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

BÁO CÁO BÀI THỰC HÀNH
HỌC PHẦN: THỰC TẬP CƠ SỞ
MÃ HỌC PHẦN: INT13147

BÀI THỰC HÀNH 4.2
LẬP TRÌNH THUẬT TOÁN MẬT MÃ HỌC

Sinh viên thực hiện: B22DCAT176 Nguyễn Thị Thùy Linh

Giảng viên hướng dẫn: PGS.TS Hoàng Xuân Dậu

HỌC KỲ 2 NĂM HỌC 2024-2025

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ tiếng Anh/Giải thích	Thuật ngữ tiếng Việt/Giải thích
SIEM	Security Information and Event Management	Giải pháp giám sát, phát hiện và phản ứng nhanh với mối đe dọa.
DNS	Domain Name System	Giao thức tên miền
DHCP	Dynamic Host Configuration Protocol	Giao thức cấp phát địa chỉ IP tự động
IIS	Internet Information Services	Dịch cụ Web Server chính của Microsoft
WAS	Windows Process Activation Service	Dịch vụ hỗ trợ cho IIS

CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH

I. 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.

II. Tìm hiểu lý thuyết

1. Lập trình số lớn với các phép toán cơ bản

Trong lập trình, kiểu dữ liệu nguyên như int, long trong C/C++ hay Java đều có giới hạn giá trị nhất định (thường từ 32-bit đến 64-bit). Tuy nhiên, trong một số bài toán thực tế như mã hóa RSA, tính toán tổ hợp lớn, số Fibonacci khổng lồ,... ta cần làm việc với những số nguyên vượt quá giới hạn này — gọi là số nguyên lớn (big integers).

Ngôn ngữ Python hỗ trợ xử lý số nguyên lớn một cách tự động, nhưng trong các ngôn ngữ như C/C++, Java hoặc trong môi trường yêu cầu tối ưu hiệu năng hoặc mật mã học, người lập trình thường phải tự xây dựng hoặc sử dụng thư viện hỗ trợ cho việc này.

Một số nguyên lớn không thể lưu trong một biến thông thường nên cần được biểu diễn dưới dạng mảng hoặc danh sách:

- Mỗi phần tử lưu một chữ số (hệ cơ số 10) hoặc một phần số trong cơ số lớn hơn (ví dụ cơ số

- Ví dụ: số 1234567890 có thể được biểu diễn trong mảng như [0, 9, 8, 7, 6, 5, 4, 3, 2, 1] (chữ số ngược lại, để dễ thao tác với chữ số hàng đơn vị trước).

Tùy cách thiết kế, có thể dùng:

- Mảng tĩnh (giới hạn kích thước tối đa),

- Danh sách động,

- Chuỗi ký tự (trong một số trường hợp đơn giản).

Các phép toán cơ bản

1. Cộng và Trừ

2. Nhân

   a. Nhân dài (Long Multiplication)

   b. Nhân Karatsuba

   c. Nhân Toom-Cook

3. Chia

a. Chia dài (Long Division)

b. Chia Newton-Raphson

c. Divide-and-Conquer Division.

4. Lũy thừa số nguyên lớn

a. Lũy thừa nhanh (Exponentiation by Squaring)

b. Modular Exponentiation (rất quan trọng trong mật mã học)

5. Tìm Ước Chung Lớn Nhất (GCD)

a. Thuật toán Euclid

b. Thuật toán Euclid mở rộng

c. Binary GCD

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 = p*q. Cần chọn p và q sao cho M < 2^(i-1) < N < 2^i
- Tính Φ(n) = (p - 1)(q - 1)
- Tìm một số e sao cho: e và Φ(n) là 2 số nguyên tố cùng nhau và 0 < e < Φ(n)
- Tìm một số d sao cho: e.d mod Φ(n) = 1 (hay: d = e^(-1) mod Φ(n))
- Chọn khóa công khai K1 là cặp (N, e), khóa riêng K2 là cặp (N, d).
- Mã hoá C = M^e mod N, hoặc C = M^d mod N nếu mã hoá chứng thực.
- Giải mã M = C^d mod N, hoặc M = C^e mod N nếu chứng thực

CHƯƠNG 2 : NỘI DUNG THỰC HÀNH

I. Chuẩn bị môi trường

Môi trường lập trình theo mong muốn

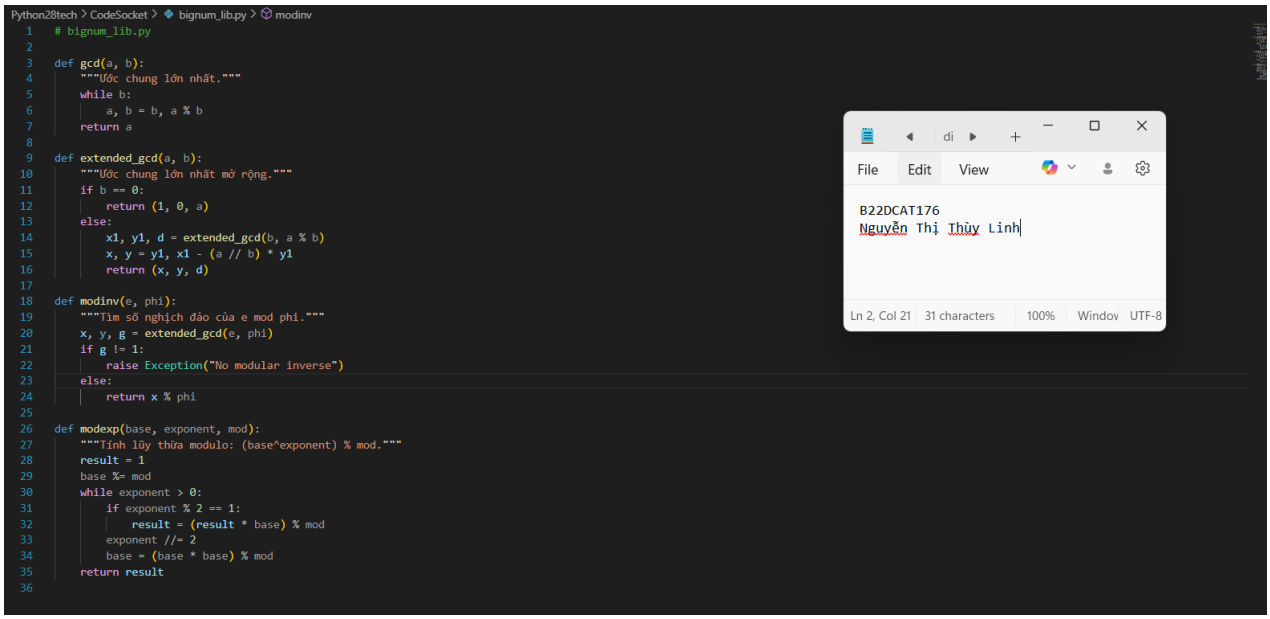
II. Các bước thực hiện

Lập trình thư viện số lớn với các phép toán cơ bản để sử dụng trong giải thuật mã hóa/ giải mã RSA

Các hàm xử lý số nguyên lớn:

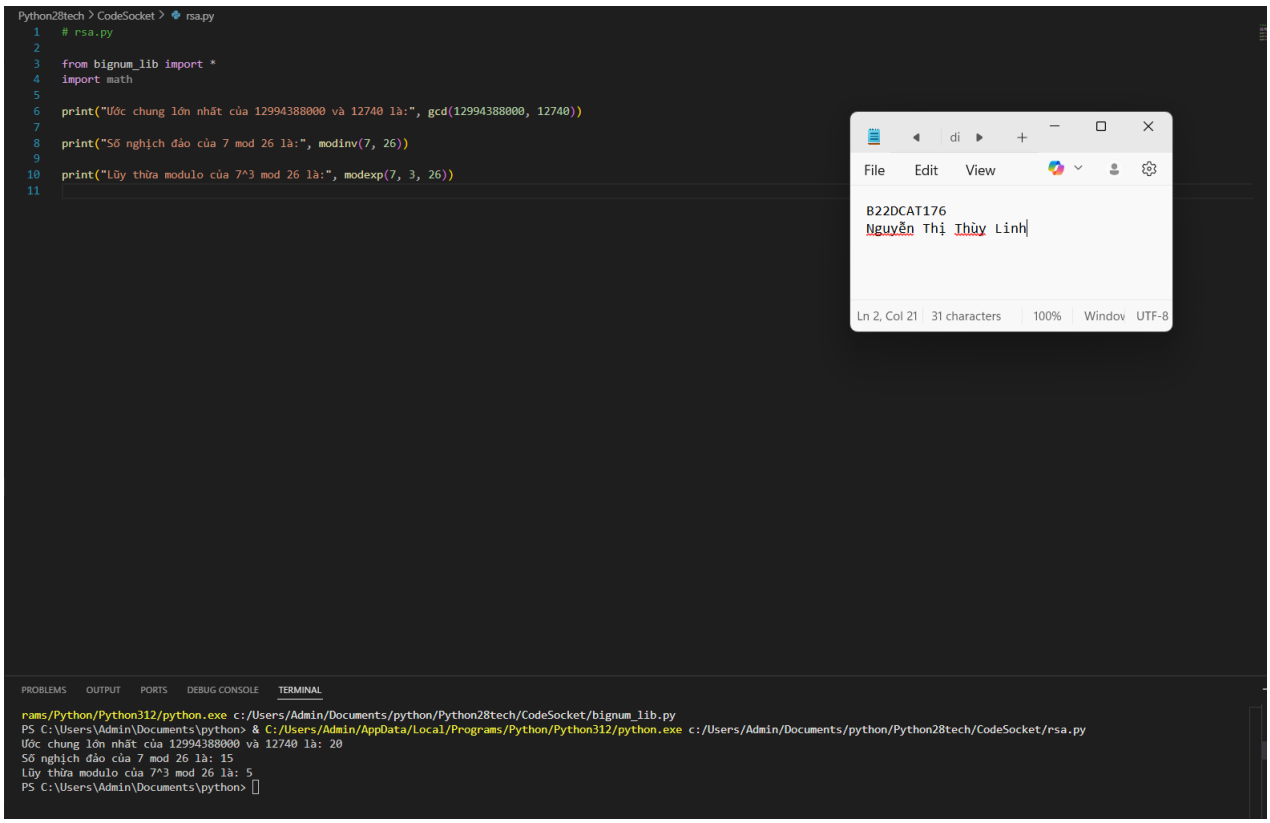
- *Gcd*: Tìm ước chung lớn nhất của 2 số
- *Extended_gcd*: hàm gcd được mở rộng, không chỉ để tìm ước chung lớn nhất (GCD) của 2 số nguyên a và b, mà còn tìm ra các hệ số nguyên x và y sao cho:

  a.x + b.y = gcd(a,b)
- *Modinv*: sử dụng hàm gcd mở rộng để tìm số nghịch đảo modulo của e mod phi, trả về x, y là ước chung lớn nhất g của e và phi
- *Modexp*: Tính lũy thừa modulo



Hình ảnh 1: Tạo thư viện để sử dụng các hàm tính toán

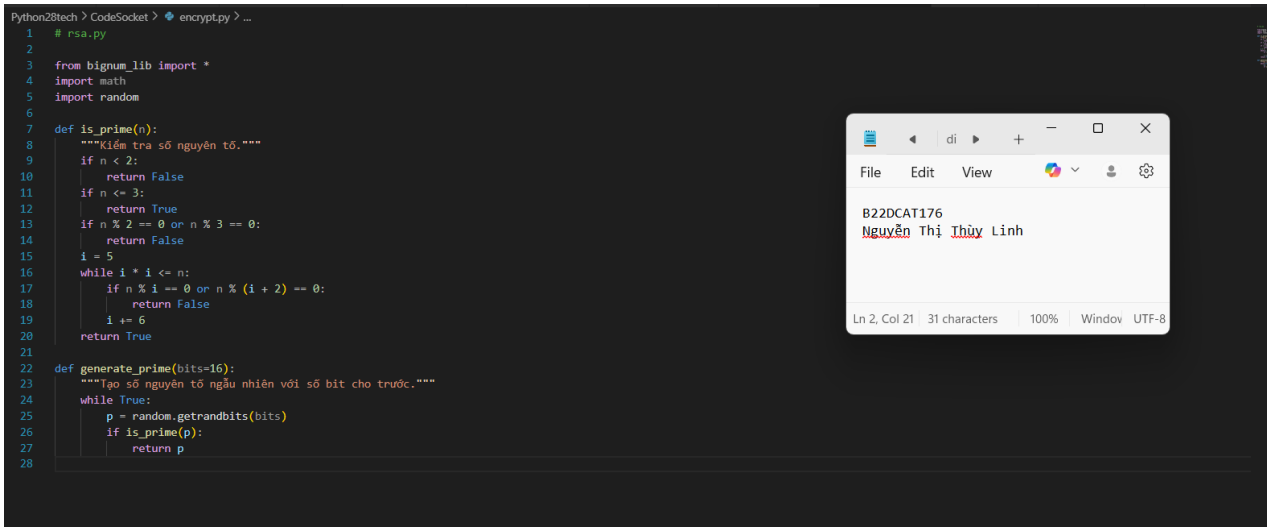
Thử nghiệm chứng minh thư viện hoạt động tốt với các ví dụ phép toán cho số lớn



Hình ảnh 2: Ví dụ thử nghiệm các hàm tính toán số lớn

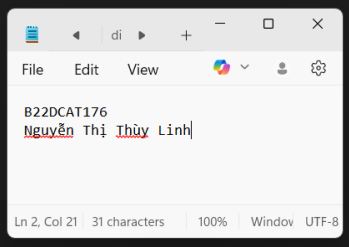
Lập trình giải thuật mã hóa và giải mã

- *is_prime*: Hàm kiểm tra số nguyên tố
- *generate_prime*: Hàm tạo số nguyên tố ngẫu nhiên
- *generate_keypair*: Hàm tạo cặp khóa công khai và khóa bí mật cho thuật toán rsa. Tạo ra 2 số nguyên tố p và q, sau đó tính toàn n = p*q và phi = (p – 1) * (q – 1). Khóa công khai là (e,n) và khóa bí mật là (d,n)
- *encrypt*: Hàm mã hóa thông điệp sử dụng khóa công khai pk, chuyển đổi từng ký tự trong thông điệp thành mã ASCII sau đó tính toàn lũy thừa modulo. Trả về danh sách số nguyên đại diện cho thông điệp
- *decrypt*: Hàm giải mã thông điệp sử dụng khóa bí mật pk, tính toán lũy thừa modulo n cho từng số sau đó chuyển lại thành mã ASCII. Trả về thông điệp



Hình ảnh 3: Hàm kiểm tra số nguyên tố và tạo số nguyên tố.

```python
def generate_keypair(bits=16):
    """Tạo cặp khóa công khai và khóa bí mật."""
    p = generate_prime(bits)
    q = generate_prime(bits)
    while p == q:
        q = generate_prime(bits)

    n = p * q
    phi = (p - 1) * (q - 1)
    e = 65537  # giá trị phổ biến cho e
    while gcd(e, phi) != 1:
        e = random.randrange(2, phi)

    d = modinv(e, phi)
    return ((e, n), (d, n))

def encrypt(pk, plaintext):
    """Mã hóa thông điệp."""
    key, n = pk
    cipher = [modexp(ord(char), key, n) for char in plaintext]
    return cipher

def decrypt(pk, ciphertext):
    """Giải mã thông điệp."""
    key, n = pk
    plain = [chr(modexp(char, key, n)) for char in ciphertext]
    return ''.join(plain)
```



Hình ảnh 4: Hàm tạo khóa, mã hóa, giải mã

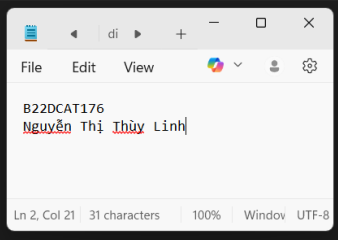
Thử nghiệm mã hóa và giải mã chuỗi ký tự:

Để đơn giản hóa và tránh số quá lớn, ta lựa chọn khóa công khai và khóa bí mật được tạo với : p = 61, q = 59

“Hello, I’m B22DCAT176” có 21 kí tự được mã hóa thành danh sách các số “Mã hóa”

Sau khi giải mã, thông điệp vẫn nguyên vẹn.

```python
from rsa import *

plaintext = "Hello, I'm B22DCAT176"

# p = 61
# q = 59

public, private = generate_keypair()
print("Khóa công khai:", public)
print("Khóa bí mật:", private)

cipher = encrypt(public, plaintext)
print("Mã hóa:", cipher)

decrypted = decrypt(private, cipher)
print("Giải mã:", decrypted)
```




```
PROBLEMS  OUTPUT  PORTS  DEBUG CONSOLE  TERMINAL
Ước chung lớn nhất của 12994388000 và 12740 là: 20
Số nghịch đảo của 7 mod 26 là: 15
Lũy thừa modulo của 7^3 mod 26 là: 5
Khóa công khai: (65537, 3328595111)
Khóa bí mật: (153785441, 3328595111)
Mã hóa: [3156327018, 1022660548, 769877151, 769877151, 2001614246, 459779059, 249156710, 2521594452, 2000767663, 986146150, 249156710, 1923442627, 828416564, 828416564, 2300869850, 2244815172
, 2596052905, 1633423223, 877887980, 2141597033, 2968107465]
Giải mã: Hello, I'm B22DCAT176
```

Hình ảnh 5: Kết quả thử nghiệm

TÀI LIỆU THAM KHẢO

[1] Phạm Hoàng Duy, Bài giảng Hệ điều hành Windows và Linux/Unix, Học viện Công Nghệ Bưu Chính Viễn Thông, 2016.

[2] Tom Carpenter, Microsoft Windows Server Operating System Essentials, Sybex, 2011.

[3] Wale Soyinka, Linux Administration A Beginners Guide, McGraw-Hill Osborne Media, 2012.