**Giải thích về Mã hóa/Giải mã với AES (Bảo mật cao)**

**AES (Advanced Encryption Standard)** là một thuật toán mã hóa đối xứng (Symmetric Encryption), có nghĩa là cùng một khóa sẽ được sử dụng để **mã hóa (encrypt)** và **giải mã (decrypt)** dữ liệu. Đây là một trong những thuật toán mã hóa mạnh nhất hiện nay và được sử dụng rộng rãi trong bảo mật thông tin.

**1. Cơ chế hoạt động của AES**

AES hoạt động theo các bước sau:

**Bước 1: Sinh khóa bí mật (Secret Key)**

* AES sử dụng một khóa bí mật có độ dài 128-bit, 192-bit hoặc 256-bit.
* Trong Java, ta có thể dùng KeyGenerator để tạo khóa này.
* Khóa này cần được lưu trữ cẩn thận vì nếu mất khóa, dữ liệu đã mã hóa sẽ không thể giải mã được.

**Bước 2: Mã hóa dữ liệu**

* Văn bản gốc (plaintext) sẽ được chia thành các khối (block) có độ dài 128-bit.
* AES sử dụng nhiều vòng (rounds) mã hóa, tùy theo độ dài khóa:
  + **10 vòng** cho khóa 128-bit.
  + **12 vòng** cho khóa 192-bit.
  + **14 vòng** cho khóa 256-bit.
* Trong mỗi vòng, dữ liệu sẽ được biến đổi qua nhiều bước:
  + **SubBytes:** Thay thế mỗi byte bằng một giá trị khác dựa trên bảng S-Box.
  + **ShiftRows:** Dịch chuyển hàng trong ma trận dữ liệu.
  + **MixColumns:** Trộn dữ liệu giữa các cột.
  + **AddRoundKey:** Thêm khóa con (subkey) vào dữ liệu.
* Sau khi trải qua tất cả các vòng, ta thu được dữ liệu đã mã hóa (ciphertext).

**Bước 3: Giải mã dữ liệu**

* Quá trình giải mã là ngược lại của mã hóa.
* AES thực hiện các bước đảo ngược của mã hóa theo thứ tự:
  1. **Inverse AddRoundKey**
  2. **Inverse MixColumns**
  3. **Inverse ShiftRows**
  4. **Inverse SubBytes**
* Sau khi hoàn thành các vòng giải mã, dữ liệu gốc được khôi phục.

**2. Minh họa bằng code**

**Mã hóa AES**

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import java.util.Base64;

public class AESCipher {

private static final String ALGORITHM = "AES";

private static SecretKey secretKey;

static {

try {

KeyGenerator keyGenerator = KeyGenerator.getInstance(ALGORITHM);

keyGenerator.init(256); // 256-bit AES

secretKey = keyGenerator.generateKey();

} catch (Exception e) {

throw new RuntimeException("Error initializing AES key", e);

}

}

public static String encrypt(String input) {

try {

Cipher cipher = Cipher.getInstance(ALGORITHM);

cipher.init(Cipher.ENCRYPT\_MODE, secretKey);

byte[] encryptedBytes = cipher.doFinal(input.getBytes());

return Base64.getEncoder().encodeToString(encryptedBytes);

} catch (Exception e) {

throw new RuntimeException("Error encrypting", e);

}

}

public static String decrypt(String encryptedInput) {

try {

Cipher cipher = Cipher.getInstance(ALGORITHM);

cipher.init(Cipher.DECRYPT\_MODE, secretKey);

byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(encryptedInput));

return new String(decryptedBytes);

} catch (Exception e) {

throw new RuntimeException("Error decrypting", e);

}

}

public static void main(String[] args) {

String originalText = "Hello, AES!";

String encryptedText = encrypt(originalText);

String decryptedText = decrypt(encryptedText);

System.out.println("Original: " + originalText);

System.out.println("Encrypted: " + encryptedText);

System.out.println("Decrypted: " + decryptedText);

}

}

**3. Ưu điểm của AES**

✅ **Bảo mật cao**: AES được thiết kế để chống lại các cuộc tấn công brute-force.  
✅ **Tốc độ nhanh**: So với các thuật toán mã hóa đối xứng khác như DES, AES có hiệu suất cao hơn.  
✅ **Được chuẩn hóa**: AES là chuẩn mã hóa của chính phủ Hoa Kỳ và được sử dụng rộng rãi trong nhiều ứng dụng bảo mật.

**4. Điểm cần lưu ý khi sử dụng AES**

❌ **Không được mất khóa bí mật**: Nếu mất khóa, bạn không thể giải mã dữ liệu.  
❌ **AES không tự động cung cấp tính năng bảo vệ toàn vẹn dữ liệu**: Bạn cần kết hợp với HMAC để đảm bảo dữ liệu không bị sửa đổi.  
❌ **Không dùng AES cho lưu trữ mật khẩu**: AES phù hợp cho bảo vệ dữ liệu, nhưng mật khẩu nên được băm (hash) bằng **BCrypt, Argon2 hoặc PBKDF2** thay vì mã hóa.

**5. Khi nào nên dùng AES?**

🔹 Khi bạn cần mã hóa dữ liệu như **file, token, JSON, nội dung tin nhắn**.  
🔹 Khi cần bảo vệ dữ liệu tạm thời và có thể quản lý khóa an toàn.  
🔹 Khi cần mã hóa dữ liệu trong **cơ sở dữ liệu hoặc lưu trữ trên đám mây**.

Bạn có câu hỏi nào thêm về AES không? 🚀