**java\_springboot\_restful\_07\_OAuth2\_JWT\_AddingPrivatePublicKey Generator**

**Removing things**

Remove the **certs** folder from the project

Remove the dependency for **configuration processor** from pom.xml

In Springrestdemoapplication.java -> remove rsa configuration

Remove **RsaKeyProperties.java** from config folder

**In SecurityConfig.java**

JWK stands for JSON Web key.

This piece of code below is nothing but, source which will provide for our encoder and decoder

private RSAKey rsaKey;

@Bean

public JWKSource<SecurityContext> jwkSource() {

   rsaKey = Jwks.generateRsa();

   JWKSet jwkSet = new JWKSet(rsaKey);

   return (jwkSelector, securityContext) -> jwkSelector.select(jwkSet);

}

**In security folder -> add KeyGenerator.java -> below is the full code**

package org.studyeasy.SpringRestdemo.security;

import org.springframework.stereotype.Component;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

@Component

final class KeyGeneratorUtils {

    private KeyGeneratorUtils() {}

    static KeyPair generateRsaKey() {

        KeyPair keyPair;

        try {

            KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");

            keyPairGenerator.initialize(2048);

            keyPair = keyPairGenerator.generateKeyPair();

        } catch (Exception ex) {

            throw new IllegalStateException(ex);

        }

        return keyPair;

    }

}

**In security folder -> add Jwks.java -> below is the full code**

package org.studyeasy.SpringRestdemo.security;

import com.nimbusds.jose.jwk.RSAKey;

import java.security.KeyPair;

import java.security.interfaces.RSAPrivateKey;

import java.security.interfaces.RSAPublicKey;

import java.util.UUID;

public class Jwks {

    private Jwks() {}

    public static RSAKey generateRsa() {

        KeyPair keyPair = KeyGeneratorUtils.generateRsaKey();

        RSAPublicKey publicKey = (RSAPublicKey) keyPair.getPublic();

        RSAPrivateKey privateKey = (RSAPrivateKey) keyPair.getPrivate();

        return new RSAKey.Builder(publicKey)

                .privateKey(privateKey)

                .keyID(UUID.randomUUID().toString())

                .build();

    }

}

**Update SecurityConfig.java -> jwtDecoder() and jwtEncoder()**

 @Bean

 JwtDecoder jwtDecoder() {

     return NimbusJwtDecoder.withPublicKey(rsaKeys.publicKey()).build();

 }

@Bean

 JwtEncoder jwtEncoder() {

   JWK jwk = new RSAKey.Builder(rsaKeys.publicKey()).privateKey(rsaKeys.privateKey()).build();

   JWKSource<SecurityContext> jwks = new ImmutableJWKSet<>(new JWKSet(jwk));

   return new NimbusJwtEncoder(jwks);

}

**Update it to**

 @Bean

 JwtEncoder jwtEncoder(JWKSource<SecurityContext> jwks) {

    return new NimbusJwtEncoder(jwks);

 }

 @Bean

JwtDecoder jwtDecoder() throws JOSEException {

    return NimbusJwtDecoder.withPublicKey(rsaKey.toRSAPublicKey()).build();

 }

**This is a very secure mechanism for the JWT Tokens**