Interlanguage Test Report - Python | Nodejs

1. **Creating Key Pairs in Java and verifying if the shared keys generated in both environments are the same.**

- **Creating a set of keypairs in Java:**

```
// Generate Key Pair for User 1.
DHKeyPair keyPair1 = KeyUtil.generateKeyPair();
System.out.println("Key Pair 1 ==> "+ keyPair1.toString());

// Generate Key Pair for User 2.
DHKeyPair keyPair2 = KeyUtil.generateKeyPair();
System.out.println("Key Pair 2 ==> "+ keyPair2.toString());
```

- **Output:**

```
Key Pair 1 ==> DHKeyPair [publicKey=MCowBQYDK2VuAyEAx4C8YJ4HFeyn3xqaiggb2wzj/EpuHhNA91L8O
DhFTzw=, privateKey=MC4CAQAwBQYDK2VuBCIEIE3ks5ncg1yyTsgPaENupalAE3CGUJKtnilmfzgfOwXI]
Key Pair 2 ==> DHKeyPair [publicKey=MCowBQYDK2VuAyEAsUsihdnQihvE6v2Dsy4zTNDHFDYrV3U6zJUtX
7wb4Ws=, privateKey=MC4CAQAwBQYDK2VuBCIEIMWmoHoC/OI/YyNNCh8sJIJBOUvcm7DctxzAis6HsIVH]
```

- **Generating Shared Keys for the above keypairs in Java:**

```
// Generate Shared Key with User 1's Private Key and User 2's Public Key.
String sharedKey1 = KeyUtil.generateSharedKey(keyPair1.getPrivateKey(),
keyPair2.getPublicKey());
System.out.println("SharedKey1 ==> "+ sharedKey1);

// Generate Shared Key with User 2's Private Key and User 1's Public Key.
String sharedKey2 = KeyUtil.generateSharedKey(keyPair2.getPrivateKey(),
keyPair1.getPublicKey());
System.out.println("SharedKey2 ==> "+ sharedKey2);
```

- **Output:**

```
SharedKey1 ==> 3e+FNtzeoAF57nB7vrl6f2tqHgyDNV2Ff29b9QWIuRo=
SharedKey2 ==> 3e+FNtzeoAF57nB7vrl6f2tqHgyDNV2Ff29b9QWIuRo=
```

- **Generating Shared keys for the same key pairs in Python:**

```
  const sharedKey1 = generateSharedKey(
 "MC4CAQAwBQYDK2VuBCIEIE3ks5ncg1yyTsgPaENupalAE3CGUJKtnilmfzgfOwXI",
 "MCowBQYDK2VuAyEAsUsihdnQihvE6v2Dsy4zTNDHFDYrV3U6zJUtX7wb4Ws="
);
console.log("SharedKey1 ==> " + sharedKey1);

// // Generate Shared Key with User 2's Private Key and User 1's Public Key.
const sharedKey2 = generateSharedKey(
 "MC4CAQAwBQYDK2VuBCIEIMWmoHoC/OI/YyNNCh8sJIJBOUvcm7DctxzAis6HsIVH",
 "MCowBQYDK2VuAyEAx4C8YJ4HFeyn3xqaiggb2wzj/EpuHhNA91L8ODhFTzw="
);
console.log("SharedKey2 ==> " + sharedKey2);
```

- **Output:**

```
shared_key1: 3e+FNtzeoAF57nB7vrl6f2tqHgyDNV2Ff29b9QWIuRo=
shared_key2: 3e+FNtzeoAF57nB7vrl6f2tqHgyDNV2Ff29b9QWIuRo=
```

Observation:

Shared keys generated from both utilities for the same set of keypairs generated from the Java utility are the same.

2. **Encrypting the text in Java and decrypting in Nodejs (Using above-shared keys)**

- **Encrypting the text in Java**

```
// Initializing the raw text to be encrypted.
String rawData = "Hello This is ONDC Test Data";

// Encrypting the raw data.
String encryptedData = EncryptionUtil.encryptData(sharedKey1, rawData);
System.out.println("Encrypted Data ===> " + encryptedData);
```

- **Output:**

```
Encrypted Data ===> eyJub25jZSI6IlA4M2VJV0xFVk5IMjUxblQiLCJlbmNyeXB0ZWRfZGF0YSI6Im5oZTFGT
mhnaDZQZUFzUC9vWWxKWThQWlhWSTJ3dkpZNVVsZnh3XHUwMDNkXHUwMDNkIiwiaG1hYyI6IlNRMEIrc2p5UkFSQX
JLc2ovS2lKSkFcdTAwM2RcdTAwM2QifQ==
```

- **Decrypting the above text in nodejs:**

```
const encryptedData =
"eyJub25jZSI6IlA4M2VJV0xFVk5IMjUxblQiLCJlbmNyeXB0ZWRfZGF0YSI6Im5oZTFGTmhnaDZQZU
FzUC9vWWxKWThQWlhWSTJ3dkpZNVVsZnh3XHUwMDNkXHUwMDNkIiwiaG1hYyI6IlNRMEIrc2p5UkFSQ
XJLc2ovS2lKSkFcdTAwM2RcdTAwM2QifQ==";

const sharedKey = "3e+FNtzeoAF57nB7vrl6f2tqHgyDNV2Ff29b9QWIuRo=";

const decryptedData = decryptData(sharedKey, encryptedData);

console.log(decryptedData);
```

- **Output:**

```
decrypted Data ===>  Hello This is ONDC Test Data
```

Observation:

Data encrypted in Java utility was decrypted successfully in Node.js Utility.

3. **Generating Key Pairs in Node.js and verifying whether the shared keys generated in both environments are the same:**

- **Generating a set of Key pairs in Python:**

```
// Generate Key Pair for User 1.
const keyPair1 = generateKeyPair();
console.log("Key Pair 1 ==> ", keyPair1);

// Generate Key Pair for User 2.
const keyPair2 = generateKeyPair();
console.log("Key Pair 2 ==> ", keyPair2);
```

- **Output:**

```
Key Pair 1 ==>  {
  publicKey: 'MCowBQYDK2VuAyEAKSH2C3vjjeJykXKzhZGISY4Z5AhBX1F5pXTyxglgkkU=',
  privateKey: 'MC4CAQAwBQYDK2VuBCIEIIiE2k0kYcVjTbQ9BSWXbdtXQsWxRX+Er8aPxspif3xL'
}
Key Pair 2 ==>  {
  publicKey: 'MCowBQYDK2VuAyEAf4SFqEYYCop97r6XFo4g7bIgMps09mwdfyjMghb460s=',
  privateKey: 'MC4CAQAwBQYDK2VuBCIEIAimThnpMhx0QitdopYVb+W4jcd0dIhlpRFvM8xTBbJT'
}
```

- **Generating Shared key in Python:**

```
// Generate Shared Key with User 1's Private Key and User 2's Public Key.
const sharedKey1 = generateSharedKey(keyPair1.privateKey, keyPair2.publicKey);
console.log("SharedKey1 ==> " + sharedKey1);

// Generate Shared Key with User 2's Private Key and User 1's Public Key.
const sharedKey2 = generateSharedKey(keyPair2.privateKey, keyPair1.publicKey);
console.log("SharedKey2 ==> " + sharedKey2);
```

- **Output:**

```
SharedKey1 ==> q6F5Fwb5O8bxptR8liG65eu4jSq1gLFjNg6EsZVuAWc=
SharedKey2 ==> q6F5Fwb5O8bxptR8liG65eu4jSq1gLFjNg6EsZVuAWc=
```

- **Generating shared key in Java for the above key pairs:**

```
      String publicKey1 =
"MCowBQYDK2VuAyEAKSH2C3vjjeJykXKzhZGISY4Z5AhBX1F5pXTyxglgkkU=";
      String privateKey1 =
"MC4CAQAwBQYDK2VuBCIEIIiE2k0kYcVjTbQ9BSWXbdtXQsWxRX+Er8aPxspif3xL";

      String publicKey2 =
"MCowBQYDK2VuAyEAf4SFqEYYCop97r6XFo4g7bIgMps09mwdfyjMghb460s=";
      String privateKey2 =
"MC4CAQAwBQYDK2VuBCIEIAimThnpMhx0QitdopYVb+W4jcd0dIhlpRFvM8xTBbJT";

      // Generate Shared Key with User 1's Private Key and User 2's Public Key.
      String sharedKey1 = KeyUtil.generateSharedKey(privateKey1, publicKey2);
      System.out.println("SharedKey1 ==> "+ sharedKey1);

      // Generate Shared Key with User 2's Private Key and User 1's Public Key.
      String sharedKey2 = KeyUtil.generateSharedKey(privateKey2, publicKey1);
      System.out.println("SharedKey2 ==> "+ sharedKey2);
```

- **Output:**

```
SharedKey1 ==> q6F5Fwb5O8bxptR8liG65eu4jSq1gLFjNg6EsZVuAWc=
SharedKey2 ==> q6F5Fwb5O8bxptR8liG65eu4jSq1gLFjNg6EsZVuAWc=
```

Observation:

The shared keys generated for both the utilities with the same set of keypairs generated from nodejs utility are the same.

4. **Encrypting the text in Python utility and decrypting it in Java utility:**

- **Encrypting in Python utility:**

```
// Initializing the raw text to be encrypted.
const MESSAGE_DATA = "Hello This is ONDC Test Data";

// Encrypting the raw data.
const encryptedData = encryptData(sharedKey1, MESSAGE_DATA);
console.log("Encrypted Data ===> " + encryptedData);
```

- **Output:**

```
Encrypted Data ===> eyJlbmNyeXB0ZWRfZGF0YSI6ImE3Q0o3QWJTMDBSNlBFUXZJVC81WmFsRUh6OW44Vko1Y2
1mcWtBPT0iLCJobWFjIjoibFBCbTQxWFB4ZzBIeC8ydzNhcW1HZz09Iiwibm9uY2UiOiJLTTZWb0M5YmNtb2Fzb3Qw
In0=
```

- **Decrypting the above text in Java utility:**

```
      String encryptedData =
"eyJlbmNyeXB0ZWRfZGF0YSI6ImE3Q0o3QWJTMDBSNlBFUXZJVC81WmFsRUh6OW44Vko1Y21mcWtBPT0iLCJob
WFjIjoibFBCbTQxWFB4ZzBIeC8ydzNhcW1HZz09Iiwibm9uY2UiOiJLTTZWb0M5YmNtb2Fzb3QwIn0=";

      // Decrypting the Encrypted data.
      String decryptedData =  EncryptionUtil.decryptData(sharedKey2, encryptedData);
      System.out.println("Decrypted Data ===> " + decryptedData);
```

- **Output:**

```
Decrypted Data ===> Hello This is ONDC Test Data
```

Observation:

Text encrypted in the Node.js utility was decrypted successfully in the Java utility.