



Student Name: Riti		USN: 1SI19CS144	Batch No: B4	Date:
Evaluation:				
Write Up (10 marks)	Clarity in concepts (10 marks)	Implementation and execution of the algorithms (10 marks)	Viva (05 marks)	Total (35 marks)
Sl.No	Name of the Faculty In-Charge			Signature
1.	Dr. H K Vedamurthy			
2.	Dr. A H Shanthakumara			

Question No: 11

Implement RSA algorithm to process blocks of plaintext (refer Figure 9.7 of the text book), where plaintext is a string of characters and let the block size be two characters. (Note: assign a unique code to each plain text character i.e., a=00, A=26). The program should support the following.

- Accept string of characters as plaintext.
- Encryption takes plaintext and produces ciphertext characters
- Decryption takes ciphertext characters obtained in step ii and produces corresponding plaintext characters.
- Display the result after each step

Algorithm:

- Generate e, p, q using random number generator.
- Calculate n value , $n=p \times q$.
- Determine public and private keys (e, n) and (d, n) .
- Accept plain text in string format and assign numbers between 0 to 26 for characters (a to z)
- Plain text in decimal string $\{p_1, p_2, p_3, \dots\}$ is encrypted using public key as shown in fig 13.

$$\begin{aligned} C_1 &= P_1^e \bmod n \\ C_2 &= P_2^e \bmod n \\ &\vdots \end{aligned}$$

Fig 1

**Recovered
decimal text**

$$\begin{aligned} P_1 &= C_1^d \bmod n \\ P_2 &= C_2^d \bmod n \\ &\vdots \end{aligned}$$

Fig 2.

- Transmit the cipher text in decimal format to server using through sockets for decryption.

Server should decrypt the cipher text $\{c_1, c_2, c_3, \dots\}$ shown in fig 2. and print the string in character format back to screen.

CODE:

Server side:

```
#include <bits/stdc++.h>
#include <arpa/inet.h>
using namespace std;

int createServer(int port)
{
    int sersock=socket(AF_INET,SOCK_STREAM,0);
    struct sockaddr_in addr={AF_INET, htons(port), INADDR_ANY};
    bind(sersock,(struct sockaddr *)&addr, sizeof(addr));
    cout << "\nServer Online. Waiting for client. ..." << endl;
    listen(sersock, 5);
    int sock = accept(sersock, NULL, NULL);
    cout << "Connection Established." << endl;
    return sock;
}
// C = M^e mod n
int encrypt(int M, int PU[2])
{
    int C=1;
    for(int i=1; i<=PU[0]; i++)
    {
        C = (C * M) % PU[1];
    }
    return C;
}

// a=00, b=01, ... A=26, B=27...
int toInt(char c)
{
    return (c < 'a') ? (c-'A'+26) : (c-'a');
}

int main()
{
    int port;
    cout << "Enter port : "; cin >> port;
    int sock = createServer(port);
    int PU[2];
    recv(sock, &PU, sizeof(PU), 0); // receive public key from client
    cout << "\nPublic key received from client : {" << PU[0] << ", " << PU[1] << "}" << endl;
    string msg; // plaintext message
    cout << "\nEnter message to encrypt : "; cin >> msg;
    if(msg.length()% 2 != 0)
        msg+="x";
    for(int i=0; i<msg.length(); i+=2) // increment 2 for block
    {
```

```

        int M = toInt(msg[i])*100 + toInt(msg[i+1]); // block consist of two msg character
        cout << "\nPlaintext block : " << M << endl;
        int C = encrypt(M, PU);
        cout << "Encrypted text : " << C << endl;
        send(sock, &C, sizeof(C), 0); // send ciphertext to client
    }
    int stop = -1; // at end send -1 to tell client to stop
    send(sock, &stop, sizeof(stop), 0); //at end send -1 to client
    cout << "\nSent ciphertext to client." << endl << endl;
}

```

Client side:

```

#include <bits/stdc++.h>
#include <arpa/inet.h>
using namespace std;

int connectToServer(const char* ip, int port)
{
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
    if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 )
    {
        cout << "\nRun server program first." << endl; exit(0);
    }
    else
    {
        cout << "\nClient is connected to Server." << endl;
    }

    return sock;
}

int gcd(int a, int b)
{
    return b==0 ? a : gcd(b, a%b);
}

//  $M = C^d \bmod n$ 
int decrypt(int C, int PR[2])
{
    int M = 1;
    for(int i=1; i<=PR[0]; i++)
    {
        M = (M*C) % PR[1];
    }
    return M;
}

// a=00, b=01, ... A=26, B=27...
char toChar(int n)
{

```

```

        return (n >= 26) ? (n+'A'-26) : (n+'a');
    }

int main()
{
    char ip[50];
    int port;
    cout << "Enter Server's IP address: ";
    cin >> ip;
    cout << "Enter port : "; cin >> port;
    int sock = connectToServer(ip, port);
    int p,q;
    cout << "\nEnter two large prime numbers(>100) : "; cin >> p >> q;
    int n = p * q ; // should be greater than 5151 (since ZZ=5151)
    int phi = (p-1) * (q-1);
    srand(time(NULL));
    int e, d;
    do
    {
        e = rand()%(phi-2)+2;
    } while(gcd(e,phi) != 1);
    for(d=1; d<phi; d++)
    {
        if((d*e)%phi == 1)
            break;
    }
    int PU[2] = {e, n}; // public key
    int PR[2] = {d, n}; // private key
    cout << "\nPublic key , PU = {" << e << " , " << n << "}" << endl;
    cout << "Private key, PR = {" << d << " , " << n << "}" << endl;
    send(sock, &PU, sizeof(PU), 0); // send public key to server
    cout << "\nSent Public key to server." << endl;
    string msg = "";
    while (true)
    {
        int C; // ciphertext
        recv(sock, &C, sizeof(C), 0);
        if(C == -1)
            break; // at the end -1 will be received
        cout << "\nCiphertext received from server : " << C << endl;
        int M = decrypt(C,PR);
        cout << "Decrypted Text : " << M << endl;
        msg += toChar(M/100); // first char in block
        msg += toChar(M%100); // second char in block
    }
    cout << "\nDecrypted message : " << msg << endl << endl;
}

```

Output Screenshots:

```
user@linux-OptiPlex-5090: ~/Desktop/15I19CS144/RSA-11
user@linux-OptiPlex-5090:~/Desktop/15I19CS144/RSA-11$ g++ 11RSA_server.cpp
user@linux-OptiPlex-5090:~/Desktop/15I19CS144/RSA-11$ ./a.out
Enter port : 1234

Server Online. Waiting for client
Connection Established.

Public key received from client : {2623, 15943}

Enter message to encrypt : helloriti

Plaintext block : 704
Encrypted text : 5490

Plaintext block : 1111
Encrypted text : 9679

Plaintext block : 1417
Encrypted text : 7906

Plaintext block : 819
Encrypted text : 9942

Plaintext block : 823
Encrypted text : 1733

Sent ciphertext to client.

user@linux-OptiPlex-5090:~/Desktop/15I19CS144/RSA-11$
```

```
user@linux-OptiPlex-5090: ~/Desktop/15I19CS144/RSA-11
user@linux-OptiPlex-5090:~/Desktop/15I19CS144/RSA-11$ g++ 11RSA_client.cpp
user@linux-OptiPlex-5090:~/Desktop/15I19CS144/RSA-11$ ./a.out
Enter Server's IP address: 192.168.224.229
Enter port: 1234

Client is connected to Server.

Enter two large prime numbers(>100):107 149
Public key , PU ={2623,15943}
Private key , PU ={4079,15943}

Sent Public Key to server.

Ciphertext recieved from server : 5490
Decrpyted Text : 704

Ciphertext recieved from server : 9679
Decrpyted Text : 1111

Ciphertext recieved from server : 7906
Decrpyted Text : 1417

Ciphertext recieved from server : 9942
Decrpyted Text : 819

Ciphertext recieved from server : 1733
Decrpyted Text : 823

Decrpyted message :helloritix
user@linux-OptiPlex-5090:~/Desktop/15I19CS144/RSA-11$
```

