SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: Pragati Shankar	**USN: 1SI19CS090**	**Batch No: B2**	**Date: 03-01-2023**

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.	H K Vedamurthy	
2.	Gururaj S P	

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.

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

Algorithm:

1. Generate *e*,*p*,*q* using random number generator.
2. Calculate n value , n=p×q.
3. Determine public and private keys *(e,n) and (d,n).*
4. Accept plain text in string format and assign numbers between 0 to 26 for characters (a to z)
5. Plain text in decimal string {p1,p2,p3….} is encrypted using public key as shown in fig 13.

$$C_1 = P_1^e \bmod n$$
$$C_2 = P_2^e \bmod n$$
$$\vdots$$

Fig 1

Recovered decimal text

$$P_1 = C_1^d \bmod n$$
$$P_2 = C_2^d \bmod n$$
$$\vdots$$

Fig 2.

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

Server should decrypt the cipher text {c1,c2,c3…} shown in fig 2. and print the string in character format back to screen.

CODE:

Server side:

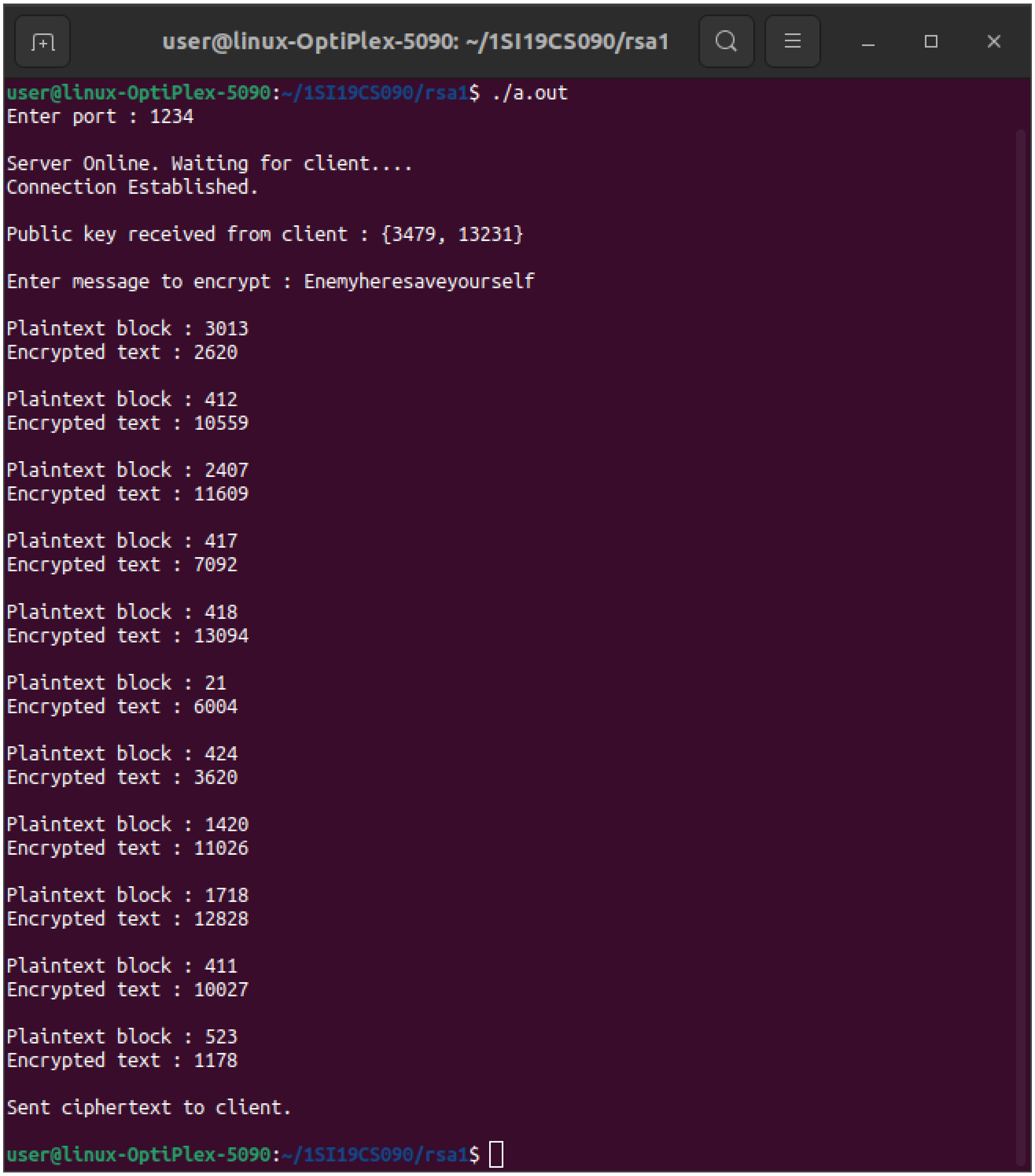
```
#include <bits/stdc++.h>
#include <arpa/inet.h>
using namespace std;

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;
}
```

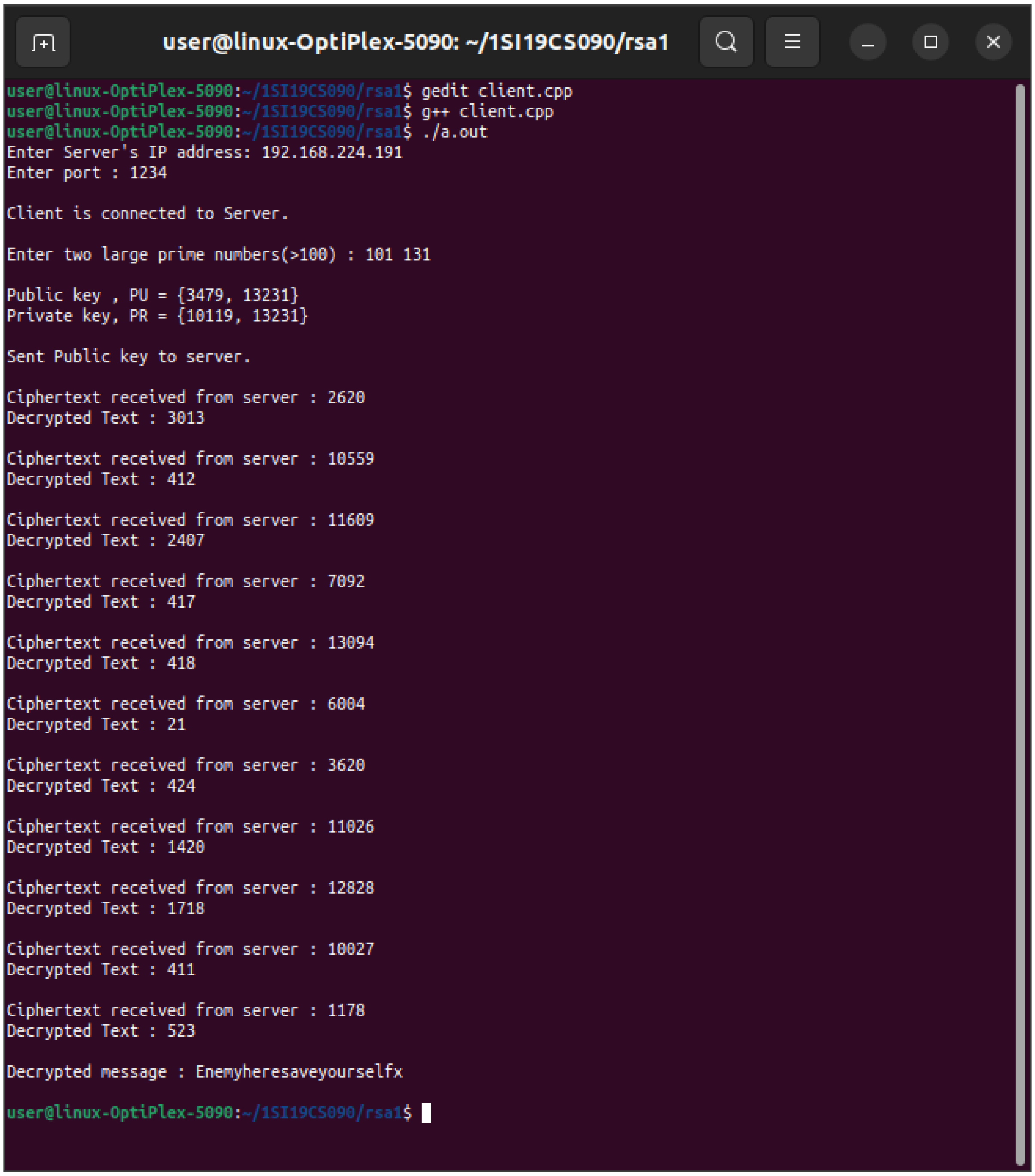
Client side:

```
# 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 mod 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:~/1SI19CS090/rsa1$ ./a.out
Enter port : 1234

Server Online. Waiting for client....
Connection Established.

Public key received from client : {3479, 13231}

Enter message to encrypt : Enemyheresaveyourself

Plaintext block : 3013
Encrypted text : 2620

Plaintext block : 412
Encrypted text : 10559

Plaintext block : 2407
Encrypted text : 11609

Plaintext block : 417
Encrypted text : 7092

Plaintext block : 418
Encrypted text : 13094

Plaintext block : 21
Encrypted text : 6004

Plaintext block : 424
Encrypted text : 3620

Plaintext block : 1420
Encrypted text : 11026

Plaintext block : 1718
Encrypted text : 12828

Plaintext block : 411
Encrypted text : 10027

Plaintext block : 523
Encrypted text : 1178

Sent ciphertext to client.

user@linux-OptiPlex-5090:~/1SI19CS090/rsa1$ 
```



```
user@linux-OptiPlex-5090:~/1SI19CS090/rsa1$ gedit client.cpp
user@linux-OptiPlex-5090:~/1SI19CS090/rsa1$ g++ client.cpp
user@linux-OptiPlex-5090:~/1SI19CS090/rsa1$ ./a.out
Enter Server's IP address: 192.168.224.191
Enter port : 1234

Client is connected to Server.

Enter two large prime numbers(>100) : 101 131

Public key , PU = {3479, 13231}
Private key, PR = {10119, 13231}

Sent Public key to server.

Ciphertext received from server : 2620
Decrypted Text : 3013

Ciphertext received from server : 10559
Decrypted Text : 412

Ciphertext received from server : 11609
Decrypted Text : 2407

Ciphertext received from server : 7092
Decrypted Text : 417

Ciphertext received from server : 13094
Decrypted Text : 418

Ciphertext received from server : 6004
Decrypted Text : 21

Ciphertext received from server : 3620
Decrypted Text : 424

Ciphertext received from server : 11026
Decrypted Text : 1420

Ciphertext received from server : 12828
Decrypted Text : 1718

Ciphertext received from server : 10027
Decrypted Text : 411

Ciphertext received from server : 1178
Decrypted Text : 523

Decrypted message : Enemyheresaveyourselfx

user@linux-OptiPlex-5090:~/1SI19CS090/rsa1$ 
```