

**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:

B22DCAT034 Trương Quốc Bảo

Giảng viên hướng dẫn: TS. Đinh Trường Duy

HỌC KỲ 2 NĂM HỌC 2024-2025

MỤC LỤC

MỤC LỤC.....	2
DANH MỤC CÁC HÌNH VẼ.....	3
DANH MỤC CÁC TỪ VIẾT TẮT.....	4
CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH	5
1.1 Mục đích.....	5
1.2 Tìm hiểu lý thuyết	5
1.2.1 Lập trình số lớn với các phép toán cơ bản	5
1.2.2 Giải thuật mật mã khóa công khai RSA	6
CHƯƠNG 2. NỘI DUNG THỰC HÀNH	8
2.1 Chuẩn bị môi trường	8
2.2 Các bước thực hiện.....	8
TÀI LIỆU THAM KHẢO	11

DANH MỤC CÁC HÌNH VẼ

Hình 1 Tạo thư viện để sử dụng các hàm tính toán.....	8
Hình 2 Ví dụ thử nghiệm các hàm tính toán số lớn	9
Hình 3 Hàm kiểm tra số nguyên tố và tạo số nguyên tố	9
Hình 4 Hàm tạo khóa, mã hóa, giải mã.....	10
Hình 5 Kết quả thử nghiệm	10

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ tiếng Anh/Giải thích	Thuật ngữ tiếng Việt/Giải thích
GCD	Greatest Common Divisor	Ước số chung lớn nhất
KU	Public Key	Khóa công khai
KR	Private Key	Khóa riêng

CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH

1.1 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.

1.2 Tìm hiểu lý thuyết

1.2.1 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 256)
- 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

- Nhân dài (Long Multiplication)
- Nhân Karatsuba
- Nhân Toom-Cook

3. Chia

- Chia dài (Long Division)
- Chia Newton-Raphson
- 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

1.2.2 Giải thuật mật mã khóa công khai RSA

Thuật toán mã hóa RSA là một thuật toán mã hóa khóa công khai. Thuật toán RSA được xây dựng bởi các tác giả Ron Rivest, Adi Shamir và Len Adleman tại học viện MIT vào năm 1977, và ngày nay đang được sử dụng rộng rãi. Về mặt tổng quát thuật toán RSA là một phương pháp mã hóa theo khối. Trong đó thông điệp M và bản mã C là các số nguyên từ 0 đến 2^i với i số bit của khối. Kích thước thường dùng của i là 1024 bit. Thuật toán RSA sử dụng hàm một chiều là vấn đề phân tích một số thành thừa số nguyên tố và bài toán Logarith rời rạc.

Nguyên tắc thực hiện quá trình mã hóa và giải mã thuật toán RSA :

Quá trình mã hóa:

Để thực hiện mã hóa và giải mã, thuật toán RSA dùng phép lũy thừa modulo của lý thuyết số. Các bước thực hiện như sau: -

- Chọn hai số nguyên tố lớn p và q và tính $N = pq$. Cần chọn p và q sao cho: $M < 2^{i-1} < N < 2^i$ với i là chiều dài bản rõ. - - - - -
- Tính $\Phi(n) = (p - 1)(q - 1)$
- Tìm một số e sao cho: $\{e \text{ và } \Phi(n) \text{ là 2 số nguyên tố cùng nhau và } 0 < e < \Phi(n)\}$
- Tìm một số d sao cho: $e \cdot d \equiv 1 \pmod{\Phi(n)}$ (hay: $d = e^{-1} \pmod{\Phi(n)}$) (d là nghịch đảo của e trong phép modulo $\Phi(n)$)
- Chọn khóa công khai KU là cặp (e, N) , khóa riêng KR là cặp (d, N)

Việc mã hóa thực hiện theo công thức:

- Theo phương án 1, mã hóa: $C = E(M, KU) = M^e \pmod{N}$
- Theo phương án 2, mã hóa chứng thực: $C = E(M, KR) = M^d \pmod{N}$

Quá trình giải mã:

Việc giải mã thực hiện theo công thức:

- Theo phương án 1, giải mã: $M = D(C, KR) = C^d \pmod{N}$
- Theo phương án 2, mã hóa chứng thực: $M = D(C, KU) = C^e \pmod{N}$

Thông điệp M có kích thước $i-1$ bit, bản mã C có kích thước i bit.

Tính đúng của thuật toán RSA:

Giả sử bản mã chính là thông điệp

$M = M$. Xét phương án 1: - Từ bước 4 suy ra được $e.d = k.n + 1$ với k là một số nào đó.

- Vậy $M = C^d \bmod N$

$$= M^{ed} \bmod N$$

$$= M^{kn+1} \bmod N$$

$$= M^{k(p-1)(q-1)+1} \bmod N$$

Điều này có nghĩa là RSA đúng khi và chỉ khi $M^{k(p-1)(q-1)+1} \bmod N \equiv M \bmod p$. Xét hai khả năng của M :

- M chia hết cho p khi đó $M \bmod p = 0$ do đó $M^{k(p-1)(q-1)+1} \bmod p \equiv M \bmod p = 0$
- M không chia hết cho p : Vì p là số nguyên tố nên M và p là hai số nguyên tố cùng nhau. Vậy: $M^{k(p-1)(q-1)+1} \bmod p = M^{(Mp-1)(q-1)k} \bmod p = M^{1k(q-1)} \bmod p$ (theo định lý Fermat) $= M \bmod p$

Vậy: $M^{k(p-1)(q-1)+1} - M \equiv 0 \bmod p \equiv M$. Hay nói cách khác $M^{k(p-1)(q-1)+1} - M$ chia hết cho p . Vì q, p là hai số nguyên tố nên suy ra được $M^{k(p-1)(q-1)+1} - M$ chia hết cho $N = p.q$. Như vậy:

$$M^{k(p-1)(q-1)+1} \equiv M \bmod N$$

$$\equiv M = M^{k(p-1)(q-1)+1} \bmod N = M \text{ (do } M < N \text{)}$$

Vì e và d đối xứng nên có thể thấy trong phương án 2, cũng có $M = M$. - Không thể suy ra KR từ KU , nghĩa là tìm cặp (d, N) từ cặp (e, N) : Có e và N , muốn tìm d , phải dựa vào công thức: $e.d \equiv 1 \bmod n$. Do đó phải tính được n .

Vì $\Phi(n) = (p-1)(q-1)$ nên suy ra phải tính được p và q . Vì $N = pq$ nên chỉ có thể tính được p và q từ N . Tuy nhiên điều này là bất khả thi vì $N = pq$ là hàm một chiều. Vậy không thể tính được KR từ KU

CHƯƠNG 2. NỘI DUNG THỰC HÀNH

2.1 Chuẩn bị môi trường

Môi trường lập trình theo mong muốn.

2.2 Các bước thực hiện

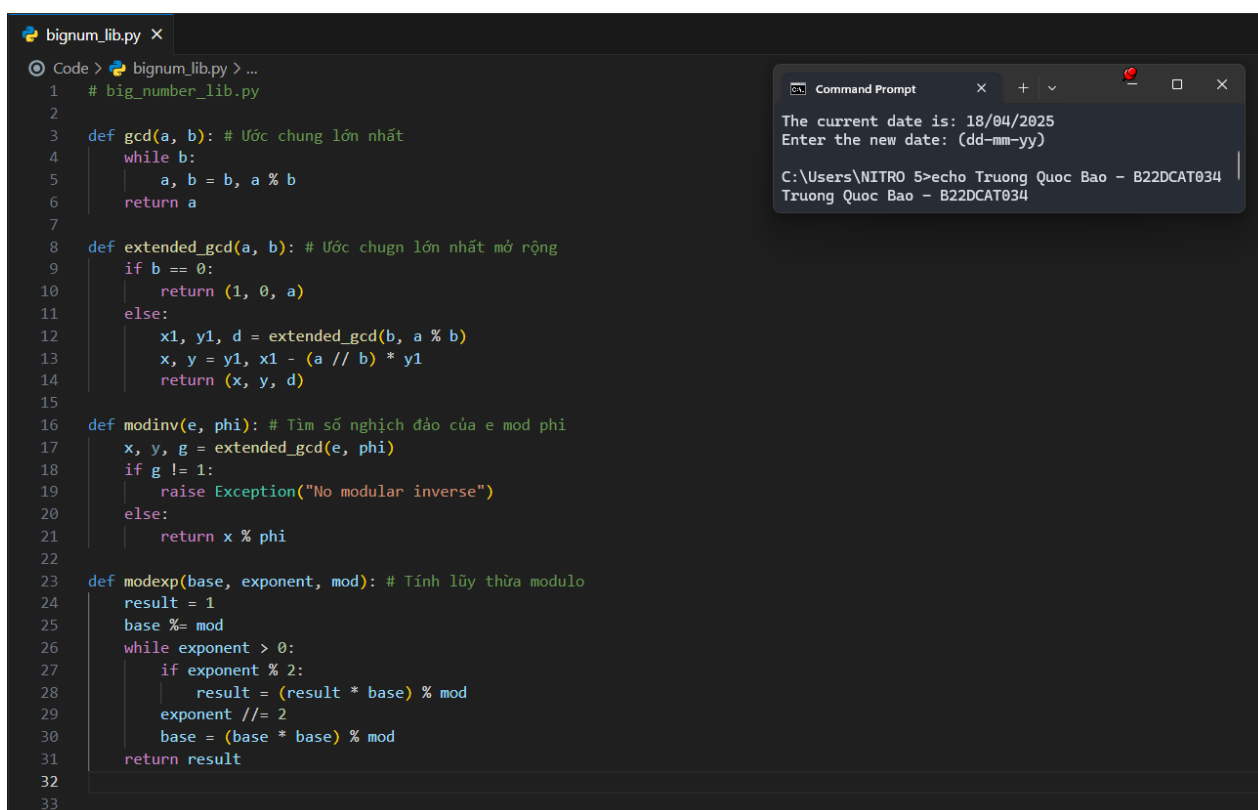
Lập trình thư viện số lớn với các phép toán cơ bản để sử dụng trong giải thuật mã hóa/giải mã RSA

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 hai số nguyên a và b, mà còn tìm ra các hệ số nguyên x và y sao cho:

$$a.x + b.y = \text{gcd}(a,b)$$

- 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



```
bignum_lib.py x
Code > bignum_lib.py > ...
1 # big_number_lib.py
2
3 def gcd(a, b): # Ước chung lớn nhất
4     while b:
5         a, b = b, a % b
6     return a
7
8 def extended_gcd(a, b): # Ước chung lớn nhất mở rộng
9     if b == 0:
10        return (1, 0, a)
11    else:
12        x1, y1, d = extended_gcd(b, a % b)
13        x, y = y1, x1 - (a // b) * y1
14        return (x, y, d)
15
16 def modinv(e, phi): # Tìm số nghịch đảo của e mod phi
17     x, y, g = extended_gcd(e, phi)
18     if g != 1:
19         raise Exception("No modular inverse")
20     else:
21         return x % phi
22
23 def modexp(base, exponent, mod): # Tính lũy thừa modulo
24     result = 1
25     base %= mod
26     while exponent > 0:
27         if exponent % 2:
28             result = (result * base) % mod
29             exponent //= 2
30             base = (base * base) % mod
31     return result
32
33
```

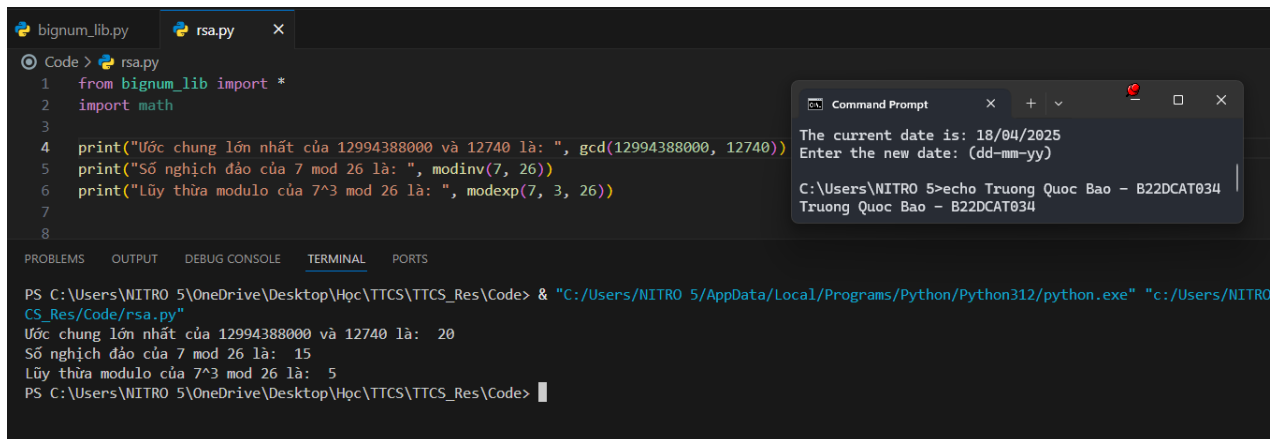
Command Prompt

The current date is: 18/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\NITRO S>echo Truong Quoc Bao - B22DCAT034
Truong Quoc Bao - B22DCAT034

Hình 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



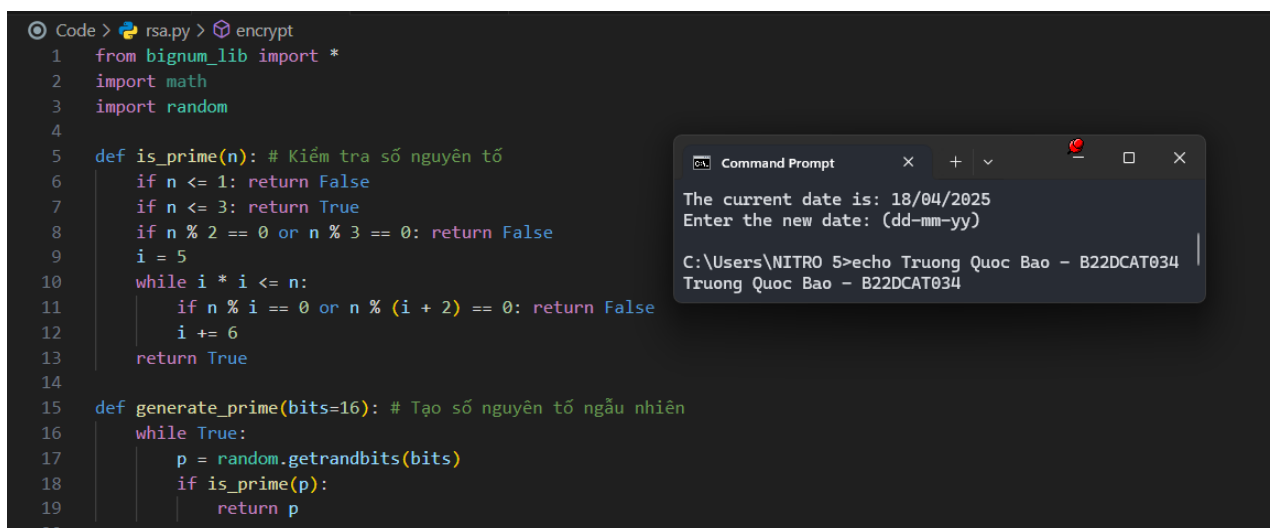
```
bignum_lib.py  rsa.py  X
Code > rsa.py
1 from bignum_lib import *
2 import math
3
4 print("Ước chung lớn nhất của 12994388000 và 12740 là: ", gcd(12994388000, 12740))
5 print("Số nghịch đảo của 7 mod 26 là: ", modinv(7, 26))
6 print("Lũy thừa modulo của 7^3 mod 26 là: ", modexp(7, 3, 26))
7
8
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\NITRO 5\OneDrive\Desktop\Hoc\TTCS\TTCS_Res\Code> & "C:/Users/NITRO 5/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/NITRO 5/OneDrive/Desktop/Hoc/TTCS/TTCS_Res/Code/rsa.py"
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
PS C:\Users\NITRO 5\OneDrive\Desktop\Hoc\TTCS\TTCS_Res\Code>

Command Prompt
The current date is: 18/04/2025
Enter the new date: (dd-mm-yy)
C:\Users\NITRO 5>echo Truong Quoc Bao - B22DCAT034
Truong Quoc Bao - B22DCAT034
```

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

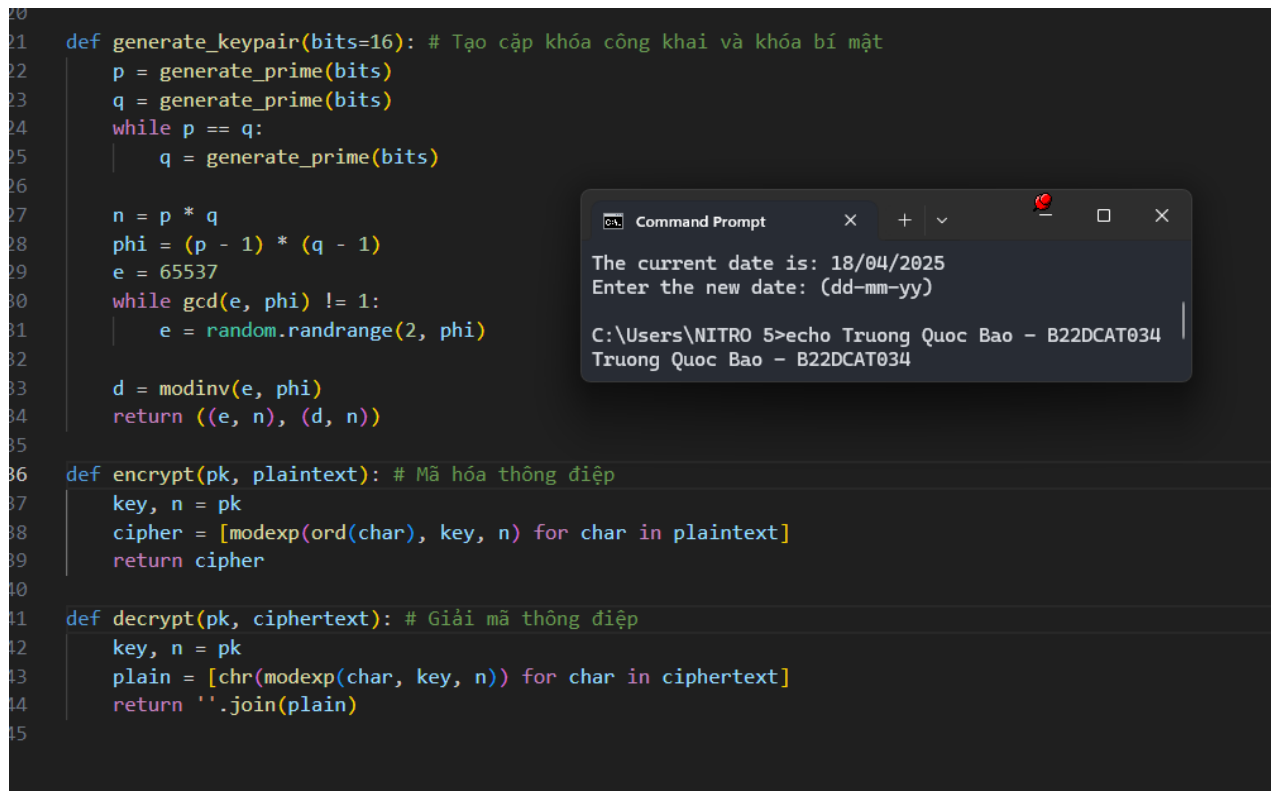
Lập trình giải thuật mã hóa và giải mã

- 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 toán $n = p * q$ 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 toán 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



```
Code > rsa.py  encrypt
1 from bignum_lib import *
2 import math
3 import random
4
5 def is_prime(n): # Kiểm tra số nguyên tố
6     if n <= 1: return False
7     if n <= 3: return True
8     if n % 2 == 0 or n % 3 == 0: return False
9     i = 5
10    while i * i <= n:
11        if n % i == 0 or n % (i + 2) == 0: return False
12        i += 6
13    return True
14
15 def generate_prime(bits=16): # Tạo số nguyên tố ngẫu nhiên
16     while True:
17         p = random.getrandbits(bits)
18         if is_prime(p):
19             return p
20
```

Hình 3 Hàm kiểm tra số nguyên tố và tạo số nguyên tố



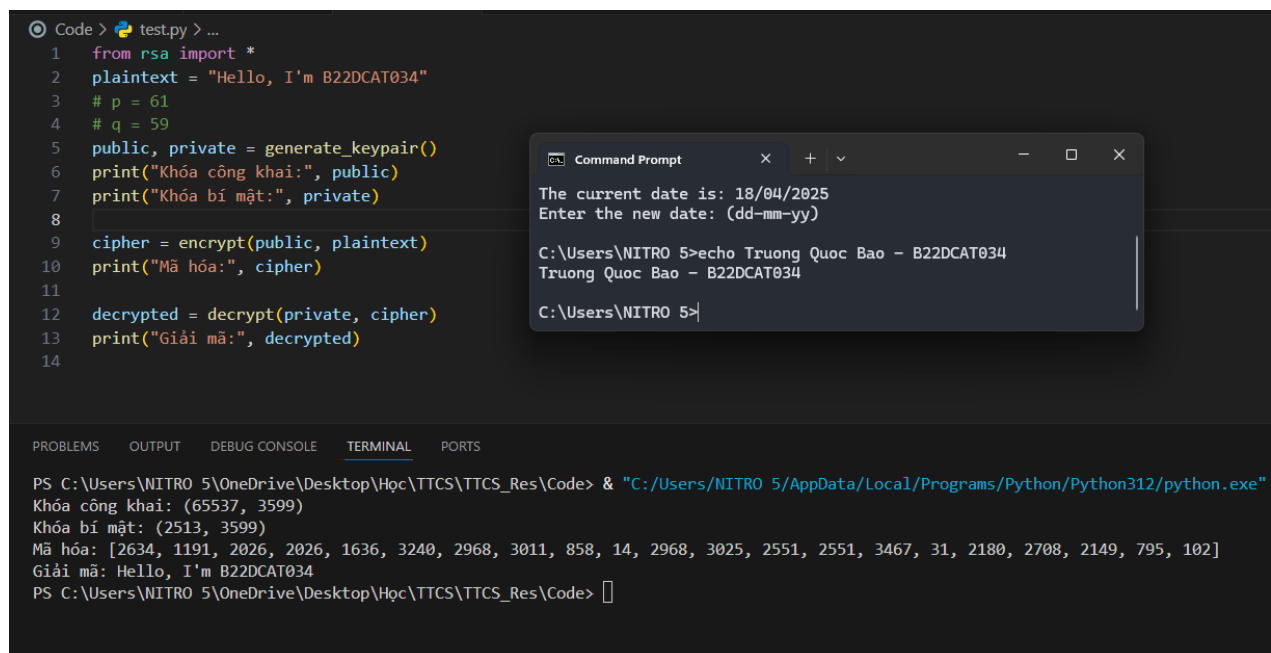
Hình 4 Hàm tạo khóa, mã hóa, giải mã

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 B22DCAT034” có 21 ký tự được mã hóa thành danh sách các số “Mã hóa”

Sau khi giải mã, thông điệp vẫn nguyên vẹn.



Hình 5 Kết quả thử nghiệm

TÀI LIỆU THAM KHẢO

- [1] Đinh Trường Duy, Phạm Hoàng Duy, Bài giảng Hệ điều hành Windows và Linux/Unix, Học viện Công Nghệ Bưu Chính Viễn Thông, 2022.
- [2] Tom Carpenter, Microsoft Windows Server Operating System Essentials, Sybex, 2011.
- [3] Đỗ Xuân Chợt, Bài giảng Mật mã học cơ sở, Học viện Công Nghệ Bưu Chính Viễn Thông, 2021