

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG  
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**

|                             |                                |
|-----------------------------|--------------------------------|
| <b>Giảng viên hướng dẫn</b> | <b>: PGS.TS Hoàng Xuân Dậu</b> |
| <b>Sinh viên thực hiện</b>  | <b>: Nguyễn Nhật Minh</b>      |
| <b>Mã sinh viên</b>         | <b>: B21DCAT132</b>            |

**Hà Nội, tháng 3 năm 2024**

# Môn học: Thực tập cơ sở

## 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. Tìm hiểu lí thuyết

#### 2.1. Lập trình số lớn với các phép toán cơ bản

Lập trình số lớn là một lĩnh vực quan trọng trong khoa học máy tính và mật mã, nơi chúng ta thường cần làm việc với các số nguyên lớn vượt qua phạm vi của kiểu dữ liệu số nguyên thông thường trong ngôn ngữ lập trình. Các phép toán cơ bản trong lập trình số lớn bao gồm cộng, trừ, nhân, chia, và các phép toán modulo.

##### 1. Cộng và Trừ:

Khi thực hiện cộng hoặc trừ với các số nguyên lớn, chúng ta cần lặp qua từng chữ số và thực hiện phép toán từng cặp chữ số tương ứng. Trong trường hợp trừ, có thể cần kiểm tra và xử lý vay nợ (borrow) nếu chữ số ở cột trừ nhỏ hơn chữ số ở cột được trừ.

##### 2. Nhân:

Để nhân hai số nguyên lớn, chúng ta thường sử dụng phép nhân giống như phép nhân trong toán học cơ bản, nhưng thay vì chỉ thực hiện với các chữ số nhỏ, chúng ta thực hiện với toàn bộ các chữ số của hai số nguyên lớn. Kỹ thuật nhân cơ bản như nhân dài (long multiplication) hoặc nhân Karatsuba có thể được sử dụng để tối ưu hóa quá trình nhân.

##### 3. Chia:

Quá trình chia một số nguyên lớn cho một số nguyên khác là phức tạp hơn so với phép nhân.

Thuật toán chia cơ bản thực hiện tương tự như thuật toán chia trong toán học cơ bản, nhưng với các số nguyên lớn.

##### 4. Modulo:

Phép toán modulo (còn gọi là phép toán chia lấy dư) thường được sử dụng trong các thuật toán mật mã và kiểm tra tính toán.

Để tính toán modulo của các số nguyên lớn, chúng ta có thể sử dụng thuật toán chia hoặc sử dụng các thuật toán như thuật toán Euclidean để tính toán nhanh chóng hơn.

Trong lập trình thực tế, có thể sử dụng các thư viện số lớn được cung cấp sẵn trong các

ngôn ngữ lập trình như Python (như thư viện **gmpy2** hoặc **pycryptodome**) hoặc các ngôn ngữ như Java có hỗ trợ số lớn trong thư viện chuẩn. Tuy nhiên, việc hiểu cách thức hoạt động của các phép toán số lớn và cách triển khai chúng từ cơ bản đến phức tạp sẽ giúp bạn hiểu rõ hơn về cách hoạt động của các giải thuật và ứng dụng trong thực tế.

## 2.2. Giải thuật mã hóa khóa công khai RSA

Giải thuật mã hóa khóa công khai RSA là một phương pháp mã hóa và giải mã sử dụng một cặp khóa: một khóa công khai (public key) và một khóa bí mật (private key). Phương pháp này được đặt tên theo ba nhà khoa học Ron Rivest, Adi Shamir và Leonard Adleman, người đã phát triển nó vào năm 1977.

Quá trình cơ bản của giải thuật RSA bao gồm:

### - Khởi tạo khóa

Ta cần chọn 2 số nguyên tố lớn ngẫu nhiên  $p$  và  $q$

Tính  $n = p \cdot q$ ,  $n$  là modulo

Tính hàm Euler của  $n$ ,  $\phi(n) = (p-1)(q-1)$ .

### - Chọn khóa công khai (public key)

Chọn một số nguyên tố  $e$  thỏa mãn  $1 < e < \phi(n)$  và  $e$  nguyên tố cùng nhau với  $\phi(n)$

Public key: cặp số  $(e, n)$

### - Tính khóa bí mật (private key)

Tìm số nguyên  $d$  sao cho  $(e \cdot d - 1)$  chia hết cho  $\phi(n)$  hay  $d$  là nghịch đảo modulo của  $e$  với  $\phi(n)$

Private key là cặp số  $(d, n)$

### - Mã hóa

Đối với 1 thông điệp  $m$ , tính toán cipher text  $c$  bằng cách sử dụng khóa công khai:

$$C = m^e \bmod n$$

### - Giải mã

Đối với 1 ciphertext  $c$ , tính toán bản rõ plaintext bằng cách sử dụng khóa bí mật:  $m = c^d \bmod n$

Quan trọng nhất là việc chọn các số nguyên tố  $p$  và  $q$  đủ lớn để đảm bảo tính an toàn của hệ thống mã hóa RSA. Cũng cần lưu ý rằng các số nguyên tố này không được công khai, và việc tìm ra chúng từ  $n$  là một phần của bài toán phân tích  $n$  thành các thừa số nguyên tố, một bài toán rất khó trong trường hợp  $n$  đủ lớn.

## 3. Chuẩn bị môi trường

- Máy ảo hoặc môi trường chứa công cụ lập trình tùy ý

#### 4. Thực hành

##### - Lập trình thư viện số lớn và thử nghiệm

```

1  class LargeNumber:
2      def __init__(self, value):
3          self.value = value
4
5      def __add__(self, other):
6          return LargeNumber(self.value + other.value)
7
8      def __sub__(self, other):
9          return LargeNumber(self.value - other.value)
10
11     def __mul__(self, other):
12         return LargeNumber(self.value * other.value)
13
14     def __pow__(self, exponent):
15         return LargeNumber(self.value ** exponent)
16
17     def __mod__(self, modulo):
18         return LargeNumber(self.value % modulo.value)
19
20     def __eq__(self, other):
21         return self.value == other.value
22
23     def __lt__(self, other):
24         return self.value < other.value
25
26     def __str__(self):
27         return str(self.value)
28
29     def __repr__(self):
30         return repr(self.value)
31
32 # Phép toán chia sẽ không được triển khai vì chúng ta không cần trong RSA.
33 # Tạo các số lớn
34 num1 = LargeNumber(12345678901234567890)
35 num2 = LargeNumber(98765432109876543210)
36
37 # Thực hiện các phép toán
38 sum_result = num1 + num2
39 difference_result = num2 - num1
40 product_result = num1 * num2
41 power_result = num1 ** 2
42 mod_result = num2 % LargeNumber(123)
43
44 # In kết quả
45 print("Số 1:", num1)
46 print("Số 2:", num2)
47 print("Tổng:", sum_result)
48 print("Hiệu:", difference_result)
49 print("Tích:", product_result)
50 print("Lũy thừa:", power_result)
51 print("Modulo:", mod_result)
52

```

```

PROBLEMS 23 TERMINAL OUTPUT DEBUG CONSOLE PORTS
Python + v

PS C:\Users\nhatm\Downloads\python> & C:/Users/nhatm/AppData/Local/Programs/Python/Python311/python.exe c:/Users/nhatm/Downloads/python/test.py
Số 1: 12345678901234567890
Số 2: 98765432109876543210
Tổng: 1111111101111111100
Hiệu: 86419753208641975320
Tích: 1219326311370217952237463801111263526900
Lũy thừa: 152415787532388367501905199875019052100
Modulo: 66
PS C:\Users\nhatm\Downloads\python>

```

### - Lập trình giải thuật mã hóa và giải mã

Sử dụng hàm `generate_keypair()` để tạo cặp khóa, sau đó sử dụng khóa công khai để mã hóa một tin nhắn và khóa bí mật để giải mã nó.

```

import random
import math

class LargeNumber:
    def __init__(self, value):
        self.value = value

    def __add__(self, other):
        return LargeNumber(self.value + other.value)

    def __sub__(self, other):
        return LargeNumber(self.value - other.value)

    def __mul__(self, other):
        return LargeNumber(self.value * other.value)

    def __pow__(self, exponent):
        return LargeNumber(self.value ** exponent)

    def __mod__(self, modulo):
        return LargeNumber(self.value % modulo.value)

    def __eq__(self, other):
        return self.value == other.value

    def __lt__(self, other):
        return self.value < other.value

    def __str__(self):
        return str(self.value)

```

```

    def __repr__(self):
        return repr(self.value)

def is_prime(num):
    if num <= 1:
        return False
    if num <= 3:
        return True
    if num % 2 == 0 or num % 3 == 0:
        return False
    i = 5
    while i * i <= num:
        if num % i == 0 or num % (i + 2) == 0:
            return False
        i += 6
    return True

def generate_large_prime():
    while True:
        prime_candidate = random.getrandbits(10)
        if is_prime(prime_candidate):
            return prime_candidate

def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

def extended_gcd(a, b):
    if a == 0:
        return b, 0, 1
    gcd, x1, y1 = extended_gcd(b % a, a)
    x = y1 - (b // a) * x1
    y = x1
    return gcd, x, y

def mod_inverse(e, phi_n):
    gcd, x, y = extended_gcd(e, phi_n)
    if gcd != 1:
        raise ValueError("The provided numbers are not coprime")
    return x % phi_n

def generate_keypair():
    p = generate_large_prime()

```

```

    q = generate_large_prime()
    n = p * q
    phi_n = (p - 1) * (q - 1)
    e = 65537 # commonly used value for e
    d = mod_inverse(e, phi_n)
    return ((e, n), (d, n))

def encrypt(public_key, plaintext):
    e, n = public_key
    ciphertext = [pow(ord(char), e, n) for char in plaintext]
    return ciphertext

def decrypt(private_key, ciphertext):
    d, n = private_key
    plaintext = ''.join([chr(pow(char, d, n)) for char in ciphertext])
    return plaintext

# Thử nghiệm
public_key, private_key = generate_keypair()
print("Public key:", public_key)
print("Private key:", private_key)

message = "Hello, i am b21dcat132 nguyen nhat minh"
encrypted_message = encrypt(public_key, message)
print("Encrypted message:", encrypted_message)

decrypted_message = decrypt(private_key, encrypted_message)
print("Decrypted message:", decrypted_message)

```

- Sử dụng hàm `ord(char)` để chuyển đổi ký tự thành mã Unicode của nó, sau đó mã hóa các mã Unicode này bằng cách sử dụng khóa công khai. Khi giải mã, chúng ta sử dụng hàm `chr()` để chuyển mã Unicode trở lại ký tự ban đầu.
- Kết quả khi thử nghiệm với chuỗi: “hello, I am b21dcat132 nguyen nhat minh”

```

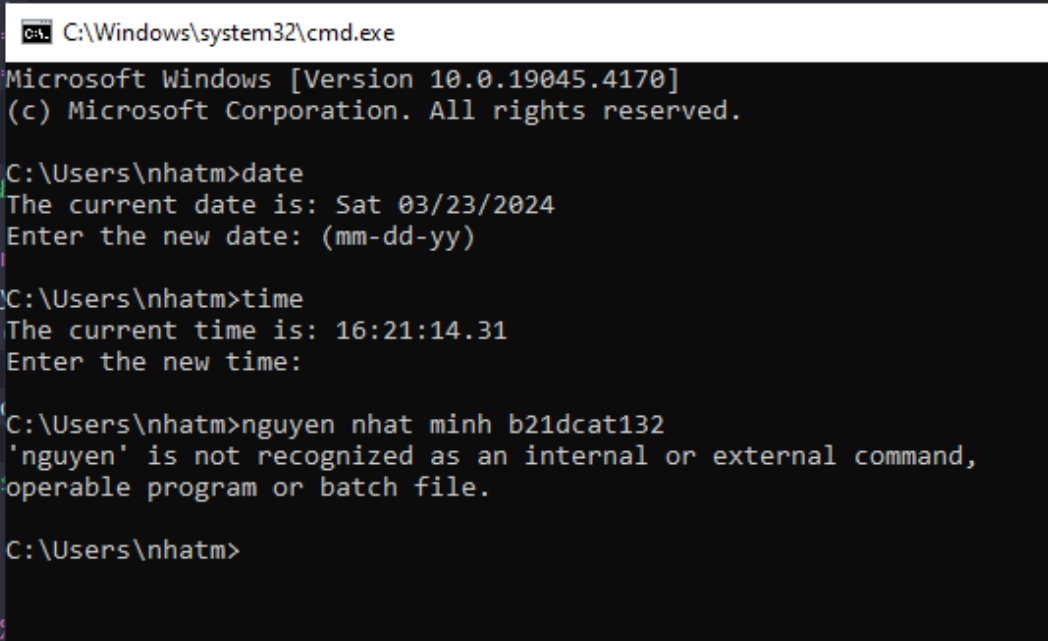
PS C:\Users\nhatm\Downloads\python> & C:/Users/nhatm/AppData/Local/Programs/Python/Python311/python.exe c:/Users/nhatm/Downloads/python/minh.py
Public key: (65537, 109493)
Private key: (8573, 109493)
Encrypted message: [69305, 50463, 6362, 6362, 106099, 4652, 63719, 45019, 63719, 991, 84940, 63719, 89706, 38111, 19219, 43257, 27177, 991, 108595, 19219, 51397, 38111, 63719, 5085, 62607, 36841, 17319, 50463, 5085, 63719, 5085, 6237, 991, 108595, 63719, 84940, 45019, 5085, 6237]
Decrypted message: Hello, i am b21dcat132 nguyen nhat minh
PS C:\Users\nhatm\Downloads\python>

```

## 5. Kết luận

- Bài thực hành hoàn thành vào ngày 23/03/2024

```
def gcd(a, b):  
    while b != 0:  
        a, b = b, a % b  
    return a  
  
def extended_gcd(a, b):  
    if a == 0:  
        return b, 0, 1  
    else:  
        g, x1, y1 = extended_gcd(b, a % b)  
        x = y1  
        y = x1 - (a // b) * y1  
        return g, x, y  
  
def mod_inverse(a, m):  
    g, x, y = extended_gcd(a, m)  
    if g != 1:  
        raise ValueError('a and m are not coprime')  
    return x % m
```



C:\Windows\system32\cmd.exe  
Microsoft Windows [Version 10.0.19045.4170]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\nhatm>date  
The current date is: Sat 03/23/2024  
Enter the new date: (mm-dd-yy)  
  
C:\Users\nhatm>time  
The current time is: 16:21:14.31  
Enter the new time:  
  
C:\Users\nhatm>nguyen nhat minh b21dcat132  
'nguyen' is not recognized as an internal or external command,  
operable program or batch file.  
  
C:\Users\nhatm>