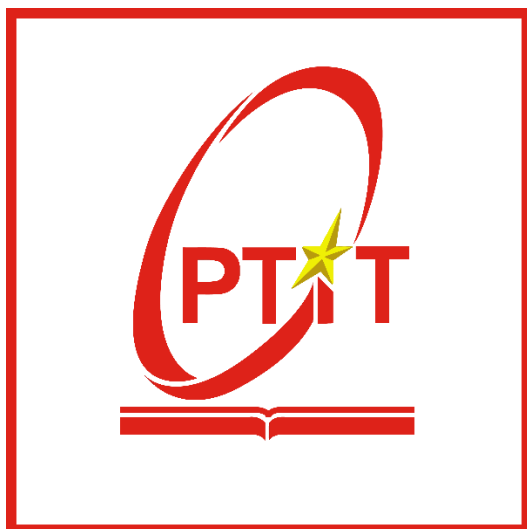


HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN



Môn học: THỰC TẬP CƠ SỞ
BÁO CÁO BÀI THỰC HÀNH SỐ 16
LẬP TRÌNH THUẬT TOÁN MẬT MÃ HỌC

Sinh viên thực hiện: Lê Anh Tuấn

Mã sinh viên: B21DCAT205

Giảng viên: Ninh Thị Thu Trang

~ Hà Nội, tháng 5/2024 ~

Mục Lục

1	Mục đích	2
2	Nội dung thực hành.....	2
2.1	Lý thuyết	2
2.1.1.	Kiến thức cơ bản về RSA.....	2
2.1.2.	Ứng dụng của thuật toán RSA	3
2.1.3.	Ưu điểm và hạn chế.....	3
2.2	Nội dung thực hành.	4
3	Kết luận	10

Bài 16: Lập trình thuật toán mật mã học

1 Mục đích

- Sinh viên 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.

2 Nội dung thực hành

2.1 Lý thuyết

2.1.1. Kiến thức cơ bản về RSA

- RSA là một giải thuật mã hóa được 3 nhà khoa học Ronald Rivest, Adi Shamir và Leonard Adleman phát minh năm 1977 (Tên giải thuật RSA lấy theo chữ cái đầu của tên 3 ông). Độ an toàn của RSA dựa trên tính khó của việc phân tích số nguyên rất lớn (số có hàng trăm chữ số thập phân).

- Kích thước khóa của RSA:

- Khóa $< 1024\text{bit}$ không an toàn hiện nay.
- Khuyến nghị dùng khóa ≥ 2048 bit. Tương lai nên dùng khóa 3072 bit.

- RSA là mã hóa bất đối xứng, sử dụng một cặp khóa:

- Khóa công khai (Public key) dùng để mã hóa;
- Khóa riêng (Private key) dùng để giải mã.
- Chỉ khóa riêng cần giữ bí mật. Khóa công khai có thể công bố rộng rãi.

- Tạo khóa:

B1: Chọn 2 số nguyên tố lớn p, q với $p \neq q$

B2: Tính $n = pq$ Tính giá trị hàm số Euler $\phi(n) = (p-1) \times (q-1)$

B3: Chọn 1 số e sao cho $1 < e < \phi(n)$ và $\text{gcd}(e, \phi(n)) = 1$

B4: Tính $d = e^{-1} \pmod{\phi(n)}$, số d thỏa mãn $ed \equiv 1 \pmod{\phi(n)}$

Public Key gồm: n – module, e – số mũ mã hóa.

Private Key gồm: n – module, d – số mũ giải mã.

- Mã hóa: Tính ra bản mã c từ bản rõ m theo công thức: $c = m^e \pmod{n}$.

- Giải mã: Tính theo công thức $m = c^d \pmod{n}$

Quá trình giải mã có thể thu lại được m ban đầu là do:

$$c^d \equiv ((m^e)^d) \equiv (m^{ed}) \pmod{n} \equiv m \pmod{n} \text{ hay } m = (c^d) \pmod{n}$$

- 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.

- Tuy nhiên, với sự phát triển của công nghệ, các siêu máy tính xuất hiện ngày càng nhiều. Cùng với chúng ta máy tính lượng tử cho phép tính toán với tốc độ cao hơn rất nhiều có thể sẽ phá vỡ sự bảo mật của RSA.
- Ngày từ năm 1993, thuật toán Shor đã được phát triển và chỉ ra rằng máy tính lượng tử có thể giải bài toán phân tích ra thừa số trong thời gian đa thức. Rất may là những điều này mới chỉ là lý thuyết vì đến thời điểm hiện tại và trong vài năm tới, máy tính lượng tử vẫn chưa hoàn thiện.

2.1.2. Ứng dụng của thuật toán RSA

- Mã hóa và Giải mã Thông tin: RSA được sử dụng để mã hóa dữ liệu truyền tải giữa các bên để đảm bảo tính toàn vẹn và bảo mật thông tin. Bạn có thể sử dụng khóa công khai để mã hóa thông tin và khóa riêng tư để giải mã.
- Chữ ký số: RSA được sử dụng để tạo chữ ký số, giúp xác minh nguồn gốc của một tài liệu hoặc dữ liệu. Người gửi có thể sử dụng khóa riêng tư để tạo chữ ký số, trong khi người nhận sử dụng khóa công khai để xác minh chữ ký.
- Bảo vệ Khóa Bí mật: RSA thường được sử dụng để bảo vệ khóa bí mật trong hệ thống mật mã. Những khóa này có thể được sử dụng trong các thuật toán mã hóa đối xứng để bảo vệ dữ liệu.
- Bảo mật Giao tiếp qua Mạng: RSA thường được sử dụng để bảo vệ giao tiếp qua mạng, đặc biệt là khi cần thiết lập kênh bảo mật trên Internet. HTTPS, một biến thể của giao thức HTTP sử dụng RSA để bảo vệ thông tin giao tiếp qua mạng.
- Quản lý Chứng chỉ Số: RSA được sử dụng trong quá trình phát hành và quản lý chứng chỉ số trong hệ thống mật mã. Chứng chỉ số thường được sử dụng để xác minh danh tính của các thực thể trên Internet.
- Bảo mật Email: RSA có thể được sử dụng để bảo mật giao tiếp qua email thông qua việc sử dụng chữ ký số và mã hóa email.
- Bảo mật Điện toán Đám mây: Trong môi trường đám mây, RSA có thể được sử dụng để bảo vệ dữ liệu và thông tin truyền tải giữa các máy chủ và các thiết bị kết nối với đám mây.
- Bảo mật Truy cập Mạng: RSA cũng có thể được tích hợp vào các hệ thống xác thực và quản lý truy cập mạng để đảm bảo an toàn trong việc quản lý quyền truy cập.

2.1.3. Ưu điểm và hạn chế

Ưu điểm của RSA:

- An toàn bảo mật: RSA dựa trên tính toán hiện đại và khả năng tính toán của con người hiện nay chưa có khả năng giải mã một cách hiệu quả nếu không có khóa riêng tư tương ứng.
- Không đòi hỏi quá trình giao tiếp để chia sẻ khóa: Trong hệ thống mã hóa truyền thống, việc chia sẻ khóa là một vấn đề lớn. RSA giải quyết vấn đề này bằng việc sử dụng khóa công khai và khóa riêng tư, không đòi hỏi quá trình giao tiếp để chia sẻ khóa.

- Dùng cho việc ký số và xác minh nguồn gốc: RSA là lựa chọn tốt cho việc tạo và xác minh chữ ký số, giúp đảm bảo nguồn gốc của dữ liệu và thông tin.
- Ứng dụng rộng rãi: RSA được sử dụng rộng rãi trong nhiều lĩnh vực bảo mật, bao gồm mã hóa dữ liệu, xác thực, và bảo mật truyền thông.

Hạn chế của RSA:

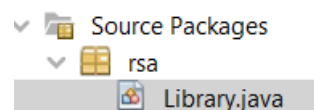
- Hiệu suất tính toán: Quá trình mã hóa và giải mã RSA yêu cầu tính toán nhiều, đặc biệt là với các khóa kích thước lớn. Điều này có thể ảnh hưởng đến hiệu suất của hệ thống, đặc biệt là trên thiết bị có tài nguyên hạn chế.
- Khóa dài: Đối với mức độ an toàn mong muốn, RSA thường sử dụng các khóa có kích thước lớn, điều này có thể làm tăng kích thước của dữ liệu mã hóa và chi phí tính toán.
- Vấn đề về quản lý khóa: Quản lý khóa là một thách thức, đặc biệt là khi số lượng người dùng tăng lên. Cần phải đảm bảo rằng khóa riêng tư được bảo vệ một cách chặt chẽ và không bị rò rỉ.
- Khả năng tấn công theo chiều ngược: Trong tương lai, với sự phát triển của công nghệ tính toán, có khả năng một số thuật toán tấn công theo chiều ngược có thể trở nên hiệu quả hơn, làm giảm độ an toàn của RSA.
- Khả năng tấn công thông qua giả mạo: RSA có thể bị tấn công thông qua giả mạo nếu một kẻ tấn công có thể giả mạo một khóa công khai sao chép và sử dụng nó để giải mã dữ liệu được mã hóa.

Mặc dù có những hạn chế, RSA vẫn là một trong những thuật toán mã hóa phổ biến và mạnh mẽ được sử dụng rộng rãi trên thế giới.

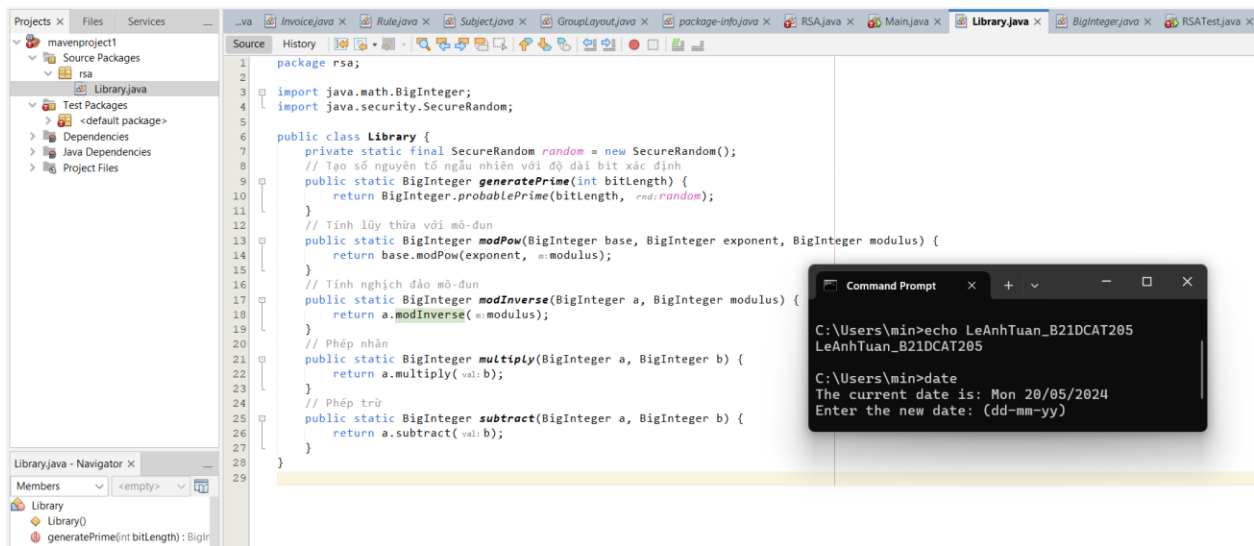
2.2 Nội dung thực hành.

Trong nội dung thực hành này, em sẽ sử dụng ngôn ngữ Java.

Đầu tiên ta sẽ tạo 1 package tên rsa, trong package này, ta sẽ tạo 1 file tên Library.java, file này sẽ chứa những hàm mà RSA cần xử lý với các số nguyên lớn

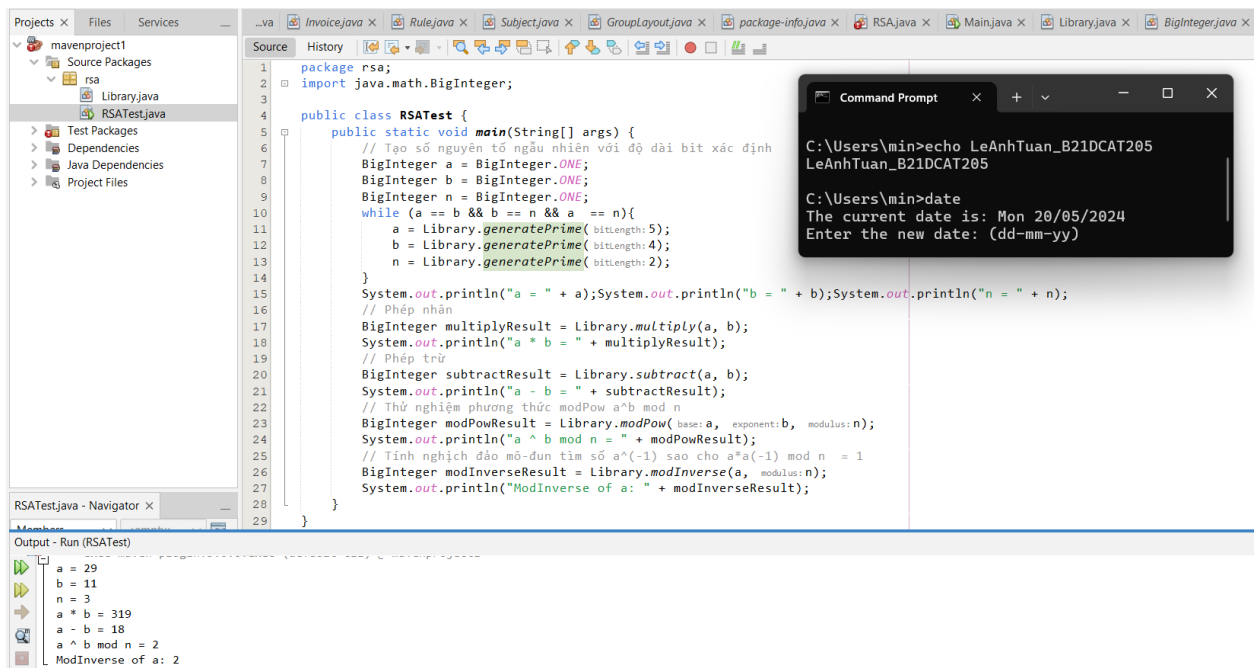


Hình 1: Tạo file Library.java



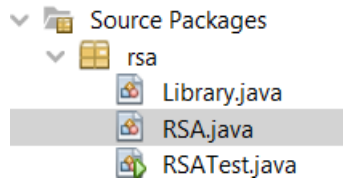
Hình 2: Tổng quan file Library.java

Ta sẽ thử nghiệm các hàm vừa viết.



Hình 3: Kết quả thử nghiệm

Tiếp theo, ta sẽ tạo 1 file RSA.java. File này sẽ là file chứa class RSA bao gồm các biến khởi tạo, các hàm khởi tạo và các hàm mã hóa và giải mã.

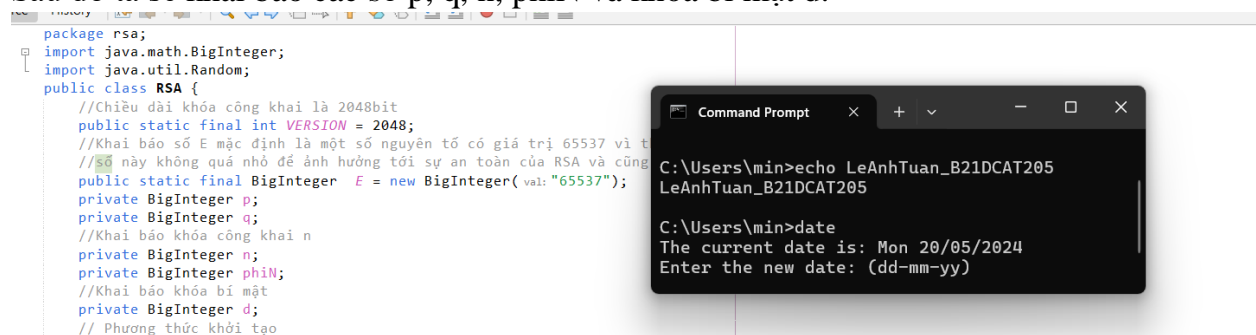


Hình 4: Tạo file RSA.java

Ban đầu ta sẽ định nghĩa chiều dài khóa là 2048bit, với kích thước này hiện nay thì các máy tính lượng tử cũng phải tốn khá nhiều thời gian mới có thể giải được.

Đối với số e, theo lý thuyết ta sẽ tìm số e sao cho e nguyên tố cùng nhau với n và $\phi(n)$ nhưng trên thực tế thì e sẽ thường là 1 một nguyên tố vì các số nguyên tố thì chắc chắn nguyên tố cùng nhau, em sẽ không sử dụng hàm sinh số nguyên tố ngẫu nhiên mà em sẽ gán mặc định cho nó là 65537, đây là một số nguyên tố được xem là khá an toàn, không quá nhỏ để ảnh hưởng tới sự an toàn và cũng không quá lớn để khiến việc tính toán phức tạp.

Sau đó ta sẽ khai báo các số p, q, n, phiN và khóa bí mật d.



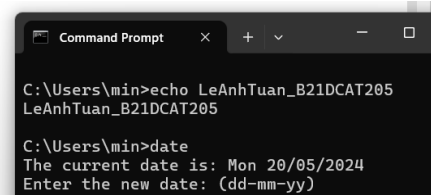
Hình 5: Khởi tạo các tham số cần thiết

Đối với phương thức khởi tạo, em sẽ sử dụng hàm probablePrime là 1 phương thức tĩnh của lớp BigInteger để khởi tạo một số nguyên tố lớn 1024bit hay VERSION /2 với VERSION = 2048 bit như đã khởi tạo ban đầu.

Tính giá trị $n = p \cdot q$

$$\phi(n) = (p - 1)(q - 1)$$

```
// Phương thức khởi tạo
public void initialize() {
    // p và q là số nguyên tố có chiều dài là 1024bit
    p = Library.generatePrime(VERSION / 2);
    q = Library.generatePrime(VERSION / 2);
    n = Library.multiply(a: p, b: q);
    // phiN = (p-1)*(q-1)
    phiN = Library.multiply(a: p.subtract(val: BigInteger.ONE), b: q.subtract(val: BigInteger.ONE));
    // Tìm d sao cho e*d mod phiN = 1
    d = Library.modInverse(a: E, modulus: phiN);
}
```



Hình 6: Khai báo phương thức khởi tạo

Hàm mã hóa sẽ được sử dụng cho bên gửi, hàm giải mã được sử dụng cho bên nhận.

Đối với bên nhận (n,e) công khai.

Bên gửi sẽ thực hiện mã hóa văn bản message bằng công thức

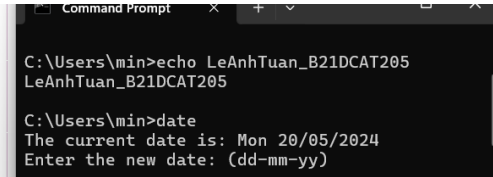
$$\text{encrypt} = \text{message}^e \bmod n$$

Hàm **getN()** chính là được bên gửi lấy n của bên nhận để sử dụng

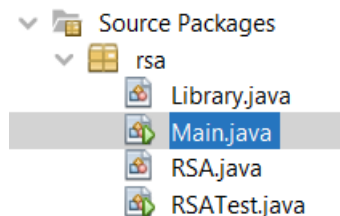
Sau đó bên nhận sẽ sử dụng khóa bí mật d và giải mã bằng công thức

$$\text{decrypt} = \text{encrypt}^d \bmod n$$

```
// Hàm mã hóa
public BigInteger encrypt(BigInteger message, BigInteger partnerN) {
    return Library.modPow(base: message, exponent: E, modulus: partnerN);
}
// Hàm giải mã
public BigInteger decrypt(BigInteger cipher) {
    return Library.modPow(base: cipher, exponent: d, modulus: n);
}
public BigInteger getN() {
    return n;
}
```



Hình 7: Các hàm mã hóa và giải mã



Hình 8: Tạo 1 file tên Main.java

Ta giả định một tình huống 2 người cần truyền thông tin cho nhau.


```
public class Main {
    public static void main(String[] args) {
        // Khai báo 2 người cần truyền thông điệp cho nhau
        // (người 1 truyền cho người 2)
        RSA person1 = new RSA();
        RSA person2 = new RSA();

        // Gọi phương thức khởi tạo
        person1.initialize();
        person2.initialize();
    }
}
```

```
Command Prompt
C:\Users\min>echo LeAnhTuan_B21DCAT205
LeAnhTuan_B21DCAT205

C:\Users\min>date
The current date is: Mon 20/05/2024
Enter the new date: (dd-mm-yy)
```

Hình 9: Người 1 truyền cho người 2

Người 1 gửi thông điệp cho người 2, người 1 sẽ phải tiến hành chuyển đổi thông điệp thành mảng byte, và từ mảng byte chuyển thành số nguyên lớn.

Với hàm `new BigInteger(int signum, byte[] magnitude)` trong đó `signum = 1` thể hiện số dương và `magnitude` là 1 mảng byte.

```
// Thông điệp người 1 gửi cho người 2
String messageString = "I am B21DCAT205";
// Chuyển đổi thông điệp thành mảng byte
byte[] messageBytes = messageString.getBytes(charset: StandardCharsets.UTF_8);
// Chuyển đổi mảng byte thành BigInteger
BigInteger message = new BigInteger(signum: 1, magnitude: messageBytes);
```

```
Command Prompt
C:\Users\min>echo LeAnhTuan_B21DCAT205
LeAnhTuan_B21DCAT205

C:\Users\min>date
The current date is: Mon 20/05/2024
Enter the new date: (dd-mm-yy)
```

Hình 10: Thông điệp người 1 muốn gửi

```
// Người 1 mã hóa lấy số N công khai của người thứ 2
BigInteger cipher = person1.encrypt(message, partnerN: person2.getN());
// Người 2 giải mã
BigInteger decrypted = person2.decrypt(cipher);
```

```
Command Prompt
C:\Users\min>echo LeAnhTuan_B21DCAT205
LeAnhTuan_B21DCAT205

C:\Users\min>date
The current date is: Mon 20/05/2024
Enter the new date: (dd-mm-yy)
```

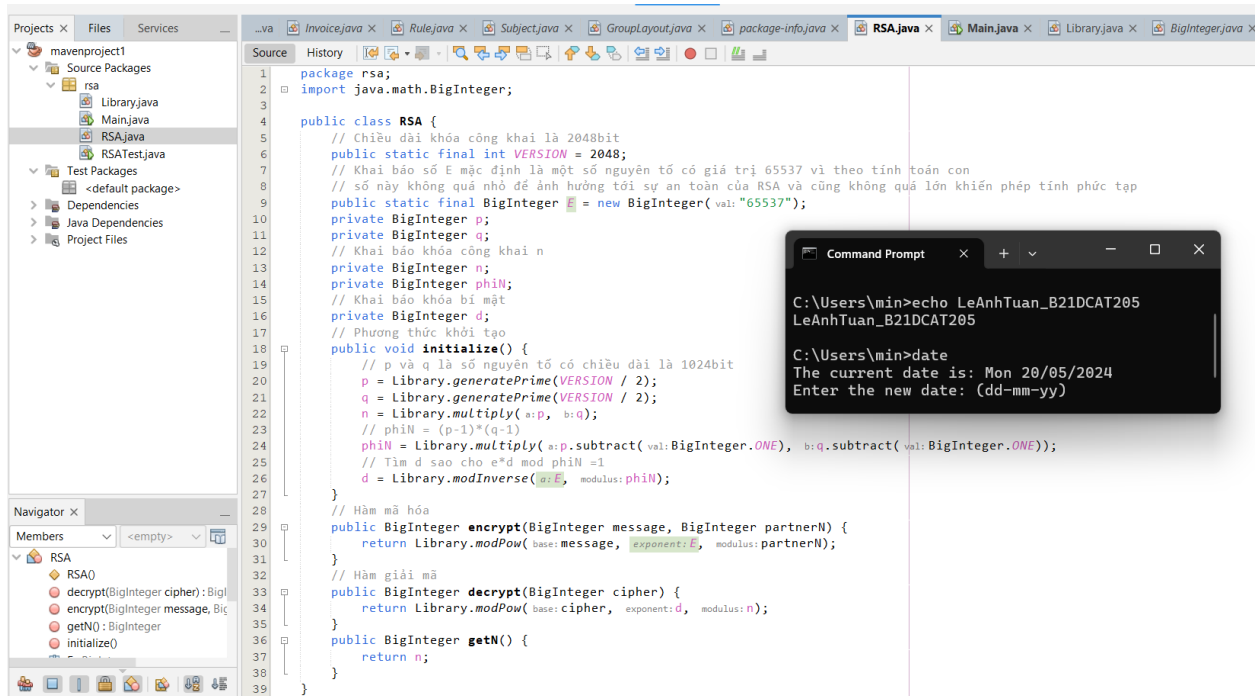
Hình 11: Người 1 thực hiện mã hóa, người 2 thực hiện giải mã

```
// Chuyển đổi BigInteger giải mã lại thành mảng byte
byte[] decryptedBytes = decrypted.toByteArray();
// Chuyển đổi mảng byte thành chuỗi
String decryptedMessage = new String(bytes: decryptedBytes, charset: StandardCharsets.UTF_8);
```

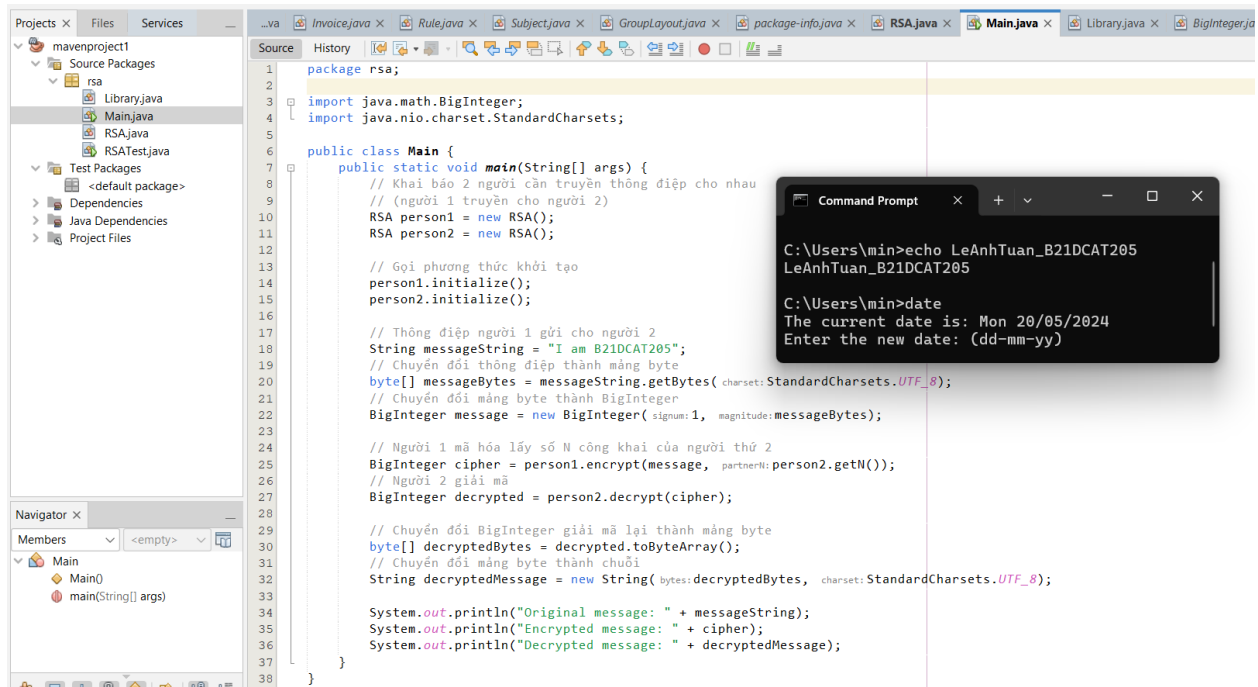
```
Command Prompt
C:\Users\min>echo LeAnhTuan_B21DCAT205
LeAnhTuan_B21DCAT205

C:\Users\min>date
The current date is: Mon 20/05/2024
Enter the new date: (dd-mm-yy)
```

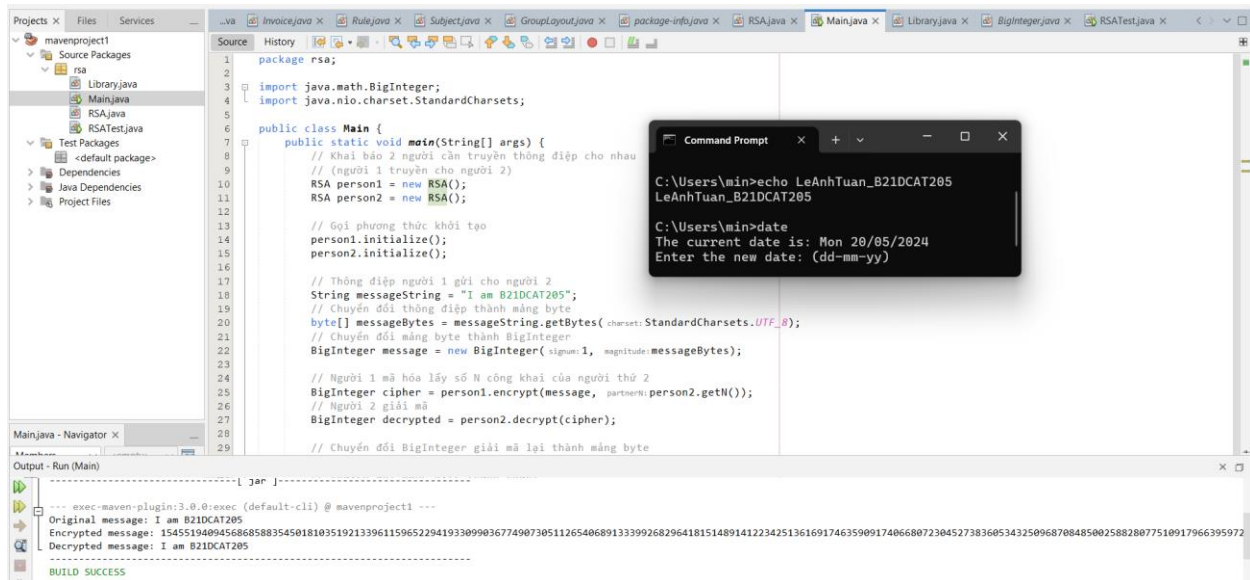
Hình 12: Chuyển đổi từ BigInteger -> mảng byte -> Chuỗi



Hình 13: Tổng quan file RSA.java



Hình 14: Tổng quan file Main.java



Hình 15: Kết quả

3 Kết luận

- Qua bài báo cáo trên, chúng ta đã cùng nhau tìm hiểu được giải thuật mã hóa phổ biến RSA và lập trình được chương trình mã hóa và giải mã sử dụng ngôn ngữ lập trình phổ biến Java, đáp ứng chạy được với số lớn. Kết quả đạt được là chúng ta đã thành công trong việc mã hóa số lớn và chuỗi ký tự, sử dụng giải thuật RSA.