

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA AN TOÀN THÔNG TIN



BÀI BÁO CÁO THỰC HÀNH SỐ 16

MÔN THỰC TẬP CƠ SỞ

Lập trình thuật toán mật mã học

Tên sinh viên: Ninh Chí Hường

Mã sinh viên: B20DCAT094

Số điện thoại: 0353770347

Giảng viên hướng dẫn : Th.s Ninh Thị Thu Trang

HÀ NỘI, THÁNG 5/2023

Contents

A. Mục đích	3
B. Tìm hiểu lý thuyết.....	3
1. Lập trình với số lớn.....	3
2. Giải thuật mật mã khóa công khai RSA	3
C. Các bước thực hiện và kết quả	4
Tài liệu tham khảo :.....	7

Bài 16: Lập trình thuật toán mật mã học

A. 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.

B. Tìm hiểu lý thuyết

1. Lập trình với số lớn

- Việc lập trình với những số có độ dài hàng nghìn bit là rất khó khăn. Thay vì tự viết hàm tính toán các số lớn, sinh viên sử dụng class BigInteger có sẵn trong Java chuyên để xử lý các số lớn.
- Class BigInteger cũng cung cấp những phép toán cơ bản như cộng add(), trừ subtract(), nhân multiply(), chia divide(), giúp việc tính toán các phép toán cơ bản dễ dàng hơn.
- Ngoài ra, BigInteger còn cung cấp hàm lũy thừa lấy phần dư modpow() hay hàm nghịch đảo modulo modInverse() giúp việc lập trình mã hoá và giải mã RSA dễ dàng hơn.

2. Giải thuật mật mã khóa công khai RSA

- Thuật toán mã hoá RSA là thuật toán mã hoá khóa công khai được sử dụng rộng rãi để truyền dữ liệu an toàn.
- Thuật toán mã hoá RSA được phát triển bởi Rivest, Shamir, Adleman.
Quy trình mã hoá của RSA được công khai năm 1977.
- Độ an toàn của RSA liên hệ chặt chẽ với độ khó của bài toán phân tích nhân tử của một số rất lớn thành hai thừa số nguyên tố. Hiện nay vẫn chưa có siêu máy tính nào có thể giải bài toán này với thời gian chấp nhận được, nhưng trong tương lai với máy tính lượng tử có thể sẽ khả thi.
- Quy trình mã hoá:
 - + Chọn hai số nguyên tố lớn p và q và tính $N = pq$. Cần chọn p và q sao cho $M < 2^{(i-1)} < N < 2^i$
 - + Tính $\Phi(n) = (p - 1)(q - 1)$
 - + Tìm một số e sao cho: $\{e \text{ và } \Phi(n) \text{ là 2 số cùng nhau và } 0 < e < \Phi(n)\}$
 - + Tìm một số d sao cho: $e.d \equiv 1 \pmod{\Phi(n)}$ (hay: $d = e^{-1} \pmod{\Phi(n)}$)
 - + Chọn khóa công khai $K1$ là cặp (e, N) , khóa riêng $K2$ là cặp (d, N) .
 - + Mã hoá $C = M^e \pmod{N}$, hoặc $C = M^d \pmod{N}$ nếu mã hoá chứng thực.
 - + Giải mã $M = C^d \pmod{N}$, hoặc $M = C^e \pmod{N}$ nếu chứng thực.

C. Các bước thực hiện và kết quả

The image shows a Visual Studio Code editor window with a Java file named `rsa.java`. The code implements an RSA encryption and decryption algorithm. The `main` method uses a `Scanner` to read a plaintext message, encrypts it using the `RSA` class, and then decrypts it back to the original message.

```
1 import java.math.BigInteger;
2 import java.security.SecureRandom;
3 import java.util.Scanner;
4
5 public class RSA {
6     private BigInteger p, q, n, e, d;
7
8     public RSA() {
9         int bits = 1024;
10        // tạo 2 số nguyên tố
11        p = BigInteger.probablePrime(bits / 2, new SecureRandom());
12        q = BigInteger.probablePrime(bits / 2, new SecureRandom());
13        // tính n và phi
14        n = p.multiply(q);
15        BigInteger phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
16        // tìm e
17        while(true){
18            e = BigInteger.probablePrime(bits / 2, new SecureRandom());
19            if(e.compareTo(phi) < 0 && e.gcd(phi).compareTo(BigInteger.ONE) == 0)
20                break;
21        }
22        d = e.modInverse(phi); // tìm d sao cho d*e=1(mod m)
23    }
24
25    public String encrypt(String message) {
26        BigInteger m = new BigInteger(message.getBytes());
27        BigInteger c = m.modPow(e, n);
28        return c.toString();
29    }
30
31    public String decrypt(String message) {
32        BigInteger c = new BigInteger(message);
33        BigInteger m = c.modPow(d, n);
34        return new String(m.toByteArray());
35    }
36
37    public static void main(String[] args) {
38        Scanner sc = new Scanner(System.in);
39        System.out.print("Plaintext: ");
40        RSA rsa = new RSA();
41        String message = sc.nextLine();
42        String encryptedMessage = rsa.encrypt(message);
43        String decryptedMessage = rsa.decrypt(encryptedMessage);
44        System.out.println("Encrypted message: " + encryptedMessage);
45        System.out.println("Decrypted message: " + decryptedMessage);
46    }
47 }
```

On the right side of the image, a web browser window is visible, showing a password reset page for a user named **Ninh Chí Hưởng**. The page includes a form for entering a new password and a "Hủy bỏ" (Cancel) button.

- Sử dụng thư viện BigInteger trong java để triển khai mã hóa RSA:
- + p, q sẽ là 2 số nguyên tố ngẫu nhiên
- + Hàm modInverse(): tính nghịch đảo của e trong modulo $\Phi(n)$
- Thuật toán mã hóa và giải mã: thông tin mã hóa cần phải được chuyển thành dạng byte vì các thông tin được truyền thường ở dạng chuỗi ký tự

The screenshot shows the Visual Studio Code editor with the file `RSA.java` open. The code defines a `RSA` class with methods for generating keys and encrypting a message. The web browser window on the right shows a login page for `B20DCAT094` with fields for username and password, and a 'Hủy bỏ' button.

```

4
5 public class RSA {
6     private BigInteger p, q, n, e, d;
7
8     public RSA() {
9         int bits = 1024;
10        // tạo 2 số nguyên tố
11        p = BigInteger.probablePrime(bits / 2, new SecureRandom());
12        q = BigInteger.probablePrime(bits / 2, new SecureRandom());
13        // tính n và phi n
14        n = p.multiply(q);
15        BigInteger phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
16        // tìm e
17        while(true){
18            e = BigInteger.probablePrime(bits / 2, new SecureRandom());
19            if(e.compareTo(phi) < 0 && e.gcd(phi).compareTo(BigInteger.ONE) == 0)
20                break;
21        }
22        d = e.modInverse(phi); // tìm d sao cho d*e=1(mod m)
23    }
24
25    public String encrypt(String message) {
26        BigInteger m = new BigInteger(message.getBytes());
27
28    }
29
30    public String decrypt(String message) {
31        BigInteger c = new BigInteger(message.getBytes());
32        BigInteger m = c.modPow(d, n);
33        return m.toString();
34    }
35
36    public static void main(String[] args) {
37        RSA rsa = new RSA();
38        String message = "RSA";
39        String encrypted = rsa.encrypt(message);
40        String decrypted = rsa.decrypt(encrypted);
41        System.out.println("Plaintext: " + message);
42        System.out.println("Encrypted message: " + encrypted);
43        System.out.println("Decrypted message: " + decrypted);
44    }
45
46 }

```

- Thử nghiệm với số lớn:

The screenshot shows the Visual Studio Code editor with the file `RSA.java` open. The code is the same as in the previous screenshot. The terminal window at the bottom shows the output of the program, including the plaintext, encrypted message, and decrypted message.

```

10 // tạo 2 số nguyên tố
11 p = BigInteger.probablePrime(bits / 2, new SecureRandom());
12 q = BigInteger.probablePrime(bits / 2, new SecureRandom());
13 // tính n và phi n
14 n = p.multiply(q);
15 BigInteger phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
16 // tìm e
17 while(true){
18     e = BigInteger.probablePrime(bits / 2, new SecureRandom());
19     if(e.compareTo(phi) < 0 && e.gcd(phi).compareTo(BigInteger.ONE) == 0)
20         break;
21 }
22 d = e.modInverse(phi); // tìm d sao cho d*e=1(mod m)
23
24
25 public String encrypt(String message) {
26     BigInteger m = new BigInteger(message.getBytes());
27     BigInteger c = m.modPow(e, n);
28     return c.toString();
29 }
30
31 public String decrypt(String message) {
32     BigInteger c = new BigInteger(message.getBytes());
33     BigInteger m = c.modPow(d, n);
34     return m.toString();
35 }
36
37 public static void main(String[] args) {
38     RSA rsa = new RSA();
39     String message = "RSA";
40     String encrypted = rsa.encrypt(message);
41     String decrypted = rsa.decrypt(encrypted);
42     System.out.println("Plaintext: " + message);
43     System.out.println("Encrypted message: " + encrypted);
44     System.out.println("Decrypted message: " + decrypted);
45 }
46
47 }

```

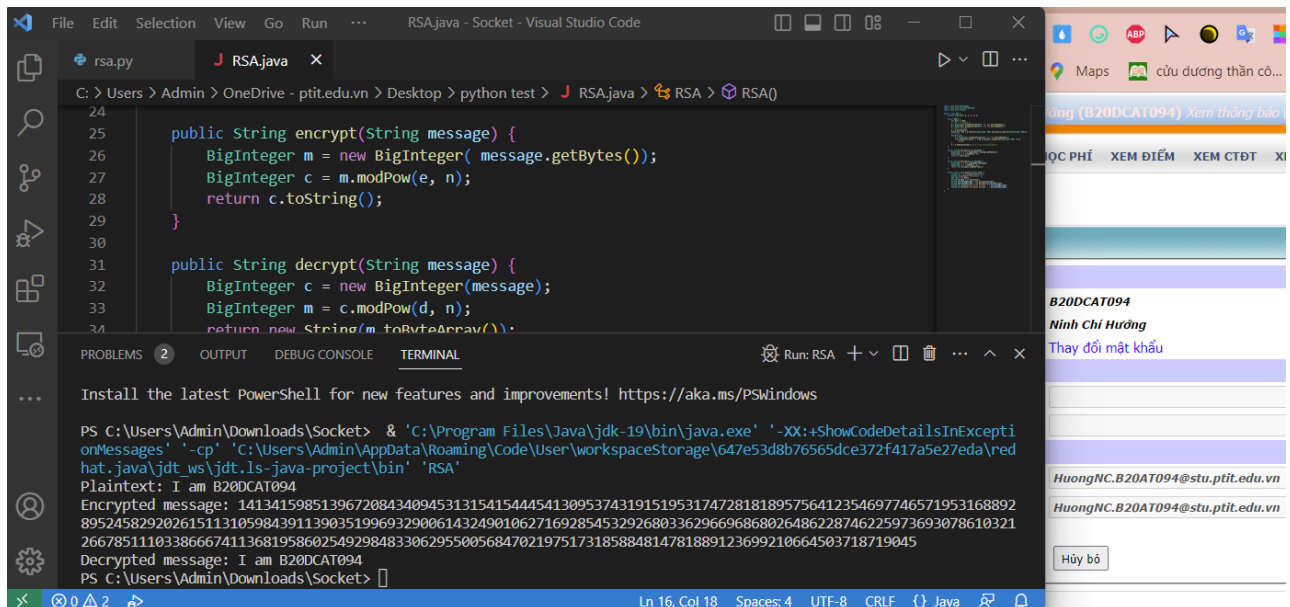
Terminal Output:

```

C:\Users\Admin\AppData\Local\Temp\Code\User\workspaceStorage\647e53db76565dce372f417a5e27eda\redhat_java\jdt_ws\jdt.ls-java-project\bin' 'RSA'
Plaintext: 64126918268641286482146734671246664
Encrypted message: 794676688220598997527033659801022016566096969961466845256198645905509007287929252474668176958768489448867841431601832164619
2817459910143623837558314121636201209030688659732207861238657817608284785805960306001509930381726326320328241328818567639448143818966495248239
97342478598978383110927482134425510432111
Decrypted message: 64126918268641286482146734671246664
PS C:\Users\Admin\Downloads\Socket>

```

- Thử nghiệm mã hóa và giải mã chuỗi ký tự: “I am B20DCAT094”



The screenshot displays a Visual Studio Code editor with a Java file named `RSA.java`. The code implements RSA encryption and decryption using `BigInteger`. The terminal window shows the execution of the program, which successfully encrypts the plaintext "I am B20DCAT094" into a long hexadecimal string and then decrypts it back to the original plaintext.

```
public String encrypt(String message) {
    BigInteger m = new BigInteger(message.getBytes());
    BigInteger c = m.modPow(e, n);
    return c.toString();
}

public String decrypt(String message) {
    BigInteger c = new BigInteger(message);
    BigInteger m = c.modPow(d, n);
    return new String(m.toByteArray());
}
```

Terminal Output:

```
PS C:\Users\Admin\Downloads\Socket> & 'C:\Program Files\Java\jdk-19\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Admin\AppData\Roaming\Code\User\workspaceStorage\647e53d8b76565dce372f417a5e27eda\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'RSA'
Plaintext: I am B20DCAT094
Encrypted message: 1413415985139672084340945313154154445413095374319151953174728181895756412354697746571953168892
89524582920261511310598439113903519969329006143249010627169285453292680336296696868026486228746225973693078610321
26678511103386667411368195860254929848330629550056847021975173185884814781889123699210664503718719045
Decrypted message: I am B20DCAT094
PS C:\Users\Admin\Downloads\Socket>
```

Tài liệu tham khảo :

- Triển khai thuật toán RSA bằng java : https://youtu.be/fDhBBu_y7L4
- Bài giảng Mật mã học cơ sở, thầy Đỗ Xuân Chợ.