NAME: **Tejas Dabgar**

Roll NO: **IT015**

EXPERIMENT 6

Aim: Perform Encryption, Authentication and both using RSA.

Apparatus: Netbeans

->RSA.java :

```
import java.util.Scanner;
import static java.lang.Math.pow;


public class RSA {
  private long p, q, d, n, e;
  RSA(long p, long q) {
    this.p = p;
    this.q = q;
    calculateKeys();
  }
  RSA(){}
  static RSA getFromPublicKey(long n, long e) {
    RSA rsa = new RSA();
    rsa.e = e;
    rsa.n = n;
    return rsa;
```

```
}

public long getD() {
  return d;
}

public long getN() {
  return n;
}

public long getE() {
  return e;
}

void calculateKeys() {
  n = p * q;
  long z = (p - 1) * (q - 1);
  for (e = 2; e < z; e++) {
    if (gcd(e, z) == 1) {
      break;
    }
  }
  for (long i = 1;; i++) {
    long x = 1 + (i * z);
```

```
    if (x % e == 0 && x / e != e) {
      d = x / e;
      break;
    }
  }
}

long gcd(long e, long z) {
  if (e == 0) {
    return z;
  } else {
    return gcd(z % e, e);
  }
}

public static void main(String[] args) {
  long p, q;
  Scanner sc = new Scanner(System.in);
  p = sc.nextInt();
  q = sc.nextInt();
  RSA rsa = new RSA(p, q);
  System.out.println("Private Key: {" + rsa.getN() + "," + rsa.getD() + "}");
  System.out.println("Public Key: {" + rsa.getN() + "," + rsa.getE() + "}");
```

```
  String data = "hello";
  String ed = rsa.encrypt(data);
  String dd = rsa.decrypt(ed);
  System.out.println(data);
  System.out.println(ed);
  System.out.println(dd);
}

long encryptChar(long c) {
  return power(c, e, n);
}

long decryptChar(long c) {
  return power(c, d, n);
}

String encrypt(String message) {
  StringBuilder buffer = new StringBuilder();
  for (int i = 0; i < message.length(); i++) {
    buffer.append((char) encryptChar(message.charAt(i)));
  }
  return buffer.toString();
}
```

```
String decrypt(String message) {
  StringBuilder buffer = new StringBuilder();
  for (int i = 0; i < message.length(); i++) {
    buffer.append((char) decryptChar(message.charAt(i)));
  }
  return buffer.toString();
}

static long power(long x, long y, long p) {
  long res = 1;
  x = x % p;
  if (x == 0) {
    return 0;
  }
  while (y > 0) {
    if ((y & 1) == 1) {
      res = (res * x) % p;
    }
    y = y >> 1;
    x = (x * x) % p;
  }
  return res;
}
```

}

->Server.java :

```
import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.InetAddress;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;


public class Server {

  Map<String, Key> keys;

  private final ServerSocket server;

  private final RSA rsa;

  int port;

  Server(int port, int p, int q) throws IOException{

    this.port = port;

    rsa = new RSA(p, q);

    server = new ServerSocket(port);

    keys = new HashMap();

    System.out.println("Private Key: {" + rsa.getN() + "," + rsa.getD() + "}");

    System.out.println("Public Key: {" + rsa.getN() + "," + rsa.getE() + "}");
```

```
}
void communicate() throws IOException {
  while(true) {
    Socket socket = server.accept();
    new Thread(() -> {
      Socket client = socket;
      DataOutputStream writer;
      DataInputStream reader;
      try {
        writer = new DataOutputStream(client.getOutputStream());
        writer.writeLong(rsa.getN());
        writer.writeLong(rsa.getE());
        reader = new DataInputStream(client.getInputStream());
        String uname;
        long n,e;
        uname = reader.readUTF();
        n = reader.readLong();
        e = reader.readLong();
        RSA clientKey = RSA.getFromPublicKey(n, e);
        System.out.println("From client: " + client.getInetAddress().getHostAddress());
        System.out.println("Received: {" + n + "," + e + "}" );
        storeKey(client, uname, n, e);

        String encrypted = reader.readUTF();
```

```
        System.out.println("Received from client: " + encrypted);
        String message = rsa.decrypt(encrypted);
        System.out.println("Decrypted using server's private key: " + message);
        encrypted = clientKey.encrypt(message);
        System.out.println("Encrypted using client's public key: " + encrypted);
        writer.writeUTF(encrypted);
        client.close();
      } catch (IOException ex) {
      }
    }).start();
  }
}
void storeKey(Socket client, String uname, long n, long e) {
    keys.put(uname, new Key(client.getInetAddress(), uname, n, e));
}
public static void main(String[] args) throws IOException {
  int p,q;
  Scanner sc = new Scanner(System.in);
  System.out.println("Enter p(A prime number bigger then 13");
  p = sc.nextInt();
  System.out.println("Enter q(A prime number bigger then 13");
  q = sc.nextInt();
  Server server = new Server(6580, p, q);
  server.communicate();
```

```
  }
  class Key {
    private final String uname;
    private final InetAddress address;
    private final long n;
    private final long e;
    public Key(InetAddress address, String uname, long n, long e) {
      this.address = address;
      this.uname = uname;
      this.n = n;
      this.e = e;
    }
    public long getN() {
      return n;
    }
    public long getE() {
      return e;
    }
    public String getUname() {
      return uname;
    }
  }
}
```

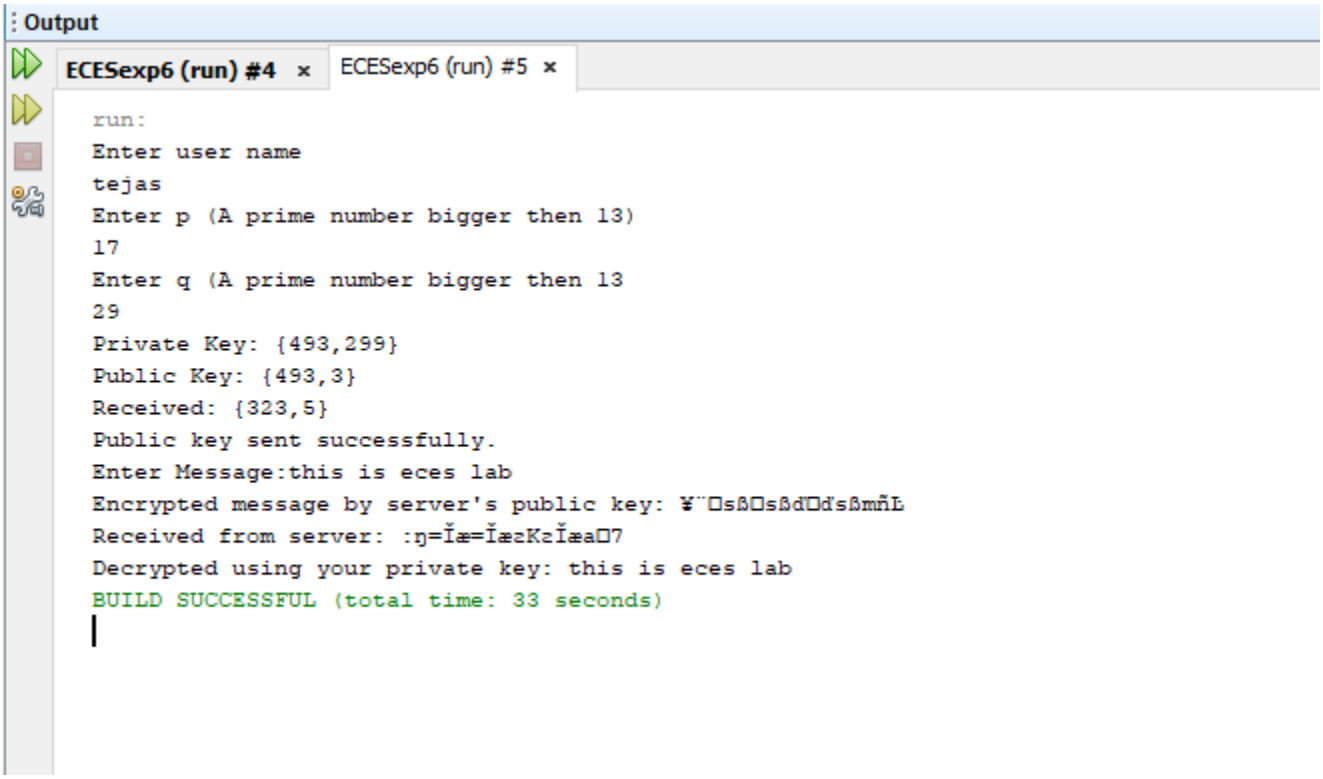
->Client.java :

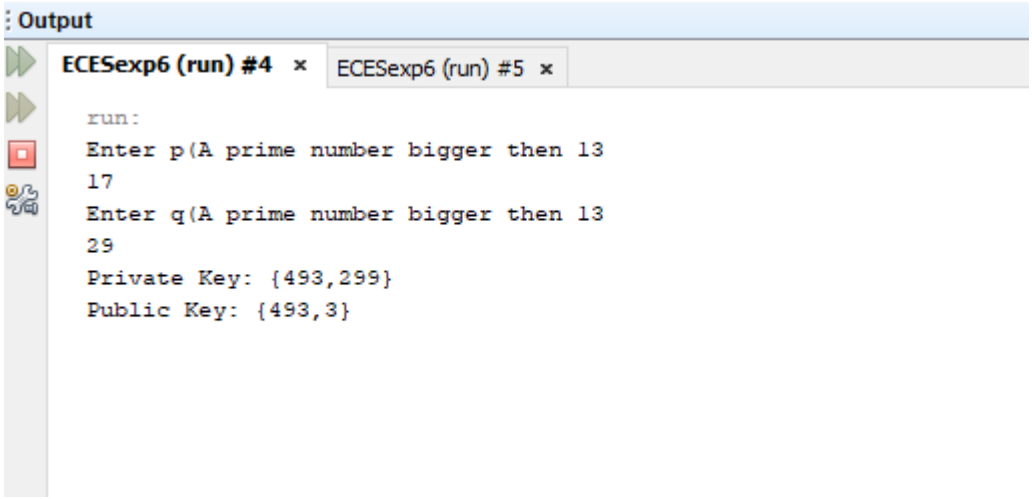
```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;

public class Client {

  private final Socket client;
  private final RSA rsa;
  private final String uname;
  private RSA serverKey;
  Client(String ip, int port, long p, long q, String uname) throws IOException {
    client = new Socket(ip, port);
    rsa = new RSA(p, q);
    this.uname = uname;
    System.out.println("Private Key: {" + rsa.getN() + "," + rsa.getD() + "}");
    System.out.println("Public Key: {" + rsa.getN() + "," + rsa.getE() + "}");
  }

  void communicate() {
    long n, e;
```

```
DataOutputStream writer;
DataInputStream reader;
try {
  reader = new DataInputStream(client.getInputStream());
  writer = new DataOutputStream(client.getOutputStream());
  n = reader.readLong();
  e = reader.readLong();
  serverKey = RSA.getFromPublicKey(n, e);
  System.out.println("Received: {" + n + "," + e + "}");
  writer.writeUTF(uname);
  writer.writeLong(rsa.getN());
  writer.writeLong(rsa.getE());
  System.out.println("Public key sent successfully.");
  System.out.print("Enter Message:");
  Scanner sc = new Scanner(System.in);
  String message = sc.nextLine();
  String encrypted = serverKey.encrypt(message);
  System.out.println("Encrypted message by server's public key: " + encrypted);
  writer.writeUTF(encrypted);

  message = reader.readUTF();
  System.out.println("Received from server: " + message);
  String decrypted = rsa.decrypt(message);
  System.out.println("Decrypted using your private key: " + decrypted);
```

```
    client.close();
   } catch (IOException ex) {
   }
  }

  public static void main(String[] args) throws IOException {
   long p,q;
   Scanner sc = new Scanner(System.in);
   System.out.println("Enter user name");
   String uname = sc.nextLine();
   System.out.println("Enter p (A prime number bigger then 13)");
   p = sc.nextInt();
   System.out.println("Enter q (A prime number bigger then 13");
   q = sc.nextInt();
   Client client = new Client("localhost", 6578, p, q, uname);
   client.communicate();
  }
}
```

->OUTPUT:


Output
ECESexp6 (run) #4
ECESexp6 (run) #5
run:
Enter user name
tejas
Enter p (A prime number bigger then 13)
17
Enter q (A prime number bigger then 13
29
Private Key: {493,299}
Public Key: {493,3}
Received: {323,5}
Public key sent successfully.
Enter Message:this is eces lab
Encrypted message by server's public key: ¥¨□sß□sßd□d'sßmñL
Received from server: :ŋ=Ĭæ=ĬæzKzĬæa□7
Decrypted using your private key: this is eces lab
BUILD SUCCESSFUL (total time: 33 seconds)


Output
ECESexp6 (run) #4
ECESexp6 (run) #5
run:
Enter p(A prime number bigger then 13
17
Enter q(A prime number bigger then 13
29
Private Key: {493,299}
Public Key: {493,3}