CHANDIGARH UNIVERSITY

UNIVERSITY INSTITUTE OF NGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Submitted By: Vivek Kumar(21BCS8129)	**Submitted To:** Er. Himanshi (13362)
Subject Name	Web and Mobile Security Lab
Subject Code	20CSP-338
Branch	Computer Science and Engineering
Semester	5th

Experiment - 8

Student Name: Vivek Kumar **UID: 21BCS8129**
Branch: BE-CSE(LEET) **Section/Group: WM-20BCS-616/A**
Semester: 5th **Date of Performance: 02/11/2022**
Subject Name: Web and Mobile Security Lab **Subject Code: 20CSP-338**

1. Aim/Overview of the practical:

Write a program to sign and verify a document using DSA algorithm.

2. Task to be done/ Which logistics used:

To generate the concept of digital signature

3. Apparatus / Simulator Used:

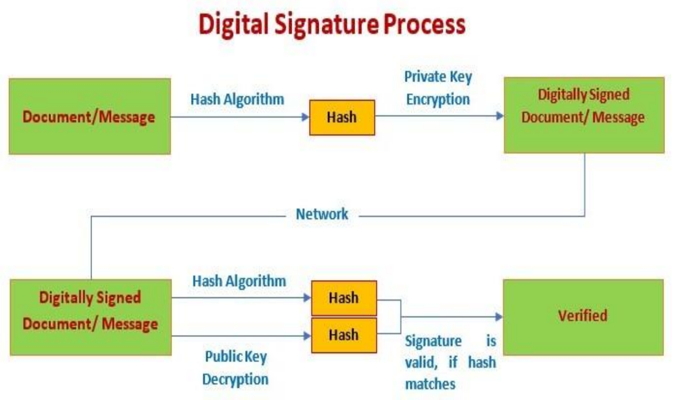
- Windows 7 & above version.
- Google Chrome C/C++
- Java
- Python platform

Discussion:

The digital signature is a mechanism that verifies the authority of digital messages as well as documents. It is very popular because it provides more security than other signatures. In Java, JDK Security API is used to create and implement digital signatures. In this section, we will discuss the digital signature mechanism and also implement the digital signature mechanism in a Java program.

The digital signature is an electronic signature to sign a document, mail, messages, etc. It validates the authenticity, and integrity of a message or document. It is the same as a handwritten signature, seal, or stamp. It is widely used to verify a digital message, financial documents, identity cards, etc.

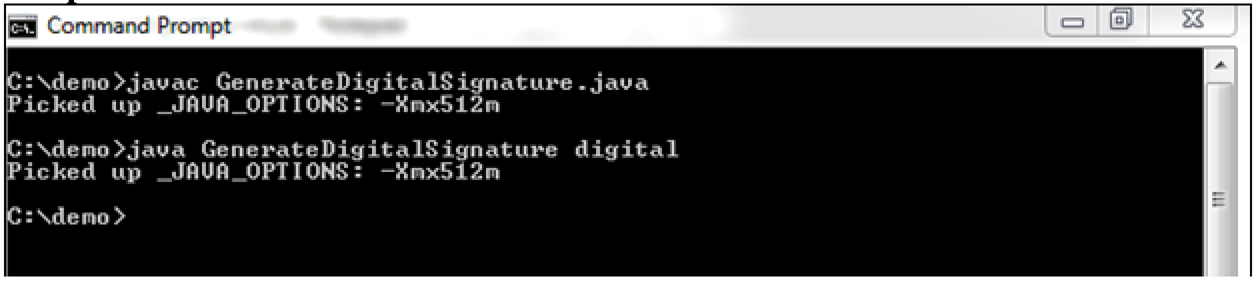
In short, we can say that it en ures the following:
o Integrity: It ensures the message or a document cannot be altered while transmitting.
o Authenticity: The author of the message is really who they claim to be.
o Non-repudiation: The author of the message can't later deny source.


Digital Signature Process
Document/Message
Hash Algorithm
Hash
Private Key Encryption
Digitally Signed Document/ Message
Network
Digitally Signed Document/ Message
Hash Algorithm
Hash
Hash
Public Key Decryption
Signature is valid, if hash matches
Verified

4. Program/ Steps/ Method:

```
import java.io.*;
import java.security.*;

public class GenerateDigitalSignature {
      public static void main(String args[]) {
            /* Generate a DSA signature */
            if (args.length != 1) {
                  System.out.println("Usage: nameOfFileToSign");
            } else
                  try {
                        /* Generate a key pair */
                        KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA",
"SUN");
                        SecureRandom random = SecureRandom.getInstance("SHA1PRNG",
"SUN");
                        keyGen.initialize(1024, random);
                        KeyPair pair = keyGen.generateKeyPair();
                        PrivateKey priv = pair.getPrivate();
                        PublicKey pub = pair.getPublic();
                        /* Create a Signature object and initialize it with the private
key */
                        Signature dsa = Signature.getInstance("SHA1withDSA", "SUN");
                        dsa.initSign(priv);
                        /* Update and sign the data */
                        FileInputStream fis = new
FileInputStream("R:\\VnjVibhash\\Assignments\\CU-Assignments\\5th
Sem\\Java\\JavaLab\\src\\Input.txt");
                        BufferedInputStream bufin = new BufferedInputStream(fis);
                        byte[] buffer = new byte[1024];
                        int len;
                        while (bufin.available() != 0) {
                              len = bufin.read(buffer);
                              dsa.update(buffer, 0, len);
                        }
                        ;
                        bufin.close();
                        /*
                         * Now that all the data to be signed has been read in, generate
a signature for
                         * it
                         */
                        byte[] realSig = dsa.sign();
                        /* Save the signature in a file */
                        FileOutputStream sigfos = new FileOutputStream("F:\\Digital
Signature Demo\\signature.txt");
                        sigfos.write(realSig);
                        sigfos.close();
                        /* Save the public key in a file */
                        byte[] key = pub.getEncoded();
                        FileOutputStream keyfos = new FileOutputStream("F:\\Digital
Signature Demo\\publickey.txt");
                        keyfos.write(key);
                        keyfos.close();
                  } catch (Exception e) {
                        System.err.println("Caught exception " + e.toString());
                  }
            };
}
```

Output:

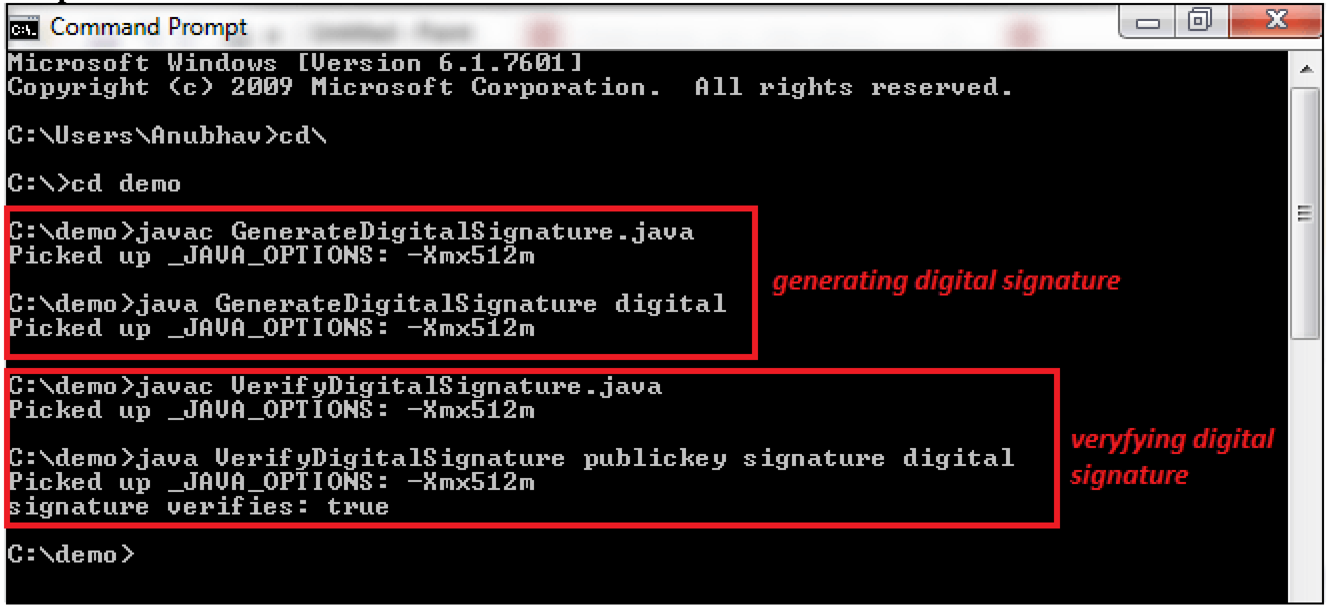


```
C:\demo>javac GenerateDigitalSignature.java
Picked up _JAVA_OPTIONS: -Xmx512m

C:\demo>java GenerateDigitalSignature digital
Picked up _JAVA_OPTIONS: -Xmx512m

C:\demo>
```

5. DBMS Script/Result/Output/Writing Summary:

```
import java.io.*;
import java.security.*;
import java.security.spec.*;

public class VerifyDigitalSignature {
      public static void main(String args[]) {
            /* Verify a DSA signature */
            if (args.length != 3) {
                  System.out.println("Usage: publickeyfile signaturefile datafile");
            } else
                  try {
                        /* import encoded public key */
                        FileInputStream keyfis = new FileInputStream("F:\\Digital
Signature Demo\\publickey.txt");
                        byte[] encKey = new byte[keyfis.available()];
                        keyfis.read(encKey);
                        keyfis.close();
                        X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec(encKey);
                        KeyFactory keyFactory = KeyFactory.getInstance("DSA", "SUN");
                        PublicKey pubKey = keyFactory.generatePublic(pubKeySpec);
                        /* input the signature bytes */
                        FileInputStream sigfis = new FileInputStream("F:\\Digital
Signature Demo\\signature.txt");
                        byte[] sigToVerify = new byte[sigfis.available()];
                        sigfis.read(sigToVerify);
                        sigfis.close();
                        /* create a Signature object and initialize it with the public
key */
                        Signature sig = Signature.getInstance("SHA1withDSA", "SUN");
                        sig.initVerify(pubKey);
                        /* Update and verify the data */
                        FileInputStream datafis = new FileInputStream("F:\\Digital
Signature Demo\\digital.txt");
                        BufferedInputStream bufin = new BufferedInputStream(datafis);
                        byte[] buffer = new byte[1024];
                        int len;
                        while (bufin.available() != 0) {
                              len = bufin.read(buffer);
                              sig.update(buffer, 0, len);
                        }
                        ;
                        bufin.close();
                        boolean verifies = sig.verify(sigToVerify);
                        System.out.println("signature verifies: " + verifies);
                  } catch (Exception e) {
                        System.err.println("Caught exception " + e.toString());
                  }
            ;
      }
}
```

Output:



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Anubhav>cd\

C:\>cd demo

C:\demo>javac GenerateDigitalSignature.java
Picked up _JAVA_OPTIONS: -Xmx512m

C:\demo>java GenerateDigitalSignature digital
Picked up _JAVA_OPTIONS: -Xmx512m

C:\demo>javac VerifyDigitalSignature.java
Picked up _JAVA_OPTIONS: -Xmx512m

C:\demo>java VerifyDigitalSignature publickey signature digital
Picked up _JAVA_OPTIONS: -Xmx512m
signature verifies: true

C:\demo>
```

Learning outcomes (What I have learnt):

With this, you have understood the importance of asymmetric cryptography, the working of digital signatures, the functionality of DSA, the steps involved in the signature verification, and its advantages over similar counterparts.

Evaluation Grid (To be created per the faculty's SOP and Assessment guidelines):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Worksheet completion including writing learning objectives/Outcomes. (To be submitted at the end of the day).		
2.	Post-Lab Quiz Result.		
3.	Student Engagement in Simulation/Demonstration/Performance and Controls/Pre-Lab Questions.		
	Signature of Faculty (with Date):	Total Marks Obtained:	