ,  253  , 252  , 229 ,  201 ,  171 ,  132 ,
129  , 130},

{215 ,  231 ,  237 ,  234  , 215 ,  190 ,  159 ,  120 ,
114 ,  111},

{196  , 210 ,  219 ,  219 ,  207  , 185 ,  150 ,  108 ,
99  , 97},

{171 ,  186  , 202 ,  210 ,  208  , 186  , 143 ,  98 ,
94 ,  90},

{151 ,  167 ,  189  , 202,  206 ,  187 ,  139 ,  94 ,
89 ,  84}};



int[][] A={{3,3,3,3,3,3,3,3,3,3,3,3,3},
{0,4,4,4,4,4,4,4,4,4,4,4,4},{0,0,5,5,5,5,5,5,5,5,5,5,5},{0,0,0,
8,8,8,8,8,8,8,8,8,8},{0,0,0,0,9

,9,9,9,9,9,9,9,9},{0,0,0,0,0,2,2,2,2,2,2,2,2},{0,0,0,0,0,0,6,6,
6,6,6,6,6},{0,0,0,0,0,0,0,7,7,7,7,7,7},{0,0,0,0,0,0,0,0,0,10,1

0,10,10},{0,0,0,0,0,0,0,0,0,0,11,11,11,11},{0,0,0,0,0,0,0,0,0,0
,12,12,12},{0,0,0,0,0,0,0,0,0,0,0,13,13},{0,0,0,0,0,0,0,0,0,0,
0,0,14}};
//key matrix should be m*m matrix and triangular matirx
int c[][]=new int[50][50];
int T[][]=new int[50][50];
for(int i=0;i<13;i++)
{
for(int j=0;j<10;j++)
{
 c[i][j]=0;
for(int k=0;k<13;k++)
{
c[i][j]=c[i][j]+A[i][k]*p[k][j];
}
}
}

System.out.println("display product in matrix format\n");
for(int i=0;i<13;i++)
{
for(int j=0;j<13;j++)
{
System.out.print(c[i][j]+"\t");
}
System.out.println();
}

// Reduce elements in T to mod 257
for(int i=0;i<13;i++)
{
for(int j=0;j<13;j++)
{
T[i][j]=c[i][j]%257;
}
}

//display encrypted matrix

System.out.println("display encrypted matrix\n");

for(int i=0;i<13;i++)
{
for(int j=0;j<13;j++)
{
System.out.print(T[i][j]+"\t");
}
System.out.println();
}
}
}
```

Output:

//encrypted pixel matrix

```
display encrypted matrix
102     164     109     127     78      100     203     142     13      168
0       0       0
26      180     162     78      156     106     240     12      105     250
0       0       0
57      76      72      134     43      194     201     76      188     68
0       0       0
160     115     104     169     93      210     154     97      186     42
0       0       0
71      91      193     247     69      72      198     183     21      152
0       0       0
198     207     130     166     231     172     196     193     175     153
0       0       0
27      197     151     253     120     57      135     229     169     109
0       0       0
113     250     233     95      203     36      24      45      120     36
0       0       0
134     230     246     169     136     27      256     88      105     222
0       0       0
96      253     65      6       201     4       76      251     244     90
0       0       0
48      74      124     119     256     14      44      2       43      168
0       0       0
74      220     200     216     242     223     68      183     66      206
0       0       0
58      25      76      1       57      48      147     31      218     148
0       0       0
```

// Convert pixel matrix to Image

```
import java.awt.image.BufferedImage;
import java.io.File;
import java.awt.image.Raster;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.awt.image.MemoryImageSource;
import java.awt.Toolkit;
import java.awt.Color;
import javax.swing.*;
import java.awt.image.*;

public class My
{
public static void main(String ar[])
{
int x,y,value=0;

 int[][] a = {{156,181,210,    237,    244,    221,

 179,    130,    128,    122},

{175,    184,    206,    224,    236,    219,

174,    142,    130,    126},

{197,    200 ,    207 ,    192 ,    209 ,    231 ,    188

 ,    138    ,  130  ,    124},

{212  ,    222    ,  220    ,  190    ,  186    ,  211  ,

190    ,  131    ,  128    , 124},

{223  ,    235 ,    242  ,    230    , 192  ,    179  ,
  181    , 138  ,    129    ,  126},

{223    , 242    , 254  ,    255    ,  224 ,    205  ,    204

  , 144    ,  145    ,  144},

{227 ,    248  ,    255  ,    255 ,    248 ,    243    , 221

 ,  148  ,    164  ,    166},

{233 ,    246    ,  255  ,    255  ,    249  ,    231    ,  196

 ,    152  ,    153  ,    154},

{225 ,    244 ,    253    , 252    ,  229  ,    201  ,    171

 ,  132 ,    129    ,  130},

{215  ,    231 ,    237 ,    234    ,  215  ,    190 ,    159

  ,    120  ,    114  ,    111},

{196    ,  210  ,    219  ,    219  ,    207    ,  185  ,

150  ,    108  ,  99    ,  97},

{171  ,    186    ,  202 ,    210  ,    208    ,  186    ,  143

 ,  98  ,    94 ,    90},

{151 ,    167 ,    189    ,  202,    206  ,    187 ,

139 ,    94  ,    89 ,    84}};

BufferedImage image = new BufferedImage(a.length,

a[0].length, BufferedImage.TYPE_INT_RGB);
   for (x = 0; x < a.length; x++) {
       for (y = 0; y < a[x].length; y++) {
          value =a[x][y];
          Color nc=new Color(value,value,value);
          image.setRGB(x,y,nc.getRGB());

       }

    }

JFrame frame = new JFrame();
  ImageIcon icon = new ImageIcon(image);
  JLabel label = new JLabel(icon);
  frame.add(label);
  frame.setDefaultCloseOperation
        (JFrame.EXIT_ON_CLOSE);
  frame.pack();
  frame.setVisible(true);

}
}
```

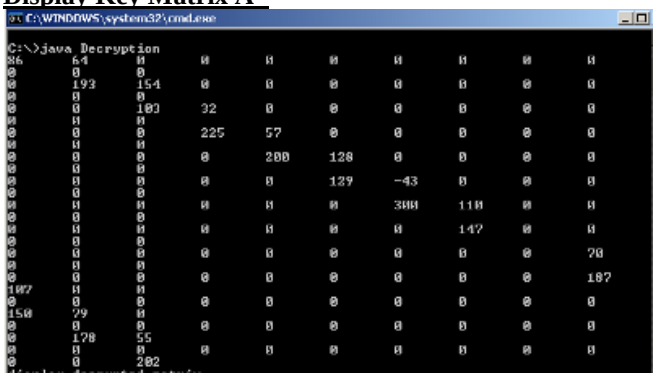
Output:

// Image decryption program

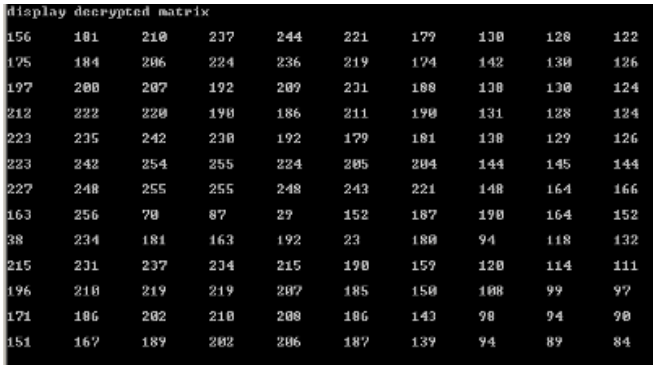
```
// encrypted pixel matrix is converted to decrypted pixel matrix
import  java.awt.image.BufferedImage;
import java.io.File;
import java.lang.*;
public class Decryption
{
public static void main(String ar[])
{
//encrpted pixel matrix
int a[][]={{102,164,109,127,78,100,203,142,13,168,0,0,0},
        {26,180,162,78,156,106,240,12,105,250,0,0,0},
        {57,76,72,134,43,194,201,76,188,68,0,0,0},
        {160,115,104,169,93,210,154,97,186,42,0,0,0},
        {71,91,193,247,69,72,198,183,21,152,0,0,0},
        {198,207,130,166,231,172,196,193,175,153,0,0,0},
        {27,197,151,253,120,57,135,229,169,109,0,0,0},
        {113,250,233,95,203,36,24,45,120,36,0,0,0},
        {134,230,246,169,136,27,256,88,105,222,0,0,0},
        {96,253,65,6,201,4,76,251,244,90,0,0,0},
        {48,74,124,119,256,14,44,2,43,168,0,0,0},
        {74,220,200,216,242,223,68,183,66,206,0,0,0},
        {58,25,76,1,57,48,147,31,218,148,0,0,0}};

int
A[][]={{3,3,3,3,3,3,3,3,3,3,3,3,3},{0,4,4,4,4,4,4,4,4,4,4,4,4}
,{0,0,5,5,5,5,5,5,5,5,5,5,5},{0,0,0,8,8,8,8,8,8,8,8,8,8},{0,0,0
,0,9,9,9,9,9,9,9,9,9},{0,0,0,0,0,2,2,2,2,2,2,2,2},{0,0,0,0,0,0,
6,6,6,6,6,6,6},{0,0,0,0,0,0,0,7,7,7,7,7,7},{0,0,0,0,0,0,0,0,0,1
0,10,10,10},{0,0,0,0,0,0,0,0,0,0,11,11,11,11},{0,0,0,0,0,0,0,0,
0,0,12,12,12},{0,0,0,0,0,0,0,0,0,0,0,13,13},{0,0,0,0,0,0,0,0,
0,0,0, 0,14}};
//key matrix should be m*m matrix and triangular matirx
int c[][]=new int[50][50];
int T[][]=new int[50][50];
for(int i=0;i<13;i++)
{
for(int j=0;j<13;j++)
{
if(i==j)
{
for(int k=0;k<=300;k++)
{
int m=A[i][j]*k;
if(m%257==1)
c[i][j]=k;
}
}
if(i!=j)
{
if(j==(i+1))
{
for(int k=0;k<=300;k++)
{
int m=A[i+1][j]*k;
int r=m%257;
if(r==1)
c[i][j]=257-k;
}
}
else if(j!=(i+1))
c[i][j]=0;
}
}
}

// A-1 of key matrix
for(int i=0;i<13;i++)
{
for(int j=0;j<13;j++)
{
System.out.print(c[i][j]+"\t");
}
System.out.println();
}

// A-1 a(encrypted matrix multiplication)
for(int i=0;i<13;i++)
{
for(int j=0;j<10;j++)
{
T[i][j]=0;
for(int l=0;l<13;l++)
{
T[i][j]=T[i][j]+(c[i][l]*a[l][j]);
}
}
}

int TT[][]=new int[50][50];
// Reduce elements in T to mod 257
for(int i=0;i<13;i++)
{
for(int j=0;j<10;j++)
{
TT[i][j]=T[i][j]%257;
}
}

//display decrypted matrix(Original Matrix)

System.out.println("display decrypted matrix\n");

for(int i=0;i<13;i++)
{
for(int j=0;j<10;j++)
{
System.out.print(TT[i][j]+"\t");
}
System.out.println();
}

}
}
```

Output:

Display Key Matrix A^{-1}



Display Decrypted pixel Matrix



Output of Decrypted pixel matrix into Original Matrix

IV CONCLUSION

Thus, the paper entitled "**Image encryption and decryption**" is successfully completed.

A complex paper involving the conversion of image into pixel matrix form, using a mathematical concept to encrypt and decrypt. It was instrumental in giving us a thorough understanding of how the concepts of JAVA and Linear Algebra together can actually be implemented in the real world. We have gained valuable skills including mathematics and programming language and learning how to form and manipulate images.

V REFERENCES

[1] William Stallings "Network Security Essentials (Applications and Standards)", Pearson Education, 2004.

[2] Bibhudendra, A., K.P. Saroj, K.P. Sarat and P. Ganapati, 2009. Image encryption using advanced hill cipher algorithm. Int. J. Recent Trends Eng., 1:663667.http://www.ijrte.academypublisher.com/vol01/no01/ijrte0101663667.pdf

[3] Eisenberg, M., 1998. Hill ciphers and modular linear algebra. Mimeographed notes. University of Massachusetts. http://www.apprendre-enligne. net/crypto/hill/Hillciph.pdf

[4] Ismail, I.A., M. Amin and H. Diab, 2006. How to repair the hill cipher. J. Zhejiang Univ. Sci. A., 7: 2022- 2030. DOI: 10.1631/jzus.2006.A2022

[5] Shahrokh Saeednia, "How to Make Hill cipher Secure," Cryp- tologia 24:4, pp. 353-360, Oct 2000.

[6] Ismail I A, Amin Mohammed, Diab Hossam, "How toRepair the Hill Cipher," Journal of Zhejiang UniversityScience, 7(12), pp. 2022-2030, 2006.