

**Bộ Thông tin – Truyền thông**  
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG TP.HCM**  
*Khoa Công Nghệ Thông Tin 2*



**Đồ án giữa kỳ - An toàn thông tin**

**CHƯƠNG TRÌNH SỬ DỤNG KHÓA ĐƯỢC TẠO TỪ GIẢI  
THUẬT DIFFIE – HELLMAN ĐỂ MÃ HÓA VÀ GIẢI MÃ  
THÔNG TIN BẰNG GIẢI THUẬT RSA**

**Nhóm thực hiện:** Nhóm 3

Phan Thị Trà Giang – N19DCPT015

Võ Thị Thanh Ngân – N19DCPT039

**Lớp:** D19CQPU01 – N      **Khóa:** 2019

**Giáo viên hướng dẫn:** Th.S Đàm Minh Lịnh

Niên khóa: 2019 – 2024

TP. Hồ Chí Minh, ngày 24 tháng 04 năm 2023

# MỤC LỤC

Chương I: CƠ SỞ LÝ THUYẾT .....	3
1. Giải thuật Trao đổi khóa Diffie – Hellman .....	3
1.1. Khái quát giải thuật .....	3
1.2. Giao thức tạo khóa trao đổi .....	4
1.3. Ví dụ minh họa .....	5
2. Giải thuật RSA .....	5
2.1. Khái quát giải thuật .....	5
2.2. Mã hóa – giải mã .....	6
2.3. Ví dụ minh họa .....	7
Chương II: ỨNG DỤNG THỰC TẾ .....	8
1. Ý tưởng chương trình .....	8
2. Xây dựng giao diện hệ thống .....	8
3. Xây dựng chương trình .....	10
3.1. Xây dựng lớp giải thuật Diffie – Hellman .....	10
3.2. Xây dựng lớp giải thuật RSA .....	10
3.3. Xây dựng hàm mã hóa .....	11
3.4. Xây dựng hàm giải mã .....	11
3.5. Gọi các lớp/hàm đã xây dựng để chạy chương trình chính .....	11

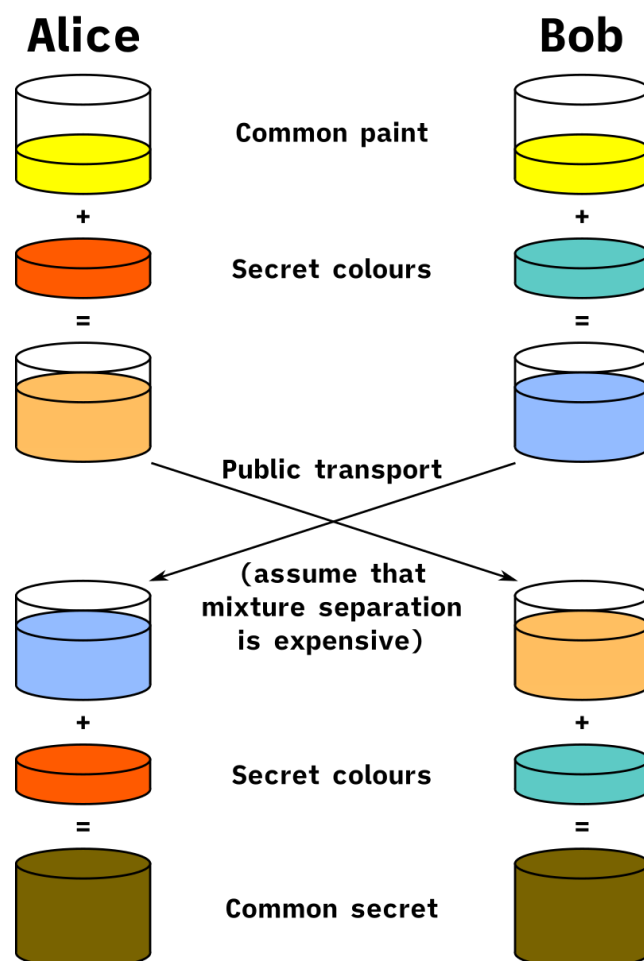
# Chương I: CƠ SỞ LÝ THUYẾT

## 1. Giải thuật Trao đổi khóa Diffie – Hellman

### 1.1. *Khái quát giải thuật*

Trao đổi khóa Diffie–Hellman (D-H) là một phương pháp trao đổi khóa được phát minh sớm nhất trong mật mã học. Phương pháp trao đổi khóa Diffie–Hellman cho phép hai bên (người, thực thể giao tiếp) thiết lập một khóa bí mật chung để mã hóa dữ liệu sử dụng trên kênh truyền thông không an toàn mà không cần có sự thỏa thuận trước về khóa bí mật giữa hai bên. Khóa bí mật tạo ra sẽ được sử dụng để mã hóa dữ liệu với phương pháp mã hóa khóa đối xứng.

Diffie–Hellman thiết lập bí mật chung để sử dụng cho trao đổi dữ liệu an toàn trên một kênh truyền thông công cộng không an toàn. Sơ đồ sau đây minh họa ý tưởng cơ bản của việc trao đổi khóa thông qua ví dụ về màu sơn.



### ❖ Ý tưởng cơ bản

Điểm chủ chốt của ý tưởng này là Alice và Bob trao đổi màu sơn bí mật thông qua hỗn hợp sơn.

- Đầu tiên Alice và Bob trộn màu đã biết chung (màu vàng) với màu bí mật riêng của mỗi người.
- Sau đó, mỗi người chuyển hỗn hợp của mình tới người kia thông qua một kênh vận chuyển công cộng.
- Khi nhận được hỗn hợp của người kia, mỗi người sẽ trộn thêm với màu bí mật của riêng mình và nhận được hỗn hợp cuối cùng.

#### 1.2. Giao thức tạo khóa trao đổi

Các dữ liệu cần để thực hiện giao thức:

- ✓ Số nguyên tố  $p$
- ✓ Căn nguyên thủy  $g$
- ✓  $a$  khóa bí mật của bên gửi
- ✓  $b$  khóa bí mật của bên nhận

Từ dữ liệu trên, bắt đầu tính toán thêm các dữ liệu:

$$\text{Khóa công khai bên gửi } A = g^a \bmod p$$

$$\text{Khóa công khai bên nhận } B = g^b \bmod p$$

Sau khi tính được khóa công khai, hai bên trao đổi cho nhau. Tiếp tục tính khóa chia sẻ bí mật chung giữa 2 bên bằng công thức:

$$\text{Khóa chia sẻ bí mật bên gửi } kA = B^a \bmod p$$

$$\text{Khóa chia sẻ bí mật bên nhận } kB = A^b \bmod p$$

Cuối cùng so sánh để xác nhận tính đúng đắn của khóa chia sẻ bí mật giữa hai bên. Vì khóa này chỉ 2 bên biết nên đảm bảo được tính an toàn trong mã hóa và giải mã truyền thông tin.

### 1.3. Ví dụ minh họa

Giả sử Alice và Bob muốn trao đổi thông tin bí mật thông qua một kênh không an toàn. Trước khi trao đổi thông tin, hai người phải thực hiện các bước sau đây để tạo ra một khóa chung bí mật.

Bước 1: Alice và Bob chọn hai số nguyên tố lớn khác nhau  $p$  và  $g$ . Sau đó, Alice chọn một số nguyên  $a$  và tính  $A = g^a \bmod p$ . Bob cũng chọn một số nguyên  $b$  và tính  $B = g^b \bmod p$ .

Bước 2: Alice và Bob trao đổi giá trị  $A$  và  $B$ . Tuy nhiên, các giá trị này không bí mật và có thể bị đánh cắp bởi một bên thứ ba.

Bước 3: Alice tính  $K = B^a \bmod p$  và Bob tính  $K = A^b \bmod p$ . Như vậy, cả hai sẽ có cùng một giá trị  $K$ , đó là khóa chung bí mật được sử dụng để trao đổi thông tin bí mật.

**Ví dụ:** Giả sử  $p = 23$  và  $g = 5$ .

Alice chọn  $a = 6$

$$A = g^a \bmod p = 5^6 \bmod 23 = 8.$$

Bob chọn  $b = 15$

$$B = g^b \bmod p = 5^{15} \bmod 23 = 19.$$

Sau đó, Alice và Bob trao đổi giá trị  $A = 8$  và  $B = 19$ .

$$\text{Alice tính } K = B^a \bmod p = 19^6 \bmod 23 = 2$$

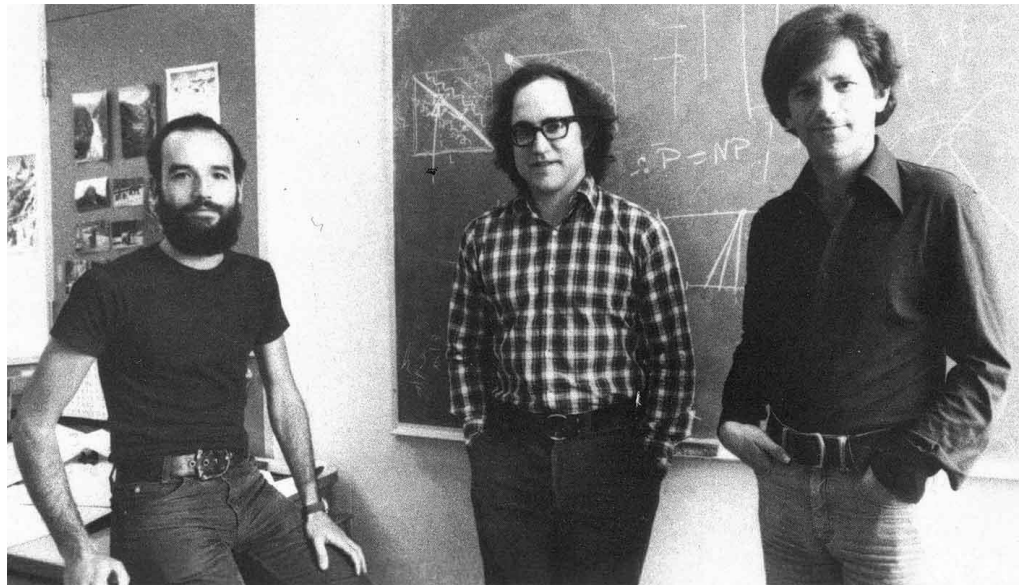
$$\text{Bob tính } K = A^b \bmod p = 8^{15} \bmod 23 = 2.$$

Như vậy, cả hai đều có giá trị  $K = 2$ , đó là khóa chung bí mật được sử dụng để trao đổi thông tin bí mật giữa họ.

## 2. Giải thuật RSA

### 2.1. Khái quát giải thuật

Giải thuật RSA là một giải thuật mã hóa công khai được sử dụng rộng rãi để bảo mật thông tin. Nó được đặt tên theo tên của ba nhà toán học Rivest, Shamir và Adleman, đã đưa ra giải thuật này vào năm 1977.



Giải thuật RSA hoạt động dựa trên hai số nguyên lớn, được gọi là khóa công khai và khóa bí mật. Khóa công khai được sử dụng để mã hóa dữ liệu và khóa bí mật được sử dụng để giải mã dữ liệu đã được mã hóa. Khóa công khai có thể được chia sẻ rộng rãi và không cần bảo mật, trong khi khóa bí mật phải được giữ bí mật.

## 2.2. Mã hóa – giải mã

Quá trình tạo khóa RSA bao gồm các bước sau:

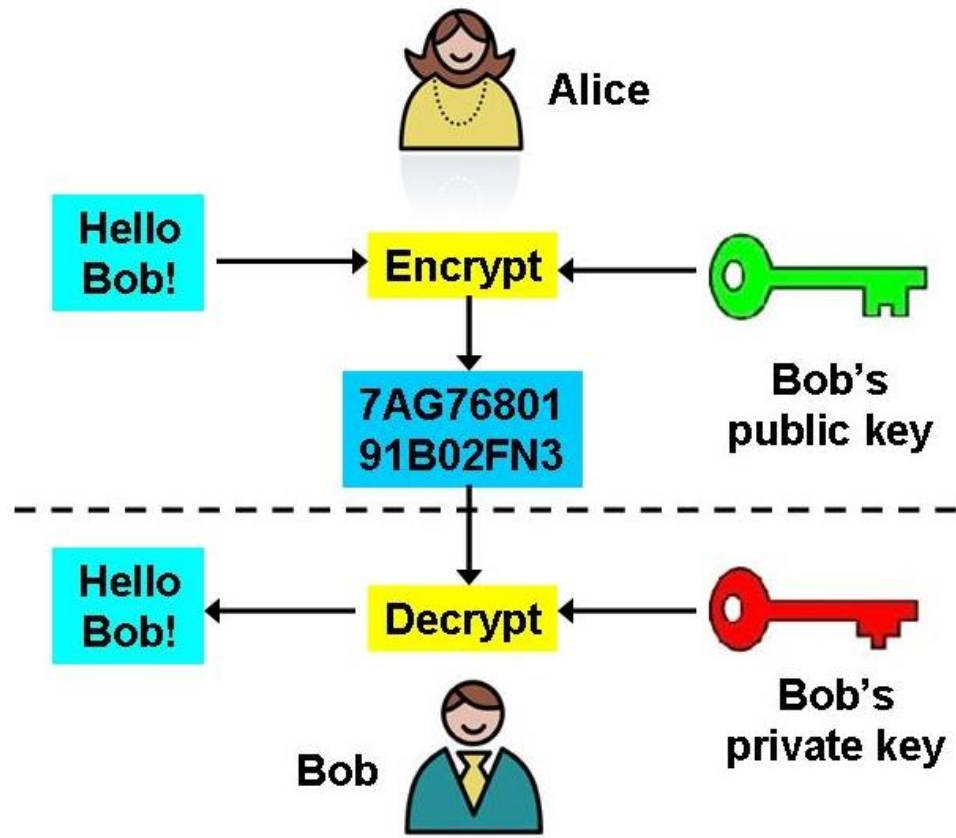
- ✓ Chọn hai số nguyên tố  $p$  và  $q$ .
- ✓ Tính  $n = pq$ .
- ✓ Tính hàm số Euler:  $\phi(n) = (p-1)(q-1)$ .
- ✓ Chọn một số nguyên  $e$  sao cho  $1 < e < \phi(n)$  và  $e$  là số nguyên tố cùng nhau với  $\phi(n)$ .
- ✓ Tìm  $d$  sao cho  $e*d \equiv 1 \pmod{\phi(n)}$ .
- ✓ Khóa công khai là  $(n, e)$  và khóa bí mật là  $d$ .

Sau khi có khóa, quá trình mã hóa và giải mã được thực hiện theo các bước sau:

**Mã hóa:** Chọn một số nguyên  $m < n$  và tính  $c = m^e \pmod{n}$ . Số  $c$  là số được mã hóa.

**Giải mã:** Tính  $m = c^d \pmod{n}$ . Số  $m$  là số được giải mã.

### 2.3. Ví dụ minh họa



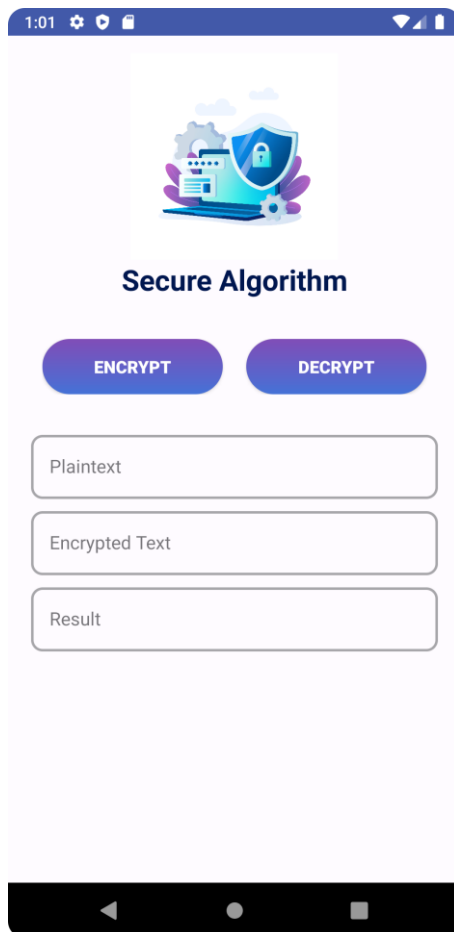
*Bài toán trao đổi thông tin giữa Alice và Bob*

## Chương II: ỨNG DỤNG THỰC TẾ

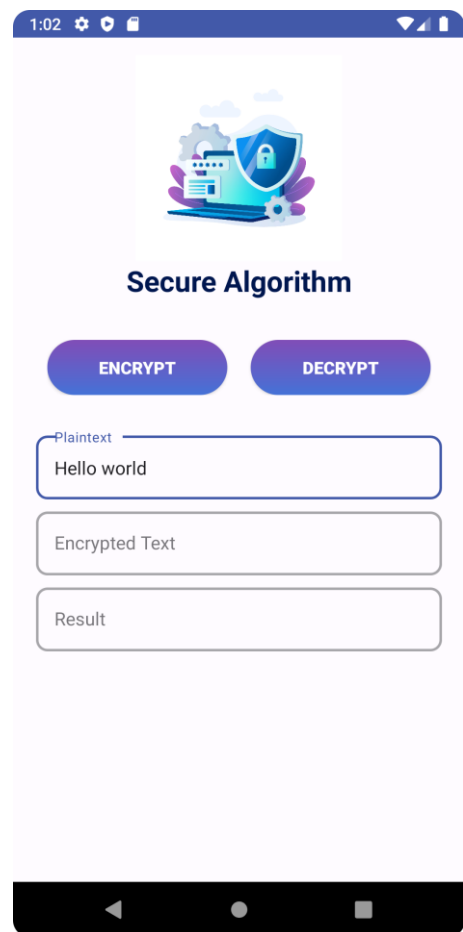
### 1. Ý tưởng chương trình

Chương trình sẽ được xây dựng dựa trên 2 thuật toán RSA và Diffie – Hellman (DH). Thuật toán DH được sử dụng để tạo ra khóa trao đổi an toàn giữa 2 đối tượng nên ta sẽ sử dụng tạo ra một khóa. Khóa được tạo ra bởi thuật toán DH sẽ được dùng làm khóa công khai trong thuật toán RSA. Thuật toán RSA nhận khóa công khai và mã hóa chuỗi thông tin (plaintext), sau đó dùng khóa công khai đó để tính toán ra khóa bí mật nhằm giải mã thông tin đã được mã hóa (encrypted text).

### 2. Xây dựng giao diện hệ thống

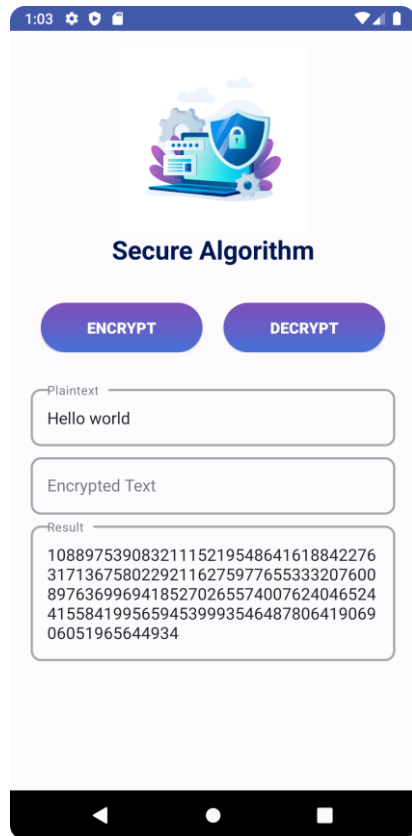


Giao diện chính

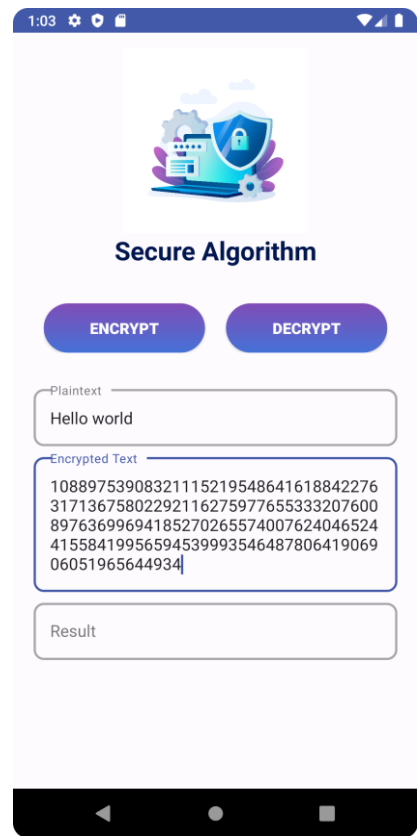


Nhập bản rõ (plain text)

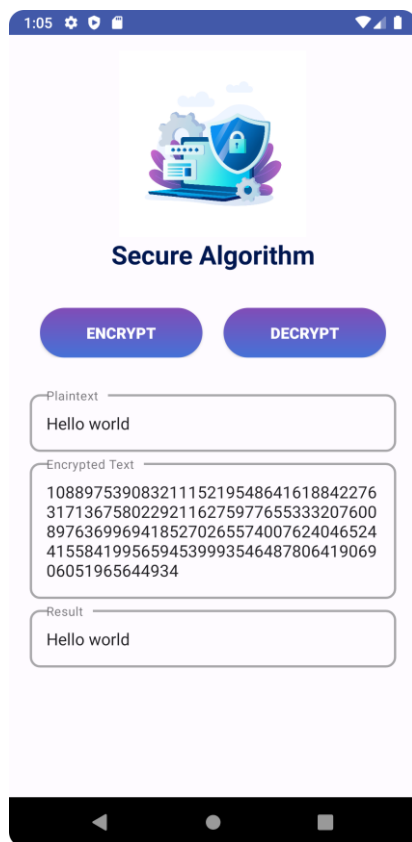




Hiện thị đoạn văn bản đã được mã hóa



Màn hình nhập chuỗi cần giải mã



Hiện thị chuỗi đã giải mã

### 3. Xây dựng chương trình

#### 3.1. Xây dựng lớp giải thuật Diffie – Hellman

```
// Get Shared Key by Diffie-Hellman algorithm
public static class DiffieHellman {
    private static final SecureRandom random = new SecureRandom();
    public static BigInteger sharedKDH;

    static {
        // Step 1: Choose a large prime number p and a primitive root g mod p
        BigInteger pDH = BigInteger.probablePrime(bitLength, random); //
        // large prime number
        BigInteger gDH = BigInteger.valueOf(2); // primitive root of p

        // Step 2: Choose a secret random number a and b, respectively
        BigInteger priADH = new BigInteger(bitLength, random);
        BigInteger priBDH = new BigInteger(bitLength, random);

        // Step 3: Exchange public keys A and B, respectively
        BigInteger pubADH = gDH.modPow(priADH, pDH);
        BigInteger pubBDH = gDH.modPow(priBDH, pDH);

        // Step 4: Compute the shared secret key K
        BigInteger K_Alice = pubBDH.modPow(priADH, pDH);
        BigInteger K_Bob = pubADH.modPow(priBDH, pDH);

        // Verify that the shared secret key K is the same for Alice and Bob
        if (K_Alice.equals(K_Bob)) {
            System.out.println("Shared secret key K: " + K_Alice);
            sharedKDH = K_Alice;
        } else {
            System.out.println("Error: Shared secret key K does not match.");
        }
    }
}
```

#### 3.2. Xây dựng lớp giải thuật RSA

```
// RSA key generation class
public static class RSA {
    private static final SecureRandom random = new SecureRandom();
    public static BigInteger N, phi;

    static {
        BigInteger p = BigInteger.probablePrime(bitLength, random);
        BigInteger q = BigInteger.probablePrime(bitLength, random);
        N = p.multiply(q);
        phi = (p.subtract(ONE)).multiply(q.subtract(ONE));
    }
}
```

```
}  
}
```

### 3.3. Xây dựng hàm mã hóa

```
// RSA encryption function (publicKey, moduleN)  
public static BigInteger rsaEncrypt(BigInteger message, BigInteger sharedKey)  
{  
    return message.modPow(sharedKey, RSA.N);  
}
```

### 3.4. Xây dựng hàm giải mã

```
// RSA decryption function (privateKey, moduleN)  
public static BigInteger rsaDecrypt(BigInteger message, BigInteger sharedKey)  
{  
    while (sharedKey.compareTo(RSA.phi) >= 0 ||  
!sharedKey.gcd(RSA.phi).equals(ONE)) {  
        sharedKey = sharedKey.add(ONE);  
    }  
    return message.modPow(sharedKey.modInverse(RSA.phi), RSA.N);  
}
```

### 3.5. Gọi các lớp/hàm đã xây dựng để chạy chương trình chính

```
sharedKey = DiffieHellman.sharedKDH;  
  
btnEncrypt.setOnClickListener(view -> {  
    String message = editPlaintext.getText().toString();  
    editPlaintext.clearFocus();  
  
    if (!message.isEmpty()) {  
        plaintext = new BigInteger(message.getBytes());  
        // Step 5: Alice and Bob use the shared secret key K to encrypt and  
decrypt a message using RSA  
        encrypted = rsaEncrypt(plaintext, sharedKey);  
        System.out.println("Encrypt message: " + encrypted);  
        result.getText().clear();  
        result.setText(String.valueOf(encrypted));  
    } else {  
        Toast.makeText(this, "Please enter plaintext!",  
Toast.LENGTH_SHORT).show();  
    }  
});  
  
btnDecrypt.setOnClickListener(view -> {  
    String message = editEncrypt.getText().toString();
```

```
BigInteger encryptText = new BigInteger(message);
editEncrypt.clearFocus();

if (!message.isEmpty()) {
    BigInteger decrypted = rsaDecrypt(encryptText, sharedKey);
    System.out.println("Decrypt message: " + decrypted);
    result.getEditText().getText().clear();
    result.getEditText().setText(new String(decrypted.toByteArray(),
StandardCharsets.UTF_8));
} else {
    Toast.makeText(this, "Please enter encrypted text!",
Toast.LENGTH_SHORT).show();
}
});
```