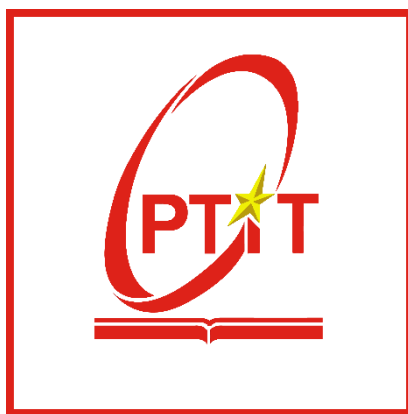


HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN
MÔN THỰC TẬP CƠ SỞ



BÁO CÁO THỰC HÀNH SỐ 16:
LẬP TRÌNH THUẬT TOÁN MẬT MÃ HỌC

Tên sinh viên: Nguyễn Bá Hải Long

Mã sinh viên: B21DCAT119

Giảng viên hướng dẫn: Thầy Đinh Trường Duy

HÀ NỘI 04/2024

Mục lục:

I. Mục đích:.....	1
II. Nội dung thực hành:	1
1. Cơ sở lý thuyết:	1
1.1. Tìm hiểu về lập trình số lớn với các phép toán cơ bản:.....	1
1.1.1. Mở đầu:	1
1.1.2. Xử lý số lớn với ngôn ngữ Python:.....	1
1.2. Tìm hiểu về giải thuật mật mã khóa công khai RSA:.....	2
1.2.1. Hệ mã hóa RSA:	2
1.2.2. Sinh khóa:.....	3
1.2.3. Mã hóa và giải mã:.....	3
2. Nội dung thực hành:.....	5
2.1. Chuẩn bị môi trường:.....	5
2.2. Các bước thực hiện và kết quả cần đạt:	6
III.Kết luận:	12

BÁO CÁO THỰC HÀNH SỐ 16

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. Nội dung thực hành:

1. Cơ sở lý thuyết:

1.1. Tìm hiểu về lập trình số lớn với các phép toán cơ bản:

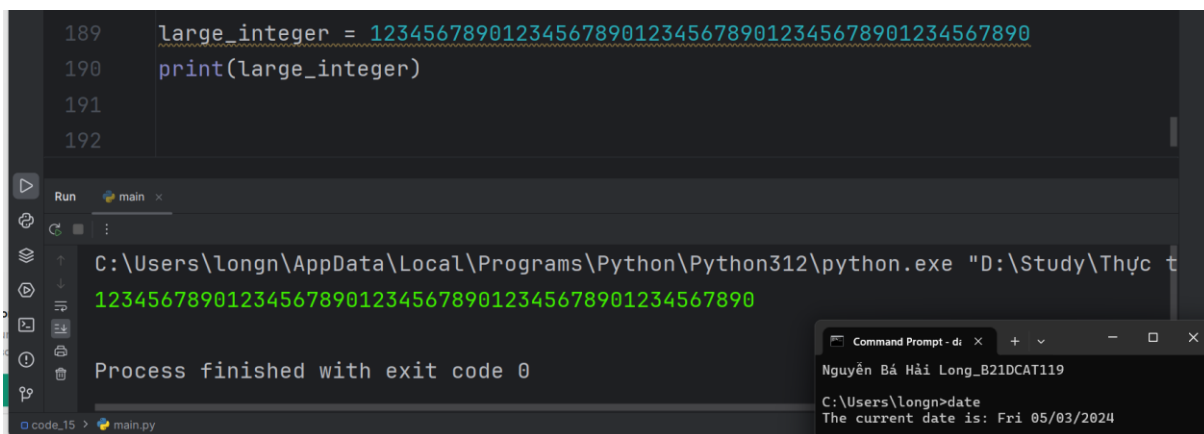
1.1.1. Mở đầu:

- Chúng ta đều biết rằng, việc giải bài toán bằng máy tính nói chung và lập trình thi đấu nói riêng luôn luôn đối mặt với dữ liệu có kích thước rất lớn. Hiển nhiên là vì những dữ liệu quá lớn vượt ra ngoài khả năng tính toán của con người, nên mới cần tới sự trợ giúp của máy tính.
- Với sự nâng cấp liên tục của máy tính điện tử, độ lớn dữ liệu mà máy tính có thể lưu trữ được cũng ngày càng tăng lên. Tuy nhiên, khả năng lưu trữ luôn luôn là hữu hạn, mà dữ liệu là vô hạn (dữ liệu dạng văn bản có thể dài vô hạn, dữ liệu dạng số có thể cực kỳ lớn,...). Ngôn ngữ lập trình đã có sẵn rất nhiều kiểu dữ liệu với khoảng giá trị rất lớn, nhưng cũng không phải luôn luôn lưu trữ được mọi giá trị.
- Riêng đối với kiểu số trong C++, chúng ta có sẵn kiểu dữ liệu nguyên thủy là long long với tầm lưu trữ lên tới khoảng 20 chữ số. Nhưng nếu như có những dữ liệu số với độ dài nhiều hơn thế thì sao? Kỹ thuật xử lý số nguyên lớn ra đời nhằm giải quyết vấn đề đó.

1.1.2. Xử lý số lớn với ngôn ngữ Python:

- Python có khả năng xử lý số lớn mà không gặp phải vấn đề về giới hạn kích thước. Điều này là nhờ vào việc sử dụng cấu trúc dữ liệu động để lưu trữ số nguyên, cho phép biểu diễn và thực hiện các phép toán trên các số có số lượng chữ số lớn.
- Trong Python, bạn có thể sử dụng kiểu dữ liệu int để biểu diễn số nguyên lớn mà không cần lo lắng về giới hạn kích thước. Dưới đây là một ví dụ đơn giản:

BÁO CÁO THỰC HÀNH SỐ 16



```
189 large_integer = 12345678901234567890123456789012345678901234567890
190 print(large_integer)
191
192
```

Run main x

C:\Users\longn\AppData\Local\Programs\Python\Python312\python.exe "D:\Study\Thực t

123456789012345678901234567890123456789012345678901234567890

Process finished with exit code 0

code_15 main.py

Command Prompt - d: x + - □ x

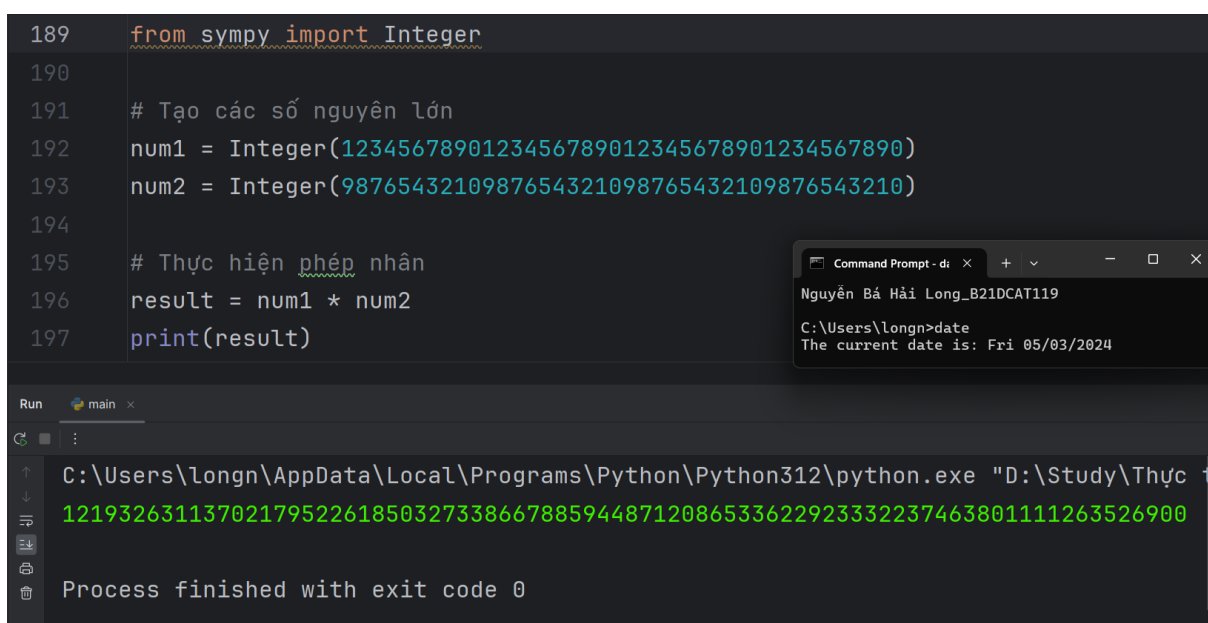
Nguyễn Bá Hải Long_B21DCAT119

C:\Users\longn>date

The current date is: Fri 05/03/2024

Hình 1: Ví dụ số nguyên lớn trong python

- Python cũng cung cấp các thư viện như decimal, fractions, math, gmpy2 hoặc sympy để thực hiện các phép toán phức tạp hoặc chính xác với số nguyên lớn. Ví dụ, thư viện sympy cung cấp các hàm và phương thức để xử lý các số nguyên lớn với hiệu suất tốt hơn, đặc biệt là khi cần độ chính xác cao.



```
189 from sympy import Integer
190
191 # Tạo các số nguyên lớn
192 num1 = Integer(1234567890123456789012345678901234567890)
193 num2 = Integer(9876543210987654321098765432109876543210)
194
195 # Thực hiện phép nhân
196 result = num1 * num2
197 print(result)
```

Run main x

C:\Users\longn\AppData\Local\Programs\Python\Python312\python.exe "D:\Study\Thực t

1219326311370217952261850327338667885944871208653362292332237463801111263526900

Process finished with exit code 0

code_15 main.py

Command Prompt - d: x + - □ x

Nguyễn Bá Hải Long_B21DCAT119

C:\Users\longn>date

The current date is: Fri 05/03/2024

Hình 2: Ví dụ xử lý số nguyên lớn sử dụng thư viện sympy

1.2. Tìm hiểu về giải thuật mật mã khóa công khai RSA:

1.2.1. Hệ mã hóa RSA:

- RSA là một hệ mã hóa bất đối xứng được phát triển bởi Ron Rivest, Adi Shamir và Leonard Adleman (tên của nó cũng chính là tên viết tắt của 3 tác giả này) và được sử dụng rộng rãi trong công tác mã hoá và công nghệ chữ ký điện tử. Trong hệ mã hóa này, public key có thể chia sẻ công khai

BÁO CÁO THỰC HÀNH SỐ 16

cho tất cả mọi người. Hoạt động của RSA dựa trên 4 bước chính: sinh khóa, chia sẻ key, mã hóa và giải mã.

1.2.2. Sinh khóa:

- Mấu chốt cơ bản của việc sinh khóa trong RSA là tìm được bộ 3 số tự nhiên e , d và n sao cho:

$$m^{ed} \equiv m \pmod{n}$$

Hình 3: Công thức sinh khóa

- Và một điểm không thể bỏ qua là cần bảo mật cho d sao cho dù biết e , n hay thậm chí cả m cũng không thể tìm ra d được.
- Cụ thể khóa RSA được sinh như sau:
 - o Chọn 2 số nguyên tố p và q
 - o Tính $n = p \cdot q$. Sau này, n sẽ được dùng làm modulus trong cả public key và private key.
 - o Tính một số giả nguyên tố bằng phi hàm Carmichael như sau:
 $\lambda(n) = \text{BCNN}(\lambda(p), \lambda(q)) = \text{BCNN}(p-1, q-1)$. Giá trị này sẽ được giữ bí mật.
 - o Chọn một số tự nhiên e trong khoảng $(1, \lambda(n))$ sao cho $\text{UCLN}(e, \lambda(n)) = 1$, tức là e và $\lambda(n)$ nguyên tố cùng nhau.
 - o Tính toán số d sao cho $d \equiv 1/e \pmod{\lambda(n)}$ hay viết dễ hiểu hơn thì $d \cdot e \equiv 1 \pmod{\lambda(n)}$. Số d được gọi là số nghịch đảo modulo của e (theo modulo mod $\lambda(n)$).
- Public key sẽ là bộ số (n, e) , và private key sẽ là bộ số (n, d) . Chúng ta cần giữ private key thật cẩn thận cũng như các số nguyên tố p và q vì từ đó có thể tính toán các khóa rất dễ dàng.
- Trong thực hành, chúng ta thường chọn e tương đối nhỏ để việc mã hóa và giải mã nhanh chóng hơn. Giá trị thường được sử dụng là $e = 65537$. Ngoài ra, chúng ta có thể tính số giả nguyên tố bằng phi hàm Euler $\varphi(n) = (p-1)(q-1)$ và dùng nó như $\lambda(n)$. Vì $\varphi(n)$ là bội của $\lambda(n)$ nên các số d thỏa mãn $de \equiv 1 \pmod{\varphi(n)}$ cũng sẽ thỏa mãn $d \equiv 1/e \pmod{\lambda(n)}$. Tuy nhiên, một số tác dụng phụ của việc này là d thường sẽ trở nên lớn hơn mức cần thiết.

1.2.3. Mã hóa và giải mã:

BÁO CÁO THỰC HÀNH SỐ 16

- Trong phần này, chúng ta sẽ tìm hiểu cách mã hóa với public key (n, e) và giải mã với private key (n, d) .
- Nếu chúng ta có bản rõ M , chúng ta cần chuyển nó thành một số tự nhiên m trong khoảng $(0, n)$ sao cho m, n nguyên tố cùng nhau. Việc này rất dễ dàng thực hiện bằng cách thêm một các kỹ thuật padding. Tiếp theo, chúng ta sẽ mã hóa m , thành c như sau:

$$c \equiv m^e \pmod n$$

Hình 4: Công thức mã hóa

- Sau đó giá trị c sẽ được chuyển cho người nhận.
- Ở phía người nhận, họ sẽ giải mã từ c để lấy được m như sau:

$$c^d \equiv m^{de} \equiv m \pmod n$$

Hình 5: Công thức giải mã

- Từ m có thể lấy lại được bản tin bằng cách đảo ngược padding.
- Chúng ta lấy một ví dụ đơn giản như sau:
 $p = 5, q = 7$
 $\Rightarrow n = p \cdot q = 35$
 $\Rightarrow \phi(n) = 24$
- Chúng ta chọn $e = 5$ vì $\text{UCLN}(5, 24) = 1$, cuối cùng chọn $d = 29$ vì $e \cdot d - 1 = 29 \cdot 5 - 1$ chia hết cho 24.
- Giả sử $m = 32$ (dấu cách), chúng ta sẽ mã hóa m và thu được:
 $c = 32^5 \% 35 = 2$
- Giải mã c để thu được m
 $m = 2^{29} \% 35 = 32$
- Đây chính là m ban đầu. Bạn có thể thử m với các giá trị khác nhau để thấy thuật toán hoàn toàn chính xác. Tất nhiên rồi, nó đã được chứng minh mà. Việc chứng minh RSA hoạt động chính xác là một vấn đề toán học khá phức tạp mà bài viết này sẽ không đi vào chi tiết của nó.

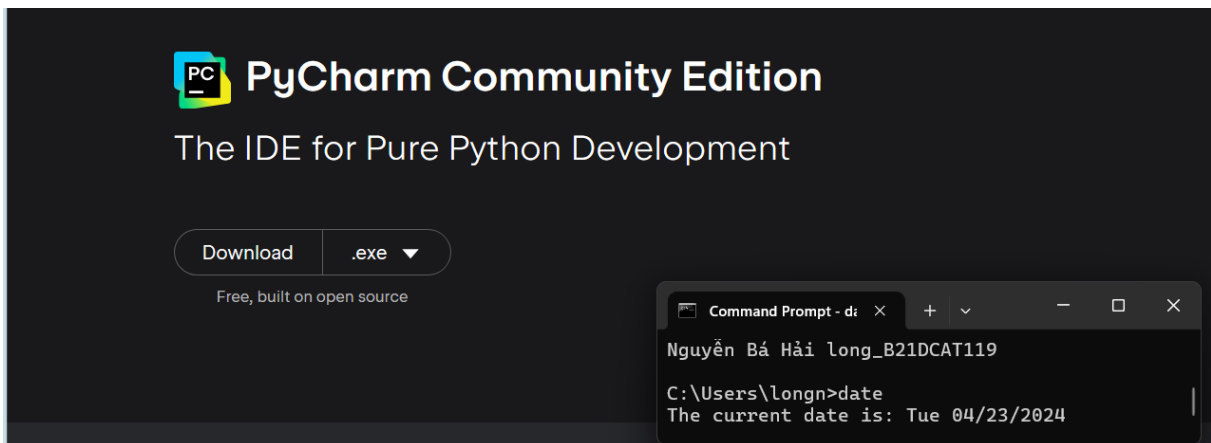
BÁO CÁO THỰC HÀNH SỐ 16

- Mức độ bảo mật của RSA phụ thuộc rất lớn vào khả năng phân tích thừa số nguyên tố của các số lớn. Bởi vì chúng ta cung cấp public một cách rộng rãi, nếu việc phân tích thừa số nguyên tố đơn giản, thì việc bị lộ private là không thể tránh khỏi.
- Vì vậy, khi sinh khóa, chúng ta cần chọn các số nguyên tố p và q một cách ngẫu nhiên. Bản thân hai số nguyên tố này cũng rất lớn, và để việc phân tích thừa số nguyên tố khó khăn hơn, hai số nguyên tố này sẽ không có cùng độ dài. Trong tương lai gần, có lẽ vẫn chưa có một phương pháp hiệu quả nào cho phép thực hiện điều này với các máy tính cá nhân.
- Tuy nhiên, với sự phát triển của công nghệ, các siêu máy tính xuất hiện ngày càng nhiều. Cùng với chúng ta máy tính lượng tử cho phép tính toán với tốc độ cao hơn rất nhiều có thể sẽ phá vỡ sự bảo mật của RSA.
- Ngày từ năm 1993, thuật toán Shor đã được phát triển và chỉ ra rằng máy tính lượng tử có thể giải bài toán phân tích ra thừa số trong thời gian đa thức. Rất may là những điều này mới chỉ là lý thuyết vì đến thời điểm hiện tại và trong vài năm tới, máy tính lượng tử vẫn chưa hoàn thiện.

2. Nội dung thực hành:

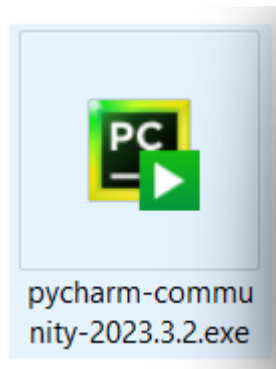
2.1. Chuẩn bị môi trường:

- Môi trường lập trình Python:
 - o Truy cập vào trang web:
<https://www.jetbrains.com/pycharm/download/?section=windows>

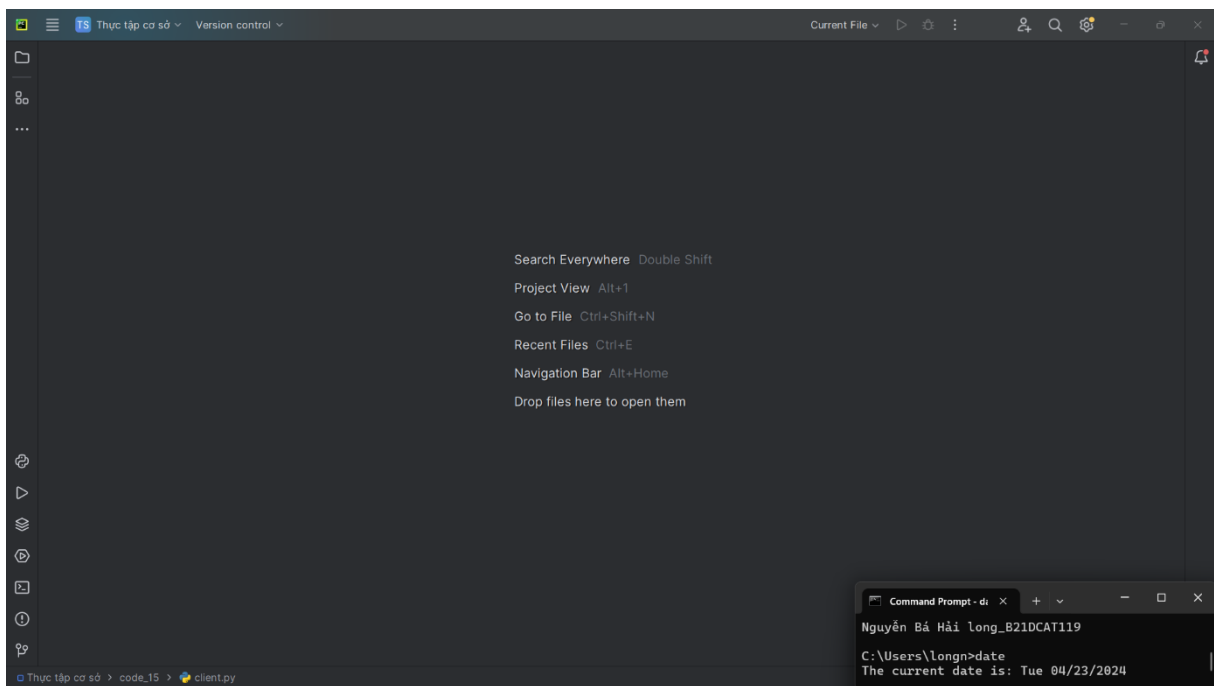


Hình 6: Chọn Download để tải PyCharm bản Community

BÁO CÁO THỰC HÀNH SỐ 16



Hình 7: File cài đặt PyCharm định dạng .exe



Hình 8: Giao diện cài đặt thành công

2.2. Các bước thực hiện và kết quả cần đạt:

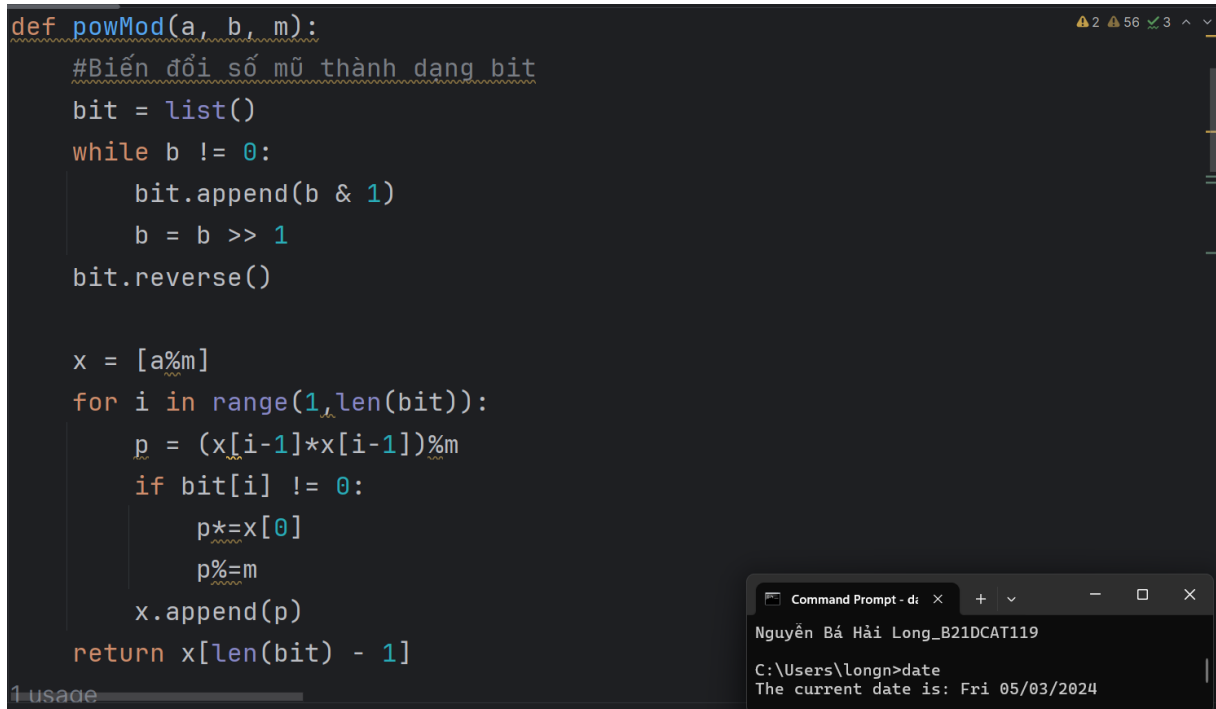
- Lập trình thư viện số lớn với các phép toán cơ bản để sử dụng trong giải thuật mã hóa/giải mã RSA:
 - Hàm powMod(a,b,m):
 - Mục đích: Tính $(a^b) \% m$ với a, b, m đều là các số nguyên vô cùng lớn.
 - Phương thức áp dụng: fast modular exponentiation

BÁO CÁO THỰC HÀNH SỐ 16

```
def powMod(a, b, m):
    #Biến đổi số mũ thành dạng bit
    bit = list()
    while b != 0:
        bit.append(b & 1)
        b = b >> 1
    bit.reverse()

    x = [a%m]
    for i in range(1, len(bit)):
        p = (x[i-1]*x[i-1])%m
        if bit[i] != 0:
            p*=x[0]
            p%=m
        x.append(p)
    return x[len(bit) - 1]
```

1 usage

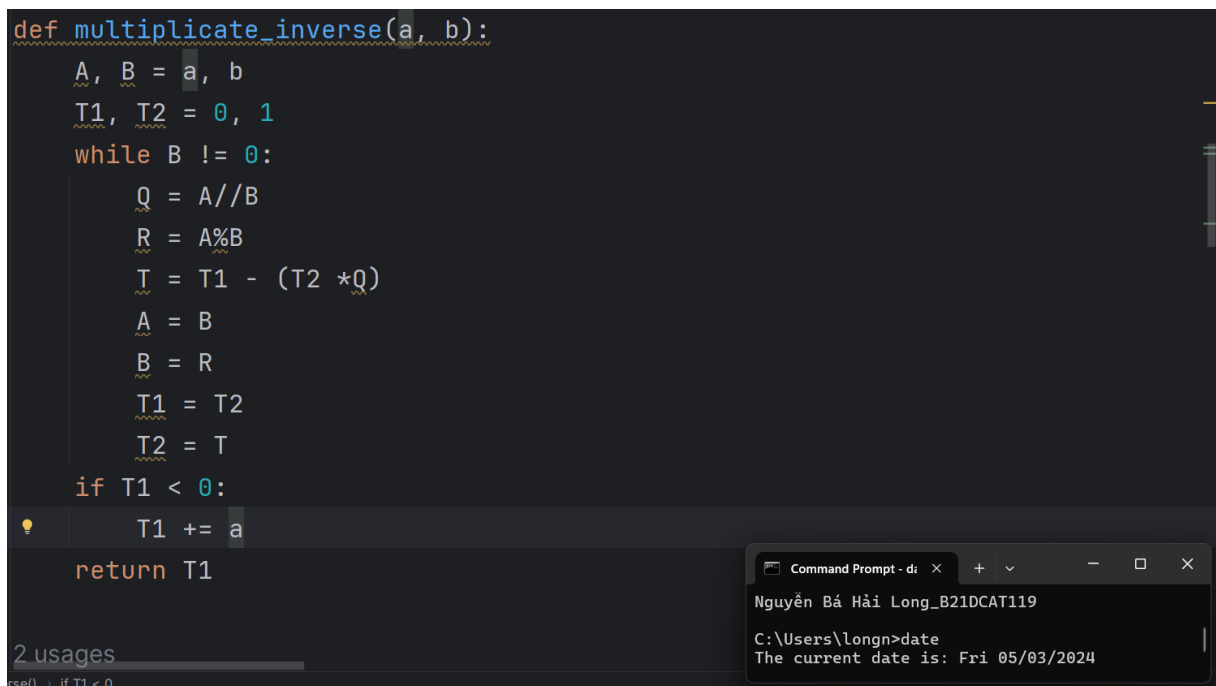


Hình 9: Code hàm powMod

- Hàm multiply_inverse:
 - Mục đích: Tìm ra giá trị d thỏa mãn điều kiện $(d * e) \bmod \phi(n) == 1$ (Công thức tìm d trong RSA)
 - Phương pháp áp dụng: multiply_inverse
 - Đầu vào: a, b là các số vô cùng lớn tương đương với $\phi(n)$, e

```
def multiply_inverse(a, b):
    A, B = a, b
    T1, T2 = 0, 1
    while B != 0:
        Q = A//B
        R = A%B
        T = T1 - (T2 * Q)
        A = B
        B = R
        T1 = T2
        T2 = T
    if T1 < 0:
        T1 += a
    return T1
```

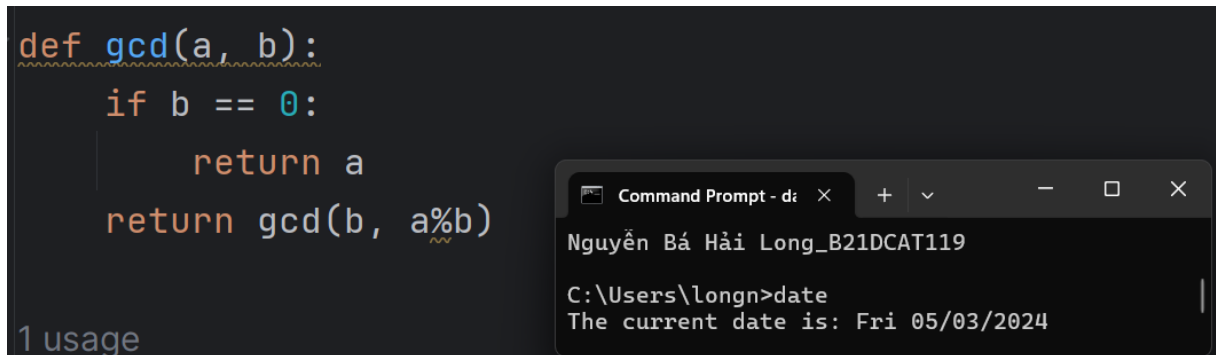
2 usages



Hình 10: Code hàm multiply_inverse

BÁO CÁO THỰC HÀNH SỐ 16

- Hàm gcd:
 - Mục đích: Tính ước chung lớn nhất với hai số lớn



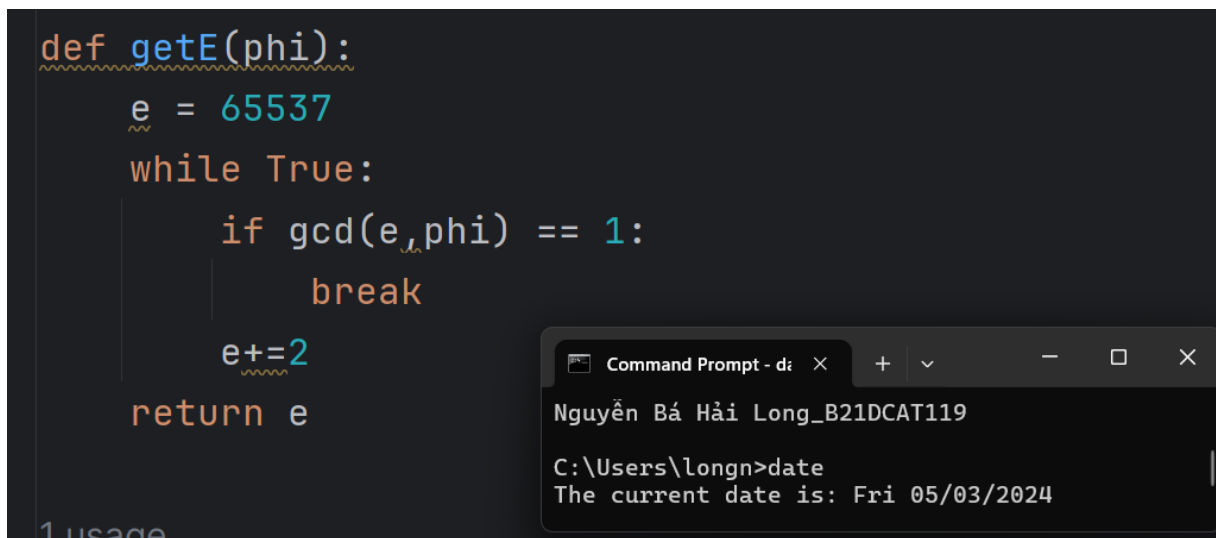
```
def gcd(a, b):  
    if b == 0:  
        return a  
    return gcd(b, a%b)
```

1 usage

Command Prompt - d: x + - □ x
Nguyễn Bá Hải Long_B21DCAT119
C:\Users\longn>date
The current date is: Fri 05/03/2024

Hình 11: Code hàm gcd

- Hàm getE:
 - Mục đích: Tìm giá trị e thỏa mãn điều kiện ước chung giữa e và phi(n) bằng -1 hay $\text{gcd}(e, \phi) == -1$.
 - Phương pháp: Ta xác định giá trị $e = 65537$. Bởi e là một số nguyên, nếu $\text{gcd}(e, \phi)$ khác -1 thì sẽ cộng thêm 2 đơn vị để giữ giá trị nguyên. Sau đó tiếp tục tìm kiếm e thỏa mã điều kiện.



```
def getE(phi):  
    e = 65537  
    while True:  
        if gcd(e, phi) == 1:  
            break  
        e+=2  
    return e
```

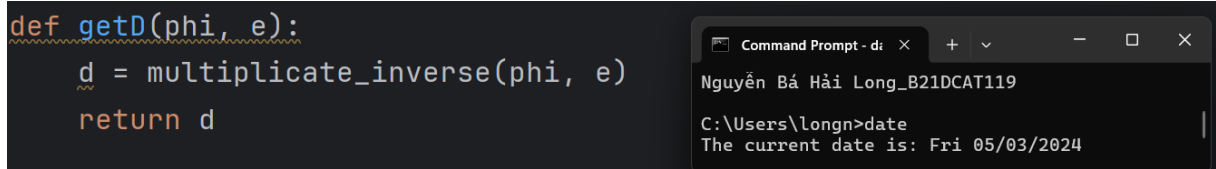
1 usage

Command Prompt - d: x + - □ x
Nguyễn Bá Hải Long_B21DCAT119
C:\Users\longn>date
The current date is: Fri 05/03/2024

Hình 12: Code hàm getE

- Hàm getD:
 - Mục đích: Tìm giá trị d trong giải thuật RSA.
 - Phương pháp: Sử dụng hàm `multiplicate_inverse(phi,e)` đã được nêu ở trên.

BÁO CÁO THỰC HÀNH SỐ 16



Hình 13: Code hàm getD

- Hàm encrypt:
 - Mục đích: Mã hóa dữ liệu đầu vào bản rõ (plaintext) bằng giải thuật RSA
 $C = (P^e) \bmod n$
 - Phương pháp: Ta thực hiện biến đổi mỗi kí tự thuộc bản rõ thành một giá trị số. Trong đó giá trị số là vị trí của kí tự đó nằm trong chuỗi “dig”. Sau đó áp dụng công thức mã hóa RSA. Kết quả trả về là danh sách các kí tự được mã hóa và được lưu vào biến result.



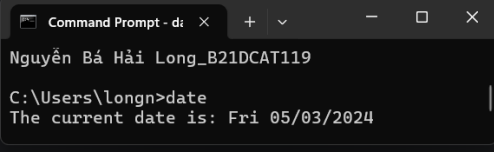
Hình 14: Code hàm encrypt

- Hàm decrypt:
 - Mục đích: Từ chuỗi mã hóa, ta thực hiện giải mã bằng công thức giải mã của giải thuật RSA.
 $P = (C^d) \bmod n$
 - Phương pháp: Đầu vào là danh sách các kí tự đã được mã hóa. Ta áp dụng công thức giải mã RSA để giải mã từng kí tự và rồi lấy kết quả thu được đem so với chuỗi “dig” để tìm ra

BÁO CÁO THỰC HÀNH SỐ 16

kí tự bản rõ. Kết quả sau cùng được đóng thành chuỗi và lưu vào biến result.

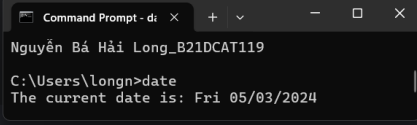
```
dig = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz-+ "
1 usage
def decrypt(d,n, ciphertext):
    global dig
    result = list()
    for i in ciphertext:
        result.append(dig[powMod(int(i),d,n)])
    return "".join(result)
```



Hình 15: Code hàm decrypt

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

```
p = 17680666608319994381062470848063706344101913730735134653044290258609510102780745030
q = 14141146446378305465026898218018594583696657361135789482125563884846210625919258352
n = p*q
phi = (p-1)*(q-1)
e = getE(phi)
d = getD(phi, e)
message = "Hello World"
encode = encrypt(e,n,message)
decode = decrypt(d,n,encode)
```



Hình 16: Code tính các giá trị phục vụ cho giải thuật RSA (có p, q là các số nguyên tố vô cùng lớn)

BÁO CÁO THỰC HÀNH SỐ 16

```
print("p: ", p)
print("q: ", q)
print("n: ", n)
print("d: ", d)
print("e: ", e)
print("phi: ", phi)
print("Message: ", message)
print("Encode: ", encode)
print("Decode: ", decode)
```

Hình 17: Code in giá trị từ các biến vừa tính được

```
C:\Users\longn\AppData\Local\Programs\Python\Python312\python.exe "D:\Study\Thực tập cơ sở\c
p: 1768066660831999438106247084806370634410191373073513465304429025860951010278074503084894
q: 1414114644637830546502689821801859458369665736113578948212556388484621062591925835254246
n: 2500248957778438553757475761053445886353918097743546135324497168749455754453994435903306
d: 2740709601092558794298442996690107152839853458991048634537923258662448714466255096743718
e: 65537
phi: 25002489577784385537574757610534458863539180977435461353244971687494557544539944359033
Message: Hello World
Encode: [1916871348879577165573977934908689784811813141848910614668391055811836552061001466
Decode: Hello World

Process finished with exit code 0
```

Hình 18: Kết quả có được với thử nghiệm “Hello World”

- Thử nghiệm mã hóa và giải mã chuỗi ký tự: “I am B21DCAT119”:

```
p = 17680666608319994381062470848063706344101913730735134653044290258609510102780745030
q = 14141146446378305465026898218018594583696657361135789482125563884846210625919258352
n = p*q
phi = (p-1)*(q-1)
e = getE(phi)
d = getD(phi, e)
message = "I am B21DCAT119"
encode = encrypt(e,n,message)
decode = decrypt(d,n,encode)
```

Hình 19: Thử nghiệm mã hóa và giải mã với chuỗi “I am B21DCAT119”

BÁO CÁO THỰC HÀNH SỐ 16

```
C:\Users\longn\AppData\Local\Programs\Python\Python312\python.exe "D:\Study\Thực tập cơ sở\c
p: 1768066660831999438106247084806370634410191373073513465304429025860951010278074503084894
q: 1414114644637830546502689821801859458369665736113578948212556388484621062591925835254240
n: 2500248957778438553757475761053445886353918097743546135324497168749455754453994435903300
d: 2740709601092558794298442996690107152839853458991048634537923258662448714466255096743718
e: 65537
phi: 25002489577784385537574757610534458863539180977435461353244971687494557544539944359033
Message: I am B21DCAT119
Encode: [2204723933832471549032937186037675590413349458822637159320187914694015404812728694
Decode: I am B21DCAT119

Process finished with exit code 0
```

Hình 20: Kết quả đạt được thành công mã hóa và giải mã chuỗi “I am B21DCAT119”

III. Kết luận:

- Hiểu về xử lý số nguyên lớn trong việc lập trình.
- Hiểu và lập trình giải thuật mã hóa/giải mã RSA kết hợp xử lý số lớn.