**Applied Cryptography (CMPS 381)Course Project**

Selected Topic: Password Store

Submission Date: 12/7/2021

Course Project Implementation

Participated student’s information:

|  |  |  |
| --- | --- | --- |
| Names: | Student QU number | QU-email |
| Talha Abdullah | 201903446 | [Tp1903446@qu.edu.qa](mailto:Tp1903446@qu.edu.qa) |
| Ibrahim Demdoum | 201701515 | [Id1701515@qu.edu.qa](mailto:Id1701515@qu.edu.qa) |
| Mohammed Omar | 201911057 | [Mo1911057@qu.edu.qa](mailto:Mo1911057@qu.edu.qa) |

**Goal:**

**This is the implementation of individual components and then aggregate all the considered modules and implementations into a real-time fully functioning password storage system.**

1. **User Interface**

**Main View**

A picture containing website

Description automatically generated

**Main View Controller**

package mainview;

import java.io.IOException;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.AnchorPane;

import javafx.stage.Stage;

public class MainViewController {

@FXML

private Button button1;

@FXML

private Button button2;

@FXML

void handleButtonAction1(ActionEvent event) {

FXMLLoader loading = new FXMLLoader(getClass().getResource("Login\\loginView.fxml"));

try {

AnchorPane pane = (AnchorPane) loading.load();

Scene scene = new Scene(pane);

Stage stage = new Stage();

stage.setScene(scene);

stage.show();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

@FXML

void handleButtonAction2(ActionEvent event) {

FXMLLoader loading = new FXMLLoader(getClass().getResource("Login\\signupView.fxml"));

try {

AnchorPane pane = (AnchorPane) loading.load();

Scene scene = new Scene(pane);

Stage stage = new Stage();

stage.setScene(scene);

stage.show();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

**Main View FXML**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane maxHeight=*"1500.0"* maxWidth=*"1500.0"* minHeight=*"-Infinity"* minWidth=*"-Infinity"* prefHeight=*"319.0"* prefWidth=*"814.0"* style=*"-fx-background-color: lightblue;"* xmlns=*"http://javafx.com/javafx/17"* xmlns:fx=*"http://javafx.com/fxml/1"* fx:controller=*"mainview.MainViewController"*>

<children>

<Label alignment=*"CENTER"* prefHeight=*"45.0"* prefWidth=*"814.0"* style=*"-fx-background-color: darkblue;"* text=*"Cloud Storage System"* textFill=*"RED"*>

<font>

<Font name=*"Arial Bold"* size=*"24.0"* />

</font>

</Label>

<Button fx:id=*"button2"* layoutX=*"211.0"* layoutY=*"196.0"* mnemonicParsing=*"false"* onAction=*"#handleButtonAction2"* prefHeight=*"86.0"* prefWidth=*"393.0"* text=*"Signup"*>

<font>

<Font size=*"18.0"* />

</font>

</Button>

<Button fx:id=*"button1"* layoutX=*"211.0"* layoutY=*"74.0"* mnemonicParsing=*"false"* onAction=*"#handleButtonAction1"* prefHeight=*"98.0"* prefWidth=*"393.0"* text=*"Login"*>

<font>

<Font size=*"18.0"* />

</font>

</Button>

</children>

</AnchorPane>

**Signup Interface**

Graphical user interface, application

Description automatically generated

**Controller Class**

import javafx.fxml.FXML;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

public class PleaseProvideControllerClassName {

@FXML

private PasswordField Password;

@FXML

private TextField Username;

@FXML

private TextField confirmPassword;

@FXML

private TextField confirmid;

@FXML

private TextField emailid;

@FXML

private TextField phoneNumber;

}

**FXML Code**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.PasswordField?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane xmlns=*"http://javafx.com/javafx/17"* xmlns:fx=*"http://javafx.com/fxml/1"*>

<children>

<AnchorPane prefHeight=*"456.0"* prefWidth=*"542.0"* style=*"-fx-background-color: lightblue;"*>

<children>

<Label alignment=*"CENTER"* prefHeight=*"54.0"* prefWidth=*"542.0"* style=*"-fx-background-color: yellow;"* text=*"Cloud Storage System"* textFill=*"#3605fa"*>

<font>

<Font size=*"24.0"* />

</font>

</Label>

<Label layoutX=*"106.0"* layoutY=*"90.0"* prefHeight=*"27.0"* prefWidth=*"99.0"* text=*"Username"*>

<font>

<Font size=*"18.0"* />

</font>

</Label>

<Label layoutX=*"106.0"* layoutY=*"140.0"* text=*"Password"*>

<font>

<Font size=*"18.0"* />

</font>

</Label>

<TextField fx:id=*"Username"* layoutX=*"285.0"* layoutY=*"91.0"* />

<PasswordField fx:id=*"Password"* layoutX=*"285.0"* layoutY=*"141.0"* />

<Button layoutX=*"234.0"* layoutY=*"399.0"* mnemonicParsing=*"false"* prefHeight=*"27.0"* prefWidth=*"75.0"* text=*"Sign Up"* />

<Label layoutX=*"103.0"* layoutY=*"188.0"* text=*"Confirm Password"*>

<font>

<Font size=*"18.0"* />

</font>

</Label>

<Label layoutX=*"101.0"* layoutY=*"243.0"* text=*"Email address"*>

<font>

<Font size=*"18.0"* />

</font>

</Label>

<Label layoutX=*"100.0"* layoutY=*"299.0"* text=*"Confirm Email"*>

<font>

<Font size=*"18.0"* />

</font>

</Label>

<Label layoutX=*"103.0"* layoutY=*"349.0"* text=*"Phone Number"*>

<font>

<Font size=*"18.0"* />

</font>

</Label>

<TextField fx:id=*"confirmPassword"* layoutX=*"285.0"* layoutY=*"189.0"* />

<TextField fx:id=*"emailid"* layoutX=*"285.0"* layoutY=*"244.0"* />

<TextField fx:id=*"confirmid"* layoutX=*"285.0"* layoutY=*"300.0"* />

<TextField fx:id=*"phoneNumber"* layoutX=*"285.0"* layoutY=*"350.0"* />

</children>

</AnchorPane>

</children>

</AnchorPane>

**Login Interface**

Graphical user interface

Description automatically generated

**Controller Class**

import javafx.fxml.FXML;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

public class PleaseProvideControllerClassName {

@FXML

private PasswordField Password;

@FXML

private TextField Username;

}

**FXML Code**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.PasswordField?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane xmlns=*"http://javafx.com/javafx/17"* xmlns:fx=*"http://javafx.com/fxml/1"*>

<children>

<AnchorPane prefHeight=*"238.0"* prefWidth=*"535.0"* style=*"-fx-background-color: lightblue;"*>

<children>

<Label alignment=*"CENTER"* prefHeight=*"54.0"* prefWidth=*"535.0"* style=*"-fx-background-color: yellow;"* text=*"Cloud Storage System"* textFill=*"#3605fa"*>

<font>

<Font size=*"24.0"* />

</font>

</Label>

<Label layoutX=*"106.0"* layoutY=*"101.0"* prefHeight=*"27.0"* prefWidth=*"99.0"* text=*"Username"*>

<font>

<Font size=*"18.0"* />

</font>

</Label>

<Label layoutX=*"106.0"* layoutY=*"150.0"* text=*"Password"*>

<font>

<Font size=*"18.0"* />

</font>

</Label>

<TextField fx:id=*"Username"* layoutX=*"285.0"* layoutY=*"102.0"* />

<PasswordField fx:id=*"Password"* layoutX=*"285.0"* layoutY=*"151.0"* />

<Button layoutX=*"242.0"* layoutY=*"199.0"* mnemonicParsing=*"false"* prefHeight=*"25.0"* prefWidth=*"51.0"* text=*"Login"* />

</children>

</AnchorPane>

</children>

</AnchorPane>

1. **Data Travel from Interface to the System**

After a user enters the details either through sign up or login, the message is sent in such a format for a login interface, [USERNAME,PASSWORD], separated by a “,”, concatenated and encrypted using AES. (AES Code is attached along with the project). Thus, the message travels in such a format. Hence encryption and decryption take place in such a manner.

Example a)

Text

Description automatically generated

The plaintext will be converted to the “Encrypted message” while on its way to the system. Hence, attacker cannot break and hack the key itself. The data is encrypted during the customer-server transfer.

More Examples are such:

b)Text

Description automatically generated

c)

Text

Description automatically generated

1. **Multiple users can access the database at any time [Basic Service provider and Client/Customer Interaction]**

[Client-Server Interaction Procedure]

**CustomerSide.java**

package CustomerServerService;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.Socket;

import java.util.concurrent.TimeUnit;

public class CustomerSide {

public CustomerSide() {

try {

int n=0;

//Customer connection to the server with port 1500

Socket clientconn = new Socket("localhost", 1500);

BufferedReader from\_server = new BufferedReader(new InputStreamReader(clientconn.getInputStream()));

BufferedReader from\_client = new BufferedReader(new InputStreamReader(System.in));

PrintWriter to\_server = new PrintWriter(clientconn.getOutputStream(),true);

//Print the connection status with the server

System.out.println("Connected with server " + clientconn.getInetAddress() + ":" + clientconn.getPort());

while(true) {

String reader;

reader=from\_server.readLine();

System.out.println(reader);

reader=from\_server.readLine();

System.out.print(reader);

String username=from\_client.readLine();

reader=from\_server.readLine();

System.out.print(reader);

String password=from\_client.readLine();

String data=username+","+password;

to\_server.println(data);

System.out.println("Verifying....");

reader=from\_server.readLine();

if(reader.equals("User Verified!")) {

System.out.println(reader);

for(int i=0;i<3;i++) {

reader=from\_server.readLine();

System.out.println(reader);

}

System.out.println("-------------------TERMINATING----------------");

System.exit(0);

}

else {

System.out.println("User is Not Registered!");

n++;

if(n>2) {

System.out.println("Your Have Failed to Enter Correct Password 3 Times");

try {

System.out.println("Wait 3 minutes before you try Again!");

TimeUnit.SECONDS.sleep(180);

} catch (InterruptedException ie) {

Thread.currentThread().interrupt();

}

}

}

}

} catch (IOException ioe) {

System.out.println("Error" + ioe);

}

}

public static void main(String [] args) {

new CustomerSide();

}

}

**ServerSide.java**

package CustomerServerService;

import java.net.\*;

import java.io.\*;

public class ServerSide {

public static void main(String[] args) {

ServerSocket server = null;

Socket nextClient = null;

try {

server = new ServerSocket(1500);

} catch (IOException e1) {

e1.printStackTrace();

}

System.out.println("Server waiting for client on port " + server.getLocalPort());

System.out.println("Password Storage System Server Started");

try {

for (;;) {

try {

nextClient = server.accept();

} catch (IOException e) {

e.printStackTrace();

}

//pass the initialized/started ports to the main Service Class

MainService s = new MainService(nextClient);

s.start();

}

} finally {

try {

if (server != null)

server.close();

if (nextClient != null)

nextClient.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

**MainService.java**

package CustomerServerService;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

public class MainService extends Thread {

Socket nextClient;

PrintWriter to\_client;

BufferedReader from\_client;

ServerSocket server;

public MainService(Socket nextClient) {

super();

this.nextClient = nextClient;

}

public void run() {

try {

//Assumed Database for Customer Login [ assumed that these users have signed up to the service ]

ArrayList <customerDataBase> customerLogInfo=new ArrayList<>();

ArrayList <ActualPasswords> passwordLogInfo1=new ArrayList<>();

ArrayList <ActualPasswords> passwordLogInfo2=new ArrayList<>();

ArrayList <ActualPasswords> passwordLogInfo3=new ArrayList<>();

ActualPasswords p1=new ActualPasswords("www.google.com", "LoveHealth");

ActualPasswords p2=new ActualPasswords("www.github.com", "Lovegit123");

passwordLogInfo1.add(p1);

passwordLogInfo1.add(p2);

customerDataBase c1=new customerDataBase("Talha", "Talha123", passwordLogInfo1);

ActualPasswords p3=new ActualPasswords("www.google.com", "HateHealth");

ActualPasswords p4=new ActualPasswords("www.github.com", "Github234");

passwordLogInfo2.add(p3);

passwordLogInfo2.add(p4);

customerDataBase c2=new customerDataBase("Omar","omar123", passwordLogInfo2);

ActualPasswords p5=new ActualPasswords("www.google.com", "34242342");

ActualPasswords p6=new ActualPasswords("www.github.com", "54644567");

passwordLogInfo3.add(p5);

passwordLogInfo3.add(p6);

customerDataBase c3=new customerDataBase("Ibrahim","ibrahim123", passwordLogInfo3);

customerLogInfo.add(c1);

customerLogInfo.add(c2);

customerLogInfo.add(c3);

to\_client = new PrintWriter(nextClient.getOutputStream(), true);

from\_client = new BufferedReader(new InputStreamReader(nextClient.getInputStream()));

// Display connection details

System.out.println("Receiving Request From " + nextClient.getInetAddress() + ":" + nextClient.getPort());

while(true) {

to\_client.println("<------------Login To the System--------->");

to\_client.println("Enter your username :");

to\_client.println("Enter your password :");

String clientData = from\_client.readLine();

System.out.println("Recieved Data is : "+clientData+ "[ From Client IP : "+nextClient.getInetAddress()+nextClient.getPort()+" ]");

String [] tokenizedData=clientData.split("[,]", 0);

//System.out.println("Username : "+ tokenizedData[0]);

//System.out.println("Password : "+ tokenizedData[1]);

int z=0;

int k=0;

for(int i=0;i<3;i++) {

String user=customerLogInfo.get(i).getUsername();

String password=customerLogInfo.get(i).getPassword();

if(tokenizedData[0].trim().equals(user.trim()) && tokenizedData[1].trim().equals(password.trim())) {

z=1;

k=1;

to\_client.println("User Verified!");

//System.out.println("Verified");

ArrayList<ActualPasswords> data=new ArrayList<>();

data=customerLogInfo.get(i).getPasswordList();

to\_client.println("<------Displaying All Your Passwords----->");

for(int j=0;j<data.size();j++) {

to\_client.println(data.get(j).getEntity()+" : "+data.get(j).getPassword());

}

break;

}

}

if(k==1) {

break;

}

if(z==0) {

to\_client.println("User Verification Failed !");

}

}

} catch (IOException e) {

e.printStackTrace();

}

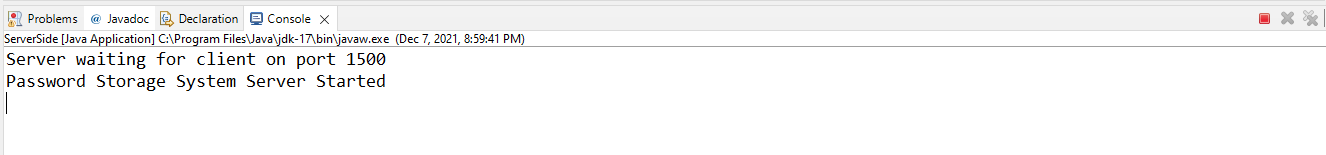
}

**}**

[Attached File inside the .rar for the Java Code]

**Screen Shots**

1. Server (Password Store) is online and ready to receive and serve any customers



1. Client Can Access his stored Passwords if the system identifies his username and password to be legit

Text

Description automatically generated

Server After Serving this client: (Confirms the Connection and Data Received from Client)

Graphical user interface, text, application

Description automatically generated

1. If a client enters a wrong username or password (A failure Message Pops up)

Graphical user interface, application

Description automatically generated with medium confidence

1. Multiple Clients can access the Service at the same time

Server can process Multiple Requests at the same Time

Graphical user interface, text, application

Description automatically generated

Multiple Customers accessing the Server at a Time (At Runtime)

Graphical user interface, text, application

Description automatically generated

1. In the Situation where a hacker is trying to break into any user’s password/data

(There is a timeout after every 3 tries. If 3 fails will lead to a 3-minute delay)

Graphical user interface, text, application, email

Description automatically generated

1. **Hash Function to Check the Authenticity of the Login**

To Authenticate the user login, there is a separate database of all the hashed values of all the signed up users in the format of [username,password]. Thus, when a user logs in, there is a hash check to verify that the user has entered correct credentials

**package** Hashing;

**import** java.security.MessageDigest;

**import** java.security.NoSuchAlgorithmException;

**import** java.util.ArrayList;

**public** **class** HashFunction {

//using MD5 Hashing

**public** **static** **void** main (String [] args) {

ArrayList <String> hashedDataValues=**new** ArrayList<>();

LoginData l1=**new** LoginData("Talha", "Talha123");

LoginData l2=**new** LoginData("Ibrahim", "ibrahim123");

LoginData l3=**new** LoginData("Omar", "omar123");

ArrayList <LoginData> data=**new** ArrayList<>();

data.add(l1);

data.add(l2);

data.add(l3);

String [] hashDatabase=**new** String[3];

**for**(**int** i=0;i<data.size();i++) {

hashDatabase[i]=data.get(i).getUsername()+","+data.get(i).getPassword();

}

String dataToHash= "Talha,Talha123";

String generatedHash="null";

**try** {

**for**(**int** i=0;i<hashDatabase.length;i++) {

MessageDigest hash=MessageDigest.*getInstance*("MD5");

hash.update(hashDatabase[i].getBytes());

**byte**[] b=hash.digest();

StringBuilder sb = **new** StringBuilder();

//this byte contains decimal format to be converted to hexadecimal format

**for** (**int** j = 0; j < b.length; j++) {

sb.append(Integer.*toString*((b[j] & 0xff) + 0x100, 16).substring(1));

}

hashedDataValues.add(sb.toString());

}

System.***out***.println("The Hashed Database Values Will be Hidden from the User (HIDDEN FROM USER): ");

**for**(**int** i=0;i<hashDatabase.length;i++) {

System.***out***.println(hashedDataValues.get(i));

}

System.***out***.println();

System.***out***.println("To check if the Entered/The one who logged in is a legitimate user");

MessageDigest checkHash=MessageDigest.*getInstance*("MD5");

checkHash.update(dataToHash.getBytes());

**byte**[]b=checkHash.digest();

StringBuilder s=**new** StringBuilder();

**for** (**int** i = 0; i < b.length; i++) {

s.append(Integer.*toString*((b[i] & 0xff) + 0x100, 16).substring(1));

}

generatedHash=s.toString();

**for**(**int** i=0; i<hashedDataValues.size();i++) {

**if**(generatedHash.equals(hashedDataValues.get(i))) {

System.***out***.println("The Hashes Match and the User is Verified!");

System.*exit*(0);

}

}

System.***out***.println("The Hash Does not Match");

}**catch**(NoSuchAlgorithmException E) {

E.printStackTrace();

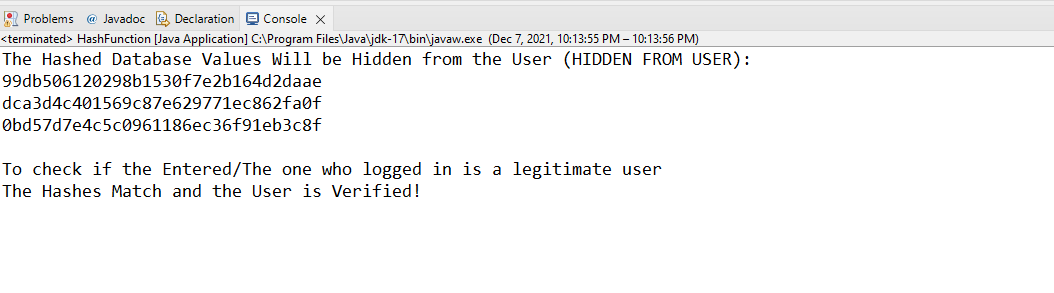
}

}

}

**Screen Shots:**

1. If the user enters correct credentials



1. If the user enters wrong credentials

Graphical user interface, text, application

Description automatically generated

1. **Encrypted Database**

If a hacker enters the system where all the users’ passwords are stored, all the data will be encrypted, and the hacker will find it difficult to decrypt the database. Full database encryption is used to secure the whole database.

**MainDataBase.java**

**package** DataBaseEncryption;

**public** **class** MainDataBase {

//Assumed Database for Customer Login [ assumed that these users have signed up to the service ]

//sample values for user1 only taken for example purpose

**private** String user1entity1="www.google.com";

**private** String user1password1="2324324";

**private** String user1entity2="www.github.com";

**private** String user1password2="21312541";

**public** String getUser1entity1() {

**return** user1entity1;

}

**public** String getUser1password1() {

**return** user1password1;

}

**public** String getUser1entity2() {

**return** user1entity2;

}

**public** String getUser1password2() {

**return** user1password2;

}

}

**EncryptingDatabase.java**

**package** DataBaseEncryption;

**import** java.io.UnsupportedEncodingException;

**import** java.security.InvalidKeyException;

**import** java.security.KeyPair;

**import** java.security.KeyPairGenerator;

**import** java.security.NoSuchAlgorithmException;

**import** java.security.Signature;

**import** javax.crypto.BadPaddingException;

**import** javax.crypto.Cipher;

**import** javax.crypto.IllegalBlockSizeException;

**import** javax.crypto.NoSuchPaddingException;

**public** **class** EncryptingDatabase {

**public** **static** **void** main(String [] args) **throws** NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException, UnsupportedEncodingException {

String FilePath=System.*getProperty*("user.dir");

System.***out***.println("Current Directory : "+FilePath);

MainDataBase MDB=**new** MainDataBase();

//Creating a Signature object

Signature sign = Signature.*getInstance*("SHA256withRSA");

//Creating KeyPair generator object

KeyPairGenerator keyPairGen = KeyPairGenerator.*getInstance*("RSA");

keyPairGen.initialize(2048);

KeyPair pair = keyPairGen.generateKeyPair();

//Creating a Cipher object

Cipher cipher = Cipher.*getInstance*("RSA/ECB/PKCS1Padding");

cipher.init(Cipher.***ENCRYPT\_MODE***, pair.getPublic());

**byte**[] input = MDB.getUser1entity1().getBytes();

cipher.update(input);

//encrypting the data

**byte**[] cipherText = cipher.doFinal();

System.***out***.println("Encrypted Data (User1Entity1):");

System.***out***.println(**new** String(cipherText, "UTF8"));

System.***out***.println();

Cipher cipher1 = Cipher.*getInstance*("RSA/ECB/PKCS1Padding");

cipher1.init(Cipher.***ENCRYPT\_MODE***, pair.getPublic());

**byte**[] input1 = MDB.getUser1password1().getBytes();

cipher1.update(input1);

//encrypting the data

**byte**[] cipherText1 = cipher1.doFinal();

System.***out***.println("Encrypted Data (User1Password1):");

System.***out***.println(**new** String(cipherText1, "UTF8"));

System.***out***.println();

Cipher cipher2 = Cipher.*getInstance*("RSA/ECB/PKCS1Padding");

cipher2.init(Cipher.***ENCRYPT\_MODE***, pair.getPublic());

**byte**[] input2 = MDB.getUser1entity2().getBytes();

cipher2.update(input2);

//encrypting the data

**byte**[] cipherText2 = cipher2.doFinal();

System.***out***.println("Encrypted Data (User1Entity2):");

System.***out***.println(**new** String(cipherText2, "UTF8"));

System.***out***.println();

Cipher cipher3 = Cipher.*getInstance*("RSA/ECB/PKCS1Padding");

cipher3.init(Cipher.***ENCRYPT\_MODE***, pair.getPublic());

**byte**[] input3 = MDB.getUser1password2().getBytes();

cipher.update(input3);

//encrypting the data

**byte**[] cipherText3 = cipher3.doFinal();

System.***out***.println("Encrypted Data (User1Password2):");

System.***out***.println(**new** String(cipherText3, "UTF8"));

}

}

**Screen Shot:**

Each Entity and Password are encrypted using SHA256RSA

Graphical user interface, text, application, email

Description automatically generated