SINH SỐ NGUYÊN TỐ LỚN DÙNG TRONG MẬT MÃ

Nguyễn Đức Thắng, Trần Vĩnh Đức
Khoa Toán-Tin, Trường Đại học Thăng Long
Email: thangdn.tlu@outlook.com
Email: tranvinhduc@gmail.com

Tóm tắt:*Bài báo này mô tả phương pháp sinh ngẫu nhiên các số nguyên tố mạnhđể sử dụng trong cáchệ mật mã công khai và chữ ký số dựa trên RSA. Cụ thể, thuật toán Rabin-Miller và thuật toán Gordon sẽ được trình bày một cách chi tiết.*

Từ khóa:RSA, chữ ký số, mật mã.

1. Giới thiệu chung

Bài báo này mô tả phương pháp sinh số nguyên tố lớn dùng trong hệ thống chữ ký số của Trường Đại Học Thăng Long. Do yêu cầu của hệ thống, việc sinh số phải đảm bảo:

1. Số được sinh phải có độ dài 3072 bit,
2. Số được sinh phải ngẫu nhiên (khó dự đoán), và
3. Thời gian sinh mỗi số phải đủ nhanh, thời gian trung bình chỉ vài giây trên máy tính cấu hình không mạnh.

Định nghĩa 1. Số nguyên dương $p > 1$ là *nguyên tố* nếu p không chia hết cho số nguyên dương nào ngoài *1* và p. Ngược lại p là *hợp số*.

Ta định nghĩa hàm phân phối của các số nguyên tố $\pi(n)$ là số lượng số nguyên tố nhỏ hơn hoặc bằng n.

Ví dụ: $\pi(10)$= 4 vì có đúng bốn số nguyên tố nhỏ hơn *10* là *2, 3, 5, 7*.

Định lý sau đây cho ta ước lượng xấp xỉ hàm $\pi(.)$.

Định lý 1.

$$\lim_{n\to\infty}\left(\frac{\pi(n)}{n/\ln n}\right) = 1$$

Nói cách khác, giá trị $\pi(n)$ xấp xỉ bằng với $n/\ln n$ khi n lớn.

Ví dụ:Với $n = 10^9$ ta có

$$\pi(n) = 50847534 \quad \text{và} \quad n/\ln n = 48254942$$

Sai số ở đây là *6%*.

Vậy nếu ta lấy ngẫu nhiên một số nguyên dương$k$ bit, xác suất để số này là số nguyên tố bằng $1/\ln 2^k$. Về trung bình, ta cần $\ln 2^k$ lần thử để lấy được một số nguyên tố k bit.

Ví dụ:Nếu $k = 3072$ thì,về trung bình, lấy ngẫu nhiên $\ln 2^{3072} \approx 2130$ số, ta sẽ có được một số nguyên tố k bit.

Từ phân tích ở trên, về trung bình, thuật toán ngẫu nhiên dưới đây sẽ dừng sau 2130 bước lặp.

Thuật toán sinh số nguyên tố:

1. Chọn ngẫu nhiên một số nhị phân p độ dài 3072 bit.
2. Đặt cả hai bit cao nhất và bit thấp nhất của p lên 1.
3. Kiểm tra xem p có là số nguyên tố.
4. Nếu có thì trả ra số nguyên tố p. Còn nếu không thì quay lại Bước 1.

Trong Bước 2 của thuật toán,

- Ta đặt bit thấp nhất của p lên 1 đảm bảo rằng p là số lẻ;
- Và ta đặt bit cao nhất của p lên 1 để đảm bảo rằng số p sinh ra thỏa mãn

$$2^{3072-1} < p \leq 2^{3072} - 1$$

điều này là cần thiếtvì để đảm bảo an toàn cho thuật toán RSA thì các số nguyên tố sinhra phải đủ lớn.

Các bước 1 và bước 3 sẽ được phân tích ở mục 2 và mục 3.

2. Sinh số ngẫu nhiên

Hệ thống chữ ký điện tử dự kiến sẽ được cài đặt trên GNU/Linux. Chúng tôi dự định dùng hàm sinh dãy bit ngẫu nhiên của hệ điều hành này.

Hệ điều hành GNU/Linux sinh các dãy bit ngẫu nhiên từ các nguồn ngẫu nhiên của hệ thống (các thao tác chuột, các thao tác bàn phím, các chương trình chạy…). Các dãy bit này được lưu trữ trong tệp tin/dev/urandom. Trong trường hợp các dãy bit sinh ra chưa đủ ngẫu nhiên do entropy nhỏ, hệ thống sẽ sử dụng một hàm giả ngẫu nhiên an toàn. Phương pháp lấy dữ liệu ngẫu nhiên này được nhiều người sử dụng và theo chúng tôi biết, cho đến nay vẫn chưa có báo cáo nào về tính không an toàn của nó.

3. Thuật toán kiểm tra số nguyên tố

Thuật toán đơn định AKS cho phép kiểm tra xem liệu một số nguyên dương n có là nguyên tố trong thời gian $O(\log^{O(1)}(n))$. Tuy nhiên, nó không được sử dụng nhiều trong mật mã vì thời gian chạy chậm.

Chúng tôi cài đặt thuật toán ngẫu nhiên Rabin-Miller kết hợp với một số bước tiền xử lý để tăng tốc thuật toán.

Thuật toán của Rabin-Miller là thuật toán kiểu Monte Carlo. Nó cho phép kiểm tra một số nguyên n có là nguyên tố hay không với một xác suất sai có thể làm nhỏ tuỳ ý. Chúng tôi chọn xác suất sai số bằng $1/2^{128}$. Cụ thể, nếu thuật toán thông báo n là hợp số, vậy n chắc chắn là hợp số. Còn nếu thuật toán thông báo n là nguyên tố thì xác suất n là hợp số chỉ là $1/2^{128}$.

Phần còn lại trong mục này mô tả từng bước xây dựng và cải tiến thuật toán.

Trước hết định lý sau đây cho một thuật toán đơn giản để kiểm tra một số là nguyên tố.

Định lý 2 (Fermat). Nếu n là số nguyên tố thì, với mọi $1 \leq a < n$,

$a^{n-1} = 1 \bmod n$

Thuật toán như sau:

Thuật toán *CheckFermat(n)*:
Trả lại 0 nếu n là nguyên tố; 1 nếu n là hợp số.

1. **if** $(2^{n-1} \neq 1 \mod n)$ **return** 1; **else return** 0.

Tất nhiên, thuật toán trên có thể cho kết quả sai, nhưng chỉ có một kiểu sai: Nếu nó trả lại 0thì vẫn có khả năng n là hợp số, nhưng nếu nó trả lại 1 thì chắc chắn n là hợp số.

Ví dụ:Với cả bốn số $n = 314,516,645$ và 1105, thuật toán $CheckFermat$cho kết quả là 0, dù các số này đều là hợp số.

Câu hỏi tự nhiên là: Số trường hợp mà thuật toán này cho kết quả sai có nhiều không? Thật ngạc nhiên là số trường hợp sai rất hiếm. Trong $10,000$ số nguyên dương đầu tiên, thuật toán chỉ nhầm 22 số.

Thuật toán $Witness$ sau đây là một cải tiến của thuật toán trước dựa trên hai ý tưởng.

- Thay cơ sở 2 trong thuật toán $CheckFermat()$ bằng một số a bất kỳ;
- Kết hợp với Định lý 3 sau đây để hạn chế số trường hợp sai.

Định lý 3. Nếu n là số nguyên tố lẻ và $e \geq 1$ thì phương trình

$x^2 = 1 \bmod n^e$

chỉ có hai nghiệm $x = 1$ và $x = -1$.

Thuật toán *Witness(a, n)*
Trả lại 0 nếu n là nguyên tố; 1 nếu n là hợp số.

1. Xét t và u thỏa mãn $t \geq 1$, u lẻ và $n-1 = 2^t u$;
2. Gán $x_0 := a^u \mod n$;
3. **for** $i := 1, 2, \ldots, t$:
4. Gán $x_i := x_{i-1}^2 \mod n$;
5. **if** ($x_i = 1$ và $x_{i-1} \neq 1$ và $x_{i-1} \neq n-1$) **return** 1;
6. **if** $(x_t \neq 1)$ **return** 1; **else return** 0;

Thuật toán trên thực hiện tính $x_i = a^{n-1} \bmod n$ bằng cách lặp lại việc tính $x_i^2 \bmod n$ bắt đầu từ $x_0 = a^u \bmod n$. Nếu $x_t = 1$ thì n là số nguyên tố.Vòng lặp có thể kết thúc sớm nếu

$$x_i = x_{i-1}^2 \neq \pm 1 \bmod n$$

vì khi đó ta kết luận n là hợp số theo theo Định lý 3.

Thuật toán $Rabin - Miller$ dưới đây lặp lại slần tính $Witness(a,n)$để giảm xác suất sai.

Thuật toán *Rabin-Miller (n,s)*
Trả lại 0 nếu n là nguyên tố; 1 nếu n là hợp số.

1. **for** $j := 1, 2, \ldots, s$:
2. Chọn một số ngẫu nhiên a;
3. **if** (*Witness(a,n)* = 1) **return** 1;
4. **return** 0;

Xác suất sai của thuật toán $Rabin - Miller$ được chỉ ra bởi định lý sau.

Định lý 4.Với mỗi số nguyên lẻ $n > 2$ và số nguyên dương s, sai số của thuật toán $Rabin - Miller(n,s)$ nhiều nhất là 2^{-s}.

Thuật toán sau đây thêm một vài bước tiền xử lý để giảm thời gian chạy của thuật toán $Rabin - Miller$.

Thuật toán kiểm tra xem liệu số nguyên dương *n* có là nguyên tố

1. Tính sẵn 400 số nguyên tố đầu tiên: $q_1, q_2, \ldots, q_{400}$;
2. **if** (*n* chia hết cho q_i nào đó) **return** "hợp số";
3. **if**(*CheckFermat*(*n*) = 1) **return** "hợp số";
4. **if**(*Rabin-Miller*(*n*, 128) = 1) **return** "hợp số"; **else return** "nguyên tố";

Bước 1 và 2 nhằm *loại rất nhanh* các số chia hết cho *400* số nguyên tố đầu tiên. Bước 3*loại nhanh* một số lượng lớn các hợp số nhờ Định lý Fermat. Chúng tôi chọn cơ sở 2 để việc tính toán nhanh trên bit. Cuối cùng ta kiểm tra những số còn lại bằng thuật toán$Rabin \quad Miller$. Bước 4 cần nhiều tính toán.

Nếu thuật toán này thông báo "n là hợp số", vậy chắc chắn n là hợp số. Còn nếu nó thông báo "n là số nguyên tố", vậy xác suất n là hợp số chỉ là $1/2^{120}$.

4. Thuật toán sinh số nguyên tố mạnh

Thuật toán $p-1$ của *Pollard* là một trong những thuật toán hiệu quả để phân tích thừa số nguyên tố $n = pq$, khi p và q thỏa mãn một số tính chất đặc biệt.Để tránh phương pháp tấn công này, các số p và q nên là các *số nguyên tố mạnh*. Đây là lý do mà chuẩn *ANSI X9.31*yêu cầu sử dụng số nguyên tố mạnh để sinh khóa cho các hệ chữ ký điện tử dựa trên RSA.

Định nghĩa 2. Số nguyên tố p được gọi là *số nguyên tố mạnh* nếu nó thỏa mãn ba điều kiện sau:

1. $p-1$ có thừa số nguyên tố u đủ lớn;
2. $p+1$ có thừa số nguyên tố s đủ lớn; và
3. $u-1$ có thừa số nguyên tố t đủ lớn.

Năm 1984 *John Gordon* đã đề xuất một thuật toán hiệu quả để sinh số nguyên tố mạnh. Thuật toán Gordon chỉ mất thêm 19% thời gian tính toán so với thời gian tìm một số nguyên tố cùng kích thước bằng thuật toán *Rabin-Miller.*

Thuật toán Gordon

1. Tìm t và s là hai số nguyên tố lớn, $|t| = |s| \approx \frac{|p|}{2}$
2. Tính u là số nguyên tố nhỏ nhất theo công thức sau:

$$u = at + 1 \quad \text{với } a = 2, 4, 6, 8, \ldots$$

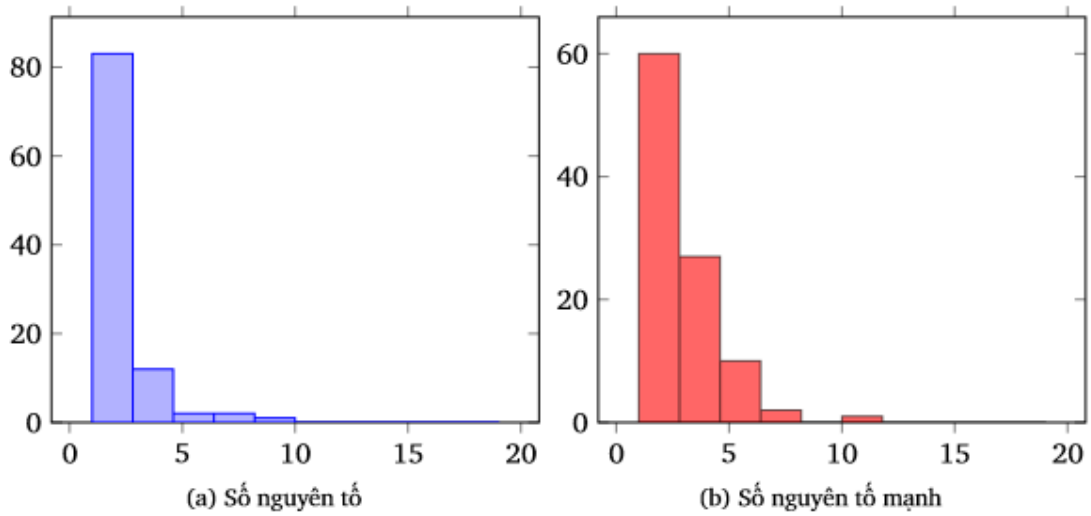
3. Tính $p_0 = (s^{u-1} - u^{s-1}) \mod us$.
4. Tính p là số nguyên tố nhỏ nhất theo công thức sau:

$$p = p_0 + aus. \quad \text{với } a = 1, 2, 3, 4, \ldots$$

Trong bước 1, 2, 4, ta sử dụng thuật toán Rabin-Miller để kiểm tra số nguyên tố.

5. Thử nghiệm

Chúng tôi đã cài đặt và chạy thử nghiệm các thuật toán mô tả trong các mục trước để sinh *100* số nguyên tố ngẫu nhiên độ dài *3072 bit.* Chương trình chạy trên máy laptop với bộ xử lý *Intel Core i7 Haswell (4500U, 1.80 GHz), Ram 4 GB, 1600 MHz,* và chạy hệ điều hành GNU/*Ubuntu 15.04.* Thời gian trung bình để sinh ra mỗi số nguyên tố khoảng *2.09* giây và khoảng *2.75* giây đối với mỗi số nguyên tố mạnh.



Hình 1: Mô tả các histogram thời gian chạy cho việc sinh ngẫu nhiên 100 số nguyên tố cỡ 3072 bit. Cột của bảng mô tả số lượng số nguyên tố chạy trong thời gian (tính theo giây) chỉ ra bởi hàng.

6. Tài liệu tham khảo

[1]. Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, Charles E. Leiserson, *Introduction to Algorithms*, McGraw-Hill Higher Education, 2001.

[2]. H. H. Khoái, P. H. Điển, *Số học thuật toán: Cơ sở lý thuyết và tính toán thực hành,* NXB Đại học Quốc Gia Hà Nội, 2003.

[3]. Jonathan Katz và Yehuda Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/CRC, 2007.

[4]. Ronald L. Rivest, Robert D. Silverman y, *Are 'Strong' Primes Needed for RSA?,* November 22, 1999.

[5]. Victor Shoup, *A computational introduction to number theory and algebra*, Cambridge University Press, 2006.

Abstract: *This paper describes the classical method to generate random strong primes used in cryptography. Specifically, the Rabin-Miller and Gordon algorithmsare presented in detail.*