IMPLEMENTING OPENSSL ENCRYPTION IN JAVA AND MAVEN PROJECTS: PROTECTING SENSITIVE INFORMATION

The demo covers the following steps:

Generating RSA Keys: You'll learn how to create a private and public key pair using OpenSSL.

Encrypting Data: You will see how to use the public key to encrypt both files and text, ensuring that sensitive data remains secure.

Decrypting Data: We'll demonstrate how to decrypt the encrypted data using the private key.

Integrating OpenSSL with Java: We'll integrate OpenSSL's encryption and decryption capabilities into a Java Maven project, providing a practical solution for securing your data within the Java ecosystem.

Prerequisites:

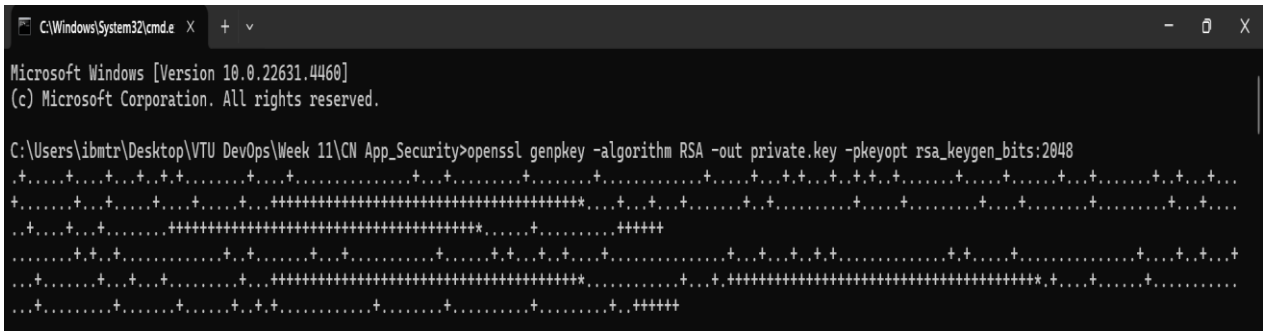
- Ensure you have OpenSSL installed on your system.
- Java and Maven set up correctly in your project.

Steps:

Step 1: Generate SSL Keys using OpenSSL

1. **Create a Private Key**: Open a terminal and run the following command to generate a private key:

openssl genpkey -algorithm RSA -out private.key -pkeyopt rsa_keygen_bits:2048



This command creates a private key (private.key) with a 2048-bit RSA encryption.

2. **Generate a Public Key**: Use the following command to generate a public key from the private key:

openssl rsa -pubout -in private.key -out public.key

```
C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security>openssl genpkey -algorithm RSA -out private.key -pkeyopt rsa_keygen_bits:2048
.+.....+....+...+..+.+........+....+...............+...+........+........+.............+.....+...+.+...+..+.+..+.......+.....+......+...+.......+..+...+...
+.......+...+.....+....+.....+...++++++++++++++++++++++++++++++++++++++++*....+...+...+.......+..+..........+.....+.........+....+........+.........+...+....
..+....+...+........++++++++++++++++++++++++++++++++++++++++*......+..........++++++
........+.+..+.............+..+.......+...+...........+......+.+...+..+....+...............+...+...+..+.+..............+.+.....+...............+....+..+...+
...+.......+...+...+.........+...++++++++++++++++++++++++++++++++++++++++*............+...+.++++++++++++++++++++++++++++++++++++++++*.+....+......+...........
...+.........+.......+......+..+.+............+........+..........+.........+..++++++

C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security>openssl rsa -pubout -in private.key -out public.key
writing RSA key

C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security>echo. >sensitive.txt
```

Now, you have two files:

- private.key: Private key used to decrypt data.
- public.key: Public key used to encrypt data.

Name	Date modified	Type	Size
.settings	03-12-2024 19:36	File folder	
backend-service	03-12-2024 20:13	File folder	
front-service	03-12-2024 20:13	File folder	
.gitignore	18-11-2024 12:26	Git Ignore Source ...	1 KB
.project	18-11-2024 12:26	PROJECT File	1 KB
deployment	18-11-2024 12:47	Yaml Source File	3 KB
LICENSE	18-11-2024 12:26	File	8 KB
pom	03-12-2024 20:08	Microsoft Edge HT...	1 KB
private.key	03-12-2024 20:04	KEY File	2 KB
public.key	03-12-2024 20:04	KEY File	1 KB
README	18-11-2024 12:26	Markdown Source ...	1 KB
sensitive.enc	03-12-2024 20:07	ENC File	1 KB
sensitive	03-12-2024 20:05	Text Document	1 KB
sensitive_dec	03-12-2024 20:07	Text Document	1 KB

Step 2: Encrypt Data Using OpenSSL (Command-Line)

You can use OpenSSL to encrypt data using the public key. Here's how to encrypt a file or text:

1. **Encrypt a File**: Create a file with sensitive data (e.g., sensitive.txt) and encrypt it using the public key:

openssl pkeyutl -encrypt -inkey public.key -pubin -in sensitive.txt -out sensitive.enc

This will create an encrypted file sensitive.enc.

2. **Encrypt Text Directly**: To encrypt a small piece of text:

echo -n "SensitiveData" | openssl pkeyutl -encrypt -inkey public.key -pubin -out sensitive.enc

Step 3: Decrypt Data Using OpenSSL (Command-Line)

To decrypt the data, use the private key:

1. **Decrypt a File**: Run the following command to decrypt an encrypted file:

   openssl pkeyutl -decrypt -inkey private.key -in sensitive.enc -out sensitive_dec.txt

2. **Decrypt Encrypted Text**: If you encrypted a piece of text, use this command to decrypt it:

   openssl pkeyutl -decrypt -inkey private.key -in sensitive.enc

Step 4: Integrating OpenSSL with Java for Encryption/Decryption

Now, let's integrate the encryption and decryption functionality into your Java Maven project using OpenSSL.

1. **Add Required Dependencies**: Ensure that you have the following Maven dependencies in your pom.xml for working with encryption algorithms:

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

        <modelVersion>4.0.0</modelVersion>

        <groupId>com.ibm.developer</groupId>

        <artifactId>managing-cn-apps-on-k8s</artifactId>

        <version>1.0.0</version>

        <packaging>pom</packaging>
```

```
    <modules>
        <module>front-service</module>
        <module>backend-service</module>
    </modules>
    <dependencies>
        <dependency>
            <groupId>org.bouncycastle</groupId>
            <artifactId>bcprov-jdk15on</artifactId>
            <version>1.70</version> <!-- Check for the latest version -->
        </dependency>
    </dependencies>
</project>
```

2. **Create Java Methods for Encryption/Decryption**: Below is a simple Java class for encrypting and decrypting data using RSA keys generated via OpenSSL.

Go to C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security\backend-service\src\main\java

Create a File Name **OpenSSLExample.java**

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.KeyFactory;
import java.security.spec.X509EncodedKeySpec;
import java.security.spec.PKCS8EncodedKeySpec;
```

```
import javax.crypto.Cipher;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class OpenSSLExample {
    public static void main(String[] args) {
        try {
            // Load public and private keys
            PublicKey publicKey = loadPublicKey("public.key");
            PrivateKey privateKey = loadPrivateKey("private.key");

            // Encrypt the data
            String dataToEncrypt = "SensitiveData";
            byte[] encryptedData = encrypt(publicKey, dataToEncrypt);
            Path encryptedFilePath = Paths.get("sensitive.enc");
            Files.write(encryptedFilePath, encryptedData);

            // Decrypt the data
            byte[] decryptedData = decrypt(privateKey, encryptedData);
            System.out.println("Decrypted Data: " + new String(decryptedData));

        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }

    // Encrypt data using the public key
    public static byte[] encrypt(PublicKey publicKey, String data) throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        return cipher.doFinal(data.getBytes());
    }

    // Decrypt data using the private key
    public static byte[] decrypt(PrivateKey privateKey, byte[] encryptedData) throws
Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        return cipher.doFinal(encryptedData);
    }

    // Load the public key from a file
    public static PublicKey loadPublicKey(String filePath) throws Exception {
        byte[] keyBytes = Files.readAllBytes(Paths.get(filePath));
        X509EncodedKeySpec spec = new X509EncodedKeySpec(keyBytes);
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        return keyFactory.generatePublic(spec);
    }
```

```
    // Load the private key from a file
    public static PrivateKey loadPrivateKey(String filePath) throws Exception {
        byte[] keyBytes = Files.readAllBytes(Paths.get(filePath));
        PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(keyBytes);
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        return keyFactory.generatePrivate(spec);
    }
}
```

3. **Build the Project**: After integrating the code above into your project, run the following Maven command to compile and package the project:

mvn clean package

```
C:\Windows\System32\cmd.e

C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security>mvn clean package
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Build Order:
[INFO]
[INFO] front-service                                                      [jar]
[INFO] backend-service                                                    [jar]
[INFO] managing-cn-apps-on-k8s                                            [pom]
[INFO]
[INFO] ------------------< com.ibm.developer:front-service >-------------------
[INFO] Building front-service 0.0.1-SNAPSHOT                              [1/3]
[INFO]   from front-service\pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.1.0:clean (default-clean) @ front-service ---
[INFO] Deleting C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security\front-service\target
[INFO]
[INFO] --- resources:3.2.0:resources (default-resources) @ front-service ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] Copying 1 resource
[INFO] Copying 0 resource
[INFO] The encoding used to copy filtered properties files have not been set. This means that the same encoding will be used to copy filtered properties fil
es as when copying other filtered resources. This might not be what you want! Run your build with --debug to see which files might be affected. Read more at
 https://maven.apache.org/plugins/maven-resources-plugin/examples/filtering-properties-files.html
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ front-service ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security\front-service\target\classes
[INFO]
[INFO] --- resources:3.2.0:testResources (default-testResources) @ front-service ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] skip non existing resourceDirectory C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security\front-service\src\test\resources
[INFO]
[INFO] --- compiler:3.8.1:testCompile (default-testCompile) @ front-service ---
[INFO] No sources to compile
[INFO]
[INFO] --- surefire:2.22.2:test (default-test) @ front-service ---
[INFO] No tests to run.
```

```
C:\Windows\System32\cmd.e

[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ backend-service ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security\backend-service\target\classes
[INFO]
[INFO] --- resources:3.2.0:testResources (default-testResources) @ backend-service ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] skip non existing resourceDirectory C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security\backend-service\src\test\resources
[INFO]
[INFO] --- compiler:3.8.1:testCompile (default-testCompile) @ backend-service ---
[INFO] No sources to compile
[INFO]
[INFO] --- surefire:2.22.2:test (default-test) @ backend-service ---
[INFO] No tests to run.
[INFO]
[INFO] --- jar:3.2.0:jar (default-jar) @ backend-service ---
[INFO] Building jar: C:\Users\ibmtr\Desktop\VTU DevOps\Week 11\CN App_Security\backend-service\target\backend-service-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:2.4.0:repackage (repackage) @ backend-service ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] -------------< com.ibm.developer:managing-cn-apps-on-k8s >--------------
[INFO] Building managing-cn-apps-on-k8s 1.0.0                            [3/3]
[INFO]   from pom.xml
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ managing-cn-apps-on-k8s ---
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] front-service 0.0.1-SNAPSHOT ....................... SUCCESS [  5.947 s]
[INFO] backend-service 0.0.1-SNAPSHOT ...................... SUCCESS [  0.871 s]
[INFO] managing-cn-apps-on-k8s 1.0.0 ....................... SUCCESS [  0.067 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  7.528 s
[INFO] Finished at: 2024-12-03T20:13:15+05:30
[INFO] ------------------------------------------------------------------------
```