|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No. 08** | **Secure Electronic Transaction** | | |
| Date of Exercise | 17-3-2015 | Date of Output Verification | 17-3-2015 |

**Question**

Implement SET in java Exchanging the information among Client, Merchant and the Bank.

**Procedure**

* The customer opens a Mastercard or Visa bank account. Any issuer of a credit card is some kind of bank.
* The customer receives a digital certificate. This electronic file functions as a credit card for online purchases or other transactions. It includes a public key with an expiration date. It has been through a digital switch to the bank to ensure its validity.
* Third-party merchants also receive certificates from the bank. These certificates include the merchant's public key and the bank's public key.
* The customer places an order over a Web page, by phone, or some other means.
* The customer's browser receives and confirms from the merchant's certificate that the merchant is valid.
* The browser sends the order information. This message is encrypted with the merchant's public key, the payment information, which is encrypted with the bank's public key (which can't be read by the merchant), and information that ensures the payment can only be used with this particular order.
* The merchant verifies the customer by checking the digital signature on the customer's certificate. This may be done by referring the certificate to the bank or to a third-party verifier.
* The merchant sends the order message along to the bank. This includes the bank's public key, the customer's payment information (which the merchant can't decode), and the merchant's certificate.
* The bank verifies the merchant and the message. The bank uses the digital signature on the certificate with the message and verifies the payment part of the message.
* The bank digitally signs and sends authorization to the merchant, who can then fill the order.

**Program**

**Client**

import java.io.PrintWriter;

import java.math.BigInteger;

import java.net.Socket;

import java.security.MessageDigest;

import java.util.Random;

import java.util.Scanner;

// @author William Scott

public class NetworkSecurityLabExp8SETClient {

//UR12CS135 - P.William Scott - Exp 8 - SET (Client)

private static Scanner in = new Scanner(System.in);

private static String oi, pi, oimd, pimd, pomd;

private static BigInteger message, ds;

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

System.out.println("UR12CS135 - SET(Client)");

RSA r = new RSA(1024);

System.out.println("Enter Payment Information");

pi = in.nextLine();

System.out.println("Enter Order Information");

oi = in.nextLine();

pimd = sha512(pi);

oimd = sha512(oi);

pomd = sha512(pimd.concat(oimd));

message = new BigInteger(pomd, 16);

ds = r.encrypt(message);

System.out.println("\npimd :" + pimd);

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

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

System.out.println("Message :" + message);

System.out.println("Dual Signature :" + ds);

Socket smerchant=new Socket("localhost",3000);

Socket sbank=new Socket("localhost",4000);

PrintWriter pwmerchant=new PrintWriter(smerchant.getOutputStream(),true);

PrintWriter pwclient=new PrintWriter(sbank.getOutputStream(),true);

pwmerchant.println(ds);

pwmerchant.println(oi);

pwmerchant.println(pimd);

pwmerchant.println(r.getE());

pwmerchant.println(r.getN());

pwclient.println(ds);

pwclient.println(pi);

pwclient.println(oimd);

pwclient.println(r.getE());

pwclient.println(r.getN());

pwmerchant.close();

pwclient.close();

smerchant.close();

sbank.close();

}

private static String sha512(String a) throws Exception {

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

sha.update(a.getBytes());

return new BigInteger(1, sha.digest()).toString(16);

}

}

class RSA {

private BigInteger p, q, n, pin, d, e = new BigInteger("5"), m, c;

private int y;

private Random r = new Random();

RSA(int i) {

this.p = BigInteger.probablePrime(i, this.r);

this.q = BigInteger.probablePrime(i, this.r);

this.n = this.p.multiply(this.q);

this.pin = this.p.subtract(BigInteger.ONE).multiply(this.q.subtract(BigInteger.ONE));

while (this.pin.gcd(this.e).doubleValue() > 1) {

this.e = this.e.add(new BigInteger("2"));

}

this.d = this.e.modInverse(this.pin);

}

public BigInteger getE() {

return this.e;

}

public BigInteger getN() {

return this.n;

}

public BigInteger encrypt(BigInteger m) {

return m.modPow(this.d, this.n);

}

public BigInteger decrypt(BigInteger c) {

return c.modPow(this.e, this.n);

}

}

**Merchant:**

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.math.BigInteger;

import java.net.ServerSocket;

import java.net.Socket;

import java.security.MessageDigest;

// @author William Scott

public class NetworkSecurityLabExp8SETMerchant {

//UR12CS135 - P.William Scott - Exp 8 - SET (Merchant)

public static BigInteger ds, e, n, a, b;

public static String pimd, st, oi;

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

System.out.println("UR12CS135 - SET(Merchant)");

ServerSocket ss = new ServerSocket(3000);

Socket s = ss.accept();

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

ds = new BigInteger(br.readLine());

oi = br.readLine();

pimd = br.readLine();

e = new BigInteger(br.readLine());

n = new BigInteger(br.readLine());

a = ds.modPow(e, n);

st = sha512(pimd.concat(sha512(oi.toString())));

b = new BigInteger(st, 16);

System.out.println("\n-------Received values------");

System.out.println("Dual Signature :" + ds);

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

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

System.out.println("Client's Public Key :" + e);

System.out.println("\n------Calculated Values-------");

System.out.println("H(pimd||H(oi)) :" + b);

System.out.println("D(k,ds) :" + a);

if (a.toString().compareTo(b.toString()) == 0) {

System.out.println("Client's signature is verified");

}

br.close();

ss.close();

s.close();

}

private static String sha512(String a) throws Exception {

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

sha.update(a.getBytes());

return new BigInteger(1, sha.digest()).toString(16);

}

}

**Bank**

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.math.BigInteger;

import java.net.ServerSocket;

import java.net.Socket;

import java.security.MessageDigest;

// @author William Scott

public class NetworkSecurityLabExp8SETBank {

//UR12CS135 - P.William Scott - Exp 8 - SET (Bank)

public static BigInteger ds, e, n, a, b;

public static String pi, oimd, st;

public static BufferedReader br;

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

System.out.println("UR12CS135 - SET(Bank)");

ServerSocket ss = new ServerSocket(4000);

Socket s = ss.accept();

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

ds = new BigInteger(br.readLine());

pi = br.readLine();

oimd = br.readLine();

e = new BigInteger(br.readLine());

n = new BigInteger(br.readLine());

a = ds.modPow(e, n);

st = sha512(sha512(pi.toString()).concat(oimd));

b = new BigInteger(st, 16);

System.out.println("\n-----Received Values------");

System.out.println("Dual Signature :" + ds);

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

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

System.out.println("Client's Public Key :" + e);

System.out.println("\n------Calculated Values-------");

System.out.println("H(H(pi)||oimd) :" + b);

System.out.println("D(k,ds) :" + a);

if (a.toString().compareTo(b.toString()) == 0) {

System.out.println("Client's signature is verified");

}

br.close();

ss.close();

s.close();

}

private static String sha512(String a) throws Exception {

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

sha.update(a.getBytes());

return new BigInteger(1, sha.digest()).toString(16);

}

public static BigInteger decrypt(BigInteger c, BigInteger e, BigInteger n) {

return c.modPow(e, n);

}

}

**Input**

OI: Bag-2000

PI: 4591577516949468-5846-296556-22

**Output**

**Client**

run:

UR12CS135 - SET(Client)

Enter Payment Information

4591577516949468-5846-296556-22

Enter Order Information

Bag-2000

pimd :6becb39540f3bf75bf274a6edfd83c1b8e44cee8bf332db5652c4108eae7f0ce1042b69409d1317d7c73289c4739e05d387f9d72f03f08b445a7322919174fd9

oimd :ab6a779f4b821236673f055f828d4f3035a6a84069f4c165a6a3ca0eae925e0ca1d4e1c42aaa962a4a1604fc7919e5c7ee0d6bb6f4a2081bf3c2f8aa0a53ab32

pomd :cf1255c06b1db19816e528904f5d1972478584e82eb880d3e5023ed33b065fccc4b409f9d44165fd02e386d37977ab97cf80cd4403eec71c694a412159a4cea0

Message :10845220787716289343175993052107627206449095162313154496898515223549887645869188303328945125648323000173395326632196001284334641808203055308336407800762016

Dual Signature :14124137169831095542149171266323335031933963083300577484505920942056902710268815978182518683994145932357221575380940273120822530128140126968606896626406809471292114952441343901493016605769163834805801329980236843104526635330536474852594500585503223445151281286287164198354453084034428689216186204087024496310264936396788667643834487320630604837311977314050391963527980951556760676725009863065583068298272170327780588660256253374653817031954625431180595791122577430256257435281319582049822526738191844598601692702733016956613270107186033419892929298477129245246860527309773570333982306164596573904050093859492226766090

BUILD SUCCESSFUL (total time: 25 seconds)

**Merchant**

run:

UR12CS135 - SET(Merchant)

-------Received values------

Dual Signature :14124137169831095542149171266323335031933963083300577484505920942056902710268815978182518683994145932357221575380940273120822530128140126968606896626406809471292114952441343901493016605769163834805801329980236843104526635330536474852594500585503223445151281286287164198354453084034428689216186204087024496310264936396788667643834487320630604837311977314050391963527980951556760676725009863065583068298272170327780588660256253374653817031954625431180595791122577430256257435281319582049822526738191844598601692702733016956613270107186033419892929298477129245246860527309773570333982306164596573904050093859492226766090

oi :Bag-2000

pimd :6becb39540f3bf75bf274a6edfd83c1b8e44cee8bf332db5652c4108eae7f0ce1042b69409d1317d7c73289c4739e05d387f9d72f03f08b445a7322919174fd9

Client's Public Key :5

------Calculated Values-------

H(pimd||H(oi)) :10845220787716289343175993052107627206449095162313154496898515223549887645869188303328945125648323000173395326632196001284334641808203055308336407800762016

D(k,ds) :10845220787716289343175993052107627206449095162313154496898515223549887645869188303328945125648323000173395326632196001284334641808203055308336407800762016

Client's signature is verified

BUILD SUCCESSFUL (total time: 23 seconds)

**Bank**

run:

UR12CS135 - SET(Bank)

-----Received Values------

Dual Signature :14124137169831095542149171266323335031933963083300577484505920942056902710268815978182518683994145932357221575380940273120822530128140126968606896626406809471292114952441343901493016605769163834805801329980236843104526635330536474852594500585503223445151281286287164198354453084034428689216186204087024496310264936396788667643834487320630604837311977314050391963527980951556760676725009863065583068298272170327780588660256253374653817031954625431180595791122577430256257435281319582049822526738191844598601692702733016956613270107186033419892929298477129245246860527309773570333982306164596573904050093859492226766090

pi :4591577516949468-5846-296556-22

oimd :ab6a779f4b821236673f055f828d4f3035a6a84069f4c165a6a3ca0eae925e0ca1d4e1c42aaa962a4a1604fc7919e5c7ee0d6bb6f4a2081bf3c2f8aa0a53ab32

Client's Public Key :5

------Calculated Values-------

H(H(pi)||oimd) :10845220787716289343175993052107627206449095162313154496898515223549887645869188303328945125648323000173395326632196001284334641808203055308336407800762016

D(k,ds) :10845220787716289343175993052107627206449095162313154496898515223549887645869188303328945125648323000173395326632196001284334641808203055308336407800762016

Client's signature is verified

BUILD SUCCESSFUL (total time: 18 seconds)

**Result**

The implementation of SET is successfully done.

[Signature of the Staff In-charge]

Name of the Staff In – charge: Mr. Manoj Kumar

Date: 17-3-2015