# Spring Boot Secure Property Storage (Encryption via Kubernetes Secrets)

## Overview

This document guides you through setting up a Spring Boot application that securely encrypts and decrypts sensitive properties using an AES secret key managed through Kubernetes Secrets. This is ideal for securing API keys, passwords, and other sensitive configurations.

## Step 1: Define Database Table

Create the `properties` table with columns to hold the property name, value, and whether it is sensitive.  
  
```sql  
CREATE TABLE properties (  
 id BIGINT AUTO\_INCREMENT PRIMARY KEY,  
 property\_name VARCHAR(255),  
 property\_value VARCHAR(4000),  
 is\_sensitive CHAR(1)  
);  
```

## Step 2: Create Kubernetes Secret

Create a Kubernetes secret that contains your AES encryption key.  
  
```bash  
kubectl create secret generic encryption-secret \  
 --from-literal=ENCRYPTION\_KEY=MyVerySecureKey123  
```

## Step 3: Mount Secret into Pod

In your Kubernetes Deployment YAML, add the following environment variable configuration:  
  
```yaml  
env:  
 - name: ENCRYPTION\_KEY  
 valueFrom:  
 secretKeyRef:  
 name: encryption-secret  
 key: ENCRYPTION\_KEY  
```

## Step 4: Encryption Utility in Spring Boot

Use this AES-based Java utility class to encrypt and decrypt values from the environment-based secret key.  
  
Refer to `EncryptionUtil.java` in the provided ZIP.

## Step 5: Property Service Logic

Service to insert and retrieve property values, encrypting only when marked sensitive.  
  
Refer to `PropertyService.java` in the provided ZIP.

## Step 6: Property Controller

Expose REST endpoints to save and retrieve properties securely.  
  
Refer to `PropertyController.java` in the provided ZIP.

## Step 7: Configuration and Schema

- Database: H2 in-memory  
- Configuration: `application.properties`  
- Schema: `schema.sql` for table creation  
  
These are included in the ZIP.

## Step 8: Running the Application

- Set the environment variable `ENCRYPTION\_KEY` before starting  
- Run the Spring Boot app as usual  
  
Use Postman or curl to test the `/api/properties/save` and `/api/properties/get` endpoints.

## Download Project

You can download the full Spring Boot project ZIP from the following link:  
  
[Download springboot-encryption-demo.zip](sandbox:/mnt/data/springboot-encryption-demo.zip)