Secure Chat - End to End Encryption

Chaitanya Vooradi

Group Name: Sunrisers

Group Member: Himani Kondra

Abstract:

This is the final report prepared for project in Network Security Course(CECE-579). The aim of this project is to design a desktop end to end secure chat application.Aim is to provide authentication and integrity to the message. Application was developed using java language. AES-256 was used as an encryption algorithm and SHA as hash algorithm for integrity.

1 Introduction:

Chatting is the process of bringing people together for the exchange of information. During the process of sharing information there are many attackers who try to steal the information. So messages exchanged between the people need to secured.

Open Whisper Systems signal chat application is one of most secured one as of now. It provides both message and voice call services.Services like sharing attachments, phone contacts, group chat etc are available. All these communications are end to end encrypted. No need to have pins and separate logins.Only the sender and intended receiver can see the plain text. Message leaving ou of client is encrypted.

Whatsapp collaborated with signal recently and many other applications like facebook messenger and Google Allo. All these applications are using open whisper protocol to achieve the secure chat.

Chat application developed in this project uses PGP algorithm which makes the message decryption harder by an unknown intruders.A complete end to end message encryption was achieved in this chat application.

2 Literature Search:

2.1 Viber:

Viber supports number of platforms like Windows, OS X, iOS, Android,Windows Phone, BlackBerry, Bada and Nokia. Viber is end to end secure chat application with services like sharing messages, attachments and videos. Authentication will be done using the option "Trust this Contact", where each of them receive key which can be verified by the individual users.
Viber is End to end Secure in windows 10 and for Android and IOS from version 6.0

2.2 Facebook Messener:

Facebook Messenger uses the signal encryption system developed by Open Whisper Systems. Messengers new layer of encryption need to be enabled manually instead of default.

2.3 Cryptocat:

Need to install the add-on for Chrome, Firefox, Safari, Opera etc.Click on the add-on and give the required information like nick-name. Need to type the user name to connect and begin the chat.

2.4 Project Approach:

In this project we are using AES-256 in CBC mode and PKCS5 padding to encrypt message. SHA-2 for integrity. RSA algorithm to encrypt the keys. RSA public keys are encrypted using secrete key shared between users and stored in Data base. Bcrypt algorithm to store password.

3 Project Report:

3.1 Problem Statement:

To implement secure end to end chat application which is CCA secure and to achieve secure secrete key exchange.
The main Goals to achieve are
1)Authentication
2)Integrity of the Message
3)Confidentiality
4)Non Repudiation
5)Password Should not be stored in Plain text.
6)Email Authentication during Registration

Before Login every novice user need to register using their unique user-name and Password. Need to enter the email Id during the Registration process. Verification message will be sent to the given mail Id which need to entered to authenticate the mail Id.

3.2 Challenges Faced:

1)Faced problems while achieving A+ certificate in SSL labs.
2)Faced problems while connecting to database using JDBC library.
3) Received some problems while authenticating the email Id.
4) Crypto Libraries for AES-256 are not working initially.

3.3 Solutions:

1) Rewrite module is not enabled initially after installation of apache and sql database. Enabled this module using command and inserted the Rewrite command lines in configuration files and restarted apache to work normally.

Command: sudo a2enmod rewrite

2) User privileges were missing initially which took time to debug the issue.
3) In order to make work crypto libraries, security files need to changed in java security folder.

3.4 Individual Work:

1)Created instances in AWS server and achieved A+ certificate in SSL labs.

2)Installed mysql database and apache server.

3)Used Jdbc libraries to connect to the database and achieved communication happen between two clients.

4)Developed an algorithm to store and retrieve messages from database.

5)Developed chat application without encryption.Integrate the code for encryption developed by Himani.

6)Used Bcrypt libraries to store the password.

7)Email Verification during registration.

8) Developed all the forms and windows used(Registration, login, otp verification and chat window.

9) Used AES Encryption Algorithm to securely store the private key in DB.

3.5 Comparison of work with teammate:

Both of them divided the equally in-order to complete the project more quickly and effectively. Initially she worked on setting up the elastic ip and integration of it. During this time I focused on getting A+ certificate. Later she started working on the security model. I worked on developing the chat process and algorithm to store and retrieve data. After that I worked on encryption of private key to securely store the key.

My Weaknesses:

1) Lazy in preparation of documents.

2) Can’t Work under Pressure.

3) No knowledge in creation of elastic ip.

My Parter weaknesses:

1) No debugging Skills

2) Lack of skills in integration of Code

3) No knowledge in getting A+ certificate in SSL labs.

3.6 Shortcomings:

1)Group chat feature is missing in the chat application which can developed if given more time to work on it.

2)GUI can be developed in a more attractive way to attract the users.

3)Because this is a Desktop application some of the user friendly features are missing in it.

3.7 Lessons Learned:

1) Learned SQl language in-order to work with the db.

2)Learned about AWS instance and how to establish a secure tunnel.

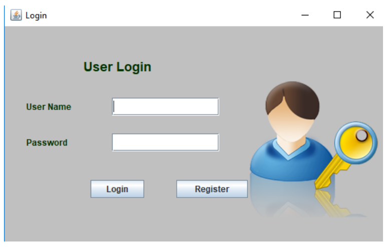
3)Learned how to work with JDBC connector.

4)Came to know about many encryption algorithms and ways to achieve a secure communication.

5) Learned more about the loop holes to attack a communication network.

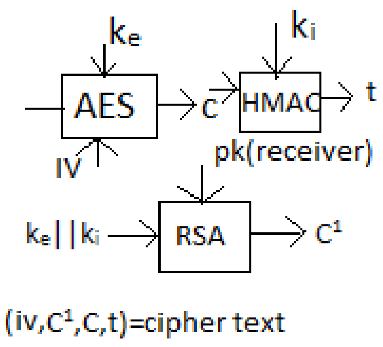
4 Implementation and Results:

4.1 User Login:

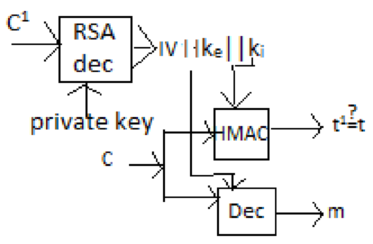
Password stored using Hash algorithm(Bcrypt). 

4.2 Encryption and Decryption:

Public keys are encrypted using secrete word shared between the users and stored in DB. AES-256 in CBC mode and SHA-2 algorithms were used.

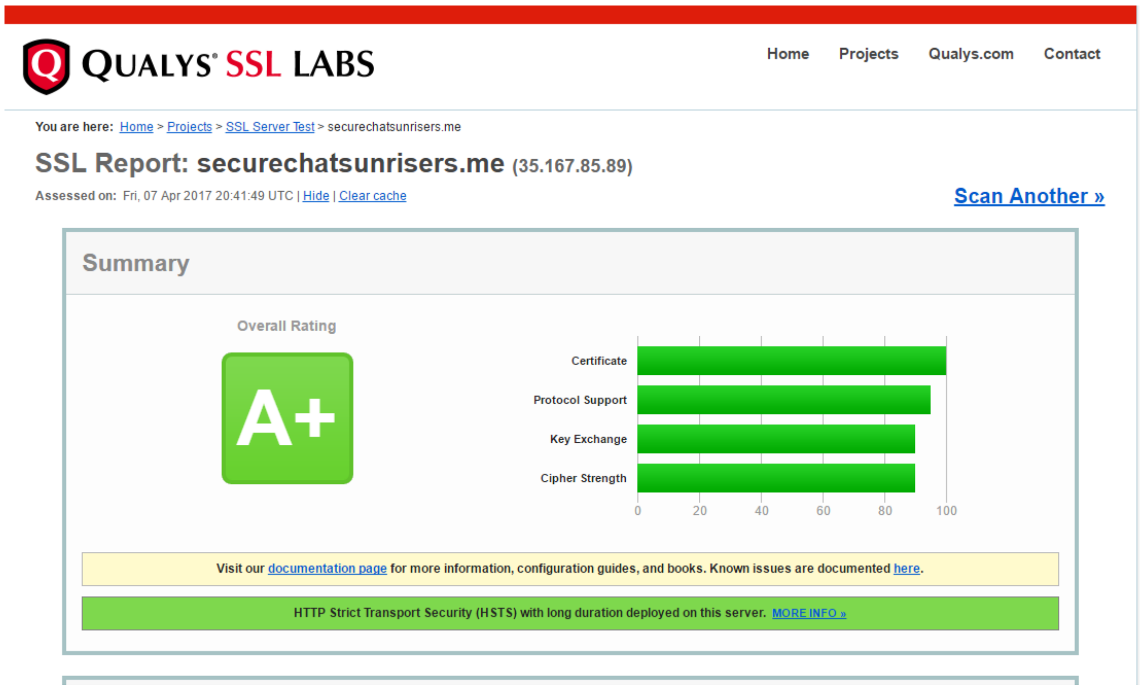


Encryption



Decryption

4.3 Server Implementation:



You are here: Home > Projects > SSL Server Test > securechatsunrisers.me

SSL Report: securechatsunrisers.me (35.167.85.89)

Assessed on: Fri, 07 Apr 2017 20:41:49 UTC | Hide | Clear cache

Scan Another »

Summary

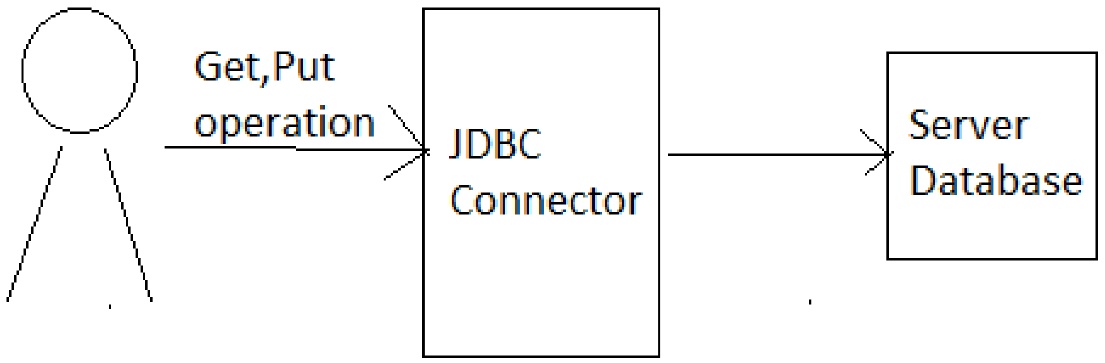
Overall Rating

A+

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. MORE INFO »

4.3.1 Chat Overview:



5 Conclusion and Future Work:

As there are many chat application in market to use. Each of them have different features which attract the users. This secure chat application developed provide complete end to end secure chat application. Only the intended receiver and sender only can read the messages. Message sent out of the client is encrypted with the one of th best algorithm present AES.

In future group chat can be developed.Some of the extra services like sharing audios, videos and attachment files can be achieved.

6 Bibliography:

1) https://whispersystems.org/

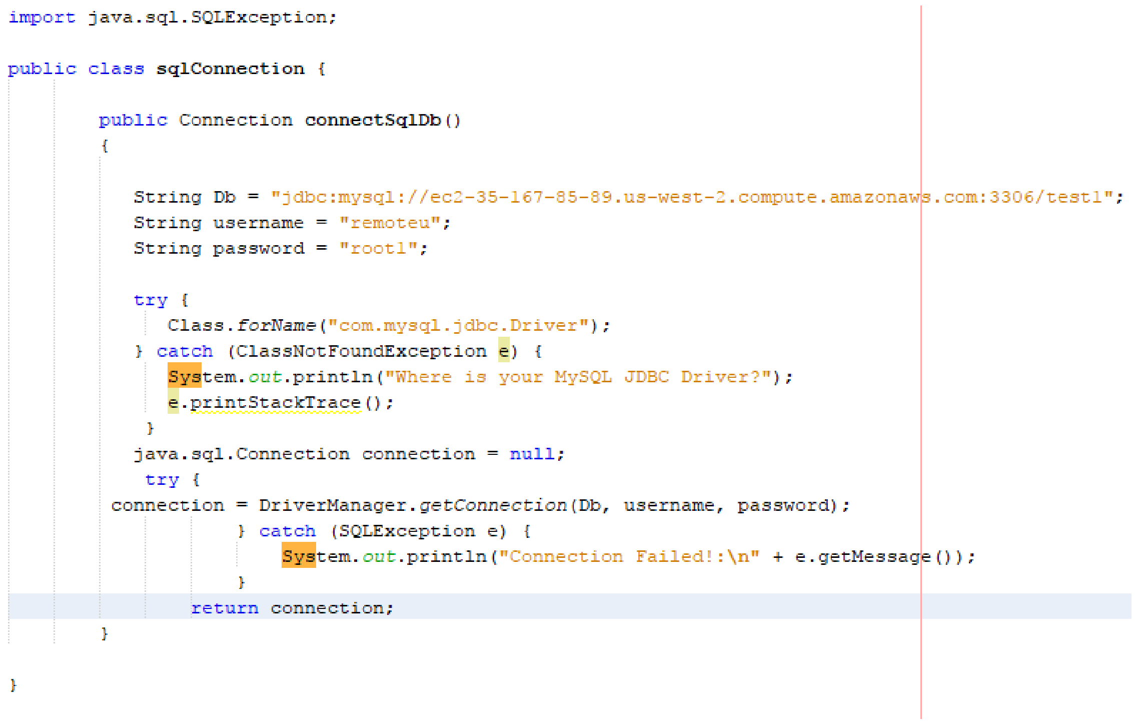
2) http://devoncmather.com/setting-aws-ec2-instance-lamp-git/

3) http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html

4)http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateWebS

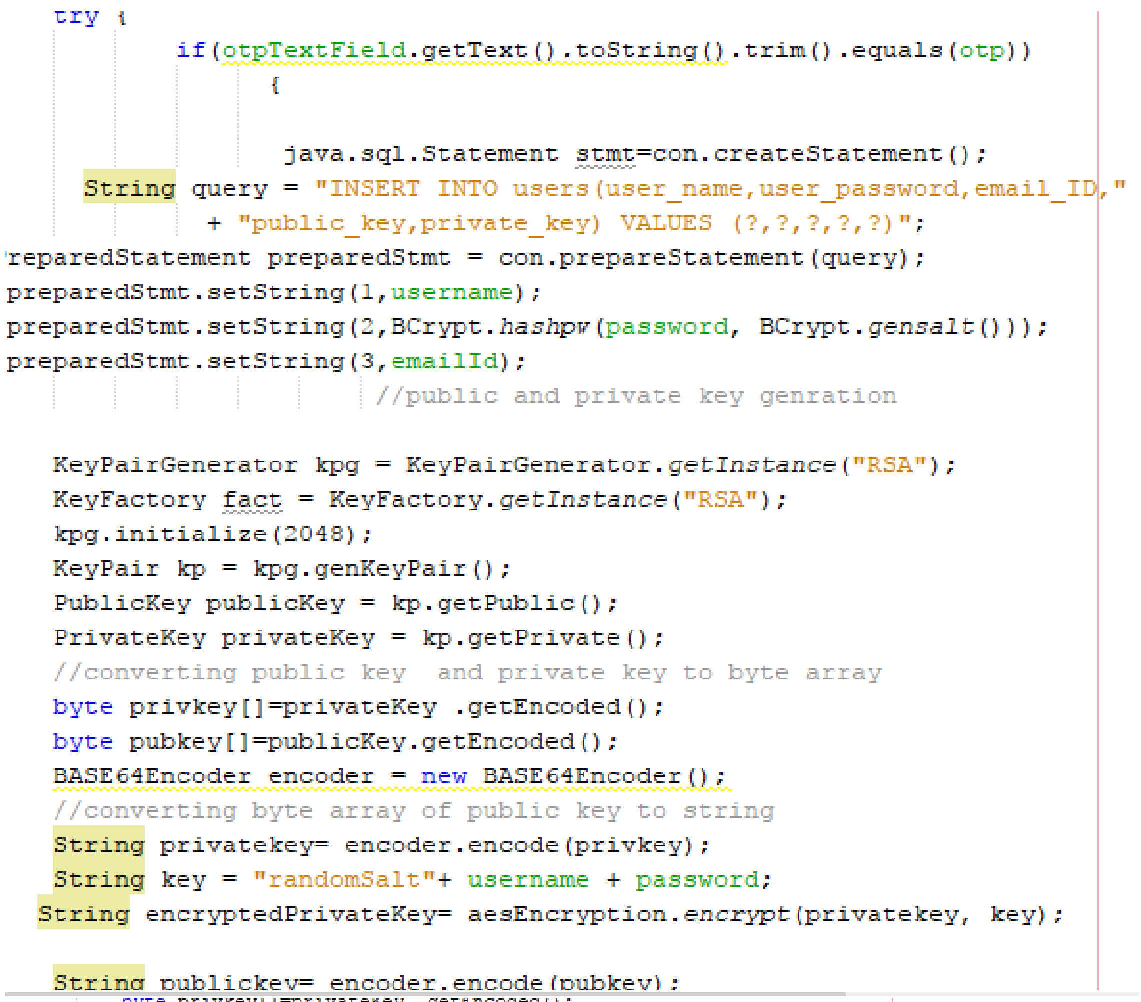
7 Appendix:

7.1 Server Connector:

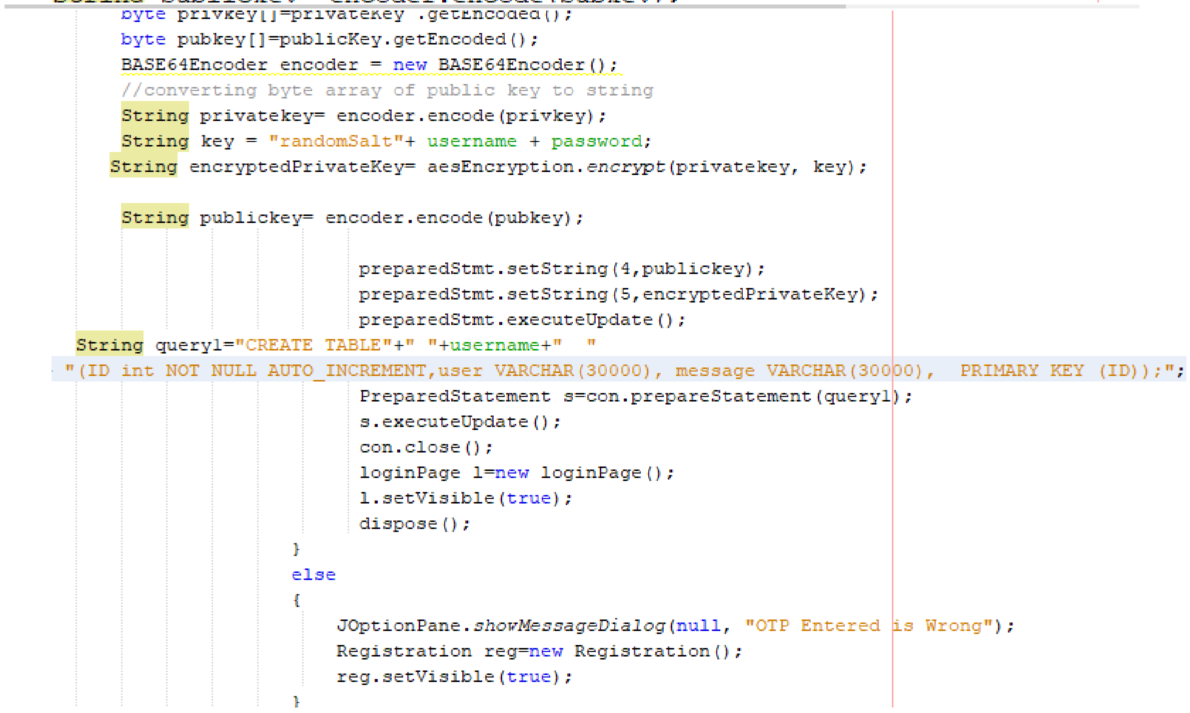


```
import java.sql.SQLException;

public class sqlConnection {

        public Connection connectSqlDb()
        {

           String Db = "jdbc:mysql://ec2-35-167-85-89.us-west-2.compute.amazonaws.com:3306/test1";
           String username = "remoteu";
           String password = "root1";

           try {
              Class.forName("com.mysql.jdbc.Driver");
           } catch (ClassNotFoundException e) {
              System.out.println("Where is your MySQL JDBC Driver?");
              e.printStackTrace();
            }
           java.sql.Connection connection = null;
            try {
         connection = DriverManager.getConnection(Db, username, password);
                       } catch (SQLException e) {
                          System.out.println("Connection Failed!:\n" + e.getMessage());
                       }
                   return connection;
        }

}
```

7.2 Registration:



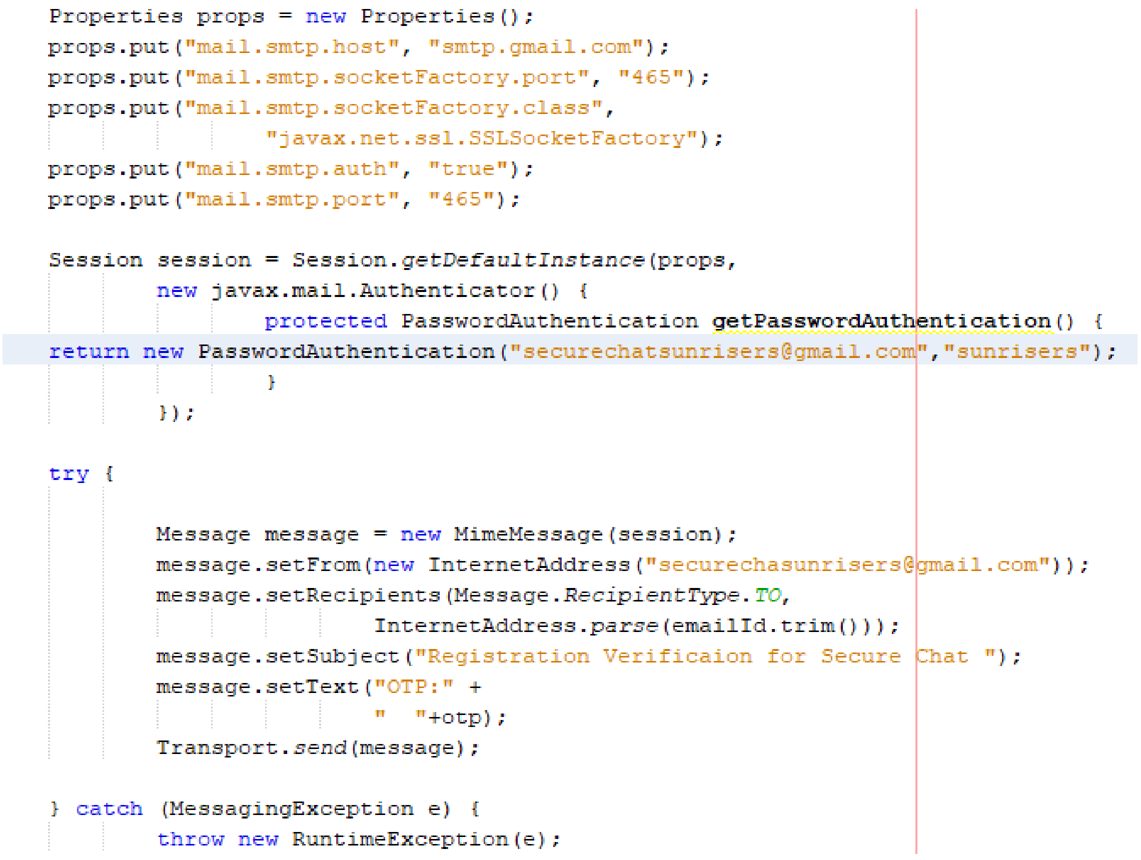
```
try {
        if(otpTextField.getText().toString().trim().equals(otp))
            {

             java.sql.Statement stmt=con.createStatement();
  String query = "INSERT INTO users(user_name,user_password,email_ID,"
          + "public_key,private_key) VALUES (?,?,?,?,?)";
reparedStatement preparedStmt = con.prepareStatement(query);
preparedStmt.setString(1,username);
preparedStmt.setString(2,BCrypt.hashpw(password, BCrypt.gensalt()));
preparedStmt.setString(3,emailId);
                          //public and private key genration

    KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
    KeyFactory fact = KeyFactory.getInstance("RSA");
    kpg.initialize(2048);
    KeyPair kp = kpg.genKeyPair();
    PublicKey publicKey = kp.getPublic();
    PrivateKey privateKey = kp.getPrivate();
    //converting public key  and private key to byte array
    byte privkey[]=privateKey .getEncoded();
    byte pubkey[]=publicKey.getEncoded();
    BASE64Encoder encoder = new BASE64Encoder();
    //converting byte array of public key to string
    String privatekey= encoder.encode(privkey);
    String key = "randomSalt"+ username + password;
   String encryptedPrivateKey= aesEncryption.encrypt(privatekey, key);

    String publickey= encoder.encode(pubkey);
```



```
        byte privkey[]=privateKey .getEncoded();
        byte pubkey[]=publicKey.getEncoded();
        BASE64Encoder encoder = new BASE64Encoder();
        //converting byte array of public key to string
        String privatekey= encoder.encode(privkey);
        String key = "randomSalt"+ username + password;
       String encryptedPrivateKey= aesEncryption.encrypt(privatekey, key);

        String publickey= encoder.encode(pubkey);
```

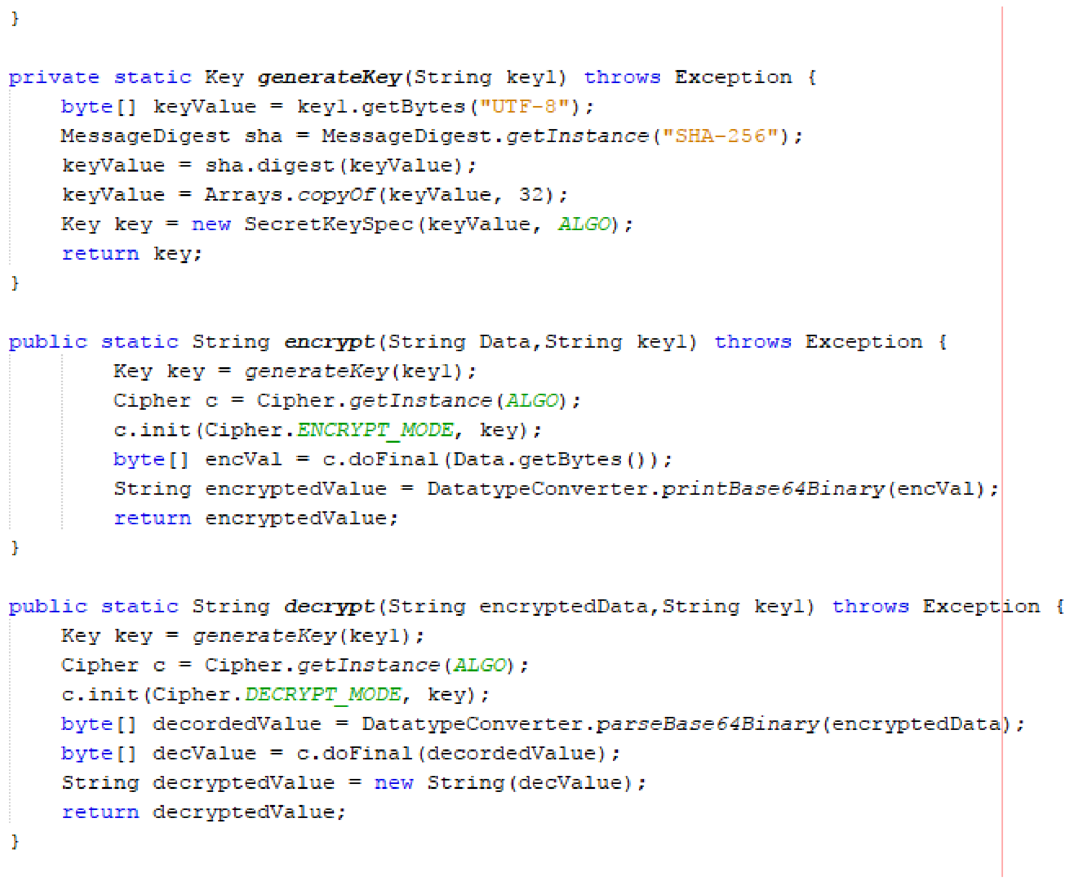
```
                     preparedStmt.setString(4,publickey);
                     preparedStmt.setString(5,encryptedPrivateKey);
                     preparedStmt.executeUpdate();
    String query1="CREATE TABLE"+" "+username+"  "
 "(ID int NOT NULL AUTO_INCREMENT,user VARCHAR(30000), message VARCHAR(30000),  PRIMARY KEY (ID));";
                     PreparedStatement s=con.prepareStatement(query1);
                     s.executeUpdate();
                     con.close();
                     loginPage l=new loginPage();
                     l.setVisible(true);
                     dispose();
                 }
                 else
                 {
                   JOptionPane.showMessageDialog(null, "OTP Entered is Wrong");
                   Registration reg=new Registration();
                   reg.setVisible(true);
                 }
```

7.3 Email Verification:



```
Properties props = new Properties();
props.put("mail.smtp.host", "smtp.gmail.com");
props.put("mail.smtp.socketFactory.port", "465");
props.put("mail.smtp.socketFactory.class",
                "javax.net.ssl.SSLSocketFactory");
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.port", "465");

Session session = Session.getDefaultInstance(props,
        new javax.mail.Authenticator() {
                protected PasswordAuthentication getPasswordAuthentication() {
return new PasswordAuthentication("securechatsunrisers@gmail.com","sunrisers");
                }
        });

try {

        Message message = new MimeMessage(session);
        message.setFrom(new InternetAddress("securechasunrisers@gmail.com"));
        message.setRecipients(Message.RecipientType.TO,
                        InternetAddress.parse(emailId.trim()));
        message.setSubject("Registration Verificaion for Secure Chat ");
        message.setText("OTP:" +
                        "   "+otp);
        Transport.send(message);

} catch (MessagingException e) {
        throw new RuntimeException(e);
```

7.4 Encrypting private key Using AES:



```
}

private static Key generateKey(String key1) throws Exception {
    byte[] keyValue = key1.getBytes("UTF-8");
    MessageDigest sha = MessageDigest.getInstance("SHA-256");
    keyValue = sha.digest(keyValue);
    keyValue = Arrays.copyOf(keyValue, 32);
    Key key = new SecretKeySpec(keyValue, ALGO);
    return key;
}

public static String encrypt(String Data,String key1) throws Exception {
        Key key = generateKey(key1);
        Cipher c = Cipher.getInstance(ALGO);
        c.init(Cipher.ENCRYPT_MODE, key);
        byte[] encVal = c.doFinal(Data.getBytes());
        String encryptedValue = DatatypeConverter.printBase64Binary(encVal);
        return encryptedValue;
}

public static String decrypt(String encryptedData,String key1) throws Exception {
    Key key = generateKey(key1);
    Cipher c = Cipher.getInstance(ALGO);
    c.init(Cipher.DECRYPT_MODE, key);
    byte[] decordedValue = DatatypeConverter.parseBase64Binary(encryptedData);
    byte[] decValue = c.doFinal(decordedValue);
    String decryptedValue = new String(decValue);
    return decryptedValue;
}
```