

**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:

B22DCAT063 Lê Tiến Dương

Giảng viên hướng dẫn: PGS. TS. Hoàng Xuân Dậu

**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
<b>CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH</b> .....	5
1.1 Mục đích.....	5
1.2 Tìm hiểu lý thuyết .....	5
<b>1.2.1</b> Tìm hiểu về lập trình số lớn với các phép toán cơ bản .....	5
<b>1.2.2</b> Tìm hiểu về giải thuật mật mã khóa công khai RSA .....	6
<b>CHƯƠNG 2. NỘI DUNG THỰC HÀNH</b> .....	9
2.1 Chuẩn bị môi trường .....	9
2.2 Các bước thực hiện.....	9
<b>2.2.1</b> Lập trình thư viện số lớn với các phép toán cơ bản, thử nghiệm chứng minh thư viện hoạt động tốt với các ví dụ phép toán cho số lớn.....	9
<b>2.2.2</b> Lập trình giải thuật mã hóa và giải mã.....	11
<b>2.2.3</b> Thử nghiệm mã hóa và giải mã chuỗi ký tự: “I am B22DCAT063” .....	12
<b>TÀI LIỆU THAM KHẢO</b> .....	14

## DANH MỤC CÁC HÌNH VẼ

Hình 1 – Hàm tính ước chung lớn nhất .....	9
Hình 2 – Hàm tính $a^b \bmod m$ .....	10
Hình 3 – Hàm tìm số nghịch đảo d sao cho $d.e \bmod m = 1$ .....	10
Hình 4 – Hàm sinh khóa, để đơn giản ta chọn $e = 65537$ .....	11
Hình 5 – Hàm mã hóa và giải mã.....	11
Hình 6 – Thử nghiệm code với mã hóa và giải mã .....	12
Hình 7 – Kết quả chạy chương trình - 1 .....	12
Hình 8 – Kết quả chạy chương trình - 2 .....	13

## DANH MỤC CÁC TỪ VIẾT TẮT

<b>Từ viết tắt</b>	<b>Thuật ngữ tiếng Anh/Giải thích</b>	<b>Thuật ngữ tiếng Việt/Giải thích</b>
RSA	Rivest–Shamir–Adleman	Một thuật toán mã hóa khóa công khai rất nổi tiếng, dùng để mã hóa dữ liệu và tạo chữ ký số.
GCD	Greatest Common Divisor	Ước chung lớn nhất.
SSL/TLS	Secure Sockets Layer / Transport Layer Security	Giao thức bảo mật được dùng để mã hóa dữ liệu truyền qua internet.

## 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 Tìm hiểu về lập trình số lớn với các phép toán cơ bản

Một số lý thuyết cơ bản về lập trình số nguyên lớn:

- **Biểu diễn số nguyên lớn:** Trong lập trình số nguyên lớn, số này thường được biểu diễn dưới dạng mảng hoặc danh sách các chữ số. Mỗi phần tử trong mảng hoặc danh sách này thường lưu trữ một chữ số của số nguyên.
- **Cộng và trừ:**
  - Các phép toán cộng và trừ có thể được thực hiện bằng cách duyệt qua từng chữ số của hai số và thực hiện phép toán tương ứng từng cặp chữ số, kèm theo việc xử lý nhớ và mượn nếu cần.
  - Bắt đầu từ chữ số đơn vị và tiếp tục lần lượt đến chữ số hàng đơn vị, thực hiện phép toán cộng hoặc trừ tương ứng.
- **Nhân:**
  - Phép nhân hai số nguyên lớn có thể thực hiện bằng thuật toán nhân dài (long multiplication) hoặc các thuật toán nhân nhanh hơn như nhân Karatsuba hoặc nhân Toom-Cook.
  - Thuật toán nhân dài thực hiện phép nhân tương tự như cách thực hiện phép nhân trong toán học cơ bản: duyệt qua từng chữ số của một số và nhân với từng chữ số của số kia, sau đó cộng các kết quả lại.
- **Chia:**
  - Phép chia hai số nguyên lớn có thể thực hiện bằng thuật toán chia dài (long division) hoặc các thuật toán chia hiệu quả hơn như thuật toán chia như thuật toán chia nhân bán (divide-and-conquer division) hoặc thuật toán chia nhưng tốt hơn (Newton-Raphson division).
  - Tương tự như phép nhân, thuật toán chia dài duyệt qua từng chữ số của số chia và thực hiện phép chia tương ứng.
- **Tính lũy thừa:** Phép tính lũy thừa của một số nguyên lớn có thể được thực hiện bằng thuật toán lũy thừa nhanh (exponentiation by squaring). Thuật toán này giảm thiểu số lượng phép toán bằng cách phân tích số mũ thành các phần nhỏ và tính lũy thừa của từng phần.

- **Tìm ước chung lớn nhất (GCD):** Phép tìm ước chung lớn nhất của hai số nguyên lớn có thể được thực hiện bằng thuật toán Euclid mở rộng (extended Euclidean algorithm) hoặc thuật toán nhị phân (binary GCD algorithm). Cả hai thuật toán này đều dựa trên việc lặp đi lặp lại một số phép chia và tính toán dựa trên phần dư.

Trong lập trình số nguyên lớn, việc tối ưu hóa và cải thiện hiệu suất là rất quan trọng. Điều này bao gồm việc sử dụng các thuật toán hiệu quả nhất để thực hiện các phép toán cơ bản và giải quyết các vấn đề liên quan đến hiệu năng và bộ nhớ.

Trong Python, số nguyên không bị giới hạn về giá trị. Tuy nhiên, số nguyên có thể lớn đến mức nào phụ thuộc vào bộ nhớ có sẵn trên hệ thống mà bạn đang sử dụng. Chính vì vậy, ta có thể xử lý số nguyên lớn dễ dàng với Python

### **1.2.2 Tìm hiểu về giải thuật mật mã khóa công khai RSA**

#### **1.2.2.1 Giải thuật mã hóa khóa công khai**

*Mã hóa khóa bất đối xứng* (Hay còn gọi là mã hóa khóa công khai): là phương pháp mã hóa mà khóa sử dụng để mã hóa sẽ khác với khóa dùng để giải mã. Khóa dùng để mã hóa gọi là khóa công khai và khóa dùng để giải mã gọi là khóa bí mật. Tất cả mọi người đều có thể biết được khóa công khai (kể cả hacker), và có thể dùng khóa công khai để mã hóa thông tin. Nhưng chỉ có người nhận mới nắm giữ khóa bí mật, nên chỉ có người nhận mới có thể giải mã được thông tin.

*Đặc điểm nổi bật* của các hệ mã hóa khóa bất đối xứng là kích thước khóa lớn, lên đến hàng ngàn bit. Do vậy, các hệ mã hóa dạng này thường có tốc độ thực thi chậm hơn nhiều lần so với các hệ mã hóa khóa đối xứng với độ an toàn tương đương. Mặc dù vậy, các hệ mã hóa khóa bất đối xứng có khả năng đạt độ an toàn cao và ưu điểm nổi bật nhất là việc quản lý và phân phối khóa đơn giản hơn do khóa công khai có thể phân phối rộng rãi.

Hệ thống mật mã hóa khóa công khai có thể sử dụng với các mục đích:

- Mã hóa: giữ bí mật thông tin và chỉ có người có khóa bí mật mới giải mã được.
- Tạo chữ ký số: cho phép kiểm tra văn bản có phải đã được tạo với một khóa bí mật nào đó hay không.
- Thỏa thuận khóa: cho phép thiết lập khóa dùng để trao đổi thông tin giữa hai bên.

Các giải thuật mã hóa khóa bất đối xứng điển hình bao gồm: RSA, Rabin, ElGamal, McEliece và Knapsack. Trong bài tiểu luận này, chúng tôi tìm hiểu và trình bày về giải thuật mã hóa RSA – một trong các giải thuật mã hóa khóa đối xứng được sử dụng rộng rãi nhất trên thực tế.

#### **1.2.2.2 Giải thuật mã hóa 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.

- **Thuật toán**

Để thực hiện mã hóa và giải mã, thuật toán RSA dùng phép lũy thừa modulo của lý thuyết số.

*Quá trình sinh khóa:*

- 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  $K_u$  là cặp  $(e, N)$ , khóa riêng  $K_r$  là cặp  $(d, N)$ .

*Quá trình mã hóa:*

- Thông điệp  $m$  được chuyển thành số:  $m < n$
- Bản mã  $c = m^e \pmod n$

*Quá trình giải mã:*

- Bản mã  $c$ ,  $c < n$
- Bản rõ  $m = c^d \pmod n$

*Lưu ý:* Khi chọn hai số nguyên tố  $p$  và  $q$  để sử dụng trong hệ mã hoá RSA, quan trọng nhất là chúng không nên quá gần nhau hoặc quá lệch nhau.

- **Đánh giá**

Giải thuật RSA là một trong những giải thuật mã hoá công khai (asymmetric cryptography) quan trọng và phổ biến nhất được sử dụng trong lĩnh vực bảo mật thông tin. Dưới đây là một số điểm đánh giá về giải thuật RSA:

- *An toàn và Bảo mật:* RSA cung cấp một cơ chế mạnh mẽ để bảo vệ thông tin truyền trực tuyến bằng cách sử dụng các khóa công khai và bí mật. Việc phân tích khóa bí mật từ khóa công khai được cho là một bài toán rất khó đối với các thuật toán hiện đại nhưng chưa có bằng chứng toàn diện chứng minh rằng nó không thể được thực hiện.
- *Khả năng chống lại các cuộc tấn công:* RSA có thể chống lại nhiều loại cuộc tấn công như tấn công brute force (tấn công dò mò), tấn công miền trung gian (man-in-the-middle), và tấn công factorization (phân tích nhân tử). Tuy nhiên, việc sử dụng các cặp khóa ngắn có thể dễ dàng bị tấn công bằng các thuật toán factorization mạnh.

- *Hiệu suất:* RSA có thể tính toán chậm đối với các khóa dài. Các phép toán như phép nhân số nguyên lớn và tính modulo có thể tốn nhiều thời gian, đặc biệt là với các số nguyên lớn có hàng trăm hoặc hàng nghìn bit.
- *Sử dụng rộng rãi:* RSA được sử dụng rộng rãi trong các ứng dụng bảo mật như trao đổi khóa, ký số, và mã hoá dữ liệu truyền trực tuyến. Nó được tích hợp vào nhiều giao thức mạng như HTTPS, SSH, và SSL/TLS.
- *Quản lý khóa:* RSA đòi hỏi quản lý khóa cẩn thận. Quản lý và bảo vệ khóa bí mật là một thách thức đối với việc triển khai và vận hành các hệ thống sử dụng RSA.
- *Chi phí tính toán:* RSA có thể tốn kém về chi phí tính toán so với các giải thuật mã hoá khác như AES (Advanced Encryption Standard) trong một số tình huống.

Tóm lại, RSA là một giải thuật mã hoá công khai mạnh mẽ và phổ biến, nhưng cũng cần cân nhắc về hiệu suất tính toán và quản lý khóa khi triển khai trong các ứng dụng thực tế.



## CHƯƠNG 2. NỘI DUNG THỰC HÀNH

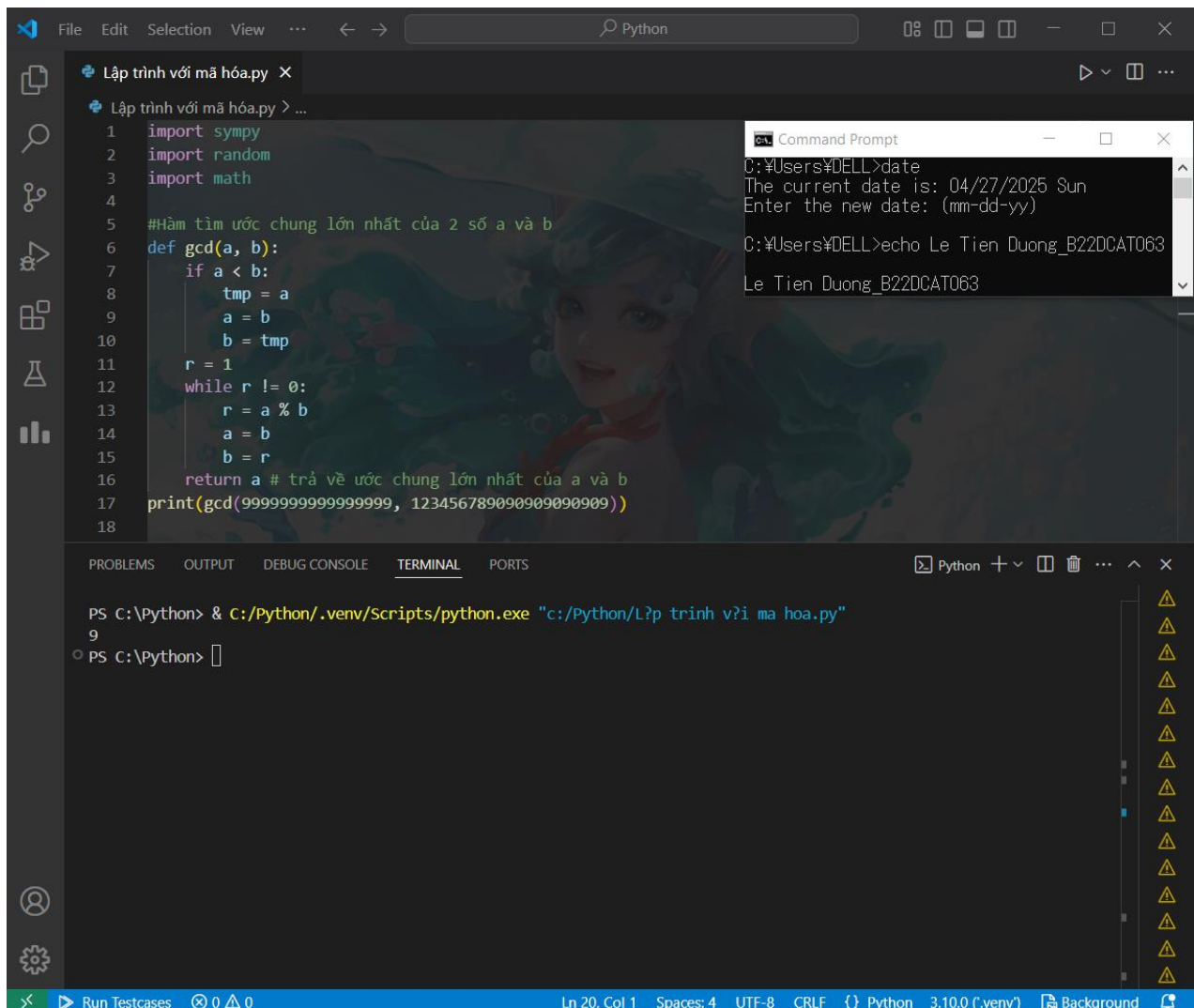
### 2.1 Chuẩn bị môi trường

- Trình biên dịch code
- Python 3.10.0

### 2.2 Các bước thực hiện

**2.2.1 Lập trình thư viện số lớn với các phép toán cơ bản, thử nghiệm chứng minh thư viện hoạt động tốt với các ví dụ phép toán cho số lớn**

#### 2.2.1.1 Hàm tính ước chung lớn nhất



The screenshot displays a Python IDE with a file named 'Lập trình với mã hóa.py'. The code defines a function `gcd(a, b)` that calculates the Greatest Common Divisor (GCD) of two numbers `a` and `b` using the Euclidean algorithm. The function includes imports for `sympy`, `random`, and `math`. It uses a `while` loop to iteratively calculate the remainder `r = a % b` until `r` becomes 0, at which point `a` is returned as the GCD. The function is tested with large numbers: `print(gcd(9999999999999999, 1234567890909090909))`.

A Command Prompt window is open, showing the execution of the script. It displays the current date and time, and the output of the `echo` command, which is 'Le Tien Duong\_B22DCAT063'.

The terminal window at the bottom shows the command to run the script: `PS C:\Python> & C:/Python/.venv/Scripts/python.exe "c:/Python/Lập trình với mã hóa.py"`. The output shows the result of the GCD calculation: `9`.

Hình 1 – Hàm tính ước chung lớn nhất

#### 2.2.1.2 Hàm tính $a^b \bmod m$

The screenshot shows a VS Code editor with a file named 'Lập trình với mã hóa.py'. The code defines a function `bipow(a, b, m)` that calculates  $a^b \mod m$  using a recursive-like approach with a while loop. The terminal shows the command `PS C:\Python> & C:/Python/.venv/Scripts/python.exe "c:/Python/Lập trình với mã hóa.py"` and the output `3`.

```

1 import sympy
2 import random
3 import math
4
5 #Hàm tính lũy thừa: a^b mod m
6 def bipow(a, b, m):
7     result = 1
8     a = a % m
9     while b > 0:
10         if b % 2 == 1:
11             result = (result * a) % m
12         a = (a * a) % m
13         b = b // 2
14     return result
15 print(bipow(9999999999999999, 9999999999999999999999999999, 4))
16
17

```

```

PS C:\Python> & C:/Python/.venv/Scripts/python.exe "c:/Python/Lập trình với mã hóa.py"
3
PS C:\Python>

```

Hình 2 – Hàm tính  $a^b \mod m$

### 2.2.1.3 Hàm tìm số nghịch đảo $d$ sao cho $d.e \mod m = 1$

The screenshot shows a VS Code editor with a file named 'Lập trình với mã hóa.py'. The code defines a function `multiplicative_inverse(e, phi)` that finds the modular inverse of  $e$  modulo  $\phi$  using the Extended Euclidean Algorithm. The terminal shows the command `PS C:\Python> & C:/Python/.venv/Scripts/python.exe "c:/Python/Lập trình với mã hóa.py"` and the output `Số d: 3617` and `Số dư d, e mod phi: 1`.

```

1 import sympy
2 import random
3 import math
4 def multiplicative_inverse(e, phi): # d là nghịch đảo của e trong phép modulo phi(n)
5     d = 0
6     x1 = 0
7     x2 = 1
8     y1 = 1
9     temp_phi = phi
10
11     while e > 0:
12         temp1 = temp_phi // e
13         temp2 = temp_phi - temp1 * e
14         temp_phi = e
15         e = temp2
16
17         x = x2 - temp1 * x1
18         y = d - temp1 * y1
19
20         x2 = x1
21         x1 = x
22         d = y1
23         y1 = y
24
25     if temp_phi == 1:
26         return d + phi
27 print(f'Số d: {multiplicative_inverse(3233, 3120)}')
28 mod = 3233 * multiplicative_inverse(3233, 3120) % 3120
29 print(f'Số dư d, e mod phi: {mod}')
30

```

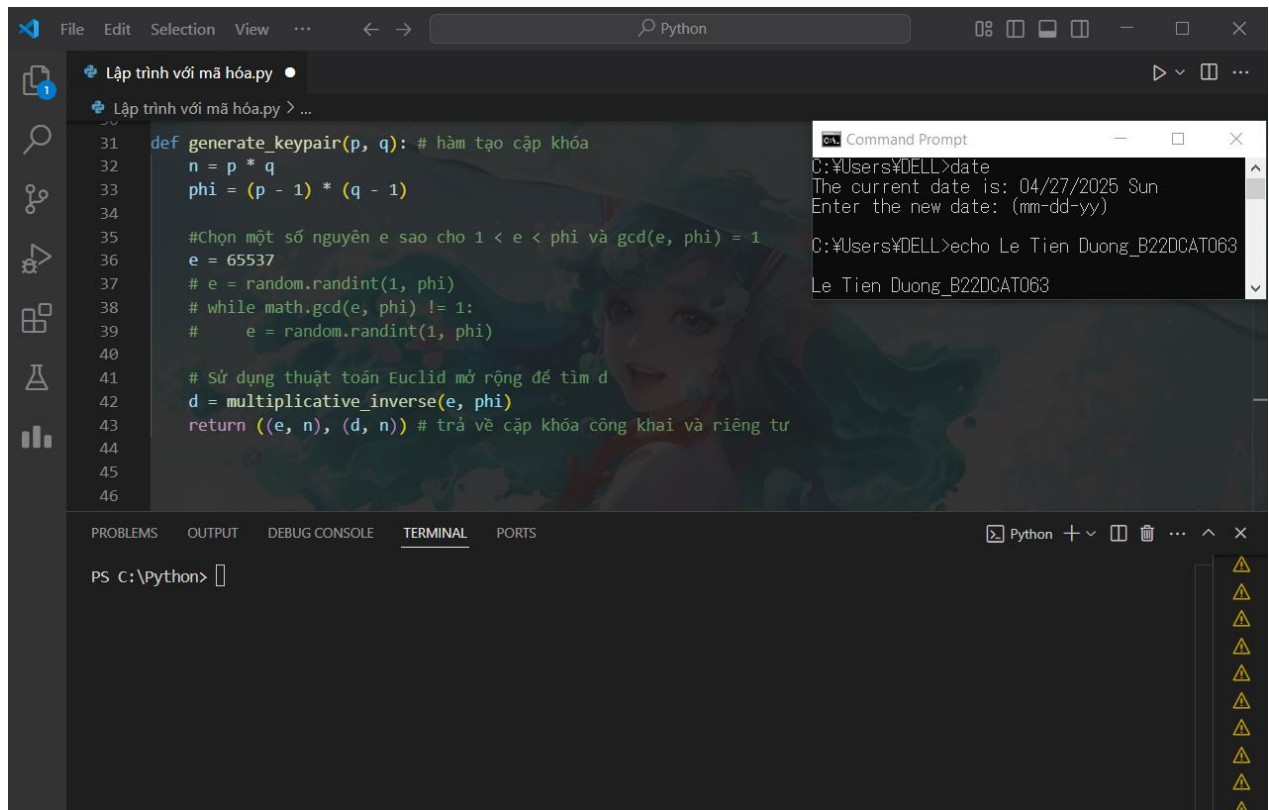
```

PS C:\Python> & C:/Python/.venv/Scripts/python.exe "c:/Python/Lập trình với mã hóa.py"
Số d: 3617
Số dư d, e mod phi: 1
PS C:\Python>

```

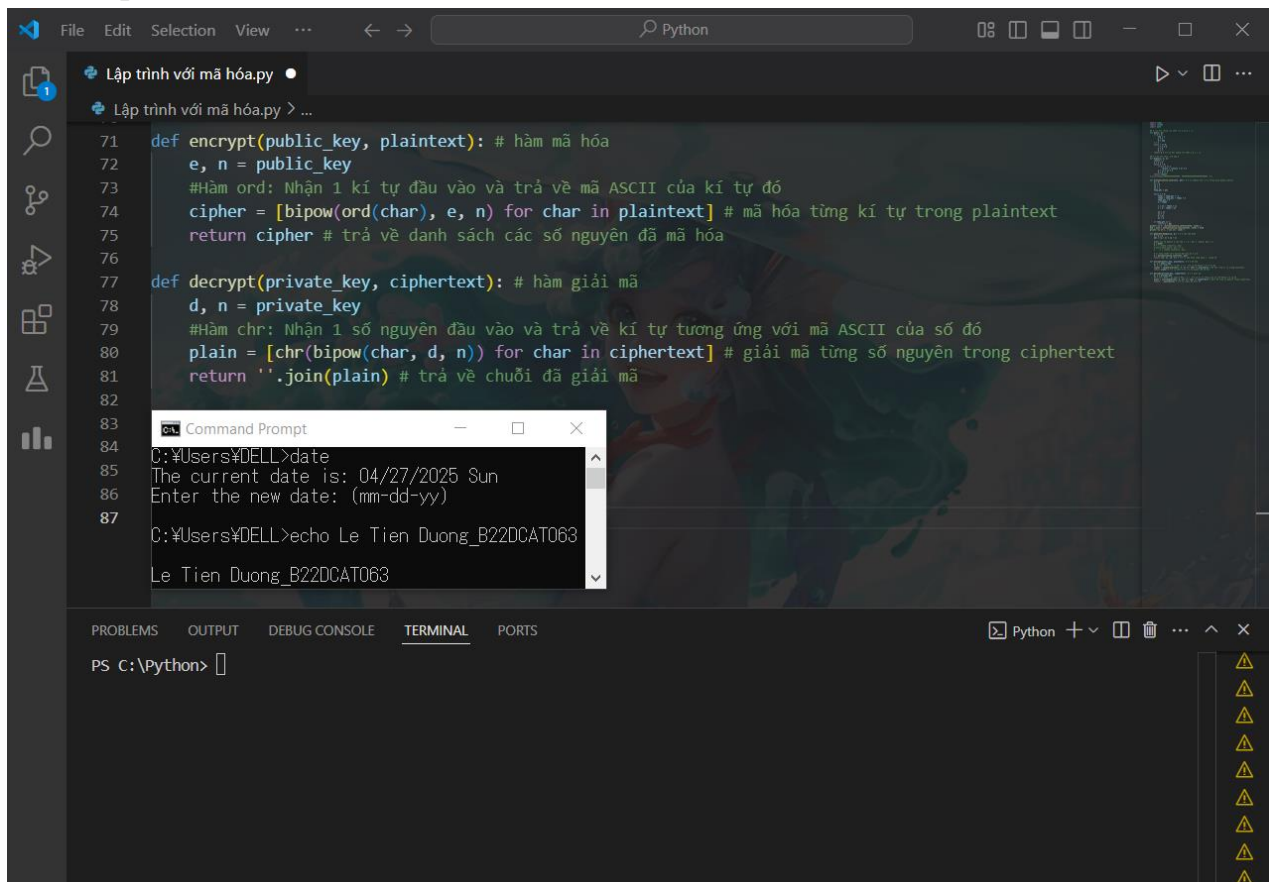
Hình 3 – Hàm tìm số nghịch đảo  $d$  sao cho  $d.e \mod m = 1$

### 2.2.1.4 Hàm sinh cặp khóa



Hình 4 – Hàm sinh khóa, để đơn giản ta chọn  $e = 65537$

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

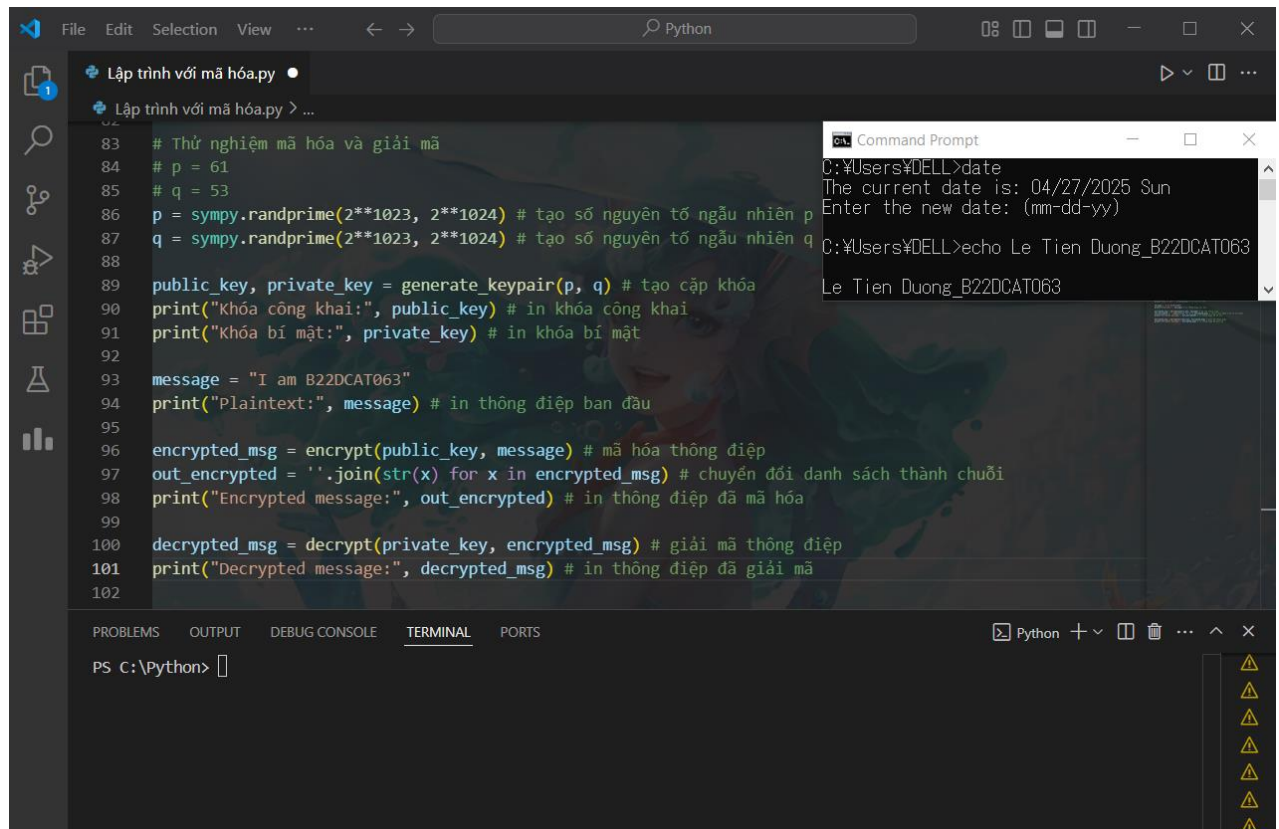


Hình 5 – Hàm mã hóa và giải mã



## 2.2.3 Thử nghiệm mã hóa và giải mã chuỗi ký tự: “I am B22DCAT063”

### 2.2.3.1 Thử nghiệm code với mã hóa và giải mã



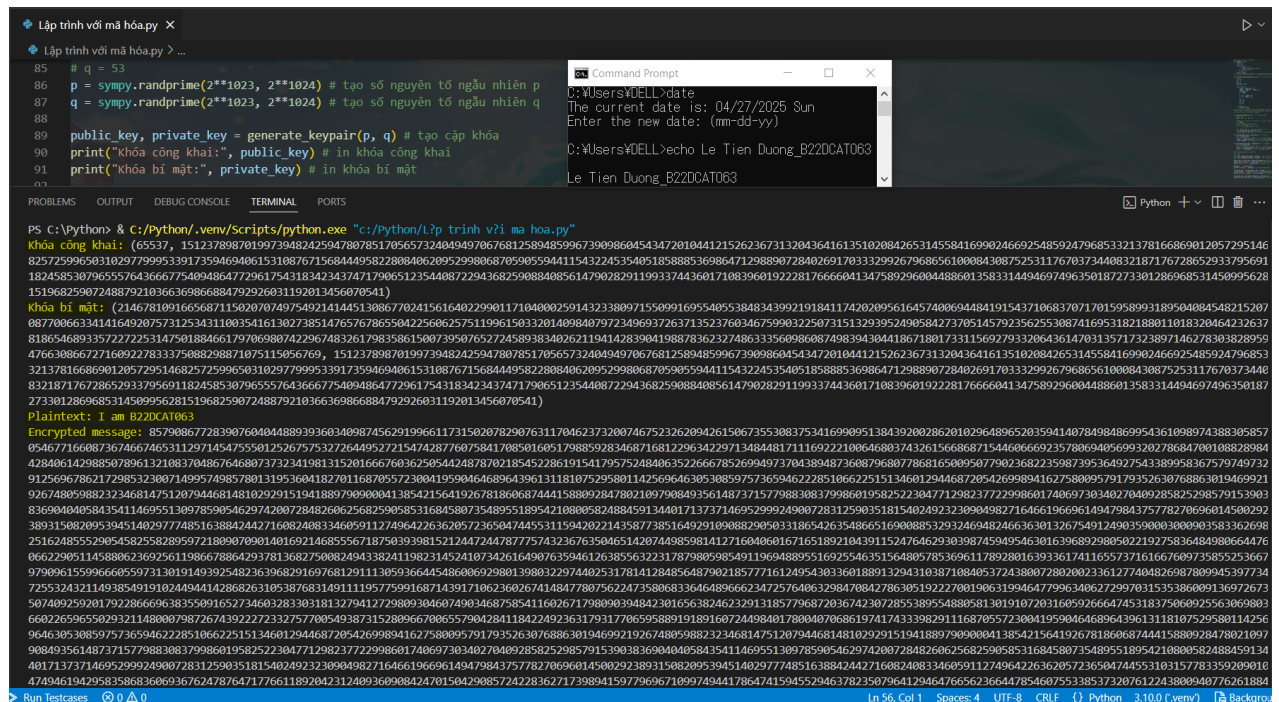
```
83 # Thử nghiệm mã hóa và giải mã
84 # p = 61
85 # q = 53
86 p = sympy.randprime(2**1023, 2**1024) # tạo số nguyên tố ngẫu nhiên p
87 q = sympy.randprime(2**1023, 2**1024) # tạo số nguyên tố ngẫu nhiên q
88
89 public_key, private_key = generate_keypair(p, q) # tạo cặp khóa
90 print("Khóa công khai:", public_key) # in khóa công khai
91 print("Khóa bí mật:", private_key) # in khóa bí mật
92
93 message = "I am B22DCAT063"
94 print("Plaintext:", message) # in thông điệp ban đầu
95
96 encrypted_msg = encrypt(public_key, message) # mã hóa thông điệp
97 out_encrypted = ''.join(str(x) for x in encrypted_msg) # chuyển đổi danh sách thành chuỗi
98 print("Encrypted message:", out_encrypted) # in thông điệp đã mã hóa
99
100 decrypted_msg = decrypt(private_key, encrypted_msg) # giải mã thông điệp
101 print("Decrypted message:", decrypted_msg) # in thông điệp đã giải mã
102
```

```
C:\Users\DELL>date
The current date is: 04/27/2025 Sun
Enter the new date: (mm-dd-yy)
C:\Users\DELL>echo Le Tien Duong_B22DCAT063
Le Tien Duong_B22DCAT063
```

Hình 6 – Thử nghiệm code với mã hóa và giải mã

### 2.2.3.2 Kết quả chạy chương trình

Do kết quả dài nên em xin phép chụp 2 ảnh.



```
PS C:\Python> & C:\Python\venv\Scripts\python.exe "c:/Python/L\p trình v\i ma hoa.py"
Khóa công khai: (65537, 1512378987819973948242594780785170565732404949706768125894859673909860453472010441215626373132043641613151020842653145584169902466925485924796853321378166869012057295146
8257259965931029779995339173594694061510876715684449582280840620952998068705905944115432245354051858885369864712988907284026917033329926796865610008430875253117670373440832187176728652933795691
1824585307965557643666775409486477296175431834234374717906512354408722943682590884085614790282911993374436017108396019222817666604134758929600448860135833144946974963501872733012869685314509595628
1519682590724887921036636986884792926031192013456070541)
Khóa bí mật: (7146781091665687115020707497549214145138867702415616402299017104000259143233809715599916955405358434399219184117420209561645740869448419154373710683701710859893189504084548215207
087700663341416492075731253411100354161302738514765767865504225606257511996150332014098407972349693726371352376034675990322507315132939524905842737051457923562553087416953182188011018320464232637
8186546893357227253147501884661797069807422967483261798358615007395076527245893834062191941283904198878362327486335608608749839430441867180173311569279332064361470317317323897146278303828959
47663086672160927833789882988781075115956769, 1512378987019973948242594780785170565732404949706768125894859673909860453472010441215626373132043641613151020842653145584169902466925485924796853
32137816686901205729514692572599533917359469406151087671568444958228084062095299806870590594411543224535405185888536986471298890728402691703332992679686561000843087525311767037344083
832187176728652933795601102458530796555764366677540948647729617543183423437471790651235440872294368259088408561479028291199337443601710839601922281766660413475892960044886013583314494697496350187
2733012869685314509956281519682590724887921036636986884792926031192013456070541)
Plaintext: I am B22DCAT063
Encrypted message: 857908677839076040488939363040987456291996611731502078290763117046237320074675232620942615067355308375341699095138439200286201029648965203594140784984869954361098974388305857
0546771660873674667465311291745475550125267573272640952721547428776075817085016051798859283468716812296342297134844817111692221006468037432615668687154460666923578069405699320278684700108828984
4284061429885078961321083704867646807373214981315201666760362505442487870218545228619514179575248846352266678526994973704389487360879680778681650095077902368225987395364927543389958367579740732
9125966786217298532300714995749857801319536041827011687057230041959046468964396131181075295801142569646305308595736594622285106622515134601294468720542699894162758009579179352630768863019469921
92674805988232346814751207944681481029291519418897990900413854215641926781860687444158809284780219790849356148737157798830837998601958252230477129823772299860174069730340270409285825298579159301
389304040584351146955130978590546297420072848260625682590585316845807354895518954210800582488459134401713717146952999249007283125903518154024923230904982716466196696149479843757827069601500292
38931588209539451402977748576388424427160824883346059112749642263620572365847445531159420221435877385164929109088290503318654263548665169008853293246948246636301326754912490359000300903583362698
2516248555290545825828959721809070901481692146855567187503939815122447244777574326763504651420744985981412716040601671651892104391152476462930398745949546301639689298050221927583648498066447
06622905114580623692511986678864293781368275008249433824119823145241073426164907635946126385563223178798059854911969488955169255463515648057853696117892801639336174116557371616670973585523667
9709061559966055973130191493925482369682916976812911130593664454860609298013980322974402531781412848564879021857771612495430336018891329431038710840537243800728020023361277404826987809945397734
725532432149385191024494414286826310538768314911119577599168714391710623602674148477807562247358068336464896662347257640632984708427863051922270019063199464779963406272997031535386009136972673
507409259201792786669638355091652734603283303181327941272980930460749034687585411602671798090394842301656382462329131857796872036742307285538955488058130191072031605926664745318375060925563069803
9602265965502932114800079872674322272332757700549387315280966700655790428411842249236317931770659588919189160724498401780040706861974174333982911168705723004195904646896439613118107529580114256
94630530859573659462228510662251513460129446872054269989416275800957917935263076886301946992192674480568148102929151941889799090041385421564192678186068744415880928478021097
9084935614873715779883083799860195825223047712982377229986017406973034027040928582529857915390383690405843541146955130978590546297420072848260625682590585316845807354895518954210800582488459134
401713737146952992490072831259035181540249232309049827164661966961494798437578270696016716518921043911524764629303987459495463016396892980502219275836484980664474946194259853586836067624787647176611892042312409360908424701504290857242283627173989415977696671099749441786474159455294637823507964129464766562366447854607553385373207612243800940776261884
```

Hình 7 – Kết quả chạy chương trình - 1



## **TÀI LIỆU THAM KHẢO**

- [1] Đỗ Xuân Chợt, Bài giảng Mật mã học cơ sở, Học viện Công Nghệ Bưu Chính Viễn Thông, 2021.
- [2] Đỗ Xuân Chợt, Bài giảng Mật mã học nâng cao, Học viện Công Nghệ Bưu Chính Viễn Thông, 2021.