Documentation of JWS-SpringBoot

**JWS-SpringBoot Project**

**Overview**

This project demonstrates how to implement **JSON Web Signature (JWS)** in a Spring Boot application using **ES256 (ECDSA P-256 + SHA-256)**.  
All JSON API responses are signed to ensure **integrity** and **tamper detection**. A **JWKS endpoint** exposes the public key for verification.

**Features**

* Sign all JSON responses with ES256.
* Include fields in responses:  
  data, sig, kid, alg, ts.
* Provide a .well-known/jwks.json endpoint for public key distribution.
* Supports **client-side verification** using the public key.
* Demonstrates tamper detection in API responses.

**Prerequisites**

* Java 21 (recommended, LTS)
* Maven 3.9+
* VS Code or any IDE
* Postman (desktop or VS Code extension)
* Internet connection for Maven dependencies

**Project Structure**

jws-springboot/

├── src/

│ ├── main/java/com/example/jws/

│ │ ├── service/JwsSigningService.java # JWS signing logic

│ │ ├── controller/HelloController.java # Sample endpoint

│ │ └── util/VerifyExample.java # Verification utility

│ └── resources/

│ └── application.properties

├── pom.xml # Maven configuration

└── README.md

**Setup & Run**

**1. Clone / extract project**

git clone <repo-url>

cd jws-springboot

or extract the ZIP into a folder.

**2. Update JDK**

Ensure **Java 21** is active:

java -version

Expected:

openjdk version "21.x.x" ...

**3. Build the project**

mvn clean package

This downloads dependencies and compiles the project.

**4. Run the Spring Boot server**

mvn spring-boot:run

Server starts on:

http://localhost:8080

**API Endpoints**

**1. Get a signed response**

* **URL:** /api/hello
* **Method:** GET
* **Response example:**

{

"data": { "message": "Hello from JWS!", "timestamp": "2025-10-24T10:20:30Z" },

"sig": "MEUCIQCsdf...",

"kid": "5d88c8d2-9bdf-4c8a-8213-1234abcd5678",

"alg": "ES256",

"ts": "2025-10-24T10:20:30Z"

}

**2. Get JWKS (public key)**

* **URL:** /.well-known/jwks.json
* **Method:** GET
* **Response example:**

{

"keys": [

{

"kty": "EC",

"crv": "P-256",

"kid": "5d88c8d2-9bdf-4c8a-8213-1234abcd5678",

"x": "...",

"y": "...",

"use": "sig",

"alg": "ES256"

}

]

}

**Testing & Verification**

**1. Using Postman**

1. Open Postman (desktop or VS Code extension).
2. Create **GET request** → http://localhost:8080/api/hello.
3. Inspect the response:
   * data contains the payload.
   * sig contains the ES256 signature.
   * kid references the key in JWKS.
4. Fetch JWKS → http://localhost:8080/.well-known/jwks.json.

**2. Using VerifyExample.java**

# Save API response to a file

curl http://localhost:8080/api/hello -o resp.json

# Run verifier

java -cp target/jws-springboot-0.0.1-SNAPSHOT.jar com.example.jws.util.VerifyExample resp.json

* **Expected output:**

Verification result: true

* If you manually change data in resp.json, re-run → it prints:

Verification result: false

This demonstrates **tamper detection**.

**Dependencies**

* Spring Boot 3.2.6
* Nimbus JOSE + JWT 9.31
* Jackson Databind
* Jakarta Annotations API 2.1.1

**Notes**

* The project uses a dynamically generated P-256 EC key at startup.
* For production, store the keys in a **PKCS12 keystore** or secure vault.
* Always verify client-side to ensure response integrity.

**Shortcut Commands (Windows)**

* Open Postman: **Win → type Postman → Enter**
* Build project: mvn clean package
* Run server: mvn spring-boot:run

This README covers everything needed to **setup, run, test, and verify** your JWS-enabled Spring Boot app.