ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Lưu Minh Phát - 20127061
Nguyễn Quốc Huy – 20127188
Trần Hoàng Minh Quang – 20127299
Đoàn Duy Phong - 20127405
Lê Cung Tiến - 20127682

BÁO CÁO

[GIÁO VIÊN HƯỚNG DẪN]
PGS.TS. Nguyễn Đình Thúc
Thầy Ngô Đình Hy

NHẬP MÔN MÃ HÓA MẬT MÃ

Thành phố Hồ Chí Minh – 2022

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

BÀI TẬP 1

NHẬP MÔN MÃ HÓA MẬT MÃ

Thành phố Hồ Chí Minh – 2022

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến thầy PGS.TS. Nguyễn Đình Thúc, thầy Ngô Đình Hy và trường Đại Học Khoa Học Tự Nhiên đã luôn tạo điều kiện tốt nhất để chúng em có thể hoàn thành bài báo cáo một cách tốt đẹp.Trong suốt quá trình thực hiện, thầy đã luôn giúp đỡ và hướng dẫn để chúng em có cơ hội làm việc tốt nhất. Qua đồ án lần này, chúng em đã có thể hiểu thêm về mã hóa mật mã, cũng như có thể thực hiện những yêu cầu đơn giản. Hơn nữa, chúng em còn rèn luyện cho bản thân kĩ năng làm việc nhóm, kĩ năng phân chia công việc và quản lí thời gian tốt hơn.

Bài báo cáo đồ án thực hiện trong khoảng thời gian gần 2 tuần. Trong suốt quá trình thực hiện đồ án, do vẫn còn nhiều hạn chế về kinh nghiệm nên chúng em không thể nào tránh khỏi những sai sót. Chúng em rất mong nhận được những góp ý kiến, chỉ bảo để có thêm kinh nghiệm và cũng mong thầy cô bỏ qua.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

YÊU CẦU

1 Lý thuyết

Trong mật mã học, việc kiểm tra một số tự nhiên n có phải số nguyên tố hay không rất quan trọng và được sử dụng thường xuyên. Phương pháp đơn giản và dễ thực hiện nhất là kiểm tra xem n có chia hết cho các số tự nhiên từ 2 đến n − 1 hay không, nhưng cách này lại rất tốn thời gian và không phù hợp khi n rất lớn (từ 2048 bits trở lên). Nhiều phương pháp kiểm tra số nguyên tố đã được tạo ra nhằm cải thiện thời gian kiểm thử, trong phần này, các bạn sinh viên được yêu cầu tìm hiểu và so sánh các phương pháp đó.

Yêu cầu 1. Tìm hiểu về các phương pháp sau:

• Phép kiểm tra tính nguyên tố của Fermat.

• Phép kiểm tra tính nguyên tố của Miller-Rabin. Phần tìm hiểu phải chỉ ra được các phương pháp trên thực hiện việc kiểm tra như thế nào, độ phức tạp trên lý thuyết của thuật toán, và thuật toán dựa trên các định lý nào.

2. So sánh về độ phức tạp và tính đúng đắn của các thuật toán ở trên.

2 Thực hành

Kích thước khóa đóng vai trò quan trọng trong việc đảm bảo tính an toàn của hệ mã RSA. Trong thực tế, từ 2015, NIST đã đưa ra khuyến cáo về kích thước khóa tối thiểu của RSA là 2048 bits1 . Trong phần này, các bạn sinh viên cần cài đặt chương trình sinh khóa cho thuật toán mã hóa RSA với độ dài khóa cho trước và đưa ra đánh giá về độ phức tạp cũng như thời gian vận hành của chương trình. Ngoài ra, trong quá trình sinh số nguyên tố, cần sử dụng 1 trong 2 thuật toán kiểm tra số nguyên tố đã tìm hiểu ở trên.

Yêu cầu

Cài đặt: Sử dụng ngôn ngữ C/C++, không dùng thư viện hỗ trợ lưu trữ/tính toán số nguyên lớn

• Cài đặt 1 trong 2 thuật toán kiểm tra số nguyên tố trong phần Lý thuyết.
• Cài đặt chương trình sinh khóa cho thuật toán mã hóa RSA. Chương trình cần cho phép người dùng chọn 1 trong 3 độ dài khóa là 512 bits, 1024 bits và 2048 bits.

Báo cáo:

• Đánh giá ưu/nhược điểm và thời gian thực hiện của thuật toán kiểm tra số nguyên tố đã cài đặt.

• Báo cáo chi tiết về chương trình sinh khóa đã cài đặt, bao gồm:

– Cách chạy chương trình

– Thiết kế chương trình, cách lưu và tính toán số nguyên lớn, các sinh số nguyên tố lớn tương ứng với từng độ dài khóa.

– Thời gian thực hiện, ưu/nhược điểm của chương trình đã cài đặt.

3 Các quy định nộp bài

3.1 Các quy định chung

3.2 Mã nguồn

3.3 Báo cáo

PHÂN CÔNG CÔNG VIỆC

Giáo viên hướng dẫn môn Nhập Môn Mã Hóa Mật Mã

Giáo viên lý thuyết: PGS.TS Nguyễn Đình Thúc, thầy Ngô Đình Hy

Phân chia công việc:

Thành viên	Phân công	Tiến độ hoàn thành
Nguyễn Quốc Huy	Tìm hiểu về các phương pháp kiểm tra tính nguyên tố của Fermat và Miller-Rabin So sánh về độ phức tạp và tính đúng đắn của các thuật toán.	100%
Lê Cung Tiến	Cài đặt 1 trong 2 thuật toán kiểm tra số nguyên tố trong phần Lý thuyết.	100%
Lưu Minh Phát	Cài đặt chương trình sinh khóa cho thuật toán mã hóa RSA. Chương trình cần cho phép người dùng chọn 1 trong 3 độ dài khóa là 512 bits, 1024 bits và 2048 bits.	100%
Trần Hoàng Minh Quang	Viết báo cáo: Đánh giá ưu/nhược điểm và thời gian thực hiện của thuật toán kiểm tra số nguyên tố đã cài đặt.	100%
Trần Hoàng Minh Quang - Đoàn Duy Phong	Báo cáo chi tiết về chương trình sinh khóa đã cài đặt, bao gồm: Cách chạy chương trình Thiết kế chương trình, cách lưu và tính toán số nguyên lớn, các sinh số nguyên tố lớn tương ứng với từng độ dài khóa. Thời gian thực hiện, ưu/nhược điểm của chương trình đã cài đặt.	Quang (70%) Phong (0%)

TÌM HIỂU PHƯƠNG PHÁP

Phép kiểm tra tính nguyên tố của Fermat.

- Định lý
  - Phép kiểm tra Fermat dựa trực tiếp vào định lý nhỏ Fermat [1]
  - Định lý nhỏ Fermat:
    - Nếu p là một số nguyên tố, thì với số nguyên a bất kỳ, $a^p - a$ sẽ chia hết cho p.

- Nguyên lý thực hiện
  - Dựa vào định lý Fermat nhỏ bên trên ta có:

$$a^{p-1} \equiv 1 \pmod{p}$$

  - Để kiểm tra số *n* có là nguyên tố không, ta lấy ngẫu nhiên các số *a'* và kiểm tra xem đẳng thức trên có đúng không.
  - Nếu nó không đúng với một giá trị *a* nào đó thì *n* là hợp số.
  - Nếu đẳng thức đúng với nhiều giá trị của *a*, ta có thể nói rằng *n* là số nguyên tố với xác suất nào đó, hay là một số giả nguyên tố [2]
  - Nếu kết quả trả về $a^{n-1} \equiv 1 \pmod{n}$ trong khi *n* là hợp số được gọi là một giả Fermat.
  - Còn nếu kết quả trả về $a^{n-1} \neq 1 \pmod{n}$ thì chứng tỏ *n* là hợp số.

Ví dụ:

- Ở đây xét n = 103 có phải là số nguyên tố hay không
- Đầu tiên chúng ta chọn một số a bất kỳ, ví dụ a = 99
- Với a = 99 thì $99^{103-1} = 99^{102} \bmod 103 = 1$ (Thỏa nên kết quả sẽ trả về có thể là số nguyên tố)
- Tương tự chọn a = 28 thì $28^{103-1} = 28^{102} \bmod 103 = 1$ (Thỏa nên kết quả sẽ trả về có thể là số nguyên tố)
- Tương tự cho a = 57 thì $57^{103-1} = 57^{102} \bmod 103 = 1$ (Thỏa nên kết quả sẽ trả về có thể là số nguyên tố)
- Với xác suất như này chúng ta dù không chắc chắn nhưng có thể ngầm công nhận 57 là một số nguyên tố.
- **Trường hợp đặc biệt:**
- Với a = 2, n = 341, mặc dù 341 là hợp số (11 x 31), đẳng thức vẫn đúng: $2^{341-1} \equiv 1 \pmod{341}$.

⇨ **Vì vậy, càng nhiều số *a* đúng với đẳng thức trên, khả năng *n* là số nguyên tố càng tăng**

- Độ phức tạp thuật toán
  - Pseudo code [2]
    - Đầu vào: n - giá trị để kiểm tra tính nguyên tố, k - tham số xác định số lần kiểm tra tíntố
    - Đầu ra: Tổng hợp nếu n là hợp số, nếu không thì **có thể** là số nguyên tố

- Thuật toán:

Lặp k lần: Chọn một số ngẫu nhiên trong phạm vi [2, n – 2] Nếu $a^{n-1} \neq 1$ (mod n) thì trả về kết quả hợp số Nếu kết quả trên không bao giờ được trả về thì kết quả trả về sẽ là **có thể** là số nguyên tố

- Thời gian thi hành của thuật toán là O(k log(n)) với k là số lần kiểm tra với mỗi số a ngẫu nhiên, và n là giá trị ta muốn kiểm tra.

Phép kiểm tra tính nguyên tố của Miller-Rabin

- Định lý
  - Giống với Fermat, phép kiểm tra Miller-Rabin cũng dựa trực tiếp vào định lý nhỏ Fermat
- Nguyên lý thực hiện [3]
  - Để kiểm tra n có phải là số nguyên tố không, đầu tiên ta cần tìm 2 số k, q với k > 0 và q là số lẻ thỏa mãn
    n = $2^k * q + 1$
  - Ta lấy ngẫu nhiên các số a' trong phạm vi từ [2, n - 2].
  - Tính x = pow(a, q) mod n
  - Nếu giá trị của x trả về 1 hoặc n – 1 thì đánh dấu là **có thể** là số nguyên tố
  - Nếu giá trị của x trả về khác 1 hoặc n – 1 thì thực hiện vòng lặp k – 1 lần
  - Chạy vòng lặp tính toán giá trị x = (x * x) mod n
  - Nếu giá trị của x = 1 thì trả về kết quả hợp số
  - Nếu giá trị của x = n – 1 thì trả về kết quả **có thể** là số nguyên tố

Ví dụ 1:

- Nhập vào số n = 13 (Số cần kiểm tra)
- Phân tích n = 13 = $2^2 * 3 + 1$ (k = 2, q = 3)
- Giả sử a = 4, tính giá trị x = 4^3 mod 13 = 12 (12 = n – 1 nên trả về có thể là số nguyên tố)

Ví dụ 2:

- Nhập vào số n = 221 (Số cần kiểm tra)
- Phân tích n = 2^2 * 55 + 1 (k = 2, q = 55)
- Giả sử a = 5, tính giá trị x = 5^{55} mod 221 = 168 (168 khác 1 và khác n – 1 nên sẽ chạy vòng lặp k – 1 lần, ở đây k – 1 = 1 lần)
- Tính giá trị của x = (168 * 168) mod 221 = 157 (157 khác n – 1 nên trả về kết quả hợp số)
- Đúng vì 221 = 13 * 17
- **Nhưng** nếu a = 21, tính giá trị x = 21^{55} mod 221 = 200 (200 khác 1 và khác n – 1 nên sẽ chạy vòng lặp k – 1 lần, ở đây k – 1 = 1 lần)
- Tính giá trị của x = (200 * 200) mod 221 = 220 (220 = n – 1 nên trả về kết quả có thể là số nguyên tố)
  - ⇨ **Như vậy, chạy càng nhiều số *a* thì xác suất *n* là số nguyên tố càng cao**

- Độ phức tạp thuật toán
  - Pseudo code
  - Đầu vào: n - giá trị để kiểm tra tính nguyên tố, d – số lẻ sau khi phân tích
  - Đầu ra: Tổng hợp nếu n là hợp số, nếu không thì **có thể** là số nguyên tố
  - Thuật toán:

Phân tích n = $2^k * q + 1$
Chọn một số ngẫu nhiên trong phạm vi [2, n – 2]
Tính giá trị x = pow(a, d) mod n
Nếu x = 1 hoặc x = n – 1, trả về có thể là số nguyên tố
Nếu không, chạy vòng lặp k – 1 lần
Chạy vòng lặp tính toán giá trị x = (x * x) mod n
Nếu giá trị của x = 1 thì trả về kết quả hợp số
Nếu giá trị của x = n – 1 thì trả về kết quả có thể là số nguyên tố

  - Thời gian thi hành của thuật toán này là O(k log(3*n*)), trong đó n là số được kiểm tra tính nguyên tố và k là số vòng được thực hiện

So sánh thuật toán

- Dù cả hai giải thuật đều dựa trên định lý Fermat nhỏ nhưng chúng ta có thể thấy giải thuật Miller-Rabin (**O(k log(3n))**) có độ phức tạp cao hơn giải thuật Fermat (**O(k log(n))**), điều này cũng tương đương về tính đúng đắn của thuật toán khi giải thuật Miller-Rabin có tính chính xác cao hơn là giải thuật Fermat. Nguyên nhân là do:
  - Giải thuật Miller-Rabin phát triển từ giải thuật Fermat (Nếu với cùng 1 bộ số mà giải thuật Miller-Rabin trả về kết quả đúng thì giải thuật Fermat sẽ đúng nhưng sẽ không có chiều ngược lại)
  - Giải thuật Miller-Rabin sử dụng thêm việc kiểm tra n có phải là số nguyên tố không bằng cách xét kết quả của $x^2 = 1 \bmod n$ có ra được $x = \pm 1$ hay không.

GIỚI THIỆU THUẬT TOÁN

1. Thuật toán Fermat' little. [4] [5]

▪ Ưu điểm:

- Đơn giản, dễ cài đặt.
- Thời gian kiểm tra nhanh.
- Nếu là số nguyên tố sẽ luôn cho kết quả đúng.

▪ Nhược điểm:

Thuật toán Fermat' little sẽ kém hiệu quả nếu:

- Với a^{n-1} = 1 (mod n) trong một số trường hợp, với a = 2 và n = 341 thì đẳng thức trên đúng, trong khi 341 là hợp số (341 = 31 * 11).
- Với số Carmichael (là các hợp số thỏa mãn b^{n-1} = 1 (mod n)) thì với mọi số nguyên tố a cùng nhau với n, vẫn cho ra a^{n-1} = 1 (mod n). Ví dụ: 561, 1105, 1729,…

▪ Khắc phục nhược điểm:

- Sử dụng vòng lặp để phát sinh nhiều số a, giúp cho việc kiểm tra số nguyên tố trở nên đúng đắn hơn.

▪ Chạy thực nghiệm:

2. Chương trình sinh khóa cho thuật toán mã hóa RSA. [6] [7] [8] [9]

- **Cách chạy chương trình:**

```
C:\Users\This Pc\Downloads>a.exe
------  KeySize  ------
|    [1]: 512 bits    |
|    [2]: 1024 bits   |
|    [3]: 2048 bits   |
-----------------------
Select your choice: 1
Key Gen Size: 512
p: 2067599129
q: 2025330179
n = p*q = 4187570914037814091
Phi(n) = (p - 1) * (q - 1) = 4187570909944884784
e: 30419
d: 171570725224267
```

- **Thiết kế chương trình:**

Mục đích: Sinh khóa cho thuật toán RSA.

Hàm chức năng:

- **uint64_t getBigPrime( ):** sinh ra số nguyên tố có độ dài cố định (32 bit) => để sinh ra RSA key cuối cùng là 64 bit cần có độ dài số nguyên tố ban đầu là 64/2 bit. Tương tự với RSA key là 512 bit => 256 bit, 1024 bit => 512 bit, 2048 bit => 1024 bit.
- Để có thể in ra số nguyên tố ta sử dụng hàm **uint64_t getRandom64()** để in ra một biến candidate.
- **uint64_t getLowLevelPrime()** để kiểm tra biến candidate có phải số nguyên tố hay không.

- **Cách lưu và tính toán số nguyên lớn, các sinh số nguyên tố lớn tương ứng với từng độ dài khóa:**

- Lưu và xử lý thao tác dưới dạng chuỗi. Cách sinh số nguyên tố dựa trên bits thông qua dùng bitset, mt199073 random_device.

- **Thời gian thực nghiệm.**

- Không thể thống kê do chưa thể lưu số có độ dài lớn hơn 64 bit.

- **Ưu điểm:**

- An toàn và đáng tin cậy.
- Không thể mạo danh với private key.
- Dễ dàng phát hiện văn bản đã kí có bị sửa đổi hay không.

- **Nhược điểm:**

- Dễ bị hóa giải nếu tìm được lỗ hổng hay đoán được bộ sinh số ngẫu nhiên p và q.
- Chiều dài chưa đủ lớn sẽ dễ bị tách thành thừa số nguyên tố để tìm p và q.
- Tốc độ mã hóa chậm.
- Có thể bị lỗi vì mã hóa hoàn chỉnh yêu cầu cả mã hóa đối xứng và mã hóa bất đối xứng.

TÀI LIỆU THAM KHẢO

[1]: https://en.wikipedia.org/wiki/Fermat%27s_little_theorem

[2]: https://en.wikipedia.org/wiki/Fermat_primality_test

[3]: https://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test

[4]: https://codelearn.io/sharing/thuat-toan-kiem-tra-tinh-nguyen-to#:~:text=Ki%E1%BB%83m%20tra%20Fermat

[5]: https://www.geeksforgeeks.org/primality-test-set-2-fermet-method/

[6]: https://www.geeksforgeeks.org/how-to-generate-large-prime-numbers-for-rsa-algorithm/

[7]: https://stackoverflow.com/questions/60733508/stdmt19937-generator-returns-the-same-values-each-execution-why

[8]: https://vi.wikipedia.org/wiki/RSA_(m%C3%A3_h%C3%B3a)

[9]: https://www.youtube.com/watch?v=lAN6MqV144g