RSA_Server.cpp:

```
#include<iostream>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<cstring>
#include<math.h>
#include<string>

using namespace std;

#define PORT 8080

int main(int argc, char const *argv[])
{
   int valread, server_fd, new_socket;
   int opt = 1;
   int e, n;
   double msgval, c;

   string message;
   char *hello = "Hi client! I'm your server.";
   char *pub = "Server has received the public key.";
   char buffer[1024] = {0};

   struct sockaddr_in address;

   int addrlen = sizeof(address);

   if((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
   {
       cout<<"ERROR::Socket failed\n";
       return -1;
   }

   // Forcefully attaching socket to the port 8080
   if(setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
   {
       cout<<"ERROR::Could not attach socket to port\n";
```

```
        return -1;
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    // Forcefully attaching socket to the port 8080
    if(bind(server_fd, (struct sockaddr *)&address, sizeof(address))<0)
    {
        cout<<"ERROR::Bind failed\n";
        return -1;
    }
    if(listen(server_fd, 3) < 0)
    {
        cout<<"ERROR::Listener set up failed\n";
        return -1;
    }
    if((new_socket = accept(server_fd, (struct sockaddr *)&address,
(socklen_t*)&addrlen))<0)
    {
        cout<<"ERROR::Could not accept connection\n";
        return -1;
    }

    valread = read(new_socket, buffer, 1024);
    cout<<buffer<<"\n";
    send(new_socket, hello, strlen(hello), 0);

    valread = read(new_socket, &e, 1024);
    cout<<"Encryption key for client e = "<<e<<"\n";
    send(new_socket, pub, strlen(pub), 0);

    valread = read(new_socket, &n, 1024);
    cout<<"Encryption key for client n = "<<n<<"\n";
    send(new_socket, pub, strlen(pub), 0);

    valread = read(new_socket, buffer, 1024);
    cout<<"\n"<<buffer<<"\n";
```

```
   cout<<"\nEnter a message for client:\n";
   cin>>message;

   for(int i=0; i<message.size(); i++)
   {
       msgval = message[i] - 'a';

       c = pow(msgval, e);
       c = fmod(c, n);

       cout<<"\nCharacter "<<message[i]<<" encrypted as "<<c<<"\n";

       send(new_socket, &c, sizeof(double), 0);
       valread = read(new_socket , buffer, 1024);
   }
   cout<<"\n"<<buffer<<"\n";

   return 0;
}
```

RSA_Client.cpp:

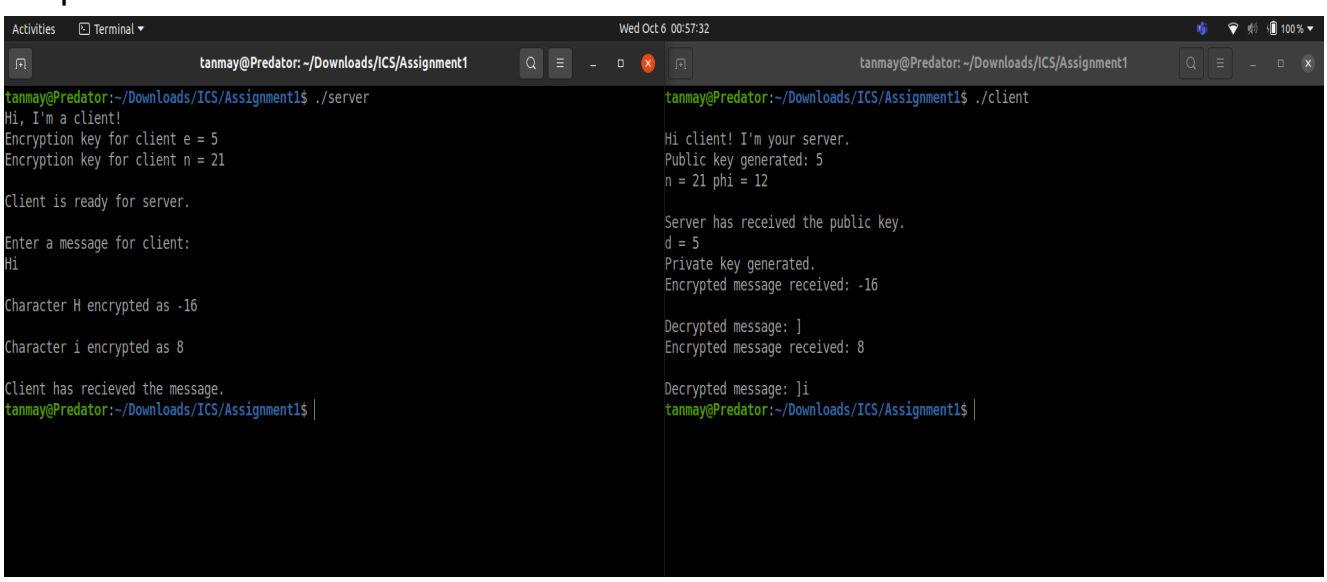
```
#include<iostream>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<cstring>
#include<math.h>
#include<string>

using namespace std;

#define PORT 8080

int gcd(int x, int y)
{
   int temp;
```

```
   while(1)
   {
       temp = x % y;
       if(temp == 0)
         return y;

       x = y;
       y = temp;
   }
}

int main(int argc, char const *argv[])
{
   int sock = 0, valread;
   int p = 3, q = 7, n, phi, e, d, k;
   double msgval, c;

   string message = "";
   char *hello = "Hi, I'm a client!";
   char *ready = "Client is ready for server.";
   char *received = "Client has recieved the message.";
   char buffer[1024] = {0};

   struct sockaddr_in serv_addr;

   if((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
   {
       cout<<"ERROR:: Couldn't create socket\n";
       return -1;
   }

   serv_addr.sin_family = AF_INET;
   serv_addr.sin_port = htons(PORT);

   // Convert IPv4 and IPv6 addresses from text to binary form
   if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0)
   {
       cout<<"ERROR:: Invalid address\n";
       return -1;
   }
```

```
   if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) <
0)
   {
       cout<<"ERROR::Connection Failed\n";
       return -1;
   }

   send(sock, hello, strlen(hello), 0);
   valread = read(sock , buffer, 1024);
   cout<<"\n"<<buffer<<"\n";

   n = p * q;
   phi = (p-1) * (q-1);

   e = 2;

   while(e < phi)
   {
       if(gcd(e, phi) == 1)
           break;
       else
           e++;
   }

   cout<<"Public key generated: "<<e<<"\n";
   cout<<"n = "<<n<<" phi = "<<phi<<"\n";

   send(sock, &e, sizeof(int), 0);
   valread = read(sock , buffer, 1024);

   send(sock, &n, sizeof(int), 0);
   valread = read(sock , buffer, 1024);
   cout<<"\n"<<buffer<<"\n";

   k = 2;
   d = (1 + (k*phi))/e;
   cout<<"d = "<<d<<"\n";

   cout<<"Private key generated.\n";
```

```
    send(sock, ready, strlen(ready), 0);

    while((valread = read(sock , &c, 1024)))
    {
        cout<<"Encrypted message received: "<<c<<"\n";

        msgval = pow(c, d);
        msgval = fmod(msgval, n);

        message += msgval + 'a';

        cout<<"\nDecrypted message: "<<message<<"\n";

        send(sock, received, strlen(received), 0);
    }



    return 0;
}
```

Output:



RSA_TextInput.java:

```java
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;

class RSA
{
   private final BigInteger N;
   private final BigInteger e;
   private final BigInteger d;

   public RSA()
   {
       Random r = new Random();
       int bitlength = 1024;
       BigInteger p = BigInteger.probablePrime(bitlength, r);
       BigInteger q = BigInteger.probablePrime(bitlength, r);
       N = p.multiply(q);
       BigInteger phi =
p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
       e = BigInteger.probablePrime(bitlength / 2, r);
       while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi)
< 0)
       {
           e.add(BigInteger.ONE);
       }
       d = e.modInverse(phi);
   }

   public RSA(BigInteger e, BigInteger d, BigInteger N)
   {
       this.e = e;
       this.d = d;
       this.N = N;
   }

   @SuppressWarnings("deprecation")
   public static void main(String[] args) throws IOException
   {
       RSA rsa = new RSA();
```

```
        DataInputStream in = new DataInputStream(System.in);
        String teststring;
        System.out.print("Enter the plain text:- ");
        teststring = in.readLine();
        System.out.println("Encrypting String:- " + teststring);
        System.out.println("String in Bytes:- " +
bytesToString(teststring.getBytes()));
        byte[] encrypted = rsa.encrypt(teststring.getBytes());
        byte[] decrypted = rsa.decrypt(encrypted);
        System.out.println("Decrypting Bytes:- " +
bytesToString(decrypted));
        System.out.println("Decrypted String:- " + new String(decrypted));
    }

    private static String bytesToString(byte[] encrypted)
    {
        StringBuilder test = new StringBuilder();
        for (byte b : encrypted)
        {
            test.append(Byte.toString(b));
        }
        return test.toString();
    }

    // Encrypt message
    public byte[] encrypt(byte[] message)
    {
        return (new BigInteger(message)).modPow(e, N).toByteArray();
    }

    // Decrypt message
    public byte[] decrypt(byte[] message)
    {
        return (new BigInteger(message)).modPow(d, N).toByteArray();
    }
}
```

Output:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

D:\Code\ICS\rsa\java>javac 43260_RSA_TextInput.java

D:\Code\ICS\rsa\java>java RSA
Enter the plain text:- This is plain text
Encrypting String:- This is plain text
String in Bytes:- 84104105115321051153211210897105110321161011201 16
Decrypting Bytes:- 84104105115321051153211210897105110321161011201 16
Decrypted String:- This is plain text

D:\Code\ICS\rsa\java>_
```

RSA_Server.java:

```
import java.io.*;
import java.net.*;
import java.math.BigInteger;

class RSA_Server {

   public static void main(String[] args) throws IOException
   {
       // Generate key pairs for server
       RsaUtil rsaUtil = new RsaUtil();
       BigInteger d = rsaUtil.getD();
       BigInteger e = rsaUtil.getE();
       BigInteger n = rsaUtil.getN();

       // Create socket
       ServerSocket ss = new ServerSocket(5003);
       Socket cs = ss.accept();

       // Reserve memory on server for sending and receiving messages
       BufferedReader fromserver = new BufferedReader(new
InputStreamReader(System.in));
       BufferedReader fromclient = new BufferedReader(new
InputStreamReader(cs.getInputStream()));
       DataInputStream fromclient_byte = new
DataInputStream(cs.getInputStream());
       DataOutputStream toclient_byte = new
DataOutputStream(cs.getOutputStream());
       PrintWriter toclient = new PrintWriter(cs.getOutputStream(), true);

       // Exchange of public keys
       toclient.println(d);
       toclient.println(n);

       String from_client_d_string = fromclient.readLine();
       String from_client_n_string = fromclient.readLine();

       BigInteger from_client_d = new BigInteger(from_client_d_string);
       BigInteger from_client_n = new BigInteger(from_client_n_string);
```

```
        System.out.println("\nReceived public key from client\n");

        // The client has communicated, so send a welcome message to him
        String message = "Hi client";
        byte[] encrypted = rsaUtil.encrypt(message.getBytes(),
from_client_d, from_client_n);
        System.out.println("Encryped Server:- " + encrypted);
        toclient_byte.writeInt(encrypted.length);
        toclient_byte.write(encrypted);

        while(true) {
            int length = fromclient_byte.readInt();
            if(length > 0) {
                byte[] fromclient_enc = new byte[length];
                fromclient_byte.readFully(fromclient_enc, 0,
fromclient_enc.length);
                System.out.println("Encrypted Client:- " + fromclient_enc);
                byte[] decrypted = rsaUtil.decrypt(fromclient_enc, e, n);
                String decrypted_string = rsaUtil.bytesToString(decrypted);
                System.out.println("Decrypted Client:- " + new
String(decrypted));
                if(new String(decrypted).equalsIgnoreCase("bye")) {
                    break;
                }
                System.out.print("Server:- ");
                String s = fromserver.readLine();
                encrypted = rsaUtil.encrypt(s.getBytes(), from_client_d,
from_client_n);
                System.out.println("Encrypted Server:- " + encrypted);
                toclient_byte.writeInt(encrypted.length);
                toclient_byte.write(encrypted);
            }
        }


        toclient.close();
        fromclient.close();
        cs.close();
        ss.close();
```

```
    }
}
```

RSA_Client.java:

```
import java.io.*;
import java.net.*;
import java.math.BigInteger;

class RSA_Client {
   public static void main(String[] args) throws IOException,
UnknownHostException {

       RsaUtil rsaUtil = new RsaUtil();
       BigInteger d = rsaUtil.getD();
       BigInteger e = rsaUtil.getE();
       BigInteger n = rsaUtil.getN();

       Socket cs = new Socket("LAPTOP-GA8C0AII", 5003);

       // reserve memory for communication with server
       BufferedReader fromclient = new BufferedReader(new
InputStreamReader(System.in));
       BufferedReader fromserver = new BufferedReader(new
InputStreamReader(cs.getInputStream()));
       DataOutputStream toserver_byte = new
DataOutputStream(cs.getOutputStream());
       DataInputStream fromserver_byte = new
DataInputStream(cs.getInputStream());
       PrintWriter toserver = new PrintWriter(cs.getOutputStream(), true);

       // Once connected, the client sees a welcome message on his machine
and he tends to respond and the conversation
       // begins
       String from_server_d_string = fromserver.readLine();
       String from_server_n_string = fromserver.readLine();

       BigInteger from_server_d = new BigInteger(from_server_d_string);
```

```
        BigInteger from_server_n = new BigInteger(from_server_n_string);

        System.out.println("\nReceived public key from server\n");

        toserver.println(d);
        toserver.println(n);

        while(true) {
            int length = fromserver_byte.readInt();
            if(length > 0) {
                byte[] fromserver_enc = new byte[length];
                fromserver_byte.readFully(fromserver_enc, 0,
fromserver_enc.length);
                System.out.println("Server:- " + fromserver_enc);
                byte[] decrypted = rsaUtil.decrypt(fromserver_enc, e, n);
                String decrypted_string = rsaUtil.bytesToString(decrypted);
                System.out.println("Decrypted Server:- " + new
String(decrypted));
                if(new String(decrypted).equalsIgnoreCase("bye")) {
                    break;
                }
            }
            System.out.print("Client:- ");
            String s = fromclient.readLine();
            byte[] encrypted = rsaUtil.encrypt(s.getBytes(), from_server_d,
from_server_n);
            System.out.println("Encrypted Client:- " + encrypted);
            toserver_byte.writeInt(encrypted.length);
            toserver_byte.write(encrypted);
        }

        toserver.close();
        fromserver.close();
        fromclient.close();
        cs.close();
    }
}
```

RSA_Util.java:

```
import java.math.BigInteger;
import java.util.Random;

class RsaUtil
{
   private final BigInteger N;
   private final BigInteger e;
   private final BigInteger d;

   public BigInteger getN() {
       return N;
   }

   public BigInteger getE() {
       return e;
   }

   public BigInteger getD() {
       return d;
   }

   public RsaUtil()
   {
       Random r = new Random();
       int bitlength = 1024;
       BigInteger p = BigInteger.probablePrime(bitlength, r);
       BigInteger q = BigInteger.probablePrime(bitlength, r);
       N = p.multiply(q);
       BigInteger phi =
p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
       e = BigInteger.probablePrime(bitlength / 2, r);
       while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi)
< 0)
       {
           e.add(BigInteger.ONE);
       }
       d = e.modInverse(phi);
   }
```

```java
    public String bytesToString(byte[] encrypted)
    {
        StringBuilder test = new StringBuilder();
        for (byte b : encrypted)
        {
            test.append(Byte.toString(b));
        }
        return test.toString();
    }

    // Encrypt message
    public byte[] encrypt(byte[] message, BigInteger e, BigInteger N)
    {
        return (new BigInteger(message)).modPow(e, N).toByteArray();
    }

    // Decrypt message
    public byte[] decrypt(byte[] message, BigInteger d, BigInteger N)
    {
        return (new BigInteger(message)).modPow(d, N).toByteArray();
    }
}
```

Output:

C:\Windows\System32\cmd.exe - java RSA_Server

```
D:\Code\ICS\rsa\java>java RSA_Server

Received public key from client

Encryped Server:- [B@1c7c054
Encrypted Client:- [B@12204a1
Decrypted Client:- Hi server
Server:- How are you?
Encrypted Server:- [B@a298b7
Encrypted Client:- [B@14991ad
Decrypted Client:- I am good! What about you?
Server:- Me too!
Encrypted Server:- [B@d93b30
```

C:\Windows\System32\cmd.exe - java RSA_Client

```
D:\Code\ICS\rsa\java>java RSA_Client

Received public key from server

Server:- [B@1c7c054
Decrypted Server:- Hi client
Client:- Hi server
Encrypted Client:- [B@12204a1
Server:- [B@a298b7
Decrypted Server:- How are you?
Client:- I am good! What about you?
Encrypted Client:- [B@14991ad
Server:- [B@d93b30
Decrypted Server:- Me too!
Client:-
```