**ECC:**

package pro;

import java.math.\*;

import java.util.\*;

public class ECC {

// Parts of one ECC system.

private EllipticCurve curve;

private Point generator;

private Point publicKey;

private BigInteger privateKey;

// We need a curve, a generator point (x,y) and a private key, nA, that will

// be used to generate the public key.

public ECC(EllipticCurve c, BigInteger x, BigInteger y, BigInteger nA) {

curve = c;

generator = new Point(curve, x, y);

privateKey = nA;

publicKey = generator.multiply(privateKey);

}

// Encryption.

public Point[] encrypt(Point plain) {

// First we must pick a random k, in range.

int bits = curve.getA().bitLength();

BigInteger k = new BigInteger(bits, new Random());

System.out.println("Picked "+k+" as k for encrypting.");

// Our output is an ordered pair, (k\*generator, plain + k\*publickey)

Point[] ans = new Point[2];

ans[0] = generator.multiply(k);

ans[1] = plain.add(publicKey.multiply(k));

return ans;

}

// Decryption - notice the similarity to El Gamal!!!

public Point decrypt(Point[] cipher) {

// This is what we subtract out.

Point sub = cipher[0].multiply(privateKey);

// Subtract out and return.

return cipher[1].subtract(sub);

}

public String toString() {

return "Gen: "+generator+"\n"+

"pri: "+privateKey+"\n"+

"pub: "+publicKey;

}

public static void main(String[] args) {

// Just use the book's curve and test.

EllipticCurve myCurve = new EllipticCurve(new BigInteger("23"), new BigInteger("1"), new BigInteger("1"));

BigInteger x = new BigInteger("6");

BigInteger y = new BigInteger("19");

BigInteger nA = new BigInteger("10");

ECC Alice = new ECC(myCurve, x, y, nA);

// I have hard-coded my plaintext point.

Point plain = new Point(myCurve, new BigInteger("2"), new BigInteger("13"));

System.out.println("encrypting "+plain);

// Encrypt and print.

Point[] cipher = Alice.encrypt(plain);

System.out.println("cipher first part "+cipher[0]);

System.out.println("cipher second part "+cipher[1]);

// Decrypt and verify.

Point recover = Alice.decrypt(cipher);

System.out.println("recovered "+recover);

}

}

ECC MAIN:

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pro;

/\*\*

\*

\* @author RenownTechnologies

\*/

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 ECCMAIN{

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(strToEncrypt.getBytes("UTF-8")));

}

catch (Exception e)

{

System.out.println("Error while encrypting: " + e.toString());

}

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(strToDecrypt)));

}

catch (Exception e)

{

System.out.println("Error while decrypting: " + e.toString());

}

return null;

}

}

**ELIPTIC CURVE:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pro;

/\*\*

\*

\* @author RenownTechnologies

\*/

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 ECCMAIN{

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(strToEncrypt.getBytes("UTF-8")));

}

catch (Exception e)

{

System.out.println("Error while encrypting: " + e.toString());

}

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(strToDecrypt)));

}

catch (Exception e)

{

System.out.println("Error while decrypting: " + e.toString());

}

return null;

}

}

**EXAMPLE ECIES:**

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.PublicKey;

import java.security.SecureRandom;

import java.security.Security;

import javax.crypto.Cipher;

import javax.crypto.CipherInputStream;

import javax.crypto.CipherOutputStream;

import de.flexiprovider.common.ies.IESParameterSpec;

import de.flexiprovider.core.FlexiCoreProvider;

import de.flexiprovider.ec.FlexiECProvider;

import de.flexiprovider.ec.parameters.CurveParams;

import de.flexiprovider.ec.parameters.CurveRegistry.BrainpoolP160r1;

public class ExampleECIES {

public static void main(String[] args) throws Exception {

Security.addProvider(new FlexiCoreProvider());

Security.addProvider(new FlexiECProvider());

KeyPairGenerator kpg = KeyPairGenerator.getInstance("ECIES", "FlexiEC");

CurveParams ecParams = new BrainpoolP160r1();

kpg.initialize(ecParams, new SecureRandom());

KeyPair keyPair = kpg.generateKeyPair();

PublicKey pubKey = keyPair.getPublic();

PrivateKey privKey = keyPair.getPrivate();

// Encrypt

Cipher cipher = Cipher.getInstance("ECIES", "FlexiEC");

IESParameterSpec iesParams = new IESParameterSpec("AES128\_CBC",

"HmacSHA1", null, null);

cipher.init(Cipher.ENCRYPT\_MODE, pubKey, iesParams);

String cleartextFile = "cleartext.txt";

String ciphertextFile = "ciphertextECIES.txt";

byte[] block = new byte[64];

FileInputStream fis = new FileInputStream(cleartextFile);

FileOutputStream fos = new FileOutputStream(ciphertextFile);

CipherOutputStream cos = new CipherOutputStream(fos, cipher);

int i;

while ((i = fis.read(block)) != -1) {

cos.write(block, 0, i);

}

cos.close();

// Decrypt

String cleartextAgainFile = "cleartextAgainECIES.txt";

cipher.init(Cipher.DECRYPT\_MODE, privKey, iesParams);

fis = new FileInputStream(ciphertextFile);

CipherInputStream cis = new CipherInputStream(fis, cipher);

fos = new FileOutputStream(cleartextAgainFile);

while ((i = cis.read(block)) != -1) {

fos.write(block, 0, i);

}

fos.close();

}

}

**NEW CLASS :**

package pro;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.net.HttpURLConnection;

import java.net.URL;

import java.net.URLDecoder;

import java.net.URLEncoder;

import java.util.Iterator;

import java.util.Vector;

import javax.net.ssl.HttpsURLConnection;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author RenownTechnologies

\*/

public class NewClass {

public void send(String uid, String pwd, String phone, String msg) throws Exception {

if ((uid == null) || (uid.length() == 0)) {

throw new IllegalArgumentException("User ID should be present.");

}

uid = URLEncoder.encode(uid, "UTF-8");

if ((pwd == null) || (pwd.length() == 0)) {

throw new IllegalArgumentException("Password should be present.");

}

pwd = URLEncoder.encode(pwd, "UTF-8");

if ((phone == null) || (phone.length() == 0)) {

throw new IllegalArgumentException("At least one phone number should be present.");

}

if ((msg == null) || (msg.length() == 0)) {

throw new IllegalArgumentException("SMS message should be present.");

}

msg = URLEncoder.encode(msg, "UTF-8");

Vector numbers = new Vector();

if (phone.indexOf(59) >= 0) {

String[] pharr = phone.split(";");

for (String t : pharr) {

try {

numbers.add(Long.valueOf(t));

} catch (NumberFormatException ex) {

throw new IllegalArgumentException("Give proper phone numbers.");

}

}

} else {

try {

numbers.add(Long.valueOf(phone));

} catch (NumberFormatException ex) {

throw new IllegalArgumentException("Give proper phone numbers.");

}

}

if (numbers.size() == 0) {

throw new IllegalArgumentException("At least one proper phone number should be present to send SMS.");

}

String temp = "";

String content = "username=" + uid + "&password=" + pwd;

URL u = new URL("http://www.vnssms.in/quicksms/api.php");

HttpURLConnection uc = (HttpURLConnection) u.openConnection();

uc.setDoOutput(true);

uc.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.5) Gecko/2008120122 Firefox/3.0.5");

uc.setRequestProperty("Content-Length", String.valueOf(content.length()));

uc.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

uc.setRequestProperty("Accept", "\*/\*");

uc.setRequestProperty("Referer", "http://www.vnssms.in/quicksms/index.php");

uc.setRequestMethod("POST");

uc.setInstanceFollowRedirects(false);

PrintWriter pw = new PrintWriter(new OutputStreamWriter(uc.getOutputStream()), true);

pw.print(content);

pw.flush();

pw.close();

BufferedReader br = new BufferedReader(new InputStreamReader(uc.getInputStream()));

while ((temp = br.readLine()) != null) {

System.out.println(temp);

}

String cookie = uc.getHeaderField("Set-Cookie");

u = null;

uc = null;

for (Iterator localIterator = numbers.iterator(); localIterator.hasNext();) {

long num = ((Long) localIterator.next()).longValue();

content = "custid=undefined&HiddenAction=instantsms&Action=custfrom450000&login=&pass=&MobNo=" + num + "&textArea=" + msg;

u = new URL("http://www.vnssms.in/quicksms/api.php?username=maxbulksms&password=12345678&to="+num+"&from=AUCSNW&message="+msg);

uc = (HttpURLConnection) u.openConnection();

uc.setDoOutput(true);

uc.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.5) Gecko/2008120122 Firefox/3.0.5");

uc.setRequestProperty("Content-Length", String.valueOf(content.getBytes().length));

uc.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

uc.setRequestProperty("Accept", "\*/\*");

uc.setRequestProperty("Cookie", cookie);

uc.setRequestMethod("POST");

uc.setInstanceFollowRedirects(false);

pw = new PrintWriter(new OutputStreamWriter(uc.getOutputStream()), true);

pw.print(content);

pw.flush();

pw.close();

br = new BufferedReader(new InputStreamReader(uc.getInputStream()));

while ((temp = br.readLine()) != null);

br.close();

u = null;

uc = null;

}

u = new URL("http://www.vnssms.in/quicksms/index.php");

uc = (HttpURLConnection) u.openConnection();

uc.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.5) Gecko/2008120122 Firefox/3.0.5");

uc.setRequestProperty("Accept", "\*/\*");

uc.setRequestProperty("Cookie", cookie);

uc.setRequestMethod("GET");

uc.setInstanceFollowRedirects(false);

br = new BufferedReader(new InputStreamReader(uc.getInputStream()));

while ((temp = br.readLine()) != null);

br.close();

u = null;

uc = null;

}

}

**POINT:**

package pro;

import java.math.\*;

public class Point {

// Store the x, y and curve.

private BigInteger x;

private BigInteger y;

private EllipticCurve curve;

// Precondition: (myX, myY) must lie on the curve c. I don't check that here!!!

public Point(EllipticCurve c, BigInteger myX, BigInteger myY) {

x = myX;

y = myY;

curve = c;

}

// Copy constructor.

public Point(Point copy) {

x = new BigInteger(copy.x.toString());

y = new BigInteger(copy.y.toString());

curve = new EllipticCurve(copy.curve);

}

// Returns 0. Not sure if this is the proper way to store the "origin".

public Point(EllipticCurve c) {

curve = c;

x = BigInteger.ZERO;

y = BigInteger.ZERO;

}

// All components must be equal...

public boolean equals(Point other) {

return x.equals(other.x) && y.equals(other.y) && curve.equals(other.curve);

}

// Returns true iff other is this point's reflection over the line y = p/2 (real division)

public boolean mirror(Point other) {

return x.equals(other.x) && curve.equals(other.curve) && y.equals(other.curve.getP().subtract(other.y));

}

// Returns the negative of this point, which is its mirror.

public Point negate() {

BigInteger newY = curve.getP().subtract(y);

return new Point(curve, x, newY);

}

// Adds this to other and returns the answer, using the formulas in Stallings (5th edition)

public Point add(Point other) {

// Can't add points on different curves.

if (!curve.equals(other.curve)) return null;

if (this.equals(other)) {

// Avoid adding to the origin...

if (this.isOrigin()) return new Point(this);

// We need these to calculate lambda.

BigInteger three = new BigInteger("3");

BigInteger two = new BigInteger("2");

BigInteger temp = new BigInteger(x.toString());

// Splitting up the calculation of lambda into all of these steps...

BigInteger lambda = temp.modPow(two, curve.getP());

lambda = three.multiply(lambda);

lambda = lambda.add(curve.getA());

BigInteger den = two.multiply(y);

lambda = lambda.multiply(den.modInverse(curve.getP()));

// Once we have lambda, just plug into these equations.

BigInteger newX = lambda.multiply(lambda).subtract(x).subtract(x).mod(curve.getP());

BigInteger newY = (lambda.multiply(x.subtract(newX))).subtract(y).mod(curve.getP());

return new Point(curve, newX, newY);

}

// Returns the origin...not sure if my origin is correct.

else if (this.mirror(other)) {

return new Point(curve);

}

// Standard case.

else {

// Avoid adding by the identity element.

if (this.isOrigin()) return new Point(other);

if (other.isOrigin()) return new Point(this);

// We need these to calculate lambda.

BigInteger three = new BigInteger("3");

BigInteger two = new BigInteger("2");

BigInteger temp = new BigInteger(x.toString());

// Lambda's a bit easier here...

BigInteger lambda = other.y.subtract(y);

BigInteger den = other.x.subtract(x);

lambda = lambda.multiply(den.modInverse(curve.getP()));

// This calculation is roughly the same as above.

BigInteger newX = lambda.multiply(lambda).subtract(x).subtract(other.x).mod(curve.getP());

BigInteger newY = (lambda.multiply(x.subtract(newX))).subtract(y).mod(curve.getP());

return new Point(curve, newX, newY);

}

}

// Subtraction is just adding the negative.

public Point subtract(Point other) {

other = other.negate();

return this.add(other);

}

// Uses "fast multiplication" to multiply this point by factor.

public Point multiply(BigInteger factor) {

BigInteger two = new BigInteger("2");

// Base cases.

if (factor.equals(BigInteger.ONE))

return new Point(this);

if (factor.equals(two))

return this.add(this);

// Even case where we can calculate half of our answer and multiply by 2!

if (factor.mod(two).equals(BigInteger.ZERO)) {

Point sqrt = multiply(factor.divide(two));

return sqrt.add(sqrt);

}

// No speed up here, but this recursive call will lead to one.

else {

factor = factor.subtract(BigInteger.ONE);

return this.add(multiply(factor));

}

}

public boolean isOrigin() {

return x.equals(BigInteger.ZERO) && y.equals(BigInteger.ZERO);

}

public String toString() {

return "(" + x +", "+y+")";

}

public static void main(String[] args) {

// Test out Point arithmetic.

EllipticCurve myCurve = new EllipticCurve(new BigInteger("23"), new BigInteger("1"), new BigInteger("1"));

Point p = new Point(myCurve, new BigInteger("3"), new BigInteger("10"));

Point q = new Point(myCurve, new BigInteger("9"), new BigInteger("7"));

// P + Q

Point pPlusq = p.add(q);

System.out.println(p+" + "+q+" = "+pPlusq);

// 2P

Point twoP = p.add(p);

System.out.println("2\* "+p+" = "+twoP);

// Test multiplication =)

Point fourP = p.multiply(new BigInteger("4"));

System.out.println("Four times p = "+fourP);

Point checkFourP = twoP.add(twoP);

System.out.println("check Four times p = "+checkFourP);

}

}

**RAHUL:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pro;

/\*\*

\*

\* @author RenownTechnologies

\*/

/\* Code for Generating key Pair \*/

import java.io.\*;

import java.security.\*;

import java.security.spec.\*;

public class Rahul {

public static void main(String args[]) {

Rahul rahul = new Rahul();

try {

String path = "D:\\rp";

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");

keyGen.initialize(1024);

KeyPair generatedKeyPair = keyGen.genKeyPair();

System.out.println("Generated Key Pair");

rahul.dumpKeyPair(generatedKeyPair);

rahul.SaveKeyPair(path, generatedKeyPair);

KeyPair loadedKeyPair = rahul.LoadKeyPair(path, "DSA");

System.out.println("Loaded Key Pair");

rahul.dumpKeyPair(loadedKeyPair);

} catch (Exception e) {

e.printStackTrace();

return;

}

}

private void dumpKeyPair(KeyPair keyPair) {

PublicKey pub = keyPair.getPublic();

System.out.println("Public Key: " + getHexString(pub.getEncoded()));

PrivateKey priv = keyPair.getPrivate();

System.out.println("Private Key: " + getHexString(priv.getEncoded()));

}

private String getHexString(byte[] b) {

String result = "";

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

result += Integer.toString((b[i] & 0xff) + 0x100, 16).substring(1);

}

return result;

}

public void SaveKeyPair(String path, KeyPair keyPair) throws IOException {

PrivateKey privateKey = keyPair.getPrivate();

PublicKey publicKey = keyPair.getPublic();

// Store Public Key.

X509EncodedKeySpec x509EncodedKeySpec = new X509EncodedKeySpec(

publicKey.getEncoded());

FileOutputStream fos = new FileOutputStream(path + "/public.key");

fos.write(x509EncodedKeySpec.getEncoded());

fos.close();

// Store Private Key.

PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new PKCS8EncodedKeySpec(

privateKey.getEncoded());

fos = new FileOutputStream(path + "/private.key");

fos.write(pkcs8EncodedKeySpec.getEncoded());

fos.close();

}

public KeyPair LoadKeyPair(String path, String algorithm)

throws IOException, NoSuchAlgorithmException,

InvalidKeySpecException {

// Read Public Key.

File filePublicKey = new File(path + "/public.key");

FileInputStream fis = new FileInputStream(path + "/public.key");

byte[] encodedPublicKey = new byte[(int) filePublicKey.length()];

fis.read(encodedPublicKey);

fis.close();

// Read Private Key.

File filePrivateKey = new File(path + "/private.key");

fis = new FileInputStream(path + "/private.key");

byte[] encodedPrivateKey = new byte[(int) filePrivateKey.length()];

fis.read(encodedPrivateKey);

fis.close();

// Generate KeyPair.

KeyFactory keyFactory = KeyFactory.getInstance(algorithm);

X509EncodedKeySpec publicKeySpec = new X509EncodedKeySpec(

encodedPublicKey);

PublicKey publicKey = keyFactory.generatePublic(publicKeySpec);

PKCS8EncodedKeySpec privateKeySpec = new PKCS8EncodedKeySpec(

encodedPrivateKey);

PrivateKey privateKey = keyFactory.generatePrivate(privateKeySpec);

return new KeyPair(publicKey, privateKey);

}

}

**REGISTER:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pro;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.Random;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import mail.mail;

/\*\*

\*

\* @author David

\*/

public class Register extends HttpServlet {

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

\* methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String name,pass,mobile,email,address,type,state,country;

name=request.getParameter("T1");

pass=request.getParameter("Password");

mobile=request.getParameter("mobile");

email=request.getParameter("Email");

address=request.getParameter("Address");

state=request.getParameter("State");

country=request.getParameter("Country");

String userkey="";

for(int i=0;i<email.length();i++)

{

userkey+=email.charAt(i);

if(i==2)

{

break;

}

}

for(int i=0;i<name.length();i++)

{

userkey+=name.charAt(i);

if(i==1)

{

break;

}

}

Random rr=new Random();

String uk=userkey;

userkey="";

int key1=rr.nextInt(99999);

int key2=rr.nextInt(99999);

int key3=rr.nextInt(99999);

userkey+=key1;

userkey+="-"+key2;

userkey+="-"+key3;

System.out.println("This is User Key "+uk+"-"+userkey);

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/bitcoindb","root","root");

PreparedStatement one=con.prepareStatement("insert into tbl\_register(name,pwd,mobile,email,address,state,country,walletid,dt\_created,status)values('"+name+"','"+pass+"','"+mobile+"','"+email+"','"+address+"','"+state+"','"+country+"','"+uk+"-"+userkey+"',now(),1)");

one.executeUpdate();

//---------------------mail part ----------------------//

String em[]= new String[1];

String subject="Wallet Credentials";

String message="Name : "+name

+"\nWallet - ID : "+uk+"-"+userkey

+"\nPassword : "+pass;

String fr="danithoneap"; //with out @gmail.com

String pw="dani@123"; // sender password

em[0]=email;

mail mmm=new mail();

mmm.sendFromGMail(fr, pw, em, subject, message);

System.out.println("email "+em[0]);

//-----------------completed mail part --------------------//

try

{

NewClass n=new NewClass();

n.send("maxbulksms", "12345678", mobile, message);

}

catch(Exception ex)

{

System.out.println(ex);

}

out.println("<script type=\"text/javascript\">");

out.println("alert(\"Successfully registered\")");

out.println("</script>");

RequestDispatcher rd=request.getRequestDispatcher("signupsucess.jsp");

rd.include(request, response);

}

catch (ClassNotFoundException | SQLException e)

{

System.out.println(e);

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**SHA512CRYPTO:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pro;

import java.security.\*;

/\*\*

\*

\* @author RenownTechnologies

\*/

public class SHA512cryptotest {

public String SHAENCY(String msg)

{

MessageDigest md;

String message = msg;String out = "";

try {

md= MessageDigest.getInstance("SHA-512");

md.update(message.getBytes());

byte[] mb = md.digest();

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

byte temp = mb[i];

String s = Integer.toHexString(new Byte(temp));

while (s.length() < 2) {

s = "0" + s;

}

s = s.substring(s.length() - 2);

out += s;

}

System.out.println(out.length());

System.out.println("CRYPTO: " + out);

} catch (NoSuchAlgorithmException e) {

System.out.println("ERROR: " + e.getMessage());

}

return out;

}

public static void main(String[] args) throws NoSuchAlgorithmException {

MessageDigest md;

String message = "password";

try {

md= MessageDigest.getInstance("SHA-512");

md.update(message.getBytes());

byte[] mb = md.digest();

String out = "";

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

byte temp = mb[i];

String s = Integer.toHexString(new Byte(temp));

while (s.length() < 2) {

s = "0" + s;

}

s = s.substring(s.length() - 2);

out += s;

}

System.out.println(out.length());

System.out.println("CRYPTO: " + out);

} catch (NoSuchAlgorithmException e) {

System.out.println("ERROR: " + e.getMessage());

}

}

}

SMS:

package pro;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.net.HttpURLConnection;

import java.net.URL;

import java.net.URLEncoder;

import java.util.Iterator;

import java.util.Vector;

/\*\*

\*

\* @author user

\*/

public class SMSS extends HttpServlet {

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

\* methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String name = request.getParameter("uname");

String x = request.getParameter("form\_x");

String y = request.getParameter("form\_y");

System.out.println("1");

try {

// String s[]=request.getParameterValues("id");

String uid="maxbulksms"; String pwd="12345678"; String phone=""; String msg="";

String temp = "";

if ((uid == null) || (uid.length() == 0)) {

throw new IllegalArgumentException("User ID should be present.");

}

uid = URLEncoder.encode(uid, "UTF-8");

if ((pwd == null) || (pwd.length() == 0)) {

throw new IllegalArgumentException("Password should be present.");

}

pwd = URLEncoder.encode(pwd, "UTF-8");

if ((phone == null) || (phone.length() == 0)) {

throw new IllegalArgumentException("At least one phone number should be present.");

}

if ((msg == null) || (msg.length() == 0)) {

throw new IllegalArgumentException("SMS message should be present.");

}

msg = URLEncoder.encode(msg, "UTF-8");

Vector numbers = new Vector();

if (phone.indexOf(59) >= 0) {

String[] pharr = phone.split(";");

for (String t : pharr) {

try {

numbers.add(Long.valueOf(t));

} catch (NumberFormatException ex) {

throw new IllegalArgumentException("Give proper phone numbers.");

}

}

} else {

try {

numbers.add(Long.valueOf(phone));

} catch (NumberFormatException ex) {

throw new IllegalArgumentException("Give proper phone numbers.");

}

}

if (numbers.size() == 0) {

throw new IllegalArgumentException("At least one proper phone number should be present to send SMS.");

}

///String temp = "";

String content = "username=" + uid + "&password=" + pwd;

URL u = new URL("http://www.vnssms.in/quicksms/api.php");

HttpURLConnection uc = (HttpURLConnection) u.openConnection();

uc.setDoOutput(true);

uc.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.5) Gecko/2008120122 Firefox/3.0.5");

uc.setRequestProperty("Content-Length", String.valueOf(content.length()));

uc.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

uc.setRequestProperty("Accept", "\*/\*");

uc.setRequestProperty("Referer", "http://www.vnssms.in/quicksms/index.php");

uc.setRequestMethod("POST");

uc.setInstanceFollowRedirects(false);

PrintWriter pw = new PrintWriter(new OutputStreamWriter(uc.getOutputStream()), true);

pw.print(content);

pw.flush();

pw.close();

BufferedReader br = new BufferedReader(new InputStreamReader(uc.getInputStream()));

while ((temp = br.readLine()) != null) {

System.out.println(temp);

}

String cookie = uc.getHeaderField("Set-Cookie");

u = null;

uc = null;

for (Iterator localIterator = numbers.iterator(); localIterator.hasNext();) {

long num = ((Long) localIterator.next()).longValue();

content = "custid=undefined&HiddenAction=instantsms&Action=custfrom450000&login=&pass=&MobNo=" + num + "&textArea=" + msg;

u = new URL("http://www.vnssms.in/quicksms/api.php?username=maxbulksms&password=12345678&to="+num+"&from=SPCATM&message="+msg);

uc = (HttpURLConnection) u.openConnection();

uc.setDoOutput(true);

uc.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.5) Gecko/2008120122 Firefox/3.0.5");

uc.setRequestProperty("Content-Length", String.valueOf(content.getBytes().length));

uc.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

uc.setRequestProperty("Accept", "\*/\*");

uc.setRequestProperty("Cookie", cookie);

uc.setRequestMethod("POST");

uc.setInstanceFollowRedirects(false);

pw = new PrintWriter(new OutputStreamWriter(uc.getOutputStream()), true);

pw.print(content);

pw.flush();

pw.close();

br = new BufferedReader(new InputStreamReader(uc.getInputStream()));

while ((temp = br.readLine()) != null);

br.close();

u = null;

uc = null;

}

u = new URL("http://www.vnssms.in/quicksms/index.php");

uc = (HttpURLConnection) u.openConnection();

uc.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.5) Gecko/2008120122 Firefox/3.0.5");

uc.setRequestProperty("Accept", "\*/\*");

uc.setRequestProperty("Cookie", cookie);

uc.setRequestMethod("GET");

uc.setInstanceFollowRedirects(false);

br = new BufferedReader(new InputStreamReader(uc.getInputStream()));

while ((temp = br.readLine()) != null);

br.close();

u = null;

uc = null;

// getServletContext().getRequestDispatcher("http://localhost:8080/E-Election/ClickPoint2a.jsp").forward(request, response);

// out.print("<table width='100%'><tr width='100%'><td width='100%'><img src='./images/Banner.jpg' width='100%'> <td> </td></tr></table><center>");

//out.print("<h2>User registered successfully</h2><br><a href='index.jsp'>Click here to login</a></center>");

//response.sendRedirect("http://localhost:8080/E-Election/ClickPoint2a.jsp");

} catch (Exception e) {

System.out.println("Driver not found" +e);//

}

finally {

out.close();

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**BUY:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pro;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.security.SecureRandom;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import javax.servlet.RequestDispatcher;

/\*\*

\*

\* @author RenownTechnologies

\*/

public class buy extends HttpServlet {

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

\* methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String noc,wd;

noc=request.getParameter("nob");

try

{

String walletid=(String)request.getSession(true).getAttribute("walletid");

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/bitcoindb","root","root");

PreparedStatement one=con.prepareStatement("insert into tbl\_coins(walletid,noofcoins,status,dt\_created,type) values('"+walletid+"',"+noc+",0,now(),1)");

one.execute();

RequestDispatcher rd=request.getRequestDispatcher("buyprovess.jsp?");

rd.include(request, response);

//buyprovess

}

catch(Exception ex)

{

System.out.println(ex);

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**DECRYE:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pro;

/\*\*

\*

\* @author RenownTechnologies

\*/

/\* code for Decryption \*/

import java.io.File;

import java.io.FileInputStream;

import java.security.Key;

import java.security.KeyFactory;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.PublicKey;

import java.security.Security;

import java.security.spec.ECParameterSpec;

import java.security.spec.EllipticCurve;

import java.security.spec.KeySpec;

import java.security.spec.PKCS8EncodedKeySpec;

import java.security.spec.X509EncodedKeySpec;

import javax.crypto.Cipher;

import javax.crypto.KeyAgreement;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESKeySpec;

import org.apache.commons.codec.binary.Base64;

import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class decryec {

KeyPairGenerator kpg;

EllipticCurve curve;

ECParameterSpec ecSpec;

KeyPair aKeyPair;

static KeyAgreement aKeyAgree;

KeyPair bKeyPair;

KeyAgreement bKeyAgree;

KeyFactory keyFac;

public static void main(String args[])

{

Security.addProvider(new BouncyCastleProvider());

try{

String path = "D:\\X";

File filePublicKey = new File(path +"\\public.key");

FileInputStream fis = new FileInputStream(path + "\\public.key");

byte[] encodedPublicKey = new byte[(int) filePublicKey.length()];

fis.read(encodedPublicKey);

fis.close();

// Read Private Key.

File filePrivateKey = new File(path + "\\private.key");

fis = new FileInputStream(path + "\\private.key");

byte[] encodedPrivateKey = new byte[(int) filePrivateKey.length()];

fis.read(encodedPrivateKey);

fis.close();

// Generate KeyPair.

KeyFactory keyFactory = KeyFactory.getInstance("ECDH");

X509EncodedKeySpec publicKeySpec = new X509EncodedKeySpec(

encodedPublicKey);

PublicKey publicKey = keyFactory.generatePublic(publicKeySpec);

PKCS8EncodedKeySpec privateKeySpec = new PKCS8EncodedKeySpec(

encodedPrivateKey);

PrivateKey privateKey = keyFactory.generatePrivate(privateKeySpec);

aKeyAgree = KeyAgreement.getInstance("ECDH", "BC");

aKeyAgree.init(privateKey);

aKeyAgree.doPhase(publicKey, true);

byte[] aBys = aKeyAgree.generateSecret();

KeySpec aKeySpec = new DESKeySpec(aBys);

SecretKeyFactory aFactory = SecretKeyFactory.getInstance("DES");

Key aSecretKey = aFactory.generateSecret(aKeySpec);

Cipher aCipher = Cipher.getInstance(aSecretKey.getAlgorithm());

aCipher.init(Cipher.DECRYPT\_MODE, aSecretKey);

byte[] decText = aCipher.doFinal(Base64.decodeBase64("0wwerdjkHbVhYI+YPxUnmw==".getBytes()));

String text = new String(decText);

System.out.println("Decoded="+text);

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

**ENCRYEC:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pro;

/\*\*

\*

\* @author RenownTechnologies

\*/

/\* Code for Encryption \*/

import java.io.File;

import java.io.FileInputStream;

import java.security.Key;

import java.security.KeyFactory;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.PublicKey;

import java.security.Security;

import java.security.spec.ECParameterSpec;

import java.security.spec.EllipticCurve;

import java.security.spec.KeySpec;

import java.security.spec.PKCS8EncodedKeySpec;

import java.security.spec.X509EncodedKeySpec;

import java.util.Scanner;

import javax.crypto.Cipher;

import javax.crypto.KeyAgreement;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESKeySpec;

import org.apache.commons.codec.binary.Base64;

import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class encryec {

KeyPairGenerator kpg;

EllipticCurve curve;

ECParameterSpec ecSpec;

KeyPair aKeyPair;

static KeyAgreement aKeyAgree;

KeyPair bKeyPair;

KeyAgreement bKeyAgree;

KeyFactory keyFac;

static String msg;

public static void main(String args[])

{

Security.addProvider(new BouncyCastleProvider());

//Scanner ss=new Scanner(System.in);

try{

String path = "D:\\rp";

File filePublicKey = new File(path+"\\public.key");

FileInputStream fis = new FileInputStream(path+"\\public.key");

byte[] encodedPublicKey = new byte[(int) filePublicKey.length()];

fis.read(encodedPublicKey);

fis.close();

// Read Private Key.

File filePrivateKey = new File(path+"\\private.key");

fis = new FileInputStream(path+"\\private.key");

byte[] encodedPrivateKey = new byte[(int) filePrivateKey.length()];

fis.read(encodedPrivateKey);

fis.close();

// Generate KeyPair.

KeyFactory keyFactory = KeyFactory.getInstance("ECDH");

X509EncodedKeySpec publicKeySpec = new X509EncodedKeySpec(

encodedPublicKey);

PublicKey publicKey = keyFactory.generatePublic(publicKeySpec);

PKCS8EncodedKeySpec privateKeySpec = new PKCS8EncodedKeySpec(

encodedPrivateKey);

PrivateKey privateKey = keyFactory.generatePrivate(privateKeySpec);

aKeyAgree = KeyAgreement.getInstance("ECDH", "BC");

aKeyAgree.init(privateKey);

aKeyAgree.doPhase(publicKey, true);

byte[] aBys = aKeyAgree.generateSecret();

KeySpec aKeySpec = new DESKeySpec(aBys);

SecretKeyFactory aFactory = SecretKeyFactory.getInstance("DES");

Key aSecretKey = aFactory.generateSecret(aKeySpec);

Cipher aCipher = Cipher.getInstance(aSecretKey.getAlgorithm());

aCipher.init(Cipher.ENCRYPT\_MODE, aSecretKey);

byte[] encText = aCipher.doFinal("Its Rahul".getBytes());

System.out.println(Base64.encodeBase64String(encText));

System.out.println(encText);

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

**SIGN CHECK:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pro;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.\*;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.Random;

import javax.mail.Session;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import mail.mail;

/\*\*

\*

\* @author RenownTechnologies

\*/

public class signncheck extends HttpServlet {

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

\* methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String pwd,wd;

pwd=request.getParameter("pwd");

wd=request.getParameter("wd");

int flag=0;

if(wd.equalsIgnoreCase("admin") && pwd.equalsIgnoreCase("12345"))

{

response.sendRedirect("admin/dashboard.jsp");

flag=9;

}

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/bitcoindb","root","root");

PreparedStatement one=con.prepareStatement("select \* from tbl\_register where walletid='"+wd+"' and pwd='"+pwd+"'");

ResultSet rs=one.executeQuery();

while(rs.next())

{

flag=9;

String id=rs.getString("id");

String name=rs.getString("name");

String mob=rs.getString("Mobile");

String email=rs.getString("email");

request.getSession(true).setAttribute("wname", name);

request.getSession(true).setAttribute("ownerid", id);

request.getSession(true).setAttribute("walletid", wd);

request.getSession(true).setAttribute("pwd", pwd);

request.getSession(true).setAttribute("mob", mob);

request.getSession(true).setAttribute("email", email);

RequestDispatcher rd=request.getRequestDispatcher("dashboard.jsp?id="+id+"&wid="+wd);

rd.include(request, response);

}

}

catch(Exception ex)

{

System.out.println(ex);

}

if(flag==0)

{

response.sendRedirect("invalidlogin.jsp");

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}