Experiment-5

Name: Dishant Modh

Roll No.: IT076

Aim: Write a C/C++/Java program to generate and exchange public keys using client server mechanism.

Code:

Program1: Server Side

```
import java.net.*;
import java.io.*;
import java.util.*;
public class Server {
        public static void main(String[] args) throws IOException
        {
                try {
                        int port = 8088;
                        Scanner sc = new Scanner(System.in);
                        // Server Key
                        System.out.println("Enter the value of server key:- ");
                        int b = sc.nextInt();

                        // Client p, g, and key
                        double clientP, clientG, clientA, B, Bdash;
                        String Bstr;

                        // Established the Connection
                        ServerSocket serverSocket = new ServerSocket(port);
```

```
            System.out.println("Waiting for client on port " +
serverSocket.getLocalPort() + "...");
            Socket server = serverSocket.accept();
            System.out.println("Just connected to " +
server.getRemoteSocketAddress());

            // Server's Private Key
            System.out.println("From Server : Private Key = " + b);

            // Accepts the data from client
            DataInputStream in = new DataInputStream(server.getInputStream());

            clientP = Integer.parseInt(in.readUTF()); // to accept p
            System.out.println("From Client : P = " + clientP);

            clientG = Integer.parseInt(in.readUTF()); // to accept g
            System.out.println("From Client : G = " + clientG);

            clientA = Double.parseDouble(in.readUTF()); // to accept A
            System.out.println("From Client : Public Key = " + clientA);

            B = ((Math.pow(clientG, b)) % clientP); // calculation of B
            Bstr = Double.toString(B);

            // Sends data to client
            // Value of B
            OutputStream outToclient = server.getOutputStream();
            DataOutputStream out = new DataOutputStream(outToclient);

            out.writeUTF(Bstr); // Sending B
```

```
				Bdash = ((Math.pow(clientA, b)) % clientP); // calculation of Bdash

				System.out.println("Secret Key to perform Symmetric Encryption = "
										+ Bdash);
				server.close();
			}

			catch (SocketTimeoutException s) {
				System.out.println("Socket timed out!");
			}
			catch (IOException e) {
			}
		}
}
```

Program2: Client Side

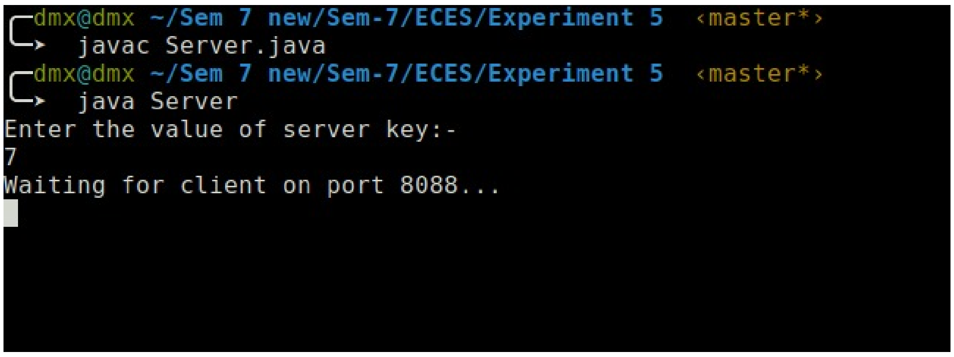
```
import java.net.*;
import java.io.*;
import java.util.*;
public class Client {
		public static void main(String[] args)
		{
			try {
				String pstr, gstr, Astr;
				String serverName = "localhost";
				int port = 8088;
				Scanner sc = new Scanner(System.in);
```

```
// Declare p, g, and Key of client
System.out.println("Enter the value of p:- ");
int p = sc.nextInt();
System.out.println("Enter the value of g:- ");
int g = sc.nextInt();
System.out.println("Enter the value of client key:- ");
int a = sc.nextInt();
double Adash, serverB;

// Established the connection
System.out.println("Connecting to " + serverName
                                        + " on port " + port);
Socket client = new Socket(serverName, port);
System.out.println("Just connected to "
                                        + client.getRemoteSocketAddress());

// Sends the data to client
OutputStream outToServer = client.getOutputStream();
DataOutputStream out = new DataOutputStream(outToServer);

pstr = Integer.toString(p);
out.writeUTF(pstr); // Sending p

gstr = Integer.toString(g);
out.writeUTF(gstr); // Sending g

double A = ((Math.pow(g, a)) % p); // calculation of A
Astr = Double.toString(A);
```

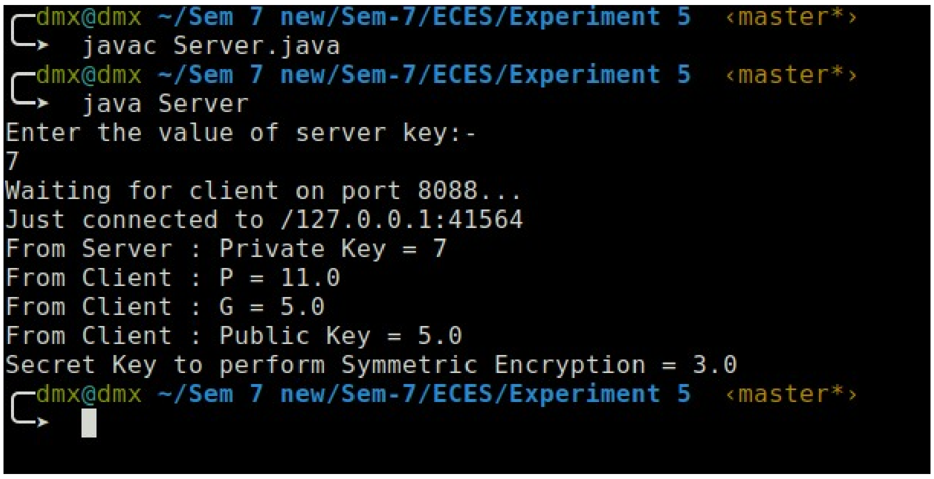
```
                    out.writeUTF(Astr); // Sending A

                    // Client's Private Key
                    System.out.println("From Client : Private Key = " + a);

                    // Accepts the data
                    DataInputStream in = new DataInputStream(client.getInputStream());

                    serverB = Double.parseDouble(in.readUTF());
                    System.out.println("From Server : Public Key = " + serverB);

                    Adash = ((Math.pow(serverB, a)) % p); // calculation of Adash

                    System.out.println("Secret Key to perform Symmetric Encryption = "
                                                                + Adash);
                    client.close();
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
        }
}
```

Output:

Server Side:

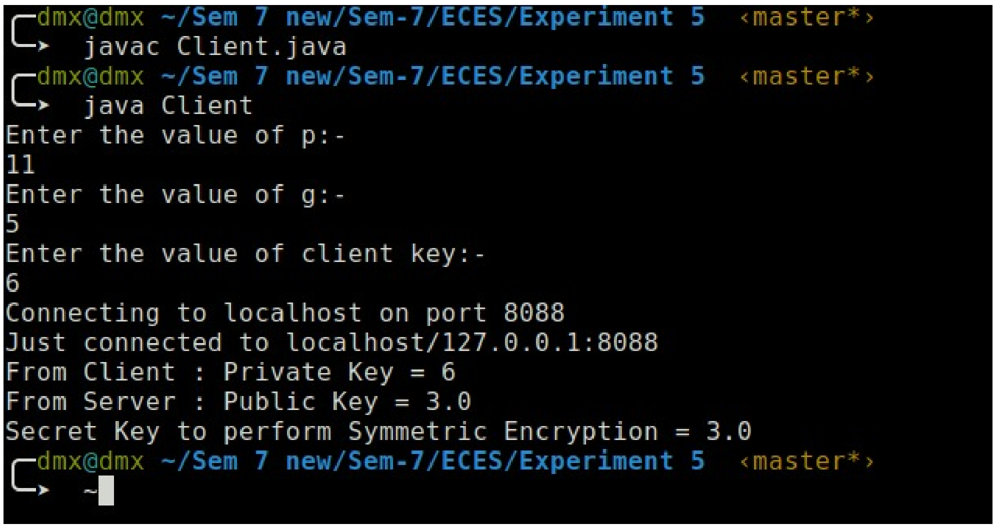


```
┌─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Experiment 5  ‹master*›
└─➤  javac Server.java
┌─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Experiment 5  ‹master*›
└─➤  java Server
Enter the value of server key:-
7
Waiting for client on port 8088...
```



```
┌─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Experiment 5  ‹master*›
└─➤  javac Server.java
┌─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Experiment 5  ‹master*›
└─➤  java Server
Enter the value of server key:-
7
Waiting for client on port 8088...
Just connected to /127.0.0.1:41564
From Server : Private Key = 7
From Client : P = 11.0
From Client : G = 5.0
From Client : Public Key = 5.0
Secret Key to perform Symmetric Encryption = 3.0
┌─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Experiment 5  ‹master*›
└─➤
```

Client Side:



```
┌─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Experiment 5  ‹master*›
└─➤  javac Client.java
┌─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Experiment 5  ‹master*›
└─➤  java Client
Enter the value of p:-
11
Enter the value of g:-
5
Enter the value of client key:-
6
Connecting to localhost on port 8088
Just connected to localhost/127.0.0.1:8088
From Client : Private Key = 6
From Server : Public Key = 3.0
Secret Key to perform Symmetric Encryption = 3.0
┌─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Experiment 5  ‹master*›
└─➤  ~
```