PRACTICAL-10

AIM : Write a program to Implement a Digital Signature algorithm.

INTRODUCTION:

- Digital Signatures are an asymmetrically encrypted hash of a digital message (data). It is a value that can provide a guarantee of authenticity, non-repudiation, and integrity.
- In other terms, it means you can verify the sender, date & time and message content have not been revealed or compromised.

Digital Signature Flow:

- Let "A" and "B" be the fictional actors in the cryptography system for better understanding.
- "A" is the sender and calculates the hash of the message and attaches signature which he wants to send using his private key.
- The other side "B" hashes the message and then decrypts the signature with A's public key and compares the two hashes
- If "B" finds the hashes matching then the message has not been altered or compromised.

CODE:

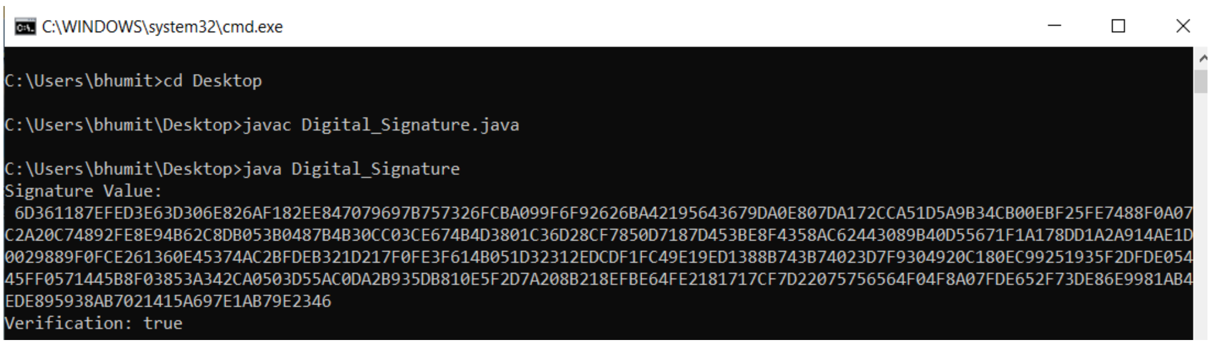
```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.Signature;
import java.util.Scanner;
import javax.xml.bind.DatatypeConverter;

public class Digital_Signature
{

        private static final String SIGNING_ALGORITHM = "SHA256withRSA";
        private static final String RSA = "RSA";
        private static Scanner sc;
public static byte[] Create_Digital_Signature( byte[] input, PrivateKey Key) throws
Exception {
                Signature signature = Signature.getInstance(SIGNING_ALGORITHM);
                        signature.initSign(Key);
                        signature.update(input);
                        return signature.sign();
                }
        public static KeyPair Generate_RSA_KeyPair() throws Exception
```

```
	{
		SecureRandom secureRandom = new SecureRandom();
		KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance(RSA);
		keyPairGenerator.initialize(2048, secureRandom);
		return keyPairGenerator.generateKeyPair();
	}
	public static boolean Verify_Digital_Signature(byte[] input,byte[]
signatureToVerify,PublicKey key)throws Exception
	{
		Signature signature = Signature.getInstance(SIGNING_ALGORITHM);
		signature.initVerify(key);
		signature.update(input);
		return signature.verify(signatureToVerify);
	}
	public static void main(String args[])throws Exception
	{
		String input = "bhumit";
		KeyPair keyPair = Generate_RSA_KeyPair();
		byte[] signature =
Create_Digital_Signature(input.getBytes(),keyPair.getPrivate());
		System.out.println("Signature Value:\n "+
DatatypeConverter.printHexBinary(signature));
		System.out.println( "Verification:
"+Verify_Digital_Signature(input.getBytes(),signature,
		keyPair.getPublic()));
	}
}
```

<u>OUTPUT</u>:



```
C:\Users\bhumit>cd Desktop

C:\Users\bhumit\Desktop>javac Digital_Signature.java

C:\Users\bhumit\Desktop>java Digital_Signature
Signature Value:
 6D361187EFED3E63D306E826AF182EE847079697B757326FCBA099F6F92626BA42195643679DA0E807DA172CCA51D5A9B34CB00EBF25FE7488F0A07
C2A20C74892FE8E94B62C8DB053B0487B4B30CC03CE674B4D3801C36D28CF7850D7187D453BE8F4358AC62443089B40D55671F1A178DD1A2A914AE1D
0029889F0FCE261360E45374AC2BFDEB321D217F0FE3F614B051D32312EDCDF1FC49E19ED1388B743B74023D7F9304920C180EC99251935F2DFDE054
45FF0571445B8F03853A342CA0503D55AC0DA2B935DB810E5F2D7A208B218EFBE64FE2181717CF7D22075756564F04F8A07FDE652F73DE86E9981AB4
EDE895938AB7021415A697E1AB79E2346
Verification: true
```