Interlanguage Test report - Java | Python

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:**

```
keypair1 = key_util.DHKeyPair(
private_key="MC4CAQAwBQYDK2VuBCIEIE3ks5ncg1yyTsgPaENupalAE3CGUJKtnilmfzgfOwXI",
public_key= "MCowBQYDK2VuAyEAx4C8YJ4HFeyn3xqaiggb2wzj/EpuHhNA91L8ODhFTzw=")

keypair2 = key_util.DHKeyPair(
private_key="MC4CAQAwBQYDK2VuBCIEIMWmoHoC/OI/YyNNCh8sJIJBOUvcm7DctxzAis6HsIVH",
public_key= "MCowBQYDK2VuAyEAsUsihdnQihvE6v2Dsy4zTNDHFDYrV3U6zJUtX7wb4Ws=")

shared_key1=key_util.generate_shared_key(keypair1.private_key,keypair2.public_key)
shared_key2=key_util.generate_shared_key(keypair2.private_key,keypair1.public_key)

println("shared_key1:",shared_key1)
println("shared_key2:",shared_key2)
```

- **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 Python (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 python:**

```
    encrypted_data =
"eyJub25jZSI6IlA4M2VJV0xFVk5IMjUxblQiLCJlbmNyeXB0ZWRfZGF0YSI6Im5oZTFGTmhnaDZQZUFzUC9vW
WxKWThQWlhWSTJ3dkpZNVVsZnh3XHUwMDNkXHUwMDNkIiwiaG1hYyI6IlNRMEIrc2p5UkFSQXJLc2ovS2lKSkF
cdTAwM2RcdTAwM2QifQ=="

  decrypted_data=encryption_util.decrypt_data(shared_key2,encrypted_data)
  println("decrypted Data ===> ",decrypted_data)
```

- **Output:**

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

Observation:

Data encrypted in Java utility was decrypted successfully in Python Utility.

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

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

```
keypair1=key_util.generate_key_pair()
keypair2=key_util.generate_key_pair()

print("key_pair_1.public_key:", keypair1.public_key)
print("key_pair_1.private_key:", keypair1.private_key)
print("key_pair_2.public_key:", keypair2.public_key)
print("key_pair_2.private_key:", keypair2.private_key)
```

- **Output:**

```
key_pair_1.public_key: MCowBQYDK2VuAyEAm9xpzjzSOzhZh/bXWIOVYftXWEZdpGl53kMQML9+vD4=
key_pair_1.private_key: MC4CAQAwBQYDK2VuBCIEIHg/4IQYx/BWwihhLhzFD4RAhYBCElHHSNz7gjMrtSRu
key_pair_2.public_key: MCowBQYDK2VuAyEATay/vLATgWbZg0VmWzYluEZ6R7BADZ6aSDrpsDorMR4=
key_pair_2.private_key: MC4CAQAwBQYDK2VuBCIEIAjYZfcV30LUTe2Jg1PtNF3X/FXEG42HKTX+04lyVEdF
```

- **Generating Shared key in Python:**

```
shared_key1=key_util.generate_shared_key(keypair1.private_key,keypair2.public_key)
shared_key2=key_util.generate_shared_key(keypair2.private_key,keypair1.public_key)

println("shared_key1:",shared_key1)
println("shared_key2:",shared_key2)
```

- **Output:**

```
shared_key1: hXgHvkFK/o1gsbFLxAi6/JkSxDz52KCzgZQZ9vMm3X8=
shared_key2: hXgHvkFK/o1gsbFLxAi6/JkSxDz52KCzgZQZ9vMm3X8=
```

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

```
      String publicKey1 =
"MCowBQYDK2VuAyEAm9xpzjzSOzhZh/bXWIOVYftXWEZdpGl53kMQML9+vD4=";
      String privateKey1 =
"MC4CAQAwBQYDK2VuBCIEIHg/4IQYx/BWwihhLhzFD4RAhYBCElHHSNz7gjMrtSRu";

      String publicKey2 =
"MCowBQYDK2VuAyEATay/vLATgWbZg0VmWzYluEZ6R7BADZ6aSDrpsDorMR4=";
      String privateKey2 =
"MC4CAQAwBQYDK2VuBCIEIAjYZfcV30LUTe2Jg1PtNF3X/FXEG42HKTX+04lyVEdF";

      // 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 ==> hXgHvkFK/o1gsbFLxAi6/JkSxDz52KCzgZQZ9vMm3X8=
SharedKey2 ==> hXgHvkFK/o1gsbFLxAi6/JkSxDz52KCzgZQZ9vMm3X8=
```

Observation:

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

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

- **Encrypting in Python utility:**

```
raw_data = "Hello This is ONDC Test Data"

encrypted_data=encryption_util.encrypt_data(shared_key1,raw_data)
println("Encrypted Data ===> ", encrypted_data)
```

- **Output:**

```
Encrypted Data ===>  eyJub25jZSI6ICJSeUdlOCs0dnRPbWlWbFJ0IiwgImVuY3J5cHRlZF9kYXRhIjogIkFoaEEvVm9sNi9jMVlYQjAyclpVUFE1K1R0RVBBdUhJbVQvOGpBPT0iLCAiaG1hYyI6ICIrNUE1d0FxemsxYXpnOHNoMnJHbWRRPT0ifQ==
```

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

```
      String encryptedData =
"eyJub25jZSI6ICJSeUdlOCs0dnRPbWlWbFJ0IiwgImVuY3J5cHRlZF9kYXRhIjogIkFoaEEvVm9sNi9jMVlYQjAyclpVUFE1K1R0RVBBdUhJbVQvOGpBPT0iLCAiaG1hYyI6ICIrNUE1d0FxemsxYXpnOHNoMnJHbWRRPT0ifQ==";

      // 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 Python utility was decrypted successfully in the Java utility.