R.V. College of Engineering
(Autonomous Institution affiliated to VTU)
Department of Information Science and Engineering
Bangalore – 560059

MESSAGE ENCRYPTION USING MODIFIED PLAYFAIR CIPHER WITH 6X6 MATRIX

PROJECT REPORT

Submitted by

Haripriya Ramesh	**USN: 1RV14IS012**
Khadija Zavery	**USN: 1RV14IS015**

Under the guidance
of

Prof. B K Srinivas
Assistant Professor,
Department of ISE, R.V.C.E.,
Bangalore - 560059

&

Prof. Geetha V
Assistant Professor,
Department of ISE, R.V.C.E.,
Bangalore - 560059

in partial fulfillment for completion
of
Computer Networks & Security Laboratory

Bachelor of Engineering
Department of Information Science & Engineering

R.V. COLLEGE OF ENGINEERING
BANGALORE - 560059
(Autonomous Institution Affiliated to VTU, Belgaum)

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

Certified that the project work entitled " Message encryption using Modified Playfair Cipher with 6x6 matrix" carried out by Ms. Haripriya Ramesh, USN: 1RV14IS012 and Ms. Khadija Zavery, USN: 1RV14IS015 who are bonafide student of R.V College of Engineering, Bangalore, in partial fulfillment for the completion of **Computer Networks & Security Laboratory (12IS63),** the requirement for the award of degree of **Bachelor of Engineering** in **Department of Information Science & Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2016-17**. It is certified that all corrections/suggestions indicated for the internal Assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed by the institution for the said degree.

Signature of	**Signature of**	**Signature of**
Lab Incharge-1	**Lab Incharge-2**	**HOD**
Prof. B K Srinivas	**Prof. Geetha V**	**Dr. N K Cauvery**

Name of Examiners

Semester End Examination **Signature with date**

1

2

R.VCOLLEGE OF ENGINEERING
BANGALORE - 560059
(Autonomous Institution, Affiliated to VTU, Belgaum)

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

DECLARATION

We, Ms. Haripriya Ramesh, USN: 1RV14IS012 and Ms. Khadija Zavery, USN: 1RV14IS015 student of sixth semester B.E., **Department of Information Science & Engineering.** , hereby declare that the project titled "**Message encryption using Modified Playfair Cipher with 6x6 matrix**" has been carried out and submitted in partial fulfillment for the completion of **Computer Networks and Security Laboratory (12IS63),** the requirement for the award of degree of **Bachelor of Engineering** in **Department of Information Science & Engineering.**

Place: Bangalore	**Name**	**Signature**
Date:	**1.**	
	2.	

ACKNOWLEDGEMENTS

The euphoria and satisfaction of the completion of this project will be incomplete without thanking the persons responsible for this venture. We acknowledge RVCE(Autonomous under VTU) for providing an opportunity to create a mini-project in the 6th semester. We would personally like to thank our lab incharges, Assistant Prof**. B K Srinivas** and Assistant Prof. **Geetha V** for providing timely assistance & guidance at the appropriate time.We are indebted to the co-operation given by lab administrators and lab assistants, who have played a major role in bringing out the mini-project in the present form.

We express our heartfelt thanks to **Dr.N K Cauvery**, Head of Department of **Information Science & Engineering**, for providing us with the required facilities and information which helped us in satisfactory completion of the project work.

ABSTRACT

In order to provide security to the information that is to be transmitted from sender to receiver we have several methods. The well known and highly used method for protecting the data during its transmission across the network is encryption. The conventional encryption system consists of plain text, encryption algorithm, secret key, cipher text and decryption algorithm.

We need a strong encryption algorithm in order to encrypt the plain text into cipher text. The sender and receiver must have obtained the secret key in a secure fashion and must keep the key secure. The cryptographic systems are generally classified according to the type of operations used for transforming plain text to cipher text, the number of keys used and the way in which the plain text is processed

Among all of the existing cryptographic systems play fair cipher got importance. The play fair cipher algorithm is based on the use of 5 X 5 matrix of letters constructed using a keyword. The matrix is constructed by filling the letters of keyword from left to right and top to bottom and filling in the remainder of matrix with the remaining alphabetic order. This algorithm can only allow the plain text containing alphabets, for this we have implemented an enhancement to the existing using a 6 X 6 matrix and that can allow the plain text containing of alphanumeric values.

The user may encrypt digits along with the characters through the Extended PlayFair cipher, because the extended algorithm using 6X6 matrix, which can allow even digits into it. This extended play fair algorithm is based on the use of a 6 X 6 matrix of letters constructed using a keyword. The matrix is constructed by filling in the letters of the keyword from left to right and from top to bottom, and the filling in the remainder of the matrix with the remaining letters in alphabetic order and digits in ascending order form 0 to 9. Through this extended PlayFair algorithm we can encrypt the plain text containing alphanumeric values very efficiently. The user does not face any ambiguity at decipherment, because the characters ‘I’ and ‘J’ are placed in separate cells of the matrix.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 About

The art and science of concealing the messages to introduce secrecy in information security is recognized as cryptography. The word ‘cryptography’ was coined by combining two Greek words, ‘Krypto’ meaning hidden and ‘graphene’ meaning writing. We work on historical cryptosystems which are systems based on symmetric key encryption scheme.The only security service these systems provide is confidentiality of information. Unlike modern systems which are digital and treat data as binary numbers, the earlier systems worked on alphabets as basic element. There are many ciphers in this category but we are mainly interested in polyalphabetic ciphers. Polyalphabetic Cipher is a substitution cipher in which the cipher alphabet for the plain alphabet may be different at different places during the encryption process. An example is the playfair cipher whose modification we have worked worked on. Encryption and decryption using this cipher has been performed on two different systems respectively by sharing data using TCP protocol.

1.2 Motivation

There were two forces that motivated us to pursue this topic as a project - The first being that we were given the topic by our lab-in-charge teacher, and the second being that we had the topic as part of the curriculum, so we could relate to what we were trying being taught in the class.

1.3. Problem statement and drawbacks with the existing system

The task at hand is to implement the modification of playfair cipher and to perform the encryption and decryption on two different systems. The performance of the generic playfair cipher is not the best as it doesn’t provide one to one mapping between the plaintext and the ciphertext, which makes the encryption and decryption difficult and ambiguous. Therefore we aim to implement modifications of this algorithm in the form of a 6x6 matrix which makes the text more unreadable when additional symbols appear in the resulting ciphertext. Also this method can be extended to encrypt and decrypt the messages of any language including numbers by taking a proper size matrix.

1.4. Objectives

The objective is to tackle the problem at hand - to implement a modified version of the popular playfair cipher and to perform the encryption and decryption on two different systems. Working of the TCP protocol needs to be studied to attain this objective.

1.5 Methodology

The methodology involved in making this cipher is explained in detail in the design section. We have used C as the programming language and have made use of many inbuilt library functions in order to implement the TCP functionality. The encryption and decryption of the playfair cipher has been implemented using the proposed algorithms.

1.6 Organisation of the Report

The rest of the report is organized as follows: Chapter 2 briefly gives the requirements specification details. It describes the perspective of the project, the functions the product is going to perform, and the characteristics of the users of the product. It specifies the functional requirements for the product development. It also specifies the hardware and software required for the development and proper functioning of the product.

Chapter 3 shows the high level design of the project. It explains the considerations for the design taken which involves the dependencies and development methods. It considers the architectural strategies for programming language, user interface paradigm and error detection and recovery. It shows the first 3 levels of Data Flow Diagrams (DFDs) of the project. The higher level DFD is the enhanced version of the lower level DFD. It also shows the detailed design of the project. Detailed design includes the overall system architecture of the project.

Chapter 4 describes the implementation of our project. It explains the programming language selection, the platform selection, the method selection and the class declaration selection

Chapter 5 describes the various testing done to check the integrity of the application. It explains the levels of testing, the various testing done on the modules and the different menus in the application.

Chapter 6 describes the results of the project and what has been accomplished. It also analyses the obtained result to know about better efficient method of implementing project.

Chapter 7 is the conclusion and future enhancement part of the project. It briefly tells what the project is doing and what has been accomplished. Future enhancements are also listed in this section.

Apart from the above, the references are listed out in bibliography. The appendix provides details about abbreviations and the coding part.

Chapter 2

Requirements Specification

2.1 Overall description

Software Requirement Specification (SRS) is a complete description of the behaviour of the system to be developed. It includes the system requirements which consists of functional and non-functional requirements, user requirements, domain requirements, hardware and software requirements. The functional requirements include what software should do and non-functional requirements are related to performance and scalability. This section of SRS contain all the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements and it also help to design their test cases to verify them to check, whether system satisfies those requirements or not.

2.2 System requirements

System Requirements are used efficiently, all computer software needs to contain certain hardware components or other software resources to be present on a computer. The system requirements giving the requirements to be met in the design of a system or subsystem. These requirements represent the ideal situation in which to run the software.

2.3 Hardware requirements

This section defines any hardware interfaces that are to be supported by the software, including logical structure, expected behaviour, etc. The hardware requirements are as follows:

Processor: Intel Core 2 Duo or above

In-built RAM: 1.00GB or above

System type: Windows XP, Windows Vista, Windows 7 (32bit or 64bit), Windows 8, Windows 10, Linux, Ubuntu

Processor: Intel Core 2 Duo or above

In-built RAM: 2.00GB or above

System type: Windows (XP and above)

2.4 Functional requirements

Functional requirements directly support the user requirements by describing the processing of the information as inputs or outputs. Most of the requirements definition focuses mainly on functional requirements, which are based upon the expected functioning of the product system to be created. The most important functional requirement of a cipher is to encrypt a message or text into cipher text. Another important functional requirement is to decrypt the text at the receiver's site and perform all this in a secure manner.

2.5 Non-functional requirements

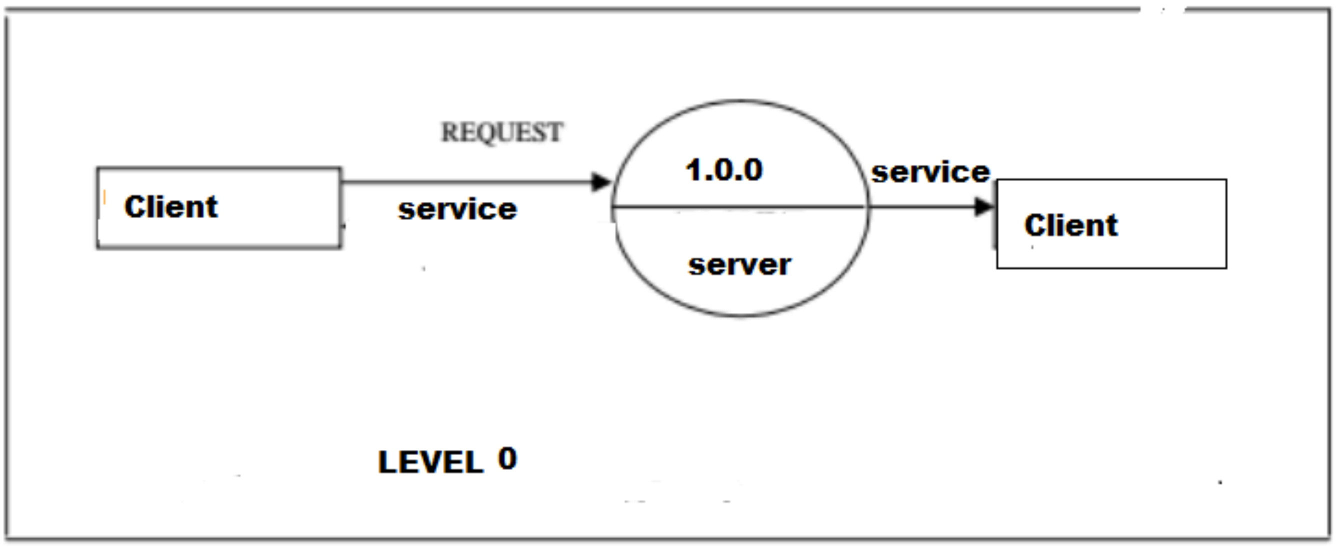
Non-functional requirements cover all the remaining requirements which are not covered by the functional requirements. They specify criteria that judge the operation of a system, rather than specific behaviours. Compatibility is an important non-functional requirement. The implementation of cipher algorithm should be compatible with other end systems and also support the basic text and file formats.

Chapter 3
Design Goals

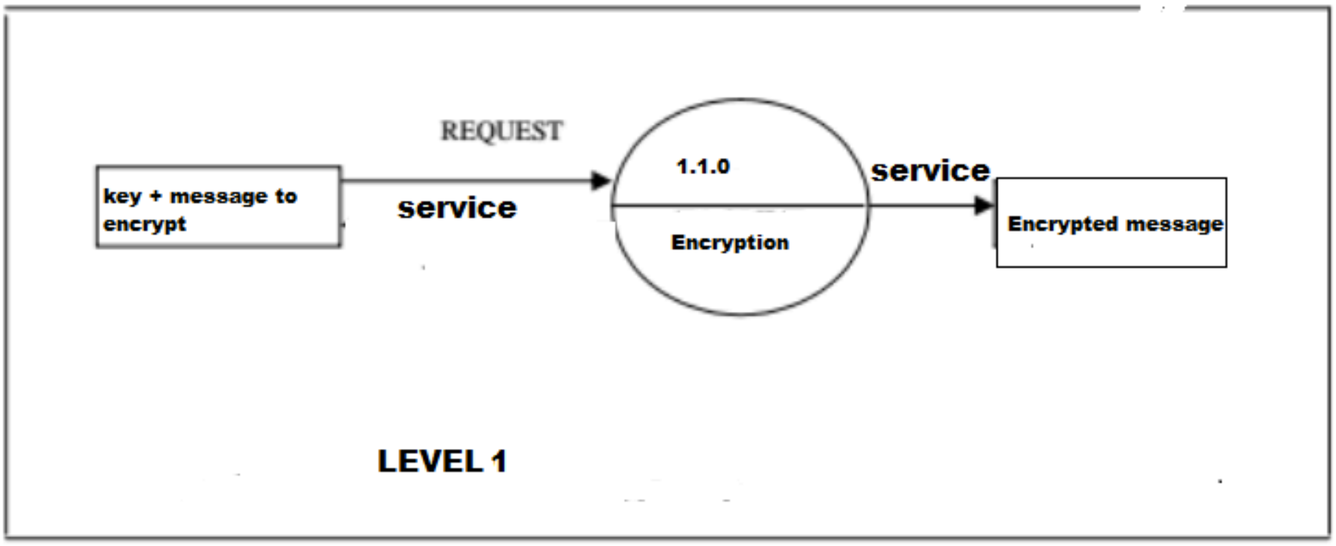
The high level design consists of the description of the development method used and data flow diagram of the Playfair Cipher and how it is implemented using the client server model of TCP. In the High level design the technical architect of the project will study the proposed application's functional and non-functional requirements and design overall solution architecture of the application which can handle those needs. A data flow diagram helps user to visualize how the system will operate and what would be the processing components of the system and what data they would process.

3.1 Data Flow Diagrams of Playfair Cipher Implementation

A Data Flow Diagram (DFD) is a graphical representation of the flow of data through an information system. Data Flow models are used to show how data flows through a sequence of processing steps. The data transformed at each step before moving on to the next stage. These processing steps or transformations are program functions when Data Flow diagrams are used to document a software design. DFD diagram is composed of four elements, which are process, data flow, external entity and data store.



Level 0 Data Flow Diagram



Level 1 Data Flow Diagram

Chapter 4

Implementation

Implementation of any software is always proceeding by import decisions regarding selection of the platform, the language used, etc. These decisions are often influenced by several factors such as the real environment in which the system works the speed that is required, the security concerns, other implementation of specific details etc.

4.1 Programming Language Selection

It was decided that using C programming language would prove to be the most fruitful, as the implementation was basically using C dictionaries and inbuilt libraries. the length of the code for the implementation was small when compared to implementing it using another language such as C++.

4.2 Platform selection

In our case of implementation, we were able to obtain optimal results on compiling our program on Linux operating system. Any operating system that supports C programming can be used.

4.3 Code snippets and methodology

Following are a few code snippets that show how the algorithm functions followed by the table of existing and modified version of playfair cipher.

Encrypt function :

```
void encrypt(int l)
{
            FILE *fp1;
           fp1 = fopen("encrypt.txt", "w");
    int i,ax,ay,bx,by;
    struct index f;
    for(i=0;i<l;i=i+2)
    {
        f = findIndex(c[i]);
        ax = f.x;
        ay = f.y;
        f = findIndex(c[i+1]);
        bx = f.x;
        by = f.y;
        if(ax == bx)
        {
            ans[i] = pc[ax][(ay+1)%6];
            ans[i+1] = pc[ax][(by+1)%6];
        }
        else if(ay==by)
        {
            ans[i] = pc[(ax+1)%6][by];
            ans[i+1] = pc[(bx+1)%6][ay];
        }
        else
        {
            ans[i] = pc[ax][by];
            ans[i+1] = pc[bx][ay];
        }
        printf("%c %c ",ans[i],ans[i+1]);

        fprintf(fp1,"%c",ans[i]);

        fprintf(fp1,"%c",ans[i+1]);

    }

}
```

Decrypt function:

```
void decrypt(int l)
{

            FILE *fp1;
           fp1 = fopen("decrypt.txt", "w");
    int i,ax,ay,bx,by;
    int x = 0;
    struct index f;
    for(i=0;i<l;i=i+2)
    {
        f = findIndex(ans[i]);
        ax = f.x;
        ay = f.y;
        f = findIndex(ans[i+1]);
```

```
bx = f.x;
by = f.y;
if(ax == bx)
{
    if(by-1<0)
    {
        decy[i+1] = pc[ax][5];
    }
    else
    {
        decy[i+1] = pc[ax][(by-1)%6];
    }
    if(ay-1<0)
    {
        decy[i] = pc[ax][5];
    }
    else
    {
        decy[i] = pc[ax][(ay-1)%6];
    }

}
else if(ay==by)
{
    if(bx-1<0)
    {
        decy[i+1] = pc[5][by];
    }
    else
    {
        decy[i+1] = pc[(bx-1)%6][by];
    }
    if(ax-1<0)
    {
        decy[i] = pc[5][ay];
    }
    else
    {
        decy[i] = pc[(ax-1)%6][ay];
```

```
                }
            }
            else
            {
                decy[i] = pc[ax][by];
                decy[i+1] = pc[bx][ay];
            }
           printf("%c %c ",decy[i],decy[i+1]);
             fprintf(fp1,"%c",decy[i]);

            fprintf(fp1,"%c",decy[i+1]);
                 decrypted[x] = decy[i];
                 x++;
                 decrypted[x] = decy[i+1];
                 x++;

        }
}
```

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Fig 1 : Existing playfair cipher 5X5 matrix for the key ‘monarchy’

M	O	N	A	R	C
H	Y	B	D	E	F
G	I	J	K	L	P
Q	S	T	U	V	W
X	Z	0	1	2	3
4	5	6	7	8	9

Fig 2 : Modified playfair cipher using 6X6 matrix with key ‘monarchy’

Chapter 5

Testing

5.1 Testing Process

Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, complies with the standards, that it was designed to. Testing process involves building of test cases, against which the product has to be tested. In some cases, one drives the test cases from the requirements of the product/software, which is to be developed.

5.2 Levels of Testing

Different levels of testing are used in the testing process, each level of testing aims to test different aspects of the system. The basic levels are unit testing, integration testing, system testing and acceptance testing.

5.3 Testing Environment

The playfair cipher is tested on following parameters:

● Speed to encrypt/decrypt data blocks of various sizes.

● Test the avalanche effect

● Balanced output

● Resist to most known attacks - cipher text only, known and chosen plain text attacks, differential and exhaustive attacks.

5.4 Unit Testing of Main Modules

Unit testing is the process of testing individual components in the system. This is a defect testing process so its goal is to expose faults in these components. Individual functions or methods are the sample type of component and tests are the set of calls to these routines with different parameters. Here different modules are tested independently and their functionality and usability is checked. During unit testing, it is necessary to simulate the interaction of the module being tested with the other parts of the system. This can be done in two ways: Bottom-up testing and Top-down testing.

5.5 Integration Testing of Modules

After testing each of the module explained in chapter 4, they must be integrated and tested again. That is, all the modules present in Level 0 are combined and tested and this is repeated for all the Levels. Any error present must be corrected and the system on which we are working should also be tested. The system selected must be as described under the system requirement specification heading.

Chapter 6

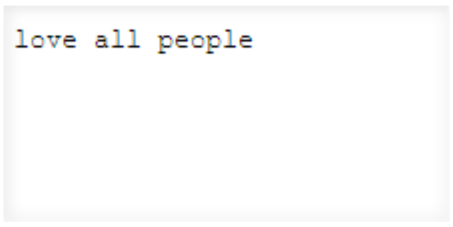
Results, Discussion and Inference

6.1 Introduction

The term experiment is defined as the systematic procedure carried out under controlled conditions in order to discover an unknown effect, to test or establish a hypothesis, or to illustrate a known effect. When analysing a process, experiments are often used to evaluate which process inputs have a significant impact on the process output, and what the target level of those inputs

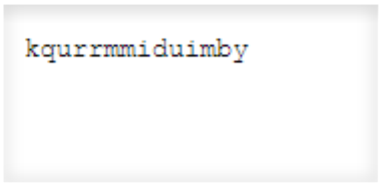
6.2 Results Obtained

The code performs all basic functions of encrypting a message,transferring it safely across a channel and then decrypting the message. Also, it is easy to use, and the output is displayed in an organised manner, making it easier for us to debug any issues along the way. The input and output is taken and obtained through files and they key is exchanged securely as well. It was also noticed that our program worked fast enough for even very large message sizes. As the text files are sent even large data can be encrypted and decrypted by sending the files and reading them through the encryption/decryption code.



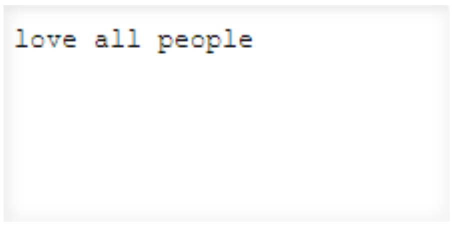
```
love all people
```

Fig 3: plain text to be encrypted stored in text file



```
kqurrmmiduimby
```

Fig 4: encrypted text after reading from file and encrypting



```
love all people
```

Fig 5: decrypted text after reading from file and decrypting on client side

6.3 Summary

This chapter gives the outcome of the experimental work carried out and results. It also explains about the evaluation metrics, experimental data set and it also talks about the performance analysis of the project.

Chapter 7
Conclusion

The modified 6X6 playfair cipher implemented works as expected on basic client server architecture. 6 X 6 Playfair cipher is the multiple letter encryption cipher, which encrypts digraphs of plaintext into corresponding cipher text digraphs. For that purpose it uses a 6 X 6 matrix to store alphabets and numerals. These alphabets and numerals are arranged in 6 X 6 matrix based on secret key. Running the program by specifying the text file's path to be encrypted/decrypted gives us the prompt output on respective client/server screens.

7.1 Limitations of the Project

Some of the limitations of the project are:

- As it uses client server architecture, we need the IP addresses of the systems before sending the message.
- The sending of the secret key needs to be more secure as this is a symmetric encryption method, the loss of key will result in compromise of the messages.
- This implementation does not support the use of special characters in the message to be encrypted and only alpha-numeric characters are supported.

7.2 Future Enhancement

- Integrating performance enhancing features like using multicore programming constructs to speed up the encryption and decryption process.
- Moving towards platform independence and program portability by removing platform specific components.
- To implement a more secure key exchange algorithm or a enhanced key distribution scheme Modifying the matrix to include various special characters which can encrypt/decrypt message containing them.

REFERENCES

[1] Andrew S. Tanenbaum, Pearson Publication, Fourth edition, Computer Networks.

[2] Professor Guevara Noubir, Fundamentals of Cryptography: Algorithms, and Security Services.

[3] Wikipedia, the free encyclopedia.

[4] Jerry li, MD5 Message Digest Algorithm. San Jose University, Computer Science department.

[5] Ruth Betcher, Secure Hashing Algorithm.

[6] MD5 is faster than SHA-1.Journal Of Omnifarious-Myth.

[7] William Stalling7, Fourth Edition, Cryptography and Network Security (Various Hash Algorithms).

[8] http://stackoverflow.com/questions/2948156/algorithm-complexity-security-md5-or-sha1.

[9] Addam Schroll, ITNS and CERIAS CISSP Luncheon Series: Cryptography.

[10]http://science.opposingviews.com/advantages-disadvantages-symmetric-key-encryption-2609.html

APPENDIX

Playenc.c (encryption code file)

```
#include<string.h>
#include<stdio.h>
char pc[6][6],c[1000],ans[1000],decy[1000],decrypted[1000];
struct index
{
    int x,y;
};

void table(char k[])
{
    int l=0,i=0,j=0,v[10]={0},av[26]={0};
    while(l < strlen(k))
    {
        if(k[l] >47 && k[l]<58 && v[k[l] - 48] == 0)
        {
            pc[i][j] = k[l];
            v[k[l] - 48] = 1;
            j++;
        }
        else if(k[l] >96 && k[l]<123 && av[k[l] - 97] == 0)
        {
            pc[i][j] = k[l];
            av[k[l] - 97] = 1;
            j++;


        }
        if(j>5)
        {
            j=0;
            i++;
        }
        l++;
    }
    for(l=0;l<10;l++)
    {
        if(v[l]==0)
        {
            pc[i][j] = l + 48;
            j++;
        }
        if(j>5)
        {
            j=0;
            i++;
        }
    }
    for(l=0;l<26;l++)
    {
        if(av[l]==0)
        {
            pc[i][j] = l + 97;
            j++;
        }
        if(j>5)
        {
            j=0;
            i++;
        }
    }
}

int convert(char s[])
{
    int i,j = 0,f=0;
    //int d = strlen(s);
    //printf("%d",d);
    for(i=0;i<strlen(s);i++)
    {
        if( (s[i]>47 && s[i]<58) || (s[i]>96 && s[i]<123) )
        {
            if((i+f)%2!=0 && c[j-1]==s[i])
            {
                c[j] = 'x';
            }
            else
            {
```

```
                c[j] = s[i];
            }
            printf("%c",c[j]);
            j++;
        }
        else
        {
            f++;
        }
    }
    if(j%2!=0)
    {
        c[j] = 'x';
        printf("%c",c[j]);
        j++;
    }
    return j;
}

struct index findIndex(char ch)
{
    int i,j;
    struct index f;
    for(i=0;i<6;i++)
    {
        for(j=0;j<6;j++)
        {
            if(pc[i][j]==ch)
            {
                f.x = i;
                f.y = j;
                i=6;
                j=6;
            }
        }
    }
    return f;
}
```

```
void encrypt(int l)
{
            FILE *fp1;
           fp1 = fopen("encrypt.txt", "w");
    int i,ax,ay,bx,by;
    struct index f;
    for(i=0;i<l;i=i+2)
    {
        f = findIndex(c[i]);
        ax = f.x;
        ay = f.y;
        f = findIndex(c[i+1]);
        bx = f.x;
        by = f.y;
        if(ax == bx)
        {
            ans[i] = pc[ax][(ay+1)%6];
            ans[i+1] = pc[ax][(by+1)%6];
        }
        else if(ay==by)
        {
            ans[i] = pc[(ax+1)%6][by];
            ans[i+1] = pc[(bx+1)%6][ay];
        }
        else
        {
            ans[i] = pc[ax][by];
            ans[i+1] = pc[bx][ay];
        }
        printf("%c %c ",ans[i],ans[i+1]);

        fprintf(fp1,"%c",ans[i]);

        fprintf(fp1,"%c",ans[i+1]);

    }
}
```

```
int main()
{
    char s[100],k[100],string[100],ch; int z;
          FILE * fp;
            FILE *fp1;
          fp=fopen("sk.txt","r");
    printf("Enter the Key : \n");
    scanf("%s",k);
    z=0;
          while(1) {
                    ch=fgetc(fp);
                    if(ch!=EOF) {
                              s[z++]=ch;
                    }
                    else if(ch==EOF)
                      break;
          }
          s[z]='\0';
          puts(s);
          //int h = strlen(s);
          //printf("%d",h);
    int i,j,f=0;
    printf("\n6x6 Playfair Matrix : ");
    table(k);
    for(i=0;i<6;i++)
    {
        for(j=0;j<6;j++)
        {
            printf("%c ",pc[i][j]);
        }
        printf("\n");
    }
    int l;
    printf("\nOrignal Message to Encrypt : ");
    l = convert(s);
    printf("\n");
    printf("\nEncrypted Message : ");
    encrypt(l);
    printf("\n");
    return 0;
}
```

Playdec.c (decryption code file)

```
#include<string.h>
#include<stdio.h>
char pc[6][6],c[1000],ans[1000],decy[1000],decrypted[1000];
struct index
{
    int x,y;
};

void table(char k[])
{
    int l=0,i=0,j=0,v[10]={0},av[26]={0};
    while(l < strlen(k))
    {
        if(k[l] >47 && k[l]<58 && v[k[l] - 48] == 0)
        {
            pc[i][j] = k[l];
            v[k[l] - 48] = 1;
            j++;
        }
        else if(k[l] >96 && k[l]<123 && av[k[l] - 97] == 0)
        {
            pc[i][j] = k[l];
            av[k[l] - 97] = 1;
            j++;


        }
        if(j>5)
        {
            j=0;
            i++;
        }
        l++;
    }
    for(l=0;l<10;l++)
    {
        if(v[l]==0)
        {
            pc[i][j] = l + 48;
            j++;
        }
        if(j>5)
        {
            j=0;
            i++;
        }
    }
    for(l=0;l<26;l++)
    {
        if(av[l]==0)
        {
            pc[i][j] = l + 97;
            j++;
        }
        if(j>5)
        {
            j=0;
            i++;
        }
    }
}

int convert(char s[])
{
    int i,j = 0,f=0;
    //int d = strlen(s);
    //printf("%d",d);
    for(i=0;i<strlen(s);i++)
    {
        if( (s[i]>47 && s[i]<58) || (s[i]>96 && s[i]<123) )
        {
            if((i+f)%2!=0 && c[j-1]==s[i])
            {
                c[j] = 'x';
            }
            else
            {
                c[j] = s[i];
            }
            printf("%c",c[j]);
            j++;
        }
        else
        {
            f++;
        }
    }
    if(j%2!=0)
    {
        c[j] = 'x';
        printf("%c",c[j]);
        j++;
    }
    return j;
}

struct index findIndex(char ch)
{
    int i,j;
    struct index f;
    for(i=0;i<6;i++)
    {
        for(j=0;j<6;j++)
        {
            if(pc[i][j]==ch)
            {
                f.x = i;
                f.y = j;
                i=6;
                j=6;
            }
        }
    }
```

```
    return f;
}


void decrypt(int l)
{

            FILE *fp1;
           fp1 = fopen("decrypt.txt", "w");
    int i,ax,ay,bx,by;
    int x = 0;
    struct index f;
    for(i=0;i<l;i=i+2)
    {
        f = findIndex(ans[i]);
        ax = f.x;
        ay = f.y;
        f = findIndex(ans[i+1]);
        bx = f.x;
        by = f.y;
        if(ax == bx)
        {
            if(by-1<0)
            {
                decy[i+1] = pc[ax][5];
            }
            else
            {
                decy[i+1] = pc[ax][(by-1)%6];
            }
            if(ay-1<0)
            {
                decy[i] = pc[ax][5];
            }
            else
            {
                decy[i] = pc[ax][(ay-1)%6];
            }
```

```
          }
          else if(ay==by)
          {
               if(bx-1<0)
               {
                    decy[i+1] = pc[5][by];
               }
               else
               {
                    decy[i+1] = pc[(bx-1)%6][by];
               }
               if(ax-1<0)
               {
                    decy[i] = pc[5][ay];
               }
               else
               {
                    decy[i] = pc[(ax-1)%6][ay];
               }
          }
          else
          {
               decy[i] = pc[ax][by];
               decy[i+1] = pc[bx][ay];
          }
         printf("%c %c ",decy[i],decy[i+1]);
           fprintf(fp1,"%c",decy[i]);

          fprintf(fp1,"%c",decy[i+1]);
                decrypted[x] = decy[i];
                x++;
                decrypted[x] = decy[i+1];
                x++;

     }

}
```

```
int main()
{
    char s[100],k[100],string[100],ch; int z,n;
            FILE * fp;
              FILE *fp1;
            fp=fopen("encrypt.txt","r");
    printf("Enter the Key : \n");
    scanf("%s",k);
    z=0; n=0;
            while(1) {
                        ch=fgetc(fp);
                        if(ch!=EOF) {
                                    s[z++]=ch;
                                    ans[n++]=ch;
                        }
                        else if(ch==EOF)
                          break;
            }
            s[z]='\0';
            ans[n]='\0';
            puts(s);
            puts (ans);

    int i,j,f=0;

    table(k);
    for(i=0;i<6;i++)
    {
        for(j=0;j<6;j++)
        {
            printf("%c ",pc[i][j]);
        }
        printf("\n");
    }
    int l;
    printf("\nOrignal Message to decrypt : ");
    l = convert(s);
    printf("\n");
    printf("\n");
    printf("\nDecrypted Message : ");
    decrypt(l);
     printf("\n");
    return 0;
}
```

Client.c (using TCP)

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

int main(){

  int clientSocket;
  char buffer[1024];
  struct sockaddr_in serverAddr;
  socklen_t addr_size;

  /*---- Create the socket. The three arguments are: ----*/
  /* 1) Internet domain 2) Stream socket 3) Default protocol (TCP in this case) */
  clientSocket = socket(PF_INET, SOCK_STREAM, 0);

  /*---- Configure settings of the server address struct ----*/
  /* Address family = Internet */
  serverAddr.sin_family = AF_INET;
  /* Set port number, using htons function to use proper byte order */
  serverAddr.sin_port = htons(7891);
  /* Set IP address to localhost */
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  /* Set all bits of the padding field to 0 */
  memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

  /*---- Connect the socket to the server using the address struct ----*/
  addr_size = sizeof serverAddr;
  connect(clientSocket, (struct sockaddr *) &serverAddr, addr_size);

  /*---- Read the message from the server into the buffer ----*/
  recv(clientSocket, buffer, 1024, 0);

  /*---- Print the received message ----*/
  printf("Data received: %s",buffer);

  return 0;
}
```

Server.c (using TCP)

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

int main(){
  int welcomeSocket, newSocket;
  char buffer[1024];
  struct sockaddr_in serverAddr;
  struct sockaddr_storage serverStorage;
  socklen_t addr_size;
  FILE * fp2;
  int x = 0;
  char ch;

  /*---- Create the socket. The three arguments are: ----*/
  /* 1) Internet domain 2) Stream socket 3) Default protocol (TCP in this case) */
  welcomeSocket = socket(PF_INET, SOCK_STREAM, 0);

  /*---- Configure settings of the server address struct ----*/
  /* Address family = Internet */
  serverAddr.sin_family = AF_INET;
  /* Set port number, using htons function to use proper byte order */
  serverAddr.sin_port = htons(7891);
  /* Set IP address to localhost */
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  /* Set all bits of the padding field to 0 */
  memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

  /*---- Bind the address struct to the socket ----*/
  bind(welcomeSocket, (struct sockaddr *) &serverAddr, sizeof(serverAddr));

  /*---- Listen on the socket, with 5 max connection requests queued ----*/
  if(listen(welcomeSocket,5)==0)
    printf("Listening\n");
  else
    printf("Error\n");

  /*---- Accept call creates a new socket for the incoming connection ----*/
  addr_size = sizeof serverStorage;
  newSocket = accept(welcomeSocket, (struct sockaddr *) &serverStorage, &addr_size);

  /*---- Send message to the socket of the incoming connection ----*/
   fp2 = fopen("encrypt.txt", "r");
   ch = fgetc(fp2);
   while (ch != EOF)
                {
                 buffer[x] = ch;
                 x++;
                 ch = fgetc(fp2);
                }

    fclose(fp2);
  //strcpy(buffer,"Hello World\n");
  send(newSocket,buffer,130,0);

  return 0;
}
```