Laboratory Manual

For

E-Commerce and E-Security

(IT 710)

B.Tech (IT)

SEM VII

June 2010

Faculty of Technology
Dharmsinh Desai University
Nadiad.
www.ddu.ac.in

Table of Contents

Sample Experiment

1 AIM: Implement Play Fair Cipher Encryption
2 TOOLS/APPARATUS: Turbo C++ IDE

3 STANDARD PROCEDURES:

3.1 Analyzing the Problem:

By analyzing the problem I found required two basic steps for implementing the data encryption using Play Fair cipher

1) Generate Key matrix

2) Encrypt the data using encryption rule and key matrix

1) Generating Key matrix

To Generate the key matrix take any random key of any length and form a 5X5 matrix. Go on filling the raws of the matrix with the key characters (if repeating character occurs then ignore it). Fill the remaining matrix with alphabets from A to Z (except those already occurred in the key).

For example for the key "monarchy" we have the matrix as follow

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

2) Encrypt the data using encryption rule and key matrix

To Encrypt the data take two characters at time from plain text file and encrypt it using one of the following rules.

Encryption rules

1) Repeating plain text letters that would fall in the same pair are separated with filler letter,

such as x.(i.e. Balloon becomes Ba, lx, lo, on)

2) If both the characters are in the same raw then replace each with the character to its right, with

the last character followed by the first, in the matrix.

3) If both the characters are in the same column then replace each with the character below it, with

the bottom character followed by the top, in the matrix.

4) Otherwise each plain text letter is replaced by the letter that lies in its own row and the column occupied by the other plain text letter

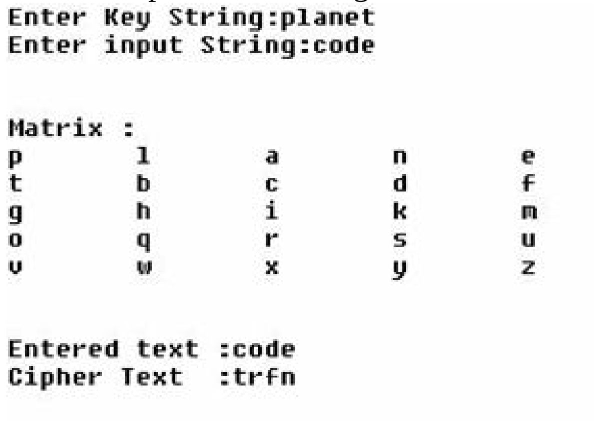
Example:

Using key as "monarchy" we have

- Encryption of AR as RM
- Encryption of MU as CM
- Encryption of BP as IM

3.2 Designing the Solution:

Solution implementation is given below:-



```
Enter Key String:planet
Enter input String:code


Matrix :
p       l       a       n       e
t       b       c       d       f
g       h       i       k       m
o       q       r       s       u
v       w       x       y       z


Entered text :code
Cipher Text  :trfn
```

For this solution we have to implement the following functions given below.

1) Input function for key & Plain Text.
2) Matrix generation.
3) Encryption function for generating Cipher Text.
4) Print function for printing Cipher Text Output.

3.3 Implementing the Solution

3.3.1 Source Code

```
//***************************************
// Play Fair Cipher Encryption
// **************************************
#include <stdio.h>
#define siz 5
void encrypt(int *i, int *j)
{
    (*i)++,(*j)++;
    if((*i)==siz) *i=0;
    else if((*j)==siz) *j=0;
}
```

```
// Playfair Logic Implementation
void playfair(char ch1,char ch2, char mat[siz][siz])
{
    int j,m,n,p,q,c,k;
    for(j=0,c=0;(c<2)||(j<siz);j++)
            for(k=0;k<siz;k++)
                    if(mat[j][k] == ch1)
                            m=j;
                            n=k;
                            c++;
                    else if(mat[j][k] == ch2)
                            p=j;
                            q=k;
                            c++;
                    if(m==p)
                            encrypt(&n,&q);
                    else if(n==q)
                            encrypt(&m,&p);
                    else
                            n+=q;
                            q=n-q;
                             n-=q;
                            printf("%c%c",mat[m][n],mat[p][q]);
}
void main()
{
   clrscr();
   char mat[siz][siz],key[10],str[25]={0};
   int m,n,i,j;
   char temp;
   printf("Enter Key String:");
   gets(key);
   m=n=0;
   // Matrix generation logic
   for(i=0;key[i]!='\0';i++)
    {
            for(j=0;j<i;j++)
                    if(key[j] == key[i]) break;
                    if(key[i]=='j') key[i]='i';
                    if(j>=i)
                    {
                            mat[m][n++] = key[i];
                            if(n==siz)
                                    n=0,m++;
                    }
        }
        for(i=97;i<=122;i++)
           {
                    for(j=0;key[j]!='\0';j++)
                            if(key[j] == i)
                             break;
                            else if(i=='j')

                                    break;
                     if(key[j]=='\0')
                     {
                            mat[m][n++] = i;
                            if(n==siz)
                                    n=0;
                                    m++;
                      }
            }
            printf("Enter input String:");
            gets(str);
            // Print Generated Matrix
            printf("\n\nMatrix :\n");
            for(i=0;i<siz;i++)
            {
                    for(j=0;j<siz;j++)
                            printf("%c\t",mat[i][j]);
                            printf("\n");
             }
             printf("\n\nEntered text :%s\nCipher Text :",str);
             for(i=0;str[i]!='\0';i++)
            {
                    temp = str[i++];
                    if(temp == 'j') temp='i';
                    if(str[i]=='\0')
                            playfair(temp,'x',mat);
                    else
                    {
                            if(str[i]=='j') str[i]='i';
                            if(temp == str[i])
                                    {
                                            playfair(temp,'x',mat);
                                            i--;
                                    }
                                    else
                                            playfair(temp,str[i],mat);
                            }
```

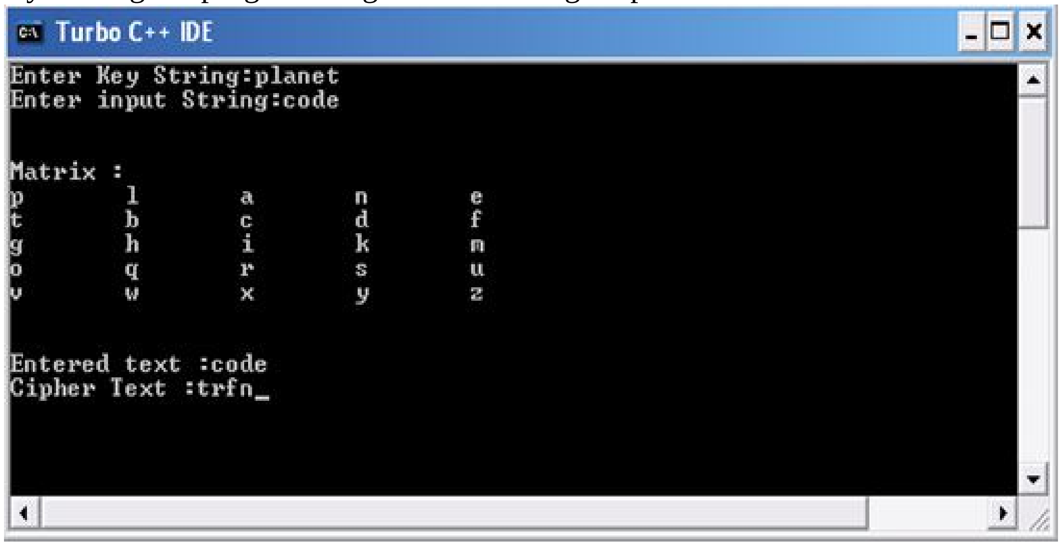
```
                }
        getch();
    }
```

3.3.2 Compilation /Running and Debugging the Solution

- Open the file Playfair.cpp.
- Compile using Alt+F9
- Run using Ctl+F9
- View output using Alt+F5

3.4 Testing the Solution

By Testing the program we get the following output.



```
Enter Key String:planet
Enter input String:code

Matrix :
p       l       a       n       e
t       b       c       d       f
g       h       i       k       m
o       q       r       s       u
v       w       x       y       z

Entered text :code
Cipher Text :trfn_
```

4 Conclusions

By this experiment; we can conclude that basic working of play fair cipher encryption methodology is working properly.

EXPERIMENT-1

Aim: Write program for Mono alphabetic cipher.
Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++
Procedure:

PART-1

Algorithm Encryption:

1. Open a file, which we want to encrypt, in read mode
2. Create new file.
3. Read one by one character of file-1 and encrpt it and put that character in file-2.

```
if character is between A to Z .
code =  Ascii(character) + key;          /* key = value between 1 to 25.
if code>ascii(Z)
code=code-26;
cipher_character = to_char(code);
```

Algorithm Decryption:

```
if character is between A to Z .
code= Ascii(character) – key;
if code<ascii(A)
code=code+26;
original_character =  to_char(code);
```

PART-2

Write program for keyword Mono alphabetic cipher.

In this case we will use the character string as Key instead of integer value

Suppose key is “how”

```
Replace   ABCDEFGHI………………………………..WXYZ  with
          HOWABCDEFGIJK...............MNPQ………UXYZ
```

In Encryption replace A with H , B with O and so on.

Hint: Make two character array as per key value..

EXPERIMENT-2

Aim: Implementation of Play Fair cipher.

Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++

Procedure:

There two step for the data encryption using Play Fair cipher

3) Generate Key matrix
4) Encrypt the data using encryption rule and key matrix

Generating Key matrix

To Generate the key matrix take any random key of any length and form a 5X5 matrix.

Go on filling the raws of the matrix with the key characters (if repeating character occurs then ignore it).

Fill the remaining matrix with alphabets from A to Z (except those already occurred in the key).

For example for the key “monarchy” we have the matrix as follow

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

To Encrypt the data take two characters at time from plain text file and encrypt it using one of the following rules.

Encryption rules

5) Repeating plain text letters that would fall in the same pair are separated with filler letter,
   such as x.(i.e. Balloon becomes Ba, lx, lo, on)
6) If both the characters are in the same raw then replace each with the character to its right, with
   the last character followed by the first, in the matrix.
7) If both the characters are in the same column then replace each with the character below it, with
   the bottom character followed by the top, in the matrix.
8) Otherwise each plain text letter is replaced by the letter that lies in its own row and the column
   occupied by the other plain text letter

Example:

Using key as “monarchy” we have

- Encryption of AR as RM
- Encryption of MU as CM
- Encryption of BP as IM

EXPERIMENT-3

Aim: Implementation of Vigenere cipher (Polyalphabetic substitution).

Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++

Procedure:

Vigenere cipher is a polyalphabetic substitution cipher which is an extension of Mono alphabetic
cipher. In this case we have more then one substitution sequences out of which we have to select only
one sequence to encrypt the plain text. The selection is based on Key value, so for the same message and different key the encryption will be different.

In this particular algorithm we have 26 different sequences which are describe as follow

1) A B C D …………………X Y Z
2) B C D E…………………..Y Z A
3) C D E F…………………Y Z A B

So on…….up to

26) Z A B C………………X Y

This sequences are arrange in a tabular form (26X26) as shown below

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

Here, each row corresponds to the message character (i.e. first raw corresponds to Message character A)
and each column corresponds to Key character (i.e. first column corresponds to Key character A)

Step to Encrypt the Message

1) Accept the key and Message and expand the key by repeating it over and over so that its length becomes equal to message length and then
2) Take the pair of key character and corresponding message character at a time
   - Find the raw corresponding to message character
   - Find the column corresponding to key character
3) Replace the message character with the character occupied by the raw and column obtain from step2
4) Repeat the process until end of message.

Example:

key:	d e c e p t i v e d e c e p t
Message:	w e a r e d i s c o v e r e d
Cipher text:	z i c v t w q n g r z g v t w

EXPERIMENT-4

Aim: Implementation of Hill cipher..

Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++

Procedure:

In this particular cipher we are accepting *m* no of plain text letters and converting them in to *m* no of cipher text letters using *m* different linear equations as shown below

For *m*=3

C1 = (K11* P1) + (K12 * P2) + (K13 * P3) mod 26
C2 = (K21* P1) + (K22 * P2) + (K23 * P3) mod 26
C3 = (K31* P1) + (K32 * P2) + (K33 * P3) mod 26

Where P1, P2, P3 are plain text letters and C1, C2, C3 are corresponding cipher text letters and constants K11, K12, K13,……,K31, K32, K33 are accepted form user as key value.

Thus we will accept the value of key (in form of *m X m* matrix) and Message (in form of a column matrix of *m*)

EXPERIMENT-5

Aim: Implementation of Gauss cipher

Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++

Procedure:

Guass cipher is a modification of mono alphabetic substitution cipher to improve upon its strength. In case of mono alphabetic cipher we have one plain text letter to be replaced by one fixed letter every time.

But in case of Gauss cipher we have plain text letter to be replaced by different letters every time (i.e. we have more replacement letters for the same plain text letter). The selection of replacement letter for particular plain text latter is random based on some criteria.

In this particular version of Guass cipher we have 4 different choice for each of the message character and the selection criteria is as follow

- If the message character occurs m^{th} then it is replaced with (m mod 4)th choice (i.e. if message character occurs first time then it replaced with first choice, second time then second choice and so on…….if it occurs fifth time then again with first choice.)

Algorithm

```
1) Assign 4 replacement characters to each of the character from A to Z randomly.
2) Assign one frequency counter (let it be F1 to F26) to each of the character from A to
   Z and Initialize each of them to 0.
3) Scan the input one character at a time
4) If end of file then go to step 9
5)     Inc the corresponding frequency count (say Fi )
6)     Replace the character with (Fi mod 4)^th choice
7)     Get next character
8)     Go to step 4
9) End
```

EXPERIMENT-6

Aim: Implementation of Rail Fence cipher
Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++

Procedure:

Rail Fence cipher is a Transposition cipher (i.e. we are encrypting the data by changing the position of the message text).Transposition is based on the value of the key.

In this particular scheme we are writing the message in a rectangle, row by row, and read the message off, column by column, but permute the order of column, the order of the column the becomes the key to the algorithm. No of the column is depends on the size of the key.

Example: Suppose we have a Key of size seven then it can have any 7 digit combination of 1 to 7

Key= 4 3 1 2 5 6 7
Message = "attack postponed until two am xyz"
Now we write the message in the matrix form where no of column in the matrix are seven

```
a t t a c k p
o s t p o n e
d u n t i l t
w o a m x y z
```

and then we inter change the position of column according to the key value

```
4   3   1   2   5   6   7
a   t   t   a   c   k   p
o   s   t   p   o   n   e
d   u   n   t   i   l   t
w   o   a   m   x   y   z
```

cipher text can be generated by reading the plain text in the column order specified by key

cipher text = ttna aptm tsuo aodw coix knly petz

EXPERIMENT-7

Aim: Implementation of S-DES algorithm for data encryption

Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++

Procedure:

S-DES algorithm uses bit wise operation on message letters to encrypt the data so it is more power full against the cryptanalysis attack. In this algorithm we will take 8-bits of the message at a time and operate on it using the 10-bit key and two rounds of iteration as explain below

Algorithm to generate key

As there are two rounds we have to generate two keys from the given 10-bit key

1: Apply permutation function P10 to 10 bit key
2: divide the result into two part each containing 5-bit L0 and L1
3: apply Circular Left Shift to both L0 and L1
4: combine both L0 and L1 which will form out 10-bit number
5: apply permutation function P8 on result to select 8 out of 10 bits for key K1 (for the first round)
6: again apply second Circular Left Shift to L0 and L1
7: combine the result, which will form out 10-bit number
8: apply permutation function P8 on result to select 8 out of 10 bits for key K2 (for the second round)

Algorithm for Encryption

1: get 8 bit message text (M) applied it to Initial permutation function (IP)
2: divide IP(M) into nibbles M0 and M1
3: apply function Fk on M0
4: XOR the result with M1 (M1 (+) Fk(M0))
5: Swap the result with M1 (i.e. make M1 as lower nibble (M0) and result as higher nibble (M1))
6: repeat the step 1 to 4 (go for the next round)
7: apply (IP^{-1}) on the result to get the encrypted data

Algorithm for function Fk

1: give the 4-bit input to EP (Expansion function) the result will be a 8-bit expanded data
2: XOR the 8-bit expanded data with 8-bit key (K1 for the first round and K2 for the second round)
2: divide result into upper (P1) and lower (P2) nibble
3: apply compression function S0 to P0 and S1 to P1, which will compress the 4-bit input to 2-bit output
4: combine 2-bit output from S0 and S1 to form a 4-bit digit
5: apply permutation function P4 to 4-bit result

Functions

```
P10 =  3  5  2  7  4  10  1  9  8  6
P8   =  6  3  7  4  8   5  10 9
P4   =  2  4  3  1
IP     = 2  6  3  1  4  8  5  7
```

IP^{-1} = 4 1 3 5 7 2 8 6

EP = 4 1 2 3 2 3 4 1

S0:

1	0	3	2
3	2	1	0
0	2	1	3
3	1	3	2

S1:

0	1	2	3
2	0	1	3
3	0	1	0
2	1	0	3

EXPERIMENT-8

Aim: Implement RSA asymmetric (public key and private key)-
Encryption.
Encryption key (e,n) & (d,n)

Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++

Procedure:

Encryption key (e,n) & (d,n)
-Select two prime no p & q.
- n = p*q
- Choose larger integer e such that is relatively prime to (p-1)*(q-1).
- Calculate d such that
e * d mod (p-1) * (q-1) =1.

cipher_code = Ascii(cipher_character)- ascii(A)
cipher_code = exp(code(character),e) mod n
code(character) = exp(cipher_code,d) mod n
let p=11 and q=13 then n = P*Q = 143 , so we can take e=141.
141 * d mod 143 =1 , so d = (141 ^ n-2) mod 143

EXPERIMENT-9

Aim: Generate digital signature using Hash code

Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++

Procedure:

Digital signature is the mechanism used to provide both authentication and confidentiality of the message. It works in following manner

Here the Hash function is used to generate the Hash Code of the message which provides the authentication and this Hash code is encrypted using the public key of receiver to provide confidentiality, and at the receiver side received Hash code is decrypted and compared against the Hash code generated from the received message.

Here we use the simple Hash function to generate the hash code of the message. In the simple Hash functions if we want to generate the m byte Hash code then we divide the message in to sub group each containing m byte and then XOR all the sub part to generate Hash code

For Example We have message M which is divided in to n sub part (M1, M2, M3, -….Mn) each containing m byte

Hi = Mi;   for i=0;

Hi= Mi (+)* Hi-1; for i=1 to n

(+)* - XOR function

EXPERIMENT-10

Aim: Generate digital signature using MAC code.
Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++
Procedure:

Procedure is same as explained in Practical 9 the only difference is here we are using MAC code instead of Hash Code to provide the authentication. We use the DES based algorithm to generate the MAC code.

The algorithms can be define using CBC mode of operation of DES with initialization vector of zero the data to be authenticated 8 bit blocks D1, D2,…………., Dn (if necessary the final block is padded with zeros). Using the DES encryption the Data Authentication Code (DAC) is generated as follow

$$O1 = E_k (D1)$$
$$O2 = E_k (D2 (+) D1)$$
$$O3 = E_k (D3 (+) D2)$$
…………….

$$On = E_k (Dn (+) Dn-1)$$

(+)* - XOR function

EXPERIMENT-11

Aim: Study of MD5 hash functions and implements the hash code using MD5
Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++
Procedure:

Step1: Append padding bits
Step2: Append length
Step3: Initialize MD buffer
Step4: Process message in 512-bit(16word)blocks
Step5: Output

EXPERIMENT-12

Aim: Study of SHA-1 hash function and implement the hash code using SHA-1.
Tools / Apparatus: O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++
Procedure:

Step1: Append padding bits
Step2: Append length
Step3: Initialize MD buffer
Step4: Process message in 512-bit(16word)blocks
Step5: Output

References

- Cryptography and Network Principles and Practice by Wiliam Stallings, Pearso Edu. 2003