Contents

1 Introduction

Cryptography is a method of protecting information and communications through the use of codes, so that only those for whom the information is intended can read and process it. The prefix "crypt-" means "hidden" or "vault" and the suffix "-graphy" stands for "writing."

In computer science, cryptography refers to secure information and communication techniques derived from mathematical concepts and a set of rule-based calculations called algorithms, to transform messages in ways that are hard to decipher. These deterministic algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on the internet, and confidential communications such as credit card transactions and email.

Nowadays, it is used everyday by billions of people for securing electronic mail and payment transactions. The science of cryptography touches on many other disciplines, both within mathematics and computer science and in engineering. In mathematics, cryptology uses, and touches on, algebra, number theory, graph and lattice theory, algebraic geometry and probability and statistics. Analysis of cryptographic security leads to using theoretical computer science especially complexity theory. The actual implementation of cryptosystems, and the hard work of carrying out security analysis for specific cryptosystems falls into engineering and practical computer science and computing.

In this paper, Graph Theory is being used to create a secure Encryption - Decryption algorithm. An undirected Graph is denoted by G(V, E) where V is the set of vertices and E is a set of edges that connect vertices with each other. Information can be stored in each of the vertices.

There are two ways in which we can represent Graphs: Adjacency matrix and adjacency list. In graph theory and computer science, an adjacency list is a collection of unordered lists used to represent a finite graph. Each list describes the set of neighbors of a vertex in the graph. An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph. Weighted graphs are those graphs in which every edge is assigned with some weight. Weighted graphs can also be stored using adjacency matrices.Here each cell at position M[u, v] is holding the weight of the edge from vertex u to vertex v.

In Graph Theory, a spanning tree T of an undirected graph G is a subgraph that is a tree which includes all of the vertices of G, with a minimum possible number of edges. A minimum spanning tree is a spanning tree whose sum of edge weights is as small as possible.

In this paper, an encryption algorithm has been proposed which uses Graph Theory. Weighted graphs have been used to store, encode and decode the data. Adjacency matrices have been used to represent the graphs. Various other concepts like Minimum Spanning Tree (MST), Prim’s Algorithm, matrix operations etc. have been used in the algorithm to encode/ decode data. This algorithm provides an efficient and secure method to encode important data like passwords and other sensitive information and send the data to the receiver where the data will be decoded and used for further processing.

2 Work Done

In this section, the algorithm and the working of the proposed cryptography method will be mentioned. Detail, flowcharts and examples related to our proposed method will also be mentioned here.

A. PROPOSED ALGORITHM

When the message has been fetched from the user, the characters in the message are represented as vertices in a graph. All the vertices are added in the graph sequentially so as to generate a cyclic graph. Then, edge weights are assigned to each of the edges in the cycle created by using a particular scheme which will be discussed shortly. All the remaining edges, i.e. the diagonals of the cycle are then drawn and weights are assigned to them. We add an extra vertex and an extra edge with a special character, suppose 'A' to point to the starting character of the message. Thus, now we have a complete weighted undirected graph. Now, the next step is to compute the Minimum Spanning Tree(MST) of the generated complete graph. The MST is represented in the adjacency matrix and its diagonal stores the order of the characters in the original message. Adjacency-matrix of the complete graph is then multiplied to the adjacency-matrix of MST. The resultant matrix is multiplied to the key matrix. The final matrix is the encryption data to be sent to the recipient.

Scheme for assigning weights to the edges:

- Start with the character that has been used to point to the 1st character of the input string ('A' in our case). Calculate the difference between the ASCII values of the 1st character of the input string and this pointer character and assign this weight to the edge between these two vertices by adding one.
- If the edge is a side of the polygon(cycle) or between the special character and the starting character, then for computing the edge weights, ASCII values of the characters are considered and the edge weight of any such edge is computed by calculating the difference between the ASCII values of the characters and then adding one to it. Calculate this difference for each edge in a cyclic manner. (See the example part for more info).
- The reason for adding one in the above method is to avoid the edge weight of two same characters from becoming zero. And if the edge weight becomes zero, then while computing the MST the program will consider that no edge lies between these vertices which is a wrong assumption.
- If the edge is a diagonal of the polygon, then edge weights are sequentially assigned starting from 256 and increasing the weight by 1 each time.

Encryption Algorithm:

- Add a special character to indicate the starting character (Let's say A).
- Add vertex for each character in the plain text to the graph sequentially.
- Link vertices together by adding an edge between each sequential character in the plain text until we form a cycle graph.
- At this point the graph should look like a cycle with one edge sticking out.
- Assign weight to each edge using the scheme as discussed earlier.
- Add more edges (diagonals) to form a complete graph M_2, each new added edge has a sequential weight starting from 256.
- Then compute the Minimum Spanning Tree: M_2.
- Then store the vertices in proper order in the M_2 matrix in the diagonal places.
- Next multiply matrices M_1 by M_2 to get M_3.
- After that we multiply M_3 by a predefined Shared-Key K to form Cipher.
- After the encryption process, the Cipher matrix and the M_1 matrix are to be sent as the encoded message.

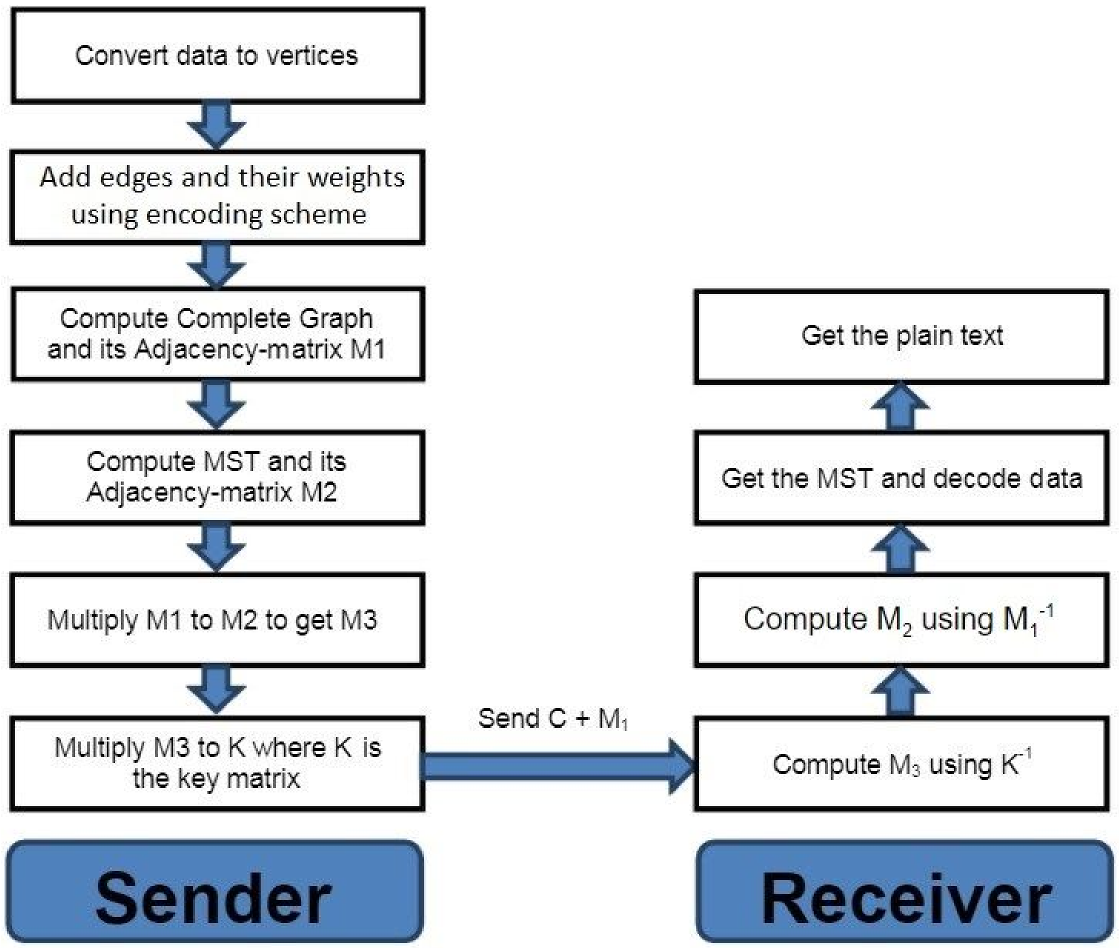

Convert data to vertices
Add edges and their weights using encoding scheme
Compute Complete Graph and its Adjacency-matrix M1
Compute MST and its Adjacency-matrix M2
Multiply M1 to M2 to get M3
Multiply M3 to K where K is the key matrix
Send C + M₁
Compute M₃ using K⁻¹
Compute M₂ using M₁⁻¹
Get the MST and decode data
Get the plain text
Sender
Receiver

Figure : Flow Chart

Decryption Algorithm:

- The receiver receives the cipher and the M1 matrix separately from the sender.
- The receiver computes M_3 by using the inverse form of the Shared-Key: K^{-1} and then multiplying the matrix with C.
- Then compute M_2 by using the inverse form of M_1, i.e. M_1^{-1} multiplied with matrix M_3.
- Now, the receiver end has computed the MST matrix, M_2.
- Then compute the original text by decoding M_2 using the encoding scheme.
- This original text is then displayed at the receiver end.

B. EXAMPLE 1

Let's encrypt the message "WAEL". INPUT STRING = "WAEL"

ENCODING:

Step1: Converting characters of the message to vertices of graph

The first step is to convert the message to a graph, by converting each character to a vertex as shown in Fig. 1.

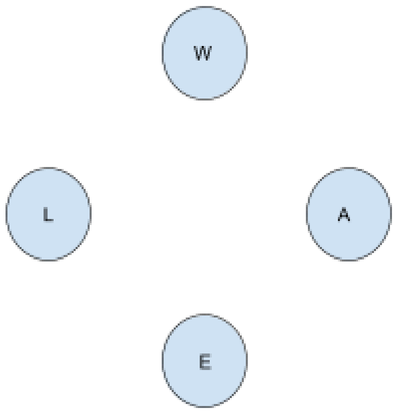


Fig 1: Step 1

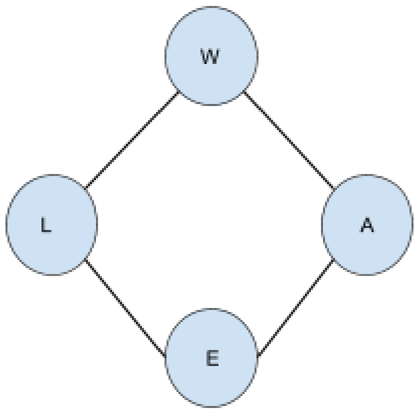


Fig 2: Step 2

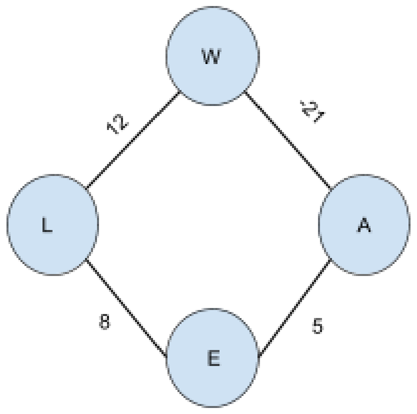


Fig 3: Step 3

Step 2: Adding edges to form a complete cycle of the input

Then, link each two sequential characters together to form a cycle graph. Now the graph should look like a cyclic polygon. (See Fig 2)

Step 3: Add weights to each of the edges of the cycle

Add weights for each of the edges using the weight scheme discussed in the algorithm section (Using difference of ASCII values of adjacent vertices- See Fig 3)

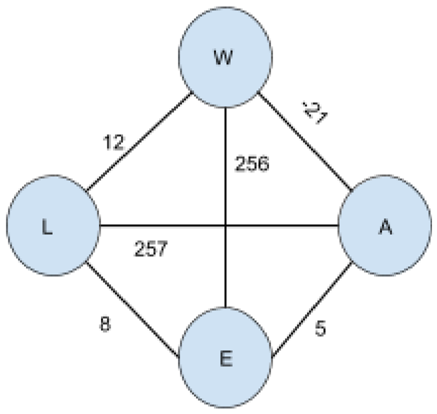


Fig 4: Step 4

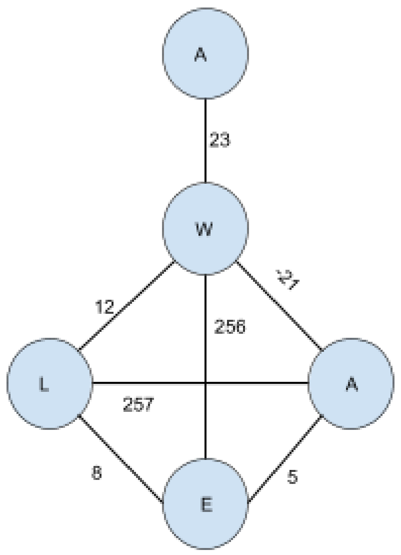


Fig 5: Step 5

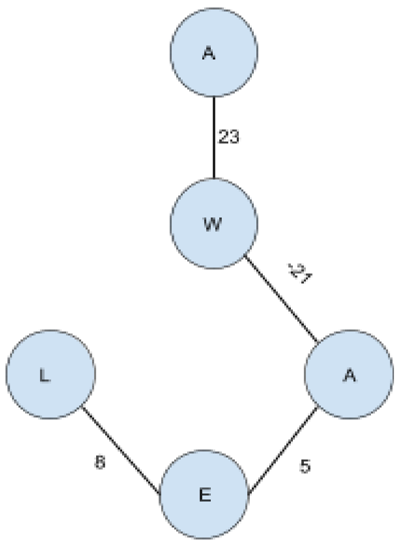


Fig 6: Step 6

Step 4: Add weights to the diagonals to make the complete graph

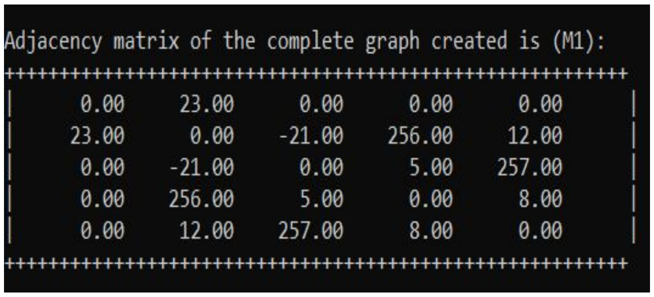
After that, keep adding edges to form a complete graph. This can be done by connecting all the diagonals. Each new added edge has sequential weight starting from the maximum weight i.e. 256. The weights of the diagonals will be 256, 257, 258… (See Fig 4)

Step 5: Add a special character to indicate the starting character

Now, add an extra vertex labelled “A” to the first character of the message and then compute the edge weight of this newly created edge using the encoding scheme. (See Fig 5) The complete adjacency matrix, M_1, is shown in Fig 7.

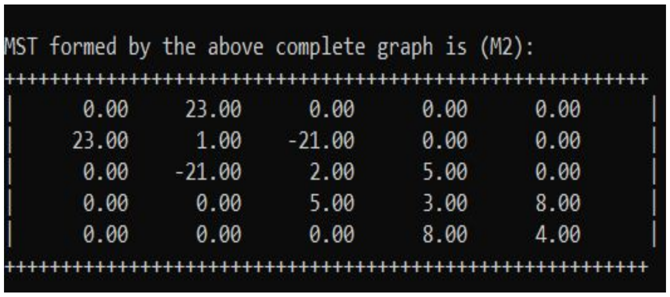
Step 6: Compute MST and its adjacency matrix M_2

As we have the complete graph now, compute the MST by using the Prim’s Algorithm. (See Fig 6) The adjacency matrix for this MST, M_2, is shown in Fig 8.



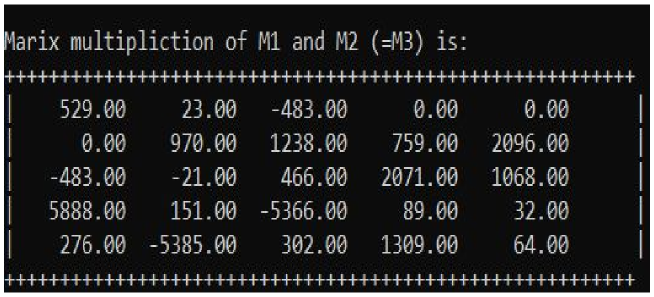
```
Adjacency matrix of the complete graph created is (M1):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|     0.00     23.00      0.00      0.00      0.00      |
|    23.00      0.00    -21.00    256.00     12.00      |
|     0.00    -21.00      0.00      5.00    257.00      |
|     0.00    256.00      5.00      0.00      8.00      |
|     0.00     12.00    257.00      8.00      0.00      |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig 7: Matrix M1



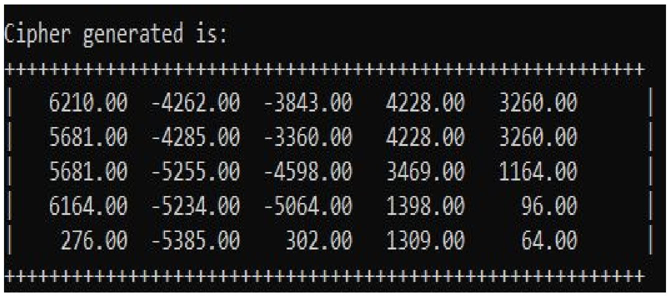
```
MST formed by the above complete graph is (M2):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|     0.00     23.00      0.00      0.00      0.00      |
|    23.00      1.00    -21.00      0.00      0.00      |
|     0.00    -21.00      2.00      5.00      0.00      |
|     0.00      0.00      5.00      3.00      8.00      |
|     0.00      0.00      0.00      8.00      4.00      |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig 8: Matrix M2



```
Marix multipliction of M1 and M2 (=M3) is:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
|    529.00     23.00   -483.00      0.00      0.00    |
|      0.00    970.00   1238.00    759.00   2096.00    |
|   -483.00    -21.00    466.00   2071.00   1068.00    |
|   5888.00    151.00  -5366.00     89.00     32.00    |
|    276.00  -5385.00    302.00   1309.00     64.00    |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig 9: Matrix M3



```
Cipher generated is:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
|   6210.00  -4262.00  -3843.00   4228.00   3260.00    |
|   5681.00  -4285.00  -3360.00   4228.00   3260.00    |
|   5681.00  -5255.00  -4598.00   3469.00   1164.00    |
|   6164.00  -5234.00  -5064.00   1398.00     96.00    |
|    276.00  -5385.00    302.00   1309.00     64.00    |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig 10: Cipher Matrix

Step 7: Computing Matrix M_3

We calculate matrix M_3 by matrix multiplication of M_1 and M_2. The matrix M_3 for "WAEL" is shown in Fig.9 above.

$$M_3 = M_1 * M_2$$

Step 8: Computing Cipher Matrix

This is the final step in encrypting the input message. Here, the Cipher matrix is generated by matrix multiplication of the pre-defined Key and M3. Now the Cipher and M1 are ready to be sent to the receiver. Fig.10 above shows the Cipher for our input message.

$$\text{Cipher} = \text{Key} * M_3$$

$$\text{Send (Cipher} + M_1\text{) to Receiver}$$

DECODING:

Step 1: Receive the data, compute inverse of Key

The receiver gets the encoded message and separates the message to retrieve the Cipher matrix (Fig 10) and the M_1 matrix (Fig 7). The receiver node then calculates the inverse of the pre- shared Key.

The Key is shown in Fig 11 and the Key^{-1} (Key- inverse) is shown in Fig 12.

Step 2: Compute M_3 using Key^{-1} and Cipher

Now, the matrix M3 is reconstructed using the key- inverse (Key^{-1}) matrix. M3 is calculated by the matrix multiplication of Key^{-1} and Cipher matrices.

$$M_3 = \text{Key}^{-1} * \text{Cipher}$$

Step 3: Compute M2 using $M1^{-1}$ and M3

Next, Matrix M2 is reconstructed using the inverse of matrix M_1. M_2 is calculated by the matrix multiplication of $M1^{-1}$ and M3 matrices.

$$M_2 = M1^{-1} * M_3$$

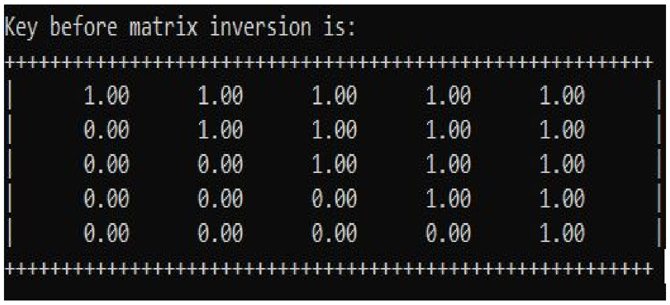
After calculation of Matrix M2, the each value in the double- data type M2 matrix has to be converted to the nearest integer by rounding off.

Matrix M_1^{-1} is shown in Fig 13 and Matrix M_2 after rounding off to the nearest integer is shown in Fig 14.

Step 4: Get the MST and decode data

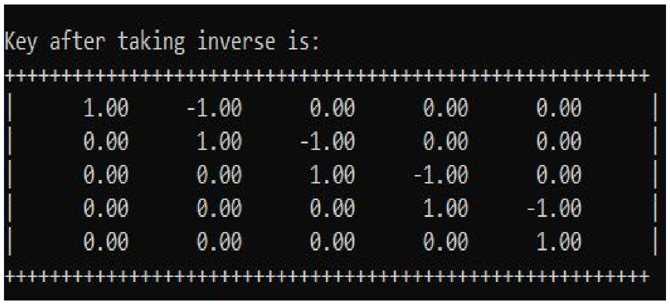
The matrix M2 (Fig 14) itself represents the MST. After the MST has been extracted, the original message is decoded from it. To get the original message back, first we traverse through the right part of the MST (Fig 6) starting from the top and keep adding the weights every time and find the ASCII character corresponding to the added weight at each edge. Thus a new character is found by adding the weight at each node and this character is concatenated to the end of the reconstructed string

But, this method of decoding the data from MST (Step 4 in decoding) will not work for all messages. This method worked for the message "WAEL" because the MST of this graph is cyclic (excluding the special character- See Fig 6). But, in some messages the computed MST may not be entirely cyclic. The MST can also be branched. In that case, we need to change our Step 4 a little bit. In example 2 the Step 4 of decoding, has been extended further to cover all cases.



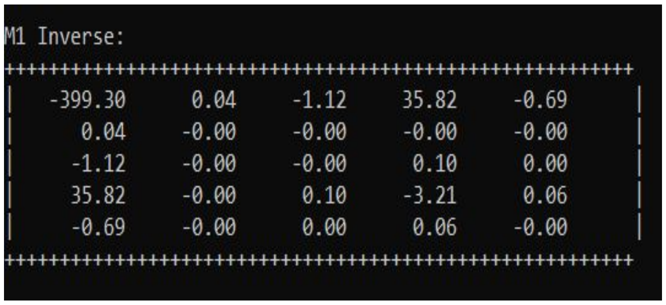
```
Key before matrix inversion is:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|     1.00     1.00     1.00     1.00     1.00          |
|     0.00     1.00     1.00     1.00     1.00          |
|     0.00     0.00     1.00     1.00     1.00          |
|     0.00     0.00     0.00     1.00     1.00          |
|     0.00     0.00     0.00     0.00     1.00          |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig 11: Shared-Key



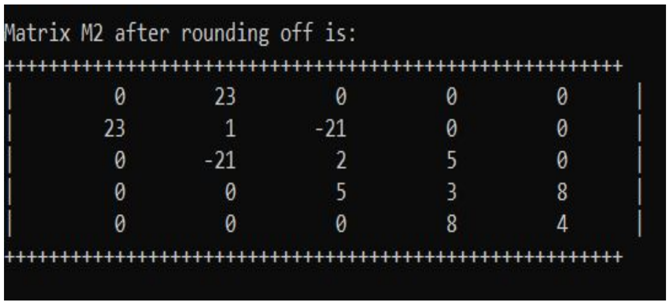
```
Key after taking inverse is:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|     1.00    -1.00     0.00     0.00     0.00          |
|     0.00     1.00    -1.00     0.00     0.00          |
|     0.00     0.00     1.00    -1.00     0.00          |
|     0.00     0.00     0.00     1.00    -1.00          |
|     0.00     0.00     0.00     0.00     1.00          |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig 12: Key Inverse



```
M1 Inverse:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  -399.30     0.04    -1.12    35.82    -0.69          |
|     0.04    -0.00    -0.00    -0.00    -0.00          |
|    -1.12    -0.00    -0.00     0.10     0.00          |
|    35.82    -0.00     0.10    -3.21     0.06          |
|    -0.69    -0.00     0.00     0.06    -0.00          |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig 13: M1 Inverse



```
Matrix M2 after rounding off is:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|        0       23        0        0        0          |
|       23        1      -21        0        0          |
|        0      -21        2        5        0          |
|        0        0        5        3        8          |
|        0        0        0        8        4          |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig 14: Matrix M2 after rounding

C. EXAMPLE 2

In the previous example, the MST corresponding to the constructed graph of the word “WAEL” is cyclic (Fig 6). So, the Step-4 of decoding (Get the MST and decode data) was quite simple. But in many messages, the generated MST can be branched. This can be visualised by using the algorithm for the word “HELLO”. Fig 15- 20 show the various steps involved in extracting the MST corresponding to the message “HELLO”. We can see that the generated MST (Fig 20) is actually branched (neglecting the 1st special character). So, the final decoding process has to be extended to cover all such cases.

In this case, first traverse through the right branch of the MST and follow the normal process of adding the weights at each node and getting the corresponding. Then if an edge is absent (denoted by 0 in adjacency matrix), stop the traversing. Come back to the first character (H in this case) and this time traverse through the left branch and every time subtract the weight from the ASCII value of the 1st character (H). The character corresponding to this new ASCII value has to be concatenated to a new string. This is done till we reach the end of the left branch. Finally reverse this string and concatenate this string with the string generated during traversal of the right branch. This will give the final decoded message.

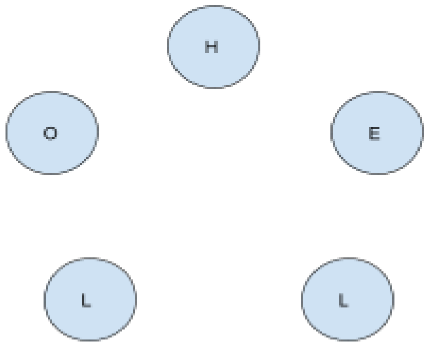


Fig 15: Step 1

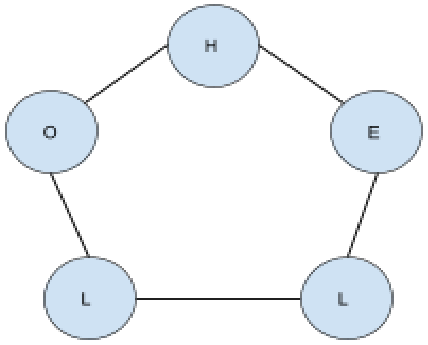


Fig 16: Step 2

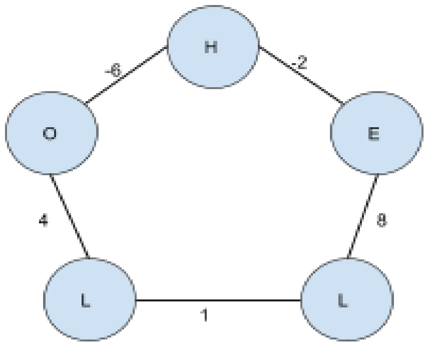


Fig 17: Step 3

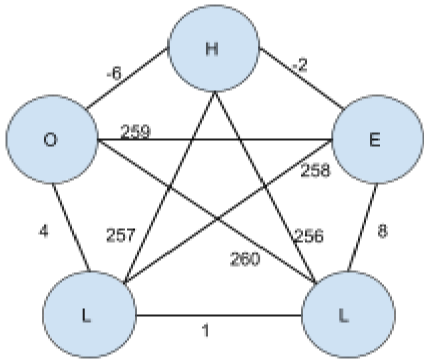


Fig 18: Step 4

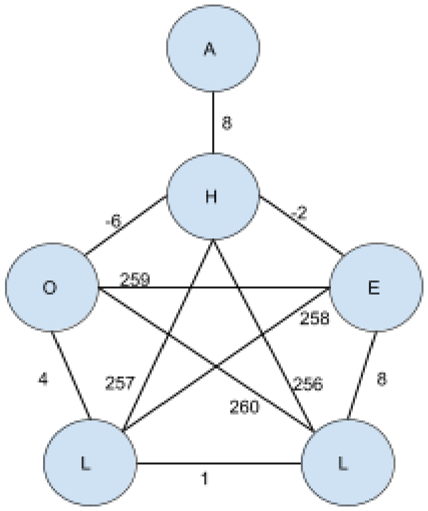


Fig 19: Step 5

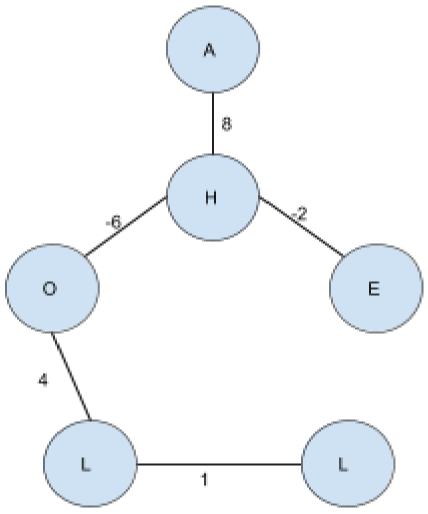
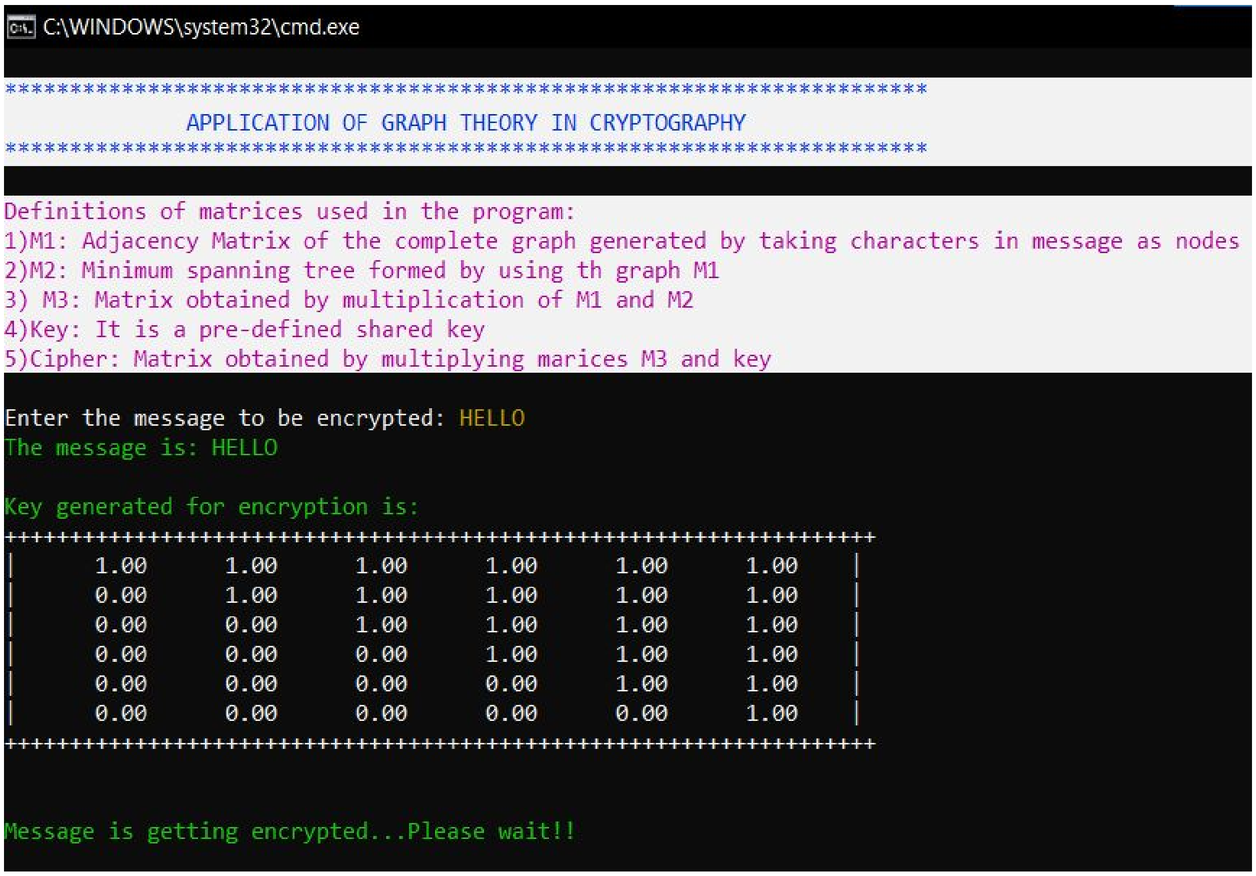


Fig 20: Step 6

3 Results and Discussion

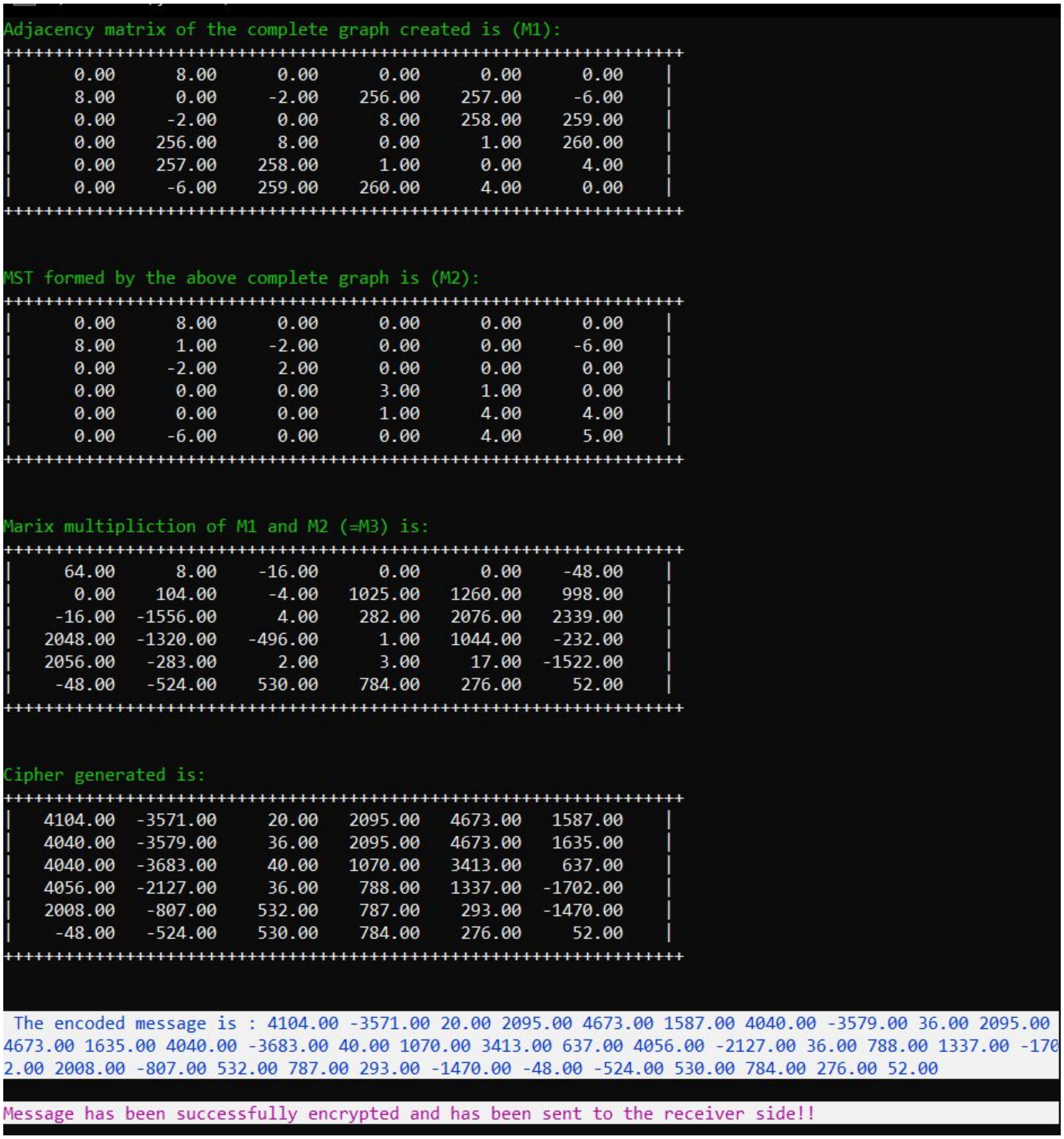
This proposed algorithm was implemented by us using C++ programming language. Our implementation takes the message as input. First the program encodes the input message using the encoding algorithm. Then this encoded message is passed on to the decoding module of our program. The decoding module decodes the message and finally displays the decoded message.

The snapshots when the word "HELLO" was given as input message to our implementation are shown below:



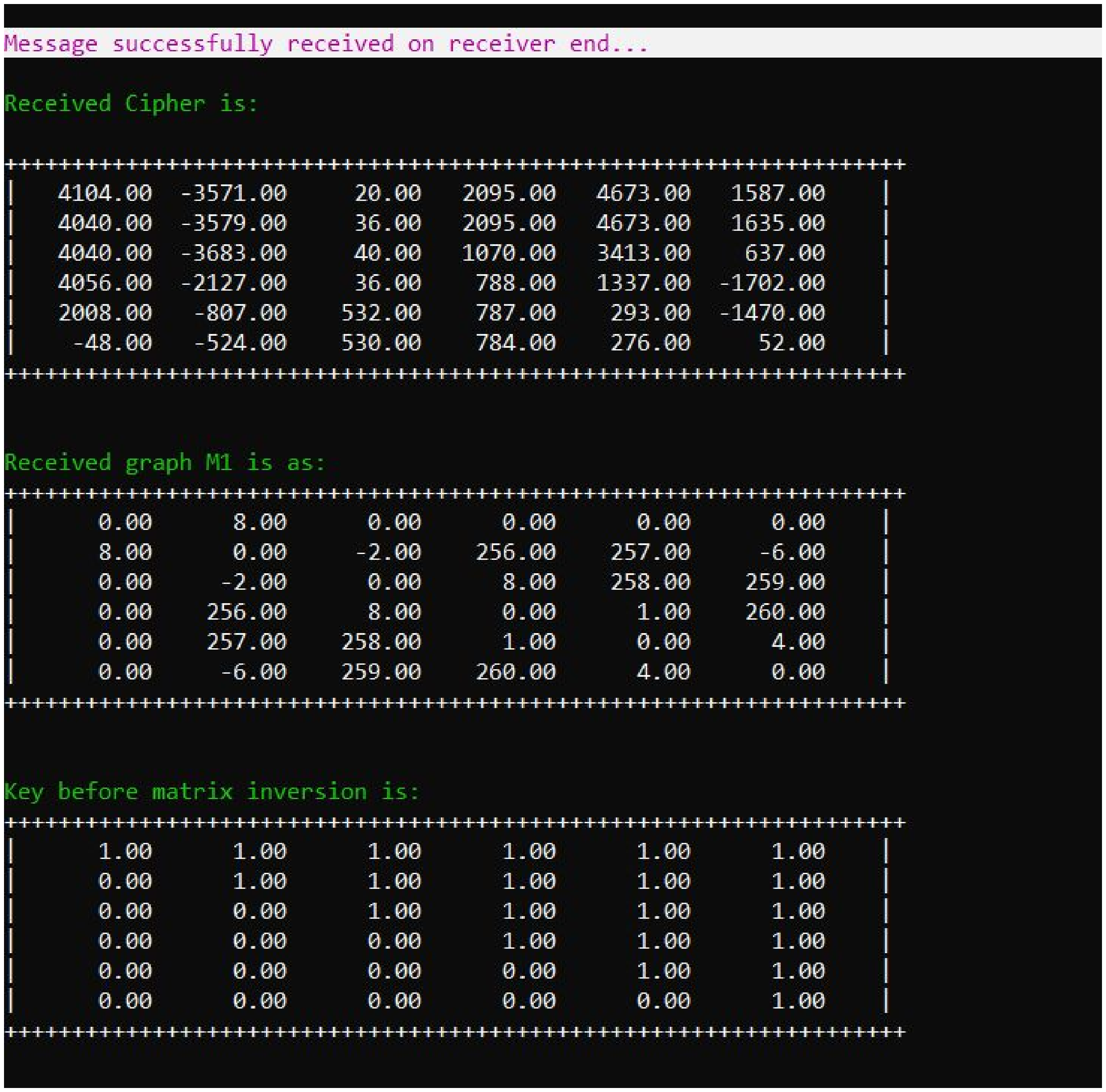
```
C:\WINDOWS\system32\cmd.exe

************************************************************************
              APPLICATION OF GRAPH THEORY IN CRYPTOGRAPHY
************************************************************************

Definitions of matrices used in the program:
1)M1: Adjacency Matrix of the complete graph generated by taking characters in message as nodes
2)M2: Minimum spanning tree formed by using th graph M1
3) M3: Matrix obtained by multiplication of M1 and M2
4)Key: It is a pre-defined shared key
5)Cipher: Matrix obtained by multiplying marices M3 and key

Enter the message to be encrypted: HELLO
The message is: HELLO

Key generated for encryption is:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|      1.00      1.00      1.00      1.00      1.00      1.00     |
|      0.00      1.00      1.00      1.00      1.00      1.00     |
|      0.00      0.00      1.00      1.00      1.00      1.00     |
|      0.00      0.00      0.00      1.00      1.00      1.00     |
|      0.00      0.00      0.00      0.00      1.00      1.00     |
|      0.00      0.00      0.00      0.00      0.00      1.00     |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Message is getting encrypted...Please wait!!
```

Fig : Message is fetched, which is to be sent



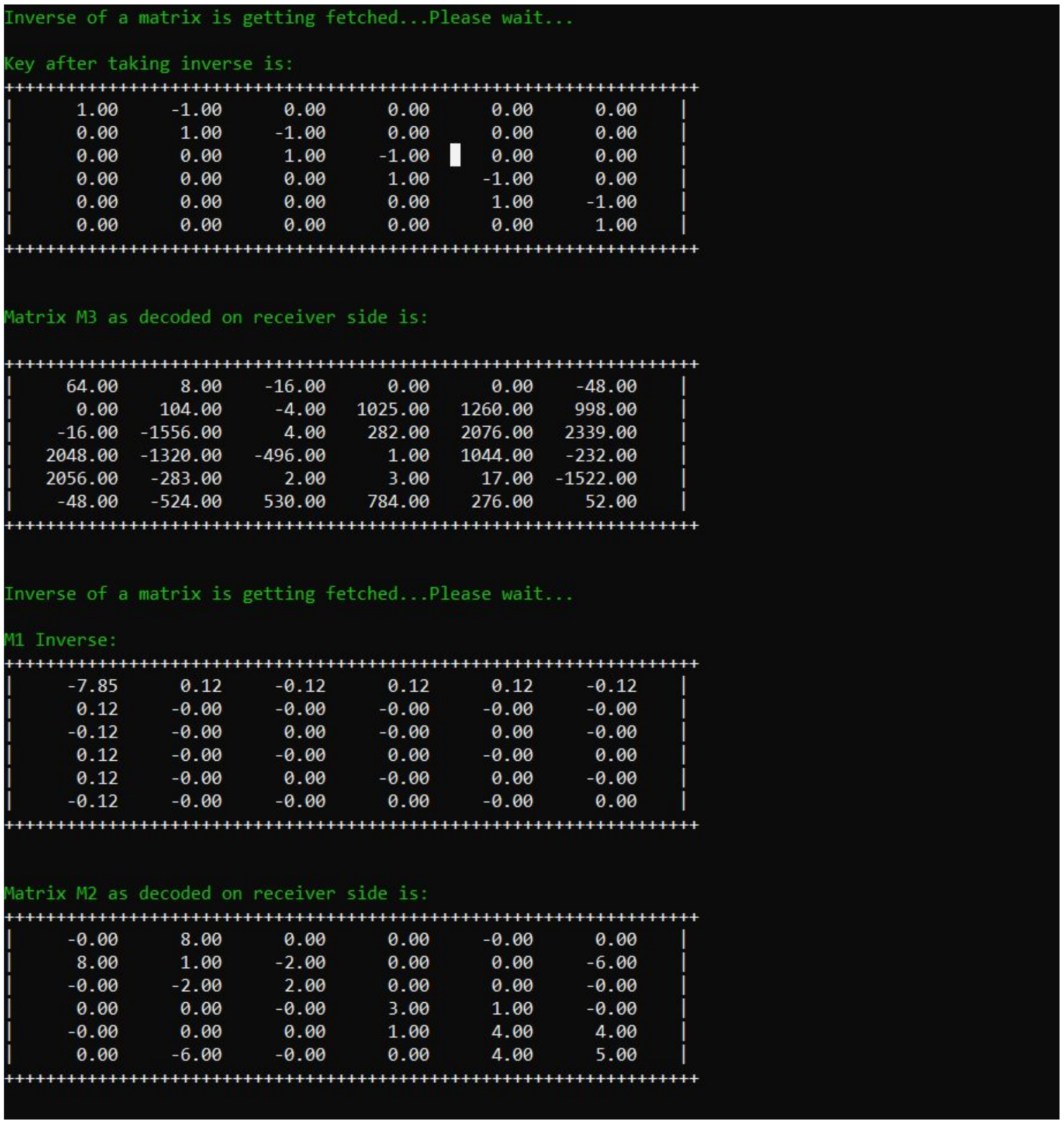
```
Adjacency matrix of the complete graph created is (M1):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|      0.00      8.00      0.00      0.00      0.00      0.00    |
|      8.00      0.00     -2.00    256.00    257.00     -6.00    |
|      0.00     -2.00      0.00      8.00    258.00    259.00    |
|      0.00    256.00      8.00      0.00      1.00    260.00    |
|      0.00    257.00    258.00      1.00      0.00      4.00    |
|      0.00     -6.00    259.00    260.00      4.00      0.00    |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


MST formed by the above complete graph is (M2):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|      0.00      8.00      0.00      0.00      0.00      0.00    |
|      8.00      1.00     -2.00      0.00      0.00     -6.00    |
|      0.00     -2.00      2.00      0.00      0.00      0.00    |
|      0.00      0.00      0.00      3.00      1.00      0.00    |
|      0.00      0.00      0.00      1.00      4.00      4.00    |
|      0.00     -6.00      0.00      0.00      4.00      5.00    |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Marix multipliction of M1 and M2 (=M3) is:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|     64.00      8.00    -16.00      0.00      0.00    -48.00    |
|      0.00    104.00     -4.00   1025.00   1260.00    998.00    |
|    -16.00  -1556.00      4.00    282.00   2076.00   2339.00    |
|   2048.00  -1320.00   -496.00      1.00   1044.00   -232.00    |
|   2056.00   -283.00      2.00      3.00     17.00  -1522.00    |
|    -48.00   -524.00    530.00    784.00    276.00     52.00    |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Cipher generated is:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|   4104.00  -3571.00     20.00   2095.00   4673.00   1587.00    |
|   4040.00  -3579.00     36.00   2095.00   4673.00   1635.00    |
|   4040.00  -3683.00     40.00   1070.00   3413.00    637.00    |
|   4056.00  -2127.00     36.00    788.00   1337.00  -1702.00    |
|   2008.00   -807.00    532.00    787.00    293.00  -1470.00    |
|    -48.00   -524.00    530.00    784.00    276.00     52.00    |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


 The encoded message is : 4104.00 -3571.00 20.00 2095.00 4673.00 1587.00 4040.00 -3579.00 36.00 2095.00
4673.00 1635.00 4040.00 -3683.00 40.00 1070.00 3413.00 637.00 4056.00 -2127.00 36.00 788.00 1337.00 -170
2.00 2008.00 -807.00 532.00 787.00 293.00 -1470.00 -48.00 -524.00 530.00 784.00 276.00 52.00

Message has been successfully encrypted and has been sent to the receiver side!!
```

Figure : Message Encrypted



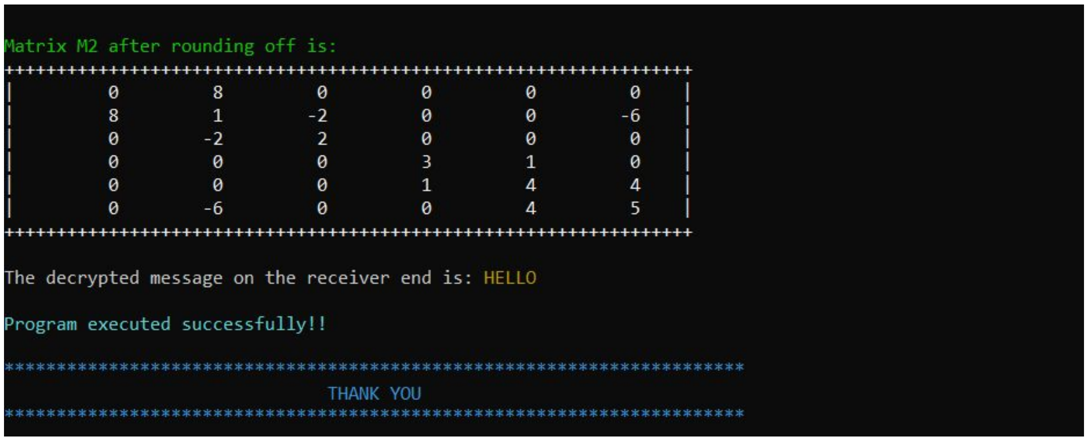
```
Message successfully received on receiver end...

Received Cipher is:

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|   4104.00  -3571.00     20.00   2095.00   4673.00   1587.00     |
|   4040.00  -3579.00     36.00   2095.00   4673.00   1635.00     |
|   4040.00  -3683.00     40.00   1070.00   3413.00    637.00     |
|   4056.00  -2127.00     36.00    788.00   1337.00  -1702.00     |
|   2008.00   -807.00    532.00    787.00    293.00  -1470.00     |
|    -48.00   -524.00    530.00    784.00    276.00     52.00     |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Received graph M1 is as:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|      0.00      8.00      0.00      0.00      0.00      0.00     |
|      8.00      0.00     -2.00    256.00    257.00     -6.00     |
|      0.00     -2.00      0.00      8.00    258.00    259.00     |
|      0.00    256.00      8.00      0.00      1.00    260.00     |
|      0.00    257.00    258.00      1.00      0.00      4.00     |
|      0.00     -6.00    259.00    260.00      4.00      0.00     |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Key before matrix inversion is:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|      1.00      1.00      1.00      1.00      1.00      1.00     |
|      0.00      1.00      1.00      1.00      1.00      1.00     |
|      0.00      0.00      1.00      1.00      1.00      1.00     |
|      0.00      0.00      0.00      1.00      1.00      1.00     |
|      0.00      0.00      0.00      0.00      1.00      1.00     |
|      0.00      0.00      0.00      0.00      0.00      1.00     |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure : Cipher and M_1 are received and Decryption starts



```
Inverse of a matrix is getting fetched...Please wait...

Key after taking inverse is:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|      1.00     -1.00      0.00      0.00      0.00      0.00     |
|      0.00      1.00     -1.00      0.00      0.00      0.00     |
|      0.00      0.00      1.00     -1.00      0.00      0.00     |
|      0.00      0.00      0.00      1.00     -1.00      0.00     |
|      0.00      0.00      0.00      0.00      1.00     -1.00     |
|      0.00      0.00      0.00      0.00      0.00      1.00     |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Matrix M3 as decoded on receiver side is:

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|     64.00      8.00    -16.00      0.00      0.00    -48.00     |
|      0.00    104.00     -4.00   1025.00   1260.00    998.00     |
|    -16.00  -1556.00      4.00    282.00   2076.00   2339.00     |
|   2048.00  -1320.00   -496.00      1.00   1044.00   -232.00     |
|   2056.00   -283.00      2.00      3.00     17.00  -1522.00     |
|    -48.00   -524.00    530.00    784.00    276.00     52.00     |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Inverse of a matrix is getting fetched...Please wait...

M1 Inverse:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|     -7.85      0.12     -0.12      0.12      0.12     -0.12     |
|      0.12     -0.00     -0.00     -0.00     -0.00     -0.00     |
|     -0.12     -0.00      0.00     -0.00      0.00     -0.00     |
|      0.12     -0.00     -0.00      0.00     -0.00      0.00     |
|      0.12     -0.00      0.00     -0.00      0.00     -0.00     |
|     -0.12     -0.00     -0.00      0.00     -0.00      0.00     |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Matrix M2 as decoded on receiver side is:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|     -0.00      8.00      0.00      0.00     -0.00      0.00     |
|      8.00      1.00     -2.00      0.00      0.00     -6.00     |
|     -0.00     -2.00      2.00      0.00      0.00     -0.00     |
|      0.00      0.00     -0.00      3.00      1.00     -0.00     |
|     -0.00      0.00      0.00      1.00      4.00      4.00     |
|      0.00     -6.00     -0.00      0.00      4.00      5.00     |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure : Message Decrypting



```
Matrix M2 after rounding off is:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|         0         8         0         0         0         0     |
|         8         1        -2         0         0        -6     |
|         0        -2         2         0         0         0     |
|         0         0         0         3         1         0     |
|         0         0         0         1         4         4     |
|         0        -6         0         0         4         5     |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

The decrypted message on the receiver end is: HELLO

Program executed successfully!!

***********************************************************************
                              THANK YOU
***********************************************************************
```

Figure : Decrypted Message is displayed at receiver end

4 Conclusion and Future Work

In this paper, we have proposed a technique to encrypt a message using concepts of Graph theory like Minimum Spanning Trees (MST), Prim's algorithm, adjacency matrix etc. Cryptography is necessary in the modern era to send and receive data safely and securely and our proposed algorithm is able to encrypt the messages with high security. We have used a secure pre-defined key which is shared between the sender and the recipient only. The cipher is then generated and passed on to the receiver along with the graph. Once the recipient receives the necessary data, the MST is re-constructed by the receiver and the message is regenerated by re-tracing the encryption algorithm in the reverse order. The order of the characters in the message are determined by traversing the values in the correct order in the diagonal of the computed MST by the receiver.

This algorithm almost works for all types of characters in the range of ASCII and can be safely used to encrypt passwords and secure data of short lengths.

Future Work:

- For messages of greater lengths, the computational time is comparatively higher as time is consumed in calculating the inverse of a matrix which takes $O(n^3)$ time complexity.
- In such cases the data of greater lengths can be divided into smaller sub parts and then encrypted. The smaller sub parts decrypted on the receiver end can then be concatenated to form the original message. This will increase the efficiency of the algorithm to a large extent.
- Further, we can try to optimize our algorithm by reducing the complexity of various modules of our algorithm.

5 References

1. *Encryption Algorithm Using Graph Theory*. 2014. N.p.: Wael Etaiwi.

   https://www.researchgate.net/publication/269803082_Encryption_Algorithm_Using_Graph_Theory.