A Proposed Enhanced Transposition Cipher Algorithm based on Rubik's Cube Transformations

Frimpong Twum
Kwame Nkrumah University of Science and Technology
Department of Computer Science
Kumasi - Ghana

J. B. Hayfron Acquah
Kwame Nkrumah University of Science and Technology
Department of Computer Science
Kumasi - Ghana

Morgan-Darko William
Kwame Nkrumah University of Science and Technology
Department of Computer Science
Kumasi - Ghana

ABSTRACT
It is unlikely any system can completely prevent unauthorized interception to transmission signal hence a more practical method that is traditionally employed for achieving privacy is to alter the message so only an authorized receiver can understand it. The method used to do this is termed encryption and decryption of information. By encrypting the massage before it is transmitted the message is unintelligible to everyone that receives it except the rightful recipient. The encryption/decryption methods process a message using an algorithm and a key. Transposition Cipher which shuffles characters around instead of substituting them with other characters is one way achieving privacy of data thereby assuring data owners of their data confidentiality. In this paper the Rubik's cube a modified Rubik's cube puzzle is employed at levels higher than $3 \times 3 \times 3$ as a transposition cipher to encrypt data. Although no system is hundred percent secured the proposed algorithm sufficiently encrypt data with sufficient rotations of the cube.

General Terms
Encryption, Decryption, Cipher, Algorithm, Security, Privacy.

Keywords
Encryption, Decryption, Transpostion Cipher, Algorithm, Rubik's Cube, Character Level Encrption.

1. INTRODUCTION
Rubik's cube puzzle has proven to be quite difficult to solve at a $3 \times 3 \times 3$ level (Mitchell, 1992). This paper explores the use of a modified Rubik's cube at levels higher than $3 \times 3 \times 3$ as a transposition cipher. A transposition cipher is one which rearranges the order of the letters in the ciphertext (encoded text), according to some predetermined method, without making any substitutions (nrich, 2018). The more sophisticated the scrambling mechanism, the stronger the encryption. Most transposition systems use a geometric process. Plaintext is written into a geometric figure, most commonly a rectangle or square, and extracted from the geometric figure by a different path than the way it was entered. When the geometric figure is a rectangle or square, and the plaintext is entered by rows and extracted by columns, it is called columnar transposition. When some route other than rows and columns is used, it is called route transposition (UMich, 2018).

2. LITERATURE REVIEW
For a perceptive operational critical data such as military or business financial data to be transmitted over an un-trusted public network such as the Internet, a system ought to be able to guarantee users of their privacy. Privacy also called confidentiality or secrecy has to do with making sure nosy people cannot read and make sense of a message intended for another recipient. Thus, the transmitted message should make sense to only the intended receiver. The method used to do this is termed encryption and decryption of information. By encrypting the massage before it is transmitted the message is unintelligible to everyone that receives it except the rightful recipient. Thus, encryption means the sender of a message transforms the original message (called plaintext) to another unintelligible form (called ciphertext) and send the transformed unintelligible message out over the network such as the Internet to the intended receiver. On receiving the ciphertext the rightful receiver apply a reverse process of the encryption method used to re-transform the ciphertext back to its original form (the plaintext) in a process called decryption (Maiwald, 2003). The encryption/decryption methods process a message using an algorithm and a key. Encryption and Decryption methods fall into two broad categories as follows: Conventional methods (aka. Secret key methods or Symmetric methods) and Public key methods (aka. Asymmetric methods) (Stallings, 2003; Stallings, 2011; Kessler, 2017). Conventional encryption algorithms are broadly grouped into two, ***character-level*** ciphers or ***bit-level*** ciphers. A cipher is a character-for-character or bit-for-bit transformation without regard to the linguistic structure of the message (Tanenbaum, 2003). Encryption under character-level method is achieved via two techniques, *Substitution ciphers* or *Transposition ciphers*. With Transposition Ciphers, the characters in the plaintext are shuffled around instead of been substituted with other characters as in the case of substitution ciphers. Like substitution cipher, **transposition cipher** is another example of character-level encryption however the plaintext characters keep their original form while their positions are altered to generate the ciphertext. The technique arranges the plaintext in a 2-dimensional table (Tanenbaum, 2003). For example, the ciphertext

'**PETHELTLDTSPLOEOOTEIRANUETGXSOCVAAX**' is transmitted for the plaintext '**Please do not touch Steve pet Alligator**'. The ciphertext is obtained through entering the characters of the plaintext into a table in row order where the table size determined by the number of columns is the encryption/decryption key and must be known to both the sender and the receiver (five in this example). The ciphertext is recorded vertically down the table from the first column while the plaintext is entered horizontally into the table - Figure 2.1 and Figure 2.2.

P	L	E	A	S
E	D	O	N	O
T	T	O	U	C
H	S	T	E	V
E	P	E	T	A
L	L	I	G	A
T	O	R	X	X

Figure 2.1: Transposition Cipher

P	O	C	P	G
L	N	H	E	A
E	O	S	T	T
A	T	T	A	O
S	T	E	L	R
E	O	V	L	X
D	U	V	I	X

Figure 2.2: Route Cipher

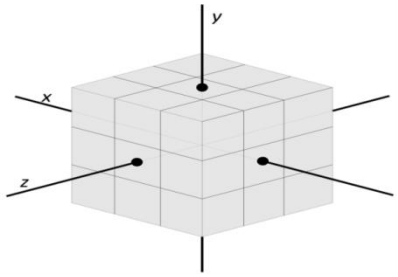
A transposition cipher is made more complex by specifying the key to determine the order of recording the columns for the ciphertext. For example the keyword 'KWAME' could be used to transform the plaintext above as entered into figure 2.1 to this ciphertext, '**EOOTEIRSOCVAAXPETHELTANUETGXLDTSPLO**'. The position of a character in the key and the order it appear in the English alphabet determines the order in which the columns are recorded to obtain the ciphertext. To decrypt, the key is used by the receiver to determine the number of table columns whiles the number of rows is determined by a count of the number of characters in the received ciphertext divided by the number of characters in the key. For instance, in the example above 35 ciphertext characters / 5 key characters = 7 rows. The ciphertext are then entered into the table following the order they appear in the English alphabet with their position in the key used in determining the column they are entered into. For example character 'A' in the key is considered first and as it is at position 3 in the key, the first seven characters of the ciphertext are entered into column 3. Likewise 'E' at position 5 of the key is treated next and hence the next 7 ciphertext characters are also entered into column 5 and so on. The plaintext is finally obtained by reading the characters from the rows. Using a key this way for a transposition cipher although makes it much harder for a snooper to decrypt, the approach is not that secure as the substitution cipher because the character frequencies are maintained and hence a more experienced snooper can decode through a trial and error attack or a frequency analysis attack although could be much difficult or a brute-force attack (Forouzan, 2001). Some other well-known examples of transposition cipher include ***Route Cipher*** which when used to encrypt the example message as entered into figure 2.2 would result with the ciphertext **XXROTAGPCOPLEASEDUVILLATEHNOTTOVETS** assuming the sender and the receiver agree a key start point to be bottom right while routing up inward in anti-clockwise direction**.** This ciphertext can be decoded easily by choosing a route around the grid. Thus, the ciphertext is decrypted by entering the characters back into the grid using the key (comprising of the table size and the key start point). The plaintext is obtained by recording the text from the columns beginning from the first column. ***Rail Fence Cipher*** is another example of transposition cipher.

3. METHODOLOGY

The experimental design research approach is adopted by following three steps to encrypt a message to be transmitted using the rotation of the Rubik's Cube as a transposition cipher as follows:

3.1 Initialization of the cube with text

The first activity to perform in using Rubik's Cube as a transposition model is to prepare the six faces of the cube to receive the plaintext.



The size of the square grid on the faces of the cube needs to be computed so that it can accommodate all the data, with minimum padding. This can be achieved by following the steps in *Listing 3.1*.

1. Take the integer ceiling from the division of the length of the data by 6.
2. Take the integer ceiling of the square root of the result from (1).
3. Define 6 two-dimensional arrays to function as the faces of cube using the result from (2) as both dimensions of the array.

Listing 3.1 *– Setting up a minimum-padding cube*

3.2 Generation of Rotation Sequence from the Key

The key for encrypting the data has to be transformed into a sequence of rotations in order for the encryption to be performed. Two things are needed to perform one rotation – the plane in which to perform the rotation and the index of the row or column on which the rotation is to be done. Any algorithm which can translate the key into a sequence of rotations is useful at this stage of the transposition process.

3.3 Rotation of the Cube

To mimic the rotation of the cube, strips of data needs to be copied from one face onto another in a set pattern. The pattern in *Listing 3.2* is one of several patterns that can be used to implement the rotation activity. This pattern visualizes faces 1, 2, 3, 4, 5 and 6 as the front, right side, back, left side, top and bottom faces respectively. n is the number of row (and columns) of each face of the cube.

To rotate a strip in row i clockwise in the y-axis

1. Copy the data in row i of face 1 to a temporary location.
2. Replace the data in row i of face 1 with the data in row i of face 2.
3. Replace the data in row i of face 2 with the data in row i of face 3.
4. Replace the data in row i of face 3 with the data in row i of face 4.

5. Replace the data in row i of face 4 with the data being held in the temporary location.

To rotate a strip in column i clockwise in the x-axis

1. Copy the data in column i of face 1 to a temporary location.
2. Replace the data in column i of face 1 with the data from column i of face 6.
3. Replace the data in column i of face 6 with the data from column $(n-i)$ of face 3, in reverse order.
4. Replace the data in column $(n-i)$ of face 3 with the data from column i of face 5, in reverse order.
5. Replace the data in column i of face 5 with the data being held in the temporary location.

To rotate a strip in column i clockwise in the z-axis

1. Copy the data in column i of face 2 to a temporary location.
2. Replace the data in column i of face 2 with the data from row i of face 6, in reverse order.
3. Replace the data in row i of face 6 with the data from column $(n-i)$ of face 4.
4. Replace the data in column $(n-i)$ of face 4 with the data from row $(n-i)$ of face 5, in reverse order.
5. Replace the data in row $(n-i)$ of face 5 with the data being held in the temporary location.

Listing 3.2 *– Clockwise rotations in the three axes of the cube*

Anticlockwise rotations in the three axes can be achieved by reversing the direction of copying of the clockwise rotations shown in *Listing 3.2*.

4. IMPLEMNTATION

4.1 Initialization of the cube with text

Following the algorithm of listing 3.1, the cube is initialized using the size of the data to be transposed as follows: Assuming the message to be transported (the plaintext) is:

[**As a second layer of security, the content of the file**]

1. Number of Characters in plaintext is 54. Hence following listing 3.1 take 54 ÷ 6 = 9
2. Take √9 = 3
3. Create array with dimensions = **6 X 3 X 3** (i.e.) **array [6][3][3]**

In order to achieve minimum-padding, some computations need to be performed to minimize the size of the cube while still being able to accommodate all the data. The function in l*isting 4.1* shows one way of achieving the minimum-padding cube.

```
function initializeCube (data[ ])
    dataLength=data→length
    initSquare=ceil(dataLength/6)
    dimension=ceil(sqrt(initSquare))
    cube[ ]=new array[6][dimension][dimension]
    return cube[ ]
end function
```

Listing 4.1 *– function to initialize the Rubik's Cube*

After the cube has been created the plaintext are copied onto the faces of the cube sequentially as shown in figure 4.1 from the first face to the sixth. The remaining cells on the face of the cube are padded with null character or zero.

4.2 Generation of Rotation Sequence from the Key

The key for encrypting the data is transformed into a sequence of rotations in order for the encryption to be performed. Two things are needed to perform one rotation – the plane in which to perform the rotation and the index of the row or column on which the rotation is to be done. Any algorithm which can translate the key into a sequence of rotations is useful at this stage of the transposition process. The algorithm used is as below:

1. Take the Key for example "**hippopotamus**"
2. Take the **SHA1** of the key = **a1219e634d04b405d90f13505c4d36578dc97241**
3. Take the ASCII value (char) of each character in the **SHA1** representation of the key

a) Determine the plane (x, y, or z) for the rotation = char % 3

b) Determine the index for the rotation $(\boldsymbol{char^4 + char^3 + char^2})$ % $\boldsymbol{cube\ size}(which\ is\ 3\ for\ above\ example)$

4. Save the result from 3(a) as the plane for the rotation and the result from 3(b) as the index on the plane at which the rotation should be done.

4.3 Rotation of the Cube

Figure 4.2 illustrates an example of rotation sequence of the cube using the key while copying data from one face of the cube to another using the rotation pattern giving in listing 3.2.

The rotation of the cube requires three arguments – the plane in which the rotation is to be done, the direction of the rotation and the index of the strip that is to be rotated. The pseudocode in l*isting 4.2* shows an example implementation of the rotation function.

```
function rotate (plane, direction, index)
 if (direction = "clockwise")
 if (plane = "Y") temp = cube.fetchRow(1, index)
        cube.setRow(1, index, cube.fetchRow(2, index))
        cube.setRow(2, index, cube.fetchRow(3, index))
        cube.setRow(3, index, cube.fetchRow(4, index))
                cube.setRow(4, index, temp)
else if (plane = "X")
temp = cube.fetchColumn(1, index)
 cube.setColumn(1, index, cube.fetchColumn(6, index))
cube.setColumnInReverse(6, index, cube.fetchColumn(3, n - index))
cubesetColumnInReverse(3, n - index, cube.fetchColumn(5, index))
cube.setColumn(5, index, temp)
else if (plane = "Z")
temp = cube.fetchColumn(2, index)
```

```
cube.setColumnInReverse(2, index, cube.fetchRow(6, index))
cube.setRow(6, index, cube.fetchColumn(4, n - index))
cube.setColumnInReverse(4, n - index, cube.fetchRow(5, n - index))
cube.setRow(5, n - index, temp)
end if
else if (direction = "anti-clockwise")
if (plane = "Y")
temp = cube.fetchRow(1, index)
cube.setRow(1, index, cube.fetchRow(4, index))
cube.setRow(4, index, cube.fetchRow(3, index))
cube.setRow(3, index, cube.fetchRow(2, index))
cube.setRow(2, index, temp)
else if (plane = "X")
temp = cube.fetchColumn(1, index)
cube.setColumn(1, index, cube.fetchColumn(5, index))
cube.setColumnInReverse(5, index, cube.fetchColumn(3, n - index))
cube.setColumnInReverse(3, n - index, cube.fetchColumn(6, index))
cube.setColumn(6, index, temp)
else if (plane = "Z")
temp = cube.fetchColumn(2, index)
cube.setColumnInReverse(2, index, cube.fetchRow(6, index))
cube.setRow(6, index, cube.fetchColumn(4, n - index))
cube.setColumnInReverse(4, n - index, cube.fetchRow(5, n - index))
cube.setRow(5, n - index, temp)
end if
end if
end function
```

***Listing 4.2** – Implementation of the rotation function in pseudocode*

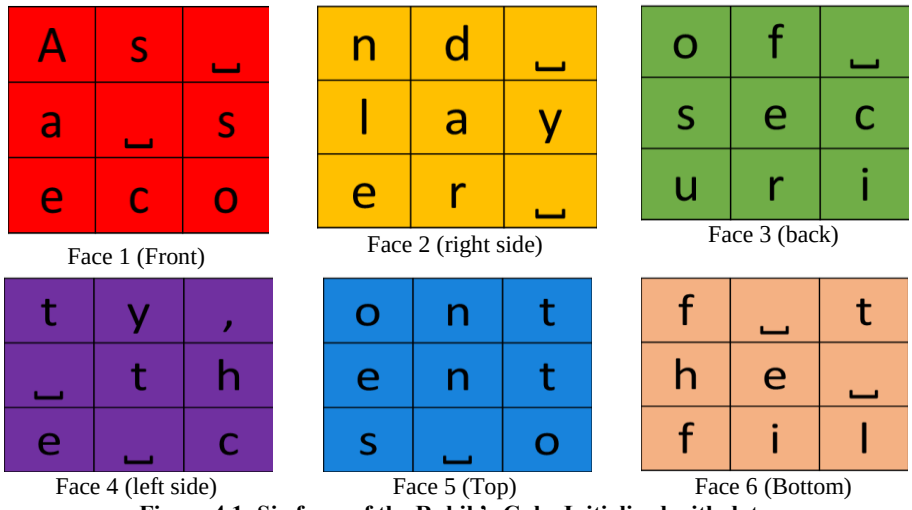
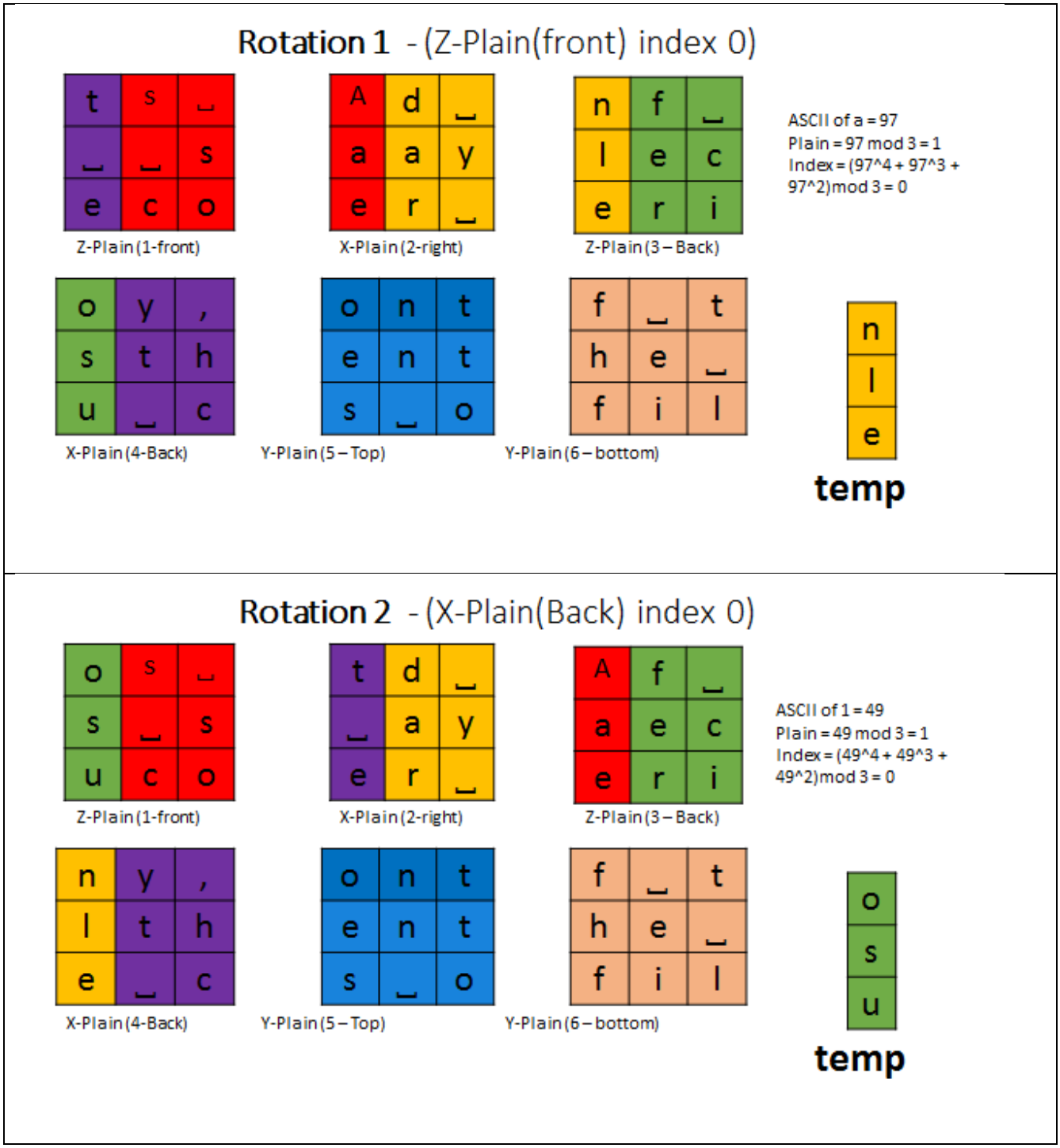
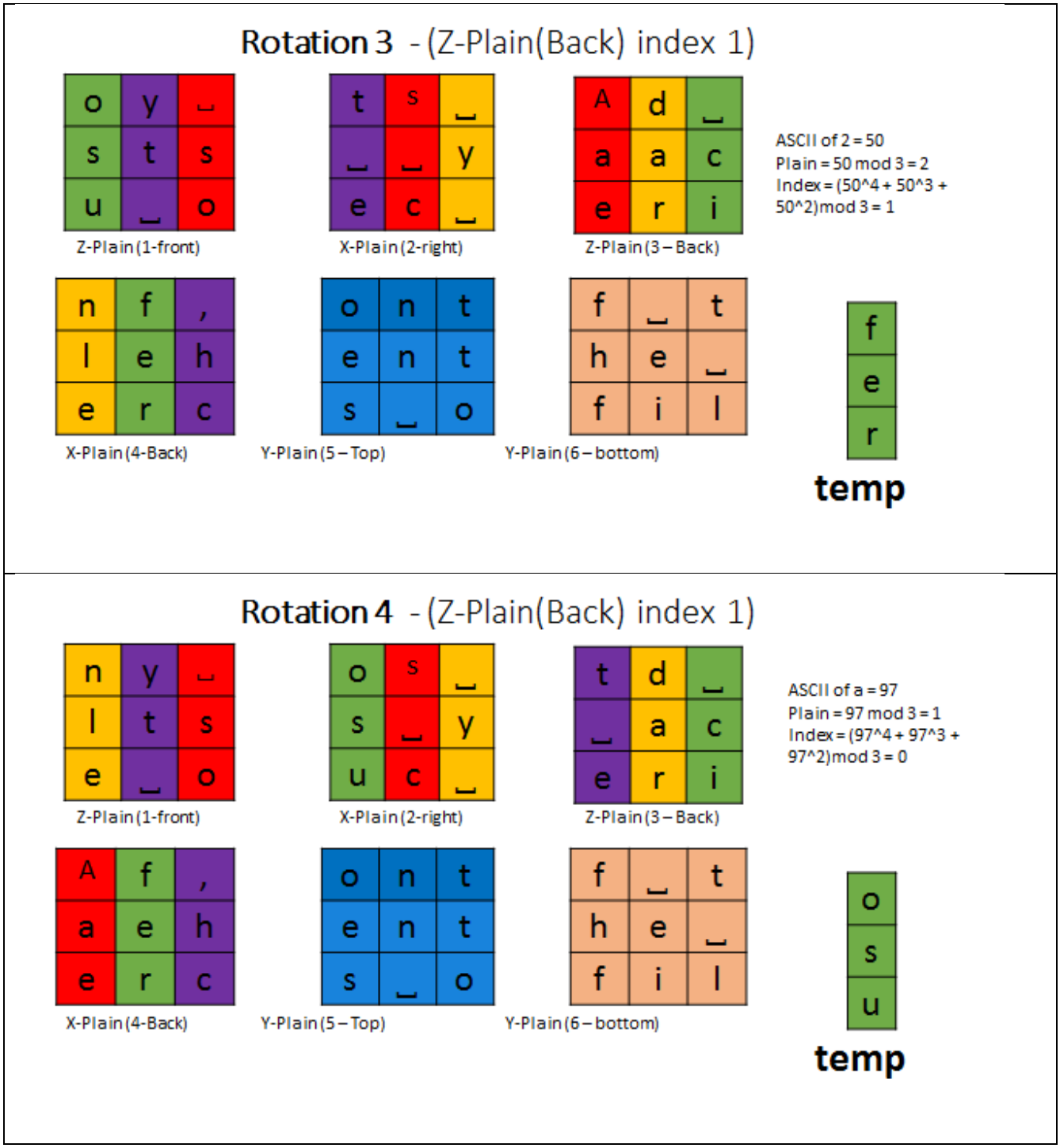


Figure 4.1: Six faces of the Rubik's Cube Initialized with data


Rotation 1 - (Z-Plain(front) index 0)
t s ␣
␣ ␣ s
e c o
Z-Plain (1-front)
A d ␣
a a y
e r ␣
X-Plain (2-right)
n f ␣
l e c
e r i
Z-Plain (3 – Back)
ASCII of a = 97
Plain = 97 mod 3 = 1
Index = (97^4 + 97^3 + 97^2)mod 3 = 0
o y ,
s t h
u ␣ c
X-Plain (4-Back)
o n t
e n t
s ␣ o
Y-Plain (5 – Top)
f ␣ t
h e ␣
f i l
Y-Plain (6 – bottom)
n
l
e
temp
Rotation 2 - (X-Plain(Back) index 0)
o s ␣
s ␣ s
u c o
Z-Plain (1-front)
t d ␣
␣ a y
e r ␣
X-Plain (2-right)
A f ␣
a e c
e r i
Z-Plain (3 – Back)
ASCII of 1 = 49
Plain = 49 mod 3 = 1
Index = (49^4 + 49^3 + 49^2)mod 3 = 0
n y ,
l t h
e ␣ c
X-Plain (4-Back)
o n t
e n t
s ␣ o
Y-Plain (5 – Top)
f ␣ t
h e ␣
f i l
Y-Plain (6 – bottom)
o
s
u
temp


Rotation 3 - (Z-Plain(Back) index 1)
o y ␣
s t s
u ␣ o
Z-Plain (1-front)
t s ␣
␣ ␣ y
e c ␣
X-Plain (2-right)
A d ␣
a a c
e r i
Z-Plain (3 – Back)
ASCII of 2 = 50
Plain = 50 mod 3 = 2
Index = (50^4 + 50^3 + 50^2)mod 3 = 1
n f ,
l e h
e r c
X-Plain (4-Back)
o n t
e n t
s ␣ o
Y-Plain (5 – Top)
f ␣ t
h e ␣
f i l
Y-Plain (6 – bottom)
f
e
r
temp
Rotation 4 - (Z-Plain(Back) index 1)
n y ␣
l t s
e ␣ o
Z-Plain (1-front)
o s ␣
s ␣ y
u c ␣
X-Plain (2-right)
t d ␣
␣ a c
e r i
Z-Plain (3 – Back)
ASCII of a = 97
Plain = 97 mod 3 = 1
Index = (97^4 + 97^3 + 97^2)mod 3 = 0
A f ,
a e h
e r c
X-Plain (4-Back)
o n t
e n t
s ␣ o
Y-Plain (5 – Top)
f ␣ t
h e ␣
f i l
Y-Plain (6 – bottom)
o
s
u
temp

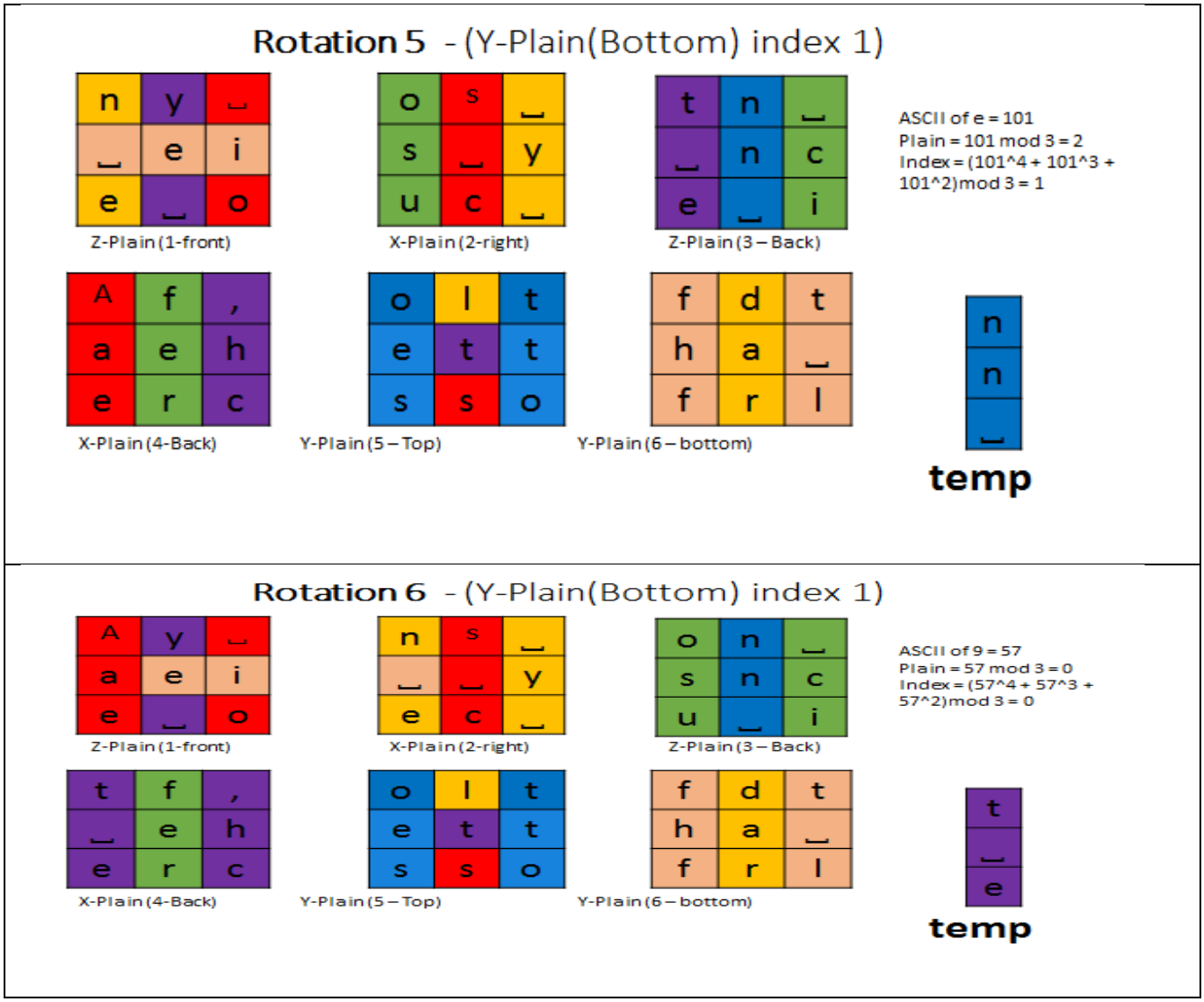


Figure 4.2: Cube rotation pattern with data

4.4 JAVA Implementation of encrypting a file content using the proposed transposition cipher algorithm based on Rubik's Cube

4.4.1 Preparation of Data for Encryption

The system writes the file's byte data onto the face of a virtual customized Rubik's Cube and uses the custom algorithm given in Listing 4.2 to create a sequence of rotations to obfuscate the data. Listing 4.3 shows a snippet of code from the method that creates the cube and writes the file data onto its faces.

```
Cube(String fileName){
   byte [] data = new byte[0];
   try {
      data = FileUtils.readFileToByteArray(new
File(fileName));
   } catch (IOException e) {
      System.err.println("Could not read file");
      e.printStackTrace();
   }
   int ceiling = (int) Math.ceil(data.length / 6.0);
   size = (int) Math.ceil(Math.sqrt(ceiling));
   int square = size * size;
   int totalNumberOfCells = 6 * square;
   data = Arrays.copyOf(data, totalNumberOfCells);
   one = new Face(size, Arrays.copyOfRange(data, 0,
square));
   two = new Face(size, Arrays.copyOfRange(data, square,
2*square));
   three = new Face(size, Arrays.copyOfRange(data, 2*square,
3*square));
   four = new Face(size, Arrays.copyOfRange(data, 3*square,
4*square));
   five = new Face(size, Arrays.copyOfRange(data, 4*square,
5*square));
   six = new Face(size, Arrays.copyOfRange(data, 5*square,
6*square));
}
```

*Listing **Error! No text of specified style in document.**.3 - Cube constructor code in Java*

4.4.2 Generation of the Rotation Sequence

Listing 4.4 shows a snippet of code from the program which implements the generation of a rotation sequence.

```
private int [][] keyToSequence(String key, int size){
   char [] list;
   list = Hash.getHash(key, "SHA1").toCharArray();
   int [][] sequence = new int[list.length][2];
   int i = 0,    plane = 0,   index = 1;
   for (char a : list){
      sequence[i][plane] = a % 3;
      sequence[i][index] = (a * a * a * a + a * a * a + a * a) %
size;
      i++;    }    return sequence;
}
```

*Listing **Error! No text of specified style in document.**.4 - Method that converts the key into a rotation sequence*

4.4.3 Encryption Function

During an encryption, the rotation sequence is followed forwards and each rotation is carried out in the clockwise direction. Listing 4.5 shows the method which performed the encryption.

```
public void encrypt(String key, String fileName){
   Cube rubik = new Cube(fileName);
   int sequence [][] = keyToSequence(key, rubik.getSize());

   for (int [] rotation : sequence)
      rubik.rotate(rotation[0], rotation[1]);
}
```

*Listing **Error! No text of specified style in document.**.5 - Method to encrypt a file using the Rubik's cube*

4.4.4 Decryption Function

During a decryption, the rotation sequence is read from the last to the first and each rotation is carried out in the counter-clockwise direction. This undoes the clockwise rotations done during the encryption. Listing 4.6 shows the method which performed the decryption.

```
public void decrypt(String key, String fileName){
   Cube rubik = new Cube(fileName);
   int sequence [][] = keyToSequence(key, rubik.getSize());

   for (int i = sequence.length - 1; i >= 0; i--)
      rubik.reverse(sequence[i][0], sequence[i][1]);
}
```

*Listing **Error! No text of specified style in document.**.6 - Method to decrypt a file using the Rubik's cube*

5. RESULTS

Following the java implementation of the algorithm in Section 4.0, the content of a file was successfully encrypted using the proposed transposition cipher algorithm based on the Rubik's Cube transformation and the results obtained is as shown in figure 5.1 and figure 5.2.

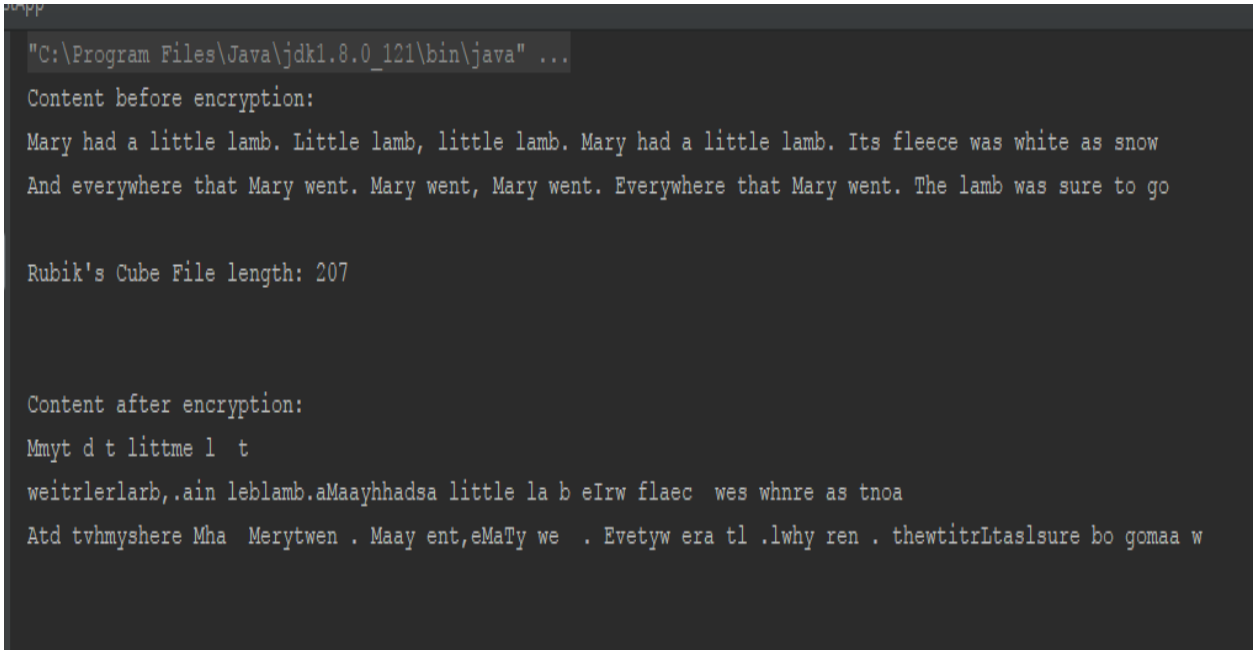


Figure Error! No text of specified style in document.**.1 - Result from encrypting file content**

```
Before Encryption
================================================================================
================================================
This is another easy stage where you shouldn't memorize any algorithm
just follow your instincts. If you have difficulties solving the white
corners,here's an easy trick you can always apply, you just have to
memorize a short algorithm and repeat it until the piece is solved
Bring the corner below the spot where it belongs (Front-Right-Down
position highlighted with grey) and repeat the algorithm above until the
white corner pops into its place oriented correctly. This algorithm sends
the piece back and forth between the spots marked with dark, always
changing the orientation.Play the animation for an example where the
sequence is repeated five times. Watch the affected white corner going to
the top then back to the bottom in each step, changing its orientation.
The sixth would bring the cube back to its original position

After Encryption
================================================================================
================================================
Thipeis atot.er easy mtaee ahere eouflwgys ap lysmorize amy  lg
rithmiju t oollowryoor ynstinits  I  you savcrkie wito dos nolvin  tte
hite  ornersg hrre s anwea y trick gouican  onho (r ont, yot just have ao
cemorize   s ort algohitem and reFeah it e dt fitnltiece as  olved
Br ng thepcgrnbe belon the epot weere id brecfli. T'isa-Rfght-Downvpo
itionphihhl ghtedqwinh crey) andareaeat t e aunoil tt  pnve until
tht wiite cirnh sopslinto itshplice oriennedccorr shatldpft talgoeitrm
sends rheapiece baek tnd forthhbeuween thewsputs mlgoi thywabgark,eal ays
chaneino theeor entatnone
Playstho aeimatiyn aoroan expmp s bumhhahs,tseouence isgrefeated fi
itttneino gmchhthe ahfeater whith ctrnsr goigg 'o whe toc trenoback ho
hhe botto  ii euch sttp,lchangingpiteW  semis eia.
 The sixts w uld bsinu t rtc .d  .ckrto itt oeigenal phsieios     w  h  t
```

Figure Error! No text of specified style in document.**.2 - Result from encrypting file content**

6. CONCLUSION

With sufficient rotations, the Rubik's cube can be used to efficiently scramble any data that needs to be encrypted. In encryption and decryption processes the important thing is that the algorithm must be reversible whether using a substitution cipher which maintains the order of the plaintext characters while simply disguising them or a transposition cipher which simply rearrange the letters but do not disguise them. Although several transposition cipher algorithms exist for example route cipher and rail fence cipher, the proposed transposition cipher algorithm based on the Rubik's cube transformations is much stronger for even the more experienced snooper to decode through a trial and error attack, a frequency analysis attack or even by a brute-force attack owing to the concealed algorithm for the cube rotations by the communication parties.

7. REFERENCES

[1] Douglas W. Mitchell (1992) "RUBIK'S CUBE" AS A TRANSPOSITION DEVICE,Cryptologia, 16:3, 250-256, DOI: 10.1080/0161-119291866928 https://www.tandfonline.com/doi/abs/10.1080/0161-119291866928

[2] nrich (2018). Transposition Cipher: https://nrich.maths.org/7940

[3] UMich, (2018). Transposition Systems http://www.umich.edu/~umich/fm-34-40-2/ch11.pdf

[4] Maiwald, E. (2003). Network Security: A Beginner's Guide. (2nd edn). McGraw-Hill/Osborne. New York. ISBN: 0072229578

[5] Stallings, W. (2003). Network Security Essentials: Applications and Standards. (2nd International edn). Upper Saddle River, NJ: Pearson Education. Chapter 2. ISBN: 0131202715

[6] Stallings, W. (2011). Network Security Essentials: Applications and Standards. (4th edn). Pearson Education, Inc., Prentice Hall. Chapter 2. ISBN: 9780136108054

[7] Kessler, G. C. (2017). An overview of Cryptography. [Online]. Available from: http://www.garykessler.net/library/crypto.html [Accessed: 1st May, 2017]

[8] Tanenbaum, A. S. (2003). Computer Ntworks. (4th edn). Upper Saddle River, N.J.: Prentice Hall. Chapter 8, pgs. 724-750. ISBN: 0130384887.

[9] Forouzan, B. A. (2001). Data Communications and Networking. (2th edn). McGraw-Hill. ISBN: 0072822945