SSN College of Engineering,
Department of Computer Science and Engineering
IT 8761 Security Laboratory

Exercise 9:

To implement the Signature Scheme - Digital Signature Standard

Programming Language: Java

Hints:

Module 1: Creating the digital signature

1. Create a KeyPairGenerator object

The **KeyPairGenerator** class provides **getInstance()** method which accepts a String variable representing the required key-generating algorithm and returns a KeyPairGenerator object that generates keys.

2. Initialize the KeyPairGenerator object

The **KeyPairGenerator** class provides a method named **initialize()** this method is used to initialize the key pair generator. This method accepts an integer value representing the key size.

3. Generate the KeyPairGenerator

Generate the **KeyPair** using the **generateKeyPair()** method.

4. Get the private key from the pair

Get the private key from the generated KeyPair object using the **getPrivate()** method.

5. Create a signature object

The **getInstance()** method of the **Signature** class accepts a string parameter representing required signature algorithm and returns the respective Signature object.

6. Initialize the Signature object

The **initSign()** method of the Signature class accepts a **PrivateKey** object and initializes the current Signature object.

7. Add data to the Signature object

The **update()** method of the Signature class accepts a byte array representing the data to be signed or verified and updates the current object with the data given.

8. Calculate the Signature

The **sign()** method of the **Signature** class returns the signature bytes of the updated data.

Module 2: Verifying Signature

1. Create a KeyPairGenerator object

The **KeyPairGenerator** class provides **getInstance()** method which accepts a String variable representing the required key-generating algorithm and returns a KeyPairGenerator object that generates keys.

2. Initialize the KeyPairGenerator object

The **KeyPairGenerator** class provides a method named **initialize()** method. This method is used to initialize the key pair generator. This method accepts an integer value representing the key size.

3. Generate the KeyPairGenerator

Generate the **KeyPair** using the **generateKeyPair()** method.

4. Get the private key from the pair

Get the private key from the generated KeyPair object using the **getPrivate()** method.

5. Create a signature object

The **getInstance()** method of the **Signature** class accepts a string parameter representing required signature algorithm and returns the respective Signature object.

6. Initialize the Signature object

The **initSign()** method of the Signature class accepts a **PrivateKey** object and initializes the current Signature object.

7. Add data to the Signature object

The **update()** method of the Signature class accepts a byte array representing the data to be signed or verified and updates the current object with the data given.

8. Calculate the Signature

The **sign()** method of the **Signature** class returns the signature bytes of the updated data.

9. Initialize the signature object for verification

To verify a Signature object you need to initialize it first using the **initVerify()** method it method accepts a **PublicKey** object.

10. Update the data to be verified

Update the initialized (for verification) object with the data the data to be verified

11. Verify the Signature

The **verify()** method of the Signature class accepts another signature object and verifies it with the current one. If a match occurs, it returns true else it returns false.

12. Check the Boolean output for whether sign is verified or not