

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



**BÁO CÁO BÀI THỰC HÀNH THỰC TẬP CƠ SỞ**  
**Bài 16: Lập trình thuật toán mật mã học**

Họ và tên: Nguyễn Huy Quang

Mã sinh viên: B20DCAT144

Giảng Viên: Nguyễn Hoa Cường

*Hà Nội – 2023*

## I. Mục đích

- Tìm hiểu một giải thuật mã hóa phổ biến và lập trình được chương trình mã hóa và giải mã sử dụng ngôn ngữ lập trình phổ biến như C/C++/Python/Java, đáp ứng chạy được với số lớn.

## II. Nội dung thực hành

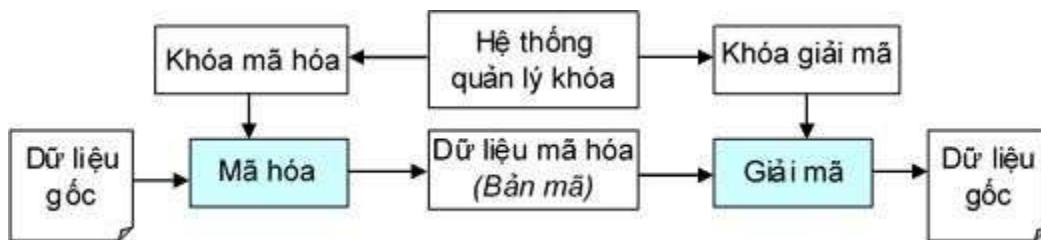
### 2.1. Lý thuyết

#### 1. Kiểu dữ liệu *BigInteger* trong java

- BigInteger tương tự như kiểu dữ liệu nguyên thủy *int*, *long* nhưng cho phép lưu trữ giá trị kiểu số nguyên cực lớn.
- Cộng BigInteger – *add()*
- Trừ BigInteger – *subtract()*
- Nhân BigInteger – *multiply()*
- So sánh BigInteger: Sử dụng *compareTo()* method để so sánh với một BigInteger khác, trả về **-1** nếu bé hơn BigInteger được so sánh, **0** nếu bằng và **1** nếu lớn hơn.

#### 2. Thuật toán RSA

- Thuật toán RSA có hai khóa: khóa công khai (hay khóa công cộng) và khóa bí mật (hay khóa cá nhân). Mỗi khóa là những số cố định sử dụng trong quá trình mã hóa và giải mã. Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được.
- Mã hóa dữ liệu là sử dụng một phương pháp biến đổi dữ liệu từ dạng bình thường sang một dạng khác, mà một người không có thẩm quyền, không có phương tiện giải mã thì không thể đọc hiểu được.
- Giải mã dữ liệu là quá trình ngược lại, là việc sử dụng một phương pháp biến đổi dữ liệu đã được mã hóa về dạng thông tin ban đầu. Có thể mô tả quy trình thực hiện mã hóa dữ liệu và giải mã dữ liệu như sau:



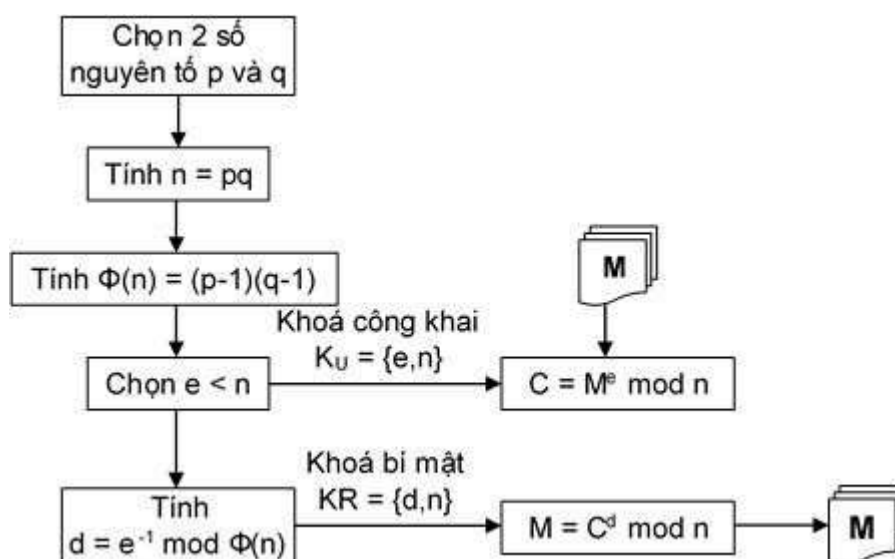
**Quy trình mã hóa dữ liệu**

- Sau đây là một số khái niệm và kí hiệu liên quan về vấn đề mã hóa dữ liệu :
- E (Encryption - Mã hóa): Là quá trình chuyển đổi dữ liệu gốc thành dữ liệu được mã hóa sao người khác không thể đọc hiểu.
- D (Decryption - Giải mã): Là quá trình ngược lại của mã hóa, biến đổi dữ liệu đã được mã hóa thành dạng gốc ban đầu.
- M (Message - Thông điệp), bản gốc hay bản rõ (Plaintext): Là tệp dữ liệu chưa được mã hóa hoặc đã được giải mã.
- C (Ciphertext - Bản mã): Tệp dữ liệu đã được mã hóa.
- K (Key - Khóa): Là dãy các bit 0, 1 thường được đưa ra dạng xâu ký tự, số...
- $K_E$ : Là khóa dùng để mã hóa.
- $K_D$ : Là khóa dùng để giải mã.

Theo quy ước, khi mã hóa thì  $C = E(M)$  và khi giải mã thì  $M = D(C) = D(E(M))$ .

Theo phương pháp truyền thống, người ta thường dùng cùng một khóa để mã hóa và giải mã. Lúc đó khóa phải được giữ bí mật tuyệt đối. Người ta gọi đây là hệ thống mã hóa cổ điển (các tên gọi khác: đối xứng, một khóa, khóa bí mật...). Một phương pháp khác, sử dụng khóa công khai (còn gọi là phương pháp mã hóa bất đối xứng, hay hệ thống hai khóa) trong đó khóa để mã hóa và khóa để giải mã là khác nhau. Các khóa này tạo thành một cặp chuyển đổi ngược nhau và không khóa nào có thể suy ra được từ khóa kia. Quy trình mã hóa khóa công khai gồm có các bước cơ bản như sau:

- Mỗi một hệ thống đầu cuối tạo một cặp khóa dùng cho quá trình mã hóa và giải mã mà hệ thống đó sẽ nhận.
- Mỗi hệ thống công bố khóa mã hóa của mình, gọi là khóa công khai, khóa còn lại gọi là khóa bí mật (khóa riêng) và phải được giữ an toàn.
- Nếu Bên gửi muốn gửi thông điệp cho Bên nhận, Bên gửi mã hóa thông điệp sử dụng khóa công khai của Bên nhận.
- Khi Bên nhận nhận thông điệp, Bên nhận giải mã bằng khóa riêng của Bên nhận.
- Sơ đồ dưới đây minh họa các bước trong thuật toán RSA.



### Sơ đồ biểu diễn thuật toán mã hóa RSA

- Tạo khóa

Mấu chốt cơ bản của việc sinh khóa trong RSA là tìm được bộ 3 số tự nhiên  $e$ ,  $d$  và  $n$  sao cho:  $m^{ed} \equiv m \pmod n$  và một điểm không thể bỏ qua là cần bảo mật cho  $d$  sao cho dù biết  $e$ ,  $n$  hay thậm chí cả  $m$  cũng không thể tìm ra  $d$  được.

Cụ thể, khóa của RSA được sinh như sau:

- Chọn 2 số nguyên tố  $p$  và  $q$
- Tính  $n=pq$ . Sau này,  $n$  sẽ được dùng làm modulus trong cả public key và private key.
- Tính một số giả nguyên tố bằng phi hàm Carmichael như sau:  $\lambda(n) = \text{BCNN}(\lambda(p), \lambda(q)) = \text{BCNN}(p-1, q-1)$ . Giá trị này sẽ được giữ bí mật.
- Chọn một số tự nhiên  $e$  trong khoảng  $(1, \lambda(n))$  sao cho  $\text{UCLN}(e, \lambda(n)) = 1$ , tức là  $e$  và  $\lambda(n)$  nguyên tố cùng nhau.
- Tính toán số  $d$  sao cho  $d \equiv 1/e \pmod{\lambda(n)}$  hay viết dễ hiểu hơn thì  $de \equiv 1 \pmod{\lambda(n)}$ . Số  $d$  được gọi là số nghịch đảo modulo của  $e$  (theo modulo  $\lambda(n)$ ).  
Public key sẽ là bộ số  $(n, e)$ , và private key sẽ là bộ số  $(n, d)$ .

Chúng ta cần giữ private key thật cẩn thận cũng như các số nguyên tố  $p$  và  $q$  vì từ đó có thể tính toán các khóa rất dễ dàng.

Trong thực hành, chúng ta thường chọn  $e$  tương đối nhỏ để việc mã hóa và giải mã nhanh chóng hơn. Giá trị thường được sử dụng là  $e = 65537$ . Ngoài ra, chúng ta có thể tính số giả nguyên tố bằng phi hàm Euler  $\phi(n) = (p-1)(q-1)$  và dùng nó như  $\lambda(n)$ .

Vì  $\varphi(n)$  là bội của  $\lambda(n)$  nên các số  $d$  thỏa mãn  $de \equiv 1 \pmod{\varphi(n)}$  cũng sẽ thỏa mãn  $d \equiv 1/e \pmod{\lambda(n)}$ . Tuy nhiên, một số tác dụng phụ của việc này là  $d$  thường sẽ trở nên lớn hơn mức cần thiết.

- Mã hóa và giải mã
  - Nếu chúng ta có bản rõ  $M$ , chúng ta cần chuyển nó thành một số tự nhiên  $m$  trong khoảng  $(0, n)$  sao cho  $m, n$  nguyên tố cùng nhau. Việc này rất dễ dàng thực hiện bằng cách thêm một các kỹ thuật padding. Tiếp theo, chúng ta sẽ mã hóa  $m$ , thành  $c$  như sau:

$$c \equiv m^e \pmod{n}$$

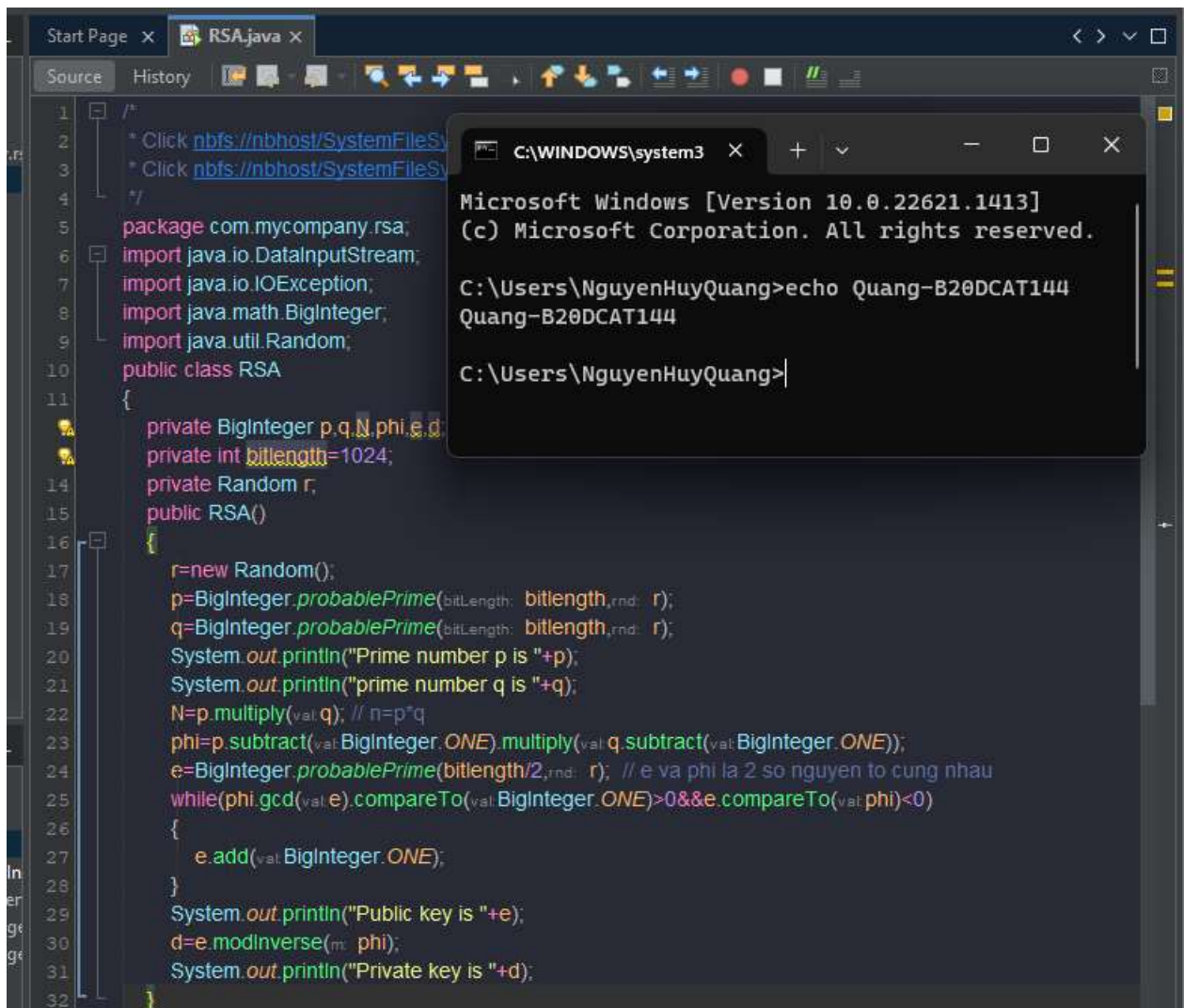
- Sau đó giá trị  $c$  sẽ được chuyển cho người nhận.
  - Ở phía người nhận, họ sẽ giải mã từ  $c$  để lấy được  $m$  như sau:

$$c^d \equiv m^{de} \equiv m \pmod{n}$$

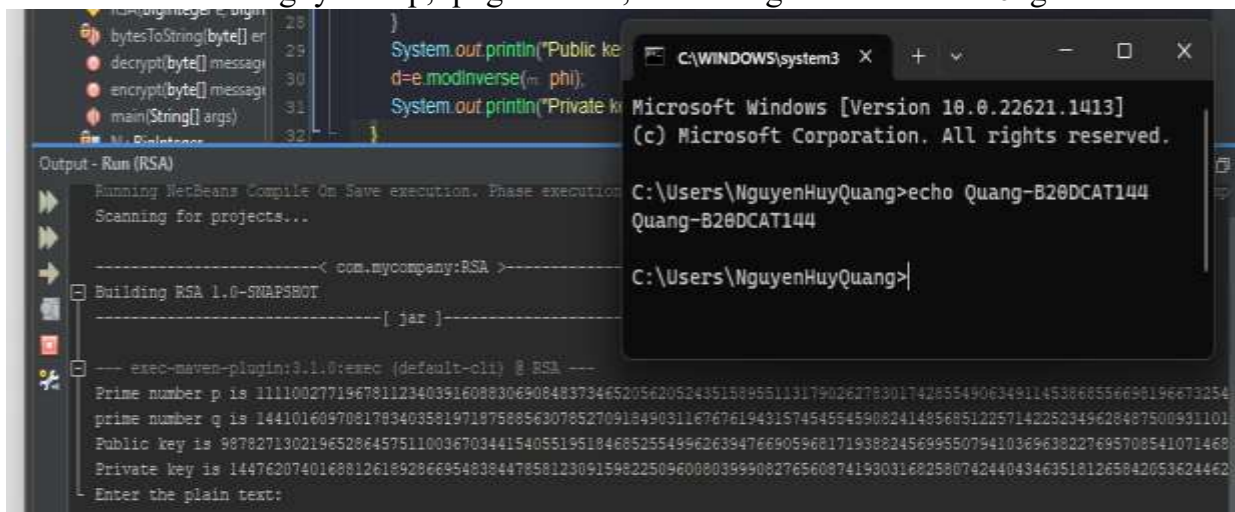
- Từ  $m$  có thể lấy lại được bản tin bằng cách đảo ngược padding
  - Mức độ bảo mật của RSA phụ thuộc rất lớn vào khả năng phân tích thừa số nguyên tố của các số lớn. Bởi vì chúng ta cung cấp public một cách rộng rãi, nếu việc phân tích thừa số nguyên tố đơn giản, thì việc bị lộ private là không thể tránh khỏi.
  - Vì vậy, khi sinh khóa, chúng ta cần chọn các số nguyên tố  $p$  và  $q$  một cách ngẫu nhiên. Bản thân hai số nguyên tố này cũng rất lớn, và để việc phân tích thừa số nguyên tố khó khăn hơn, hai số nguyên tố này sẽ không có cùng độ dài. Trong tương lai gần, có lẽ vẫn chưa có một phương pháp hiệu quả nào cho phép thực hiện điều này với các máy tính cá nhân.

## 2.2. Cài đặt

- Lập trình thư viện số lớn với các phép toán cơ bản để sử dụng trong giải thuật mã hóa/giải mã RSA: sử dụng BigInteger của Java( số nguyên cực lớn)

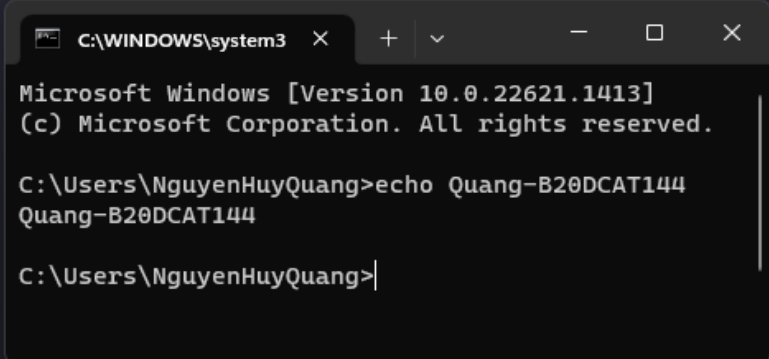


- Thử nghiệm chứng minh thư viện hoạt động tốt với các ví dụ phép toán cho số lớn: in ra 2 số nguyên tố  $p, q$  ngẫu nhiên, khóa công khai và khóa riêng



- Lập trình giải thuật mã hóa và giải mã
- Hàm `encrypt()` để mã hóa
- Hàm `decrypt()` để giải mã

```
}
private static String bytesToString(byte[] encrypted)
{
    String test=" ";
    for(byte b:encrypted) test+=Byte.toString(b);
    return test;
}
public byte[]encrypt(byte[]message)
{
    return(new BigInteger(val: message)).modPow(exponent: e,m: N).toByteArray();
}
public byte[]decrypt(byte[]message)
{
    return(new BigInteger(val: message)).modPow(exponent: d,m: N).toByteArray();
}
}
```



C:\WINDOWS\system32

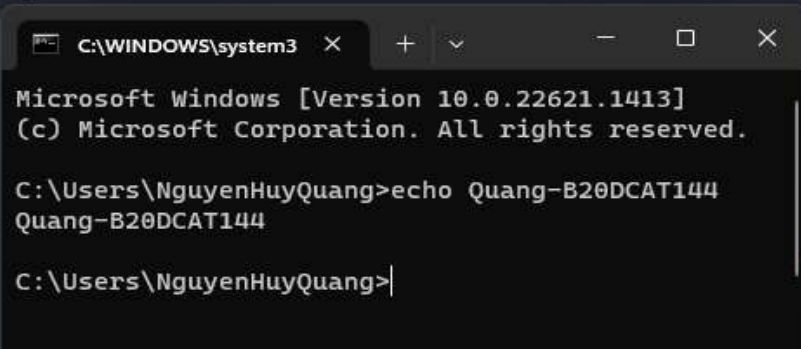
Microsoft Windows [Version 10.0.22621.1413]  
(c) Microsoft Corporation. All rights reserved.

C:\Users\NguyenHuyQuang>echo Quang-B20DCAT144  
Quang-B20DCAT144

C:\Users\NguyenHuyQuang>

- Chương trình chính

```
}
public static void main(String[] args)throws IOException
{
    RSA rsa=new RSA();
    DataInputStream in=new DataInputStream(in: System.in);
    String testString;
    System.out.println("Enter the plain text:");
    testString=in.readLine();
    System.out.println("Encrypting string:"+testString);
    System.out.println("string in bytes:"+bytesToString(encrypted: testString.getBytes()));
    byte[] encrypted=rsa.encrypt(message: testString.getBytes());
    byte[] decrypted=rsa.decrypt(message: encrypted);
    System.out.println("Decrypting Bytes:"+bytesToString(encrypted: decrypted));
    System.out.println("Decrypted string:"+new String(bytes: decrypted));
}
}
```



C:\WINDOWS\system32

Microsoft Windows [Version 10.0.22621.1413]  
(c) Microsoft Corporation. All rights reserved.

C:\Users\NguyenHuyQuang>echo Quang-B20DCAT144  
Quang-B20DCAT144

C:\Users\NguyenHuyQuang>

- Thử nghiệm mã hóa và giải mã số lớn



```
Out C:\Users\NguyenHuyQuang>echo Quang-B20DCAT144
>> Quang-B20DCAT144
>> C:\Users\NguyenHuyQuang>

Enter the plain text:
100099049878978932453045903423452389043245890345803454390248239048349064
Encrypting string:100099049878978932453045903423492389043245890345803494390248239048349064
string in bytes: 49484848575748525756555657555657505152565748515052575148525357485152505152575051565748525150525356574851525356
Dcrypting Bytes: 49484848575748525756555657555657505152565748515052575148525357485152505152575051565748525150525356574851525356
Dcrypted string:100099049878978932453045903423492389043245890345803494390248239048349064
-----
```

- Thử nghiệm mã hóa và giải mã chuỗi ký tự: Quang-B20DCAT144

```
Enter the plain text:
Quang-B20DCAT144
Encrypting string:Quang-B20DCAT144
string in bytes: 81117971101034566504868676584495252
Dcrypting Bytes: 81117971101034566504868676584495252
Dcrypted string:Quang-B20DCAT144
```