ST. ANNE'S COLLEGE OF ENGINEERING AND TECHNOLOGY

ANGUCHETTYPALAYAM, PANRUTI – 607 110

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

LAB MANUAL

EC 3401 – NETWORKS AND SECURITY LABORATORY

Regulation 2021

Year / Semester: II / IV

MARCH 2024 - JUN 2024

EC 3401 – NETWORKS AND SECURITY LABORATORY

OBJECTIVES:

- To learn the Network Models and datalink layer functions.
- To understand routing in the Network Layer.
- To explore methods of communication and congestion control by the Transport Layer.
- To study the Network Security Mechanisms.
- To learn various hardware security attacks and their countermeasures.

PRACTICAL EXERCISES: EXPERIMENTS USING C

1.Implement the Data Link Layer framing methods,

i) Bit stuffing, (ii) Character stuffing

2. Implementation of Error Detection / Correction Techniques

i) LRC, (ii) CRC, (iii) Hamming code

3. Implementation of Stop and Wait, and Sliding Window Protocols

4. Implementation of Go back-N and Selective Repeat Protocols.

5. Implementation of Distance Vector Routing algorithm (Routing Information Protocol) (Bellman-Ford).

6. Implementation of Link State Routing algorithm (Open Shortest Path First) with 5 nodes (Dijkstra's).

7. Data encryption and decryption using Data Encryption Standard algorithm.

8. Data encryption and decryption using RSA (Rivest, Shamir and Adleman) algorithm.

9. Implement Client Server model using FTP protocol.

Experiments using Tool Command Language

1. Implement and realize the Network Topology - Star, Bus and Ring using NS2.

2. Implement and perform the operation of CSMA/CD and CSMA/CA using NS2.

PRACTICAL: 30 PERIODS

TABLE OF CONTENTS

1.Implement the Data Link Layer framing methods,

i) Bit stuffing

Aim::

To write a c program using Implement the Data Link Layer framing methods in Bit Stuffing.

Algorithm::

STEP-1: Read the Frame Size from the user.

STEP-2: Read the key value from the user.

STEP-3: If the size convert add the loop values.

STEP-4: Values convert the frames

STEP-5: Display the Bit Stuffing obtained above

Program::

```
#include<stdio.h>
#include<string.h>
int main()
{
   int a[20],b[30],i,j,k,count,n;
   printf("Enter frame size (Example: 8):");
   scanf("%d",&n);
   printf("Enter the frame in the form of 0 and 1 :");
   for(i=0; i<n; i++)
      scanf("%d",&a[i]);
   i=0;
   count=1;
   j=0;
   while(i<n)
   {
      if(a[i]==1)
      {
         b[j]=a[i];
         for(k=i+1; a[k]==1 && k<n && count<5; k++)
         {
            j++;
```

```
            b[j]=a[k];
            count++;
            if(count==5)
            {
               j++;
               b[j]=0;
            }
            i=k;
         }
      }
      else
      {
         b[j]=a[i];
      }
      i++;
      j++;
   }
   printf("After Bit Stuffing :");
   for(i=0; i<j; i++)
      printf("%d",b[i]);
   return 0;
}
```

Output:

```
Enter frame size (Example: 8):12
Enter the frame in the form of 0 and 1 :0 1 0 1 1 1 1 1 1 0 0 1
After Bit Stuffing :0101111101001
```

Result::

Thus, the above program was Successfully completed and verified.

(ii) Character stuffing

Aim::

To write a c program using Implement the Data Link Layer framing methods in Character Stuffing.

Algorithm::

STEP-1: Read the Frame Size from the user.

STEP-2: Read the Character from the user.

STEP-3: convert the Starting and ending delimiter

STEP-4: Values convert the frames

STEP-5: Display the Character Stuffing obtained above

Program:

```
#include<stdio.h>
#include<string.h>
main()
{
    char a[30], fs[50] = " ", t[3], sd, ed, x[3], s[3], d[3], y[3];
    int i, j, p = 0, q = 0;
    clrscr();
    printf("Enter characters to be stuffed:");
    scanf("%s", a);
    printf("\nEnter a character that represents starting delimiter:");
    scanf(" %c", &sd);
    printf("\nEnter a character that represents ending delimiter:");
    scanf(" %c", &ed);
    x[0] = s[0] = s[1] = sd;
    x[1] = s[2] = '\0';
    y[0] = d[0] = d[1] = ed;
    d[2] = y[1] = '\0';
    strcat(fs, x);
    for(i = 0; i < strlen(a); i++)
    {
       t[0] = a[i];
       t[1] = '\0';
       if(t[0] == sd)
          strcat(fs, s);
       else if(t[0] == ed)
          strcat(fs, d);
       else
```

```
            strcat(fs, t);
        }
        strcat(fs, y);
        printf("\n After stuffing:%s", fs);
        getch();
    }
```

Output:-

Enter characters to be stuffed: goodday

Enter a character that represents starting delimiter: d

Enter a character that represents ending delimiter: g

After stuffing: dggooddddayg.

Result::

Thus, the above program was Successfully completed and verified.

2. Implementation of Error Detection / Correction Techniques

Aim::

To write a c program using Implement of Error Detection / Correction Techniques in LRC,CRC.

Algorithm::

STEP 1 :Get the data and generator polynomial.
STEP 2 :Let n be the length of the generator polynomial.
STEP 3 :Append n-1 zeros to data.
STEP 4 :Call the CRC function.
STEP 5 :
STEP 6 :If the first bit is 1, then perform a xor operation with the first n bits of data and the generator polynomial.
STEP 7 :Shift the bits by 1 position to leave the first bit.
STEP 8 :Append a bit from the data. Repeat the process until all the bits in the data are appended.

i) LRC,

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int l1,bit[100],count=0,i,choice;
 clrscr();
 printf("Enter the length of data stream: ");
 scanf("%d",&l1);
 printf("\nEnter the data stream ");
 for(i=0;i<l1;i++)
 {
  scanf("%d",&bit[i]);
  if(bit[i]==1)
  count=count+1;
```

```
 }
 printf("Number of 1's are %d",count);
 printf("\nEnter the choice to implement parity bit");
 printf("\n1-Sender side\n2-Receiver side\n");
 scanf("%d",&choice);
 switch(choice)
 {
  case 1:
        if(count%2==0)
              bit[l1]=0;
        else
              bit[l1]=1;

        printf("\nThe data stream after adding parity bit is\n");
        for(i=0;i<=l1;i++)
              printf("%d",bit[i]);
        break;

  case 2:
        if(count%2==0)
              printf("There is no error in the received data stream");
        else
              printf("There is error in the received data stream");
        break;

  default:
        printf("Invalid choice");
        break;
 }
getch();
}
```

Output::

Enter the length of data stream: 10

Enter the data stream 1 1 0 1 0 1 1 1 0 1

Number of 1's are 7

Enter the choice to implement parity bit

1-Sender side

2-Receiver side

1

The data stream after adding parity bit is

11010111011

Enter the length of data stream: 10

Enter the data stream 1 1 1 1 1 0 0 0 1 0

Number of 1's are 6

Enter the choice to implement parity bit

1-Sender side

2-Receiver side

2

There is no error in the received data stream

(ii) CRC,

Program::

```
#include <stdio.h>
 #include <conio.h>
 #include <string.h>
 void main() {
        int i,j,keylen,msglen;
        char input[100], key[30],temp[30],quot[100],rem[30],key1[30];
        clrscr();
        printf("Enter Data: ");
        gets(input);
        printf("Enter Key: ");
        gets(key);
        keylen=strlen(key);
        msglen=strlen(input);
        strcpy(key1,key);
        for (i=0;i<keylen-1;i++) {
                input[msglen+i]='0';
        }
        for (i=0;i<keylen;i++)
         temp[i]=input[i];
        for (i=0;i<msglen;i++) {
                quot[i]=temp[0];
                if(quot[i]=='0')
                 for (j=0;j<keylen;j++)
                 key[j]='0'; else
                 for (j=0;j<keylen;j++)
                 key[j]=key1[j];
                for (j=keylen-1;j>0;j--) {
                        if(temp[j]==key[j])
                         rem[j-1]='0'; else
                         rem[j-1]='1';
                }
                rem[keylen-1]=input[i+keylen];
                strcpy(temp,rem);
```

```
        }
        strcpy(rem,temp);
        printf("\nQuotient is ");
        for (i=0;i<msglen;i++)
         printf("%c",quot[i]);
        printf("\nRemainder is ");
        for (i=0;i<keylen-1;i++)
         printf("%c",rem[i]);
        printf("\nFinal data is: ");
        for (i=0;i<msglen;i++)
         printf("%c",input[i]);
        for (i=0;i<keylen-1;i++)
         printf("%c",rem[i]);
        getch();
}
```

Output::

Enter the Number::

1

0

1

0

0

0

0

1

Enter the divisor

1

0

0

1

The quotient is 10110111 and the remainder is 0111

Result::

Thus, the above program was Successfully completed and verified.

3. Implementation of Stop and Wait, and Sliding Window Protocols

Aim::

To write a c program using Implementation of Stop and Wait, and Sliding Window Protocols

Algorithm::

STEP 1: Start the Program

STEP 2: Import all the necessary packages

STEP 3: Create two application sender and receiver

STEP 4: Connect both applications using socket

STEP 5: Sender port number and the frame is input as receiver

STEP 6: Sender frame is send to the receiver and display to the reciever

Program::

```
#include<stdio.h>
#include<conio.h>
void main()
{
char sender[50],receiver[50];
int i,winsize;
clrscr();
 printf("\n ENTER THE WINDOWS SIZE : ");
scanf("%d",&winsize);
 printf("\n SENDER WINDOW IS EXPANDED TO STORE MESSAGE OR WINDOW \n");
 printf("\n ENTER THE DATA TO BE SENT: ");
fflush(stdin);
gets(sender);
for(i=0;i<winsize;i++)
receiver[i]=sender[i];
receiver[i]=NULL;
 printf("\n MESSAGE SEND BY THE SENDER:\n");
 puts(sender);
 printf("\n WINDOW SIZE OF RECEIVER IS EXPANDED\n");
 printf("\n ACKNOWLEDGEMENT FROM RECEIVER \n");
for(i=0;i<winsize;i++);
```

```
printf("\n ACK:%d",i);
 printf("\n MESSAGE RECEIVED BY RECEIVER IS : ");
 puts(receiver);
 printf("\n WINDOW SIZE OF RECEIVER IS SHRINKED \n");
getch();
}
```

Output::

```
Enter the windows size : 10
Sender window is expanded to store message or window
Enter the data to be sent: forgetcode.com
Message send by the sender:
forgetcode.com
Window size of receiver is expanded
Acknowledgement from receiver
Ack:5
Message received by receiver is : forgetcode
Window size of receiver is shrinked
```

Result::

Thus, the above program was Successfully completed and verified.

4. Implementation of Go back-N and Selective Repeat Protocols.

Aim::

To write a c program using Implementation of Go back-N and Selective Repeat Protocols

Algorithm::

STEP 1. Start.
STEP 2. Establish connection (recommended UDP)
STEP 3. Accept the window size from the client(should be <=40)
STEP 4. Accept the packets from the network layer.
STEP 5. Calculate the total frames/windows required.
STEP 6. Send the details to the client(totalpackets,totalframes.)
STEP 7. Initialise the transmit buffer.
STEP 8. Built the frame/window depending on the windowsize.
STEP 9. Transmit the frame.
STEP 10. Wait for the acknowledgement frame.
STEP 11. Close the connection.

Program::

```
#include<stdio.h>
int main()
{
        int windowsize,sent=0,ack,i;
        printf("enter window size\n");
        scanf("%d",&windowsize);
        while(1)
        {
                for( i = 0; i < windowsize; i++)
                        {
                                printf("Frame %d has been transmitted.\n",sent);
                                sent++;
                                if(sent == windowsize)
                                        break;
                        }
                        printf("\nPlease enter the last Acknowledgement received.\n");
                        scanf("%d",&ack);

                        if(ack == windowsize)
                                break;
```

```
                    else
                        sent = ack;
            }
return 0;
}
```

Output:-

```
enter window size
8
Frame 0 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Frame 6 has been transmitted.
Frame 7 has been transmitted.

Please enter the last Acknowledgement received.
2
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Frame 6 has been transmitted.
Frame 7 has been transmitted.

Please enter the last Acknowledgement received.   8
```

Result::

Thus, the above program was Successfully completed and verified.

5. Implementation of Distance Vector Routing algorithm (Routing Information Protocol) (Bellman-Ford).

Aim::

To write a c program using Implementation of Distance Vector Routing algorithm (Routing Information Protocol) (Bellman-Ford)

Algorithm::

STEP 1. Open VI-RTSIM software from desktop

STEP 2. Click the Simulation menu bar

STEP 3. Select the "Distance – Vector Routing Algorithm" option from Routing algorithm menu bar.

STEP Network with routers connected through link is drawn by using option in editor(add router, join link, delete router, delete link, Add caption to link, add caption to router)

STEP 5. Select any two nodes to find the shortest distance between them.

STEP 6:Click the Find path Button to run the program.

STEP 7. Now the shortest paths between the two nodes are calculated.

Program::

```
#include<stdio.h>
struct node
{
   unsigned dist[20];
   unsigned from[20];
}rt[10];
int main()
{
   int costmat[20][20];
   int nodes,i,j,k,count=0;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&nodes);//Enter the nodes
   printf("\nEnter the cost matrix :\n");
   for(i=0;i<nodes;i++)
   {
      for(j=0;j<nodes;j++)
      {
```

```
            scanf("%d",&costmat[i][j]);
            costmat[i][i]=0;
            rt[i].dist[j]=costmat[i][j];//initialise the distance equal to cost matrix
            rt[i].from[j]=j;
        }
    }
        do
        {
            count=0;
            for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance
from the node i to k using the cost matrix
            //and add the distance from k to node j
            for(j=0;j<nodes;j++)
            for(k=0;k<nodes;k++)
                if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
                {//We calculate the minimum distance
                    rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                    rt[i].from[j]=k;
                    count++;
                }
        }while(count!=0);
        for(i=0;i<nodes;i++)
        {
            printf("\n\n For router %d\n",i+1);
            for(j=0;j<nodes;j++)
            {
                printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
            }
        }
    printf("\n\n");
    getch();
}
/*
```

Output::

A sample run of the program works as:-
Enter the number of nodes :
3
Enter the cost matrix :
0 2 7
2 0 1
7 1 0
For router 1
node 1 via 1 Distance 0
node 2 via 2 Distance 2
node 3 via 3 Distance 3
For router 2
node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 1
For router 3
node 1 via 1 Distance 3
node 2 via 2 Distance 1
node 3 via 3 Distance 0

Result::

Thus, the above program was Successfully completed and verified.

6. Implementation of Link State Routing algorithm (Open Shortest Path First) with 5 nodes (Dijkstra's).

Aim::

To write a c program using Implementation of Link State Routing algorithm (Open Shortest Path First) with 5 nodes (Dijkstra's).

Algorithm::

STEP 1. Create a simulator object

STEP 2. Define different colors for different data flows

STEP 3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.

STEP 4. Create n number of nodes using for loop

STEP 5. Create duplex links between the nodes

STEP 6. Setup UDP Connection between n(0) and n(5)

STEP 7. Setup another UDP connection between n(1) and n(5)

STEP 8. Apply CBR Traffic over both UDP connections

STEP 9. Choose Link state routing protocol to transmit data from sender to receiver.

STEP 10. Schedule events and run the program.

Program::

```
#include <limits.h>
#include <stdio.h>
#define V 9
int minDistance(int dist[], bool sptSet[]) {
  int min = INT_MAX, min_index;
  for (int v = 0; v < V; v++)
  if (sptSet[v] == false && dist[v] <= min)
    min = dist[v], min_index = v;
  return min_index;
}
int printSolution(int dist[], int n) {
   printf("Vertex Distance from Source\n");
   for (int i = 0; i < V; i++)
     printf("%d \t %d\n", i, dist[i]);
}
```

```
void dijkstra(int graph[V][V], int src) {
  int dist[V];
  bool sptSet[V];
  for (int i = 0; i < V; i++)
    dist[i] = INT_MAX, sptSet[i] = false;
    dist[src] = 0;
  for (int count = 0; count < V - 1; count++) {
    int u = minDistance(dist, sptSet);
    sptSet[u] = true;
    for (int v = 0; v < V; v++)
      if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v])
dist[v] = dist[u] + graph[u][v];
  }
  printSolution(dist, V);
}
int main()
{
  int graph[V][V] = { { 0, 6, 0, 0, 0, 0, 0, 8, 0 },
    { 6, 0, 8, 0, 0, 0, 0, 13, 0 },
    { 0, 8, 0, 7, 0, 6, 0, 0, 2 },
    { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
    { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
    { 0, 0, 6, 14, 10, 0, 2, 0, 0 },
    { 0, 0, 0, 0, 0, 2, 0, 1, 6 },
    { 8, 13, 0, 0, 0, 0, 1, 0, 7 },
    { 0, 0, 2, 0, 0, 0, 6, 7, 0 }
  };
  dijkstra(graph, 0);
  return 0;
}
```

Output::

Vertex Distance from Source

0	0
1	6
2	14
3	21
4	21
5	11
6	9
7	8
8	15

Result::

Thus, the above program was Successfully completed and verified.

7. Data encryption and decryption using Data Encryption Standard algorithm.

Aim::

To write a c program using Data encryption and decryption using Data Encryption Standard algorithm.

Algorithm::

STEP-1: Read the 64-bit plain text.

STEP-2: Split it into two 32-bit blocks and store it in two different arrays.

STEP-3: Perform XOR operation between these two arrays.

STEP-4: The output obtained is stored as the second 32-bit sequence and the original second 32-bit sequence forms the first part.

STEP-5: Thus the encrypted 64-bit cipher text is obtained in this way. Repeat the same process for the remaining plain text characters.

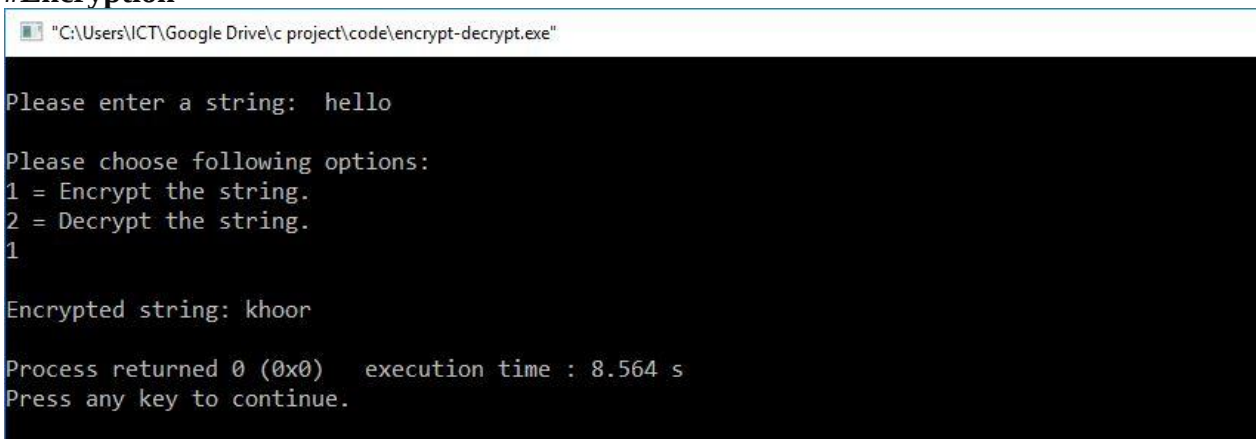
Program::

```
//Simple C program to encrypt and decrypt a string

#include <stdio.h>
int main()
{
  int i, x;
  char str[100];

  printf("\nPlease enter a string:\t");
  gets(str);

  printf("\nPlease choose following options:\n");
  printf("1 = Encrypt the string.\n");
  printf("2 = Decrypt the string.\n");
  scanf("%d", &x);

  //using switch case statements
  switch(x)
  {
  case 1:
    for(i = 0; (i < 100 && str[i] != '\0'); i++)
      str[i] = str[i] + 3; //the key for encryption is 3 that is added to ASCII value

    printf("\nEncrypted string: %s\n", str);
```

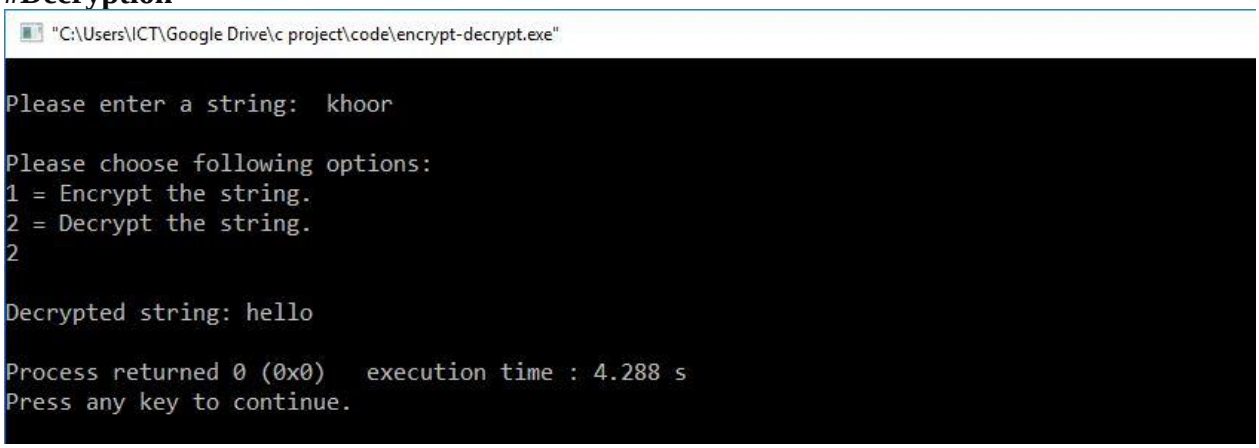
```
    break;

  case 2:
    for(i = 0; (i < 100 && str[i] != '\0'); i++)
      str[i] = str[i] - 3; //the key for encryption is 3 that is subtracted to ASCII value

    printf("\nDecrypted string: %s\n", str);
    break;

  default:
    printf("\nError\n");
  }
  return 0;
}
```

Output:

#Encryption



#Decryption



Result::

Thus, the above program was Successfully completed and verified.

8. Data encryption and decryption using RSA (Rivest, Shamir and Adleman) algorithm.

Aim::

To write a c program using Data encryption and decryption using RSA (Rivest, Shamir and Adleman) algorithm

Algorithm::

STEP-1: Select two co-prime numbers as p and q.

STEP-2: Compute n as the product of p and q.

STEP-3: Compute (p-1)*(q-1) and store it in z.

STEP-4: Select a random prime number e that is less than that of z.

STEP-5: Compute the private key, d as e * mod-1(z).

STEP-6: The cipher text is computed as messagee * mod n.

STEP-7: Decryption is done as cipherdmod n.

Program::

```
#include<stdio.h>
#include<math.h>

//to find gcd
int gcd(int a, int h)
{
   int temp;
   while(1)
   {
      temp = a%h;
      if(temp==0)
      return h;
      a = h;
      h = temp;
   }
}

int main()
{
   //2 random prime numbers
   double p = 3;
   double q = 7;
   double n=p*q;
   double count;
   double totient = (p-1)*(q-1);
```

```
    //public key
    //e stands for encrypt
    double e=2;

    //for checking co-prime which satisfies e>1
    while(e<totient){
    count = gcd(e,totient);
    if
(count==1)
       break;
    else
       e++;
    }

    //private key
    //d stands for decrypt
    double d;

    //k can be any arbitrary value
    double k = 2;

    //choosing d such that it satisfies d*e = 1 + k * totient
    d = (1 + (k*totient))/e;
    double msg = 12;
    double c = pow(msg,e);
    double m = pow(c,d);
    c=fmod(c,n);
    m=fmod(m,n);

    printf("Message data = %lf",msg);
    printf("\np = %lf",p);
    printf("\nq = %lf",q);
    printf("\nn = pq = %lf",n);
    printf("\ntotient = %lf",totient);
    printf("\ne = %lf",e);
    printf("\nd = %lf",d);
    printf("\nEncrypted data = %lf",c);
    printf("\nOriginal Message Sent = %lf",m);

    return 0;
}
```

Output:

Message data = 12.000000

p = 3.000000

q = 7.000000

n = pq = 21.000000

totient = 12.000000

e = 5.000000

d = 5.000000

Encrypted data = 3.000000

Original Message Sent = 12.000000

Result::

Thus, the above program was Successfully completed and verified.

9. Implement Client Server model using FTP protocol.

Aim::

To write a c program using Implement Client Server model using FTP protocol.

Algorithm::

STEP 1. The server starts and **waits for filename**.

STEP 2. The client **sends a filename**.

STEP 3. The server receives filename.

If file is present,

server starts reading file

and continues t**o send a buffer filled with**

file contents encrypted until file-end is reached.

STEP 4. **End is marked by EOF**.

STEP 5. File is received as buffers **until EOF is received**. Then it is decrypted.

STEP 6. If Not present, a **file not found** is sent.

Program::

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/stat.h>
#include <sys/sendfile.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>

#define FILENAME   "a.txt"
#define SERVER_IP  "127.0.0.1"
#define SERVER_PORT       65496

int main(int argc , char **argv)
{
```

```
int       socket_desc;
struct    sockaddr_in server;
char      request_msg[BUFSIZ],
          reply_msg[BUFSIZ];

// Variables for the file being received
int       file_size,
          file_desc;
char      *data;

socket_desc = socket(AF_INET, SOCK_STREAM, 0);
if (socket_desc == -1)
{
          perror("Could not create socket");
          return 1;
}
server.sin_addr.s_addr = inet_addr(SERVER_IP);
server.sin_family = AF_INET;
server.sin_port = htons(SERVER_PORT);
// Connect to server
if (connect(socket_desc, (struct sockaddr *)&server, sizeof(server)) < 0)
{
          perror("Connection failed");
          return 1;
}

// Get a file from server
strcpy(request_msg, "Get ");
strcat(request_msg, FILENAME);
write(socket_desc, request_msg, strlen(request_msg));
recv(socket_desc, reply_msg, 2, 0);

// Start receiving file
if (strcmp(reply_msg, "OK") == 0) {
          recv(socket_desc, &file_size, sizeof(int), 0);
          data = malloc(file_size);
```

```
                file_desc = open(FILENAME, O_CREAT | O_EXCL | O_WRONLY, 0666);
                recv(socket_desc, data, file_size, 0);
                write(file_desc, data, file_size);
                close(file_desc);
        }
        else {

                fprintf(stderr, "Bad request\n");
        }

        return 0;
}
```

Output:

Connection failed: Connection refused

Result::

Thus, the above program was Successfully completed and verified.