Eastern Mediterranean University Department of Computer Engineering CMPE553: Cryptography and network security

Project: Encrypt and decrypt a message using hill cipher n=2 Submitted to: Assoc. Prof. Dr. Alexander Chefranov

Submitted by:

AHMET ÖZDEMİR 20510466

ALP DAVUTOĞLU 20510418

ANWAR ALBARIQI 20500161

HASAN ERBILEN 19500647

1. Task Definition

The purpose of the project is to develop application to encrypt and decrypt a certain text using hill cipher n=2 , which include both letters and numbers .The project is a network application that should be able to test ciphers working over respective alphabets using hill cipher n=2, which is able to encrypt and decrypt, in our case we shall have two parties each part of the communication process will run on his own computer and able to send or receive data, which is have to be encrypted during its transferring process through some channel, we will represent this project via report including a simple description of the hill cipher n=2 encryption algorithm, also a code with C# programming language for encryption and decryption process .

2. Brief definition Of An Algorithm To Be Implemented

2.1.Hill Cipher

Hill cipher is a polytrophic substitution cipher based on linear algebra . Each letter is represented by a number modulo 26 . Often the simple scheme A = 0, B = 1, …, Z = 25 is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.

The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).

2x2 Example:

We shall encrypt the plaintext message "ortexa" using the keyword hill and a 2 x 2 matrix.The first step is to turn the keyword into a matrix. If the keyword was longer than the 4 letters needed, we would only take the first 4 letters, and if it was shorter, we would fill it up with the alphabet in order.

With the keyword in a matrix, we need to convert this into a key matrix. We do this by converting each letter into a number by its position in the alphabet (starting at 0). So, A = 0, B = 1, C= 2, D = 3, etc.

The key matrix is

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}$$

Figure1:Key matrix

We now split the plaintext into digraphs, and write these as column vectors. That is, in the first column vector we write the first plaintext letter at the top, and the second letter at the bottom. Then we move to the next column vector, where the third plaintext letter goes at the top, and the fourth at the bottom. This continues for the whole plaintext.

$$\begin{pmatrix} o \\ r \end{pmatrix} \begin{pmatrix} t \\ e \end{pmatrix} \begin{pmatrix} x \\ a \end{pmatrix}$$

Figure2:The plaintext "short example" split into column vectors.

Now we must convert the plaintext column vectors in the same way that we converted the keyword into the key matrix. Each letter is replaced by its appropriate number.

$$\begin{pmatrix} 14 \\ 17 \end{pmatrix} \begin{pmatrix} 19 \\ 4 \end{pmatrix} \begin{pmatrix} 23 \\ 0 \end{pmatrix}$$

Figure3:The plaintext converted into numeric column vectors.

Now we must perform some matrix multiplication. We multiply the key matrix by each column vector in turn. We shall go through the first of these in detail, then the rest shall be presented in less detail. We write the key matrix first, followed by the column vector.

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 14 \\ 17 \end{pmatrix}$$

$$7 \times 14 + 8 \times 17 = 234$$

$$11 \times 14 + 11 \times 17 = 341$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 14 \\ 17 \end{pmatrix} = \begin{pmatrix} 234 \\ 341 \end{pmatrix}$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 14 \\ 17 \end{pmatrix} = \begin{pmatrix} 234 \\ 341 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \ mod\ 26$$

$$\begin{pmatrix} H & I \\ L & L \end{pmatrix}\begin{pmatrix} o \\ r \end{pmatrix} = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 14 \\ 17 \end{pmatrix} = \begin{pmatrix} 234 \\ 341 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \ mod\ 26 = \begin{pmatrix} A \\ D \end{pmatrix}$$

Figure4:Encryption of the first digraph:

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 19 \\ 4 \end{pmatrix}$$

$$7 \times 19 + 8 \times 4 = 165$$

$$11 \times 19 + 11 \times 4 = 253$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 19 \\ 4 \end{pmatrix} = \begin{pmatrix} 165 \\ 253 \end{pmatrix}$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 19 \\ 4 \end{pmatrix} = \begin{pmatrix} 165 \\ 253 \end{pmatrix} = \begin{pmatrix} 9 \\ 19 \end{pmatrix} \ mod\ 26$$

$$\begin{pmatrix} H & I \\ L & L \end{pmatrix}\begin{pmatrix} t \\ e \end{pmatrix} = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 19 \\ 4 \end{pmatrix} = \begin{pmatrix} 165 \\ 253 \end{pmatrix} = \begin{pmatrix} 9 \\ 19 \end{pmatrix} \ mod\ 26 = \begin{pmatrix} J \\ T \end{pmatrix}$$

Figure5:Encryption of the second digraph:

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 23 \\ 0 \end{pmatrix}$$

$$7 \times 23 + 8 \times 0 = 161$$

$$11 \times 23 + 11 \times 0 = 253$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 23 \\ 0 \end{pmatrix} = \begin{pmatrix} 161 \\ 253 \end{pmatrix}$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 23 \\ 0 \end{pmatrix} = \begin{pmatrix} 161 \\ 253 \end{pmatrix} = \begin{pmatrix} 5 \\ 19 \end{pmatrix} \ mod\ 26$$

$$\begin{pmatrix} H & I \\ L & L \end{pmatrix}\begin{pmatrix} x \\ a \end{pmatrix} = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}\begin{pmatrix} 23 \\ 0 \end{pmatrix} = \begin{pmatrix} 161 \\ 253 \end{pmatrix} = \begin{pmatrix} 5 \\ 19 \end{pmatrix} \ mod\ 26 = \begin{pmatrix} F \\ T \end{pmatrix}$$

Figure6:Encryption of the first digraph:

This gives us a final ciphertext of "ADJ TFT".

We shall decrypt the example above, so we are using the keyword *hill* and our ciphertext is "APADJ TFTWLFJ". We start by writing out the keyword as a matrix and converting this into a key matrix as for encryption. Now we must convert this to the inverse key matrix, for which there are several steps.

Step1- Find the Multiplicative Inverse of the Determinant:
The determinant is a number that relates directly to the entries of the matrix. It is found by multiplying the top left number by the bottom right number and subtracting from this the product of the top right number and the bottom left number. This is shown algebraically below. Note that the notation for determinant has straight lines instead of brackets around our matrix.

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Figure7: Find the Determinant:

Once we have found this value, we need to take the number modulo 26. Below is the way to calculate the determinant for our example.

$$\begin{vmatrix} 7 & 8 \\ 11 & 11 \end{vmatrix} = 7 \times 11 - 8 \times 11 = -11 = 15 \; mod \; 26$$

Figure7: Find the Determinant:take the number modulo 26

We now have to find the multiplicative inverse of the determinant working modulo 26. That is, the number between 1 and 25 that gives an answer of 1 when we multiply it by the determinant. So, in this case, we are looking for the number that we need to multiply 15 by to get an answer of 1 modulo 26. There are algorithms to calculate this, but it is often easiest to use trial and error to find the inverse.

$$dd^{-1} = 1 \; mod \; 26$$

Figure8:If d is the determinant, then we are looking for the inverse of d.

$$15 \times x = 1 \; mod \; 26$$

Figure9:The multiplicative inverse is the number we multiply 15 by to get 1 modulo 26.

$$15 \times 7 = 105 = 1 \; mod \; 26$$

Figure10:This calculation gives us an answer of 1 modulo 26.

So the multiplicative inverse of the determinant modulo 26 is 7. We shall need this number later.

Step 2 - Find the Adjugate Matrix:

The adjugate matrix is a matrix of the same size as the original. For a 2 x 2 matrix, this is fairly straightforward as it is just moving the elements to different positions and changing a couple of signs. That is, we swap the top left and bottom right numbers in the key matrix, and change the sign of the the top right and bottom left numbers. Algebraically this is given below.

$$adj \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Figure11:the adjugate matrix of a 2 x 2 matrix.

Again, once we have these values we will need to take each of them modulo 26 (in particular, we need to add 26 to the negative values to get a number between 0 and 25. For our example we get the matrix below.

$$adj\begin{pmatrix}7 & 8\\ 11 & 11\end{pmatrix} = \begin{pmatrix}11 & -8\\ -11 & 7\end{pmatrix} = \begin{pmatrix}11 & 18\\ 15 & 7\end{pmatrix}$$

Figure12:The adjugate matrix of the key matrix.

Step 3 - Multiply the Multiplicative Inverse of the Determinant by the Adjugate Matrix:

To get the inverse key matrix, we now multiply the inverse determinant (that was 7 in our case) from step 1 by each of the elements of the adjugate matrix from step 2. Then we take each of these answers modulo 26.

$$7 \times \begin{pmatrix}11 & 18\\ 15 & 7\end{pmatrix} = \begin{pmatrix}77 & 126\\ 105 & 49\end{pmatrix} = \begin{pmatrix}25 & 22\\ 1 & 23\end{pmatrix} \ mod\ 26$$

$$if\ K = \begin{pmatrix}7 & 8\\ 11 & 11\end{pmatrix}, then\ K^{-1} = \begin{pmatrix}25 & 22\\ 1 & 23\end{pmatrix}$$

Figure13:Multiply the Multiplicative Inverse of the Determinant by the Adjugate Matrix

Now we have the inverse key matrix, we have to convert the ciphertext into column vectors and multiply the inverse matrix by each column vector in turn, take the results modulo 26 and convert these back into letters to get the plaintext.

$$\begin{pmatrix}25 & 22\\ 1 & 23\end{pmatrix}\begin{pmatrix}A\\ D\end{pmatrix} = \begin{pmatrix}25 & 22\\ 1 & 23\end{pmatrix}\begin{pmatrix}0\\ 3\end{pmatrix}$$

$$= \begin{pmatrix}25 \times 0 + 22 \times 3\\ 1 \times 0 + 23 \times 3\end{pmatrix}$$

$$= \begin{pmatrix}66\\ 69\end{pmatrix}$$

$$= \begin{pmatrix}14\\ 17\end{pmatrix}\ mod\ 26$$

$$= \begin{pmatrix}o\\ r\end{pmatrix}$$

Figure14:Decryption of the first digraph:

$$\begin{pmatrix}25 & 22\\ 1 & 23\end{pmatrix}\begin{pmatrix}J\\ T\end{pmatrix} = \begin{pmatrix}25 & 22\\ 1 & 23\end{pmatrix}\begin{pmatrix}9\\ 19\end{pmatrix}$$
$$= \begin{pmatrix}25\times 9 + 22\times 19\\ 1\times 9 + 23\times 19\end{pmatrix}$$
$$= \begin{pmatrix}643\\ 446\end{pmatrix}$$
$$= \begin{pmatrix}19\\ 4\end{pmatrix} \ mod\ 26$$
$$= \begin{pmatrix}t\\ e\end{pmatrix}$$

Figure15:Decryption of the first digraph:

$$\begin{pmatrix}25 & 22\\ 1 & 23\end{pmatrix}\begin{pmatrix}F\\ T\end{pmatrix} = \begin{pmatrix}25 & 22\\ 1 & 23\end{pmatrix}\begin{pmatrix}5\\ 19\end{pmatrix}$$
$$= \begin{pmatrix}25\times 5 + 22\times 19\\ 1\times 5 + 23\times 19\end{pmatrix}$$
$$= \begin{pmatrix}543\\ 442\end{pmatrix}$$
$$= \begin{pmatrix}23\\ 0\end{pmatrix} \ mod\ 26$$
$$= \begin{pmatrix}x\\ a\end{pmatrix}$$

Figure16:Decryption of the first digraph:

Finally, we get back our plaintext of "ortexa".

3. Description of your network setting and developed application

To associate two PCs to one another, socket programming was used in this project. The socket is an end-point of the bidirectional communication link between server and client that runs on the same network.The server socket program has a port number that waits for the client's incoming request. Server listens for the client's request, and when it gets a request from client socket, it sends a respond to client.After a request is initiated, an infinite loop is initiated to monitor the request from client's side.When the server socket accepts a request from the client's side, it reads the data from NetworkStream and also it writes the response to NetworkStream.

In this project,We used UDP Sockets and the steps are shown as follows

1.First we create server to the bind port

```
//create server to bind the port
public void Server(string address, int port)
{
    _socket.SetSocketOption(SocketOptionLevel.IP, SocketOptionName.ReuseAddress, true);
    _socket.Bind(new IPEndPoint(IPAddress.Parse(address), port));
    Receive();
}
```

Figure17:Create server to the bind port

2.We create the client

```
//create the client to
public void Client(string address, int port)
{
    _socket.Connect(IPAddress.Parse(address), port);
    Receive();
}
```

Figure18:Create the client

3.Then send the message via UDP

```
// send message via udp
public void Send(string text)
{
    byte[] data = Encoding.ASCII.GetBytes(text);
    _socket.BeginSend(data, 0, data.Length, SocketFlags.None, (ar) =>
    {
        State so = (State)ar.AsyncState;
        int bytes = _socket.EndSend(ar);
        Console.WriteLine("SEND: {0}, {1}", bytes, text);
    }, state);
}
```

Figure19:Send the message via UDP

4- Recieve the message

```
private void Receive()
{
    _socket.BeginReceiveFrom(state.buffer, 0, bufSize, SocketFlags.None, ref epFrom, recv = (ar) =>
    {
        State so = (State)ar.AsyncState;
        int bytes = _socket.EndReceiveFrom(ar, ref epFrom);
        string rcv_str = Encoding.ASCII.GetString(so.buffer, 0, bytes);
        _socket.BeginReceiveFrom(so.buffer, 0, bufSize, SocketFlags.None, ref epFrom, recv, so);

        //Console.WriteLine("RECV: {0}: {1}, {2}", epFrom.ToString(), bytes, rcv_str);

        //send the message to the form
        this.m_parent.setReceivedText(rcv_str);
    }, state);
}
```

Figure20:Recieve the message

This server socket program was done as a console based application in C# language by using visual studio. When the code is run, the server console appears which is given below in figure 21

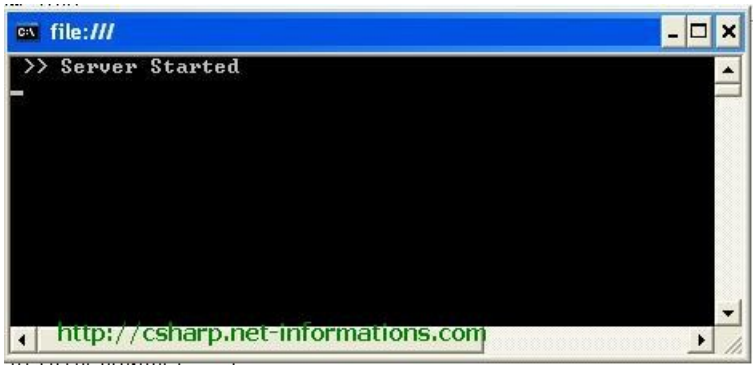

Figure21:Server Started

The client socket program have to know the IP address(hostname) of the computer that server socket resides and the port number assigned for listening for client's request. When the client socket program execute, it will establish a connection to a server program and send request to the server syncing receive response from the server. When the client socket program stars, it connects to the server socket program and begin to read data from NetworkStream and also, write to the NetworkStream. When the client program starts, server gives a message "client startes" and when the button at the bottom of the client window is pressed, client can send a message to the server and receive responce from the server.

The client socket program is the second part of the server socket program. In this project, the client is connected to the same port number.The client socket program was done as a windows based application. When the code is run, the client's window which is given below appears.Now, server console and client window is connected to each other and they can communicate with each other but because of the undefined reasons, the data transmission between the server and client does not work so it was decided to show the

data transmission between the server and client on one application without connecting server and client via network.The picture of this application form1 is given below in figure 22

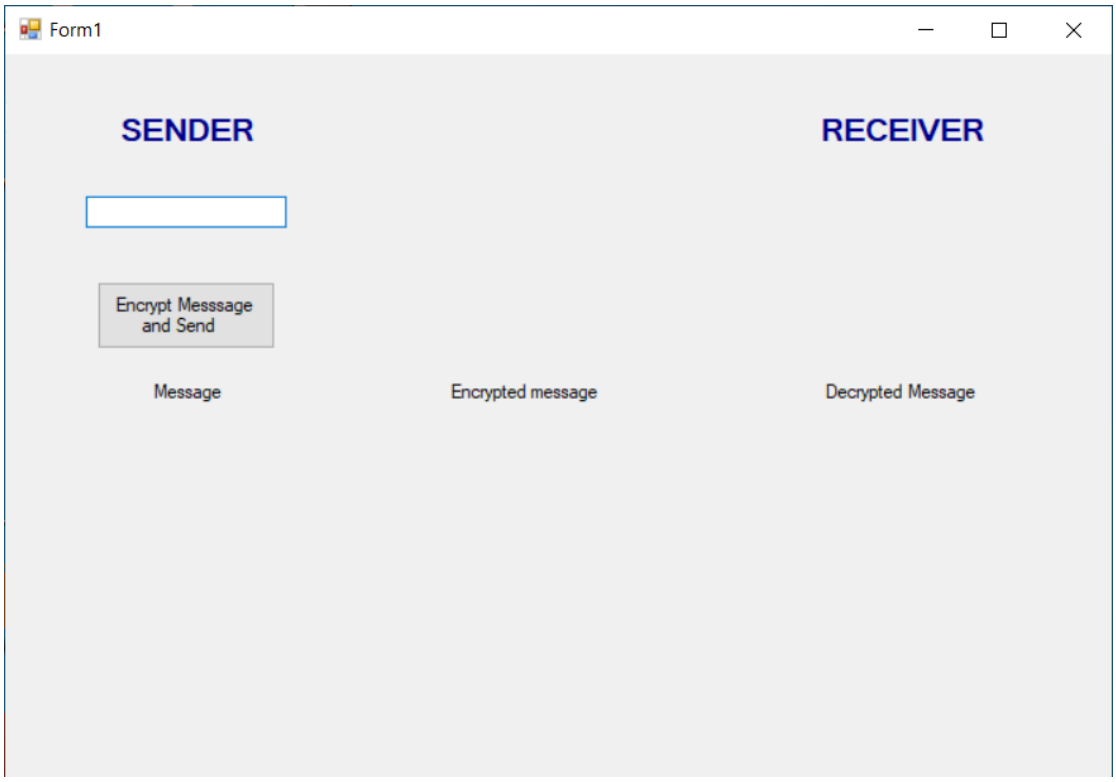

Figure22:Application Form1

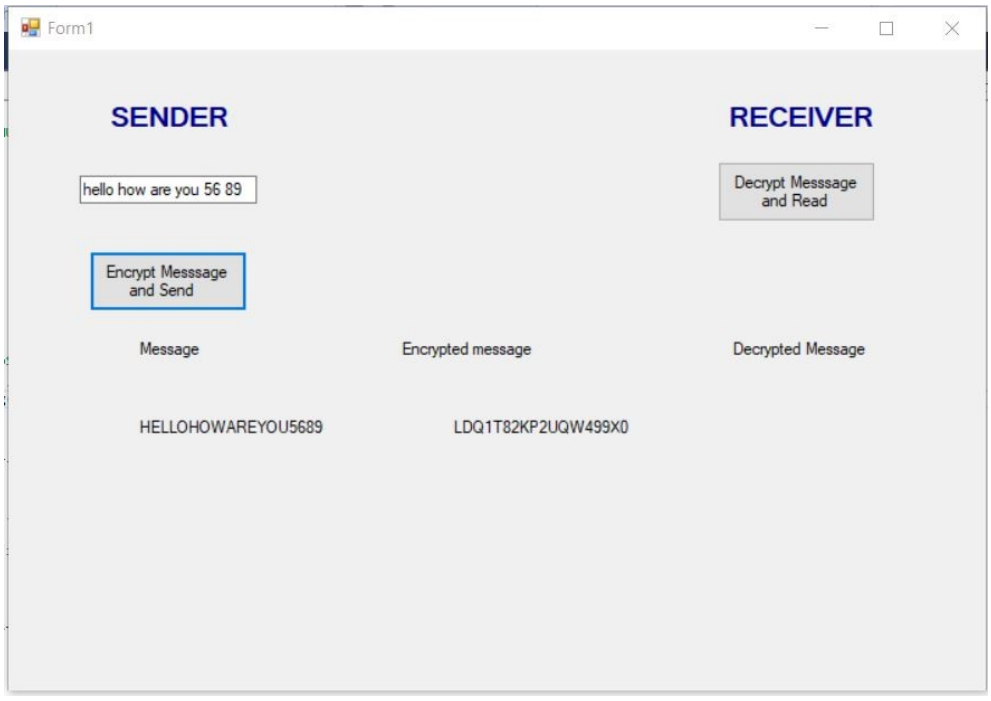

Figure23: Encyption of the plaintext

4. Experiments results and discussion

Encryption/Decryption In this case, we use C# object oriented language on back-end side by using form application. We will show some details of back-end part to see how we encrypt/decrypt plaintext and cipher text

4.1.Encryption :

When we click “Encrypt Message and Send” button as shown on Figure … plaintext will be encrypted by using hill cipher which n value is 2. After clicking the button the “encryption function” is triggered . In this part plaintext is taken from the textbox and dividing parts as 2 by 2. Each parts is converted to its alphabetic numbers and then each of them are converted to de cipher text by using hill cipher features. Only sender and receiver knows the key value and all calculations are depending on this key value which is assigned in the coding part. After calculations the result string value is assigned on the label which is under the “Message” block. The source code of the encryption function is shown below:

```
// Following function encrypts the message
void encrypt(int[,] cipherMatrix, int[,] messageVector)
{
    int x, i, j;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 1; j++)
        {
            cipherMatrix[i, j] = 0;

            for (x = 0; x < 2; x++)
            {
                cipherMatrix[i, j] += enc_mat[i, x] *
                                      messageVector[x, j];
            }

            cipherMatrix[i, j]  = validate_number(cipherMatrix[i, j]);
        }
    }
}
```

Figure24: Encryption Code

4.2.Decryption :

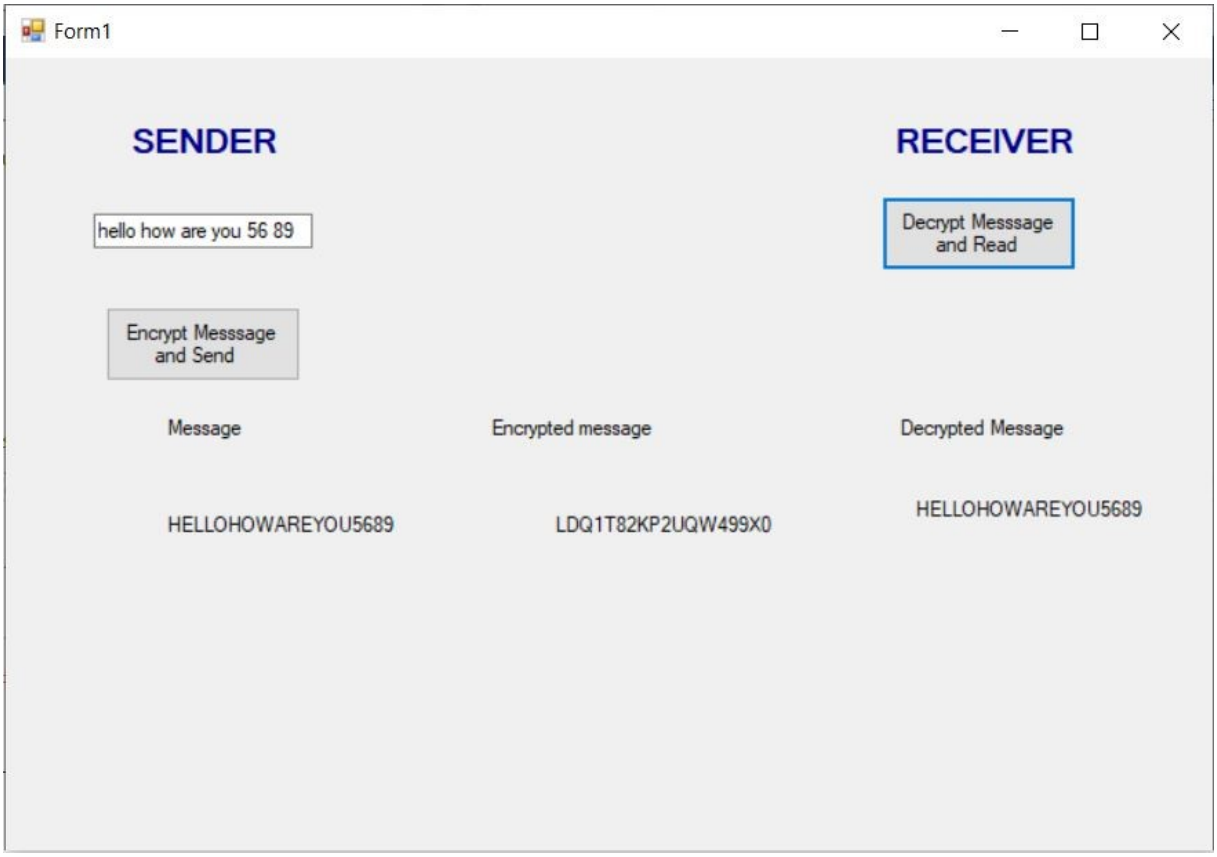

Figure25: Encyption of the plaintext

When we click “Decrypt Message and Read” button as shown on Figure … cipher text will be decrypted by using hill cipher which n value is 2. After clicking the button the “decryption function” is triggered . In this part cipher text is taken from the sender and dividing parts as 2 by 2. Each parts is converted to its alphabetic numbers and then each of them are converted to de plaintext by using hill cipher features. Only sender and receiver knows the key value and all calculations are depending on this key value which is assigned in the coding part. After calculations the result string value is assigned on the label which is under the “Decrypted Message” block. The source code of the encryption function is shown below:

```
// Following function encrypts the message
void decrypt(int[,] cipherMatrix, int[,] messageVector)
{
    int x, i, j;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 1; j++)
        {
            cipherMatrix[i, j] = 0;

            for (x = 0; x < 2; x++)
            {
                cipherMatrix[i, j] += dec_mat[i, x] *
                                      messageVector[x, j];
            }

            cipherMatrix[i, j] = validate_number(cipherMatrix[i, j]);
        }
    }
}
```

Figure25:Decyption Code

5.Conclusion

In conclusion, in this term project, encryption and decryption process have done successfully in this project via hill cipher algorithm n=2 wtih using C# language.To associate two PCs to one another, socket programming was used in this project.Cipher key matrix checking, decipher key matrix calculation, encryption/decryption for both cases of local computers with two input methods of files and dynamic texts are all implemented and tested successfully.

6.References

Alexcander Chefranov (2020),"CLASSICAL ENCRYPTION TECHNIQUES ".lecture note.

http://csharp.net-informations.com/communications/csharp-socket-programming.htm

https://japp.io/cryptography/hill-cipher-algorithm-program-in-c-c/

https://crypto.interactive-maths.com/hill-cipher.html#3x3encypt