Dharmsinh Desai University, Nadiad
Department of Information Technology
ECES, IT718
B.Tech. IT, Sem: VII

Experiment 7

Submitted By
Name: - Dishant Modh
Roll No: - IT076

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

1. Server.c

```c
#include <stdio.h>
#include <sys/socket.h>
#include <unistd.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>
#include "function.h"
#define SERV_PORT 7069
int listenSD, clientSD, noOfBytesRead = 0;
struct sockaddr_in servAddr, clientAddr;
ll privateKey, sessionKey, message;
publicKeyInformation server, client, tmp;
void processClient(int clientSD)
{
    if ((noOfBytesRead = read(clientSD, &client, sizeof(client))) > 0)
    {
        printf("\nServer recieved Client Name : %s.\n", client.name);
        printf("Server recieved Client Key : < %llu >\n", client.publicKey);
        write(clientSD, &server, sizeof(server));
        sessionKey = findSessionKey(privateKey, client.publicKey);
        printf("Session Key of Server : < %llu >\n", sessionKey);
    }
    while ((noOfBytesRead = read(clientSD, &message, sizeof(message))) > 0)
    {
        printf("\n\tServer recieved cipher text from Client: %llu.\n", message
);
        message = decryption(message, sessionKey);
        printf("\tPlain text: %llu\n", message);
        printf("\tDouble of %llu : %llu\n", message, message * 2);
        message *= 2;
        message = encryption(message, sessionKey);
        printf("\tServer sent cipher text to Client: %llu.\n", message);
        write(clientSD, &message, noOfBytesRead);
    }
}
int main()
{
    if ((listenSD = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("Error: Socket creation not allowed.\n");
        return -1;
    }
    bzero(&servAddr, sizeof(servAddr));
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(SERV_PORT);
    servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    if (bind(listenSD, (struct sockaddr *)&servAddr, sizeof(servAddr)) < 0)
    {
        printf("Error: Socket not bind for server.\n");
        return -1;
    }
    if (listen(listenSD, 5) < 0)
    {
        printf("Error: Socket not available for listening.\n");
        return -1;
    }
    printf("Enter Server Name: ");
    scanf("%s", server.name);
    printf("Enter Private Key: ");
    scanf("%llu", &privateKey);
    server.publicKey = findPublicKey(privateKey);
    printf("\nPublic key of Server: < %llu >\n", server.publicKey);
    printf("Private key of Server: < %llu >\n", privateKey);
    while (1)
    {
        clientSD = accept(listenSD, (struct sockaddr *)NULL, NULL);
        if (fork() == 0)
        {
            close(listenSD);
            processClient(clientSD);
            close(clientSD);
            return 0;
        }
        close(clientSD);
    }
    return 0;
}
```

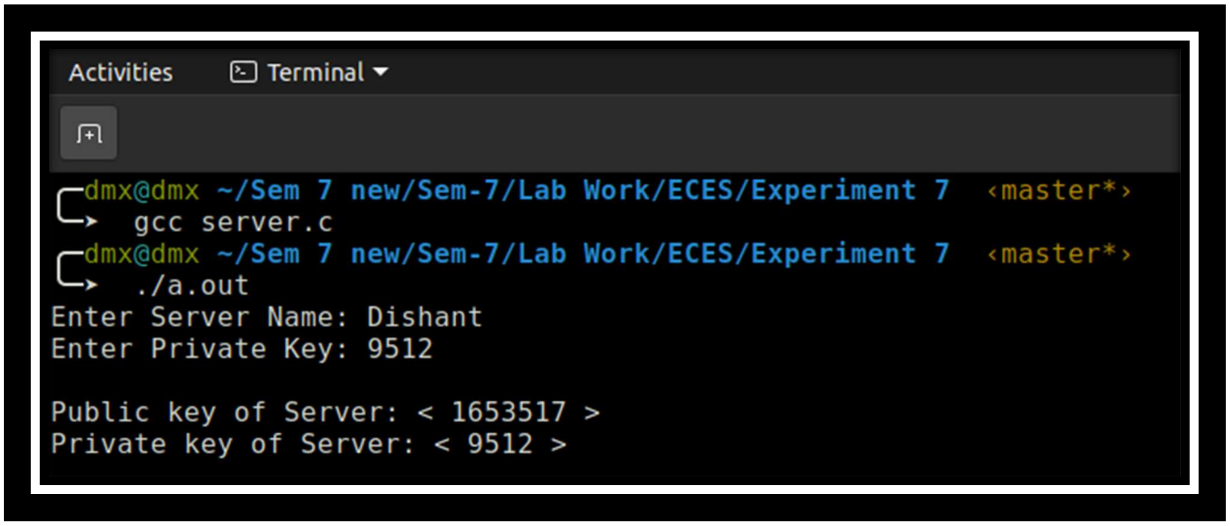
2. Client.c

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include "function.h"
#define SERV_PORT 7069
int main(int argc, char **argv)
{
    int connectSD, noOfBytesRead = 0;
    struct sockaddr_in servAddr;
    ll privateKey, sessionKey, message, tmp;
    publicKeyInformation client, server;
    if (argc != 2)
    {
        printf("Usage: %s IP-Address\n", argv[0]);
        return -1;
    }
    if ((connectSD = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("Error: Socket creation not allowed.\n");
        return -1;
    }
    bzero(&servAddr, sizeof(servAddr));
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(SERV_PORT);
    if (inet_pton(PF_INET, argv[1], &servAddr.sin_addr) < 0)
    {
        printf("Error: Socket not bind for server.\n");
        return -1;
    }
    if (connect(connectSD, (struct sockaddr *)&servAddr, sizeof(servAddr)) < 0
)
    {
        printf("Error: Connecting to server.\n");
        return -1;
    }
    printf("Enter Client Name: ");
    scanf("%s", client.name);
    printf("Enter Private Key: ");
    scanf("%llu", &privateKey);
    client.publicKey = findPublicKey(privateKey);
    printf("\nPublic key of client: < %llu >\n", client.publicKey);
    printf("Private key of client: < %llu >\n", privateKey);
    write(connectSD, &client, sizeof(client));
    if ((noOfBytesRead = read(connectSD, &server, sizeof(server))) < 0)
        return -1;
    printf("\nClient recieved Server Name: %s\n", server.name);
    printf("Client recieved Server Public Key : < %llu >\n", server.publicKey)
;
    sessionKey = findSessionKey(privateKey, server.publicKey);
    printf("Session Key of Client : < %llu >\n", sessionKey);
    printf("\n");
    printf("Enter -1 to close connection.\n");
    while (1)
    {
        printf("\nEnter number to send to server : ");
        scanf("%llu", &message);
        if (message == -1)
            break;
        tmp = message;
        printf("\tPlain text: %llu\n", message);
        message = encryption(message, sessionKey);
        printf("\tCipher text: %llu\n", message);
        write(connectSD, &message, sizeof(message));
        if ((noOfBytesRead = read(connectSD, &message, sizeof(message))) < 0)
            return -1;
        printf("\tServer sent to you cipher text : %llu\n", message);
        message = decryption(message, sessionKey);
        printf("\tServer sent to you double of %llu : %llu\n", tmp, message);
    }
    printf("\n");
    return 0;
}
```

3. Function.h

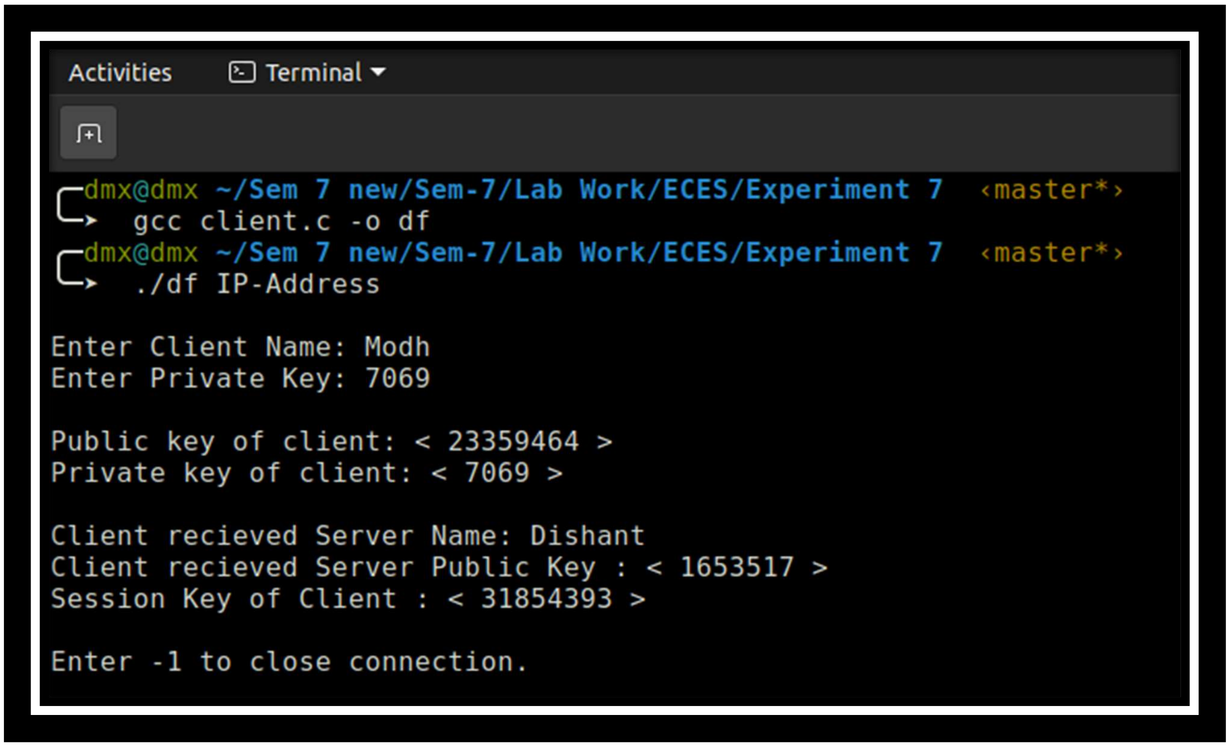
```
#define ll unsigned long long int
static ll a = 56;
static ll q = 34568357;
ll power(ll msg, ll key, ll modulas)
{
    ll ans = 1;
    for (ll i = 0; i < key; i++)
        ans = (msg * ans) % modulas;
    return ans;
}
typedef struct
{
    char name[10];
    ll publicKey;
} publicKeyInformation;
ll findPublicKey(ll privateKey)
{
    return power(a, privateKey, q);
}
ll findSessionKey(ll privateKey, ll publicKey)
{
    return power(publicKey, privateKey, q);
}
ll encryption(ll message, ll key)
{
    uint8_t keyI, msgI, iv, ctI;
    ll cipher = 0;
    iv = a % 256;
    printf("\t\tEncryption Process: plain=%2llx\tkey=%2llx\tIV0:%2x\n", messag
e,
           key, iv);
    for (int i = 0; i < 4; i++)
    {
        keyI = (uint8_t)((key & (255 << (8 * i))) >> (8 * i));
        msgI = (uint8_t)((message & (255 << (8 * i))) >> (8 * i));
        if (i)
            ctI = ctI ^ keyI ^ msgI;
        else
            ctI = iv ^ keyI ^ msgI;
        cipher = cipher | (ctI << 8 * i);
    }
    return cipher;
}
ll decryption(ll cipher, ll key)
{
    uint8_t keyI, ptI, iv, ctI, ctI_1 = 0;
    ll plain = 0;
    iv = a % 256;
    printf("\t\tDecryption Process: cipher=%2llx\tkey=%2llx\tIV0:%2x\n", ciphe
r,
           key, iv);
    for (int i = 0; i < 4; i++)
    {
        keyI = (uint8_t)((key & (255 << (8 * i))) >> (8 * i));
        ctI = (uint8_t)((cipher & (255 << (8 * i))) >> (8 * i));
        if (i)
            ptI = ctI_1 ^ keyI ^ ctI;
        else
            ptI = iv ^ keyI ^ ctI;
        plain = plain | (ptI << 8 * i);
        ctI_1 = ctI;
    }
    return plain;
}
```

Output



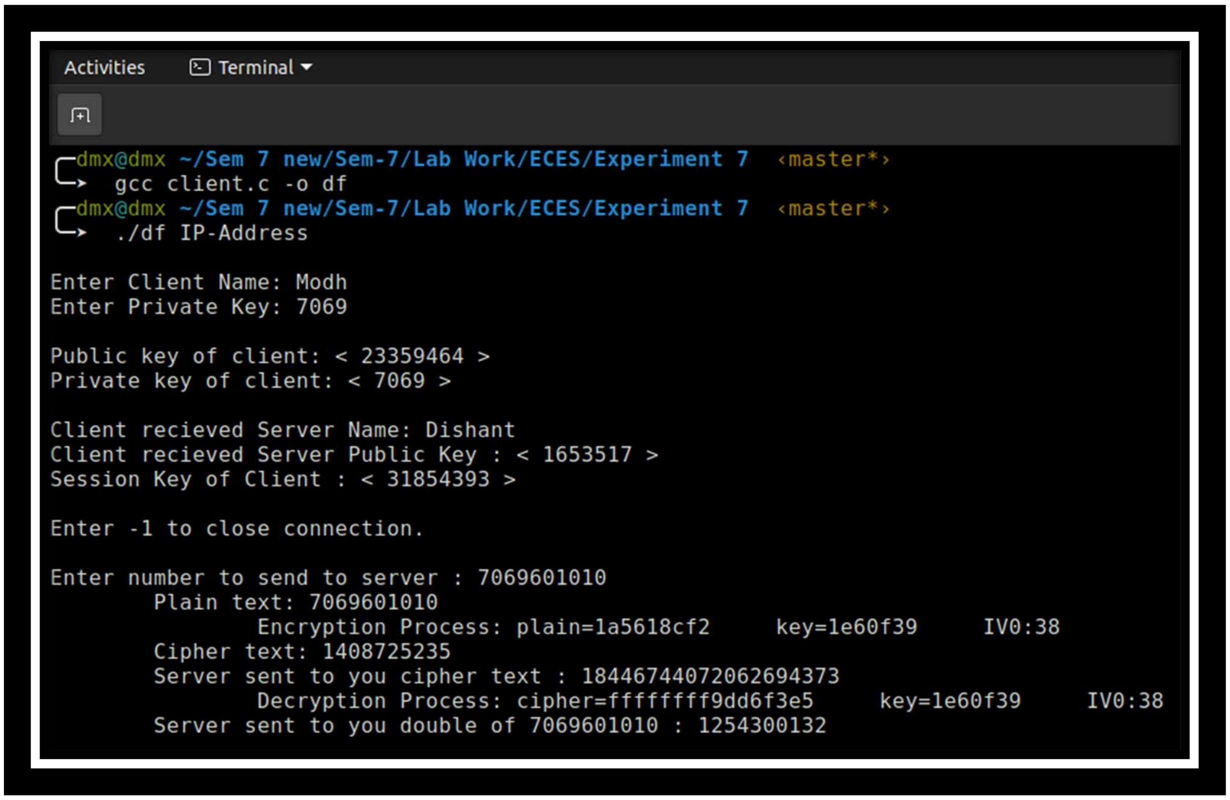
```
╭─dmx@dmx ~/Sem 7 new/Sem-7/Lab Work/ECES/Experiment 7  ‹master*›
╰─➤  gcc server.c
╭─dmx@dmx ~/Sem 7 new/Sem-7/Lab Work/ECES/Experiment 7  ‹master*›
╰─➤  ./a.out
Enter Server Name: Dishant
Enter Private Key: 9512

Public key of Server: < 1653517 >
Private key of Server: < 9512 >
```

Fig. Starting Server giving private key



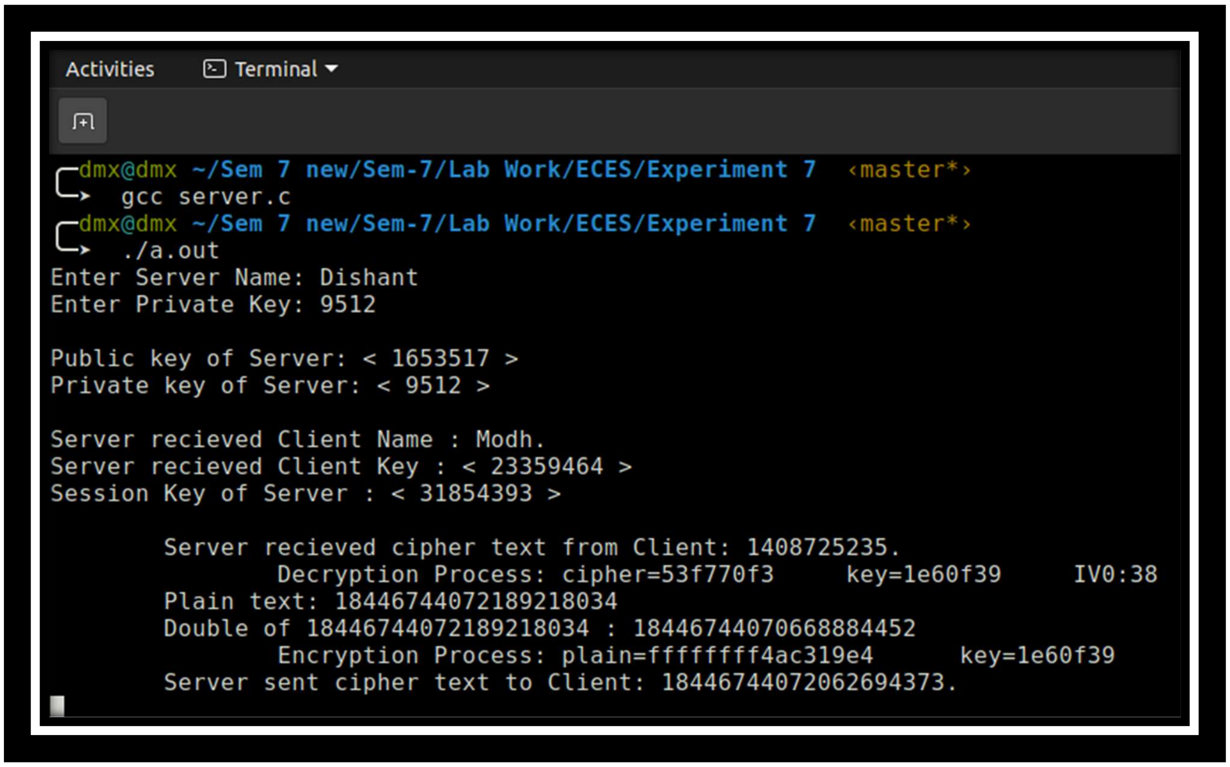
```
╭─dmx@dmx ~/Sem 7 new/Sem-7/Lab Work/ECES/Experiment 7  ‹master*›
╰─➤  gcc client.c -o df
╭─dmx@dmx ~/Sem 7 new/Sem-7/Lab Work/ECES/Experiment 7  ‹master*›
╰─➤  ./df IP-Address

Enter Client Name: Modh
Enter Private Key: 7069

Public key of client: < 23359464 >
Private key of client: < 7069 >

Client recieved Server Name: Dishant
Client recieved Server Public Key : < 1653517 >
Session Key of Client : < 31854393 >

Enter -1 to close connection.
```

Fig. Starting Client



```
┌dmx@dmx ~/Sem 7 new/Sem-7/Lab Work/ECES/Experiment 7  ‹master*›
└➤ gcc client.c -o df
┌dmx@dmx ~/Sem 7 new/Sem-7/Lab Work/ECES/Experiment 7  ‹master*›
└➤ ./df IP-Address

Enter Client Name: Modh
Enter Private Key: 7069

Public key of client: < 23359464 >
Private key of client: < 7069 >

Client recieved Server Name: Dishant
Client recieved Server Public Key : < 1653517 >
Session Key of Client : < 31854393 >

Enter -1 to close connection.

Enter number to send to server : 7069601010
        Plain text: 7069601010
                Encryption Process: plain=1a5618cf2     key=1e60f39     IV0:38
        Cipher text: 1408725235
        Server sent to you cipher text : 18446744072062694373
                Decryption Process: cipher=fffffff9dd6f3e5     key=1e60f39     IV0:38
        Server sent to you double of 7069601010 : 1254300132
```

Fig. Enter Message to send at server side



```
┌dmx@dmx ~/Sem 7 new/Sem-7/Lab Work/ECES/Experiment 7  ‹master*›
└➤ gcc server.c
┌dmx@dmx ~/Sem 7 new/Sem-7/Lab Work/ECES/Experiment 7  ‹master*›
└➤ ./a.out
Enter Server Name: Dishant
Enter Private Key: 9512

Public key of Server: < 1653517 >
Private key of Server: < 9512 >

Server recieved Client Name : Modh.
Server recieved Client Key : < 23359464 >
Session Key of Server : < 31854393 >

        Server recieved cipher text from Client: 1408725235.
                Decryption Process: cipher=53f770f3     key=1e60f39     IV0:38
        Plain text: 18446744072189218034
        Double of 18446744072189218034 : 18446744070668884452
                Encryption Process: plain=fffffff4ac319e4      key=1e60f39
        Server sent cipher text to Client: 18446744072062694373.
```

Fig. Server side