Chương trình KC-01:
Nghiên cứu khoa học phát triển công nghệ thông tin và truyền thông

Đề tài KC-01-01:
Nghiên cứu một số vấn đề bảo mật và an toàn thông tin cho các mạng dùng giao thức liên mạng máy tính IP

Báo cáo kết quả nghiên cứu

ĐẢM BẢO TOÁN HỌC CHO CÁC HỆ MẬT

Quyển 3B: "Sinh tham số an toàn cho hệ mật Elgamal"

HÀ NỘI-2002

Báo cáo kết quả nghiên cứu

ĐẢM BẢO TOÁN HỌC CHO CÁC HỆ MẬT

Quyển 3B: "Sinh tham số an toàn cho hệ mật Elgamal"

Chủ trì nhóm nghiên cứu:
TS. Lều Đức Tân

MỤC LỤC

CHƯƠNG III-CHƯƠNG TRÌNH SINH SỐ NGUYÊN TỐ MẠNH CHO HỆ MẬT ELGAMAL

PHỤ LỤC 1. CÁC KẾT QUẢ THỬ NGHIỆM

PHỤ LỤC 2. VÍ DỤ VỀ SỐ CÁC SỐ PEPIN, POCKLINGTON VÀ SOPHIE

CHƯƠNG I
VAI TRÒ CỦA SỐ NGUYÊN TỐ DẠNG p=2q+1 TRONG MẬT MÃ

MỞ ĐẦU

Số nguyên tố dạng p=2q+1 với q cũng nguyên tố, tự nó trong lý thuyết số cũng là một vấn đề được nhiều nhà toán học lớn quan tâm, nhưng từ khi một số hệ mật khoá công khai ra đời thì một trong những lớp hệ mật đó có các hệ mật mà độ an toàn của nó dựa trên tích khó giải của bài toán logarit rời rạc trên trường GF(p) thì vấn đề sử dụng các số nguyên tố này càng trở nên cấp thiết. Trong chương này chúng tôi chỉ điểm lại các kết quả đã được nghiên cứu về vấn đề trên để cuối cùng khẳng định sự định hướng trong đề tài của chúng tôi là cần thiết. Sự cần thiết này không gì khác là tạo ra cho chúng ta một "máy" sinh ra được các sản phẩm tốt nhất phục vụ cho các hệ mật nói trên, đó là các số nguyên tố mạnh.

Kết cấu của chương bao gồm 2 phần chính, một là giới thiệu bài toán logarit rời rạc trên trường GF(p) cùng với các ứng dụng trong mật mã của nó và hai là các thuật toán giải bài toán logarit với mục đích như là một minh chứng cho việc khẳng định số nguyên tố dạng p=2q+1 với q cũng nguyên tố là loại tham số tốt nhất dùng cho các hệ mật nêu trên.

1.1 BÀI TOÁN LOGARIT RỜI RẠC VÀ CÁC ỨNG DỤNG TRONG MẬT MÃ

1.1.1 Bài toán logarit rời rạc trên trường GF(p)

Cho p là số nguyên tố lẻ, theo lý thuyết số ta có GF(p)={a:0≤a<p} với hai phép toán cộng và nhân các số theo modulo p là một trường, khi này GF(p)*=GF(p)\{0} là một nhóm nhân cyclic.

Giả sử ε là phần tử sinh của nhóm nhân trên (hay còn gọi là phần tử nguyên thuỷ của GF(p)) khi đó ta có $\forall a \in GF(p)^*$ luôn $\exists\ b \in GF(p)^*$ sao cho $\varepsilon^b = a \pmod p$. Giá trị b nói trên được gọi là logarit theo cơ số ε của giá trị a trên trường GF(p) và ký hiệu là $b=\log_\varepsilon a \pmod p$.

Một vấn đề đặt ra là:

Cho trước p và $a \in GF(p)^*$ hãy tìm $b=\log_\varepsilon a \pmod{p-1}$.

Vấn đề trên chính là nội dung của bài toán tìm logarit rời rạc trên trường GF(p). Trong lý thuyết thuật toán thì bài toán trên được coi là một bài toán khó theo nghĩa cho đến nay vẫn chưa tồn tại một thuật toán thời gian đa thức hoặc gần đa thức để giải nó và cũng chính vì vậy nhiều ứng dụng trong mật mã được ra đời với độ an toàn dựa vào tính khó của bài toán nói trên.

1.1.2 Hệ mật Elgamal

Ứng dụng trực tiếp là xây dựng được một hệ mật có độ an toàn tính toán đó là hệ mật khoá công khai nổi tiếng mang tên Elgamal. Hệ mật này được mô tả như sau.

Trong hệ thống liên lạc mật, mọi người dùng chung các tham số bao gồm p là số nguyên tố và ε là phần tử nguyên thuỷ của trường GF(p).

Mỗi người A trong hệ thống tự chọn một tham số mật s(A) cho riêng mình rồi tính và công khai tham số $b(A)=\varepsilon^{s(A)} \pmod p$ cho mọi người.

Một người nào đó muốn gửi cho A thông báo M (giả thiết $M \in GF(p)^*$) thì làm như sau:

Quá trình mã hoá M

Chọn ngẫu nhiên khoá $k \in Z_{p-1}$, tính và gửi cho A cặp C(M)=(x,y) như sau.

$x=\varepsilon^k \pmod p$ và

$y=Mb(A)^k \pmod p$.

Khi nhận được C(M)=(x,y) thì A tìm lại được M như sau.

Quá trình giải mã C(M)

$M=y(x^{s(A)})^{-1} \pmod p$.

Hệ mật nêu trên gọi là hệ mật Elgamal.

Do b(A) là công khai nên nếu như bài toán logarit là giải được thì có thể tính được $s(A)=\log_\varepsilon b(A) \pmod{p-1}$ và do đó hệ mật Elgamal cũng bị phá. Ngược lại cũng chưa có một kết quả nào nói rằng việc giải được mọi bản mã theo hệ Elgamal thì sẽ tìm được logarit cho nên chính xác mà nói thì độ an toàn của hệ mật này là chưa bằng tính khó của bài toán logarit song cũng chưa có một khẳng định nào nói rằng vấn đề trên thực sự là dễ hơn cho nên trên thực tế người ta vẫn coi hệ Elgamal là có độ mật tương đương với tính khó của bài toán logarit.

1.1.3 Chữ ký số Elgamal

Ứng dụng tiếp sau là thiết lập một sơ đồ chữ ký số cũng mang tên Elgamal. Sơ đồ này được giới thiệu đầu tiên trong một bài báo năm 1985 và bản cải tiến của nó được Viện Tiêu chuẩn và Công nghệ Quốc gia Mỹ chấp nhận làm chuẩn chữ ký số.

Trong hệ thống cần xác thực chủ quyền trên các văn bản thông qua chữ ký điện tử, mọi người dùng chung các tham số bao gồm p là số nguyên tố và ε là phần tử nguyên thuỷ của trường GF(p).

Mỗi người trong hệ thống A tự chọn một tham số mật s(A) cho riêng mình rồi tính và công khai tham số $b(A)=\varepsilon^{s(A)} \pmod p$ cho mọi người.

A muốn ký trên một thông báo M (giả thiết $M\in GF(p)^*$) thì làm như sau:

Quá trình ký trên M

Chọn ngẫu nhiên giá trị $k\in Z_{p-1}$, tính cặp S(M)=(x,y) như sau.

$x=\varepsilon^k \pmod p$ và

$y=(M-s(A)x)k^{-1} \pmod p$.

Cặp giá trị (x,y) trên gọi là chữ ký của A trên M và ký hiệu là $S_A(M)$.

Khi có thông báo M có kèm theo chứ ký $S_A(M)=(x,y)$ thì một người bất kỳ có thể kiểm tra tính đúng đắn rằng $S_A(M)$ có phải là là chữ ký của A trên M hay không như sau.

Quá trình kiểm tra chữ ký S(M)

Tính đúng đắn được của chữ ký thông qua tính đúng đắn của đẳng thức sau:

$\varepsilon^M=b(A)^x x^y \pmod p$.

Sơ đồ chữ ký nêu trên gọi là sơ đồ chữ ký Elgamal.

Do b(A) là công khai nên nếu như ai đó giải được bài toán logarit thì rõ ràng người đó sẽ tính được $s(A)=\log_\varepsilon b(A) \pmod{p-1}$ và do đó luôn giả mạo được chữ ký của A hay nói một cách khác là sơ đồ chữ ký đã bị phá. Ngược lại, việc giả mạo được chữ ký của một người nào đó trên một văn bản cụ thể nào đó tuy chưa có lời giải cụ thể nhưng dường như nó cũng chưa gắn được với một bài toán đã được nghiên cứu kỹ nào nên vẫn còn có khả năng thực hiện được mà không cần đến việc tính logarit. Hiện thời chưa ai tìm được cách giải xong cũng chưa ai khẳng định rằng nó có thể giải được.

1.1.4 Sơ đồ phân phối khoá Diffie-Hellman

Một trong những vấn đề cần phải thực hiện đầu tiên trong một mạng liên lạc mật đó là các bên trao đổi thông tin mật cần phải có một sự thoả thuận với nhau về khoá được dùng. Việc làm này được gọi là quá trình phân phối khoá và ứng dụng tiếp sau của bài toán logarit là thiết lập được một sơ đồ phân phối khoá tự động một cách công khai, đó là sơ đồ phân phối khoá Diffie-Hellman và được mô tả như sau.

Trong hệ thống liên lạc mật, mọi người dùng chung các tham số bao gồm p là số nguyên tố và ε là phần tử nguyên thuỷ của trường GF(p).

Hai người A và B muốn thoả thuận với nhau về một khoá sẽ được dùng trong một phiên liên lạc mật nào đó, họ làm như sau:

Trước hết, mỗi người tự chọn một tham số mật s(A) và s(B) cho riêng mình, tính rồi công bố cho nhau tham số $b(A)=\varepsilon^{s(A)} \pmod p$ và $b(B)=\varepsilon^{s(B)} \pmod p$.

Khi này cả hai A và B đều có thể tính được một tham số chung đó là $k=\varepsilon^{s(A)s(B)} \pmod p$. Cụ thể:

Đối với A thì tính $k=[b(B)]^{s(A)} \pmod p$.

Đối với B thì tính $k=[b(A)]^{s(B)} \pmod p$.

Tham số k nói trên gọi là khoá chung của A và B.

Bài toán "Cho biết p, ε, b(A) và b(B). Hãy tính k" được gọi là bài toán Diffie-Hellman. Hiển nhiên nếu giải được bài toán logarit thì ta luôn tìm được k. Điều ngược lại cho rằng nếu có thuật toán giải được bài toán Diffie-Hellman thì sẽ giải được bài toán logarit đến nay vẫn chưa có một chứng minh, tuy nhiên người ta vẫn coi là hai bài toán này là tương đương và do đó độ an toàn của việc phân phối khoá theo sơ đồ Diffie-Hellman vẫn được quy về tính khó giải của bài toán logarit.

1.2 CÁC THUẬT TOÁN TÌM LOGARIT RỜI RẠC

1.2.1 Thuật toán Shanks

Một cố gắng đầu tiên trong việc giải bài toán logarit trên trường hữu hạn là thuật toán của Danied Shanks. Ý tưởng có thể trình bày như sau :

Ký hiệu: $q=\lfloor \sqrt{p-1} \rfloor$.

Giả sử $x=\log_\varepsilon a \pmod p$ chúng ta sẽ tìm được giá trị này dưới dạng q phân $x=x_0+x_1q+...$

Trước hết ta thấy rằng do $0\le x\le p-1$ nên $x_i=0$ với mọi $i>1$ do đó :

$x=x_0+x_1q$.

Bây giờ từ đẳng thức $a=\varepsilon^x \pmod p$ ta có :

$a\varepsilon^{-x_0} = \varepsilon^{qx_1} \pmod p$.

Việc tìm b, thực chất là tìm cặp x_0 và x_1, được tiến hành bằng cách vét cạn các cặp i,j với 0≤i,j≤q-1cho đến khi tìm được i,j sao cho $a\varepsilon^{-i}=\varepsilon^{jq}$ (mod p). Khi đó rõ ràng x_0=i và x_1=j và ta được $x=\log_\varepsilon a=i+jq$.

Như vậy bằng thuật toán này có thể tìm được logarit rời rạc với thời gian tính cỡ O(q) và không gian nhớ cỡ O(q) (bỏ qua các thừa số logarit).

Kết quả 1.2. *Thời gian tính tiệm cận của thuật toán Shanks để tìm được logarit trên trường GF(p) là:*

$$L(p)=exp\{\frac{1}{2}\ln p\}. \qquad (1\text{-}1)$$ □

1.2.2 Thuật toán Pohlig - Hellman

Thuật toán thứ hai chúng tôi muốn đề cập đến là thuật toán Pohlig - Hellman. Cơ sở toán học của thuật toán Pohlig - Hellman là định lý phần dư Trung hoa sau đây.

Định lý phần dư Trung hoa. *Giả sử $m_1, m_2,...,m_r$ là các số nguyên dương nguyên tố cùng nhau từng đôi một và cho $x_1, x_2,..., x_r$ là các số nguyên.*

Khi đó từ hệ r đồng dư thức $x=x_i \pmod{m_i}$ $(i=1\div r)$ sẽ có một nghiệm duy nhất theo modulo $M= m_1.m_2...m_r$ được cho theo công thức :

$$x=\sum_{i=1}^{\Gamma} a_i M_i y_i \pmod M$$

Trong đó $M_i=M/m_i$ và $y_i= M_i^{-1} \pmod{m_i}$ với $(i=1\div r)$. □

Từ định lý trên, nếu $p-1 = \prod_{i=1}^{r} q_i^{\alpha_i}$ thì rõ ràng để tính $x=\log_\varepsilon a \pmod{p-1}$ chúng ta có thể thông qua việc tính r giá trị $x_i=\log_\varepsilon a \pmod{m_i}$ với $m_i=q_i^{\alpha_i}$ (i=1÷r). Chi tiết của thuật toán có thể xem trong [Stinson], một điều đáng phân tích ở đây là nếu p-1 chỉ toàn những ước nguyên tố nhỏ thì việc tìm $x=\log_\varepsilon a \pmod p$ rất là dễ dàng và như vậy điều kiện cần thiết đối với tham số

p là nó phải không có tính chất trên. Đến đây ta có thể thu được kết luận sau về thời gian tính của thuật toán Pohlig - Hellman.

Kết quả 1.3. *Thời gian tính tiệm cận của thuật toán Pohlig - Hellman để tìm được logarit trên trường GF(p) là:*

$L(p)=exp\{\ln q\}$ *với q là ước lớn nhất của p-1.* *(1-2)* □

Với kết quả trên của thuật toán Pohlig-Hellman chúng ta thấy rằng tính khó của việc giải bài toán logarit rời rạc trên GF(p) có thể quy về tính khó của việc tìm giá trị này theo modulo q với q là ước lớn nhất của p-1 (tức là tìm $x_q=x \pmod q$), chính vì lý do này mà từ nay về sau khi trình bày các thuật toán khác chúng tôi chỉ tập trung vào việc tìm giá trị x_q nói trên mà thôi.

1.2.3 Thuật toán sàng bậc q

Để tìm x_q với $x=\log_\varepsilon a \pmod p$ và q là ước của p-1, thuật toán sàng bậc q dựa vào cơ sở sau.

Kết quả 1.4. *Nếu tìm được cặp s,t sao cho gcd(t,q)=1 và $\varepsilon^s a^t$ là một thặng dư bậc q trong GF(p) tức là $\exists w \in GF(p)^*$ sao cho $\varepsilon^s a^t=w^q \pmod p$ thì $x_q=-st^{-1} \pmod q$.*

Chứng minh.

Từ định nghĩa $x=\log_\varepsilon a \pmod p$ ta có $a=\varepsilon^x \pmod p$ *(1-3).*

Từ giả thiết $\varepsilon^s a^t=w^q \pmod p$, thay vào *(1.3)* ta được

$\varepsilon^s(\varepsilon^x)^t= w^q \pmod p$. *(1-4).*

Do ε là phần tử nguyên thuỷ của GF(p) nên luôn tồn tại r sao cho $w=\varepsilon^r \pmod p$ và như vậy từ *(1.4)* ta có.

$\varepsilon^s(\varepsilon^x)^t=(\varepsilon^r)^q \pmod p$, suy ra

$s+xt=rq \pmod{p-1}$ hay

$s+xt=0 \pmod q$ *(1-5).*

Từ giả thiết gcd(t,q)=1 nên tồn tại t^{-1} (mod q) và do đó từ *(1-5)* ta có ngay $x=-st^{-1}$ (mod q) và đây là điều cần chứng minh. □

Kỹ thuật để tìm cặp s,t nêu trong kết quả 1.4 được thực hiện như sau.

Chọn B là một số nguyên nào đó gọi là ngưỡng của cơ sở phân tích, giả sử m là số các số nguyên tố không quá B, sau đó tiến hành các bước sau:

Bước 1.Tìm m+1 cặp số s_i,t_i (i=1÷m+1) thoả mãn điều kiện:

$$\varepsilon^{s_i} a^{t_i} \text{ (mod p)}=v_i^q \prod_{j=1}^{m} p_{ji}^{\alpha_{i,j}} \text{ (với } 0\le\alpha_{i,j}<q) \qquad (1\text{-}6).$$

Ký hiệu véc tơ $\beta_i=(\alpha_{i,1}, \alpha_{i,2},..., \alpha_{i,m})$ với i=1÷m+1, rõ ràng hệ m+1 véc tơ trong không gian m chiều nên phải phụ thuộc tuyến tính tức là tồn tại bộ m+1 số $(k_1,k_2,...,k_{m+1})$ không đồng thời bằng 0 với $0\le k_i<q$ sao cho.

$$k_1\beta_1+ k_2\beta_2+...+ k_{m+1}\beta_{m+1}=\theta=(0,0,...,0). \qquad (1\text{-}7).$$

Bước 2. Tìm bộ $(k_1,k_2,...,k_{m+1})$ nói trên.

Lấy $s= k_1s_1+ k_2s_2+...+ k_{m+1}s_{m+1}$ và $t= k_1t_1+ k_2t_2+...+ k_{m+1}t_{m+1}$, dễ dàng kiểm tra được s,t thoả mãn điều kiện $\varepsilon^s a^t=w^q$ (mod p).

Chú ý rằng, bước 1 được thực hiện theo cách Lấy-Kiểm tra cho đến khi tìm được đầy đủ số cặp theo yêu cầu, còn việc làm của bước 2 chính là giải một hệ phương trình đại số tuyến tính hệ số trên GF(q) mà hệ này luôn có nghiệm. Tóm lại ta luôn tìm được cặp s,t theo mong muốn, tuy nhiên để có thể đưa ra một dẫn giải tường minh về thời gian tính của thuật toán này là một điều không đơn giản. Chúng ta bằng lòng với kết quả đã được công bố về thời gian tính của phương pháp sàng bậc q như sau (xem [Stinson]).

Kết quả 1.5. *Thời gian tính tiệm cận của thuật toán sàng bậc q để tìm được logarit trên trường GF(p) là*

$$L(p)=\exp\{(1+O(1))\ln^{\frac{1}{2}} q.\ln\ln^{\frac{1}{2}} q\} \qquad (1\text{-}8).$$

ở trên q là ước nguyên tố lớn nhất của p-1, còn O(1) là một vô cùng bé khi $q\to\infty$. □

1.2.4 Thuật toán sàng trường số

Giống như ý tưởng của thuật thoán sàng bậc q, phương pháp sàng trường số cũng thực hiện theo kiểu tìm cặp s,t sao cho $\varepsilon^s a^t = w^q \pmod p$, sự khác biệt cơ bản là thay vì việc tìm các cặp s,t trên trực tiếp trên GF(p) của sàng bậc q thì sàng trường số lại đi tìm chúng trong trường mở rộng K nào đó. Tính hiệu quả của thuật toán sàng trường số là ở chỗ có thể khéo léo lựa chọn được trường K thích hợp để việc tìm cặp s,t được dễ dàng hơn. Để có thể trình bày cặn kẽ các bước thực hiện của phương pháp này chúng ta cần phải có một loạt kiến thức bổ trợ về đại số cao cấp (xem chi tiết trong [P. M. Hoà]), mục đích của đề tài này không phải là lặp lại một việc làm như vậy mà ở đây chúng tôi chỉ muốn dẫn ra kết quả cuối cùng về thời gian tính của thuật toán đó là.

Kết quả 1.6. *Thời gian tính tiệm cận của thuật toán sàng trường số để tìm được logarit trên trường GF(p) là*

$$L(p)=\exp\{(C+O(1))\ln^{\frac{1}{3}} q.\ln\ln^{\frac{2}{3}} q\} \qquad (1\text{-}9).$$

ở trên q là ước nguyên tố lớn nhất của p-1, $C\approx 1.9229$ còn O(1) là một vô cùng bé khi $q\to\infty$. □

Kết luận

Để các hệ mật mà độ mật dựa trên cơ sở tính khó giải của bài toán logarit trên trường GF(p) có độ an toàn cao thì:

1.Độ dài nhị phân của số nguyên tố p phải lớn. Theo các đánh giá thì $\log p>500$.

2. p-1 phải có ước nguyên tố lớn, tốt nhất là các số nguyên tố mạnh.

Với các kết luận trên rõ ràng việc sinh các số nguyên tố mạnh để sử dụng trong Ngành là một điều tất yếu và vô cùng cần thiết trong giai đoạn này.

TÀI LIỆU DẪN

[P. M. Hoà] Phạm Thị Minh Hoà, **Nghiên cứu phương pháp sàng trường số, tính logarit rời rạc trên trường hữu hạn.** *Đề tài cấp cơ sở, Học viện KTMM, Hà nội 2000.*

[Stinson] Douglas Robert Stinson, **Mật mã Lý thuyết và Thực hành.** *Bản dịch tiếng Việt Hà nội 1995.*

CHƯƠNG II
SINH SỐ NGUYÊN TỐ LỚN BẰNG PHƯƠNG PHÁP TĂNG DẦN ĐỘ DÀI

MỞ ĐẦU

Một thuật toán sinh các số nguyên tố thông thường được coi là một hệ quả của một thuật toán kiểm tra tính nguyên tố nào đó theo phương thức "Chọn ngẫu nhiên số tự nhiên x độ dài n, sau đó lấy và kiểm tra các số trong dãy x+k (với k=0,1,2,...) cho đến khi được số nguyên tố". Như vậy tự nhiên mà nói thì thuật toán sinh bao giờ cũng "lâu" hơn thuật toán kiểm tra mà nó dựa vào. Cho đến bây giờ, chưa tồn tại một thuật toán kiểm tra tất định tính nguyên tố trong thời gian đa thức do vậy mọi thuật toán sinh theo cách cổ truyền trên không thể thực hiện được trong thời gian đa thức. Đối với thuật toán xác suất thì với phương pháp kiểm tra tính xác suất của Rabin-Miller hay của Salovay-Strassen chúng ta có ngay được một thuật toán sinh với thời gian tính cỡ $O(n^6)$ và trong trường hợp giả thuyết Riemann mở rộng là đúng đắn thì nó cũng là một thuật toán tất định.

Trong chương này chúng tôi đưa ra một phương thức mới để xây dựng thuật toán sinh và với phương thức này chúng tôi thu được một kết quả khá thú vị đó là thuật toán xác suất được thực hiện trong thời gian $O(n^8)$. Điểm khác biệt cơ bản giữa thuật toán mà chung tôi đưa ra với thuật toán xác suất của Rabin-Miller hay của Salovay-Strassen là ngay cả trong trường hợp giả thuyết Riemann mở rộng chưa được chứng minh thì các số thu được tại đầu ra của thuật toán này luôn là nguyên tố trong khi đó của thuật toán sau là chưa chắc. Kết quả thu được của chúng tôi chỉ là một đóng góp khiêm tốn trong lĩnh vực lý thuyết số và thuật toán bởi vì nó mới chỉ là một ví dụ chứng tỏ sự "Không phải là hệ quả của thuật toán sinh đối với thuật toán kiểm tra" mà vốn đã là như vậy thì tính đa thức của thuật toán sinh cũng chưa chắc đã đóng góp được gì cho khả năng tạo được thuật toán kiểm tra mà theo chúng tôi thì sự

thiết kế được thuật toán kiểm tra nhanh mới là đóng góp lớn. Một đặc điểm trong việc xây dựng thuật toán sinh của chúng tôi là các công cụ được sử dụng rất đơn giản thậm chí là rất "cũ kỹ" không đòi hỏi một bổ trợ cấp cao nào cho nên việc lập trình thực hiện nó có thể phổ cập đến mọi đối tượng. Đơn giản nhưng hiệu quả có lẽ là đóng góp cao nhất của chúng tôi trong công bố thuật toán ở chương này.

Kết quả đạt được chính trong chương của chúng tôi có thể nêu như sau:

Thứ nhất. Từ những phân tích về sai lầm loại 1 của thuật toán kiểm tra tính nguyên tố các số trong lớp $\mathbf{L}^F$ chúng ta có được thời gian thực hiện của thuật toán Pock_test_F dùng để kiểm tra tính nguyên tố của một số tự nhiên độ dài n là $T_{\text{Pock-test}}(n)\le C_\alpha n^4 \ln n$ với C_α là một hằng số tính được theo xác suất sai lầm loại 1 của thuật toán là α.

Thứ hai. Từ định lý 2.6 về sự tồn tại số nguyên tố trong đoạn $[yF+1;(y+\Delta)F+1]$ với $\Delta\le \ln F(\ln\ln F+6)$ chúng ta có được định lý 2.7 về thời gian tối đa của thuật toán sinh $\textbf{\textit{POCK-GEN}}_F$ ký hiệu là $T_{\text{POCK-GEN}}(n)\le C_0 n^6$.

Cuối cùng. Bằng việc chứng minh được thời gian sinh một số nguyên tố độ dài n bằng thuật toán sinh các số nguyên tố độ dài <n (định lý 2.11) chúng ta có được kết luận quan trọng nhất của chương đó là thời gian tính của thuật toán sinh số của chúng ta xây dựng là $O(n^7)$.

2.1 MỘT SỐ KẾT QUẢ TRONG LÝ THUYẾT SỐ

Một số kết quả trong lý thuyết số được trích dẫn dưới đây (xem [Ribenboim], [L. Đ. Tân]...) sẽ được sử dụng để xây dựng thuật toán sinh số nguyên tố và quan trọng hơn cả là chứng minh tính đa thức của thuật toán sinh này.

Định lý Pocklington. *Cho $x=RF+1$, trong đó $\gcd(R,F)=1$. Khi đó nếu mỗi ước nguyên tố q của F tồn tại giá trị a sao cho:*

(a). $a^{x-1}=1 \pmod x$. *(2-1)*

(b). $(a^{(x-1)/q}-1,x)=1$. *(2-2)*

Thì mọi ước nguyên tố p của x đều có dạng p=tF+1.□

***Khái niệm thặng dư bậc q**. Ta nói a là thặng dư bậc q modulo x nếu tồn tại b sao cho $a=b^q \pmod x$.□*

***Định lý về thặng dư bậc q**. Cho p là số nguyên tố lẻ sao cho q là ước của p-1. Khi đó:*

(a). Điều kiện cần và đủ để giá trị $m \in GF(p)^*$ là thặng dư bậc q là

$$m^{(p-1)/q}=1 \pmod p \qquad (2\text{-}3).$$

(b). Số các thặng dư bậc q trong $GF(p)^*$ đúng bằng (p-1)/q. (2-4).□

Một vài điều kiện đủ về tính nguyên tố.

***Một điều kiện đủ về tính nguyên tố.** Cho x=RF+1 thoả mãn điều kiện của định lý Pocklington. Khi đó*

(a). Nếu $R \le F$ thì x là số nguyên tố.

(b). Nếu $F<R \le F^2$ và B^2-4A là số không chính phương thì x là số nguyên tố. Trong (b) thì A=R (div F) và B=R (mod F).□

Định lý Dirichlet

Số các số nguyên tố có dạng Ak+B với gcd(A,B)=1 không vượt quá x ký hiệu là $\pi_{A,B}(x)$ là vô cùng lớn tương đương với $\frac{1}{\varphi(A)}\frac{x}{\ln x}$ khi $x \to \infty$ tức là

$$\pi_{A,B}(x) \sim \frac{1}{\varphi(A)}\frac{x}{\ln x} \qquad (2\text{-}5).$$

ở đây $\varphi(A)$ là số các số không quá A và nguyên tố với A.□

Chú thích.

Định lý đầu tiên do Dirichlet đưa ra và chứng minh vào năm 1837 mới dừng ở kết luận là có vô số số nguyên tố dạng Ak+B, sau này Valée Poussin

bổ xung thêm công thức về mật độ. Ngoài ra nhiều tác giả đã chỉ ra sự không như nhau của các giá trị $\pi_{A,B}(x)$ với cùng một giá trị A còn $1 \le B < A$, chẳng hạn vào năm 1853 Tschebycheff chỉ ra $\pi_{3,1}(x) < \pi_{3,2}(x)$ còn $\pi_{4,1}(x) < \pi_{4,3}(x)$ với một số giá trị x nhỏ; vào năm 1957 Leech đã tính được với số x=26861 là số nguyên tố nhỏ nhất để $\pi_{4,1}(x) > \pi_{4,3}(x)$ và tương tự Bays & Hudson (1978) tìm được x=608981813029 là số nguyên tố nhỏ nhất để $\pi_{3,1}(x) > \pi_{3,2}(x)$ việc chỉ ra này Hudson & Brauer đã phải bỏ ra vài năm để nghiên cứu (xem [Ribenboim] trang 148-150).

2.2 THUẬT TOÁN POCKLINGTON

2.2.1 Thuật toán kiểm tra tính nguyên tố Pocklington trên lớp L^F

Với cơ sở là các kết quả đã nêu trong mục 0, chúng ta có thể xây dựng được thuật toán xác suất định hướng chấp nhận để kiểm tra tính nguyên tố của các số nguyên thuộc lớp L^F như sau.

Giả sử $F=\prod_{i=1}^{r} p^{\alpha_i}$, với mỗi i=1÷r ta lấy số tự nhiên M_i gọi là các tham số của thuật toán. Các tham số này sẽ được phân tích sau.

Thuật toán 2.1. Thuật toán Pocklington.ký hiệu là ***Pock-test$_F$.***

Đầu vào $x \in L^F$.

Bước 1. Lấy i=1;

Bước 2. $p=p_i$; $M=M_i$; $m=1$;

Bước 3. Lấy a=random(x).

Bước 4. Kiểm tra đồng dư thức $a^{N-1} \equiv 1 \pmod x$.

Nếu đúng, sang bước 5.

Ngược lại, ***Pock-test$_F$****(x)=0, thuật toán dừng.* (*1).

Bước 5. Kiểm tra điều kiện $a^{(x-1)/p} \equiv 1 \pmod x$

Nếu đúng, sang bước 6.

Ngược lại, sang bước 7.

Bước 6. Kiểm tra điều kiện m<M.

Nếu đúng, m=m+1, quay về bước 3.

Ngược lại, ***Pock-test***$_F$*(x)=0, thuật toán dừng.* (*2).

Bước 7. Kiểm tra điều kiện gcd($a^{(x-1)/p}$-1,x)=1.

Nếu đúng, sang bước 8.

Ngược lại, ***Pock-test***$_F$*(x)=0, thuật toán dừng.* (*3).

Bước 8. Kiểm tra điều kiện i<r.

Nếu đúng, i=i+1, quay về bước 2.

Ngược lại, sang bước 9.

Bước 9. Kiểm tra điều kiện R≤F.

Nếu đúng, ***Pock-test***$_F$*(x)=1, thuật toán dừng.*

Ngược lại, sang bước 10.

Bước 10. Kiểm tra điều kiện $(R \bmod F)^2-4(R \operatorname{div} F)=Q^2$.

Nếu đúng, ***Pock-test***$_F$*(x)=0, thuật toán dừng.* (*4).

Ngược lại, ***Pock-test***$_F$*(x)=1, thuật toán dừng.*□

2.2.2 Đánh giá xác suất sai lầm của thuật toán *Pock-test*$_F$.

Theo thuật toán trình bày ở phần trước thì ***Pock-test***$_F$(x)=0 xảy ra tại 1 trong 4 trường hợp sau.

(*1). $a^{x-1}\neq 1 \pmod x$. (bước 4)

(*2). $a^{(x-1)/p}\equiv 1 \pmod x$ trong cả M lần lấy ngẫu nhiên a. (bước 6)

(*3). $a^{(x-1)/p}\neq 1 \pmod x$ và $\gcd(a^{(x-1)/p}-1,x)>1$. (bước 7)

(*4). $(R \bmod F)^2-4(R \operatorname{div} F)=Q^2$. (bước 10)

Hiển nhiên các trường hợp (*1), (*3) và (*4) kết luận là đúng, vậy kết luận sai chỉ có thể xảy ở điều kiện (*2). Điều xảy ra (*2) tương đương với sự kiện trong cả M lần chọn ngẫu nhiên a chúng đều thoả mãn điều kiện $a^{(x-1)/p}\equiv 1 \pmod x$. Theo định lý về thặng dư bậc p, thì a là p-thặng dư modulo x và xác suất lấy được một p-thặng dư trong một lần chọn ngẫu nhiên chỉ là $\frac{1}{p}$, do đó

sự kiện trong M lần đều lấy được p-thặng dư chỉ xảy ra với xác suất Prob=$\frac{1}{p^M}$. Tóm lại chúng ta đã chứng minh được kết quả sau.

Bổ đề 2.2. *Xác suất sai lầm loại 1 của thuật toán* ***Pock-test$_F$*** *trên lớp* $\mathbf{L}^F$ *với* $F=p_1^{\alpha_1}...p_r^{\alpha_r}$ *theo bộ tham số* $M_1,...,M_r$ *là* $P_{error1}\le\frac{1}{p_1^{M_1}}+...+\frac{1}{p_r^{M_r}}$ *(2-6).*□

Bổ đề 2.3*. Cho trước giá trị* $\alpha>0$*, luôn tồn tại hằng số C tính được theo* α *và xây dựng được thuật toán* ***Pock-test$_F$*** *với bộ tham số* $M_1,...,M_r$ *sao cho có xác suất sai lầm loại 1 không vượt quá* α *và* $\sum_{i=1}^{r}M_i\le n(\ln n+C)$ *(2-7).*

Chứng minh.

Để có được xác suất sai lầm của thuật toán Pocklington không vượt quá một giá trị $\alpha>0$ cho trước, theo bổ đề 2.2, một cách đơn giản chúng ta chỉ cần chọn bộ tham số M_i thoả mãn điều kiện $M_i\ge\lfloor\log_{p_i}\frac{r}{\alpha}\rfloor$.

Do $r\le Logx=n$ và $\log_{p_i}\frac{1}{\alpha}\le Log\frac{1}{\alpha}$ cho nên nếu ta lấy $M_i\ge LogLogN+Log\frac{1}{\alpha}$ thì rõ ràng điều kiện $M_i\ge\lfloor\log_{p_i}\frac{r}{\alpha}\rfloor$ được thoả mãn. Với cách lấy trên ta có $\sum_{i=1}^{r}M_i\le r(LogLogN+Log\frac{1}{\alpha})\le n(\ln n+Log\frac{1}{\alpha})$.

Lấy $C=Log\frac{1}{\alpha}$ chúng ta có ngay điều cần chứng minh. □

Từ nay về sau, không giảm tổng quát, ta luôn coi α là một giá trị cố định cho trước và do đó C luôn là một hằng số và để tiện lợi trong trình bày chúng ta dùng ký hiệu ***Pock-test$_F$*** để chỉ thuật toán kiểm tra tính nguyên tố các số tự nhiên trong lớp $\mathbf{L}^F$ với mặc định là bộ tham số M_i được lấy như trong bổ đề 2.3 và như vậy một kết quả tự nhiên mà chúng ta có thể thu được ở đây là.

Định lý 2.4. *Thời gian thực hiện việc kiểm tra tính nguyên tố của số tự nhiên x độ dài n bit trong lớp* $\mathbf{L}^F$ *ký hiệu là* $T_{Pock\text{-}test}(n) \le C_\alpha n^4 \ln n$. *(2-8)*□

2.2.3 Thuật toán sinh số nguyên tố trên lớp $\mathbf{L}^F$

2.2.3.1 Mở đầu

Như phần trước chúng ta đã xây dựng được một thuật toán kiểm tra nhanh tính nguyên tố của các số trên lớp $\mathbf{L}^F$, đó là thuật toán ***Pock-test***$_F$. Tại phần này chúng ta tiến hành việc sinh các số nguyên tố trong lớp $\mathbf{L}^F$ dựa vào thuật toán kiểm tra pocklington đã nêu. Từ đặc thù của lớp $\mathbf{L}^F$ là chưa chắc với mọi n là độ dài của các số thuộc lớp này đã tồn tại số nguyên tố có độ dài tương ứng trong lớp đó do vậy việc sinh các số nguyên tố có độ dài cho trước là không luôn luôn được do vậy thuật toán sinh của chúng ta xây dựng ở đây chỉ cần đạt được chỉ tiêu sau:

Nếu đầu vào là độ dài số nguyên tố cần sinh n thì đầu ra phải là một số nguyên tố có độ dài không nhỏ hơn n.□

Thuật toán sinh số nguyên tố trên $\mathbf{L}^F$ ký hiệu là ***POCK-GEN***$_F$ được thực hiện như sau.

Thuật toán 2.5

Đầu vào n (length(F)<n<2length(F)) là độ dài tối thiểu của số nguyên tố cần sinh.

Bước 1. Xác định R_{0n} *là số nhỏ nhất và* R_{1n} *là số lớn nhất để RF+1 có độ dài n*

Bước 2. Lấy ngẫu nhiên số y=random$[R_{0n};R_{1n}]$*;tính x=yF+1.*

Bước 3. Xét ***Pock-test***$_F(x)=1$.

Nếu đúng. Đầu ra của thuật toán là x. Thuật toán dừng.

Ngược lại. Chuyển sang 4.

Bước 4. y=y+1; x=yF+1; Chuyển về 3.

*Khi này ta ký hiệu x=****POCK-GEN***$_F(n)$.□

2.2.3.2 Một số phân tích về khả năng tồn tại số nguyên tố độ dài n trong lớp số L^F

Định lý 2.6. *Ký hiệu m=*ln*F thì với m đủ lớn ta có với mọi y≥1 thì trongΔ số nguyên liên tiếp của dãy aF+1 bắt đầu từ yF+1 luôn tồn tại ít nhất một số nguyên tố. với Δ= m(*ln*m+6)* *(2-9)*

Chứng minh.

Xét giá trị x=yF+1 và x'=(y+Δ)F+1 với $1\le y<y+\Delta\le F^2$ *(2-10)*, để đảm bảo x và x' thuộc L^F. Theo định lý Dirichlet ta có số các số nguyên tố có dạng aF+1 nằm trong khoảng [x;x'] là

$$\begin{aligned}\pi &= \pi_F(x')-\pi_F(x)\\ &\sim \frac{F}{\varphi(F)}\left(\frac{y+\Delta}{\ln((y+\Delta)F)}-\frac{yF}{\ln(yF)}\right)\\ &> \left(\frac{y+\Delta}{\ln((y+\Delta)F)}-\frac{yF}{\ln(yF)}\right)\\ &= \frac{\Delta\ln(yF)-y[\ln((y+\Delta)F)-\ln(yF)]}{\ln(yF)\ln((y+\Delta)F)}\\ &= \frac{\Delta\ln(yF)-y\ln(1+\frac{\Delta}{y})}{\ln(yF)\ln((y+\Delta)F)} \qquad (2\text{-}11).\end{aligned}$$

Nếu lấy y=y(m) và Δ=Δ(m) sao cho $\frac{\Delta(m)}{y(m)}$ là vô cùng bé khi $m\to\infty$ *(2-12)*

ta có $\ln(1+\frac{\Delta}{y})$ tương đương với $\frac{\Delta}{y}$. Thay vào *(2.11)* ta được

π tương đương với $\frac{\Delta\ln(yF)-y\frac{\Delta}{y}}{\ln(yF)\ln((y+\Delta)F)}=\frac{\Delta(\ln(yF)-1)}{\ln(yF)\ln((y+\Delta)F)}$ *(2-13)*

Từ điều kiện *(2.10)* là $y+\Delta\le F^2$ nên $\ln((y+\Delta)F)\le 3m$ *(2-14)*

thêm vào nữa ta có $\lim_{m\to\infty}\frac{\ln(yF)-1}{\ln(yF)}=1$ *(2-15).*

Thay *(2-14)* và *(2-15)* vào vế phải của *(2-13)* thì từ *(2-11)* ta có π tương đương với một đại lượng $>\frac{\Delta}{3m}$. Bây giờ chỉ cần lấy Δ(m)=6m còn y(m)≥*m*ln*m*, hiển nhiên điều kiện *(2.12)* được thỏa mãn và do đó π tương đương với một đại lượng >2 khi m→∞.

Như vậy với m đủ lớn thì π>1, tức là trong khoảng [x;x'] nếu y≥y_0=*m*ln*m* luôn tồn tại số nguyên tố dạng aF+1, nếu y<*m*ln*m*. thì do khoảng [x_0;x_0'] với x_0=y_0F+1 có ít nhất một số nguyên tố nên trong khoảng [x;x_0'] cũng tồn tại số nguyên tố. Rõ ràng chúng ta đã chứng minh được rằng với mọi x=yF+1∈$\mathbf{L}^F$ luôn tồn tại số nguyên tố dạng aF+1 với a-y≤*m*(ln*m*+6) và đây là điều cần chứng minh.□

Từ định lý trên chúng ta thu được định lý quan trọng sau.

Định lý 2.7. *Với m*=ln*F đủ lớn thì:*

(1). Thuật toán sinh số nguyên tố ***POCK-GEN***$_F$ *trên lớp* $\mathbf{L}^F$ *luôn sinh được số nguyên tố độ dài n bit trong thời gian ký hiệu là* $T_{POCK\text{-}GEN}(n)\leq C_0 n^6$ *(2-16).*

(2). Thêm nữa, nếu đầu vào của thuật toán là n thì số nguyên tố sinh được tại đầu ra có độ dài là l không quá $n+\frac{m}{3}$ *(2-17).*

Chứng minh.

Ta biết, theo công thức *(2-8)* (định lý 2.4) thì để kiểm tra tính nguyên tố của số tự nhiên độ dài n bit bằng thuật toán ***Pock-test*** là $T_{Test}(n)\leq C_\alpha n^4 \ln n$. Lại từ công thức *(2-9)* (định lý 2.6) thì số lần lấy và kiểm tra trong thuật toán ***POCK-GEN*** là không quá Δ=*m*(ln*m*+6)≤*n*(ln*n*+6) như vậy ta có ngay thời gian thực hiện thuật toán này là

$T_{POCK\text{-}GEN}(n)\leq C_\alpha n^4 \ln n\ n(\ln n+6)$ *(2-18).*

Do $\ln^2 n$ là vô cùng lớn bậc thấp hơn n nên với n đủ lớn, tồn tại hằng số C_0 sao cho $C_\alpha \ln^2 n\leq C_0 n$ *(2-19).*

Thay *(2-18)* vào *(2-19)* ta có ngay $T_{POCK\text{-}GEN}(n)\leq C_0 n^6$ và công thức *(2-16)* của định lý đã được chứng minh.

Giả sử y là giá trị đầu tiên được chọn trong thuật toán với đầu vào là n thì rõ ràng độ dài của y là k≈n-m (do số được thử đầu tiên là x=yF+1 có độ dài n) như vậy số nguyên tố tìm được trong thuật toán giả sử là p=y'F+1 thì theo công thức *(2-9)* (định lý 2.6) ta có y'≤y+Δ=y+m(ln$m$+6). Rõ ràng $\frac{y'}{y} \le \frac{y+m(\ln m+6)}{y} < m(\ln m+6)+1$ nên độ dài của p là

l≤n+log(m(ln$m$+6)+1) *(2-20).*

Trong công thức *(2-20)*, với m đủ lớn ta sẽ có log(m(ln$m$+6)+1)≤$\frac{m}{3}$ và công thức *(2-17)* đã được chứng minh.□

2.3 THUẬT TOÁN SINH CÁC SỐ NGUYÊN TỐ ≥N BIT TỪ THUẬT TOÁN SINH CÁC SỐ NGUYÊN TỐ <N BIT

2.3.1 Mở đầu

Trong mục này chúng tôi giải quyết vấn đề sau:

Biết thuật toán sinh toàn bộ các số nguyên tố độ dài không đến n. Hãy xây dựng thuật toán sinh các số nguyên tố độ dài không dưới n sao cho có thể sinh toàn bộ các số nguyên tố độ dài n.

Ý tưởng chủ đạo để giải quyết vấn đề trên của chúng tôi là từ khả năng có thể sinh được toàn bộ các số nguyên tố độ dài không đến n của thuật toán đã có chúng tôi sinh ngẫu nhiên các số F thoả mãn hai điều kiện sau:

(F1). n>length(F)≥$\frac{n}{3}$.

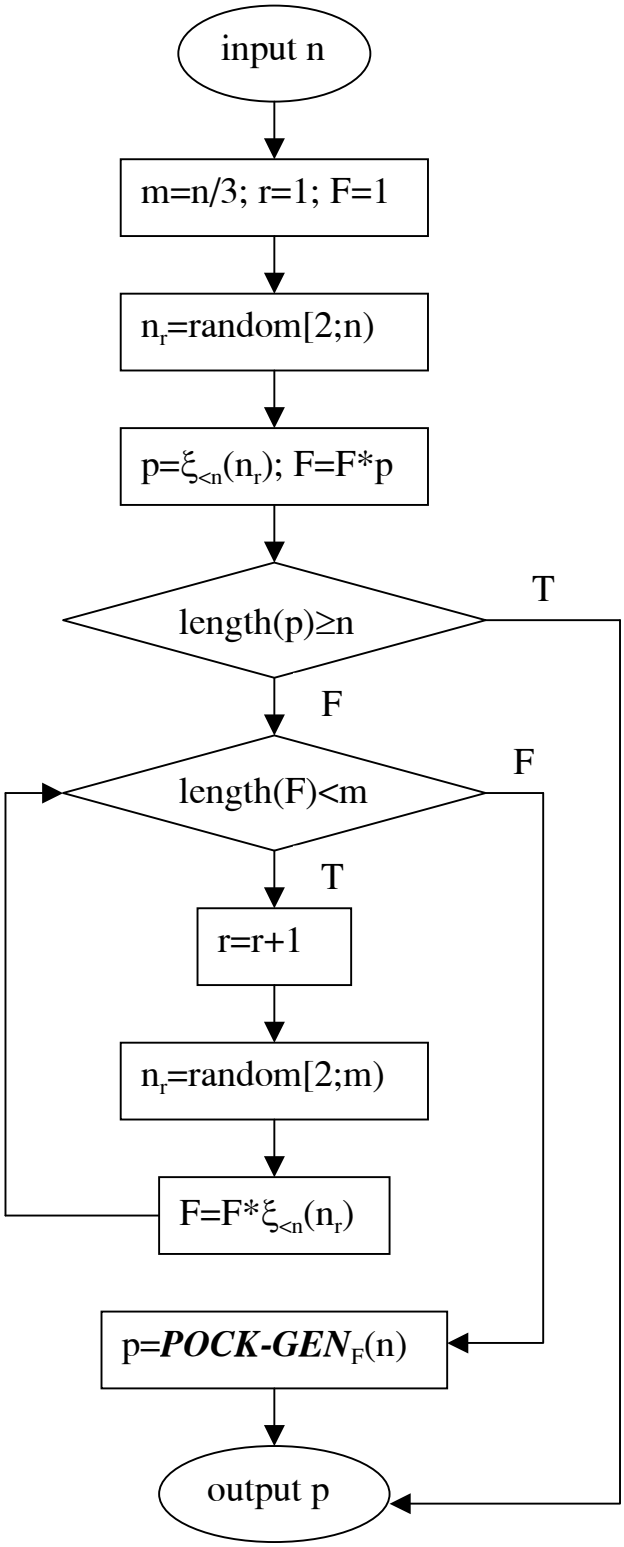
(F2). Biết được phân tích của F ra thừa số nguyên tố.

Tiếp đến sử dụng thuật toán sinh Pocklington để sinh các số nguyên tố độ dài không dưới n trong lớp $\mathbf{L}^F$.

Việc giải quyết vấn đề được thể hiện qua sơ đồ ở trang sau:

2.3.2 Thuật toán

Sơ đồ thuật toán 2.8.


input n
m=n/3; r=1; F=1
n_r=random[2;n)
p=$\xi_{<n}(n_r)$; F=F*p
length(p)≥n
T
F
length(F)<m
F
T
r=r+1
n_r=random[2;m)
F=F*$\xi_{<n}(n_r)$
p=***POCK-GEN***$_F$(n)
output p

2.3.3 Phân tích khả năng sinh các số nguyên tố dộ dài n của thuật toán

Chúng ta biết rằng nếu p là một số nguyên tố có độ dài n bit, không giảm tổng quát ta giả sử n>2 do đó nó là số lẻ nên có dạng p=2x+1 trong đó x là số có độ dài <n, do đó mọi ước nguyên tố của x đều có độ dài không quá n-1 bit. Nói một cách khác là x sẽ là bội của F nào đó có thể được tạo từ thuật toán và do đó p sẽ thuộc lớp L^F hay p có thể được sinh từ thuật toán. Tóm lại chúng ta đã thu được kết quả sau.

Định lý 2.9. *Mọi số nguyên tố độ dài n bit đều có thể được sinh từ thuật toán* ξ_n *xây dựng ở trên.*□

Chú ý: Thuật toán ξ_n có một số đặc điểm sau.

Thứ nhất: Đầu ra của thuật toán chỉ là những số nguyên tố thoả mãn điều kiện có độ dài không dưới n bit.

Thứ hai: Hợp toàn bộ các lớp L^F thu được bởi toàn bộ các số F có thể xây dựng được trong thuật toán không phủ toàn bộ các số tự nhiên có độ dài n bit đó là các số có dạng x=2q với q cũng là nguyên tố. Tuy nhiên may mắn là các số này đều là hợp số do đó chúng ta không cần quan tâm đến.

Thứ ba: Việc đánh giá khả năng sinh được các số nguyên tố độ dài n của thuật toán là một điều cực kỳ khó khăn, nó đòi hỏi việc phải đánh giá được khả năng xây dựng các số F khác nhau và thêm nữa phải đánh giá được số các lớp L^F khác nhau cùng chứa một số nguyên tố p độ dài n bit, tuy nhiên chúng ta có thể hình dung được một điều là xác suất sinh được các số nguyên tố khác nhau cùng độ dài n là không như nhau.

2.3.4 Phân tích thời gian thực hiện việc sinh một số nguyên tố độ dài n

Theo sơ đồ thực hiện thuật toán thì để sinh một số nguyên tố độ dài không dưới n bit ta phải thực hiện việc tạo F và sau đó là sinh số nguyên tố p theo thuật toán ***POCK-GEN***$_F$.

Thứ nhất. Hiển nhiên nếu số nguyên tố p thu được tại đầu ra của thuật toán có độ dài là n thì riêng công đoạn thực hiện thuật toán ***POCK-GEN***$_F$, theo công thức *(2-16)* (định lý 2.7), chúng ta cần đến một thời gian tính là C_0n^6.

Tiếp đến. Để thực hiện việc xây dựng F trong thuật toán chúng ta cần sử dụng thuật toán sinh để sinh các ước nguyên tố của F. Theo như thuật toán đã chỉ ra thì số lượng các ước nguyên tố để tạo nên F chính là số r thu được khi thực hiện xong công đoạn này.

Rõ ràng nếu r=1 thì thời gian để thực hiện bước này chính là thời gian để thực hiện thuật toán sinh $\xi_{<n}$ với đầu vào n_1.

Ngược lại, chúng ta cần tiến hành sử dụng r lần thuật toán sinh $\xi_{<n}$ với đầu vào $n_1,...,n_r$ với chú ý sau:

(a).$2\le n_i<m$ với mọi i=1÷r.

(b).Tích của r-1 số nguyên tố sinh được trong r-1 lần sinh đầu có độ dài <m bit.

Ta biết rằng.

length(x)+length(y)-1≤length(x*y)≤length(x)+length(y) nên từ điều kiện (b) ta có $\sum_{i=1}^{r-1} n_i$-(r-1)≤length(F)<m *(2-21).*

Từ (a) thì $2\le n_i$ nên thay vào *(2.21)* ta có 2(r-1)-(r-1)<m hay r-1<m như vậy $\sum_{i=1}^{r-1} n_i<2m$ *(2-22).*

Tóm lại chúng ta cần phải sinh được r-1 số nguyên tố với tổng độ dài <2m bit.

Bây giờ xét đến số nguyên tố cuối cùng (số thứ r). Để có được số này chúng ta sử dụng thuật toán $\xi_{<n}$ với đầu vào là $n_r<m$. Theo công thức *(2-17)* (định lý 2.6) thì số nguyên tố thu được tại đầu ra sẽ có độ dài là l thoả mãn $n_r\le l<2m$ *(2-23).*

Bổ đề 2.10. *Với mọi d>1 và với mọi n>0 ta có* $(n-1)^d+n^{d-1}\le n^d$ *(2-24).*

Chứng minh.

$n^d-(n-1)^d$ $=(n-(n-1))(n^{d-1}+n^{d-2}(n-1)+...+(n-1)^{d-1})$

$\geq n^{d-1}$ hay

$n^d-(n-1)^d \geq n^{d-1}$ nên ta có ngay điều cần chứng minh. □

Đến đây chúng ta chứng minh một kết quả sau đây về thời gian thực hiện thuật toán.

Định lý 2.11. *Nếu thời gian để sinh một số nguyên tố độ dài $l<n$ của thuật toán $\xi_{<n}$ là $T(l)\leq Cl^d$ với $C\geq C_0$ và $d>6$* *(2-25).*

Thì thời gian sinh một số nguyên tố độ dài $l\geq n$ của thuật toán $\xi_{<n}$ là $T(l)\leq Cl^d$ *(2-26).*

Chứng minh.

Với r=1 thì thời gian thực hiện việc xây dựng F sẽ là $T_1\leq Cn_1^d\leq C(n-1)^d$. Trong khi đó trong trường hợp r>1 thì thời gian tính sẽ là:

$T_1 \leq (Cn_1^d+...+Cn_{r-1}^d)+Cn_r^d$

$=C(n_1^d+...+n_{r-1}^d)+Cn_r^d$

$\leq C(n_1+...+n_{r-1})^d+Cn_r^d$

$<2C(2m)^d.$

$=2C(2n/3)^d.$

Do $A=\frac{2}{\sqrt[d]{2^3}}<1$ với $d\geq 6$ nên với n đủ lớn ta có $2C(2n/3)^d.\leq C(n-1)^d$. Trong mọi trường hợp ta đều thu được:

$T_1 \leq C(n-1)^d.$

Thời gian thực hiện thuật toán ***POCK-GEN***$_F$ để sinh được một số nguyên tố độ dài 1 bit trong lớp $\mathbf{L}^F$ theo công thức *(2-16)* (định lý 2.7) là $T_2\leq C_0l^6$, do đó tổng thời gian thực hiện thuật toán là

$T =T_1+T_2$

$\leq C(n-1)^d+C_0l^6.$ *(2-27).*

Do $l\geq n$ và $d>6$ tức là $d-1\geq 6$, thay vào *(2.27)* ta có

$T \leq C(l-1)^d+Cl^{d-1}$

$$=C[(l-1)^d+l^{d-1}]. \qquad (2\text{-}28).$$

Áp dụng công thức *(2.24)* (bổ đề 1.10) ta có ngay điều cần chứng minh.□

2.3.5 Sự tồn tại thuật toán nhanh sinh được toàn bộ các số nguyên tố

2.3.5.1 Thuật toán

Qua các kết quả thu được trước đó, giả sử N là số sao cho các phát biểu nêu trong định lý 2.6 và định lý 2.7 là đúng với mọi n>N. Khi này thuật toán sinh các số nguyên tố được xây dựng như sau:

(a). Với đầu vào n≤N ta dùng phương pháp chẳng hạn như sàng Eratosthenes. Rõ ràng thời gian tính của thuật toán trong trường hợp này luôn giới hạn bởi hằng số $C^*=2^N$. Do đó ta có thể giả định rằng thuật toán $\xi_{<N}$ này có thời gian sinh được một số nguyên tố độ dài l<N là không quá Cl^7 với $C\geq C_0$ trong đó C_0 là hằng số nêu trong kết quả 2.4.

(b). Với đầu vào n>N, dùng phương pháp truy hồi theo độ dài số nguyên tố cần sinh thực hiện bằng cách sử dụng thuật toán ξ_n như đã trình bày.

Bằng phương pháp quy nạp dễ dàng chúng ta thấy rằng thuật toán trên sinh được toàn bộ các số nguyên tố và thời gian để sinh một số nguyên tố độ dài n là không quá Cn^7.

Kết quả cuối cùng mà chúng ta thu được ở đây là.

Định lý 2.12. *Thuật toán sinh được xây dựng ở trên có thể sinh toàn bộ số nguyên tố với thời gian sinh được mỗi số nguyên tố độ dài n là $O(n^7)$ (2-29).*□

2.3.5.2 Kết luận

Thuật toán trình bày ở trên nói chung có ý nghĩa rất lớn về mặt lý thuyết với sự khẳng định tính đa thức của nó. Tuy nhiên trên góc độ thực hành thì từ nguyên nhân là giá trị N tồn tại nêu trong thuật toán có thể là rất lớn trong khi đó chúng ta chỉ cần sinh những số nguyên tố với độ dài trong một phạm vi vừa phải nào đó mà thôi hơn nữa giá trị này cũng chưa chỉ ra cụ thể

nên rõ ràng ta chưa thể lập trình thực hiện nó. Theo quan điểm của chúng tôi việc sử dụng ý tưởng trong xây dựng thuật toán để tiến hành thiết lập một thuật toán có ý nghĩa thực hành sẽ thiết thực hơn nhiều. Chúng ta có thể lấy N=32 và cứ tiến hành sinh các số nguyên tố lớn theo phương pháp đã chỉ ra ở trên, tất nhiên có thể sẽ gặp phải những ngoại lệ nào đó mà chúng ta có thể không thành công trong một vài lần thực hiện nhưng bù lại thuật toán sinh này lại là thuật toán nhanh và việc lập trình thực hiện chúng lại dễ dàng. Do sự có thể khác nhau giữa giá trị N_0=32 so với giá trị sẽ tồn tại nêu trong phần chứng minh lý thuyết là N chúng ta sẽ gặp một số ngoại lệ khi tiến hành sinh các số nguyên tố có độ dài bit nằm trong khoảng từ N_0 đến N, ngoại lệ đáng kể nhất đó là sự không thoả mãn các tính chất được phát biểu trong định lý 2.6, nhưng điều này không có nghĩa là tính đa thức về thời gian tính của thuật toán bị sai và như vậy thuật toán dù xuất phát từ N_0>1 nào cũng vẫn là thuật toán thời gian đa thức bởi vì mọi ngoại lệ trong một khoảng hữu hạn N_0 đến N sẽ được bù thêm bằng một hằng số cộng về thời gian tính. Cuối cùng, trên quan điểm kinh tế, sẽ thiết thực hơn nhiều nếu chúng ta có được số liệu về thời gian sinh trung bình của thuật toán trong một vài độ dài số cần sinh cụ thể nào đó để đối với thời gian sinh của một số thuật toán sinh khác mà cơ sở dựa vào của chúng là các thuật toán kiểm tra tất định tất nhiên có thể là không đa thức.

TÀI LIỆU DẪN

[L. Đ. Tân], Lều Đức Tân. ***Một số thuật toán kiểm tra nhanh tính nguyên tố của các số trên một số lớp số***. *Luận án phó tiến sĩ Hà nội 1993*.

[Ribenboim], Paulo Ribenboim. ***The Little Book of Big Primes***. *Springe-Verlag 1991*.

CHƯƠNG III
CHƯƠNG TRÌNH SINH SỐ NGUYÊN TỐ MẠNH CHO HỆ MẬT ELGAMAL

MỞ ĐẦU

Trong chương II chúng ta đã biết đến một thuật toán nhanh mà bất cứ một số nguyên tố nào cũng có thể được sinh từ nó, tuy dược đánh giá là thời gian đa thức nhưng bậc khá cao (bậc 7) nên nếu chúng ta tiến hành việc sinh các số nguyên tố lớn (độ dài từ 500 đến 1500 bit) thì thời gian chi phí cho việc sinh sẽ rất lớn trong khi đó để có thể tìm được một số nguyên tố mạnh thì theo các đánh giá lý thuyết nêu trong mục 2 của chương I và đánh giá thực hành nêu trong phụ lục...thì rõ ràng chi phí này sẽ khó lòng chấp nhận được. Chính vì lý do trên, thêm vào nữa từ đánh giá của chương I thì độ an toàn của hệ mật dựa vào bài toán logarit trên trường GF(p) có thể nói chủ yếu dựa vào tính mạnh của tham số p, nên để có thể tìm nhanh và do đó sẽ được nhiều số nguyên tố mạnh để dùng chúng tôi quyết định chỉ tìm các số nguyên tố lớn và do đó các số nguyên tố mạnh chỉ trên những lớp số tìm nhanh nhất. Lớp số nguyên tố lớn mà chúng tôi lập trình để tìm là các số dạng $q=R_1q_1+1$ với độ dài của q_1 và R_1 xấp xỉ nhau và q_1 là số nguyên tố dạng $q_1=R_0 2^k+1$ với độ dài R_0 xấp xỉ k. Số lượng các số nguyên tố độ dài n bit mà chúng ta có thể tìm được trong phần mềm của chúng tôi đã được đánh giá bởi công thức 2-7 là $\pi_{GEN}=\frac{2^{3(m-1)}}{m^2}$ với m=n div 4.

Bên cạnh các trình bày mô tả thuật toán cần xây dựng, chúng tôi còn đưa ra một số kết quả đã có về lý thuyết liên quan đến việc đánh giá số các số nguyên tố mạnh (dưới tên các số Sophie theo cách gọi trong lý thuyết số). Việc đánh giá mật độ thật của các số nguyên tố mạnh và gần mạnh trong lớp số sinh được bởi thuật toán của chúng tôi sẽ được giải quyết trong chương III.

Mục 3 của chương nêu những cải tiến nho nhỏ trong một số phép tính số học cơ bản đã được gài đặt trong chương trình sinh số nguyên tố.

Tóm lại toàn bộ các vấn đề trình bày trong chương là những minh chứng cho việc nhóm đề tài quyết định tìm những số nguyên tố mạnh độ dài lớn trong lớp các số nguyên tố Pocklington tức là các số có dạng $q=Rq_1+1$ với R lẻ, q_1 là số nguyên tố dạng $q_1=r2^k+1$ với r lẻ mà chúng tôi gọi là các số nguyên tố Pepin và lập trình để thực hiện việc sinh các số nguyên tố mạnh dạng này. Để lấy làm ví dụ cho việc không khó tìm lắm của các số nguyên tố mạnh trong lớp trên, tại cuối của bản báo cáo nhóm đề tài trình bày trong phụ lục I toàn bộ các số nguyên tố mạnh không quá 2^{33} với nhân là 31 số nguyên tố Pepin đầu tiên của dãy $r2^{16}+1$.

3.1 LỚP Lp VÀ SỐ LƯỢNG SỐ NGUYÊN TỐ TRONG LỚP Lp

3.1.1 Lớp Lp(k)

Định nghĩa 3.1. ***L****p(k)={x=yp^k+1: với p là một số nguyên tố và 1≤y≤p^{2k}}.*□

Lớp **L**p(k) theo định nghĩa trên thực chất là lớp $\mathbf{L}^F$ với $F=p^k$ như vậy việc sinh các số nguyên tố trên lớp này chúng ta có thể dùng thuật toán ***POCK-GEN***$_F$ đã được trình bày trong chương trước.

3.1.2 Số các số nguyên tố độ dài n=⌊*3k*log*p*⌋ bit có trong lớp Lp(k)

Định lý 3.2. *Số các số nguyên tố độ dài n=⌊3klogp⌋bit có trong lớp* ***L****p(k) là*

$$\pi(p,k,n)\sim\frac{2^{\frac{2n}{3}}}{n}. \qquad (3\text{-}1).$$

Chứng minh.

Ta biết các số x có độ dài n bit là các số thoả mãn bất đẳng thức $2^{n-1}\le x<2^n$, do đó theo định lý Dirichlet về số các số nguyên tố có trong dãy At+B với gcd(A,B)=1 thì nếu ký hiệu $A=p^k$ thì $\varphi(A)=(p-1)p^{k-1}$ và ta có.

$$\begin{aligned}\pi(p,k,n) &= \pi_A(2^n)-\pi_A(2^{n-1})\\ &\sim\frac{2^n}{\varphi(A)\ln 2^n}-\frac{2^{n-1}}{\varphi(A)\ln 2^{n-1}}\end{aligned}$$

$$= \frac{2^{n-1}}{(p-1)p^{k-1}\ln 2}\left(\frac{2}{n} - \frac{1}{(n-1)}\right)$$
$$= \frac{(n-2)2^{n-1}}{n(n-1)(p-1)p^{k-1}\ln 2}$$

Do n=$\lfloor 3k\log p \rfloor$ ta có $2^n \approx p^{3k}$ nên $\pi(p,k,n) \sim \frac{2^{\frac{2n}{3}}}{n}$ và đây là điều cần chứng minh.□

Từ kết quả trên thì lực lượng các số nguyên tố trong mỗi lớp đặc biệt (lớp **L**p(k)) là rất lớn và đủ cho chúng ta sử dụng.

3.1.3 Thuật toán sinh số nguyên tố n bit trên các lớp Lp(k) với p nhỏ

Trước hết khái niệm p nhỏ mà chúng tôi muốn đề cập ở đây là những số có độ dài không quá 32 bit. Như trên đã nói đến là việc sinh các số nguyên tố chúng ta dùng thuật toán ***POCK-GEN$_F$***, nhưng do F chỉ có dạng đặc biệt ($F=p^k$) nên thời gian thực hiện thuật toán sẽ được giảm bớt với nguyên nhân sau đây.

Thứ nhất. F chỉ có duy nhất ước nguyên tố (đó là p) nên bộ tham số M_i của thuật toán ***Pock-test$_F$*** với xác suất sai lầm loại 1 không quá α chỉ là một tham số $M=\left\lfloor \log_p \frac{1}{\alpha} \right\rfloor$. (3-2).

Do đó thời kiểm tra một số tự nhiên độ dài n bit là $T_{Test}(n) \le Mn^3$, tương ứng, thời gian để sinh một số nguyên tố độ dài n bit của thuật toán sinh ***POCK-GEN$_F$*** là $T_{Gen}(n) \le Mn^3 m(\ln m+6)$ vì n=3m nên $T_{Gen}(n) \le 2Mn^4 \ln n$.

Thứ hai. Việc xây dựng F là rất đơn giản vì $F=p^k$ mà những số nguyên tố nhỏ là rất dễ tìm bằng phương pháp đơn giản là sàng Eratosthenes với không quá 6514 phép chia cho các số nguyên tố nhỏ hơn 17 bit, còn giá trị k cũng tìm được với không quá $k \le \frac{n}{3}$ phép nhân với một số nhỏ (nhân với p). Như vậy thời

gian sinh được một số nguyên tố n bit có thể coi chính là $T_{Gen}(n)$ như đã nói ở trên.

3.1.4 Trường hợp p=2

Như tác giả trong [L. Đ. Tân] đã xem xét đến, trường hợp p=2 được hỗ trợ bởi một kết quả khá đơn giản đó là định lý Pepin mà chúng ta có thể trình bày lại ở đây như sau:

Định lý Pepin. *Cho* $p=r2^k+1$ *với* $k>1$ *và* $r<2^k$ *(3-3).*
Khi đó p là nguyên tố khi và chỉ khi tồn tại giá trị $a<p$ *thoả mãn điều kiện sau*

$$a^{(p-1)/2}=-1 \pmod p \quad (3\text{-}4).$$

Chứng minh.

Điều kiện cần là hiển nhiên.

Ngược lại, từ (3-4) ta có ngay $a^{(p-1)/2}\neq 1 \pmod p$ và $a^{p-1}=1 \pmod p$ đồng thời $a^{(p-1)/2}-1=-2 \pmod p$ nên hiển nhiên $\gcd(a^{(p-1)/2}-1,p)=\gcd(-2,p)=1$ nên theo định lý Pocklington ta có mọi ước nguyên tố q của p đều có dạng $q=s2^k+1$. Do điều kiện (3-3) là $r<2^k$ nên p chỉ có 1 ước khác 1 hay nó là số nguyên tố. □

Chú ý 3.3. *Giá trị a nêu trong định lý Pepin chính là giá trị thoả mãn điều kiện.*

$$J(a/p)=-1 \text{ (với } J(a/p) \text{ là ký hiệu Jacobi)} \quad (3\text{-}5).$$

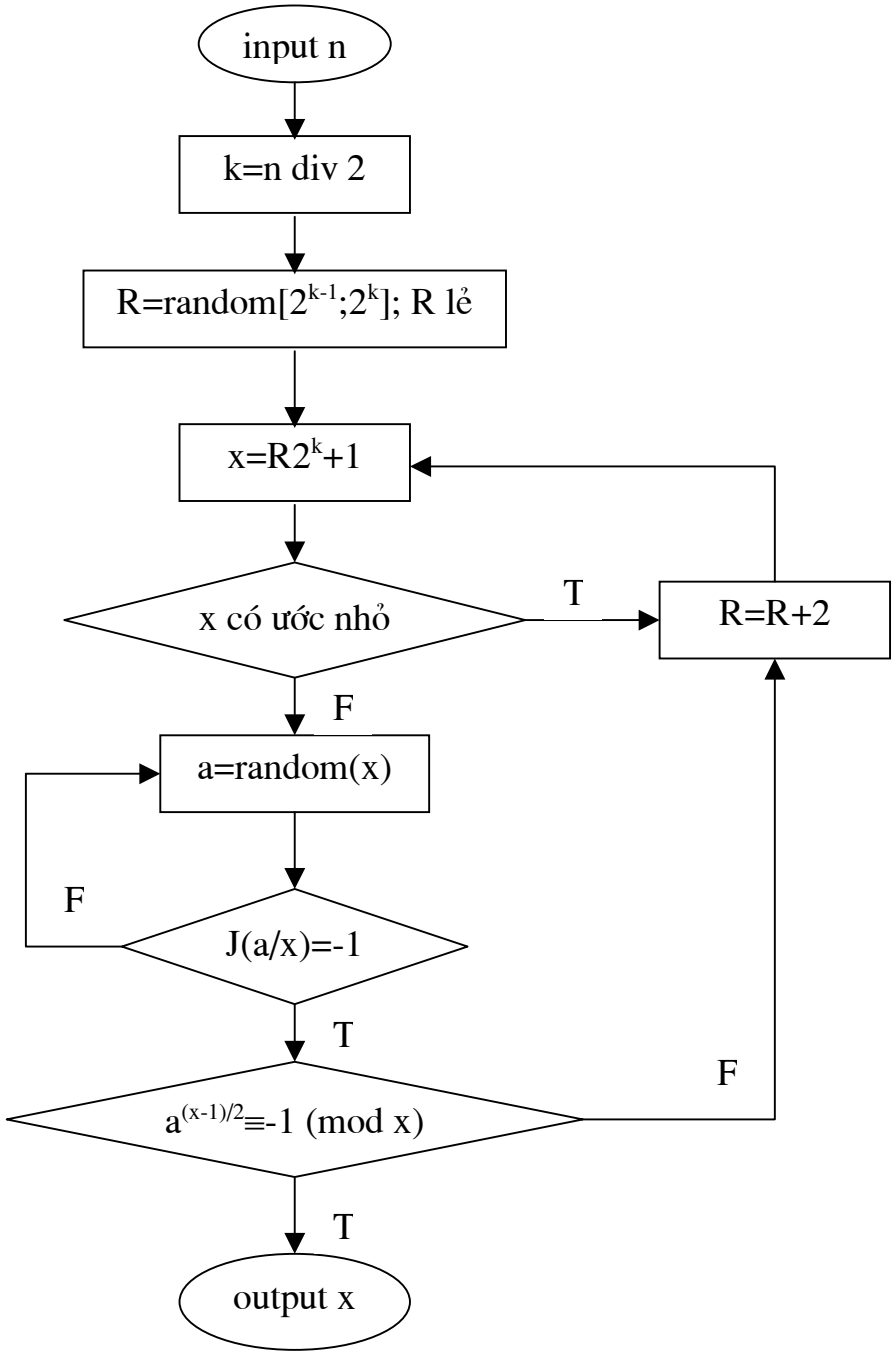
Chứng minh.

Nếu p là nguyên tố thì ký hiệu Jacobi trùng với ký hiệu Legangdre tức là $J(a/p)=L(a/b)=a^{(p-1)/2} \pmod p$. □

Với chú ý trên thì thay vì cho việc thử có thể đến M lần để tìm một không thăng dư bậc hai bằng cách xét điều kiện (3-4) là $a^{(x-1)/2}\neq 1 \pmod x$ chỉ bằng xét điều kiện (3-5) là $J(a/n)=-1 \pmod x$ mà thôi. Nếu như việc tính một luỹ thừa modulo cần đến n^3 phép tính trên bít thì việc tính J(a/n) theo định lý bình phương tương hỗ chỉ cần đến n^2 phép toán. Như vậy trong trường hợp

p=2 thì chúng ta chỉ cần thực hiện cùng lắm M lần tính J(a/n) và chỉ cần đúng một lần tính $a^{(x-1)/2}$ (mod x). Nói tóm lại, nếu như theo thuật toán thông thường chúng ta cần đến Mn^3 phép toán để kiểm tra một số n bít thì bằng cách sử dụng kết quả trên chúng ta chỉ cần đến n^3+Mn^2 phép toán mà thôi. Đây là một sự rút gọn đáng kể bởi vì theo công thức (3-2) khi p=2 thì $M=\left\lfloor \log\frac{1}{\alpha} \right\rfloor$ không phải là nhỏ nếu α rất nhỏ. Trong chương trình sinh số nguyên tố mạnh, chúng tôi sẽ sử dụng thuật toán tìm các số nguyên tố lớn trên lớp $\mathbf{L}^F$ với $F=2^k$ với những lý do đã nêu trên.

Sơ đồ thuật toán 2.3. (sinh số nguyên tố dạng $x=R2^k+1$ gài đặt trong chương trình).



3.2 VIỆC SINH CÁC SỐ NGUYÊN TỐ MẠNH VÀ GẦN MẠNH

3.2.1 Khái niệm số nguyên tố mạnh và gần mạnh

Mục đích của đề tài là tìm những số nguyên tố dạng p=2q+1 với q cũng là số nguyên tố, những số nguyên tố này với tư cách là tham số modulo cho

các hệ mật mà độ mật dựa vào tính khó giải của bài toán logarit trên trường GF(p) sẽ làm cho hệ mật đạt được tính an toàn cao nhất và cũng chính vì vậy mà chúng được gọi là các số nguyên tố mạnh. Cũng đạt được tính năng không thua kém mấy về độ an toàn là các số dạng p=Rq+1 với R<<p cụ thể ở đây R≤log*p*, cụ thể là nếu như bài toán logarit chỉ để lộ duy nhất 1 bit có ý nghĩa thấp nhất nếu dùng các số nguyên tố mạnh thì nó cũng sẽ chỉ để lộ log*R* bit thấp nhất nếu dùng các số dạng p=Rq+1, cho nên việc sử dụng các số nguyên tố dạng này cũng có thể được chấp nhận trong các hệ mật nói trên. Định nghĩa dưới đây là một sự thống nhất trước về mặt khái niệm các đối tượng chúng ta quan tâm trong đề tài này.

Định nghĩa 3.4. *Số nguyên tố p=Rq+1 với q cũng là nguyên tố được gọi là số nguyên tố gần mạnh nếu R≤log q, trường hợp R=2 thì p được gọi là số nguyên tố mạnh.*

Số nguyên tố q nêu trong trên được gọi là nhân nguyên tố của số nguyên tố p. □

Việc kiểm tra tính gần mạnh của một số được dựa vào kết quả sau đây.

Định lý 3.5. *Cho p=Rq+1 với q nguyên tố và R≤log q* *(3-6).*

Khi đó p là nguyên tố khi và chỉ khi thoả mãn các điều kiện sau

(a). $2^{p-1}=1 \pmod p$ *(3-7).*

(b). $gcd(2^R-1,p)=1$ *(3-8).*

Chứng minh.

Điều kiện cần là hiển nhiên.

Ngược lại từ điều kiện (3-6) là R≤log q ta có $2^{(p-1)/q}=2^R<p$ vì vậy hiển nhiên $2^{(p-1)/q}\neq 1 \pmod p$, cùng với điều kiện (3-8) thì giá trị 2 chính là bằng chứng để p thoả mãn điều kiện của định lý Pocklington do đó p là số nguyên tố và theo định nghĩa 3.4 nó là số nguyên tố gần mạnh. □

3.2.2 Số nguyên tố Sophie

Trong lý thuyết số một khái niệm được Sophie Germain đưa ra vào năm 1825 có liên quan đến các số nguyên tố cần tìm của chúng ta, đó là các số nguyên tố Sophie Germain và được định nghĩa như sau.

Định nghĩa số nguyên tố Sophie

Số nguyên tố q được gọi là số nguyên tố Sophie khi p=2q+1 cũng là số nguyên tố. □

Bà đã chứng minh được một định lý đó là.

Định lý Sophie. *Nếu q là số nguyên tố Sophie thì hoặc không tồn tại hoặc tồn tại các số nguyên x, y, z khác 0 và không là bội của q sao cho* $x^q+y^q=z^q$. □

Định lý trên về mặt lý thuyết là một khẳng định tính đúng đắn cho trường hợp đầu tiên của định lý cuối cùng của Fermat, tuy nhiên cái mà chúng ta quan tâm hơn là kết quả sau, do Powell chứng minh năm 1980, về số các số nguyên tố q≤x thoả mãn Aq+B cũng nguyên tố với AB chẵn và gcd(A,B)=1, ký hiệu là $S_{(A,B)}(x)$ như sau.

Định lý Powell. $S_{(A,B)}(x)<\frac{Cx}{(Logx)^2}$ *với C là một hằng số. Hơn nữa ta có* $\lim_{x\to\infty}\frac{S_{(A,B)}(x)}{\pi(x)}=0$. □

Trường hợp riêng với A=2 và B=1, thì ta có số các số nguyên tố Sophie, ký hiệu là S(x). Công việc mà đề tài này hướng tới có thể hiểu không gì khác là đi tìm bằng thực hành các số nguyên tố Sophie. Qua giới hạn nêu trong định lý Powell thì công việc của chúng ta sẽ cực kỳ khó khăn bởi vì không những bởi việc tìm những số nguyên tố càng lớn thì càng khó do thưa hơn mà trong những số nguyên tố lớn này thì số các số Sophie cũng càng thưa hơn.

3.2.3 Thuật toán sinh số nguyên tố gần mạnh

3.2.3.1 Thuật toán

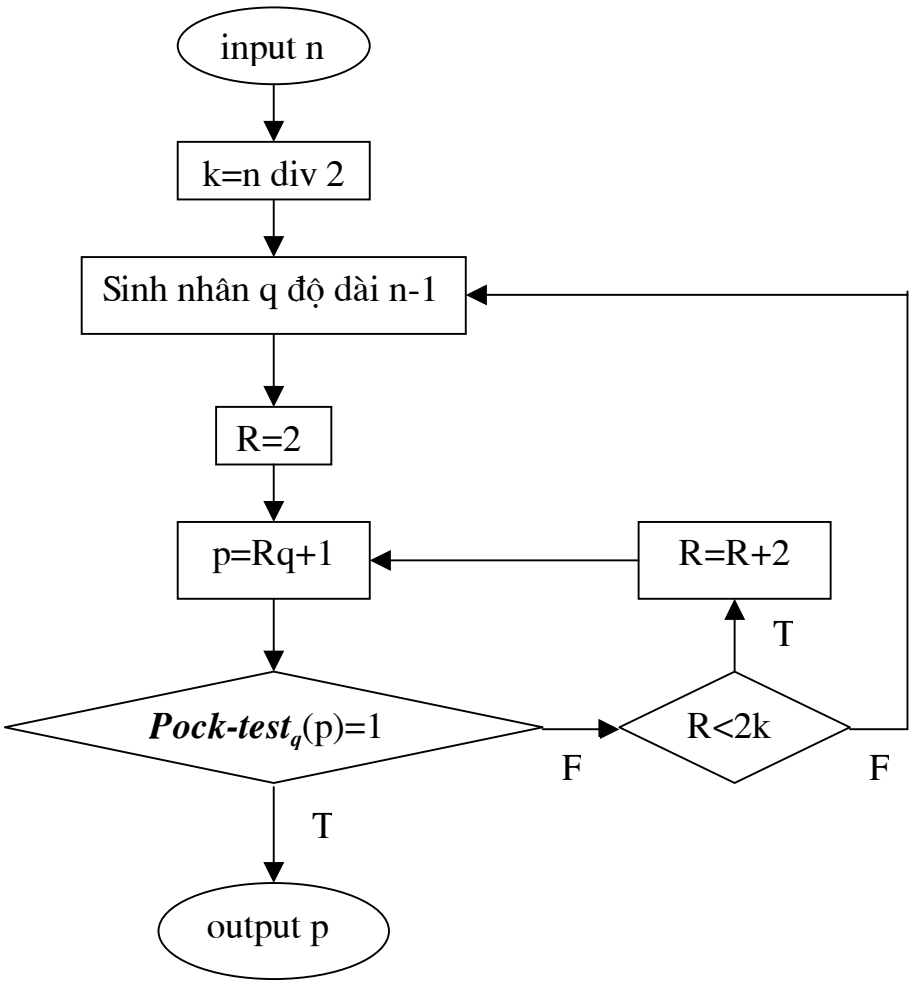
Theo như định nghĩa 3.4. về các số nguyên tố gần mạnh thì việc tìm các số loại này sẽ gồm hai bước.

Bước 1. Tìm nhân nguyên tố q.

Bước 2. Tìm số nguyên tố gần mạnh có nhân là q (nếu có).

Rõ ràng để tìm được số nguyên tố mạnh độ dài n bit thì trong bước 1 chúng ta cần tìm các nhân nguyên tố n-1 bit, vấn đề này đã được giải quyết ở mục trên. Công việc ở bước hai chỉ là tìm số nguyên tố đầu tiên (nếu có) trong đoạn dãy 2q+1, 4q+1,....,2kq+1 với k=n div 2 và thuật toán dùng để kiểm tra tính nguyên tố các số trong đoạn dãy trên không gì khác là thuật toán ***Pock-test***$_F$ với F=q. Tóm lại thuật toán trên có thể mô tả theo sơ đồ sau.

Sơ đồ thuật toán 3.6. (sinh số nguyên tố gần mạnh)



3.2.4 Thuật toán sinh nhanh các nhân nguyên tố lớn được gài đặt

Trong thuật toán sinh các số nguyên tố mạnh và gần mạnh tại mục trước thì các hàm và thủ tục đều là trực tiếp trừ ra thủ tục sinh nhân nguyên tố lớn, tại mục này chúng tôi sẽ trình bày chi tiết thủ tục này. Để sinh nhanh các số nguyên tố lớn (độ dài từ 500 đến 1500 bit) sẽ được dùng để tạo nhân của các số nguyên tố mạnh và gần mạnh chúng tôi sử dụng hai phương pháp chính như sau:

3.2.4.1 Phương pháp sinh nhanh từ số nguyên tố nhỏ

Phương pháp sinh nhanh từ số nguyên tố nhỏ mà chúng tôi sẽ thực hiện việc lập trình thực chất là một thu hẹp của thuật toán đã được trình bày trong mục 3.1.3. Phương pháp này dùng để sinh các số nguyên tố có độ dài bằng nửa độ dài của nhân nguyên tố q. Sự khác biệt đôi chút so với thuật toán đã trình bày trước đó là tham số k được lấy ở đây sẽ là số đầu tiên sao cho $\log(p^k)\geq\frac{m}{2}$ (với m là độ dài bit của số nguyên tố cần sinh) chứ không phải là $\geq\frac{m}{3}$. Việc lấy này không phải xuất phát từ lý do giảm bớt được thủ tục xác định tính chính phương của biệt thức B^2-4A mà ở chỗ đảm bảo cho các số nguyên tố ở các lớp ứng với p khác nhau là hoàn toàn khác nhau do đó chúng ta sẽ thuận lợi hơn nhiều khi cần tính đến lực lượng của các số nguyên tố có thể sinh được.

Bằng cách lặp lại các lập luận đã thực hiện trong chứng minh định lý 3.3 chúng ta thu được số các số nguyên tố độ dài m bit trong mỗi lớp $\mathbf{L}_p$ vào khoảng $\frac{2^{\frac{m}{2}}}{m}$. Trong khi đó chúng ta có khoảng $\pi(2^{32})\approx 2^{27}$ số nguyên tố nhỏ và như vậy số các số nguyên tố khác nhau độ dài m bit được sinh từ phương pháp này sẽ là

Tuy nhiên trong chương trình thực chúng tôi chỉ gài đặt với p=2 và như vậy số các số nguyên tố 2m bit có thể sinh được trong phần này chỉ là

Công thức 3.7. *Số các số nguyên tố độ dài 2m bit dạng $R2^m+1$ sinh được trong phần mềm là*

$$\pi_1=\frac{2^{m-1}}{m}. \qquad (3\text{-}9).\ \square$$

3.2.4.2 Phương pháp gấp đôi độ dài từ số nguyên tố lớn

Nội dung của phương pháp này là xuất phát từ một số nguyên tố F độ dài 2m sinh được từ mục 4.1, chúng ta tìm các số nguyên tố q độ dài 4m dưới dạng q=RF+1 với độ dài của R cũng là m. Thuật toán dùng để sinh cũng vẫn là thuật ***POCK-GEN$_F$*** và cũng vẫn dùng lập luận ở trên thì ứng với mỗi số nguyên tố F độ dài 2m bit khác nhau ta có khoảng $\frac{2^{2m}}{4m}=\frac{2^{2m-2}}{m}$ số nguyên tố độ dài 4m bit khác nhau. Kết hợp với công thức 3.7 chúng ta thu được.

Công thức 3.8. *Số các nhân nguyên tố độ dài 4m bit sinh được là*

$$\pi_{GEN}=\frac{2^{3(m-1)}}{m^2} \qquad (3\text{-}10).\square$$

Một thực tế thường xảy ra là độ dài của số nguyên tố mạnh cần sinh không phải luôn luôn có dạng n=4m+1 và vì vậy số nhân nguyên tố q không phải lúc nào cũng có độ dài chia hết cho 4 nên chúng ta cần có một sự điều chỉnh thích hợp. Cụ thể trong chương trình của chúng tôi nếu n là độ dài bit của số nguyên tố mạnh cần được sinh thì độ dài của số nguyên tố dạng $R2^m+1$ cần sinh theo phương pháp sinh nhanh từ số nguyên tố nhỏ sẽ là n_1=n div 2 và giá trị m=n_1 div 2.

3.3 TÍNH TOÁN TRÊN CÁC SỐ LỚN

Như đã đăng ký trong đề tài, trong phần này chúng tôi chú ý đặc biệt đến một số phép toán cơ bản quyết định đến tốc độ tính toán của chương trình đó là gồm các phép toán nhân, chia và luỹ thừa số lớn. Việc trình bày của chúng tôi trong phần này không đi vào viết lại những thuật toán đã có ở những tài liệu mà chúng tôi tham khảo mà chủ yếu là đưa ra và phân tích các

cải tiến của chúng tôi theo hai nghĩa: một là để thực hiện lập trình được thuật toán sẵn có và hai là cải thiện được đôi chút về thời gian tính.

3.3.1 Phép nhân số lớn

Chúng ta đều biết cơ sở để xây dựng phép toán nhân trên các số lớn là công thức nhân sau.

Công thức 3.9.

Cho $X=x_0+x_1q+...+x_mq^m$ *và* $Y=y_0+y_1q+...+y_nq^n$ *ta có* $XY=\sum_{k=0}^{m+n}\sum_{i+j=k}x_iy_jq^k$ *(3-11).*□

Theo công thức nhân (3-11) trên thì để thực hiện một phép nhân hai số lớn có độ dài q phân là n, chúng ta cần tối thiểu n^2 phép toán nhân hai số trong phạm vi q. Trong [Rieshel] tác giả có trình bày trong phần phụ lục một thuật toán nhân có thời gian tính chỉ là $O(n^{1.5})$, cụ thể như sau.

Đầu tiên chúng ta xét trường hợp tích của hai số có độ dài 2 trong hệ Q phân nào đó. Giả sử $X=x_0+x_1Q$ và $Y=y_0+y_1Q$, dễ dàng kiểm tra được đẳng thức sau.

Công thức 3.10.

$$XY=x_0y_0+[x_0y_0+(x_0-x_1)(y_1-y_0)+x_1y_1]Q+x_1y_1Q^2$$
$$=x_0y_0(1+Q)+(x_0-x_1)(y_1-y_0)Q+x_1y_1(Q+Q^2) \qquad (3\text{-}12).\ \square$$

Như vậy để thực hiện tính toán theo công thức (3-12) chúng ta chỉ cần tính 3 phép nhân các số trong phạm vi Q. Bây giờ, nếu chúng ta xét $Q=q^{2^k}$ thì bằng cách truy hồi theo công thức (3-12) k bước thì tổng số phép nhân hai số trong phạm vi q phục vụ thuật toán này chỉ là $n=3^k$. Rõ ràng $2^{k-1}<n\le 2^k$ với n là độ dài q phân của các thừa số nhân cho nên nếu theo công thức (3-11) ta phải cần đến $n^2>2^{2(k-1)}=4^{k-1}$ phép nhân. Tóm lại thời gian tính toán của phép nhân

hai số lớn độ dài khai triển q phân là n theo cách trên sẽ chỉ là $O(n^{Log3})\approx O(n^{1.5})$.

Trong một số chương trình nguồn tính toán trên các số lớn như [N. V. Khán], [V. V. Xứng], [Kapp],... mà chúng tôi có trong tay thì chưa có chương trình này thực hiện phép nhân theo công thức (3-11). Để thực hiện thuật toán theo công thức (3-12) vừa trình bày cần đến kỹ thuật lập trình cao vì bản chất của thuật toán là đệ quy nên rất khó thực hiện. Chúng tôi tránh việc phải thực hiện đệ quy bằng chia thuật toán nhân ra các thuật toán nhân con với số thuật toán con bằng số k nêu ở trên, cụ thể với $q=2^{16}$ độ dài tối đa cần tính toán $n=2^5$ (theo đăng ký là 1500 bit) . Rất tiếc do trình độ lập trình của mình còn thấp nên trong gài đặt thực nghiệm chúng tôi chưa thấy ưu điểm rõ rệt của thuật toán này. Chú ý rằng phần mềm trình bày trong [V. V. Xứng], các tác giả đã thành lập riêng thuật toán bình phương hai số lớn, thuật toán bình phương này có thời gian tính nhanh gấp đôi so với thuật toán nhân hai số cùng độ dài theo công thức (3-11) cho nên việc phát hiện tính nhanh hơn của thuật toán mới càng khó.

3.3.2 Phép chia hai số lớn

Các thuật toán chia hai số lớn được các tác giả của các tài liệu [N. V. Khán], [Khán-Tân] trình bày khá kỹ lưỡng, cho nên chúng tôi không trình bày lại ở đây mà chỉ giới thiệu và phân tích cụ thể thuật toán được cài đặt trong phần mềm sinh số nguyên tố mạnh.

Cơ sở của thuật toán dựa vào kết quả đoán thương nhanh sau.

Công thức 3.11.

Giả sử $X<QY$ và độ dài Q phân của Y là $n>1$, ký hiệu $x=x_{n-1}+x_nQ+x_{n+1}Q^2$ và $y= y_{n-1}+y_nQ$.

Khi đó nếu x div y=a thì X div Y=a hoặc a-1 *(3-13).*□

Chúng tôi quan tâm đến một trường hợp đặc biệt của mẫu số đó là $y_n=1$ và $y_{n-1}=0$. Trong trường hợp này chúng ta có ngay giá trị thương mà không cần tính x, y và việc chia x cho y bởi hệ quả sau.

Công thức 3.12.

Nếu $x_{n+1}=1$ *thì X div Y=Q-1.* *(3-14).*

Ngược lại X div Y=x_n *hoặc* x_n*-1* *(3-15).* □

Dựa vào một số đặc điểm sau của chương trình cần xây dựng là.

(i).Chương trình thực hiện thuật toán kiểm tra tính nguyên tố mà thời gian tính toán của nó chủ yếu là phục vụ việc tính phép luỹ thừa modulo các số lớn.

(ii).Trong phép toán này phép chia được thực hiện rất nhiều lần (trung bình là 1.5LogN phép chia) với đặc điểm là mẫu số (ký hiệu là M) không đổi.

(iii).Phép chia luôn được thực hiện với độ dài tử số được giới hạn bởi đúng 2 lần độ dài mẫu số.

Chính từ những đặc điểm trên chúng ta có thực hiện được một số vấn đề như sau.

(i). Tạo trước Log n giá trị M_i (ở đây n là độ dài theo cơ số $q=2^{16}$ của giá trị modulo) mỗi khi thực hiện phép luỹ thừa các mẫu số trung gian. M_i là các số thoả mãn điều kiện sau.

M_i là bội của số modulo M và có dạng $q^{t(i)}+R_i$ với $R_i<N$, $t(i)=n+2^i$.

(ii).Thuật toán tìm N mod M được thực hiện theo công thức sau

N mod M=((...(N mod M_r) mod M_{r-1})...) mod M).

Nhận xét rằng tại bước truy hồi thứ i, nếu xét cơ số khai triển là $Q=q^{t(i)-n}$ thì tử số và mẫu số của phép chia thoả mãn giả thiết của công thức 3.12 cho nên chúng ta dễ dàng đoán được thương đúng theo công thức này với chú ý rằng độ dài q phân của thương là $t(i)-n=2^i$. Phép tính của chúng tôi vừa nêu tuy không giảm được về bậc so với thuật toán chia dùng trực tiếp công thức (3-11) nhưng trong gài đặt thực tế nó có thời gian tính nhanh hơn một chút (khoảng từ 15÷20%).

3.3.3 Phép luỹ thừa modulo các số lớn

3.3.3.1 Công thức luỹ thừa theo khai triển nhị phân của số mũ

Cho $B=b_0+2b_1+...+2^kb_k$, chúng ta có các công thức tính luỹ thừa một số lớn dựa trên khai triển nhị phân của số mũ như sau.

Công thức 3.13. (công thức luỹ thừa xuôi)

$$X^B=X^{b_0}.(X^2)^{b_1}.....(X^{2^k})^{b_k} \qquad (3\text{-}16).$$

Công thức 3.14. (công thức luỹ thừa ngược)

Ký hiệu $X_k=X^{b_k}$, *và với* $i<k$ *thì* $X_{i-1}=X^{b_i}X_i^2$ *ta có* $X^B=X_0$. $(3\text{-}17).$

Trong cả hai công thức trên ta thấy rằng, để thực hiện việc tính toán luỹ thừa một số lớn chúng ta cần đến k phép bình phương và một số phép nhân bằng số bít 1 có trong khai triển nhị phân của số mũ B (trọng số của B). Nói chung việc giải đặt phép luỹ thừa theo một trong hai công thức trên là tương đương về thời gian tính tuy nhiên nếu dùng công thức luỹ thừa ngược thì trong trường hợp X là số nhỏ thì việc tính toán sẽ nhanh hơn một chút.

3.3.3.2 Công thức luỹ thừa theo khai triển a phân của số mũ

Chúng ta không bàn đến lợi thế trong lĩnh vực này của phép luỹ thừa ngược mà khai thác ưu điểm của nó trong khả năng thay đổi cơ số biểu diễn của số mũ.

Giả sử biểu diễn của B theo cơ số a nào đó là $B=c_0+ac_1+...+a^hc_h$. Khi đó các công thức tính luỹ thừa tương ứng sẽ là.

Công thức 3.13'. $X^B=X^{c_0}.(X^a)^{c_1}.....(X^{a^h})^{c_h}$. $(3\text{-}18).$

Công thức 3.14'. *Ký hiệu* $X_h=X^{c_h}$, *và với* $i<h$ *thì* $X_{i-1}=X^{c_i}X_i^a$ *ta có* $X^B=X_0$.$(3\text{-}19).$

Chúng ta xét trường hợp $a=2^t$. Trong trường hợp này, nếu như sử dụng công thức 3-13' thì việc tính toán vẫn như hệt như khi sử dụng công thức 3-13 cần đến (vì việc tính mỗi thừa số $(X^{a^i})^{c_i}$, ta sử dụng kết quả đã được tính của

bước trước đó là $X^{a^{i-1}}$, luỹ thừa a giá trị này rồi tiếp đến luỹ thừa c_i kết quả thu được), thì đối với công thức 3.14' xuất hiện sự khác biệt mà chúng ta có thể lợi dụng được đó là chúng ta có thể tính sẵn các giá trị X^j với j=2÷(a-1).

Chúng ta dừng một chút để phân tích cải tiến đầu tiên này.

Ta để ý rằng, trọng số trung bình của số mũ là xấp xỉ $\frac{1}{2}$ độ dài của nó do vậy khi áp dụng công thức khai triển nhị phân của số mũ chúng ta cần đến trung bình là $\frac{n}{2}$ phép nhân.

Trong phần mềm của Kapp, tác giả sử dụng công thức khai triển tứ phân (a=4 hay t=2) và khi này trung bình giá trị tứ phân khác 0 của số mũ xấp xỉ $\frac{3}{4}$, do X^2 và X^3 đã được tính sẵn nên thuật toán chỉ cần trung bình là phép nhân $\frac{3n}{8}<\frac{n}{2}$.

Bây giờ, nếu ta sử dụng khai triển $a=2^t$ phân, theo lập luận trên chúng ta chỉ cần trung bình là $\frac{2^t-1}{2^t}\frac{n}{t}<\frac{n}{t}$ phép nhân. Tự nhiên mà nói, nếu chọn t càng lớn thì số phép toán phải thực hiện càng giảm đi, tuy nhiên vấn đề không đơn giản như vậy. Để gài đặt được thuật toán với t lớn thì chúng ta phải tính sẵn và khó khăn hơn nữa là phải lưu trữ các giá trị tính sẵn đó. Trong tính toán thực hành chúng tôi rút ra rằng việc chọn t=5 (a=32) là thích hợp nhất.

3.3.3.3 Phương pháp khai triển số mũ theo cơ số thay đổi (cơ số động)

Giả sử $B=d_0+d_1 2^{t_1}+...+d_s 2^{t_s}$ với $2^r \le d_i < 2^{r+1}$ với mọi i=1÷s, riêng d_0 có thể nhỏ hơn 2^r. Khi này ta sẽ có $X^B=X_0$ trong đó $X_s=X^{d_s}$ và $X_{i-1}= X^{d_i} X_i^{\,2^{t_i}}$.

Như vậy chỉ cần tính sẵn 2^r giá trị X^d với $d=2^r\div(2^{r+1}-1)$ ta có thể tính được các giá trị X_i với mọi i=1÷s mà trong đó cần đến t_i phép bình phương để tính $X_i^{\,2^{t_i}}$ và một phép nhân với số đã tính sẵn là X^{d_i} riêng X_0 trong trường hợp $d_0<2^r$ thì ta cần phải tính X^{d_0} theo cách thông thường.

Chú ý rằng.

(i). Tổng số phép bình phương phải thực hiện trong s+1 bước tính toán trên cũng chính bằng độ dài nhị phân của B.

(ii).Do điều kiện $2^r \le d_i < 2^{r+1}$ nên ta luôn có $t_i - t_{i-1} > r$ do đó giá trị s ở đây luôn không quá giá trị h tương ứng trong khai triển $a=2^{r+1}$ phân của B, cho nên số phép nhân của thuật toán này sẽ được giảm đi và đặc biệt là số lượng các số cần phải tính sẵn được giảm đi một nửa.

Tóm lại để tính được luỹ thừa X^B chúng ta cần đến LogB+s phép nhân hoặc bình phương.

Phương pháp vừa trình bày trên còn được gọi là phương pháp trượt cửa sổ.

Trên đây chúng tôi đã giới thiệu một vài cải tiến nho nhỏ trong thuật toán tính luỹ thừa, tất nhiên các cải tiến này không mang tính đột biến về bậc tính toán mà chỉ là sự thay đổi đôi chút về hệ số của bậc cao nhất. Hy vọng rằng với những sự góp gió trên về ba thuật toán cơ bản đó là nhân, chia và luỹ thừa chúng ta có thể cải thiện đôi chút về tốc độ tính toán của các phần mềm sử dụng các hệ mật khoá công khai cũng như các phần mềm liên quan đến tính toán trên các số lớn nói chung.

TÀI LIỆU DẪN

[N. V. Khán]. Nguyễn Văn Khán. ***Nghiên cứu hệ mật RSA.*** *Đề tài cấp cơ sở 1996.*

[L. Đ. Tân]. Lều Đức Tân. ***Một số thuật toán kiểm tra nhanh tính nguyên tố của các số trên một số lớp số****. Luận án phó tiến sĩ Hà nội 1993.*

[V. V. Xứng]. Vũ Văn Xứng. ***Bảo mật thông tin kinh tế xã hội trong mạng máy tính.*** *Đề tài cấp Ban 1999.*

PHỤ LỤC 1. **CÁC KẾT QUẢ THỬ NGHIỆM**

1.1. Giới thiệu về phần mềm

Chương trình được viết bằng ngôn ngữ C với hai tham số đầu vào là số lượng và độ dài bit của các số nguyên tố mạnh cần sinh (hai tham số trên được nhập từ bàn phím) và đầu ra là các số nguyên tố mạnh sinh được.

1.1.1. Về lưu trữ các số nguyên tố mạnh sinh được

Các số nguyên tố mạnh sinh được bởi chương trình sẽ được ghi vào tệp với tên tương ứng là "prim_M.str" (M là số ghi độ dài bit của số được sinh) và để trong các thư mục "store_st".

Đối với số nguyên tố mạnh M bit được lưu trữ dưới dạng một dãy q phân với $q=2^{16}$ với độ dài N được tính bằng công thức $N=(M \text{ div } 16)+\Delta$ trong đó $\Delta=1$ nếu $(M \bmod 16)>0$ và $\Delta=0$ trong trường hợp ngược lại.

1.1.2. Vấn đề ghi lại bằng chứng về tính nguyên tố và tính nguyên tố mạnh của các số sinh được

Trong chương trình sinh số nguyên tố mạnh chúng tôi có lưu lại trong tệp "ho_so.txt" các tham số cơ bản như độ dài bit, thời điểm sinh, số lượng số nguyên, số lượng số nguyên tố của quá trình sinh ra một số nguyên tố mạnh. Ngoài các tham số cơ bản trên chúng tôi còn lưu thêm một số tham số phục vụ cho việc thẩm định lại tính nguyên tố và tính mạnh của số được sinh.

Ta biết rằng số nguyên tố mạnh được sinh trong chương trình là số p có dạng $p=2q+1$ với $q=rq_1+1$ và q_1 là số nguyên tố Pepin tức là $q_1=r_1 2^k+1$, trong đó $r<q$ và $r_1<2^k$. Việc khẳng định tính nguyên tố mạnh của p chính là chứng minh q_1, q và p nguyên tố.

(1). Để chứng tỏ q_1 là số nguyên tố, theo định lý Pepin, chúng ta cần chỉ ra số $a_1<2^k$ thoả mãn điều kiện:

(1.a). $a_1^{\frac{q_1-1}{2}} = -1 \pmod{q_1}$.

(2). Để chứng minh q là số nguyên tố, theo định lý Pocklington, chúng ta cần chỉ ra được số a<q thoả mãn các điều kiện:

(2.a). $a^{q-1}=1 \pmod q$.

(2.b). $a^{\frac{q-1}{q_1}} = a^r \neq 1 \pmod q$.

(2.c). $\gcd(a^r-1,q)=1$.

(3). Để chứng minh p là nguyên tố, theo định lý 3.5 , chúng ta cần chỉ ra rằng:

(3.a). $2^{p-1}=1 \pmod p$.

(3.b). $\gcd(2^{(p-1)/q}-1,p)=\gcd(2^2-1,p)=\gcd(3,p)=1$ (hay 3 không là ước của p).

Như vậy, bằng chứng để chứng minh tính nguyên tố mạnh của số p bao gồm các tham số: q, r, a, q_1, a_1 và k. Và việc chứng minh tính nguyên tố mạnh của p chính là việc thực hiện các đẳng thức nêu trên. Bộ tham số (q,r,a,q_1,a_1,k) cho mỗi số nguyên tố mạnh được ghi trong tệp "ho_so.txt" dưới dạng text.

1.2. Khả năng sinh số nguyên tố mạnh của chương trình

1.2.1. Số nguyên tố mạnh lớn nhất sinh được

Chúng tôi đã sinh được một số nguyên tố mạnh độ dài 2200 bit đó là:

Số nguyên tố mạnh 2200 bit (663 chữ số thập phân)=

1302988093316615905246035664589091920557123416389328384360400937741039798406401668670474461762768627498797800710282781995460099474176058056232465118819265852572401018277569587880619591023217476522085212976423616259462022897624726051726987589329830095135216037678404705350683885082585173921201045884708613765036215583726495163235994876868637350054784865457342786232293446016213960102608893628260662866530044003402771278719309024138177795033415450736910419602261065613457232835874992626306569974318503899453909205292072224567801181322565695912625789033450167281166805505486423173075183736752793733376475590232434467311553352085809953529714588408301287471234338143 03.

Hồ sơ về việc sinh ra số nguyên tố 2200 bit nêu trên như sau:

So nguyen to manh 2200 bits sinh luc Thu Mar 28 10:46:13 2002
P=2(R*PEPIN+1)+1=391f d358 d8ea 3586 840e b2b1 e9d4 7cc2 57b5 a41 eea8 c161 ef4b 74cc c5d9 dd7 b0ad 552b a860 49e6 1053 b 28b9 721c a8e3 2921 6505 328e 95fb 7780 7880 90d3 240 793b 3a31 3d3d 5669 eddc e93e 235a 48be fa84 bada c74d 55d8 7dc9 6193 95cd 639 9311 9bd8 3bc4 901 fce1 e0cf 65e3 f7b0 5ca6 57e 7a9b f849 ebf8 3cd0 e80f f7b6 9db3 39a9 9bb7 2e86 a578 3e2 540b 7e3a 7a86 dd46 df05 a3a7 902b 16ed e0b5 7570 6692 eecb 72a7 1d1c 6fe5 3cab c7b8 b922 a998 3db6 8382 50ef 82a2 9530 7860 f5c4 2e63 c38b 817d a903 47fc bf53 ac51 bc33 b0b5 c147 27f6 2ff3 9b9d 401f e3e6 db3c 5315 f1c4 63ba 5eda c6ff 81d3 aff5 782b c344 ae05 ad67 8910 7ddc ed0e 45c7 4884 fb5c 1c86 b4d9 477a a61b 5c8b 97e4 cbe5 b4
R=1c8e 69ac 6c75 1ac3 c207 5958 74ea be61 abda 520 f754 e0b0 77a5 ba66 e2ec 86eb d856 2a95 5430 a4f3 8829 8005 145c b90e d471 9490 3282 9947 4afd 3bc0 bc40 4869 8120 bc9d 9d18 6a4 556 651c 9463 69c6 618f 382e 55ef a721 4e13 c672 bd31 f602 f908 1b7e 351c dd5 6f9c d4d3 2499 cce4 2bcb 526b 90a0 fcb3 bfc6 ee6 daed 3104 9df8 a5b 9354 ce6f
Bang chung de R*PEPIN+1 nguyen to theo Pocklington la
a=b35b e7b1 f85b 4393 650a 2829 7d13 2b38 e3b5 f5d 8da3 330f 4982 5282 af15 ec97 e83 1c8f 70a6 58bf 57ee c8f9 3cc4 b2e6 6ee5 bb07 6cb1 5c8a 4fe7 65ca 5ebf e688 5fe1 53b7 c130 3c05 3dda 71d7 f3b4 eff8 b645 62a6 858c a24c d9ec fa83 de41 94ac 2684 decc fe0d 4be7 e8d8 3893 65d2 5ef7 9f99 98f4 cbb 63a9 9bcf eb0f 947f f7e3 ace5 979b 7f3b e88e f259 ba21 9bf3 8617 d50e 7dd0 7fb0 79e 1535 96f7 2f43 17b4 ffe2 e117 e59d 7fca bc37 1a9f ead6 f334 cb35 b643 a1c3 fdd0 8b2b e5ed a73f 64d f7c3 a65c 1701 8215 71b2 454 eb21 3bcf bd0b 727b 8035 bc8b b26 48cc 3a0e dd86 3337 57ae 481a 6d8f 276 adad 8164 fa15 aef3 67d2 702d c4c6 6b05 b695 6f31 bedc 1d56 f079 ef85 c202 2991 d041 d814 d7d7 6bb9
Pepin=1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 a19b 41d6 8ed5 2a71 9a6c bccb aaba 312d 843b 88dc 94ff 27ae 647 5e4a 6412 1f48 3f19

bee2 fe2d 1c75 7668 890 513d 3bb5 edc6 ba99 bd5d 16 d2ee c872 94f6 c15f 55b5 1a35 70
MU_k=560; JACOBI=7
Phai kiem tra 47380 so nguyen, co 61 so nguyen to

1.2.2. Một số kết luận thống kê thu được

Bảng 1. (thời gian sinh trung bình số nguyên tố và số nguyên tố mạnh M bit.)

M (bit)	$T_{tổng\ cộng}(M)$	$N_{lớn}(M)$	$t_{lớn}(M)$	$N_{mạnh}(M)$	$t_{mạnh}(M)$
512	1348 giờ	351123	***0.23 phút***	1330	***1.01 giờ***
1024	1431 giờ	56321	***1.52 phút***	134	***10.68 giờ***
1500	785 giờ	8630	***5.5 phút***	10	***78.50 giờ***

Bảng 2. (mật độ thực nghiệm). $\rho_{mạnh}(M)= N_{mạnh}(M)/N_{lớn}(M)$.

M	$N_{lớn}(M)$	$N_{mạnh}(M)$	$\rho_{mạnh}(M)$
128	33310	499	***0.0150***
256	66011	531	***0.0080***
336	175383	1000	***0.0057***
512	351123	1330	***0.0038***
660	246526	683	***0.0028***
1024	56321	134	***0.0024***

Chú thích: Các ký hiệu được ghi trong các bảng trên như sau:
$t_{lớn}(M)$ và $t_{mạnh}(M)$ là thời gian sinh trung bình một số nguyên tố lớn và tương tự một số nguyên tố mạnh độ dài M bit
$T_{tổng\ cộng}(M)$ là tổng số thời gian để thực hiện việc sinh các số nguyên tố mạnh độ dài M bit.
$N_{lớn}(M)$ **và** $N_{mạnh}(M)$ là số các số nguyên tố lớn và tương tự là số các nguyên tố mạnh độ dài M đã sinh được.

PHỤ LỤC 2. **VÍ DỤ VỀ SỐ CÁC SỐ PEPIN, POCKLINGTON VÀ SOPHIE**

Trong phần này chúng tôi đưa ra cho bạn đọc một ví dụ nhỏ về các số Pepin, Pocklington và Sophie để có thể hình dung ra được sự phong phú của các số nguyên tố mạnh trong lớp các số mà chương trình của chúng ta sẽ tìm kiếm, tất nhiên với độ dài bit lớn hơn nhiều.

1. Bảng số lượng các số Pepin $=r2^{16}+1$ với r lẻ và không quá 32 bit

b	N(b)	b	N(b)	b	N(b)	b	N(b)
17	**1**	21	**2**	25	**15**	29	**206**
18	**0**	22	**3**	26	**25**	30	**389**
19	**0**	23	**3**	27	**58**	31	**766**
20	**0**	24	**7**	28	**105**	32	**1480**

Chú thích: b là số bit; N(b) là số các Pepin b bit.

2. Bảng số lượng các số Pocklington $q=R(2^{16}+1)+1$ và số Sophie không quá 32 bit

b	$\mathbf{N_P}$	$\mathbf{N_S}$	**b**	$\mathbf{N_P}$	$\mathbf{N_S}$	**b**	$\mathbf{N_P}$	$\mathbf{N_S}$	**b**	$\mathbf{N_P}$	$\mathbf{N_S}$
17	***0***	**0**	21	***2***	**0**	25	***12***	**1**	29	***231***	**15**
18	***0***	**0**	22	***2***	**0**	26	***32***	**3**	30	***471***	**20**
19	***0***	**0**	23	***5***	**0**	27	***55***	**4**	31	***742***	**46**
20	***1***	**0**	24	***7***	**0**	28	***110***	**8**	32	***1541***	**89**

Chú thích: b là số bit; N_P là số các Pocklington; N_S là số các số Sophie.

3. Bảng tất cả các số Sophie dạng $q=R(2^{16}+1)+1$ và không quá 32 bit

3.1 Bảng tất cả các số Sophie dạng q=R(2^{16}+1)+1 (từ 25 đến 31 bit)

b	**Tất cả các số Sophie từ 25 đến 31 bit** (viết dưới dạng thập phân)
25	29229503;
26	3 46138049; 48104159; 56755043;
27	83494139; 89785691; 91751801; 122816339;
28	153094433; 156240209; 166070759; 200281073; 221515061; 223481171; 231738833; 249433823;
29	296227241;304484903;339088439;343413881;355603763; 403970069;425990501;430315943;489299243;512892563; 518004449;526262111;530194331;530587553;534519773;
30	541597769;552214763;571089419;584852189;612377729; 659957591;697706903;728378219;823537943;854602481; 936785879;958806311;978074189;978467411;997735289; 998128511;1003633619;1035484601;1064583029;1066942361;
31	1096434011;1126318883;1182942851;1196705621;1204176839; 1267485581;1283214461;1305234893;1324895993;1332760433; 1360285973;1365791081;1389384401;1401574283;1402753949; 1421235383;1482184793;1503812003;1560042749;1568300411; 1611948053;1623351491;1631215931;1653236363;1654809251; 1697670449;1718117993;1743677423;1745643533;1757440193; 1774741961;1784179289;1809738719;1812491273;1819962491; 1825860821;1839230369;1991407283;1994553059;2005170053; 2014214159;2026404041;2063760131;2072017793;2093645003; 2148696083;

3.2 Bảng tất cả các số Sophie dạng q=R(2^{16}+1)+1 (32 bit)

b	**Tất cả các số Sophie 32 bit** (viết dưới dạng thập phân)
32	2148696083;2151841859;2164031741;2173469069;2203353941; 2236777811;2286323783;2297333999;2299300109;2307164549; 2329578203;2339015531;2347273193;2415300599;2439680363; 2470351679;2513606099;2535626531;2549389301;2554894409; 2563152071;2635504919;2665389791;2668928789;2677579673; 2726732423;2777851283;2791614053;2863966901;2864360123; 2891885663;2896997549;2911546763;2932780751;2956767293; 2971709729;2976428393;2998055603;3029120141;3068049119; 3075913559;3094394993;3103439099;3164781731;3186802163; 3192307271;3292578881;3331901081;3375155501;3407006483; 3522613751;3530871413;3532051079;3544634183;3620526029; 3620919251;3623278583;3626424359;3642546461;3738492629; 3742424849;3746750291;3753041843;3813598031;3817137029; 3841516793;3890276321;3912296753;3916228973;3937462961; 3942968069;3959483393;3970886831;3976785161;3978358049; 4003917479;4031836241;4045599011;4066832999;4089246653; 4115985749;4141938401;4157667281;4178901269;4222942133; 4234738793;4254399893;4275633881;4287823763;

Bảng số lượng số nguyên tố Pocklington và số nguyên tố Sophie với nhân là 30 số nguyên tố Pepin liên tiếp đầu tiên dạng $r2^{16}+1$ với r>1 lẻ (các số từ 21 đến 26 bit).

pepin \ bit	23	24	25	26	27	28	29	30	31	32
1376257				1/0		5/0	10/0	16/1	34/1	60/2
1769473	1/0		1/0	2/0		8/0	4/0	16/0	28/1	49/4
2424833		***1/0***		***1/0***	***1/0***	***2/0***	***7/0***	***8/0***	***21/2***	***43/4***
3604481			***1/0***	***1/0***		***1/1***	***4/1***	***8/2***	***14/2***	***26/1***
pepin \ bit	23	24	25	26	27	28	29	30	31	32
3735553			***1/0***	***1/0***	***2/1***	***2/0***	***6/0***	***7/0***	***16/0***	***25/3***

5308417					***1/0***	***2/0***	***2/0***	***6/0***	***16/1***	***19/2***
6750209		***1/1***			***1/0***	***1/0***	***2/0***	***4/0***	***11/2***	***18/0***
7667713					***1/0***	***1/0***	***2/0***		***9/0***	***14/0***
8716289						1/0	2/0	1/0	4/0	8/0
9502721			1/0	1/0	1/0	1/0	1/0	2/0	7/0	11/2
10027009						1/0		4/0	3/1	7/0
11468801							1/0	2/0	6/0	13/0
11599873					2/0	2/1	1/0	2/0	3/0	10/0
13565953						1/0	1/0	1/0	5/0	10/0
16580609								4/0	3/0	6/0
17367041							***2/0***	***3/0***	***5/0***	***5/0***
19070977								***2/0***		***5/0***
20119553							***1/0***	***1/0***		***2/0***
20512769							***3/0***	***2/0***	***3/0***	***6/0***
23789569					***1/0***		***1/0***		***3/0***	***6/0***
24576001								***3/1***		***4/0***
25231361							***1/0***	***1/0***	***3/0***	***2/0***
26017793						***1/0***		***2/0***	***2/0***	***2/0***
26411009								***1/1***	***2/0***	***4/0***
27328513								***1/0***	***1/0***	***1/0***
27590657								***1/0***	***1/0***	***1/0***
29294593							***1/0***		***4/0***	***3/0***
29687809								***1/0***	***1/0***	***4/0***
31916033								***1/0***		***4/0***
32440321								***1/0***	***2/0***	***2/0***

Bảng các số Sophie đã được liệt kê trong bảng trên

Pepin	bit	Các số Sophie
1376257	30	922092191;
	31	1904739689;
	32	3118598363; 4258139159;
1769473	31	1344799481;
	32	2353399091; 2406483281; 3139045103; 4211345741;
2424833	31	1125122513; 1998062393;
	32	3525707183; 3758491151; 3962177123; 4194961091;
3604481	28	223477823;
	29	504627341;
	30	720896201; 764149973;
	31	1369702781; 2083390019;
	32	2559181511;
3735553	27	127008803;
	32	2502820511; 2614887101; 3870032909;
5308417	31	2091516299;
	32	2441871821; 3110732363;
6750209	24	13500419;
	31	1552548071; 2038563119;
9502721	32	3325952351; 3725066633;
10027009	31	1784807603;
11599873	28	185597969;
24576001	30	688128029;
26411009	30	845152289;