**MỤC LỤC**

[1. Sử dụng Java Cryptography Architecture (JCA), viết class SymmetricCryp để thực hiện các chức năng mã hóa và giải mã đối xứng theo thuật toán AES gồm các hàm 3](#_Toc152673942)

[1.1.Sinh khóa đối xứng 3](#_Toc152673943)

[1.2.Tạo khóa đối xứng từ một chuỗi cho trước 3](#_Toc152673944)

[1.3.Mã hóa thông điệp: Tham số 1 là chuỗi cần mã hóa, tham số thứ 2 là khóa được tạo ở 1.1 4](#_Toc152673945)

[1.4.Mã hóa thông điệp: Tham số 1 là chuỗi cần mã hóa tham số 2 là khóa được tạo ở 1.2 4](#_Toc152673946)

[1.5.Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.1 5](#_Toc152673947)

[1.6.Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.2 5](#_Toc152673948)

[1.7.Sử dụng class SymetricCryp để mã hóa một chuỗi cho trước hoặc giải mã ra bản rõ 6](#_Toc152673949)

[2. Sử dụng Java Cryptography Architecture (JCA) để viết các hàm băm thông điệp cho trước 7](#_Toc152673950)

[2.1. MD5 7](#_Toc152673951)

[2.2. SHA1 7](#_Toc152673952)

[2.3. SHA-256 8](#_Toc152673953)

[3. Sử dụng Java Crypto Architecture (JCA), viết chương trình mã hóa và giải mã sử dụng RSA 8](#_Toc152673954)

[3.1 Sinh bộ khóa theo thuật toán RSA 9](#_Toc152673955)

[3.2 Lưu bộ khóa ra file 9](#_Toc152673956)

[3.3 Hàm đọc file để nạp khóa bí mật, tham số truyền vào là tên file 10](#_Toc152673957)

[3.4 Hàm đọc file để nạp khóa công khai, tham số truyền vào là tên file 10](#_Toc152673958)

[3.5 Hàm mã hóa một thông điệp cho trước bằng khóa bí mật 11](#_Toc152673959)

[3.6 Hàm giải mã thông điệp bằng khóa công khai. 11](#_Toc152673960)

[3.7 Sử dụng lớp RSACryp để mã hóa 1 chuỗi và băm chuỗi theo 1 thuật toán băm cho trước 12](#_Toc152673961)

[3.8 Đọc file chứa bản mã và bản băm của một thông điệp, sử dụng lớp RSACryp. Sau đó đem so sánh hai bản băm để xác nhận tính toàn vẹn 12](#_Toc152673962)

[4. Lập trình mô tả lỗi SQL Injection và cách khắc phục 13](#_Toc152673963)

## 1. Sử dụng Java Cryptography Architecture (JCA), viết class SymmetricCryp để thực hiện các chức năng mã hóa và giải mã đối xứng theo thuật toán AES gồm các hàm

**public** **class** SymmetricCrypt {

**private** **final** String Algorithm = "AES";

//khóa

**private** SecretKey secretKey;

//khóa theo chuỗi cho trước

**private** SecretKeySpec skeySpec;

//bản sinh khóa theo thuật toán đã chọn

**private** KeyGenerator keyGen;

//bản mã

**private** Cipher cipher;

## Sinh khóa đối xứng

**public** SymmetricCrypt()

**throws** NoSuchAlgorithmException {

keyGen = KeyGenerator.getInstance(Algorithm);

secretKey = keyGen.generateKey();

}

## Tạo khóa đối xứng từ một chuỗi cho trước

**public** SymmetricCrypt(String key)

**throws** NoSuchAlgorithmException {

skeySpec = **new** SecretKeySpec(key.getBytes(), "AES");

}

**public** SecretKey getSecretKey() {

**return** secretKey;

}

**public** SecretKey getSecretKeySpec() {

**return** skeySpec;

}

## Mã hóa thông điệp có 2 tham số: Tham số 1 là chuỗi cần mã hóa, tham số thứ 2 là khóa được tạo ở 1.1

**public** String encryptText(String msg, SecretKey key)

**throws** NoSuchAlgorithmException,

NoSuchPaddingException,

InvalidKeyException,

UnsupportedEncodingException,

IllegalBlockSizeException,

BadPaddingException

{

cipher = Cipher.*getInstance*(Algorithm);

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

**return** Base64.*getEncoder*().encodeToString(

cipher.doFinal(msg.getBytes("UTF-8")));

}

## Mã hóa thông điệp có 2 tham số: Tham số 1 là chuỗi cần mã hóa tham số 2 là khóa được tạo ở 1.2

**public** String encryptText(String msg, SecretKeySpec key)

**throws** NoSuchAlgorithmException,

NoSuchPaddingException,

InvalidKeyException,

UnsupportedEncodingException,

IllegalBlockSizeException,

BadPaddingException

{

cipher = Cipher.*getInstance*(Algorithm);

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

**return** Base64.*getEncoder*().encodeToString(

cipher.doFinal(msg.getBytes("UTF-8")));

}

## Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.1

**public** String decryptText(String msg, SecretKey key)

**throws** NoSuchAlgorithmException,

NoSuchPaddingException,

InvalidKeyException,

IllegalBlockSizeException,

BadPaddingException,

UnsupportedEncodingException

{

cipher = Cipher.*getInstance*(Algorithm);

cipher.init(Cipher.***DECRYPT\_MODE***, key);

**return** **new** String(cipher.doFinal(

Base64.*getDecoder*().decode(msg)),"UTF-8");

}

## Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.2

**public** String decryptText(String msg, SecretKeySpec key)

**throws** NoSuchAlgorithmException,

NoSuchPaddingException,

InvalidKeyException,

IllegalBlockSizeException,

BadPaddingException,

UnsupportedEncodingException

{

cipher = Cipher.*getInstance*(Algorithm);

cipher.init(Cipher.***DECRYPT\_MODE***, key);

**return** **new** String(cipher.doFinal(

Base64.*getDecoder*().decode(msg)),"UTF-8");

}

}

## Sử dụng class SymetricCryp để mã hóa một chuỗi cho trước hoặc giải mã ra bản rõ từ bản mã cho trước.

**public** **class** SymmetricApp {

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

**throws** Exception {

SymmetricCrypt SC = **new** SymmetricCrypt();

String msg = "Nguyễn Văn Hiền";

String encrypted\_msg =

SC.encryptText(msg, SC.getSecretKey());

System.***out***.println("Plain text: " + msg);

System.***out***.println("Encrypted text: " + encrypted\_msg);

String decrypted\_msg =

SC.decryptText(encrypted\_msg, SC.getSecretKey());

System.***out***.println("Decrypted text: " + decrypted\_msg);

SymmetricCrypt SC2 = **new** SymmetricCrypt("stackjava.com.if");

String msg2 = "Nguyễn Văn Hiền";

String encrypted\_msg2 =

SC.encryptText(msg2, SC2.getSecretKeySpec());

System.***out***.println("Plain text: " + msg2);

System.***out***.println("Encrypted text: " + encrypted\_msg2);

String decrypted\_msg2 =

SC.decryptText(encrypted\_msg2, SC2.getSecretKeySpec());

System.***out***.println("Decrypted text: " + decrypted\_msg2);

}

}

## 2. Sử dụng Java Cryptography Architecture (JCA) để viết các hàm băm thông điệp cho trước theo thuật toán:

public class Digest {

## 2.1. MD5

public String md5Digest(String msg){

try{

MessageDigest md = MessageDigest.getInstance("MD5");

md.update(msg.getBytes("UTF-8"),0,msg.length());

return DatatypeConverter.printHexBinary(md.digest());

} catch (Exception ex){

ex.printStackTrace();

}

return null;

}

## 2.2. SHA1

public String sha1Digest(String msg){

try {

MessageDigest md = MessageDigest.getInstance("SHA-1");

md.update(msg.getBytes("UTF-8"),0, msg.length());

return DatatypeConverter.printHexBinary(md.digest());

} catch (Exception ex){

ex.printStackTrace();

}

return null;

}

## 2.3. SHA-256

public String getHashSHA256(String origin) throws Exception {

String sha256;

MessageDigest m = MessageDigest.getInstance("SHA-256");

m.update(origin.getBytes("UTF-8"), 0, origin.length());

sha256 = new BigInteger(1, m.digest()).toString(16);

return sha256;

}

}

public static void main(String[] args){

Digest digest = new Digest();

String msg ="Hello Word";

String md5Hash = digest.md5Digest(msg);

System.out.println("MD5 Hash: " + md5Hash);

String sha1Hash = digest.sha1Digest(msg);

System.out.println("SHA-1 Hash: "+sha1Hash);

String sha256Hash = digest. getHashSHA256 (msg);

System.out.println("SHA-256 Hash: "+sha256Hash);

}

## 3. Sử dụng Java Crypto Architecture (JCA), viết chương trình mã hóa và giải mã sử dụng mã hóa công khai RSA:

**Viết class RSAKeyGen để sinh bộ khóa công khai gồm các hàm**

public class RSAKeyGen {

private KeyPairGenerator keyGen;

private KeyPair keypair;

private PublicKey publicKey;

private PrivateKey privateKey;

public PublicKey getPublicKey() { return publicKey; }

public void setPublicKey(PublicKey publicKey) { this.publicKey = publicKey; }

public PrivateKey getPrivateKey() { return privateKey; }

public void setPrivateKey(PrivateKey privateKey) {this.privateKey = privateKey;}

## 3.1 Sinh bộ khóa theo thuật toán RSA

public RSAKeyGen(int keyLength){

try{

keyGen = KeyPairGenerator.getInstance("RSA");

keyGen.initialize(keyLength);

keypair = keyGen.generateKeyPair();

setPublicKey(keypair.getPublic());

setPrivateKey(keypair.getPrivate());

}catch (Exception ex){

ex.printStackTrace();

}

}

## 3.2 Lưu bộ khóa ra file

public void writeToFile(String path, byte[] key){

try{

File f = new File(path);

f.getParentFile().mkdir();

FileOutputStream fos = new FileOutputStream(f);

fos.write(key);

fos.flush();

fos.close();

}catch (Exception ex){

ex.printStackTrace();

}

}

}

**\*Viết class RSACryp để sd bộ khóa đã lưu trong file để mã hóa và giải mã theo RSA**

public class RSACryp {

private Cipher cipher;

## 3.3 Hàm đọc file để nạp khóa bí mật, tham số truyền vào là tên file

public PrivateKey getPrivateKey(String filePath){

try{ byte keyByte[] = Files.readAllBytes(new File(filePath).toPath());

PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(keyByte);

KeyFactory kf = KeyFactory.getInstance("RSA");

return kf.generatePrivate(spec);

}catch (Exception ex){ ex.printStackTrace(); }

return null;

}

## 3.4 Hàm đọc file để nạp khóa công khai, tham số truyền vào là tên file

public PublicKey getPublicKey(String filePath){

try{

byte keyByte[] = Files.readAllBytes(new File(filePath).toPath());

X509EncodedKeySpec spec = new X509EncodedKeySpec(keyByte);

KeyFactory kf = KeyFactory.getInstance("RSA");

return kf.generatePublic(spec);

}catch (Exception ex){

ex.printStackTrace();

}

return null;

}

## 3.5 Hàm mã hóa một thông điệp cho trước bằng khóa bí mật, tham số truyền vào gồm tham số thứ nhất là thông điệp cần mã hóa, tham số thứ 2 là khóa bí mật

public String encrypt(String msg, PrivateKey key){

try{

cipher = Cipher.getInstance("RSA");

cipher.init(Cipher.ENCRYPT\_MODE, key);

return Base64.getEncoder().encodeToString(cipher.doFinal(msg.getBytes("UTF-8")));

}catch (Exception ex){

ex.printStackTrace();

}

return null;

}

## 3.6 Hàm giải mã thông điệp bằng khóa công khai, tham số truyền vào gồm 1 là bản mã cần giải mã, 2 là khóa công khai.

public String decrypt(String msg, PublicKey publicKey){

try{

cipher = Cipher.getInstance("RSA");

cipher.init(Cipher.DECRYPT\_MODE, publicKey);

return new String(cipher.doFinal(Base64.getDecoder().decode(msg)), "UTF-8");

}catch (Exception ex){

ex.printStackTrace();

}

return null;

}

## 3.7 Sử dụng lớp RSACryp để mã hóa 1 chuỗi và băm chuỗi theo 1 thuật toán băm cho trước rồi lưu cả bản mã và bản băm vào 1 file

public static void main(String[] args) {

RSACryp rsaCryp = new RSACryp();

try {

MessageDigest digest = MessageDigest.getInstance("SHA-1");

RSAKeyGen rsaKeyGen = new RSAKeyGen(2048);

rsaKeyGen.writeToFile("E:\\publicKey.txt", rsaKeyGen.getPublicKey().getEncoded());

rsaKeyGen.writeToFile("E:\\privateKey.txt", rsaKeyGen.getPrivateKey().getEncoded());

String msg = "Pham Nhat Nam";

String encrypted\_msg = rsaCryp.encrypt(msg, rsaCryp.getPrivateKey("E:/privateKey.txt"));

byte[] firstHash = digest.digest(msg.getBytes("UTF-8"));

String hashCiper = encrypted\_msg + ";" + Base64.getEncoder().encodeToString(firstHash);

rsaCryp.writeToFile("E:/result.txt", hashCiper);

## 3.8 Đọc file chứa bản mã và bản băm của một thông điệp, sử dụng lớp RSACryp để giải mã ra thông điệp gốc và băm lại bằng thuật toán băm cho trước. Sau đó đem so sánh hai bản băm để xác nhận tính toàn vẹn

String dataRead = rsaCryp.readFile("E:\\result.txt");

String result[] = dataRead.split(";");

String decrypted\_msg = rsaCryp.decrypt(result[0], rsaCryp.getPublicKey("E:/publicKey.txt"));

byte[] hashAgain = digest.digest(decrypted\_msg.getBytes("UTF-8"));

String hashAgainString = Base64.getEncoder().encodeToString(hashAgain);

System.out.println("Plain Text:" + msg);

System.out.println("Encrypt Text: " + encrypted\_msg);

System.out.println("MD5 hash: " + new String(firstHash));

System.out.println("MD5 hash: " + result[1]);

System.out.println("Decrypt Text: " + decrypted\_msg);

System.out.println("Hash Again: " + hashAgainString);

if(hashAgainString.equals(result[1])){

System.out.println("Same");

} else {

System.out.println("Not same");

}

} catch (NoSuchAlgorithmException | UnsupportedEncodingException e) {

e.printStackTrace();

}

}

## 4. Lập trình mô tả lỗi SQL Injection và cách khắc phục

**Chương trình SSH App-script : (Base System 1: ch11, Base System 2: ch12)**

**Base System 1: ch11**

Bước 1: Vì trong thư mục hiện tại ta không có quyền tạo thư mục và file nên ta sẽ tạo một thư mục của riêng ta ở trong /tmp bằng lệnh: mkdir /tmp/lagi

Bước 2: Copy chương trình cat của hệ thống nằm trong /bin vào trong thư mục vừa tạo với tên là ls bằng lệnh

cp   /bin/cat   /tmp/lagi/ls

Bước 3: Để hệ thống có thể tìm kiếm đến ls trong /tmp/lagi ta thêm đường dẫn tới /tmp/lagi vào biến môi trường $PATH bằng lệnh

export   PATH=/tmp/lagi:$PATH

Bước 4: Chạy chương trình ch11 để hiển thị mật khẩu bằng lệnh

./ch11

**Base System 2:ch12**

**B1** : Tạo thư mục tên là “Tên thư mục” trong thư mục /tmp bằng lệnh :

mkdir /tmp/Tên thư mục

**B2** : Tạo một file shell-script với tên là ls trong thư mục vừa tạo bằng trình soạn thảo văn bản nano :

nano /tmp/Tên thư mục/ls

**B3** : Sau khi trình soạn thảo nano xuất hiện ta soạn thảo file shell với hai lệnh:

* lệnh thứ nhất khai báo /bin/sh
* lệnh thứ hai là chạy trình đọc file /bin/cat với tham số là tham số thứ hai trên dòng lệnh .

*Sau đây là nội dung file ls cần soạn thảo :*

#!/bin/sh

/bin/cat "/challenge/app-script/ch12/.passwd"

**B4** : Lưu nội dung file ls bằng phím nóng Ctrl – O rồi thoát khỏi nano bằng phím nóng Ctrl – X

**B5**: Thêm quyền thực thi cho file shell tên là ls vừa soạn thảo bằng lệnh

chmod +x /tmp/Tên thư mục/ls

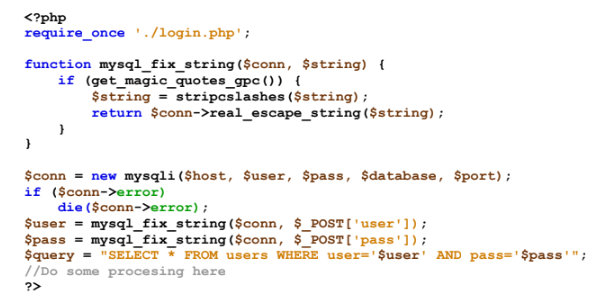
**B6**: Để hệ thống có thể tìm kiếm đến ls trong /tmp/Tên thư mục ta thêm đường dẫn tới /tmp/Tên thư mục vào biến môi trường $PATH bằng lệnh

export PATH=/tmp/Tên thư mục:$PATH

**B7**: Chạy chương trình **ch12** để hiển thị mật khẩu bằng lệnh

**./ch12**

**SQL Injection** --- Cách 1: Vô hiệu hóa các ký tự đặc biệt: loại bỏ các ký tự đặc biệt như dấu ngoặc kép, dấu nháy bằng đặt trước chúng dấu gạch chéo ngược. Hoặc sử dụng phương thức get\_magic\_quotes\_gpc() hoặc sử dụng phương thức real\_escape\_string() khi gọi đến MySQL.



--- Cách 2: Tham số hóa cho câu truy vấn: Không cho phép dữ liệu do người dùng gửi được hiểu là các câu lệnh MySQL. Câu lệnh truy vấn có thể sử dụng tham số ‘?’ như 1 từ thay thế cho dữ liệu.

<?php

$con = mysqli\_connect(“localhost”,”root”,””,”sql\_demo”);

If(isset($\_POST[‘username’]) && isset($\_POST[‘password])){

$user = $\_POST[‘username’];

$pass = $\_POST[‘password];

$query = “Select \* from users Where username =? And password = ?”;

$result = $con->prepare($query);

$result->bind\_param(‘ss’,$user,$pass);

$result->execute();

If($row = mysqli\_fetch\_assoc($result)){ Echo “Login success”;}

Else{ echo “Login failed”; }

}

?>