SEM PRACTICAL EXAMINATION

NAME: Ashwin.S

Date: 17/12/20

Reg_NO: 312217104019

DSS Algorithm

AIM:

To Develop a java program to Implement the Digital Signature Standard.

PROCEDURE:/ Algorithm:

(i) First a large prime number is choosen, Let it be P.

Now P is choosen in such a way that the no. of bits representing it (i.e., the length of P) must be a multiple of 64 and should be between 512 and 1024.

(ii) Then, a number 'q' is selected which is a 160-bit prime factor of p-1., here p-1 is even.

(iii) Then a number g is selected such that

$g = h^{((p-1)/q)} \bmod p$

(iv) Now a random integer x is choosen for user's private key where $0 < x < q$

(v) From this the user's public key is calculated using the formula $y = g^x \bmod p$

(vi) Then a random integer k is selected where $0 < k < q$, but once it is used it won't be used again, a new k value will be used.

(vii) Then comes the signing of the message

A function which is called the signing function takes in the input as :

- the Private key
- the Public key
- the message
- the value (K)

and produces the signature 'r' and 's'

(viii) Now,

$$r = (g^k \bmod p) \bmod q$$

$$s = [k^{-1} \cdot (H(M) + x r)] \bmod q$$

Here H(m) is the hash of the message

(ix) Now when the reciever recieves the message, it will have the signature (r' and s') along with the message

(x) The verification process is as follows :

$$w = (s')^{-1} \bmod q$$

$$a = [H(m') * w] \bmod q$$

$$b = (r')^{w} \bmod q$$

$$v = [(g^{a} y^{b}) \bmod p] \bmod q$$

(xi) After calculating the value of v, it is compared with the recieved r value which is represented as v',

(xii) If they match then we have recieved the correct message, if not then the message recieved is wrong.

Digital Signature Standard(DSS):

Name: Ashwin S **Date: 17/12/2020**

Reg. NO.: 312217104019

CODE:

```
import java.util.*;
import java.io.UnsupportedEncodingException;
import java.security.*;
import java.security.interfaces.*;

public class Main{
  public static void main(String[] args) throws
NoSuchAlgorithmException,InvalidKeyException,SignatureException,Un
supportedEncodingException{
    Scanner sc = new Scanner(System.in);
    System.out.println("Text: ");
    String msg = sc.nextLine();

    KeyPairGenerator keys = KeyPairGenerator.getInstance("DSA");
    keys.initialize(1024);
```

```
    KeyPair key = keys.generateKeyPair();
    DSAPrivateKey pk = (DSAPrivateKey)key.getPrivate();
    DSAPublicKey puk = (DSAPublicKey)key.getPublic();
    System.out.println();
    System.out.println("Private Key: \n"+ pk.getX());
    System.out.println();
    System.out.println("Public key: \n"+ puk.getY());
    System.out.println();

    Signature sign = Signature.getInstance("SHA256withDSA");

    sign.initSign(pk);

    byte[] text = msg.getBytes();
    sign.update(text);

    byte[] signature = sign.sign();
    sign.initVerify(puk);
    System.out.println();
    System.out.println("Digital signature for given text: "+
bytesToHex(signature));
    System.out.println();
```

```
   System.out.println("Data:");
   String data = sc.nextLine();
   System.out.println();
   sign.update(data.getBytes());



   if(sign.verify(signature))
   {
     System.out.println("Signature Verified");
   }
   else{
     System.out.println("Signature verification failed");
   }
  }

private static final char[] HEX_ARRAY = "0123456789ABCDEF". toCharArray();
public static String bytesToHex(byte[] bytes) {
char[] hexChars = new char[bytes.length * 2];
for (int j = 0; j < bytes.length; j++) {
int v = bytes[j] & 0xFF;
```

```
hexChars[j * 2] = HEX_ARRAY[v >>> 4];
hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
}
return new String(hexChars);
}
}
```

OUTPUT:

```
javac -classpath .:/run_dir/junit-4.12.jar:target/dependency/* -d . Main.java
java -classpath .:/run_dir/junit-4.12.jar:target/dependency/* Main
javac -classpath .:/run_dir/junit-4.12.jar:target/dependency/* -d . Main.
javac -classpath .:/run_dir/junit-4.12.jar:target/dependency/* -d . Main.java
java -classpath .:/run_dir/junit-4.12.jar:target/dependency/* Main
Text:
hello

Private Key:
491121509683689564990497485931394755700615870509

Public key:
14894739587238615709718997073192865031995758914441895819287014953774813704618822153644760891000269546471452384316521589897087422429140258552909145721868629388523029919659008685641988057018564601811250244825413413461951541448672302588757707518880789901506159887080403092551411775553802754576854860300283622539 4


Digital signature for given text: 302C021468F265C8CA210F834A70A0BE3851061B2DE61AC502143EB571FD4B238512154670B2973F5DB121CDBBAE

Data:
hello

Signature Verified
```

RESULT:

Hence a java program has been developed to implement the Digital Signature Standard.