ĐẠI HỌC QUỐC GIA TP. HCM

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

--------------------



**MÔN MẬT MÃ VÀ AN NINH MẠNG**

**BÁO CÁO BÀI TẬP LỚN**

GVHD: NGUYỄN HỮU HIẾU

SVTH: Dương Quang Tuấn

MSSV: 1920069

Hồ Chí Minh, tháng 06 năm 2021

# DANH SÁCH THÀNH VIÊN

|  |  |
| --- | --- |
| **Tên** | **MSSV** |
| Dương Quang Tuấn | 1920069 |

# MỤC LỤC

[**DANH SÁCH THÀNH VIÊN**](#_30j0zll) **2**

[**MỤC LỤC**](#_1fob9te) **3**

[**CHƯƠNG 1: GIỚI THIỆU**](#_3znysh7) **4**

[Giới thiệu về RSA](#_2et92p0) 4

[RSA là gì?](#_tyjcwt) 4

[Lịch sử của RSA như thế nào?](#_3dy6vkm) 4

[Mô tả quá trình hoạt động của RSA](#_1t3h5sf) 5

[Sinh khóa](#_4d34og8) 5

[Mã hóa và giải mã](#_2s8eyo1) 6

[**CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**](#_17dp8vu) **9**

[Các module của hệ thống](#_3rdcrjn) 9

[Thuật toán Euclid mở rộng](#_942rvkyehnm0) 9

[Thuật toán tra tính nguyên tố.](#_nab5mo2i4gg4) 11

[Thuật toán Fermat](#_hbq4nnw1jqvq) 12

[Thuật toán Miller-Rabin](#_aasr2mg6e6lc) 13

[**CHƯƠNG 3: HIỆN THỰC VÀ ĐÁNH GIÁ HỆ THỐNG**](#_26in1rg) **15**

[Thuật toán Euclid mở rộng.](#_x1717ut7bu2z) 15

[Thuật toán Miller-Rabin kiểm tra tính nguyên tố.](#_oye7k38vegu4) 16

[Thuật toán Luỹ thừa mô-đun nhanh](#_2ilmmukan602) 18

[Thời gian thực thi](#_cqnn1go9kero) 18

[**CHƯƠNG 4: KẾT LUẬN**](#_lnxbz9) **19**

[Ưu điểm](#_dsnss8fx0mgy) 19

[Nhược điểm](#_fc4byq9cembr) 19

[Hướng phát triển](#_fl4o1srx5kg1) 19

[**TÀI LIỆU THAM KHẢO**](#_35nkun2) **20**

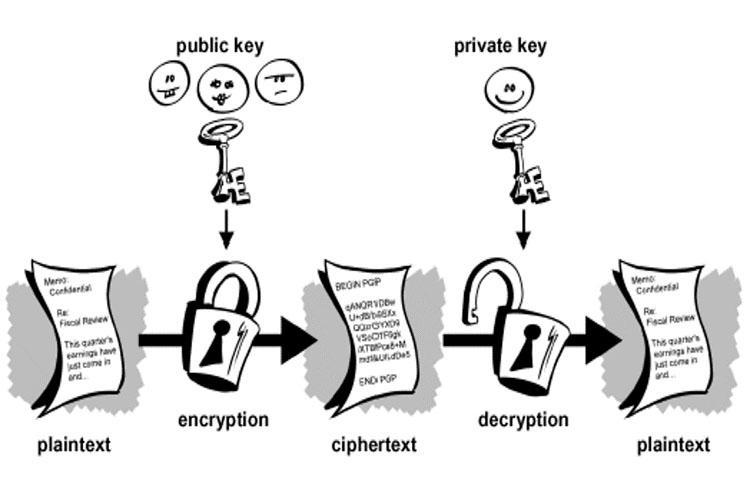
[**PHỤ LỤC**](#_1ksv4uv) **21**

# CHƯƠNG 1: GIỚI THIỆU

## **Giới thiệu về RSA**

### RSA là gì?

Trong mật mã học, RSA là một thuật toán mã học hóa khóa công khai. Đây là thuật toán thuật toán đầu tiên phù hợp với việc tạo ra chữ ký điện tử đồng thời với việc mã hóa nó. RSA đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công cộng. RSA đang được sử dụng phổ biến trong lĩnh vực thương mại điện tử và được cho là đảm bảo về an toàn với điều kiện độ dài khóa đủ lớn.



*RSA là gì ?*

### Lịch sử của RSA như thế nào?

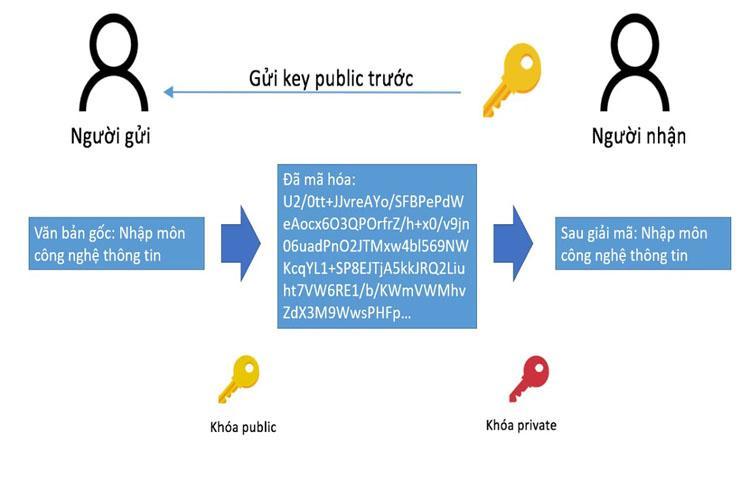
RSA là thuật toán được Ron Rivest, Adi Shamir và Len Adleman mô tả lần đầu vào năm 1977 tại học viện công nghệ Massachusetts (MIT). RSA chính là lấy từ 3 chữ cái đầu của 3 tên tác giả.

Trước đó, năm 1973 Clifford Cocks, một nhà toán học người Anh làm việc tại GCHQ đã có một bản mô tả thuật toán tương tự. Với khả năng tính toán tại thời điểm này thì thuật toán này không khả thi và nó chưa được thực nghiệm. Nhưng, phát minh này lại được công bố vào năm 1997 vì nó được xếp vào loại tuyệt mật.

Thuật toán RSA được MIT đăng ký bằng sáng chế tại Hoa Kỳ năm 1983, số đăng ký là 4,405, 829). Bằng sáng chế RSA này hết hạn vào ngày 21/09/2000. Do thuật toán này đã được công bố trước khi có đăng ký bảo hộ, vì thế sự bảo hộ hầu như không hề có giá trị bên Hoa Kỳ. Bên cạnh đó, nếu như công trình của Clifford Cocks đã được công bố trước đó thì bằng sáng chế RSA đã không thể đăng ký.

## **Mô tả quá trình hoạt động của RSA**

Hệ mã hóa RSA có một cơ chế hoạt động rất dễ hiểu. Cơ chế public key sẽ được chia sẻ một cách công khai nhưng RSA lại có thể tối ưu rất tốt cơ chế công khai và bảo mật. Trải qua những bước sau đây, RSA sẽ hoàn thiện mọi yếu tố để có thể hoàn thiện chức năng bảo mật.



*Quá trình dùng mã hóa bất đối xứng để trao đổi thông tin*

### ***Sinh khóa***

Sinh khóa làm nhiệm vụ tìm kiếm được 1 bộ phận có 3 số tự nhiên e, d, n cần thỏa mãn công thức sau đây:

***Med trùng m mod n***

Trong đó, giá trị d cần phải được bảo mật 1 cách tuyệt đối để khi có biết các giá trị khác là n, e hay m thì cũng không có cách nào để tìm được giá trị của d. Với công thức này, khóa RSA sẽ có cơ chế sinh hóa theo quy trình:

* Chọn ra 2 số nguyên tố là p và q.
* Tính phương trình: n=pq. Giá trị của n đóng vai trò modulus ở cả 2 loại private key và public key.
* Có một vài giả nguyên tố dựa trên Carmichael sẽ được tính toán và giữ bí mật.
* Chọn lấy 1 số e ở khoảng 1 và giữ nguyên tố n sao cho ước chung lớn nhất của 2 số này có giá trị bằng 1. Nghĩa là, giá trị e và giả nguyên tố n có cùng nguyên tố với nhau.
* Giá trị của d trùng với 1/e, viết theo 1 cách khác là de trùng 1. Số tự nhiên d lúc này chính là nghịch đảo của modulo của e theo công thức modulo mod λ(n).

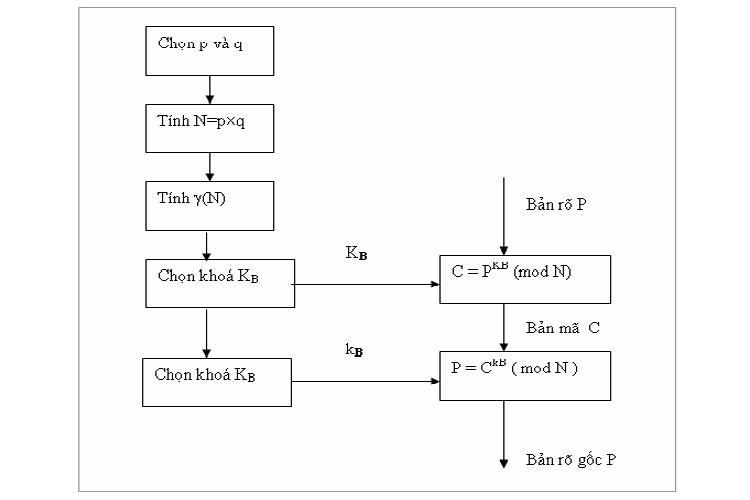
Lúc này, Public Key được tìm ra chính là bộ số (n,e) còn và private key bộ số (n,d). Nhiệm vụ của bạn là cần phải giữ cẩn thận private key và số nguyên tố p,d từ đó việc tính toàn các khóa sẽ trở nên một cách dễ dàng.

Đưa vào thực tiễn thực hành thì con người hay lựa chọn những giá trị e nhỏ để có được lợi thế giải mã nhanh chóng, thông thường thì e có giá trị = 65537. Ngoài công thức tính giả nguyên tố đã nêu trên thì hàm E φ(n) cũng có thể được sử dụng để thay thế công thức tính như sau: ***Euler φ(n) = (p-1)(q-1) =*** ***λ(n).***

Bạn có thể hiểu công thức trên một cách như sau: φ(n) là bội số của λ(n), vì thế, d cần thỏa mãn điều kiện ***de = 1 (mod φ(n)) = d ≡ 1/e (mod λ(n))***. Thuật toán này không hoàn toàn mang tới độ hoàn hảo tuyệt đối, vì nó có thể khiến giá trị d trở nên lớn hơn.

### ***Mã hóa và giải mã***

RSA được bảo mật 1 cách tuyệt đối và việc mở khóa chỉ người nhận mới có thể thực hiện được. Vì thế, nếu bạn là người nhận trong quá trình này, thì bạn đã biết cách giải mã chưa?



*Mã hóa và giải mã là như thế nào?*

Nếu bạn tìm hiểu trọn vẹn từ khâu mã hóa đến việc giải mã, sẽ giúp bạn có được một logic đúng để giải mã thành công. Bạn cần tìm hiểu một cách cụ thể về cách sử dụng public key (n, e) để có thể mã hóa và dùng Private key (n, d) để giải mã.

Nếu có M thì chuyển nó sang thành số tự nhiên m ở khoảng (0, n) và có thể đảm bảo giá trị m và n có cùng nguyên tố. Bạn hãy thêm kỹ thuật padding vào, sau đó có thể tiến hành mã hóa m để chuyển m thành c. Bạn có thể áp dụng theo công thức sau:

***c ≡ me mod n***

c sẽ được chuyển tới người nhận. Người nhận lúc này sẽ có nhiệm vụ là giải mã c để có thể lấy được giá trị của m bằng công thức sau:

***cd ≡ mde ≡ m mod n***

Sau đó, lấy giá trị m, bạn đảo ngược padding lại nhé thì sẽ lấy được thông tin gốc. Bạn có thể xem một ví dụ đơn giản sau đây:

* Nếu ta có giá trị p=5, q=7 thì n= pq = 35. Vì thế, φ(n) = 24

Sau đó, chọn e=5 vì UCLN của 5 và 24 là 1

Chọn d = 29 do ed -1 = 29\*5 -1 đều có giá trị chia hết cho 24.

* Lấy giả sử, giá trị m = 32 (cách) thì mã hóa m sẽ có thể thu được kết quả là: c = 32 ^ 5 % 35 = 2

Ngược lại, khi giải mã c sẽ có thể thu được giá trị của m: m = 2 ^ 29% 35 = 32.

m = 32 là giá trị ban đầu mà người nhận muốn và cần phải giải mã, cũng là giá trị người gửi cần mã hóa.

Công thức trên được áp dụng theo thuật toán đã được chứng minh, vì thế bạn có thể thử với bất kỳ giá trị nào của m theo công thức chuẩn RSA đã được công bố thì vẫn có được một kết quả chính xác.

RSA được bảo mật phần lớn là phụ thuộc vào chính khả năng phân tích thừa số nguyên tố từ các giá trị lớn. Lý do là chúng ta đã cung cấp rộng rãi chế độ public cho nên theo logic, việc phân tích các thừa số nguyên tố sẽ trở nên dễ dàng thì các private bị lộ cũng trở nên dễ dàng, khiến cho RSA không được bảo mật.

Dựa vào nguyên lý này, việc sinh khóa cần lựa chọn ngẫu nhiên các số p và q đã khiến việc phân tích các thừa số nguyên tố trở nên khó khăn gây khó mở khóa. Muốn vậy, p và q sẽ không có cùng độ dài. Có thể khiến máy tính cá nhân vẫn chưa thể thực hiện được nhiệm vụ này. Nhưng dưới thời đại công nghệ số phát triển như vũ bão hiện nay thì những siêu máy tính sẽ cùng con người tìm ra những phương án tính toán tốt nhất cho RSA. Đến nay, việc một người sở hữu máy tính lượng tử với những ưu thế về tốc độ cao có khả năng phá vỡ sự bảo mật trong RSA. Đối với máy tính lượng tử, nó cần rất nhiều thời gian để có thể hoàn thiện rồi mới tính đến việc phá vỡ sự bảo mật của RSA.

Nếu đi từ public key thì nó khó mà suy ra được Private key cho người khác và yêu cầu họ mã hóa thì bạn đang tạo điều kiện cho kẻ tấn công có được cơ hội tính toán ra public key một cách dễ dàng. Từ đó, thay đổi hoàn toàn nội dung hay cuộc thoại quan trọng đang cần nhờ RSA bảo mật. Mọi bảo mật sẽ trở nên vô nghĩa vì bí mật đã bị tấn công. Điều này thật sự rất nguy hiểm. Đặc biệt, bí mật đó liên quan đến chính trị, quân sự và những vấn đề quan trọng của một quốc gia.

1. Yêu cầu:

- Phân tích và lựa chọn giải thuật cần thiết và tối ưu để hệ thống hoạt động nhanh.

- Tạo được số nguyên tố lớn.

- Tạo được cặp khóa public key và private key từ số nguyên tố.

- Hiện thực được mã hóa và giải mã từ cặp khóa.

- Lưu cặp khóa thành file.

# CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## Các module của hệ thống

- Tìm số nguyên tố lớn khi cho số lượng bit của số nguyên tố lớn cần tìm.

- Tính ước số lớn nhất khi cho hai số nguyên lớn.

- Tính toán khóa giải mã d khi cho khoá mã hóa e và hai số nguyên tố lớn.

- Tạo bộ khóa ngẫu nhiên khi cho 2 số nguyên tố lớn.

- Mã hóa khi cho thông điệp và khóa mã hóa e và n.

- Giải mã khi cho thông điệp mã hóa và khóa giải mã d và n.

Các giải thuật cần thiết:

* 1. Thuật toán Euclid mở rộng.
  2. Thuật toán Miller-Rabin kiểm tra tính nguyên tố.
  3. Thuật toán Luỹ thừa mô-đun nhanh.

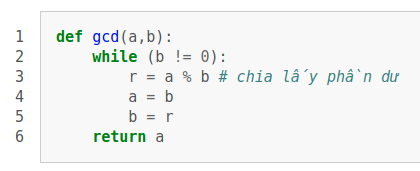
## Thuật toán Euclid mở rộng

Thuật toán Euclid là một trong những thuật toán cổ nhất được biết đến, từ khi nó xuất hiện trong cuốn Euclid’s Elements khoảng năm 300 trước công nguyên. Euclid khởi đầu đã trình bày rõ ràng vấn đề về phương diện hình học, như vấn đề tìm ra một thước đo chung cho độ dài hai đường thẳng, và thuật toán của ông đã xử lý bằng cách lặp lại phép trừ đoạn dài hơn cho đoạn ngắn hơn. Tuy nhiên, thuật toán đã hầu như không được phát hiện ra bởi Euclid và nó đã có thể được biết đến sớm hơn 200 năm. Nó cũng đã được biết đến bởi Eudoxus of Cnidus (khoảng năm 375 trước công nguyên) và Aristotle (khoảng năm 330 trước công nguyên).

Mô tả thuật toán:

Cho 2 số tự nhiên a và b, không đồng thời bằng 0: kiểm tra nếu b bằng 0, thì a là ước chung lớn nhất (UCLN). Nếu không, lặp lại xử lý sử dụng b và phần còn lại sau khi lấy a chia cho b. Phần còn lại sau khi chia a cho b thường được viết là a mod b. Các thuật toán này có thể sử dụng trong bất kì hoàn cảnh nào khi còn phần dư. Điều này bao gồm các nhóm đa thức như nhóm số nguyên Gauss. Thuật toán không chỉ áp dụng cho số tự nhiên mà còn áp dụng cho nhiều trường hợp tổng quát khác sẽ được mô tả chi tiết sau.

Dựa trên định lý GCD(a,b)=GCD(b,a mod b).



**Phương trình diophantine: ax+by=c** (1)

**Theo định lý Bézout (Bézout’s identify):** Cho hai số nguyên a, b khi đó luôn tồn tại hai số x, y sao cho: ax + by = gcd(a, b)

Người ta cũng chứng minh được phương trình trên có nghiệm khi và chỉ khi gcd(a, b) = c. Và như thế có nghĩa là phương trình diophante có thể có vô số nghiệm, và từ mỗi một nghiệm ta có thể sinh ra những nghiệm khác.

Định nghĩa số nguyên k sao cho: c = gcd(a, b)k. Chia vế theo vế phương trình (1) cho k ta được phương trình mới

as + bt = gcd(a, b) <=> a(sk) + b(tk) = gcd(a,b)k = c (2)

Dễ thấy x = sk và y = tk chính là một nghiệm của phương trình (1). Giả sử x_{1}, y_{1} là một nghiệm khác của phương trình (1), ta có:

a(x_{1} - x) + b(y_{1} - y) = 0

Suy ra: a(x_{1} - x) = b(y_{1} - y) (3), nghĩa là a là ước của b(y - y_{1}), và  frac{a}{gcd(a, b)} là ước của y - y_{1}. Nên ta có: y = y_{1} + rfrac{a}{gcd(a, b)} với r là một số nguyên. Thay vào (3), ta được:

a(x_{1} - x)=rb(\frac{a}{gcd(a, b)})

Trong đó:

gcd(a, b)a(x_{1} - x) = rba

hay x = x_{1} - r\frac{b}{gcd(a, b)}

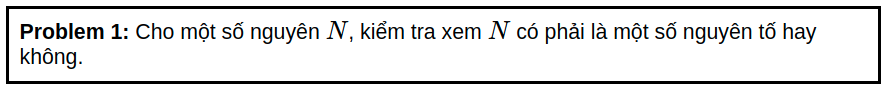
Như vây, nếu ax_{1} + by_{1} = c  có một nghiệm bất kì thì tất cả các nghiệm sẽ có dạng:

x = x_{1} - r\frac{b}{gcd(a, b)}, y = y_{1} + r\frac{a}{gcd(a, b)}

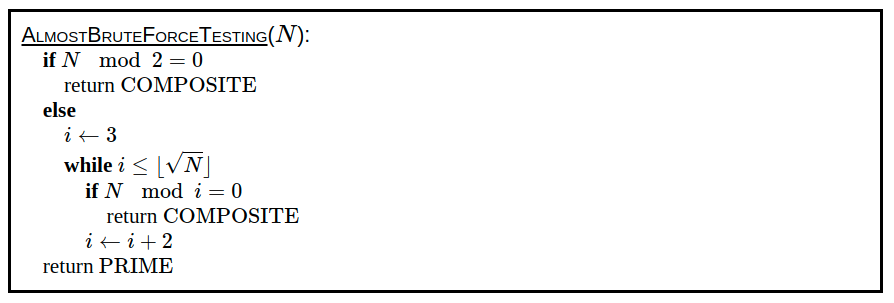
Ví dụ: Tìm tất cả các nghiệm của phương trình 258x + 147y = 369.

* Đầu tiên, ta xét gcd(258, 147) = 3, vì 369 \vdots  3 nên phương trình chắc chắn có nghiệm.
* Áp dụng thuật toán Euclid, ta có:
  + 258 = 147(1) + 111
  + 147 = 111(1) + 36
  + 111 = 36(3) + 3
  + 36 = 3(12)
* Bây giờ, ta tìm cách biểu diễn 3 bằng một mối quan hệ tuyến tính giữa 258 và 147. Dựa vào trên, ta có:
  + 3 = 111 - 36(3)
  + = 111 - 3(147 - 111)
  + = 4(111) - 3(147)
  + = 4(258 - 147) - 3(147)
  + = 4(258) - 7(147)
* Tiếp theo, ta viết 258(4) + 147(-1) = 3, và nhân phương trình cho 123 vì 3\times123 = 369. Ta được:
  + 258(492) + 147(-861)= 369.
* Vậy một nghiệm của phương trình là: x = 492, y = -861, và các nghiệm khác có dạng:
  + x = 492 - \frac{147r}{3} = 492 - 49r,
  + y = -861 + \frac{258r}{3} = 86r - 861 (r\in \mathbb{Z})

## Thuật toán tra tính nguyên tố.

Trong phần này chúng ta sẽ tìm hiểu bài toán kiểm tra tính nguyên tố (primality testing) của một số nguyên. Đây là một bài toán có rất nhiều ứng dụng trong các lĩnh vực của tin học, đặc biệt trong bảo mật thông tin. Bài toán phát biểu như sau:

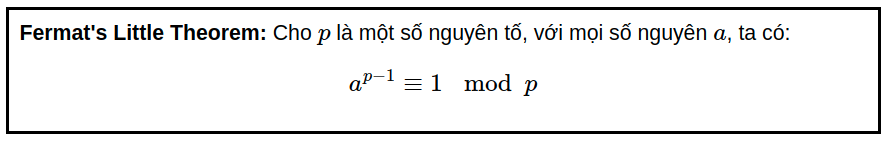
Ta có thuật toán gần như là bruteforce như dưới:



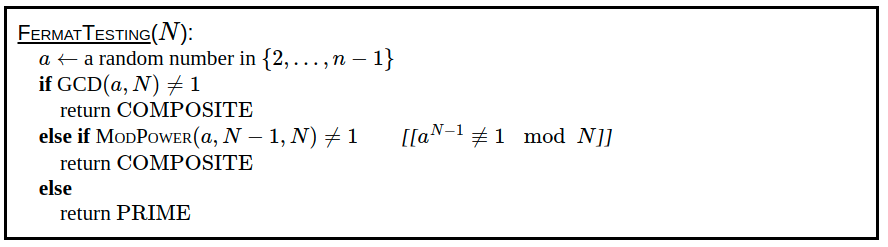
Độ phức tạp của giải thuật này không phù hợp để kiểm tra số nguyên tố rất lớn.

## Thuật toán Fermat

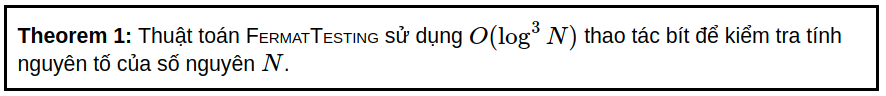
Định lý nhỏ Fermat phát biểu như sau:



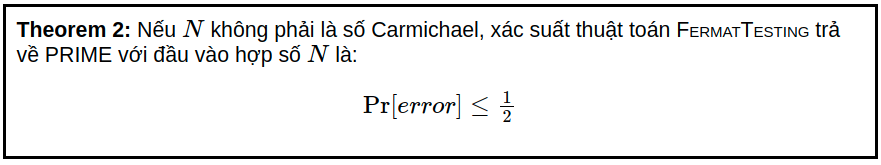
Dựa trên Fermat's Little Theorem, ta có thuật toán kiểm tra số nguyên tố của một số nguyên:



Ta có độ phức tạp của giải thuật như sau:

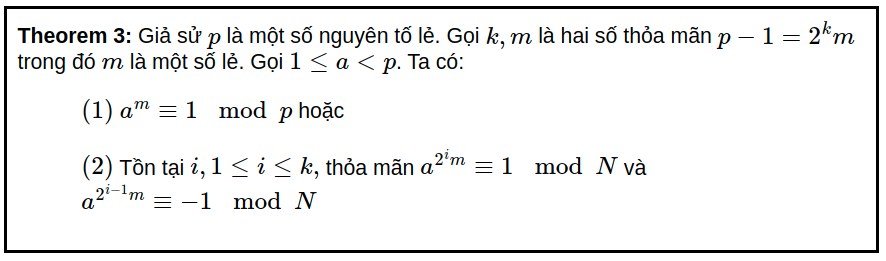


Vì giải thuật này không cho biết chính xác số N có phải là số nguyên tố hay không, mà chỉ cho biết xác suất để số N là số nguyên tố, do đó chúng ta sẽ thực hiện việc test lại nhiều lần để xác suất số N không phải là số nguyên tố giảm gần về 0. Khi đó ta có thể coi số N là số nguyên tố. Tuy nhiên, khi số N là số Carmichael thì tính chính xác của giải thuật giảm.

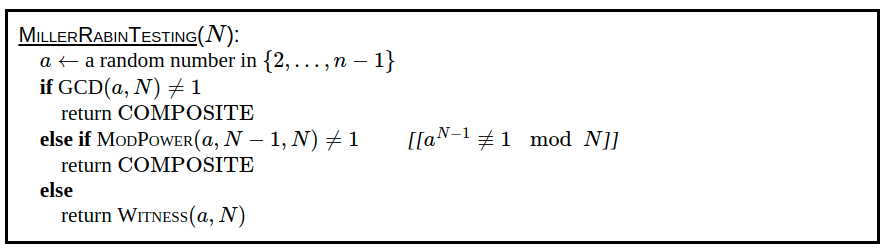


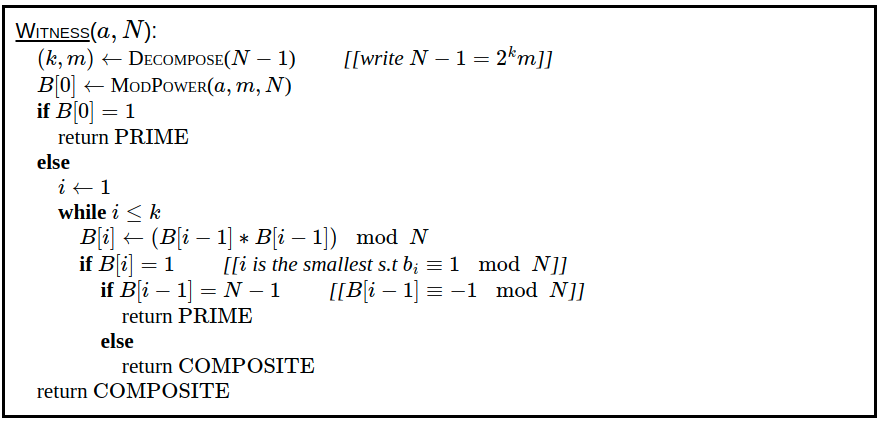
## Thuật toán Miller-Rabin

Thuật toán Miller-Rabin có độ chnh xác khá cao ngay cả khi đầu vào là số Carmichael. Thuật toán được xây dựng dựa trên định lí sau:



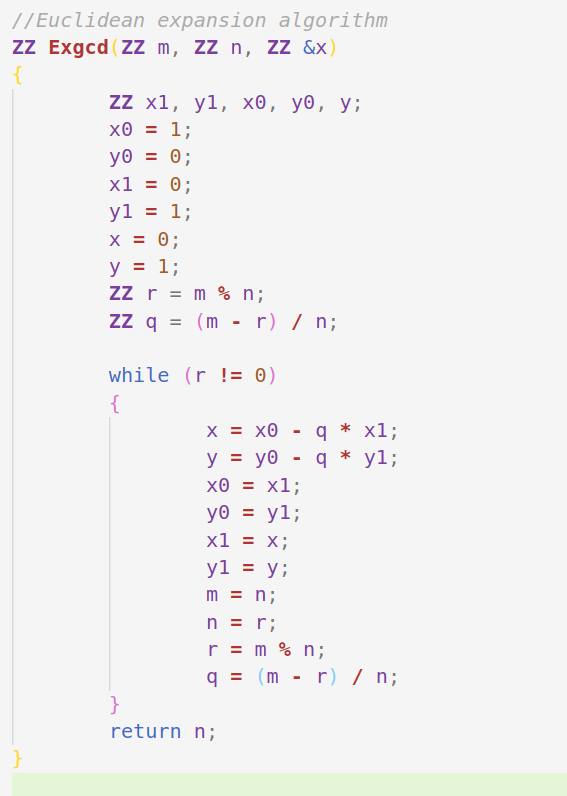
Mã giả của giải thuật:



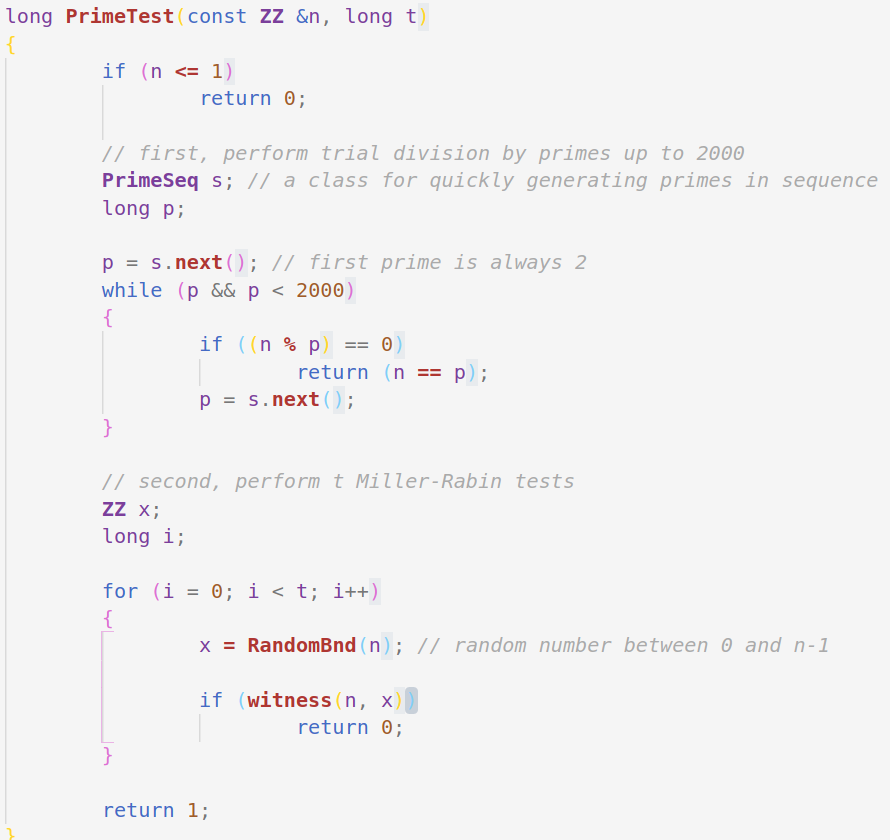


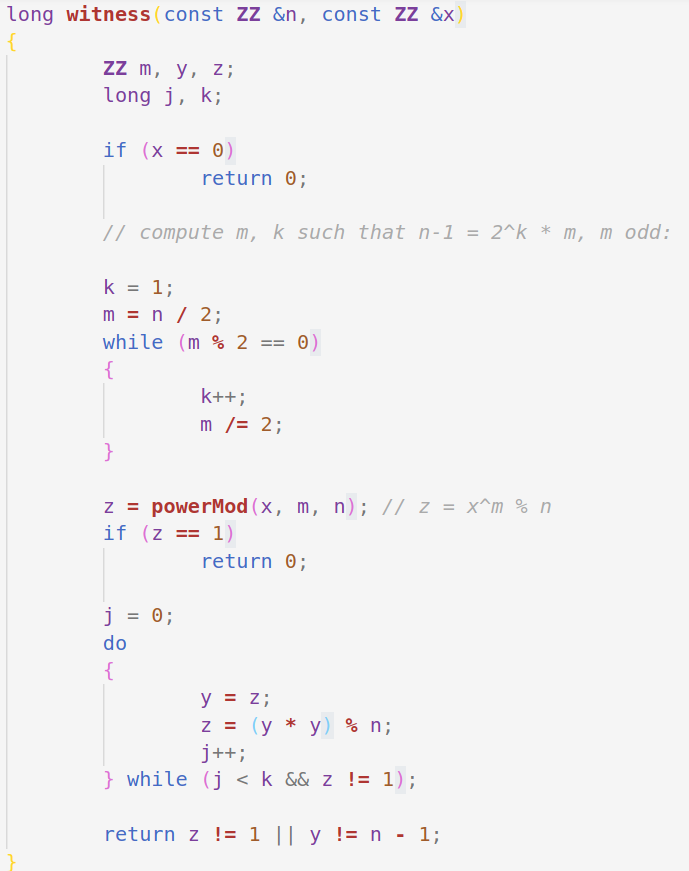
# CHƯƠNG 3: HIỆN THỰC VÀ ĐÁNH GIÁ HỆ THỐNG

## Thuật toán Euclid mở rộng.

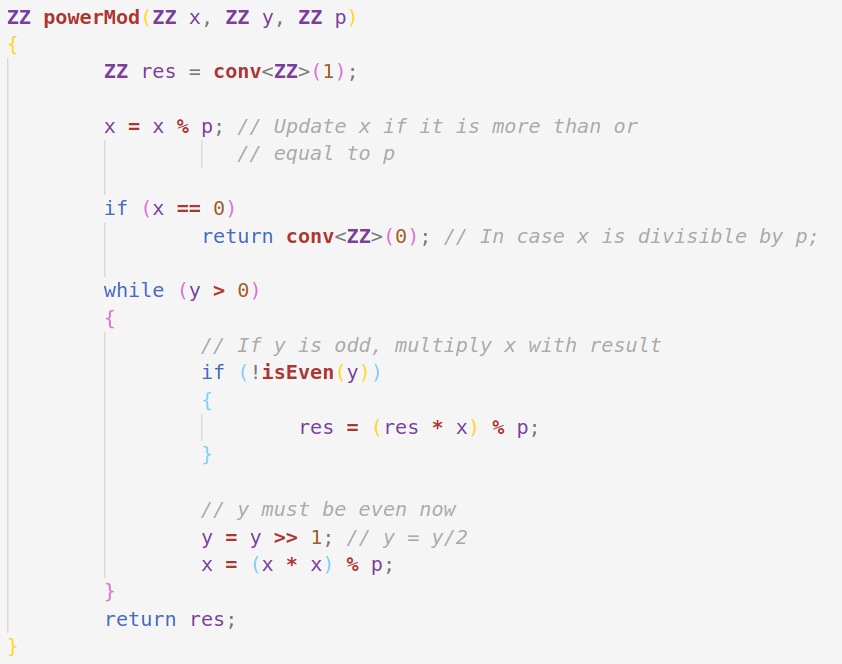


## Thuật toán Miller-Rabin kiểm tra tính nguyên tố.



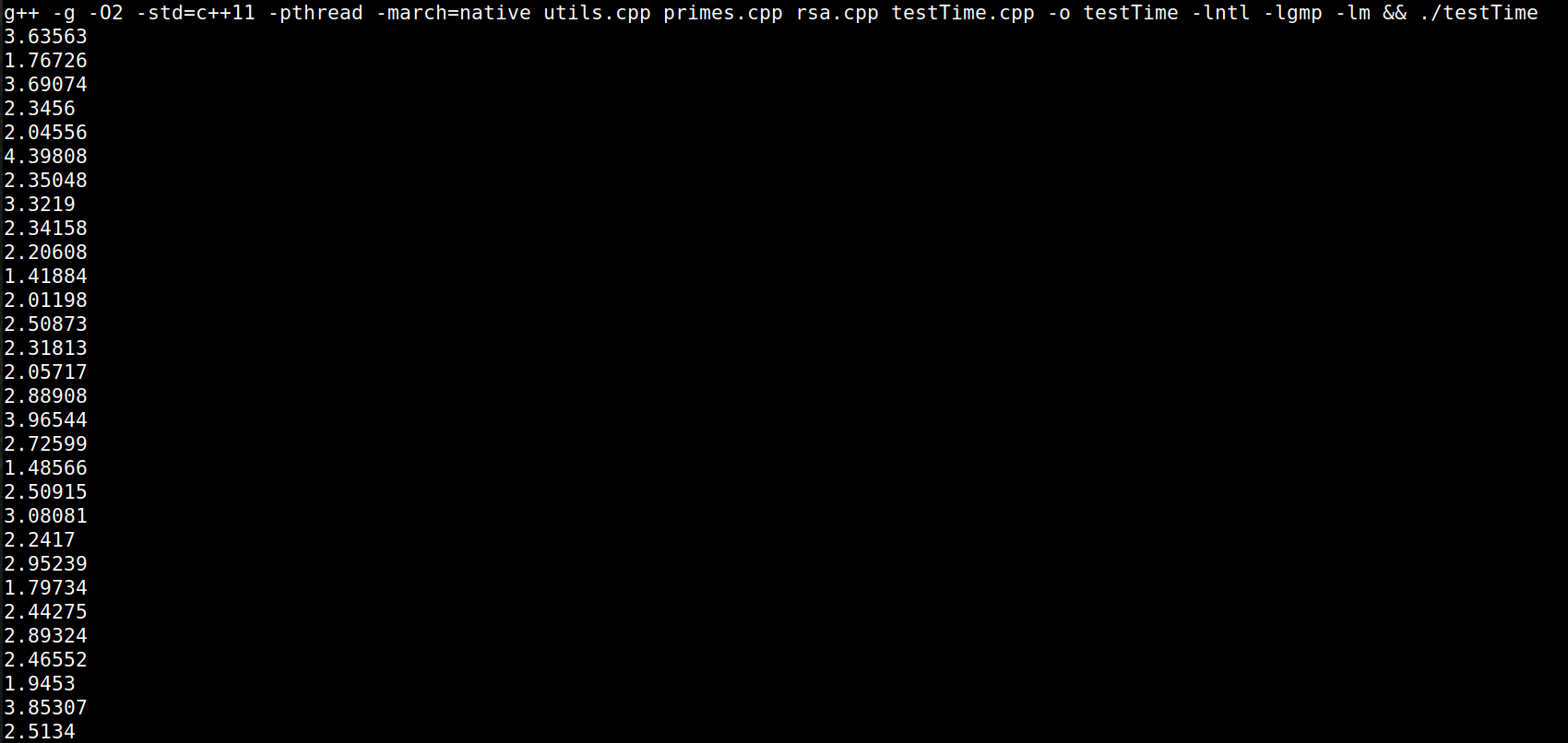


## Thuật toán Luỹ thừa mô-đun nhanh



## Thời gian thực thi

Tốc độ trung bình 30 lần thực thi: Đối với số nguyên tố có độ dài là 2048 bits và 100 vòng thử Miller-Rabin là: 2.605953333



# CHƯƠNG 4: KẾT LUẬN

Hệ thống đã hoàn thành cơ bản việc tạo ra các số nguyên tố, tạo cặp public key và private key. Đã hoàn thành việc mã hóa và giải mã sử dụng public key và private key.

## Ưu điểm

* Hệ thống đơn giản, mã nguồn nhỏ gọn.
* Tốc độ xử lý nhanh, có thể tạo ra được các cặp khóa với thời gian trung bình dưới 3 giây.

## Nhược điểm

* Mã hóa và giải mã đang trực tiếp từ số chứ không phải từ string.
* Vì mã nguồn đơn giản nên còn chưa tối ưu về bộ nhớ.
* Chưa định dạng lại cặp khóa bằng chuẩn Base64.

## Hướng phát triển

* Áp dụng tiêu chuẩn RSA của thế giới.
* Hiện thực phần mã hóa và giải mã từ string.
* Định dạng lại cặp khóa để thân thiện hơn với người dùng.

# TÀI LIỆU THAM KHẢO

RSA là gì? Cơ chế hoạt động của RSA ra sao?:

<http://nghelaptrinh.net/rsa-la-gi/>

A Tour of NTL:

<https://libntl.org/doc/tour.html>

Examples: Big Integers:

<https://libntl.org/doc/tour-ex1.html>

Obtaining and Installing NTL for UNIX:

<https://libntl.org/doc/tour-unix.html>

Thuật toán Euclid mở rộng - Extended Euclidean Algorithm:

<https://laptrinhthidau.wordpress.com/2016/08/23/thuat-toan-euclid-mo-rong/>

Thuật toán Miller-Rabin kiểm tra tính nguyên tố - Miller-Rabin Primality Testing: <https://www.giaithuatlaptrinh.com/?p=278>

Mã nguồn:

<https://github.com/andrewkiluk/RSA-Library/blob/master/rsa.c>

<https://github.com/ijleesw/RSA-NTL/blob/master/src/User.hpp>

<https://github.com/thangdnsf/PrGlib>

RSA encryption algorithm c++ simple implementation:

<https://www.programmersought.com/article/2948859498/>

# PHỤ LỤC

NTL là một thư viện C ++ di động, hiệu suất cao cung cấp cấu trúc dữ liệu và thuật toán cho các số nguyên có độ dài tùy ý; cho vectơ, ma trận và đa thức trên các số nguyên và trên các trường hữu hạn; và cho số học dấu phẩy động chính xác tùy ý.

NTL cung cấp triển khai chất lượng cao của các thuật toán hiện đại cho:

* Số nguyên độ dài tùy ý số học và số học dấu phẩy động chính xác tùy ý;
* Số học đa thức trên các số nguyên và trường hữu hạn bao gồm số học cơ bản, phân tích nhân tử đa thức, kiểm tra tính bất khả quy, tính toán các đa thức tối thiểu, dấu vết, định mức và hơn thế nữa;
* Giảm cơ sở mạng tinh thể, bao gồm các triển khai rất mạnh mẽ và nhanh chóng của Schnorr-Euchner, giảm khối Korkin-Zolotarev, và phương pháp cắt tỉa Schnorr-Horner mới cho khối Korkin-Zolotarev;
* Đại số tuyến tính cơ bản trên số nguyên, trường hữu hạn và số dấu phẩy động chính xác tùy ý.

Số học đa thức của NTL là một trong những số học nhanh nhất có sẵn ở bất cứ đâu và đã được sử dụng để thiết lập "kỷ lục thế giới" về nhân tử đa thức và xác định thứ tự của đường cong elliptic.

Mã giảm mạng của NTL cũng là một trong những mã tốt nhất có sẵn ở bất kỳ đâu, về cả tốc độ và độ mạnh mẽ, và là một trong số ít cách triển khai giảm khối Korkin-Zolotarev với kinh nghiệm cắt tỉa Schnorr-Horner. Nó đã được sử dụng để "bẻ khóa" một số hệ thống mật mã.

**Cài đặt thư viện**

Quy trình này sẽ hoạt động trên hầu hết các đĩa cứng giống Unix hoặc Unix (bao gồm cả Mac OS và Windows với công cụ MinGW hoặc Cygwin).

Để lấy mã nguồn và tài liệu cho NTL, hãy tải xuống ntl-xxx.tar.gz, đặt nó vào một thư mục, sau đó làm việc trong thư mục này, hãy thực hiện như sau. Ở đây, "xxx" biểu thị số phiên bản hiện tại.

% gunzip ntl-xxx.tar.gz

% tar xf ntl-xxx.tar

% cd ntl-xxx / src

% ./configure

% làm

% kiểm tra

% sudo thực hiện cài đặt

Điều này sẽ xây dựng, kiểm tra và cài đặt NTL trong / usr / local. Để điều này hoạt động, GMP phải được cài đặt (hầu hết các bản phân phối Unix đã được cài đặt GMP, nhưng hãy xem trang này để biết thêm chi tiết). Nếu bạn thực sự không muốn sử dụng GMP, bạn có thể chuyển tùy chọn NTL\_GMP\_LIP = tắt để cấu hình; tuy nhiên, NTL sẽ chạy nhanh hơn đáng kể với GMP, vì vậy điều này không được khuyến khích.