

EX.NO: 3

ROLL.NO: 210701275

DATE: Digital Signature Algorithm

AIM:

Demonstrating digital signature generation and verification using RSA and SHA-256.

ALGORITHM:

1. Generate RSA key pair with a 2048-bit key size.
2. Create digital signature by hashing input with SHA-256 and encrypting with private key.
3. Verify signature by decrypting with public key and comparing hash with input.
4. Output signature in hexadecimal format.
5. Output verification result as boolean.

PROGRAM:

```
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 Dsa {

    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);
}
```

```
// Driver Code
public static void main(String args[])
throws Exception
{

    String input
    = "GEEKSFORGEEKS IS A"
    + " COMPUTER SCIENCE PORTAL";
```

```

KeyPair keyPair
= Generate_RSA_KeyPair();

// Function Call
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()));
}
}

```

OUTPUT:



```

C:\Users\REC\cns>javac Dsa.java
C:\Users\REC\cns>java Dsa
Signature Value:
638257EB4DC16FFB8D1F4F338FEA98EB5069856EDB4A004376D699289798A2FD6466DB640BAD3C3
EC6C9E474728ADBDEF9FD0DD8D057F89C4E8310A9BBE6D50948E493ABDA02026BC225023665073E
EEA9DAADA1D718E27262BEC8CF93067F1E2C79C4E5C20E973F8393E317488933E58EFC17CB1F2A4
45E607576FC284689A444346A69426302953ABF41DF40CFF3639AEB1E66E79FC76841D4ABC73E505
0EF92DA7FDF2CA7D619DE7BB92849FB30DBA6F58B26DF9AE7C2AA1EF61A09ECB8AC2449E2D4ED29B
4C145CD9EEE781C131FCFCF9C43FD6BBAB5621E7B2150859F4D5B1B633D6A06B87EE13478A355A76
EDD1656164CE13C154DA3458F9C7A073B
Verification: true
C:\Users\REC\cns>_

```

RESULT: