

Bộ Giáo Dục Và Đào Tạo
Trường Đại Học Ngoại Ngữ - Tin Học Thành Phố Hồ Chí Minh
Khoa Công Nghệ Thông Tin



MÔN HỌC : LẬP TRÌNH MẠNG NÂNG CAO

ĐỀ TÀI : CHƯƠNG TRÌNH QUÉT SUBDOMAIN CÓ XÁC THỰC

Giáo Viên Hướng Dẫn : Phan Gia Lượng

Thành Viên :

1. Trần Quang Khải – MSSV: 22DH114583

Tp. Hồ chí minh, Ngày tháng năm 2025

Lời cảm ơn

Nhận xét của giảng viên

Mục lục

I GIỚI THIỆU	6
1. Giới thiệu chung.....	6
2. Mục tiêu.....	6
II CƠ SỞ LÝ THUYẾT	7
1. Mã hóa bất đối xứng RSA.....	7
a. Khái niệm.....	7
b. Nguyên lý hoạt động	7
2. Mã hóa đối xứng AES-CBC.....	7
a. Khái niệm.....	7
a1. AES (Advanced Encryption Standard)	7
a2. CBC (Cipher Block Chaining) là gì?	8
b. Nguyên lý hoạt động	8
3. Chữ ký số (SHA256withRSA)	8
a. Khái niệm.....	8
a1. Chữ ký số là gì?.....	8
a2. SHA256withRSA là gì?.....	8
b. Nguyên lý hoạt động	8
III Triển khai.....	10
1. Khởi tạo chữ ký số.....	10
1.1 Tạo private_key.pem và public_key.pem	10
1.2 Tiến hành tạo chữ ký số SHA256withRSA	11
2. Mã hóa publickey bằng AES mode CBC	11
2.1 Tạo khóa AES	11
2.2 Tạo IV key	11
2.3 Mã hóa publickey	12
3. Tạo json	12
4. Giải mã thông tin	12
4.1 Tiếp nhận thông tin	12
4.2 Giải mã publickey	13
4.3 Xác minh chữ ký số.....	13

5. Scan subdomain	13
6. Tính năng thêm	14
6.1 Xác thực UUID	14
6.2 Bảo mật TLS	16
Kết luận	20

Danh mục hình ảnh

Hình 1 Lệnh tạo key.....	10
Hình 2 private_key.pem.....	10
Hình 3 public_key.pem.....	11
Hình 4 SHA256withRSA.....	11
Hình 5 createAES().....	11
Hình 6 createIv().....	12
Hình 7 Mã hóa pubkey 12	12
Hình 8 Đóng gói dữ liệu 12	12
Hình 9 Tiếp nhận, giải mã thông tin..... 13	13
Hình 10 decodePublickey()..... 13	13
Hình 11 verifySignature() 13	13
Hình 12 Scan subdomain 14	14
Hình 13 Kiểm tra uuid có tồn tại và lấy giá trị uuid..... 14	14
Hình 14 Khởi tạo UUID..... 14	14
Hình 15 Lưu AES và IV vào file..... 15	15
Hình 16 Save uuid, aes, iv vào database..... 15	15
Hình 17 Load file..... 16	16
Hình 18 Load aes, iv key 16	16
Hình 19 Lệnh khởi tạo server.crt và server.key 16	16
Hình 20 server.crt 17	17
Hình 21 server.key 18	18
Hình 22 pipeline 19	19
Hình 23 pipeLine 19	19

I GIỚI THIỆU

1. Giới thiệu chung

Trong các hệ thống mạng hiện đại, việc đảm bảo an toàn khi trao đổi dữ liệu giữa client và server là rất quan trọng. Nếu không có cơ chế xác thực và mã hóa, dữ liệu có thể bị giả mạo, chỉnh sửa hoặc đánh cắp trong quá trình truyền.

Đề tài này xây dựng một mô hình **Client-Server có xác thực**, trong đó client sẽ gửi yêu cầu có mã hóa và chữ ký số để chứng minh danh tính. Nếu server xác thực thành công, nó sẽ thực hiện một tác vụ cụ thể — **quét subdomain của một tên miền** — và trả kết quả về cho client. Nếu xác thực không thành công, server sẽ từ chối xử lý.

Hệ thống kết hợp giữa **mã hóa bất đối xứng RSA** (để xác thực) và **mã hóa đối xứng AES-CBC** (để bảo vệ dữ liệu), giúp đảm bảo tính bảo mật, toàn vẹn và chính xác trong quá trình giao tiếp.

2. Mục tiêu

- Xây dựng một hệ thống client-server có xác thực thông qua chữ ký số.
- Sử dụng kết hợp mã hóa RSA và AES để đảm bảo an toàn thông tin.
- Server chỉ xử lý yêu cầu từ client hợp lệ.
- Sau khi xác thực, server thực hiện quét subdomain từ danh sách có sẵn.
- Tăng cường tính bảo mật và hiệu quả trong quá trình trao đổi dữ liệu và xử lý thông tin.

II CƠ SỞ LÝ THUYẾT

1. Mã hóa bất đối xứng RSA

a. Khái niệm

- **Mã hóa bất đối xứng:** Là phương pháp mã hóa trong đó sử dụng **hai khóa khác nhau**:
 - **Khóa công khai (Public Key):** Dùng để mã hóa dữ liệu, có thể chia sẻ công khai.
 - **Khóa riêng (Private Key):** Dùng để giải mã dữ liệu, **phải được giữ bí mật**.
- **RSA (Rivest–Shamir–Adleman)** là một trong những thuật toán mã hóa bất đối xứng phổ biến và lâu đời nhất.

b. Nguyên lý hoạt động

- Khi một người muốn gửi dữ liệu an toàn, họ dùng **public key** của người nhận để mã hóa.
- Người nhận dùng **private key** của mình để giải mã.
- Trong chữ ký số:
 - Người gửi dùng **private key** để ký dữ liệu.
 - Người nhận dùng **public key** để kiểm tra chữ ký đó có đúng không.

2. Mã hóa đối xứng AES-CBC

a. Khái niệm

Mã hóa đối xứng là một kỹ thuật **mã hóa dữ liệu trong đó cùng một khóa (key)** được sử dụng để **mã hóa** và **giải mã** thông tin. Điều đó có nghĩa là cả người gửi và người nhận đều phải biết và giữ bí mật cùng một khóa.

a1. AES (Advanced Encryption Standard)

AES là **tiêu chuẩn mã hóa tiên tiến**, được NIST (Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ) công nhận từ năm 2001. AES là một thuật toán mã hóa **dạng khối** (block cipher), nghĩa là nó xử lý dữ liệu theo từng **khối có độ dài cố định**, thường là **128 bit (16 byte)** mỗi khối.

- **Độ dài khóa** có thể là:
 - 128 bit (AES-128)
 - 192 bit (AES-192)

- 256 bit (AES-256)

AES rất mạnh và hiện vẫn được dùng phổ biến trong các hệ thống bảo mật.

a2. CBC (Cipher Block Chaining) là gì?

CBC là một **chế độ hoạt động** (mode of operation) được dùng kèm với các thuật toán mã hóa khối như AES. Mục tiêu của CBC là **ngăn chặn các mẫu (pattern)** bị lộ ra khi dữ liệu đầu vào có tính lặp lại.

Trong CBC, mỗi khối dữ liệu **trước khi mã hóa sẽ được XOR với khối mã hóa trước đó**, tạo ra **sự phụ thuộc giữa các khối**, do đó tăng cường bảo mật.

b. Nguyên lý hoạt động

- Dữ liệu được chia thành từng khối nhỏ.
- Mỗi khối sẽ **kết hợp với khối trước đó** (qua phép XOR), rồi mới mã hóa.
- Khối đầu tiên kết hợp với **IV (chuỗi khởi tạo)**.
- Khi giải mã, quá trình được làm ngược lại.

3. Chữ ký số (SHA256withRSA)

a. Khái niệm

a1. Chữ ký số là gì?

Chữ ký số (digital signature) là một đoạn dữ liệu được tạo ra từ một thông điệp gốc, dùng để xác minh:

- Ai là người gửi (xác thực danh tính)
- Nội dung có bị thay đổi không (tính toàn vẹn)
- Người gửi không thể chối bỏ đã ký (tính chống chối bỏ)

a2. SHA256withRSA là gì?

SHA256withRSA là một thuật toán kết hợp giữa:

- **SHA-256**: dùng để tạo ra bản tóm tắt (hash) của nội dung cần ký.
- **RSA**: dùng để mã hóa (ký) bản tóm tắt bằng khóa riêng (private key) và xác minh bằng khóa công khai (public key).

Vì vậy, SHA256withRSA là một phương pháp chữ ký số sử dụng thuật toán băm SHA-256 kết hợp với thuật toán mã hóa bất đối xứng RSA.

b. Nguyên lý hoạt động

- Client dùng thuật toán SHA-256 để tạo ra một đoạn mã ngắn từ nội dung gửi (gọi là hash).

- Sau đó dùng **private key RSA** để mã hóa đoạn hash đó → tạo thành chữ ký số.
- Server sẽ:
 - Dùng **public key** để giải mã chữ ký → lấy lại đoạn hash.
 - Tự tạo hash mới từ nội dung message.
 - So sánh 2 hash: nếu giống → chữ ký đúng.

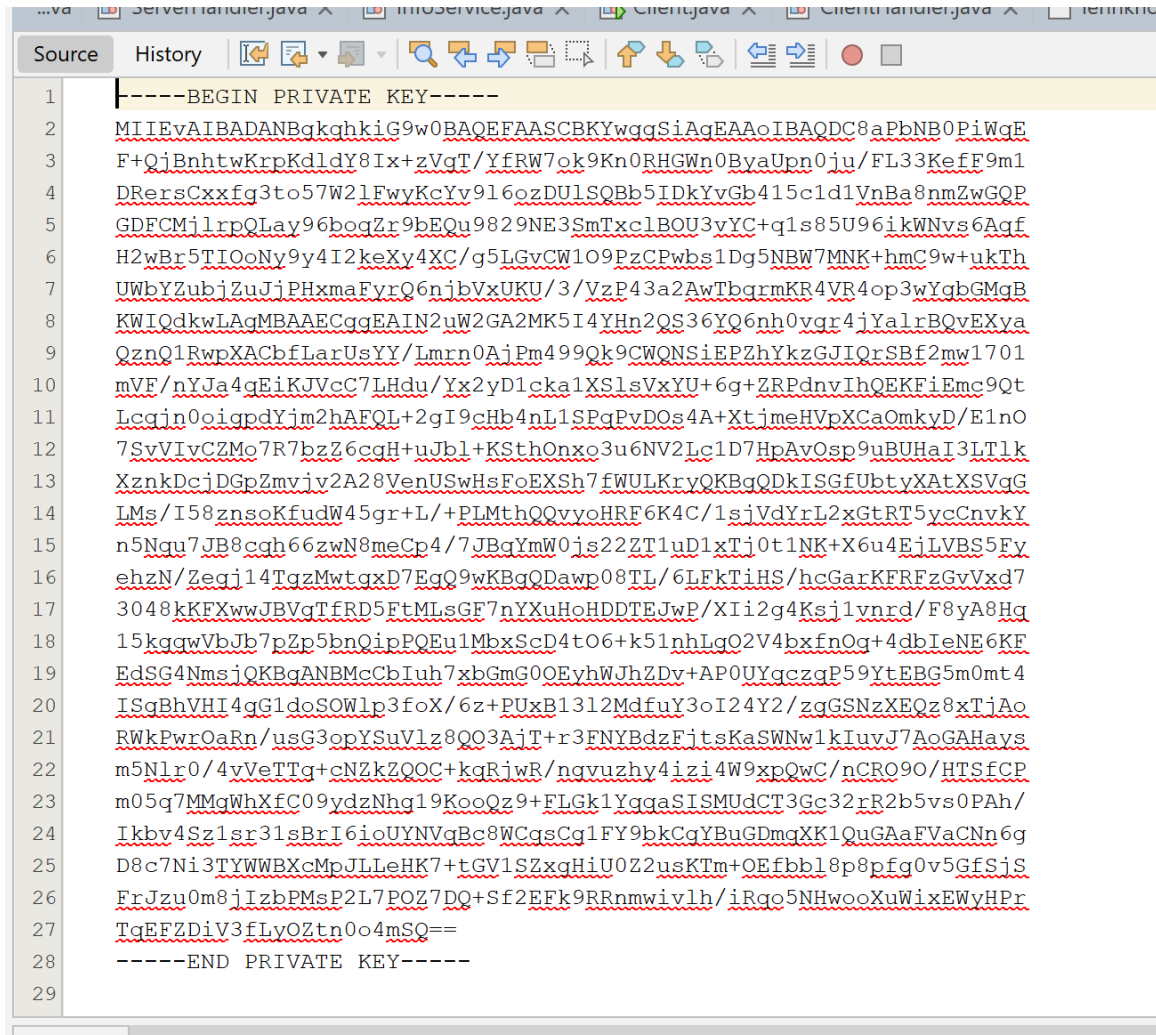
III Triển khai

1. Khởi tạo chữ ký số

1.1 Tạo private_key.pem và public_key.pem

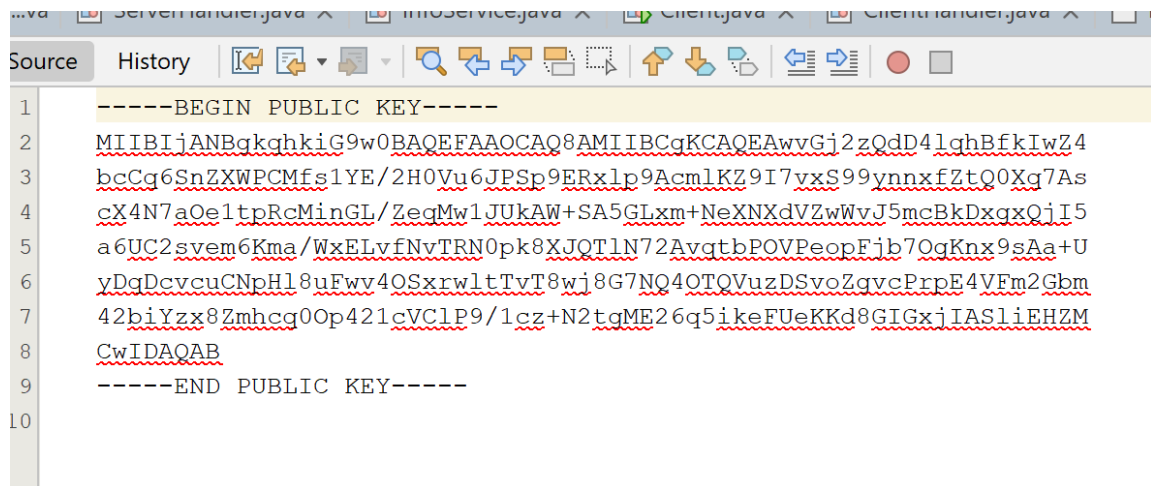
```
#Lệnh khởi tạo private_key và public_key  
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048  
openssl rsa -in private_key.pem -pubout -out public_key.pem
```

Hình 1 Lệnh tạo key



```
1  -----BEGIN PRIVATE KEY-----  
2  MIEFvAIBADANBqkqhkiG9w0BAQEFFAASCBKYwggSiAgEAAoIBAODC8aPbNB0PiWqE  
3  F+QjBnhtwKrpKdldY8Ix+zVqT/YfRW7ok9Kn0RHGWn0ByaUpn0ju/FL33Keff9m1  
4  DRersCxxfg3to57W2lFwyKcYv9l6ozDULSOBb5IDkYvGb415c1d1VnBa8nmZwGQP  
5  GDFCMilrpQLay96boqZr9bEQ9829NE3SmTxclBOU3vYC+q1s85U96ikWNvs6Aqf  
6  H2wBr5TIOoNy9y4I2keXy4XC/g5LGvCW1O9PzCPwbs1Dg5NBW7MNK+hmC9w+ukTh  
7  UWbYZubjZuJiPHxmaFyrQ6nibVxUKU/3/VzP43a2AwTbgrmKR4VR4op3wYqbGMqB  
8  KWIQdkwLAqMBAECggEAIn2uW2GA2MK5I4YHn2QS36YQ6nh0vqr4iYalrBQvEXYa  
9  QznQ1RwpXACbfLarUsYY/Lmrn0AiPm499Qk9CWONSiEPZhYkzGJIQrSBf2mw1701  
10 mVF/nYJa4qEiKJVC7LHdu/Yx2yD1cka1XSlsVxYU+6g+ZRPdnvIhOEKFiEmc9Qt  
11 Lcqnj0oigpdYjm2hAFOL+2gI9cHb4nL1SPqPvDOs4A+XtimeHVpXCaOmkyD/E1nO  
12 7svViVcZMo7R7bzZ6cgH+uJbl+KStHOnxo3u6NV2Lc1D7HpAvOsp9uBUHaI3Ltlk  
13 XznkDcijDGPzmviv2A28VenUSWhsFoEXSh7fWULKrvOKBgODkISGfUbtvXATXSVqG  
14 LMs/I58znsoKfudW45gr+L/+PLMthQOvyoHRF6K4C/1sjVdYrL2xGtRT5ycCnvkY  
15 n5Nqu7JB8cqh66zwN8meCp4/7JBqYmW0is22ZT1uD1xTj0t1NK+X6u4EjLVBS5Fy  
16 ehzN/Zegj14TqzMwtgxD7EqQ9wKBQDawp08TL/6LFkTiHS/hcGarKFRFzGvVxd7  
17 3048kKFXwwJBVqTfRD5fTMLSgf7nYXuHoHDDTEJwP/XIi2g4Ks1vnrd/F8yA8Hq  
18 15kqgwVbJb7pZp5bnOipQOEu1MbxScD4tO6+k51nhLqO2V4bxfnOg+4dbIeNE6KF  
19 EdSG4NmsjOKBgANBMcCbIuh7xbGmG0OEyhwJhZDv+AP0UYgczaP59YtEBG5m0mt4  
20 ISqBhVHI4gG1doSOWlp3foX/6z+PUxB1312MdfuY3oI24Y2/zqGSNzXEQz8xTiAo  
21 RWkPwrOaRn/usG3opYSuVlz8QO3AiT+r3FNYBdzFitsKaSWNw1kIuvJ7AoGAHays  
22 m5Nlr0/4vVeTTg+cNZkZQOC+kqRiwr/ngvuzhy4izi4W9xpQwC/nCRO9O/HTSfCP  
23 m05q7MMgWhXfc09ydzNhq19KooQz9+FLGk1YqgaSISMUDCT3Gc32rR2b5vs0PAh/  
24 Ikbv4Sz1sr31sBrI6ioUYNVqBc8WCqsCg1FY9bkCqYBuGDmqXK1QuGAaFVaCnN6g  
25 D8c7Ni3TYWWBXcMpJLLeHK7+tGV1SZxgHiU0Z2usKTm+Oefbbl8p8pfg0v5GfSjS  
26 FrJzu0m8jIzbPMsP2L7POZ7DO+Sf2EFk9RRnmwivlh/iRgo5NHwoXUWixEWyHPr  
27 TqEFZDiV3fLyOZtn0o4mSQ==  
28 -----END PRIVATE KEY-----  
29
```

Hình 2 private_key.pem



```

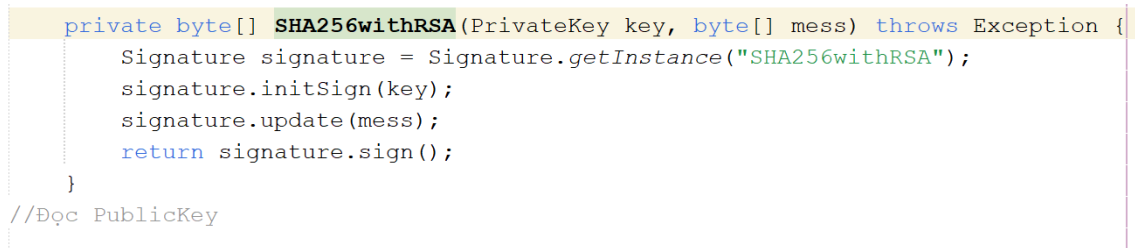
1  -----BEGIN PUBLIC KEY-----
2  MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwwGj2zOdD4lqhBfkIwZ4
3  bcCq6SnZXWPCMFs1YE/2H0Vu6JPSP9ERxlp9AcmlKZ9I7vxS99ynnxfZtQ0Xq7As
4  cX4N7aOe1tpRcMinGL/ZeqMw1JUKAW+SA5GLxm+NeXNXdVZwWvJ5mcBkDxgxQjI5
5  a6UC2svem6Kma/WxELvfNvTRN0pk8XJQT1N72AvqtbPOVPeopFjb7OgKnxsAa+U
6  yDqDcvCuCNpHl8uFwv4OSxrwltTvT8wj8G7NQ40TQVuzDSvoZqvcPrpE4VFm2Gbm
7  42biYzx8Zmhcg0Op421cVC1P9/1cz+N2tgME26q5ikeFUeKKd8GIGxjIASliEHZM
8  CwIDAQAB
9  -----END PUBLIC KEY-----
10

```

Hình 3 public_key.pem

1.2 Tiến hành tạo chữ ký số SHA256withRSA

Sử dụng message được người dùng nhập kèm với nội dung private_key.pem để tiến hành tạo chữ ký số



```

private byte[] SHA256withRSA(PrivateKey key, byte[] mess) throws Exception {
    Signature signature = Signature.getInstance("SHA256withRSA");
    signature.initSign(key);
    signature.update(mess);
    return signature.sign();
}
//Đọc PublicKey

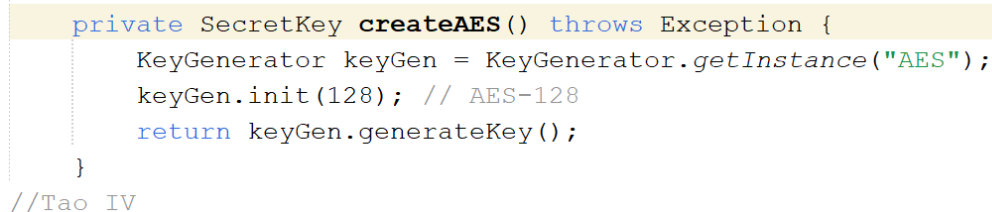
```

Hình 4 SHA256withRSA

2. Mã hóa publickey bằng AES mode CBC

2.1 Tạo khóa AES

Khởi tạo với thuật toán AES và độ dài 128 bit



```

private SecretKey createAES() throws Exception {
    KeyGenerator keyGen = KeyGenerator.getInstance("AES");
    keyGen.init(128); // AES-128
    return keyGen.generateKey();
}
//Tạo IV

```

Hình 5 createAES()

2.2 Tạo IV key

Khởi tạo một vector khởi tạo (IV) dài 16 byte ngẫu nhiên

```

private IvParameterSpec createIV() {
    byte[] iv = new byte[16];
    SecureRandom random = new SecureRandom();
    random.nextBytes(iv);
    return new IvParameterSpec(iv);
}
Mã hóa public key bằng AES-CBC

```

Hình 6 createIV()

2.3 Mã hóa publickey

Mã hóa mảng byte key bằng thuật toán AES ở chế độ CBC với padding PKCS5, sử dụng khóa aes và vector khởi tạo iv.

```

// Mã hóa public key bằng AES-CBC

private byte[] encodePublicKey(SecretKey aes, IvParameterSpec iv, byte[] key) throws Exception {
    Cipher aesCipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    aesCipher.init(Cipher.ENCRYPT_MODE, aes, iv);
    return aesCipher.doFinal(key);
}
//Tạo JSON object

```

Hình 7 Mã hóa pubkeykey

3. Tạo json

Tạo một đối tượng JSON chứa thông tin: uuid, raw_message, chữ ký, khóa công khai đã mã hóa, và (nếu có) khóa AES cùng IV – tất cả được mã hóa dưới dạng Base64.

```

//Tạo JSON object

private JSONObject getJson(String mess, String uuid, byte[] sign, byte[] key, byte[] aes, byte[] iv) throws Exception {
    JSONObject packet = new JSONObject();
    packet.put("uuid", uuid);
    packet.put("raw_message", mess);
    packet.put("signature", Base64.getEncoder().encodeToString(sign));
    packet.put("encodePuclickey", Base64.getEncoder().encodeToString(key));
    if (aes != null) {
        packet.put("aes_key", Base64.getEncoder().encodeToString(aes));
    }
    if (iv != null) {
        packet.put("iv_key", Base64.getEncoder().encodeToString(iv));
    }
    return packet;
}

```

Hình 8 Đóng gói dữ liệu

4. Giải mã thông tin

4.1 Tiếp nhận thông tin

Server nhận các thông tin uuid, raw-message, các thông tin khác như sign, publickey, aes và iv thì cần decode base 64 để tiếp nhận.

```

JSONObject packet = new JSONObject(msg);
String uuid = packet.getString("uuid");
String mess = packet.getString("raw_message");
byte[] sign = Base64.getDecoder().decode(packet.getString("signature"));
byte[] encodePublicKey = Base64.getDecoder().decode(packet.getString("encodePuclickey"));
byte[] aes_byte, iv_byte;
InfoEntity info = infoService.getInfoByUuid(uuid);
if (info != null) {
    aes_byte = info.getAes();
    iv_byte = info.getIv();
} else {
    aes_byte = Base64.getDecoder().decode(packet.getString("aes_key"));
    iv_byte = Base64.getDecoder().decode(packet.getString("iv_key"));
    boolean saved = infoService.saveIfNotExist(uuid, aes_byte, iv_byte);
    if (saved) {
        System.out.println("Saved new UUID");
    } else {
        System.out.println("UUID already exists");
    }
}
}

```

Hình 9 Tiếp nhận, giải mã thông tin

4.2 Giải mã publickey

Giải mã dữ liệu key đã được mã hóa bằng AES/CBC/PKCS5Padding, sử dụng khóa bí mật aes và vector khởi tạo iv.

```

}

private byte[] decodePublicKey(byte[] key, byte[] aes, byte[] iv) throws Exception {
    SecretKey aesKey = new SecretKeySpec(aes, "AES");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    Cipher aesCipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    aesCipher.init(Cipher.DECRYPT_MODE, aesKey, ivSpec);
    return aesCipher.doFinal(key);
}

```

Hình 10 decodePublicKey()

4.3 Xác minh chữ ký số

Xác minh chữ ký số signature của thông điệp mess bằng khóa công khai pubKey với thuật toán SHA256withRSA.

```

private boolean verifySignature(PublicKey pubKey, String mess, byte[] signature) throws Exception {
    Signature signatureVerifier = Signature.getInstance("SHA256withRSA");
    signatureVerifier.initVerify(pubKey);
    signatureVerifier.update(mess.getBytes());
    return signatureVerifier.verify(signature);
}
//Scan Subdomain

```

Hình 11 verifySignature()

5. Scan subdomain

Quét danh sách các subdomain của domain gốc bằng cách đọc từng dòng trong file list, sau đó thử phân giải DNS cho từng subdomain và gửi kết quả hợp lệ qua ctx.

```

public static void subdomainScan(String domain, String list, ChannelHandlerContext ctx) {
    ctx.writeAndFlush("*** Danh sách subdomain của \"" + domain + "\" ***");
    try (BufferedReader br = new BufferedReader(new FileReader(list))) {
        String sub;
        while ((sub = br.readLine()) != null) {
            String fullDomain = sub.trim() + "." + domain;

            try {
                InetAddress inetAddress = InetAddress.getByName(fullDomain);
                String result = " - " + fullDomain;
                ctx.writeAndFlush(result);
            } catch (UnknownHostException e) {
                //Subdomain không tồn tại
            }
        }
    } catch (IOException e) {
        System.err.println("Lỗi khi đọc file: " + e.getMessage());
    }
}

```

Hình 12 Scan subdomain

6. Tính năng thêm

6.1 Xác thực UUID

Kiểm tra sự tồn tại của uuid trên máy tính của client.

```

private String getUUID() {
    if (UUIDManager.uuidFileExists()) {
        return UUIDManager.readUUIDFromFile();
    }
    return null;
}

```

Hình 13 Kiểm tra uuid có tồn tại và lấy giá trị uuid

❖ Nếu không tồn tại:

CLIENT

Khởi tạo uuid và lưu vào máy của client.

```

private static final String FILE_NAME = ".my_uuid"; // tên file UUID
private static final String FILE_PATH = System.getProperty("user.home") + File.separator + FILE_NAME

// Hàm tạo UUID và lưu vào file
public static void createUUID() {
    String uuid = UUID.randomUUID().toString();
    try (FileWriter writer = new FileWriter(FILE_PATH)) {
        writer.write(uuid);
    } catch (IOException e) {
        e.getMessage();
    }
}

```

Hình 14 Khởi tạo UUID

Khởi tạo AES và IV key rồi cũng lưu vào máy client.

```

//Luu data vao file
private static void saveToFile(String fileName, byte[] data) throws Exception {
    String base64 = Base64.getEncoder().encodeToString(data);
    Path filePath = Paths.get(HOME_DIR, fileName);
    Files.write(filePath, base64.getBytes());
}

//Luu AES va IV

public static void saveAESKeyAndIV(byte[] aesKeyBytes, byte[] ivBytes) throws Exception {
    saveToFile("aes.key", aesKeyBytes);
    saveToFile("iv.key", ivBytes);
}

```

Hình 15 Lưu AES và IV vào file

SERVER

Nhận các thông tin uuid, aes, iv lưu vào database.

```

public boolean saveIfNotExist(String uuid, byte[] aes, byte[] iv) {
    if (infoRepository.existsByUuid(uuid)) {
        return false;
    }

    InfoEntity info = new InfoEntity();
    info.setUuid(uuid);
    info.setAes(aes);
    info.setIv(iv);

    infoRepository.save(info);
    return true;
}

```

Hình 16 Save uuid, aes, iv vào database

Sau đó server sẽ dùng chính các aes và iv key này để giải mã signature.

❖ Nếu tồn tại:

CLIENT

Load các key aes và iv từ file trên máy client.


```
//Doc data tu file
private static byte[] readFromFile(String fileName) throws Exception {
    Path filePath = Paths.get(HOME_DIR, fileName);
    String base64 = Files.readString(filePath);
    return Base64.getDecoder().decode(base64);
}

//Load AES key từ file và tạo lại SecretKey
public static SecretKey loadAESKey() throws Exception {
    byte[] keyBytes = readFromFile("aes.key");
    return new SecretKeySpec(keyBytes, "AES");
}

// Load IV từ file và tạo lại IvParameterSpec
public static IvParameterSpec loadIV() throws Exception {
    byte[] ivBytes = readFromFile("iv.key");
    return new IvParameterSpec(ivBytes);
}
```

Hình 17 Load file

Dùng các aes và iv key của load được để mã hóa thông tin.

SERVER

Dựa vào uuid để load aes và iv key từ database sau đó dùng chúng để giải mã dữ liệu

```
InfoEntity info = infoService.getInfoByUuid(uuid);
if (info != null) {
    aes_byte = info.getAes();
    iv_byte = info.getIv();
} else {
```

Hình 18 Load aes, iv key

6.2 Bảo mật TLS

Khởi tạo server.crt và server.key

```
#Lệnh khởi tạo server.key và server.crt
openssl req -x509 -newkey rsa:2048 -keyout server.key -out server.crt -days 365 -nodes
```

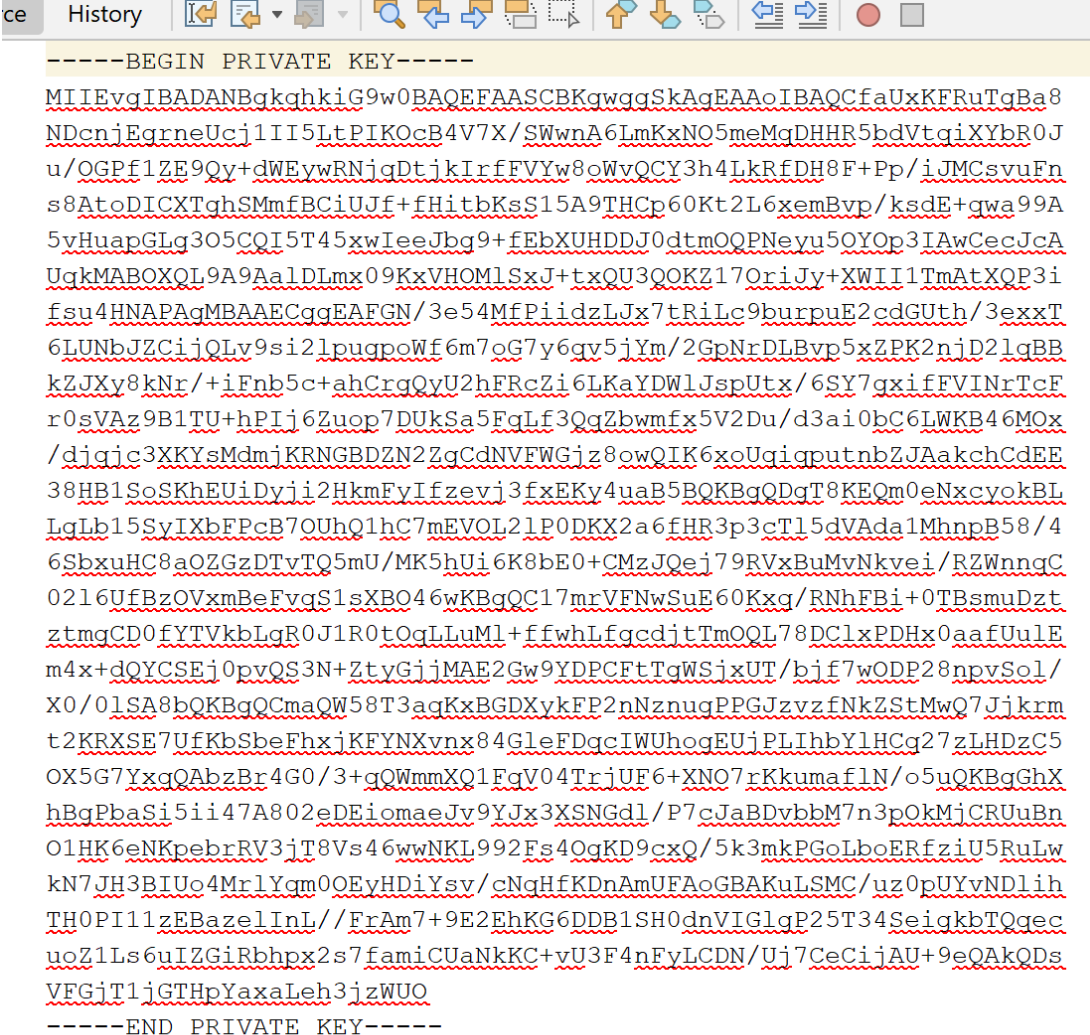
Hình 19 Lệnh khởi tạo server.crt và server.key

-----BEGIN CERTIFICATE-----

MIIDazCCAlOqAwIBAgIULz3qxBR5XdmhdbJTQb7x99Jv/IwwDQYJKoZIhvcNAQEL
BQAwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgMC1NvbWUtU3RhdGUxITAfBgNVBAoM
GEludGVybmV0IFdpZGdpdHMqUHR5IEEx0ZDAeFw0yNTA4MDEwODIxMjNaFw0yNjA4
MDEwODIxMjNaMEUxCzAJBgNVBAYTAkFVMRMwEQYDVOQIDApTb211LVN0YXRlMSEw
HwYDVQOKDBhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQwggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEKAoIBAQCfaUxKFRuTgBa8NDcnjEgrneUcj1II5LtPIK0cB4V7
X/SWwnA6LmKxNO5meMqDHHR5bdVtqiXYbR0Ju/OGPf1ZE9Qy+dWEyWRNjgDtjkIr
fFVYw8oWvOCY3h4LkRfDH8F+Pp/iJMCsvuFns8AtoDICXTghSMmfBCiUJf+fHitb
KsS15A9THCp60Kt2L6xemBvp/ksdE+qwa99A5vHuapGLg3O5CQI5T45xwIeeJbg9
+fEbXUHDDJ0dtmOQPNeyu5OYOp3IAwCecJcAUqkMABOXQL9A9AalDLmx09KxVHOM
lSxJ+txQU3QOKZ17OriJy+XWII1TmAtXQP3ifsu4HNAPAgMBAAGjUzBRMB0GA1Ud
DgQWBBO8hR9k44FwceftLnFYmwLvQUOQCtAFBgNVHSMEGDAWgBQ8hR9k44Fwceft
LnFYmwLvQUOQCtAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBGwUAA4IBAQAU
nUX3nOAWS7OF6cU6RT2DWvxvFabb7T6MWQVT2FxH+9uL3hiPqFbhqsk67UwuTXuB
/tNETCBBKfAqd9ErQ4ZeaR0YYMx5F8xrPqHLCpjnz1ULE8lqk/DUm87NsScA7b9C
RcDDDWwZa1omMJ/D+eoJ9ebX6xRYSeUX7HWAJ/fb5qtPzSPb5AENFZFBCaiLeipj
19jWCWRuFM18uAJnT7fbcKd7DsYb8xAYxRNNlHMHhi5CJHr+/LsDLT6E8Rzl1ga3
0EPNfpcRaM32pQXGPHBBv0GKc1TkyuoPTRYMkf+IcOleCkiKq32L2jUvXGHYuDxQ
XvRoUCYJzoJWVjqcD6JL

-----END CERTIFICATE-----

Hinh 20 server.crt



```

-----BEGIN PRIVATE KEY-----
MIIEVgIBADANBgkqhkiG9w0BAQEFFAASCBKQwwgSkAgFAAoIBAQCfaUxKFRuTqBa8
NDcnjEgrneUcj1II5LtPIKOb4V7X/SWwnA6LmKxNO5meMqDHRH5bdVtqiXYbR0J
u/OGPf1ZE9Qy+dWEyWRNjqDtjkIrfFVYw8oWvQCY3h4LkRfDH8F+Pp/iJMCsvuFn
s8AtoDICXTghSMmfBCiUJf+fHitbKsS15A9THCp60Kt2L6xemBvp/ksdE+qwa99A
5vHuapGLq3O5CQI5T45xwLeeJbg9+fEbXUHDDJ0dtmOQPNeYu5OYOp3IAwCecJcA
UqkMABOXQL9A9AalDLmx09KxVHOM1SxJ+txQU3QOKZ17OriJy+XWII1TmAtXQP3i
fsu4HNAPAgMBAAECggEAFGN/3e54MfPiidzLJx7tRiLc9burpuE2cdGUth/3exxT
6LUNbJZCijQLv9si2lpugpoWf6m7oG7y6qv5jYm/2GpNrDLBvp5xZPK2njD2lqBB
kZJXy8kNr/+iFnb5c+ahCrgQyU2hFRcZi6LKAYDWLJspUtx/6SY7qxifFVINrTcF
r0sVAz9B1TU+hPIj6Zuop7DUkSa5FqLf3QqZbwmfx5V2Du/d3ai0bC6LWKB46MOx
/djqjc3XKYsMdmjKRNGBDZN2ZqCdNVFWGjz8owQIK6xoUqigputnbZJAakchCdEE
38HB1SoSKhEUIDyji2HkmFyIfzevj3fxEKy4uaB5BQKBgQDqT8KEQm0eNxcyokBL
LqLb15SyIXbFPcB7OUhQ1hC7mEVOL2lP0DKX2a6fHR3p3cTl5dVAda1MhnpB58/4
6SbxuHC8aOZGzDTvTQ5mU/MK5hUi6K8bE0+CMzJQej79RVxBuMvNkvei/RZWnnqC
02l6UfBzOVxmBeFvqS1sXBO46wKBgQC17mrVFNwSuE60Kxq/RNhFbi+0TBsmuDzt
ztmgCD0fYTVkbLqR0J1R0tOqLLuMl+ffwhLfgcdjtTmOQL78DC1xPDHx0aafUulE
m4x+dQYCSEj0pvQS3N+ZtyGjjMAE2Gw9YDPCFtTqWSjxUT/bjf7wODP28npvSol/
X0/0lSA8bQKBgQCmaQW58T3aqKxBGDYkFP2nNznugPPGJzvfNkZStMwQ7Jikrm
t2KRXSE7UfKbSbeFhxjKFYNXvnx84GleFDqcIWUhogEUjPLIhbY1HCq27zLHDzC5
OX5G7YxgQAbzBr4G0/3+qQWmmXQ1FqV04TrjUF6+XNO7rKkumafLN/o5uQKBgGhX
hBgPbaSi5ii47A802eDEiomaEJv9YJx3XSNGdl/P7cJaBDvbbM7n3pOkMjCRUuBn
0lHK6eNKpebrRV3jT8Vs46wwNKL992Fs4OqKD9cxQ/5k3mkPGolBoERfziU5RuLw
kn7JH3BIUo4MrlYgm0OEyHdiYsv/cNgHfKDNAmUFAoGBAKuLSMC/uz0pUYvNDlih
TH0PI11zEBazelInL//FrAm7+9E2EhKG6DDB1SH0dnVIGlgP25T34SeigkbTQqec
uoZ1Ls6uIZGiRbhpx2s7famiCUaNkKC+vU3F4nFyLCDN/Uj7CeCijAU+9eQAKQDs
VFGjT1jGTHpYaxaLeh3jzWUO
-----END PRIVATE KEY-----

```

Hình 21 server.key

SERVER

Tiến hành sử dụng SslContent để tạo một SslContext để cấu hình SSL/TLS cho server, sử dụng cặp key và cert này.

```

@Override
protected void initChannel(SocketChannel ch) throws Exception {
    File cert = new File("server.crt");
    File key = new File("server.key");

    SslContext sslContext = SslContextBuilder.forServer(cert, key).build();

    ChannelPipeline pipeline = ch.pipeline();
    pipeline.addFirst(sslContext.newHandler(ch.alloc()));
    pipeline.addLast(new StringDecoder());
    pipeline.addLast(new StringEncoder());
    pipeline.addLast(new ServerHandler(infoService));
}

```

Hình 22 pipeline

CLIENT

Sau đó client chấp nhận mọi chứng chỉ, kể cả tự ký từ phía server

```

/**
public class ClientInitializer extends ChannelInitializer<SocketChannel> {
    private final String host;
    private final int port;

    public ClientInitializer(String host, int port) {
        this.host = host;
        this.port = port;
    }

    @Override
    protected void initChannel(SocketChannel ch) throws SSLException {
        SslContext sslContext = SslContextBuilder.forClient()
            .trustManager(InsecureTrustManagerFactory.INSTANCE) // chấp nhận cert tự ký (DEV only)
            .build();

        ChannelPipeline pipeline = ch.pipeline();
        pipeline.addFirst(sslContext.newHandler(ch.alloc(), host, port));
        pipeline.addLast(new StringDecoder());
        pipeline.addLast(new StringEncoder());
        pipeline.addLast(new ClientHandler());
    }
}

```

Hình 23 pipeLine

TLS đảm bảo mọi dữ liệu trao đổi giữa client và server đều được mã hóa, giúp ngăn chặn việc nghe lén hoặc can thiệp từ bên thứ ba.

Kết luận

Qua quá trình tìm hiểu và triển khai đề tài, em đã xây dựng thành công một mô hình **Client - Server có xác thực và bảo mật**, với các tính năng như:

- Áp dụng **mã hóa bất đối xứng RSA** kết hợp với **chữ ký số SHA256withRSA** để đảm bảo tính xác thực và toàn vẹn của dữ liệu.
- Sử dụng **mã hóa đối xứng AES-CBC** để tăng hiệu quả trong việc bảo vệ dữ liệu khi truyền tải.
- Tự động xử lý và đóng gói thông tin dưới dạng **JSON**, giúp dễ dàng trao đổi giữa client và server.
- Hệ thống có khả năng kiểm tra và quản lý **UUID**, giúp phân biệt người dùng, đồng thời lưu trữ thông tin mã hóa an toàn cho mỗi client.
- Triển khai chức năng **quét subdomain** sau khi xác thực thành công, giúp minh họa rõ ràng việc chỉ cho phép người dùng hợp lệ thực hiện hành động.
- Cuối cùng, hệ thống được **tăng cường bảo mật với giao thức TLS**, đảm bảo toàn bộ luồng giao tiếp giữa client và server đều được mã hóa, ngăn chặn việc rò rỉ dữ liệu hay bị tấn công trung gian (MITM).

Thông qua đề tài này, em đã củng cố kiến thức về lập trình mạng, mã hóa, xác thực và bảo mật trong giao tiếp client-server. Đây là nền tảng quan trọng để phát triển các ứng dụng mạng an toàn và hiệu quả hơn trong thực tế.