

- ❖ Write a Java program that allows users to create a digital signature for a piece of digital data provided by the user and then authenticate the digital signature.

The Implementation Java code is given below:

```
DigitalSignatureDemo.java
1  import java.security.*;
2  import java.util.Base64;
3  import java.util.Scanner;
4  public class DigitalSignatureDemo {
5      public static void main(String[] args) {
6          try {
7              // Step 1: Key Generation
8              KeyPair keyPair = generateKeyPair();
9              // Step 2: Create Digital Signature
10             String originalData = getUserInput( prompt: "Enter the original data: ");
11             byte[] digitalSignature = createDigitalSignature(originalData, keyPair.getPrivate());
12             // Display the generated digital signature
13             System.out.println("Generated Digital Signature: " + Base64.getEncoder().encodeToString(digitalSignature));
14             // Step 3: Authenticate Digital Signature
15             String inputData = getUserInput( prompt: "Enter the original data for authentication: ");
16             String signatureInput = getUserInput( prompt: "Enter the digital signature for authentication: ");
17             boolean isAuthentic = verifyDigitalSignature(inputData, Base64.getDecoder().decode(signatureInput), keyPair.getPublic());
18             // Display the authentication result
19             System.out.println("Digital Signature Authenticity: " + (isAuthentic ? "Valid" : "Invalid"));
20         } catch (Exception e) {
21             System.err.println("An error occurred: " + e.getMessage());
22         }
23     }
24     1 usage
25     private static KeyPair generateKeyPair() throws NoSuchAlgorithmException {
26         KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance( algorithm: "RSA");
27         keyPairGenerator.initialize( keysize: 2048); // You can adjust the key size
28         return keyPairGenerator.generateKeyPair();
29     }
30     1 usage
31     @ private static byte[] createDigitalSignature(String data, PrivateKey privateKey) throws NoSuchAlgorithmException, InvalidKeyException, SignatureException {
32         Signature signature = Signature.getInstance( algorithm: "SHA256withRSA");
33         signature.initSign(privateKey);
34         signature.update(data.getBytes());
35         return signature.sign();
36     }
37     1 usage
38     @ private static boolean verifyDigitalSignature(String data, byte[] signature, PublicKey publicKey)
39         throws NoSuchAlgorithmException, InvalidKeyException, SignatureException {
40         Signature verifier = Signature.getInstance( algorithm: "SHA256withRSA");
41         verifier.initVerify(publicKey);
42         verifier.update(data.getBytes());
43         return verifier.verify(signature);
44     }
45     3 usages
46     private static String getUserInput(String prompt) {
47         Scanner scanner = new Scanner(System.in);
48         System.out.print(prompt);
49         return scanner.nextLine();
50     }
51 }
```

❖ Output:

1. **If data and signature is valid:** Here, the original data is 255 and the data is 260 for authentication. Because the data has been Same, the authenticity of the data is therefore valid.

```
Output Clear
java -cp /tmp/J7iL1L28ej DigitalSignatureDemo
Enter the original data: 255
Generated Digital Signature: jtGcT48GwlrIs7mCI05IJHsFH6pXEaH0dTCBDdNY54sNvy
/oarteDLwy4q8HDNGZ09H22q4BLsBP0PKTpOnIQGSwoR5FVAufMrBD2u7wKYJer7Q1bBf8qfYqjSZ4FC
+CCY9aNtcKy9P032D6DvRaaiAZChh824mL0xF4xP5ztIZ8GHqEvcGNskw
+DFvcNC8nvH3rMQ4Ug89BambSPDWptJ4QIT4192WBZ5av2gyaovsad8vuEhz8aMEViTgZAU4s+rm9NU6tYwTLjnTPoLYL
+sBLPfSZvLvvgWdGbXWIHFci0i5pbAVcFeLZVRooTuPa7DFYxeNlB0z63vhB49oSAA==
Enter the original data for authentication: 255
Enter the digital signature for authentication: jtGcT48GwlrIs7mCI05IJHsFH6pXEaH0dTCBDdNY54sNvy
/oarteDLwy4q8HDNGZ09H22q4BLsBP0PKTpOnIQGSwoR5FVAufMrBD2u7wKYJer7Q1bBf8qfYqjSZ4FC
+CCY9aNtcKy9P032D6DvRaaiAZChh824mL0xF4xP5ztIZ8GHqEvcGNskw
+DFvcNC8nvH3rMQ4Ug89BambSPDWptJ4QIT4192WBZ5av2gyaovsad8vuEhz8aMEViTgZAU4s+rm9NU6tYwTLjnTPoLYL
+sBLPfSZvLvvgWdGbXWIHFci0i5pbAVcFeLZVRooTuPa7DFYxeNlB0z63vhB49oSAA==
Digital Signature Authenticity: Valid
```

2. **If one is invalid (data or signature):** Here, the original data is 255 and the data is 260 for authentication. Because the data has been altered, the authenticity of the data is therefore invalid.

```
Output Clear
java -cp /tmp/kUTc2Zvz5I DigitalSignatureDemo
Enter the original data: 255
Generated Digital Signature: HzAGNtVlRq+4lvo8B89JQ9bpRNmRoXMU9gdEzvaZyDjn4HkovYb9mo944
+ZtNebSILxxJzN8ZijwY0tNpIUG7diN8TXtQ/gxp00
/fGpZXI7F2Eq70mYvaAf3Y81A83ygFh7Jyln3td5cEnEq8XYSo3Tt0TmM2lHZoLaTmBj9l10qu6HApma81mNEDOEow
++3jgmpmp3DXZFbTaEmSfoUMGym/q8kaku8LPnkK7vgOnkJcZZd/pdwfVfGzB/94BVXxHmXdfyBqElHFhC45ZghEV7ChjH7
+R8xwwigjxynP9m3Y/8W0nIjur63cD9ocsXPOY5b1Lc7Lgjz7PNhkJhJScA==
Enter the original data for authentication: 260
Enter the digital signature for authentication: HzAGNtVlRq+4lvo8B89JQ9bpRNmRoXMU9gdEzvaZyDjn4HkovYb9mo944
+ZtNebSILxxJzN8ZijwY0tNpIUG7diN8TXtQ/gxp00
/fGpZXI7F2Eq70mYvaAf3Y81A83ygFh7Jyln3td5cEnEq8XYSo3Tt0TmM2lHZoLaTmBj9l10qu6HApma81mNEDOEow
++3jgmpmp3DXZFbTaEmSfoUMGym/q8kaku8LPnkK7vgOnkJcZZd/pdwfVfGzB/94BVXxHmXdfyBqElHFhC45ZghEV7ChjH7
+R8xwwigjxynP9m3Y/8W0nIjur63cD9ocsXPOY5b1Lc7Lgjz7PNhkJhJScA==
Digital Signature Authenticity: Invalid
```