**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

## LIST OF EXPERIMENTS

1. Implement Ceaser and mono alphabetic cipher
2. Implementation of  Play Fair cipher
3. Implementation of Hill cipher.
4. Implementation of S-DES algorithm for data encryption along with key generation of S-DES.
5. Write a program to generate and exchange public keys using client server mechanism.
6. Perform Encryption, Authentication and both using RSA. (Use public key shared in above practical)
7. Write a program to implement Diffie-Hellman Key exchange algorithm and perform encryption and decryption.
8. Write a program to authenticate a user with system using MD5 or SHA-1 Hashing technique.
9. Configure VPN using packet tracer and demonstrate the importance of IPSec.
10. Create Self Signed Certificate and configure it for website.

## LABWORK BEYOND CURRICULA

1 Study of Kerberos protocol using Linux.

2 Study of HMCA hash function and implement the hash code using HMAC.

## 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

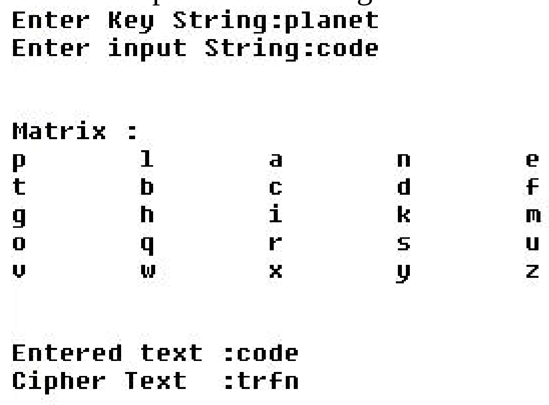
Example:

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])
                                        {
```

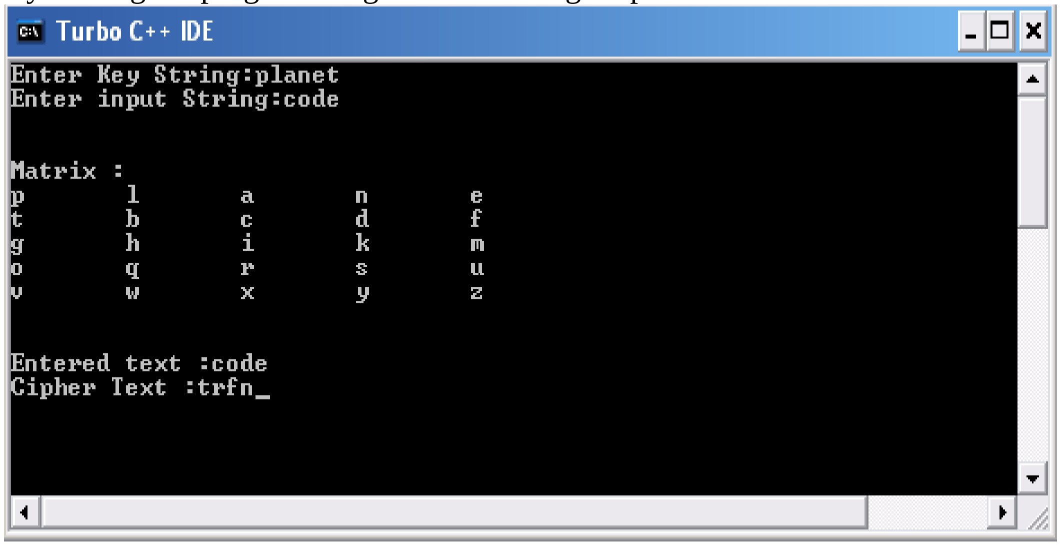
```
                                        playfair(temp,'x',mat);
                                        i--;
                                    }
                                    else
                                        playfair(temp,str[i],mat);
                                }
                            }
                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 Ceaser cipher and 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 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-4

**Aim:** Implementation of S-DES algorithm for data encryption along with key generation of S-DES.

**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-5

**Aim:** Write a program to generate and exchange RSA keys using client server mechanism.
**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
```

Example:
Given p as 5
Given q as 7
Compute n = p*q: n = 5*7 = 35
Compute m = (p-1)*(q-1): m = 4*6 = 24
Select e, such that e and m are co-prime numbers: e = 5
Compute d, such that d*e mod m = 1: d = 29
The public key {n,e} is = {35,5}
The private key {n,d} is = {35,29}

## EXPERIMENT-6

**Aim:** Perform Encryption, Authentication and both using RSA. (Use public key shared in above practical)

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

**Procedure:**

From above experiment now user have
public key {n,e} as {35,5} and
private key {n,e} as {35,29} and same procedure done on another client machine.
Now create client server socket program using which user can exchange public key with each other and perform encryption using following procedure.

**User A Side(Have public key of User B):**
**That public key is** {n,e} as {35,5}
Given clear text M represented in number as 23
Compute encrypted block C = M^e mod n:
C = 23^5 mod 35 = 6436343 mod 35 = 18
The cipher text C represented in number is 18
User B Side (who have its own private key)
User B have private key {n,e} as {35,29}.
Use this private key to decrypt the received message from user A.
M = 18^29 mod 35=23

Do same procedure for user B send message which is encrypted by User A's Public key and User A will decrypt message using its own private key.

## EXPERIMENT-7

**Aim:** Write a program to implement Diffie-Hellman Key exchange algorithm and perform encryption and decryption.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

Alice and Bob publicly agree to use a modulus $p$=23 and base $g$=5(which is a primitive root modulo 23).

1.Alice chooses a secret integer $a = 4$, then sends Bob $A = g^a \bmod p$

•$A = 5^4 \bmod 23=4$

2.Bob chooses a secret integer $b= 3$, then sends Alice $B = g^b \bmod p$

•$B=5^3 \bmod 23=10$

3.Alice computes Secret key $S =B^a \bmod p$

•$S = 10^4 \bmod 23 = 18$

4.Bob computes Secret key $S = A^b \bmod p$

•$S = 4^3 \bmod 23 = 18$

5.Alice and Bob now share a secret (the number 18).

Both Alice and Bob have arrived at the same value S.

$$A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p$$

## EXPERIMENT-8

**Aim:** Write a program to authenticate a user with system using MD5 or SHA-1 Hashing technique.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

User A send use name and password to User B using client server implemented using socket programming. User B save this username and password by creating its hash using MD5 or SHA-1 in file. Again when user A will supply username and password then User B will calculate Hash code using MD5 or SHA-1 and compare it with previously stored Hash code. If code match then authentication successful else fail.

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

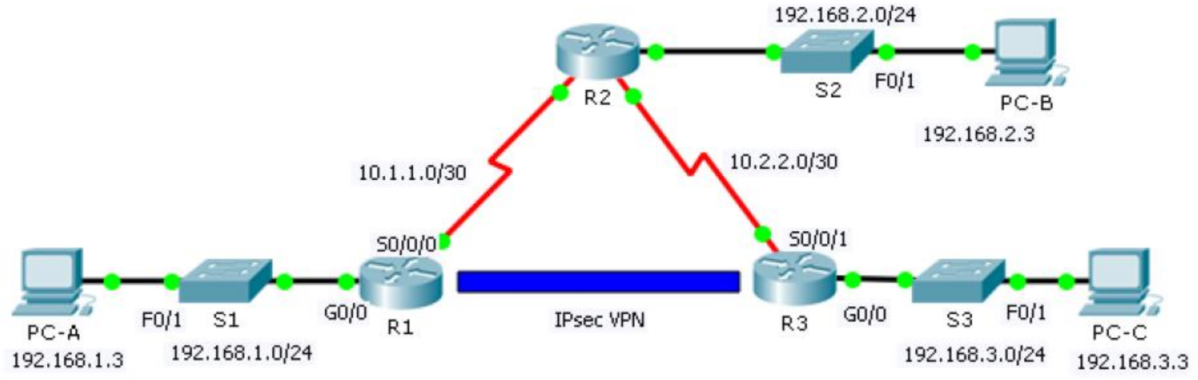
## EXPERIMENT-9

**Aim:** Configure VPN using Packet Tracer and demonstrate the importance of IPSec.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Packet Tracer

**Procedure:**

**Topology**



**Objectives**
Part 1: Enable Security Features
Part 2: Configure IPsec Parameters on R1 and R3
Part 3: Verify the IPsec VPN

**Part 1: Enable Security Features**

1. Issue the **show version** command in the user EXEC or privileged EXEC mode to verify that the Security Technology Package license is activated.
2. If not, activate the securityk9 module for the next boot of the router, accept the license, save the configuration, and reboot
   ```
   R1(config)# license boot module c2900 technology-package securityk9
   R1(config)# end
   R1# copy running-config startup-config
   R1# reload
   ```
3. After the reloading is completed, issue the show version again to verify the Security Technology Package license activation.
   Do in Router R3

**Part 2: Configure IPsec Parameters on R1 and R3**

1. Test connectivity
2. Identify interesting traffic on R1.
3. R1(config)# access-list 110 permit ip 192.168.1.0 0.0.0.255 192.168.3.0 0.0.0.255
4. Configure the ISAKMP Phase 1 properties on R1.
   ```
   R1(config)# crypto isakmp policy 10
   R1(config-isakmp)# encryption aes
   R1(config-isakmp)# authentication pre-share
    R1(config-isakmp)# group 2
   R1(config-isakmp)# exit
   R1(config)# crypto isakmp key cisco address 10.2.2.2
   ```
5. Configure the ISAKMP Phase 2 properties on R1.
   ```
   R1(config)# crypto ipsec transform-set VPN-SET esp-3des esp-sha-hmac
   R1(config)# crypto map VPN-MAP 10 ipsec-isakmp
   R1(config-crypto-map)# description VPN connection to R3
   R1(config-crypto-map)# set peer 10.2.2.2
   R1(config-crypto-map)# set transform-set VPN-SET
   R1(config-crypto-map)# match address 110
   R1(config-crypto-map)# exit
   ```
6. Configure the crypto map on the outgoing interface
   ```
   R1(config)# interface S0/0/0
   R1(config-if)# crypto map VPN-MAP
   ```
7. Configure IPsec Parameters on R3 same as R1

**Part 3: Verify the IPsec VPN**

1. Verify the tunnel prior to interesting traffic
2. R1# show crypto ipsec sa

## EXPERIMENT-10

**Aim:** Create Self Signed Certificate and configure it for website.
**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS
**Packages:** Turbo/Borland/GNU - C/C++
**Procedure:**

open the “Visual Studio Command Promt (2010)” or the “Developer Command Prompt for VS2012” and run the following two steps:

```
makecert.exe -sv TestCodeSign.pvk -n “CN=Test Code Sign” TestCodeSign.cer
```

### To install a self-signed certificate in the Trusted Root Certification Authorities

1. Open the certificate snap-in.
2. View certificates in the MMC snap-in
3. Select **Run** from the **Start** menu, and then enter *mmc*. The MMC appears.
4. From the **File** menu, select **Add/Remove Snap In**.
5. The **Add or Remove Snap-ins** window appears.
6. From the **Available snap-ins** list, choose **Certificates**, then select **Add**.
7. In the **Certificates snap-in** window, select **Computer account**, and then select **Next**.
8. Optionally, you can select **My user account** for the current user or **Service account** for a particular service.
9. In the **Select Computer** window, leave **Local computer** selected, and then select **Finish**.
10. In the **Add or Remove Snap-in** window, select **OK**.
11. Open the folder to store the certificate, either the **Local Computer** or the **Current User**.
12. Open the **Trusted Root Certification Authorities** folder.
13. Right-click the **Certificates** folder and click **All Tasks**, then click **Import**.
14. Follow the on-screen wizard instructions to import the RootCA.pfx into the store.

## EXPERIMENT-11

**Aim:** Study of Kerberos protocol using Linux.
**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS
Packages: Turbo/Borland/GNU - C/C++

## EXPERIMENT-12

**Aim:** Study of HMCA hash function and implement the hash code using HMAC
**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS