VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

Lab Assessment-IV, OCTOBER 2020

B.Tech., Fall-2020-2021

NAME	PRIYAL BHARDWAJ
REG. NO.	18BIT0272
COURSE CODE	ITE3001
COURSE NAME	DATA COMMUNICATION & COMPUTER NETWORKS
SLOT	L15+L16
FACULTY	Prof. DINAKARAN MURUGANANDAM

1. Implement a TCP/IP socket-based ATM System. Make the server to maintain the customer details (name, card no, pin and balance). When a client wants to withdraw amount, validate his login with card no & pin, display a welcome message and perform the withdraw operation if he is having sufficient balance or display a warning message.

- Maintain customer details as in Database
- Creating TCP sockets for server and client
- Validate of user with database and verification with the balance for transfer of IP packets across multiple networks

TCP_Server.java

```
import java.net.*;
import java.io.*;
public class TCP_Server {
    public static void main(String args[]) {
        int count=0;
        String a[][] = {{"Priyal","112233","1234","10000"},
{"Piyush","123123","4321","15000"}, {"Mitushi","142312","2201","1200"}};
        int user=0;
        int port = 5046; // just a random port number between 1025 and 65535
        try {
            ServerSocket ss = new ServerSocket(port);
            System.out.println("Waiting for a user...");
            Socket socket = ss.accept();
            InputStream sin = socket.getInputStream();
            OutputStream sout = socket.getOutputStream();
            BufferedReader keyboard = new BufferedReader(new InputStreamReader
(System.in));
            DataInputStream in = new DataInputStream(sin);
            DataOutputStream out = new DataOutputStream(sout);
            String cardno = null;
            String pin=null;
            while(true) {
                cardno = in.readUTF();
                pin=in.readUTF();
                for(int i=0;i<3;i++) {
                    if(a[i][1].equals(cardno)&&a[i][2].equals(pin)) {
                        count++;
                        user=i;
                        break;
                    }
                }
                if(count==1) {
                    String msg="Login successful";
                    String q="Enter withdraw amount";
                    out.writeUTF(msg);
                    System.out.println("User login successful");
                    out.writeUTF(q);
                    System.out.println("Starting Withdraw process");
```

```
                    System.out.println("Checking account balance");
                    String amt=in.readUTF();
                    String acc1=a[user][3];
                    int amt1=Integer.parseInt(amt);
                    int acc=Integer.parseInt(acc1);
                    int bal=acc-amt1;
                    if(bal>=500) {
                        String bal1=String.valueOf(bal);
                        a[user][3]=bal1;
                        String msg3="Withdraw complete";
                        out.writeUTF(msg3);
                        out.writeUTF(bal1);
                        System.out.println("Withdraw complete");
                    }
                    else {
                        String msg4="Insufficient Balance";
                        out.writeUTF(msg4);
                        System.out.println("Insufficient Balance");
                    }
                }
                else {
                    String msg1="Login unsuccessful";
                    out.writeUTF(msg1);
                    System.out.println("User login unsuccessful");
                }
                ss.close();
            }
        }
        catch(Exception x) { }
    }
}
```

TCP_Client.java

```
import java.net.*;
import java.io.*;
public class TCP_Client {
    public static void main(String[] ar) {
        try {
            Socket socket = new Socket("localhost",5046);
            InputStream sin = socket.getInputStream();
            OutputStream sout = socket.getOutputStream();
            DataInputStream in = new DataInputStream(sin);
            DataOutputStream out = new DataOutputStream(sout);
            BufferedReader keyboard = new BufferedReader(new InputStreamReader
(System.in));
            String cardno = null;
            String pin=null;
            String msg=null;
```

```
            String msg1=null;
            System.out.println();
            while(true) {
                System.out.println("Enter card number: ");
                cardno = keyboard.readLine();
                System.out.println("Enter pin: ");
                pin=keyboard.readLine();
                out.writeUTF(cardno);
                out.writeUTF(pin);
                String a = in.readUTF();
                System.out.println(a);
                if(a.equals("Login successful")) {
                    String b=in.readUTF();
                    System.out.println();
                    System.out.println(b);
                    String amt=keyboard.readLine();
                    out.writeUTF(amt);
                    String c=in.readUTF();
                    System.out.println(c);
                    if(c.equals("Withdraw complete")) {
                        String d=in.readUTF();
                        System.out.println("Remaining Balance: Rs."+d);
                    }
                }
                out.flush();
                break;
            }
        }
        catch(Exception x) { System.out.println("exception caught"+x);}
    }
}
```

Successful login and withdrawal:

```
C:\Users\18BIT0272\Desktop\DCCN>javac TCP_Server.java

C:\Users\18BIT0272\Desktop\DCCN>java TCP_Server
Waiting for a user...
User login successful
Starting withdraw process
Checking account balance
Withdraw complete
```

```
C:\Users\18BIT0272\Desktop\DCCN>javac TCP_Client.java

C:\Users\18BIT0272\Desktop\DCCN>java TCP_Client

Enter card number:
142312
Enter pin:
2201
Login successful

Enter withdraw amount
500
Withdraw complete
Remaining Balance: Rs.700
```

Unsuccessful login due to incorrect credentials:

```
C:\Users\18BIT0272\Desktop\DCCN>java TCP_Server
Waiting for a user...
User login unsuccessful
C:\Users\18BIT0272\Desktop\DCCN>java TCP_Client

Enter card number:
123123
Enter pin:
1234
Login unsuccessful
```

Successful login but unsuccessful withdrawal due to insufficient balance:

```
C:\Users\18BIT0272\Desktop\DCCN>java TCP_Server
Waiting for a user...
User login successful
Starting withdraw process
Checking account balance
Insufficient Balance
C:\Users\18BIT0272\Desktop\DCCN>java TCP_Client

Enter card number:
142312
Enter pin:
2201
Login successful

Enter withdraw amount
1000
Insufficient Balance
```

2. The finance office of VIT wishes to make the transactions more secured. If you are a programmer how you will implement a system to validate the login credentials obtained from the user thereby denying them access to unauthorized users.
   - Understating of HTTPS for transactions.
   - Encryption and decryption of each transaction of message data with security algorithms.
   - Code for generation of a key for encryption and decryption using TCP/IP client server sockets for validation and verification of authorized users.

➢ Https is a protocol used to transfer files from a Web server onto a browser in order to view a Web page that is on the Internet. FTP is a network protocol used to transfer files from one computer to another over a TCP network. In HTTPS there is no visibility of ones IP address. It is hidden under the website name for example: if we take vit website we do not know the IP address as it is hidden under the name and we can know all the complete details of the website in whois.net or to know the IP we can also use ping command. So, if we type “ping” we get the IP address as this. HTTPS is used to hide this IP. If the IP is known there can be a lot of things that can be done and there will be no security to these organizations as we can get the complete URL information in whois.net or by using ping command.

➢ Encryption Algorithms:

- Symmetric encryption algorithms: Symmetric algorithms use the same key for encryption and decryption. These algorithms, can either operate in block mode (which works on fixed-size blocks of data) or stream mode (which works on bits or bytes of data). They are commonly used for applications like data encryption, file encryption and encrypting transmitted data in communication networks (like TLS, emails, instant messages, etc.).
- Asymmetric (or public key) encryption algorithms: Unlike symmetric algorithms, which use the same key for both encryption and decryption operations, asymmetric algorithms use two separate keys for these two operations. These algorithms are used for computing digital signatures and key establishment protocols.

Server.java

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
public class Server {
    private static User currentUser = new User();
    private static final String secretKey = "secretKey!!!";
    public static void main(String[] args) throws Exception {
        ServerSocket ss = new ServerSocket(5046);
        Socket s = ss.accept();
        DataOutputStream dos = new DataOutputStream(s.getOutputStream());
        DataInputStream dis = new DataInputStream(s.getInputStream());
        System.out.println("Server Is Connected");
        while (true) {
            String str = AES.decrypt(dis.readUTF(), secretKey);
            System.out.println(str);
            while (str.equalsIgnoreCase("connected")) {
                String cardNo = AES.decrypt(dis.readUTF().split("=")[1], secre
tKey);
                String pin = AES.decrypt(dis.readUTF().split("=")[1], secretKe
y);
```

```
                System.out.println("Card Number(Decrypted) - Server Side: " +
cardNo);
                System.out.println("Pin Number(Decrypted) - Server Side: " + p
in);
                currentUser.setCardNo(cardNo);
                currentUser.setPin(pin);
                User u = (User) Repository.checkCredentials(currentUser);
                if (u != null) {
                    currentUser.setBalance(u.getBalance());
                    currentUser.setName(u.getName());
                    dos.writeUTF(AES.encrypt("success", secretKey));
                    dos.writeUTF(AES.encrypt(currentUser.getName(), secretKey)
);
                    double amt = Double.parseDouble(AES.decrypt(dis.readUTF().
split("=")[1], secretKey));
                    System.out.println("Decrypted Amount - Server Side: " + am
t);
                    String[] res = Repository.withdraw(currentUser, amt);
                    dos.writeUTF(AES.encrypt(res[0], secretKey));
                    dos.writeUTF(AES.encrypt(res[1], secretKey));
                    break;
                }
                else {
                    dos.writeUTF(AES.encrypt("failed", secretKey));
                    String msg = AES.decrypt(dis.readUTF(), secretKey);
                    System.out.println("msg " + msg);
                    if (msg.equalsIgnoreCase("quit"))
                        break;
                    else
                        continue;
                }
            }
        }
    }
}
```

Client.java

```
import java.io.Console;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.util.Scanner;
public class Client {
    private static final String secretKey = "secretKey!!!";
    public static void main(String[] args) throws Exception {
        Socket s = new Socket("localhost", 5046);
        Console c = System.console();
        DataOutputStream dos = new DataOutputStream(s.getOutputStream());
        DataInputStream dis = new DataInputStream(s.getInputStream());
```

```
        System.out.println(
                "\n\nWelcome to TCP/IP - Encryption/Decryption ATM system \n\n
");
        while (true) {
            dos.writeUTF(AES.encrypt("connected", secretKey));
            System.out.println("Press '1' to withdraw");
            String ip = c.readLine();
            while (!ip.equalsIgnoreCase("quit")) {
                if (ip.equalsIgnoreCase("1")) {
                    System.out.println("\nEnter Credentials-\n");
                    System.out.print("Enter Card Number: ");
                    String cardNo = AES.encrypt(c.readLine(), secretKey);
                    System.out.print("Enter Pin Number: ");
                    String pin = AES.encrypt(String.valueOf(c.readPassword()),
 secretKey);
                    System.out.println("Encrypted Card Number At Client Side:
" + cardNo);
                    System.out.println("Encrypted Pin At Client Side: " + pin)
;
                    dos.writeUTF("CardNumber=" + cardNo);
                    dos.writeUTF("Pin Number=" + pin);
                    String msg = AES.decrypt(dis.readUTF(), secretKey);
                    if (msg.equalsIgnoreCase("success")) {
                        System.out.println("\nWelcome, " + AES.decrypt(dis.rea
dUTF(), secretKey));
                        System.out.println("\nEnter The Amount That You Want T
o Withdraw");
                        String amt = AES.encrypt(c.readLine(), secretKey);
                        System.out.println("Encrypted Amount At Client Side: "
 + amt);
                        dos.writeUTF("Amount=" + amt);
                        msg = AES.decrypt(dis.readUTF(), secretKey);
                        String res = AES.decrypt(dis.readUTF(), secretKey);
                        String send = AES.encrypt("quit", secretKey);
                        if (msg.equalsIgnoreCase("success")) {
                            System.out.println(res);
                            dos.writeUTF(send);
                            //System.out.println("\nThanks for using our servi
ce");
                            break;
                        }
                        else {
                            System.out.println(res);
                            dos.writeUTF(send);
                            //System.out.println("\nThanks for using our servi
ce");
                            break;
                        }
```

```
                    }
                    else {
                        System.out.println("\nnInvalid Credentials!, Recheck c
redentials");
                        System.out.println("\nContinue?[y/n]");
                        String bool = c.readLine();
                        if (bool.equalsIgnoreCase("y")) {
                            dos.writeUTF(AES.encrypt("again", secretKey));
                            continue;
                        }
                        else {
                            dos.writeUTF(AES.encrypt("quit", secretKey));
                            break;
                        }
                    }
                }
                else
                    break;
            }
        }
    }
}
```

AES.java

```
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
public class AES {
    private static SecretKeySpec secretKey;
    private static byte[] key;
    public static void setKey(String myKey) {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }
```

```
    public static String encrypt(String strToEncrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncr
ypt.getBytes("UTF-8")));
        } catch (Exception e) {
            System.out.println("Error occurred during encryption: " + e.toStri
ng());
        }
        return null;
    }
    public static String decrypt(String strToDecrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return new String(cipher.doFinal(Base64.getDecoder().decode(strToD
ecrypt)));
        } catch (Exception e) {
            System.out.println("Error occurred during decryption: " + e.toStri
ng());
        }
        return null;
    }
}
```

Repository.java

```
public class Repository {
    private static User[] users = { new User("Priyal", "1234", "4321", 20033.6
3), new User("Mitushi", "1235", "5321", 143034.32), new User("Kanishka", "1236
", "6321", 890089.85), new User("Sakina", "1237", "7321", 27443.92), };
    public static User checkCredentials(User u) {
        try {
            for (User user : users) {
                if (user.getCardNo().equals(u.getCardNo()) && user.getPin().eq
uals(u.getPin()))
                    return user;
            }
            return null;
        } catch (Exception e) {
            throw e;
        }
    }
    public static String[] withdraw(User u, double amount) {
        try {
            for (User user : users) {
                if (user.getCardNo().equals(u.getCardNo())) {
```

```
                    double currBal = user.getBalance();
                    if (currBal >= amount) {
                        user.setBalance(currBal - amount);
                        return new String[] { "success", "Withdrawal Amount: "
 + amount + "\n New Balance is " + user.getBalance() };
                    }
                    else
                        return new String[] { "failed", "Withdrawal Not Possib
le. Insufficient Balance" };
                }
            }
            return new String[] { "failed", "User Not Found!" };
        } catch (Exception e) {
            throw e;
        }
    }
}
```

User.java

```
public class User {
    private String name;
    private String cardNo;
    private String pin;
    private double balance;
    public User() {
    }
    public User(String name, String cardNo, String pin, double balance) {
        this.name = name;
        this.cardNo = cardNo;
        this.pin = pin;
        this.balance = balance;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setCardNo(String cardNo) {
        this.cardNo = cardNo;
    }
    public void setPin(String pin) {
        this.pin = pin;
    }
    public void setBalance(double balance) {
        this.balance = balance;
    }
    public String getName() {
        return name;
    }
    public String getCardNo() {
        return cardNo;
```

```
    }
    public String getPin() {
        return pin;
    }
    public double getBalance() {
        return balance;
    }
    public String toString() {
        return "Name: " + name + "\tPin: " + pin + "\tCard Number: " + cardNo
+ "\tBalance: " + balance;
    }
}
```

Valid Credentials and Successful withdrawal:

```
C:\Users\18BIT0272\Desktop\DCCN>javac Server.java

C:\Users\18BIT0272\Desktop\DCCN>java Server
Server Is Connected
connected
Card Number(Decrypted) - Server Side: 1234
Pin (Decrypted) - Server Side: 4321
Decrypted Amount - Server Side: 2000.0
quit
connected
```

```
C:\Users\18BIT0272\Desktop\DCCN>javac Client.java

C:\Users\18BIT0272\Desktop\DCCN>java Client


Welcome to TCP/IP - Encryption/Decryption ATM system


Press '1' to withdraw
1

Enter Credentials-

Enter Card Number: 1234
Enter Pin Number:
Encrypted Card Number At Client Side: NpBnu6aJ7LD9ipMaJGTSeA==
Encrypted Pin At Client Side: 7IoD9hmVARD+t4P3uVNjdg==

Welcome, Priyal

Enter The Amount That You Want To Withdraw
2000
Encrypted Amount At Client Side: ApMlx8Mx34D0UdO2jV9HBg==
Withdrawal Amount: 2000.0
 New Balance: 20033.87
```

Valid Credentials but Unsuccessful withdrawal:

```
C:\Users\18BIT0272\Desktop\DCCN>javac Server.java

C:\Users\18BIT0272\Desktop\DCCN>java Server
Server Is Connected
connected
Card Number(Decrypted) - Server Side: 1234
Pin (Decrypted) - Server Side: 4321
Decrypted Amount - Server Side: 23000.0
quit
connected
```

```
C:\Users\18BIT0272\Desktop\DCCN>javac Client.java

C:\Users\18BIT0272\Desktop\DCCN>java Client


Welcome to TCP/IP - Encryption/Decryption ATM system


Press '1' to withdraw
1

Enter Credentials-

Enter Card Number: 1234
Enter Pin Number:
Encrypted Card Number At Client Side: NpBnu6aJ7LD9ipMaJGTSeA==
Encrypted Pin At Client Side: 7IoD9hmVARD+t4P3uVNjdg==

Welcome, Priyal

Enter The Amount That You Want To Withdraw
23000
Encrypted Amount At Client Side: UIi/geFy8dlE8TcePdb4fA==
Withdrawal Not Possible. Insufficient Balance
```

Invalid Credentials:

```
C:\Users\18BIT0272\Desktop\DCCN>javac Server.java

C:\Users\18BIT0272\Desktop\DCCN>java Server
Server Is Connected
connected
Card Number(Decrypted) - Server Side: 1234
Pin (Decrypted) - Server Side: 5432
msg quit
```

```
C:\Users\18BIT0272\Desktop\DCCN>javac Client.java

C:\Users\18BIT0272\Desktop\DCCN>java Client


Welcome to TCP/IP - Encryption/Decryption ATM system


Press '1' to withdraw
1

Enter Credentials-

Enter Card Number: 1234
Enter Pin Number:
Encrypted Card Number At Client Side: NpBnu6aJ7LD9ipMaJGTSeA==
Encrypted Pin At Client Side: x+8JlAR01FaQ4ZEK1rp0Cw==

nInvalid Credentials! Please recheck
```

****