# Bab 14

# Matematika VII - Penguraian Bilangan Bulat

Mengalikan dua bilangan yang sangat besar relatif merupakan sesuatu yang mudah. Sebaliknya, menguraikan suatu bilangan yang sangat besar untuk mendapatkan faktor-faktornya, secara umum merupakan sesuatu yang sulit. Untuk bilangan $n$ yang tidak terlalu besar, kita dapat mencoba membagi $n$ dengan setiap bilangan prima $\leq \sqrt{n}$, akan tetapi ini tidak praktis dan akan memakan waktu yang terlalu lama jika $n$ sangat besar. Algoritma untuk test bilangan prima jika gagal hanya menyatakan bahwa bilangan adalah komposit, tetapi tidak menolong mencarikan faktor-faktornya. Keamanan algoritma RSA (lihat bagian 16.1), satu algoritma kriptografi *public key*, didasarkan pada pengamatan bahwa belum ada algoritma yang efisien untuk menguraikan bilangan yang sangat besar. Meskipun secara umum penguraian bilangan besar sangat sulit, ada beberapa kondisi yang dapat menyebabkan suatu bilangan besar mudah untuk diuraikan. Juga, kemajuan dibidang teknologi komputer dan teknik penguraian membuat jangkauan bilangan yang dapat diuraikan semakin besar, dan beberapa prediksi dimasa lalu ternyata jauh meleset. Sebagai contoh, tahun 1976, Martin Gardner menulis dalam Scientific American bahwa kunci RSA sebesar 129 digit akan aman untuk sekitar 40 *quadrillion* tahun. Ternyata kunci tersebut dapat diuraikan menggunakan metode *quadratic sieve* tahun 1994.

Kita akan bahas beberapa algoritma penguraian dengan harapan bahwa ini akan membantu kita memahami betapa sukarnya penguraian, dan menolong kita untuk mewaspadai kondisi yang dapat membuat suatu bilangan besar mudah untuk diuraikan.

## 14.1 Metode Rho

Metode Rho (*Rho method*) adalah algoritma Las Vegas untuk menemukan faktor suatu bilangan besar secara probabilistik. Algoritma Las Vegas adalah algoritma probabilistik yang jika sukses maka hasilnya dijamin benar. Untuk mencari faktor dari $n$, metode Rho menggunakan pemetaan $f(x)$ dari $\mathbf{Z}/n\mathbf{Z}$ ke $\mathbf{Z}/n\mathbf{Z}$, contohnya *polynomial* $f(x) = x^2 + 1$. Kita pilih nilai awal untuk $x = x_0$ (contohnya $x_0 = 1$ atau $x_0 = 2$), lalu kalkulasi secara bertahap $x_1, x_2, \ldots, x_m$ sebagai berikut

$$\begin{aligned} x_1 &= f(x_0) \\ x_2 &= f(x_1) \\ \ldots & \\ x_m &= f(x_{m-1}), \end{aligned}$$

jadi

$$x_{j+1} = f(x_j), j = 0, 1, 2, \ldots, m-1.$$

Langkah berikutnya adalah mencari pasangan $x_j, x_k$ dimana $x_j$ dan $x_k$ berada dalam *congruence class* yang berbeda modulo $n$, tetapi berada dalam *congruence class* yang sama modulo $d$ untuk suatu $d$ yang membagi $n$. Ini dapat dilakukan menggunakan gcd, jadi jika

$$d = \gcd(x_k - x_j, n),$$

maka $d$ merupakan faktor dari $n$. Sebagai contoh, jika $n = 119$, $f(x) = x^2 + 1$ dan $x_0 = 1$, maka

$$x_1 = 2, x_2 = 5, x_3 = 26, \ldots,$$

dan kita dapatkan

$$d = \gcd(x_3 - x_2, 119) = \gcd(21, 119) = 7.$$

Jadi 7 merupakan faktor dari 119. Tentunya 119 bukan suatu bilangan yang besar, dan dapat diuraikan dengan mencoba membagi 119 dengan setiap bilangan prima $\leq \sqrt{119}$, contoh diatas hanya menjelaskan bagaimana faktor dicari.

Untuk metode Rho, pemetaan $f(x)$ sebaiknya adalah sesuatu pemetaan yang "acak," jadi bukan *polynomial* linear dan sebaiknya bukan pemetaan yang *bijective*. Untuk mendapatkan gambaran mengenai tingkat kesuksesan metode Rho dan waktu yang dibutuhkan untuk menemukan faktor, kita butuh konsep *"average map"* yang merepresentasikan pemetaan acak. Metode Rho mencari indeks pertama $k$ dimana terdapat indeks $j$ dengan $j < k$ dan

$$x_j \equiv x_k \pmod{d}.$$

Kita pelajari ini dengan memperlakukan pemetaan $f(x)$ sebagai pemetaan dari $\mathbf{Z}/d\mathbf{Z}$ ke $\mathbf{Z}/d\mathbf{Z}$ dan mencari tahu probabilitas bahwa $d$ tidak ditemukan setelah $x_k$ dikalkulasi dan diperiksa terhadap setiap $x_i$ dengan $0 \leq i < k$.

**Teorema 98** *Jika $f$ adalah suatu pemetaan dari $\mathbf{Z}/d\mathbf{Z}$ ke $\mathbf{Z}/d\mathbf{Z}$, $x_0 \in \mathbf{Z}/d\mathbf{Z}$, $x_{j+1} = f(x_j)$ untuk $j = 0, 1, 2, \ldots$, dan indeks $k$ merupakan bilangan bulat positif dan $\lambda$ adalah bilangan nyata positif dengan*

$$k = 1 + \sqrt{2\lambda d},$$

*maka proporsi pasangan $f, x_0$ yang menghasilkan*

$$x_0, x_1, x_2, \ldots, x_k$$

*yang berbeda modulo $d$ ($x_i \not\equiv x_j \pmod{d}$ untuk $i \neq j$) adalah $< e^{-\lambda}$, dimana semua kemungkinan pemetaan $f$ dari $\mathbf{Z}/d\mathbf{Z}$ ke $\mathbf{Z}/d\mathbf{Z}$ dan semua kemungkinan $x_0 \in \mathbf{Z}/d\mathbf{Z}$ diperhitungkan.*

Mari kita buktikan teorema 98. Ada $d$ kemungkinan untuk $x_0$ dan $d^d$ kemungkinan untuk pemetaan $f$ dari $\mathbf{Z}/d\mathbf{Z}$ ke $\mathbf{Z}/d\mathbf{Z}$, jadi total ada $d^{d+1}$ kemungkinan pasangan $f, x_0$. Dari semua kemungkinan pasangan $f, x_0$ kita pilih pasangan-pasangan yang menghasilkan $x_0, x_1, x_2, \ldots, x_k$ yang berbeda modulo $d$. Ada $d$ kemungkinan untuk $x_0$, $d - 1$ kemungkinan untuk $f(x_0) = x_1$ untuk setiap $x_0$ karena $x_1$ harus berbeda dari $x_0$, dan seterusnya sampai dengan $d - k$ kemungkinan untuk $f(x_{k-1}) = x_k$ untuk setiap $x_{k-1}$. Sisanya untuk $d-k$ elemen $x_{k+1}, \ldots, x_d$ tidak ada syarat untuk $f$ selain harus menghasilkan elemen dalam $\mathbf{Z}/d\mathbf{Z}$, jadi ada $d^{d-k}$ kemungkinan hasil $f$ untuk $x_{k+1}, \ldots, x_d$. Jadi total jumlah pasangan $f, x_0$ yang menghasilkan $x_0, x_1, x_2, \ldots, x_k$ yang berbeda modulo $d$ ada

$$d^{d-k} \prod_{j=0}^{k} (d - j),$$

jadi proporsi pasangan $f, x_0$ yang menghasilkan $x_0, x_1, x_2, \ldots, x_k$ yang berbeda modulo $d$ adalah

$$\begin{aligned} \frac{d^{d-k} \prod_{j=0}^{k} (d-j)}{d^{d+1}} &= d^{-k-1} \prod_{j=0}^{k} (d - j) \\ &= \prod_{j=1}^{k} (1 - \frac{j}{d}). \end{aligned}$$

Berdasarkan fakta bahwa $\log(1 - x) < -x$ untuk $0 < x < 1$, maka

$$\log \left( \prod_{j=1}^{k} (1 - \frac{j}{d}) \right) < \sum_{j=1}^{k} -\frac{j}{d}$$

$$\begin{aligned} &= \frac{-k(k+1)}{2d} \\ &< \frac{-k^2}{2d} \\ &< \frac{-(\sqrt{2\lambda d})^2}{2d} \\ &= -\lambda. \end{aligned}$$

Jadi proporsi pasangan $f, x_0$ yang menghasilkan $x_0, x_1, x_2, \ldots, x_k$ yang berbeda modulo $d$ adalah

$$\prod_{j=1}^{k}(1 - \frac{j}{d}) < e^{-\lambda}.$$

Selesailah pembuktian teorema 98. Teorema 98 memberi gambaran mengenai tingkat kesuksesan metode Rho: jika $d$ adalah faktor dari $n$, probabilitas bahwa metode Rho tidak menemukan $d$ setelah $x_k$ dikalkulasi adalah $< e^{-\lambda}$ jika $f$ direratakan untuk semua kemungkinan, dengan kata lain jika $f$ merupakan suatu *"average map."*

Dalam implementasi metode Rho, jika untuk setiap $x_k$ kita periksa setiap $x_j$ dengan $j < k$, akan sangat tidak efisien jika kita sedang menguraikan $n$ yang besar. Kita akan modifikasi metode Rho sehingga untuk setiap $x_k$ kita hanya periksa satu $x_j$: jika $x_k$ merupakan bilangan bulat dengan $h$ bit ($2^{h-1} \leq x_k < 2^h$), maka kita hanya periksa $x_j$ dimana $j = 2^{h-1} - 1$. Sebagai contoh, untuk $x_2$ dan $x_3$, kita hanya periksa $x_1$, dan untuk $x_4, x_5, x_6$ dan $x_7$, kita hanya periksa $x_3$. Meskipun $x_k$ yang ditemukan bukanlah yang pertama yang menghasilkan

$$x_k \equiv x_j \pmod{d}$$

untuk $j < k$, kita dapat tunjukkan bahwa jika metode Rho tanpa modifikasi menghasilkan

$$x_{k_0} \equiv x_{j_0} \pmod{d}$$

dengan $j_0 < k_0$, maka metode Rho yang telah dimodifikasi menghasilkan

$$x_k \equiv x_j \pmod{d}$$

dengan $k - j = k_0 - j_0$ dan $k \geq k_0$. Ini karena jika $k = k_0 + m$ untuk suatu $m \geq 0$, maka jika kita aplikasikan *polynomial* $f$ ke dua sisi dari $x_{k_0} \equiv x_{j_0} \pmod{d}$ kita akan dapatkan $x_k \equiv x_j \pmod{d}$. Sebagai contoh, jika metode Rho tanpa modifikasi menemukan

$$x_3 \equiv x_2 \pmod{d}$$

maka metode Rho dengan modifikasi akan menemukan

$$x_4 \equiv x_3 \pmod{d}.$$

Jika $k_0$ merupakan bilangan dengan $h$ bit, maka $j = 2^h - 1$ (contoh diatas $k_0 = 3$ adalah bilangan dengan $h = 2$ bit, jadi $j = 2^2 - 1 = 3$) dan $k = j + (k_0 - j_0)$ (contoh diatas $k = 3 + (3 - 2) = 4$). Perhatikan bahwa $k$ adalah bilangan dengan $h + 1$ bit, jadi

$$k < 2^{h+1} = 4(2^{h-1}) \leq 4k_0.$$

**Teorema 99** *Jika $n$ adalah bilangan komposit ganjil dan $d$ adalah faktor non-trivial dari $n$ ($d|n$ dan $1 < d < n$) dengan $d < \sqrt{n}$, maka metode Rho yang telah dimodifikasi akan menemukan faktor $d$ dengan probabilitas $1 - e^{-\lambda}$ menggunakan $C\sqrt{\lambda}\sqrt[4]{n}\log^3 n$ operasi bit, dimana $C$ adalah suatu konstan.*

Untuk membuktikan teorema 99, kita umpamakan bahwa $\gcd(x - y, n)$ (dengan $n > x - y$) dapat dikalkulasi menggunakan algoritma Euclid dengan $C_1 \log^3 n$ operasi bit, dimana $C_1$ adalah suatu konstan (ini berdasarkan pengamatan bahwa algoritma Euclid melakukan $O(\log(n))$ operasi pembagian dan setiap pembagian memerlukan $O(\log^2 n)$ operasi bit, jadi secara total diperlukan $O(\log^3 n)$ operasi bit). Kita juga umpamakan bahwa komputasi $f(x)$ modulo $n$ memerlukan $C_2 \log^2 n$ operasi bit, dimana $C_2$ adalah suatu konstan (karena komputasi *polynomial* memerlukan konstan perkalian dengan setiap perkalian membutuhkan $O(\log^2 n)$ operasi bit, dan operasi modulo atau pembagian juga membutuhkan $O(\log^2 n)$ operasi bit). Jika $k_0$ merupakan indeks pertama yang menghasilkan

$$x_{k_0} \equiv x_{j_0} \pmod{d}$$

untuk suatu $j_0$ dengan $j_0 < k_0$, maka metode Rho yang telah dimodifikasi akan menemukan $d$ saat memeriksa $x_k$ dimana $k < 4k_0$. Sebetulnya ada kemungkinan bahwa $\gcd(x_k - x_j, n)$ menghasilkan sesuatu yang lebih besar dari $d$, dengan kata lain $\gcd((x_k - x_j)/d, n/d) > 1$, tetapi kemungkinan ini sangat kecil dan dapat diperhitungkan dengan membuat $C$ cukup besar. Jadi jika $d$ ditemukan saat memeriksa $x_k$, maka penemuan $d$ memerlukan tidak lebih dari $4k_0(C_1 \log^3 n + C_2 \log^2 n)$ operasi bit. Jika $k_0 \leq 1 + \sqrt{2\lambda d}$ maka untuk menemukan $d$ diperlukan

$$\begin{aligned} &< 4(1 + \sqrt{2\lambda d})(C_1 \log^3 n + C_2 \log^2 n) \\ &< 4(1 + \sqrt{2}\sqrt{\lambda}\sqrt[4]{n})(C_1 \log^3 n + C_2 \log^2 n) \end{aligned}$$

operasi bit. Jika kita buat

$$C = 4\sqrt{2}(\frac{(C_1 + C_2)}{\sqrt{2}\sqrt{\lambda}\sqrt[4]{n}} + (C_1 + C_2))$$

kita dapatkan bahwa $d$ ditemukan dengan menggunakan tidak lebih dari

$$C\sqrt{\lambda}\sqrt[4]{n}\log^3 n$$

operasi bit dengan probabilitas $1 - e^{-\lambda}$ (menurut teorema 98). Selesailah pembuktian teorema 99. Tentunya teorema 99 hanya berlaku jika $f$ yang digunakan merupakan suatu *"average map."* Dalam prakteknya, beberapa *polynomial* yang sangat sederhana bersifat *"average map,"* termasuk $f(x) = x^2 + 1$.

Teorema 98 memberikan gambaran mengenai kompleksitas algoritma untuk metode Rho. Dalam bidang teori kompleksitas, pengukuran kompleksitas algoritma biasanya didasarkan pada besarnya input dalam ukuran bit. Jika $r$ merupakan banyaknya bit dalam $n$, maka kompleksitas algoritma untuk metode Rho diperkirakan sekitar

$$O(e^{C\sqrt{r}})$$

dimana $C = \frac{1}{4} log 2$. Jadi untuk $r$ yang sangat besar, metode Rho lebih lambat dari algoritma dengan kompleksitas *polynomial-time*[1], meskipun tidak selambat algoritma dengan kompleksitas *exponential-time.*

## 14.2 Fermat Factorization

Jika suatu bilangan komposit ganjil $n$ merupakan produk dari dua bilangan yang berdekatan, jadi $n = pq$ dengan $p \geq q > 0$, $p - q$ tidak terlalu besar, dan $p$ dan $q$ keduanya ganjil, maka $p$ dan $q$ dapat dicari dengan mudah menggunakan *Fermat factorization.* Teknik ini didasarkan pada fakta bahwa jika

$$s = \tfrac{p+q}{2}, \quad t = \tfrac{p-q}{2},$$

jadi

$$p = s + t, \quad q = s - t,$$

maka

$$\begin{aligned} n &= pq \\ &= (s+t)(s-t) \\ &= s^2 - t^2. \end{aligned}$$

Sebelum kita lanjutkan pembahasan *Fermat factorization*, kita buktikan dahulu teorema berikut.

**Teorema 100** *Jika $p \geq q \geq 0$ dimana $p$ dan $q$ adalah bilangan bulat, maka*

$$p^2 + q^2 \geq 2pq$$

[1]Istilah *super-polynomial-time* kerap digunakan untuk kompleksitas yang lebih lambat dari *polynomial-time.*

Kita buktikan teorema 100 menggunakan induksi pada $p$ dengan *base case* $p = q$ (jadi sebenarnya induksi adalah pada $p - q$). Untuk $p = q$ kita dapatkan

$$\begin{aligned} p^2 + q^2 &= 2q^2 \\ &= 2pq, \end{aligned}$$

jadi $p^2 + q^2 \geq 2pq$. Untuk langkah induksi kita dapatkan

$$\begin{aligned} (p+1)^2 + q^2 &= p^2 + 2p + 1 + q^2 \\ &\geq 2pq + 2p + 1 \qquad \text{(menggunakan hipotesis induksi)} \\ &> 2pq + 2q \qquad \text{(karena } p \geq q\text{)} \\ &= 2(p+1)q. \end{aligned}$$

Jadi $(p+1)^2+q^2 \geq 2(p+1)q$ dan selesailah pembuktian teorema 100 menggunakan induksi. Teorema 100 kita gunakan untuk menunjukkan bahwa $(p+q)/2 \geq \sqrt{n}$ dengan menunjukkan bahwa $(p+q)^2/4 \geq n$:

$$\begin{aligned} \frac{(p+q)^2}{4} &= \frac{p^2 + 2pq + q^2}{4} \\ &\geq \frac{4pq}{4} \qquad \text{(menggunakan teorema 100)} \\ &= n. \end{aligned}$$

Jadi jika $p$ dan $q$ berdekatan, maka $t = (p - q)/2$ merupakan bilangan yang kecil dan $s = (p + q)/2 \geq \sqrt{n}$ dan perbedaan antara $s$ dan $\sqrt{n}$ tidak terlalu besar. Jadi kita dapat mencari $s$ mulai dari $s = \lfloor\sqrt{n}\rfloor + 1$, lalu $s = \lfloor\sqrt{n}\rfloor + 2$, dan seterusnya hingga kita temukan nilai $s^2 - n$ yang merupakan *perfect square* ($s^2 - n$ merupakan kuadrat) yang kita jadikan nilai untuk $t^2$ (karena $n = s^2 - t^2$). Sebagai contoh kita coba uraikan 200819:

| | | |
|---|---|---|
| $s = \lfloor\sqrt{200819}\rfloor + 1 = 449$ | $449^2 - 200819 = 782$ | bukan kuadrat; |
| $s = \lfloor\sqrt{200819}\rfloor + 2 = 450$ | $450^2 - 200819 = 1681 = 41^2$ | $t = 41$. |

Kita dapatkan $p = s + t = 450 + 41 = 491$ dan $q = s - t = 450 - 41 = 409$, jadi $n = pq = 491 \cdot 409 = 200819$. Jadi jika suatu bilangan komposit ganjil $n$ merupakan produk dari dua bilangan ganjil $p$ dan $q$ yang berdekatan, maka $p$ dan $q$ dapat ditemukan dengan mudah, termasuk jika $p$ dan $q$ merupakan bilangan prima ganjil yang berdekatan. Jika $n$ bukan merupakan produk dari dua bilangan ganjil yang berdekatan, maka *Fermat factorization* akan mencoba banyak nilai $s$ sebelum menemukan nilai $s$ yang mendapatkan faktor, jadi bukan merupakan suatu algoritma yang efisien.

## 14.3 Metode Dixon

*Fermat factorization* secara umum bukan merupakan algoritma yang efisien untuk menguraikan bilangan yang sangat besar, tetapi lebih berupa konsep bahwa suatu bilangan komposit ganjil merupakan perbedaan dari kuadrat (*difference of squares*). Sekitar tahun 1920an, Maurice Kraitchik menyarankan ide bahwa yang dicari adalah perbedaan kuadrat modulo bilangan yang diuraikan. Jadi kita bukan mencari $s$ dan $t$ yang menghasilkan $n = s^2 - t^2$, tetapi cari $s$ dan $t$ yang menghasilkan

$$s^2 \equiv t^2 \pmod{n}.$$

Ada kalanya ini akan menghasilkan

$$s \equiv \pm t \pmod{n}$$

yang membuat kita harus mencari pasangan $s$ dan $t$ yang lain. Akan tetapi jika

$$s \not\equiv \pm t \pmod{n}$$

maka kita dapatkan faktor dari $n$ dengan mengkalkulasi

$$\gcd(s+t, n) \qquad \text{atau} \qquad \gcd(s-t, n).$$

Ini karena $n$ membagi $s^2 - t^2 = (s+t)(s-t)$ tetapi $n$ tidak membagi $s+t$ atau $s-t$ (karena $s \not\equiv \pm t \pmod{n}$), jadi $a = \gcd(s+t, n)$ merupakan faktor *non-trivial* dari $n$ ($a|n$ dan $1 < a < n$). Kita juga dapatkan $b = n/a$ membagi $\gcd(s-t, n)$. Sebagai contoh, dengan $n = 4633$, $s = 118$ dan $t = 5$, kita temukan

$$118^2 \equiv 5^2 \pmod{4633} \qquad \text{dan} \qquad 118 \not\equiv \pm 5 \pmod{4633},$$

jadi

$$\gcd(118+5, 4633) = 41 \qquad \text{dan} \qquad \gcd(118-5, 4633) = 113$$

merupakan faktor dari 4633.

Ide inilah yang menjadi dasar dari berbagai metode modern untuk menguraikan bilangan yang besar, termasuk metode Dixon, metode *continued fraction*, metode *quadratic sieve* dan metode *number field sieve.*

Contoh diatas tidak menunjukkan bagaimana kita menemukan $s = 118$ yang jika dikuadratkan ($s^2$) mempunyai *least non-negative residue*[2] modulo 4633 yaitu 25 yang juga merupakan kuadrat ($t^2 = 5^2$). Jika $n$ merupakan bilangan yang sangat besar, maka probabilitas bahwa suatu bilangan yang dipilih secara acak jika dikuadratkan mempunyai *least non-negative residue* modulo $n$ yang juga merupakan kuadrat, adalah sangat kecil.

Metode Dixon termasuk metode yang mencoba menemukan $s$ dan $t$ secara sistematis menggunakan *factor base.* Ide yang digunakan untuk mendapatkan $s$ adalah untuk memilih beberapa $b_i$ dimana $b_i^2 \bmod n$ merupakan

[2]Di bagian ini kita juga akan menggunakan konsep *least absolute residue* jadi kita harus bedakan kedua konsep.

produk dari beberapa pemangkatan bilangan prima kecil ($p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_m^{\alpha_m}$ dimana $p_1, p_2, \ldots, p_m$ semua merupakan bilangan prima kecil), dan produk dari $b_i$ mod $n$ menghasilkan $s$. Sebelum membahas metode secara rinci, kita perlu definisikan dahulu konsep *least absolute residue*. Suatu bilangan $b$ merupakan *least absolute residue* dari $a$ modulo $n$, dimana $n$ merupakan bilangan ganjil, jika:

$$a \equiv b \pmod{n} \qquad \text{dan} \qquad -(n-1)/2 \leq b \leq (n-1)/2.$$

Sekarang kita definisikan konsep *factor base*:

**Definisi 47 (Factor Base)** *Suatu factor base merupakan himpunan bilangan prima yang berbeda $B = \{p_1, p_2, \ldots, p_h\}$, kecuali $p_1$ dapat berupa $-1$.*

Kuadrat dari bilangan bulat $b$, $b^2$ merupakan *B-number* modulo $n$ jika *least absolute residue* dari $b^2$ mod $n$ dapat diuraikan menjadi produk dari elemen-elemen $B$.

**Definisi 48 (B-number)**

$$b^2 \equiv t \pmod{n}$$

*merupakan B-number jika $t$ adalah least absolute residue modulo $n$, dan*

$$t = \prod_{i=1}^{h} p_i^{\alpha_i}$$

*dimana $B = \{p_1, p_2, \ldots, p_h\}$ adalah factor base dan $\alpha_i \geq 0$ untuk setiap $i$.*

Sebagai contoh, dengan $n = 4633$ dan $B = \{-1, 2, 3\}$, maka $67^2$, $68^2$ dan $69^2$ masing-masing merupakan *B-number* karena

$$\begin{array}{rcrlcl} 67^2 & \equiv & -144 & \pmod{4633} & = & -1 \cdot 2^4 \cdot 3^2; \\ 68^2 & \equiv & -9 & \pmod{4633} & = & -1 \cdot 3^2; \\ 69^2 & \equiv & 128 & \pmod{4633} & = & 2^7. \end{array}$$

Untuk menentukan apakah suatu kuadrat merupakan *B-number* tentunya perlu penguraian, meskipun setiap faktor harus merupakan pemangkatan dari bilangan dalam *factor base*. Penguraian ini dapat dilakukan dengan *trial division* (mencoba membagi) menggunakan elemen-elemen *factor base*. Ada cara untuk mempercepat proses ini, misalnya menggunakan metode Pollard-Strassen, akan tetapi kita tidak akan bahas cara-cara untuk mempercepat proses ini.

Jika untuk $1 \leq j \leq m$, setiap $t_j$ merupakan *least absolute residue* dari $b_j^2$ mod $n$ dimana $b_j^2$ adalah *B-number*, dan produk dari semua $t_j$,

$$\prod_{j=1}^{m} t_j = \prod_{j=1}^{m} \prod_{i=1}^{h} p_{i,j}^{\alpha_{i,j}}$$

mempunyai setiap $\alpha_{i,j}$ berupa bilangan genap, maka produk semua $t_j$ merupakan kuadrat dari

$$t = \prod_{i=1}^{h} p_i^{\gamma_i}$$

dimana $\gamma_i = \frac{1}{2}\sum_{j=1}^{m} \alpha_{i,j}$. Kita buat juga $s$ sebagai produk dari semua $b_j$:

$$s = \prod_{j=1}^{m} b_j$$

dan kita dapatkan $s^2 \equiv t^2 \pmod{n}$. Jika $s \not\equiv \pm t \pmod{n}$ maka kita dapatkan faktor *non-trivial* dari $n$ menggunakan $\gcd(s+t, n)$ atau $\gcd(s-t, n)$ seperti diawal bagian ini. Jika $s \equiv \pm t \pmod{n}$ maka kita harus cari produk dari kumpulan lain. Bagaimana kita mencari kumpulan $b_i$ untuk *factor base*? Tiga cara untuk mencari *factor base* akan dibahas dalam buku ini:

1. tentukan *factor base* sebagai beberapa bilangan prima pertama dan $-1$, dan kemudian cari beberapa $b_i$ secara acak, dimana $b_i^2$ merupakan *B-number*;

2. cari beberapa $b_i$ yang mempunyai *least absolute residue* dari $b_i^2 \bmod n$ yang kecil, dan kemudian cari *factor base* sehingga setiap $b_i^2$ merupakan *B-number*; atau

3. tentukan *factor base* sebagai semua bilangan prima $p \leq P$ dimana $\left(\frac{n}{p}\right) = 1$ jika $p$ ganjil, dan $P$ adalah suatu batas yang dipilih secara optimal.

Metode Dixon menggunakan cara pertama dalam menentukan *factor base*; metode *continued fraction* (lihat bagian 14.4) menggunakan cara kedua; dan metode *quadratic sieve* (lihat bagian 14.5) menggunakan cara ketiga.

Setelah *factor base* dan beberapa $b_i$ terpilih, kita pilih *subset* dari kumpulan $b_i$ yang jika dikalikan menghasilkan produk dari pemangkatan genap elemen-elemen *factor base* (setiap elemen dipangkatkan dengan suatu bilangan genap). Karena kita hanya perlu fokus pada kegenapan pangkat elemen *factor base*, kita gunakan konsep ruang vektor $\mathbf{F}_2^h$. Untuk suatu bilangan komposit ganjil $n$, *factor base* $B$ dengan $h$ elemen, suatu *B-number* $b_i^2$, dengan $t_i = \prod_{j=1}^{h} p_j^{\alpha_j}$ sebagai *least absolute residue* dari $b_i \bmod n$, maka vektor $\overrightarrow{\epsilon_i}$ didefinisikan sebagai berikut

**Definisi 49**

$$\overrightarrow{\epsilon_i} = (\epsilon_{i,1}, \epsilon_{i,2}, \ldots, \epsilon_{i,h})$$

*dengan*

$$\epsilon_{i,j} = \begin{cases} 0 & \textit{jika } \alpha_j \textit{ genap} \\ 1 & \textit{jika } \alpha_j \textit{ ganjil} \end{cases}$$

*untuk* $1 \leq j \leq h$.

Sebagai contoh, dengan $n = 4633$ dan $B = \{-1, 2, 3\}$, kita dapatkan:

$$\begin{array}{r|c|l} b_1 = 67 & t_1 = -1 \cdot 2^4 \cdot 3^2 & \overrightarrow{\epsilon_1} = (1, 0, 0), \\ b_2 = 68 & t_2 = -1 \cdot 3^2 & \overrightarrow{\epsilon_2} = (1, 0, 0), \\ b_3 = 69 & t_3 = 2^7 & \overrightarrow{\epsilon_3} = (0, 1, 0). \end{array}$$

Jadi mencari subset dari berbagai $b_i$ yang menghasilkan produk yang diinginkan (setiap elemen $p_i \in B$ dipangkatkan dengan bilangan genap) sama dengan mencari kombinasi linear dari berbagai $\overrightarrow{\epsilon_i}$ yang menghasilkan $(0, 0, \ldots, 0)$. Ini dapat dilakukan, jika terdapat *linear dependence* diantara berbagai $\overrightarrow{\epsilon_i}$, menggunakan teknik *row reduction* dari aljabar linear. Jika banyaknya $b_i$ melebihi $h$ $(> h)$ maka dijamin terdapat *linear dependence* diantara berbagai $\overrightarrow{\epsilon_i}$.

Tingkat kesuksesan dari metode Dixon sangat tergantung pada mudahnya mencari berbagai $b_i$ dengan *least absolute residue* dari $b_i^2 \bmod n$ yang dapat diuraikan sebagai produk dari pemangkatan elemen-elemen *factor base.* Perlu diperhatikan bahwa $b_i$ yang menghasilkan $\overrightarrow{\epsilon_i} = (0, 0, \ldots, 0)$ tidak ada gunanya karena menghasilkan *trivial congruence* $s^2 \equiv s^2 \pmod{n}$, jadi $b_i < \sqrt{n/2}$ tidak ada gunanya. Strategi yang kerap digunakan adalah mencari $b_i$ yang dekat dengan $\sqrt{kn}$ untuk berbagai kelipatan $k$ yang kecil. Kompleksitas metode Dixon, jika diimplementasikan dengan optimal, diestimasikan adalah

$$O(e^{C\sqrt{log\, n\, log\, log\, n}}).$$

Karena terlalu rumit, kita tidak akan bahas bagaimana estimasi kompleksitas diatas didapatkan. Untuk yang ingin mengetahui bagaimana estimasi didapatkan, dan juga untuk mempelajari metode Dixon lebih lanjut, dipersilahkan untuk membaca [dix81]. Seperti halnya dengan metode Rho, metode Dixon mempunyai kompleksitas *super-polynomial-time* meskipun tidak seburuk *exponential-time*, akan tetapi kompleksitas metode Dixon lebih baik dibandingkan dengan kompleksitas metode Rho.

## 14.4 Metode Continued Fraction

Metode Dixon efektif jika terdapat cara yang efisien untuk mencari $b_i$ antara 1 dan $n$ dengan *least absolute residue* dari $b_i^2 \bmod n$ berupa produk dari bilangan-bilangan prima yang kecil. Tingkat kesuksesan akan semakin besar jika $b_i^2 \bmod n$ adalah bilangan kecil. Metode *continued fraction* adalah metode untuk mencari berbagai $b_i$ dimana $|b_i^2 \bmod n| < 2\sqrt{n}$. Metode ini detemukan oleh Morisson dan Brillhart [mor75].

Kita mulai dengan penjelasan konsep *continued fraction.* Suatu bilangan nyata $x$ dapat dituliskan dalam bentuk *continued fraction* sebagai berikut:

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots + \cfrac{1}{a_i + x_{i+1}}}}$$

dimana

- $x_0 = x$, dan
- untuk $i \geq 0$, $a_i = \lfloor x_i \rfloor$ adalah bilangan bulat terbesar yang tidak lebih besar dari $x_i$, dan $x_{i+1} = \frac{1}{x_i - a_i}$.

Deretan $a_0, a_1, a_2, \ldots a_i$ diatas dapat dikomputasi menggunakan algoritma $CFA(x)$ (*continued fraction algorithm*) sebagai berikut:

1. $i \leftarrow 0$
2. $x_0 \leftarrow x$
3. $a_0 \leftarrow \lfloor x_0 \rfloor$
4. output($a_0$)
5. jika ($x_i = a_i$) berhenti disini
6. $x_{i+1} \leftarrow \frac{1}{x_i - a_i}$
7. $i \leftarrow i + 1$
8. $a_i \leftarrow \lfloor x_i \rfloor$
9. output($a_i$)
10. kembali ke langkah 5

Tidak terlalu sulit untuk membuktikan bahwa algoritma $CFA(x)$ akan berhenti pada suatu $i \geq 0$ jika dan hanya jika $x$ merupakan bilangan rasional. Mari kita bahas pembuktiannya. Jika algoritma $CFA(x)$ berhenti pada suatu $i \geq 0$, maka sangat jelas bahwa

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots + \cfrac{1}{a_i}}}$$

adalah bilangan rasional karena $a_0, a_1, \ldots a_i$ semua merupakan bilangan bulat. Sebaliknya jika $x$ merupakan bilangan rasional, maka setiap $x_i$ dengan $i \geq 0$

merupakan bilangan rasional (karena $x_0$ rasional dan jika $x_i$ rasional maka $x_{i+1} = \frac{1}{x_i - a_i}$ juga rasional untuk $i \geq 0$), sebut saja

$$x_i = u_i / u_{i+1}$$

dimana $u_i$ dan $u_{i+1}$ merupakan bilangan bulat. Jadi $a_i = \lfloor x_i \rfloor = \lfloor u_i / u_{i+1} \rfloor$ dan

$$\begin{aligned} x_{i+1} &= \frac{1}{x_i - a_i} \\ &= \frac{1}{u_i/u_{i+1} - \lfloor u_i/u_{i+1} \rfloor} \\ &= \frac{u_{i+1}}{u_i - u_{i+1}\lfloor u_i/u_{i+1} \rfloor} \\ &= \frac{u_{i+1}}{u_i \bmod u_{i+1}}. \end{aligned}$$

Jadi dari $x_i$ ke $x_{i+1}$, pasangan $(u_i, u_{i+1})$ berubah menjadi $(u_{i+1}, u_i \bmod u_{i+1})$. Tetapi ini persis sama dengan transformasi yang dilakukan algoritma Euclid (lihat 3.4) dimana pasangan $(r_i, r_{i+1})$ berubah menjadi $(r_{i+1}, r_i \bmod r_{i+1})$ (operasi mod sama dengan *residue*). Karena kita telah buktikan bahwa algoritma Euclid selalu berhenti, maka kita telah buktikan juga bahwa algoritma $CFA(x)$ selalu berhenti jika $x$ adalah bilangan rasional. Jadi telah kita buktikan bahwa algoritma $CFA(x)$ selalu berhenti jika dan hanya jika $x$ rasional.

Jika $x$ merupakan bilangan irasional, maka

$$\frac{b_i}{c_i} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots + \cfrac{1}{a_i}}} \qquad (14.1)$$

dinamakan juga konvergen ke-$i$ (*i-th convergent*) dari *continued fraction* dari $x$. Sangat jelas bahwa konvergen $\frac{b_i}{c_i}$ adalah bilangan rasional.

Pembaca mungkin bertanya bagaimana kita gunakan algoritma $CFA(x)$ terhadap bilangan irasional untuk mencari konvergen. Nanti kita kan bahas bagaimana caranya mencari konvergen untuk akar dari bilangan bulat, untuk sekarang anggap saja algoritma $CFA(x)$ sebagai algoritma konseptual, jadi bukan suatu algoritma yang dapat digunakan secara mentah.

**Teorema 101 (Definisi Konvergen)** *Dengan menggunakan notasi konvergen seperti diatas, maka*

$$\begin{aligned} \frac{b_0}{c_0} &= \frac{a_0}{1} \qquad (b_0 = a_0, c_0 = 1); \\ \frac{b_1}{c_1} &= \frac{a_0 a_1 + 1}{a_1} \qquad (b_1 = a_0 a_1 + 1, c_1 = a_1); \;\; \textit{dan} \end{aligned}$$

$$\frac{b_i}{c_i} = \frac{a_i b_{i-1} + b_{i-2}}{a_i c_{i-1} + c_{i-2}} \qquad (b_i = a_i b_{i-1} + b_{i-2}, c_i = a_i c_{i-1} + c_{i-2}) \textit{ jika } i \geq 2.$$

Kita buktikan teorema ini dengan induksi, dengan tidak menggunakan asumsi bahwa setiap $a_j$ berupa bilangan bulat, hanya bahwa persamaan 14.1 berlaku untuk konvergen. Untuk $i = 0$ dan $i = 1$ sangat jelas bahwa

$$\begin{aligned} \frac{b_0}{c_0} &= a_0 \\ &= \frac{a_0}{1}; \\ \frac{b_1}{c_1} &= a_0 + \frac{1}{a_1} \\ &= \frac{a_0 a_1 + 1}{a_1}. \end{aligned}$$

Untuk $i = 2$ kita dapatkan

$$\begin{aligned} \frac{b_i}{c_i} &= \frac{b_2}{c_2} \\ &= a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2}} \\ &= a_0 + \frac{a_2}{a_1 a_2 + 1} \\ &= \frac{a_0 a_1 a_2 + a_0 + a_2}{a_1 a_2 + 1} \\ &= \frac{a_2(a_0 a_1 + 1) + a_0}{a_2 c_1 + c_0} \\ &= \frac{a_2 b_1 + b_0}{a_2 c_1 + c_0} \\ &= \frac{a_i b_{i-1} + b_{i-2}}{a_i c_{i-1} + c_{i-2}}. \end{aligned}$$

Untuk langkah induksi, kita umpamakan bahwa teorema 101 berlaku untuk $i$, jadi kita mempunyai

$$\frac{b_i}{c_i} = \frac{a_i b_{i-1} + b_{i-2}}{a_i c_{i-1} + c_{i-2}} \qquad (b_i = a_i b_{i-1} + b_{i-2}, c_i = a_i c_{i-1} + c_{i-2})$$

sebagai hipotesis induksi, dan kita harus buktikan bahwa teorema berlaku untuk $i+1$. Konvergen ke $i+1$ didapat dengan menukar $a_i$ dengan $a_i + \frac{1}{a_{i+1}}$ pada konvergen ke $i$, jadi dengan menggunakan hipotesis induksi, kita dapatkan

$$\frac{b_{i+1}}{c_{i+1}} = \frac{(a_i + \frac{1}{a_{i+1}})b_{i-1} + b_{i-2}}{(a_i + \frac{1}{a_{i+1}})c_{i-1} + c_{i-2}}$$

$$\begin{aligned}
&= \frac{a_{i+1}(a_i b_{i-1} + b_{i-2}) + b_{i-1}}{a_{i+1}(a_i c_{i-1} + c_{i-2}) + c_{i-1}} \\
&= \frac{a_{i+1} b_i + b_{i-1}}{a_{i+1} c_i + c_{i-1}}.
\end{aligned}$$

Selesailah pembuktian teorema 101.

**Teorema 102**

$$b_i c_{i-1} - b_{i-1} c_i = (-1)^{i-1} \qquad \textit{untuk } x \geq 1.$$

Kita buktikan teorema 102 dengan induksi. Untuk $i = 1$ kita dapatkan

$$\begin{aligned}
b_i c_{i-1} - b_{i-1} c_i &= b_1 c_0 - b_0 c_1 \\
&= (a_0 a_1 + 1) - a_0 a_1 \\
&= 1 \\
&= (-1)^0 \\
&= (-1)^{i-1}.
\end{aligned}$$

Untuk langkah induksi, dengan hipotesis

$$b_i c_{i-1} - b_{i-1} c_i = (-1)^{i-1},$$

kita dapatkan

$$\begin{aligned}
b_{i+1} c_i - b_i c_{i+1} &= (a_{i+1} b_i + b_{i-1}) c_i - b_i (a_{i+1} c_i + c_{i-1}) \text{ (dari teorema 101)} \\
&= a_{i+1} b_i c_i + b_{i-1} c_i - a_{i+1} b_i c_i - b_i c_{i-1} \\
&= b_{i-1} c_i - b_i c_{i-1} \\
&= -(-1)^{i-1} \text{ (menggunakan hipotesis induksi)} \\
&= (-1)^i.
\end{aligned}$$

Selesailah pembuktian teorema 102.

**Teorema 103** *Menggunakan teorema 101 sebagai definisi konvergen, maka*

$$\gcd(b_i, c_i) = 1.$$

Untuk $i = 0$ pembuktian teorema 103 sangat mudah karena $\gcd(a_0, 1) = 1$. Untuk $i \geq 1$, dari teorema 102 kita dapat menyimpulkan bahwa pembagi $b_i$ dan $c_i$ harus membagi $(-1)^i$ yang mempunyai nilai $\pm 1$, jadi $\gcd(b_i, c_i) = 1$. Selesailah pembuktian teorema 103.

Jika kita bagi persamaan dalam teorema 102 dengan $c_i c_{i-1}$ maka kita dapatkan

$$\frac{b_i}{c_i} - \frac{b_{i-1}}{c_{i-1}} = \frac{(-1)^i}{c_i c_{i-1}}.$$

Jadi konvergen berosilasi dengan amplitudo yang mengecil, semakin besar $i$. Mari kita tunjukkan bahwa semakin besar $i$, konvergen semakin mendekati $x$. Perhatikan bahwa kita bisa mendapatkan $x$ dari konvergen ke $i+1$ dengan menukar $a_{i+1}$ dengan $x_{i+1}$. Jadi menggunakan teorema 101 (ingat bahwa pembuktian teorema 101 tidak mengasumsikan bahwa $a_i$ adalah bilangan bulat) kita dapatkan

$$\begin{aligned} x &= \frac{b_i/x_{i+1} + b_{i-1}}{c_i/x_{i+1} + c_{i-1}} \\ &= \frac{b_i + x_{i+1}b_{i-1}}{c_i + x_{i+1}c_{i-1}}, \end{aligned}$$

Jika $\mathbf{u} = (c_i, b_i)$ dan $\mathbf{v} = (c_{i-1}, b_{i-1})$ dianggap sebagai vektor dalam bidang (dua dimensi) maka kedua vektor berada pada kuadran yang sama, dan kemiringan dari vektor $\mathbf{u} + x_{i+1}\mathbf{v} = (c_i + x_{i+1}c_{i-1}, b_i + x_{i+1}b_{i-1})$ adalah

$$\frac{b_i + x_{i+1}b_{i-1}}{c_i + x_{i+1}c_{i-1}} = x$$

dan berada antara kemiringan $\mathbf{u}$ yaitu $\frac{b_i}{c_i}$ dan kemiringan $\mathbf{v}$ yaitu $\frac{b_{i-1}}{c_{i-1}}$. Jadi deretan $\frac{b_i}{c_i}$ (konvergen) dengan $i = 0, 1, 2, \ldots$ berosilasi antara bawah dan atas $x$ dan mendekati $x$ secara monoton (jadi limit dari konvergen adalah $x$).

**Teorema 104** *Jika $x > 1$ merupakan bilangan irasional dengan konvergen $\frac{b_i}{c_i}$ untuk $i = 0, 1, 2, \ldots$, maka untuk setiap $i$, $|b_i^2 - x^2c_i^2| < 2x$.*

Untuk membuktikan teorema 104 kita gunakan fakta bahwa nilai $x$ berada diantara $\frac{b_i}{c_i}$ dan $\frac{b_{i+1}}{c_{i+1}}$, dan karena menurut teorema 102 perbedaan antara kedua konvergen adalah $1/c_ic_{i+1}$, maka kita dapatkan

$$\begin{aligned} |b_i^2 - x^2c_i^2| &= c_i^2|x - \frac{b_i}{c_i}||x + \frac{b_i}{c_i}| \\ &< c_i^2\frac{1}{c_ic_{i+1}}(x + (x + \frac{1}{c_ic_{i+1}})). \end{aligned}$$

Berarti

$$\begin{aligned} |b_i^2 - x^2c_i^2| - 2x &< 2x(-1 + \frac{c_i}{c_{i+1}} + \frac{1}{2xc_{i+1}^2}) \\ &< 2x(-1 + \frac{c_i}{c_{i+1}} + \frac{1}{c_{i+1}}) \\ &< 2x(-1 + \frac{c_{i+1}}{c_{i+1}}) \\ &= 0. \end{aligned}$$

Jadi $|b_i^2 - x^2c_i^2| < 2x$ dan selesailah pembuktian teorema 104.

**Teorema 105** *Jika $n$ merupakan bilangan bulat positif yang bukan merupakan kuadrat ($n$ bukan perfect square) dan deretan $\frac{b_i}{c_i}$ dengan $i = 0, 1, 2, \ldots$ merupakan deretan konvergen dari $\sqrt{n}$, maka least absolute residue dari $b_i^2 \bmod n$ lebih kecil dari $2\sqrt{n}$.*

Teorema 105 didapat dari teorema 104 dengan $x = \sqrt{n}$. Teorema 105 menjadi dasar dari penggunaan metode *continued fraction* untuk menguraikan bilangan, dengan mengatakan bahwa terdapat deretan $b_i$ dengan kuadrat yang mempunyai *residue* yang kecil.

Sekarang kita bahas akar dari bilangan bulat. Kita akan tunjukkan bahwa jika suatu bilangan bulat $n$ bukan berupa *perfect square* (kuadrat dari bilangan bulat), maka $\sqrt{n}$ adalah bilangan irasional.

**Teorema 106** *Jika $n$ adalah bilangan bulat positif yang bukan berupa kuadrat bilangan bulat juga ($n$ bukan perfect square) maka $\sqrt{n}$ adalah bilangan irasional.*

Pembuktian teorema ini sangat mudah, karena jika $\sqrt{n}$ adalah bilangan rasional, sebut saja $\frac{a}{b}$, maka $\frac{a^2}{b^2} = n$ adalah bilangan bulat ($a^2$ dapat dibagi $b^2$), dan $\frac{a}{b}$ juga bilangan bulat (karena jika $a^2$ dapat dibagi oleh $b^2$ maka $a$ dapat dibagi oleh $b$). Jadi jika $\sqrt{n}$ adalah bilangan rasional maka $n$ berupa *perfect square*. Sebaliknya jika $n$ berupa *perfect square* maka $\sqrt{n}$ adalah bilangan bulat yang tentu saja juga rasional. Berarti $\sqrt{n}$ merupakan bilangan rasional jika dan hanya jika $n$ merupakan *perfect square*. Jadi jika $n$ bukan *perfect square* maka $\sqrt{n}$ adalah bilangan irasional. Selesailah pembuktian kita.

Kita lanjutkan dengan penjelasan mengenai kalkulasi $CFA(x)$ jika $x$ merupakan akar dari bilangan bulat, jadi $x = \sqrt{n}$ dimana $n$ adalah suatu bilangan bulat. Jika $n$ merupakan *perfect square*, maka algoritme $CFA(x)$ langsung berhenti karena $x = \sqrt{n}$ merupakan bilangan bulat. Jika $n$ bukan merupakan *perfect square* maka menurut teorema 106, $\sqrt{n}$ merupakan bilangan irasional dan algoritma $CFA(x)$ dapat berjalan terus tanpa berahir. Yang perlu dijelaskan adalah bagaimana melakukan langkah 3, 6 dan 8 dari algoritma $CFA(x)$. Untuk langkah 3, $x_0 = \sqrt{n}$, jadi $a_0 = \lfloor x_0 \rfloor$ merupakan bilangan bulat terbesar yang jika dikuadratkan tidak melebihi $n$ ($a_0^2 < n$ dan $(a_0 + 1)^2 > n$). Cara yang cukup efisien untuk mencari $a_0$ adalah dengan teknik *binary search* pada semua bilangan antara 1 dan $n$. Pertama kali langkah 6 dilakukan, $i = 0$, jadi

$$\begin{aligned} x_1 &= \frac{1}{\sqrt{n} - a_0} \\ &= \frac{\sqrt{n} + a_0}{n - a_0^2} \\ &= \frac{\sqrt{n} + a_0}{m} \end{aligned}$$

untuk suatu bilangan bulat $m > 0$ (karena $n > a_0^2$). Tidak terlalu sulit untuk melihat bahwa

$$\begin{aligned} a_1 &= \lfloor x_1 \rfloor \\ &= \lfloor \frac{\lfloor\sqrt{n}\rfloor + a_0}{m} \rfloor \end{aligned}$$

dan

$$\begin{aligned} x_1 - a_1 &= \frac{\sqrt{n} + a_0 - km}{m} \\ &= \frac{\sqrt{n} - j}{m} \end{aligned}$$

untuk suatu bilangan bulat $j = km - a_0$. Jadi

$$\begin{aligned} x_2 &= \frac{1}{x_1 - a_1} \\ &= \frac{m}{\sqrt{n} - j} \end{aligned}$$

yang dapat dikalikan dengan $\frac{\sqrt{n}+j}{\sqrt{n}+j}$ agar denominator menjadi bilangan bulat. Proses ini dapat diulang untuk mengkalkulasi $a_2, a_3, \ldots$ dan seterusnya.

Sekarang kita tiba pada penjelasan mengenai bagaimana metode *continued fraction* dapat digunakan untuk menguraikan bilangan bulat. Algoritma untuk menguraikan bilangan bulat menggunakan metode *continued fraction* adalah sebagai berikut:

1. $i \leftarrow 0$
2. $b_{-1} \leftarrow 1$, $x_0 \leftarrow \sqrt{n}$
3. $b_0 \leftarrow a_0 \leftarrow \lfloor x_0 \rfloor$
4. $x_{i+1} \leftarrow \frac{1}{x_i - a_i}$
5. $i \leftarrow i + 1$
6. $a_i \leftarrow \lfloor x_i \rfloor$
7. $b_i \leftarrow a_i b_{i-1} + b_{i-2} \pmod{n}$
8. kalkulasi $b_i^2 \bmod n$
9. kembali ke langkah 4

Setelah langkah-langkah 4 sampai dengan 9 diulang beberapa kali, kita periksa bilangan-bilangan hasil kalkulasi langkah 8. Kita buat *factor base* terdiri dari $-1$ dan semua bilangan prima yang merupakan faktor dari lebih dari satu $b_i^2 \bmod n$ atau merupakan faktor dengan pangkat genap dari hanya satu $b_i^2 \bmod n$. Selanjutnya, seperti halnya dengan metode Dixon, kita cari kumpulan dari $b_i$ yang jika dikalikan menghasilkan produk dari pemangkatan genap elemen-elemen *factor base* dan menghasilkan

$$\begin{aligned} b^2 &\equiv c^2 \pmod{n} \text{ dan} \\ b &\not\equiv \pm c \pmod{n}. \end{aligned}$$

Jika kumpulan tidak dapat ditemukan, maka langkah-langkah 4 sampai dengan 9 harus diteruskan hingga kita mendapatkan kumpulan yang diinginkan.

Sebagai contoh penggunaan metode *continued fraction* mari kita coba uraikan 17873. Jika kita kalkulasi $a_i$, $b_i$ dan $b_i^2 \bmod n$ untuk $i = 0, 1, 2, 3, 4, 5$ kita dapatkan tabel berikut:

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| $a_i$ | 133 | 1 | 2 | 4 | 2 | 3 |
| $b_i$ | 133 | 134 | 401 | 1738 | 3877 | 13369 |
| $b_i^2 \bmod n$ | $-184$ | 83 | $-56$ | 107 | $-64$ | 161 |

Kita dapatkan *factor base* $B = \{-1, 2, 7, 23\}$ dan mendapatkan *B-number* untuk $i = 0, 2, 4, 5$ dengan $\vec{\epsilon_i}$ masing-masing $(1, 1, 0, 1)$, $(1, 1, 1, 0)$, $(1, 0, 0, 0)$ dan $(0, 0, 1, 1)$. Jika kita tambahkan vektor pertama, kedua dan keempat menggunakan aritmatika modulo 2, kita dapatkan vektor nol. Tetapi jika kita kalkulasi:

$$\begin{aligned} b &= \prod b_i \bmod n \\ &= 133 \cdot 401 \cdot 13369 \\ &\equiv 1288 \pmod{17873} \end{aligned}$$

dan

$$\begin{aligned} c &= \prod p_j^{\gamma_j} \\ &= 2^3 \cdot 7 \cdot 23 \\ &= 1288 \end{aligned}$$

kita dapatkan $b \equiv c \pmod{17873}$, jadi produk tidak bisa digunakan. Karena tidak ada produk lain yang dapat menghasilkan vektor nol dari tabel diatas, kita lanjutkan langkah-langkah 4 sampai dengan 9 untuk $i = 6, 7, 8$ untuk memperbesar tabel:

| $i$ | 6 | 7 | 8 |
|---|---|---|---|
| $a_i$ | 1 | 2 | 1 |
| $b_i$ | 17246 | 12115 | 11488 |
| $b_i^2 \bmod n$ | $-77$ | 149 | $-88$ |

Kita dapatkan *factor base* yang lebih besar yaitu $\{-1, 2, 7, 11, 23\}$ dan *B-number* untuk $i = 0, 2, 4, 5, 6, 8$ masing-masing dengan $\vec{\epsilon_i}$ sebagai berikut: $(1, 1, 0, 0, 1)$, $(1, 1, 1, 0, 0)$, $(1, 1, 0, 0, 0)$, $(0, 0, 1, 0, 1)$, $(1, 0, 1, 1, 0)$, $(1, 1, 0, 1, 0)$. Jika kita tambahkan vektor kedua, ketiga, kelima dan keenam menggunakan aritmatika modulo 2, kita dapatkan vektor nol, tetapi sekarang kita dapatkan $b = 7272$ dan $c = 4928$, jadi $\gcd(7272 + 4928, 17873) = 61$ merupakan faktor dari 17873. Kita dapatkan penguraian $17873 = 61 \cdot 293$.

Kompleksitas metode *continued fraction* seperti metode Dixon diestimasikan adalah

$$O(e^{C\sqrt{log\, n\, log\, log\, n}})$$

tetapi dengan konstan $C$ yang lebih kecil. Untuk analisa yang cukup mendalam mengenai kompleksitas metode ini, silahkan membaca [pom82].

## 14.5 Metode Quadratic Sieve

Seperti halnya dengan metode *continued fraction*, metode *quadratic sieve* juga membuat himpunan *B-number* secara sistematis. Perbedaan terdapat pada cara menghasilkan himpunan. Metode *continued fraction* menggunakan algoritma $CFA(x)$ untuk menghasilkan himpunan *B-number*, sedangkan metode *quadratic sieve* menggunakan proses penyaringan untuk menghasilkan himpunan tersebut. Kita akan bahas metode *quadratic sieve* dan proses penyaringan yang digunakannya.

Untuk *factor base*, metode *quadratic sieve* menggunakan himpunan bilangan prima yang relatif kecil dan untuk bilangan prima $p$ yang ganjil, $p$ mematuhi persamaan

$$\left(\frac{n}{p}\right) = 1$$

jadi $n$ (bilangan yang hendak diuraikan) adalah *quadratic residue* modulo $p$. Biasanya batas $P$ untuk himpunan dipilih dengan nilai sekitar

$$e^{\sqrt{\log n \log\log n}}.$$

Jadi *factor base* terdiri dari 2 dan semua bilangan prima ganjil $p \leq P$ dimana $n$ adalah *quadratic residue* modulo $p$.

Untuk membatasi banyaknya *B-number*, dipilih $A$ yang lebih besar dari $P$ tetapi tidak lebih dari pemangkatan kecil dari $P$, sebagai contoh:

$$P < A < P^2.$$

Kandidat untuk menghasilkan *B-number* yang diberi notasi $t$ dibatasi sebagai berikut:

$$t = \lfloor\sqrt{n}\rfloor + 1, \lfloor\sqrt{n}\rfloor + 2, \ldots, \lfloor\sqrt{n}\rfloor + A.$$

Tentunya kandidat harus disaring lagi sehingga yang lolos benar merupakan *B-number*. Untuk proses penyaringan, kita gunakan tabel dengan kolom-kolom untuk $t$, $t^2 - n$, setiap bilangan dalam *factor base*, dan hasil pembagian $t^2 - n$ dengan faktor-faktor dalam *factor base*. Karena tujuan penyaringan adalah setiap $t^2 - n$ yang lolos adalah suatu *B-number*, maka $t^2 - n$ yang lolos harus merupakan produk dari faktor-faktor dalam *factor base*. Istilah matematikanya adalah $t^2 - n$ *smooth over the factor base*. Mari kita pelajari satu varian dari algoritma penyaringan untuk metode *quadratic sieve*.

1. Buat tabel dengan kolom-kolom untuk $t$, $t^2 - n$, $x$, dan bilangan-bilangan prima yang berada dalam *factor base*. Awalnya ada baris-baris untuk $t = \lfloor\sqrt{n}\rfloor + 1, \lfloor\sqrt{n}\rfloor + 2, \ldots, \lfloor\sqrt{n}\rfloor + A$, dan nilai $t^2 - n$ dimasukkan ke kolom untuk $t^2 - n$ dan kolom untuk $x$.

2. Untuk setiap $p$ ganjil dalam *factor base*, dapatkan solusi persamaan $t^2 \equiv n \pmod{p^\alpha}$ dengan $\alpha = 1, 2, \ldots, \beta$ dimana $\beta$ adalah pangkat terbesar yang memberi solusi dengan $\lfloor\sqrt{n}\rfloor + 1 \leq t \leq \lfloor\sqrt{n}\rfloor + A$. Ini dapat dilakukan, misalnya dengan cara yang dibahas di bagian 11.2. Solusi untuk persamaan $t^2 \equiv n \pmod{p^\beta}$ ada dua, sebut saja $t_1$ dan $t_2$, yang keduanya tidak harus berada dalam interval $(\lfloor\sqrt{n}\rfloor + 1, \lfloor\sqrt{n}\rfloor + A)$. Dengan $\alpha = 1$ lantas 2 dan seterusnya hingga $\beta$, untuk setiap $t$ yang berbeda kelipatan $p^\alpha$ dari $t_1$ dan setiap $t$ yang berbeda kelipatan $p^\alpha$ dari $t_2$ kita bagi $x$ dengan $p$ dan taruh $\alpha$ dalam kolom untuk $p$.

3. Untuk $p = 2$, jika $n \equiv 1 \pmod 8$ maka prosedurnya sama dengan untuk $p$ ganjil, tetapi untuk $\beta \geq 3$ solusinya ada 4 ($t_1, t_2, t_3$ dan $t_4$). Jika $n \equiv 5 \pmod 8$ maka prosedurnya sama dengan untuk $p$ ganjil, tetapi $\beta = 2$ dan solusinya ada 2. Jika $n \not\equiv 1 \pmod 4$ maka untuk setiap $t$ yang ganjil, kita taruh 1 dalam kolom untuk $p = 2$ dan kita bagi $x$ dengan 2.

4. Buang $t$ yang tidak menghasilkan $x = 1$.

Jadi setelah langkah-langkah diatas dilakukan, yang tersisa adalah baris-baris untuk $t$ dimana $t^2 - n$ terbagi habis oleh faktor-faktor dalam *factor base* ($t^2 - n$ *smooth over the factor base*), dengan kata lain $t^2 - n$ merupakan *B-number*. Kita tinggal mencari kombinasi linear beberapa $t$ yang menghasilkan pemangkatan genap elemen-elemen *factor base*. Caranya sama dengan metode Dixon dan metode *continued fraction* yang telah dibahas.

Sedikit penjelasan untuk $p = 2$. Persamaan yang harus dicari solusinya adalah:

$$t^2 \equiv n \pmod{2^\alpha}.$$

Karena $n$ ganjil maka $t$ harus berupa bilangan ganjil. Jadi $t$ dapat ditulis dalam bentuk $t = 2m + 1$ dimana $m$ adalah bilangan bulat. Jadi

$$n \equiv (2m+1)^2 \pmod{2^\alpha}$$

$$\begin{aligned} &\equiv 4m^2 + 4m + 1 \pmod{2^\alpha} \\ &\equiv 1 + 4m(m+1) \pmod{2^\alpha}. \end{aligned}$$

Karena $m$ adalah bilangan bulat, maka $m(m+1)$ merupakan bilangan genap, jadi dapat ditulis dalam bentuk $m(m+1) = 2j$ dimana $j$ adalah bilangan bulat. Jadi

$$\begin{aligned} n &\equiv 1 + 4(2j) \pmod{2^\alpha} \\ &\equiv 1 + 8j \pmod{2^\alpha}. \end{aligned}$$

Untuk $\alpha \geq 3$, ini berarti $n \equiv 1 \pmod 8$. Untuk $\alpha = 2$, kita dapatkan

$$\begin{aligned} n &\equiv 1 + 8j \pmod 4 \\ &\equiv 1 \pmod 4. \end{aligned}$$

Jika $n \equiv 5 \pmod 8$ maka ada dua solusi untuk

$$t^2 \equiv 1 \pmod 4$$

yaitu $t_1 \equiv 1 \pmod 4$ dan $t_2 \equiv 3 \pmod 4$. Jika $n \not\equiv 1 \pmod 4$ maka solusi untuk

$$t^2 \equiv 1 \pmod 2$$

adalah $t \equiv 1 \pmod 2$ yang berarti $t$ ganjil.

Sebagai contoh penggunaan metode *quadratic sieve*, mari kita coba uraikan $n = 1042387$. Kita dapatkan $\lfloor\sqrt{n}\rfloor = 1020$ dan $e^{\sqrt{\log n \log\log n}} \approx 418$. Kita pilih $P = 50$ dan $A = 500$. Kita dapatkan *factor base* $\{2, 3, 11, 17, 19, 23, 43, 47\}$ ($n$ adalah *quadratic residue* untuk setiap bilangan prima dalam *factor base*). Untuk $p = 2$, karena $n \not\equiv 1 \pmod 4$, maka kolom berisi 1 untuk setiap $t$ ganjil dan kosong (atau 0) untuk setiap $t$ genap. Untuk kolom $p = 3$, kita perlu solusi

$$t_1 = t_{1,0} + 3t_{1,1} + 3^2 t_{1,2} + \cdots + 3^{\beta-1} t_{1,\beta-1}$$

untuk persamaan

$$t_1^2 \equiv 1042387 \pmod{3^\beta}$$

dimana setiap $t_{1,i}$ merupakan *ternary digit* ($t_{1,i} \in \{0, 1, 2\}$). Karena $p = 3$ tidak terlalu besar, kita dapat melakukan *trial and error* untuk mendapatkan $t_{1,0} = 1$ dari

$$t_{1,0}^2 \equiv 1042387 \pmod 3.$$

Jika $p$ terlalu besar, kita dapat menggunakan teknik yang telah dibahas di bagian 11.2. Berikutnya, kita cari $t_{1,1}$ dari

$$(1 + 3t_{1,1})^2 \equiv 1042387 \pmod{3^2},$$

mendapatkan

$$1 + 6t_{1,1} \equiv 7 \pmod{9}.$$

Jadi $6t_{1,1} \equiv 6 \pmod{9}$ atau $2t_{1,1} \equiv 2 \pmod{3}$, jadi kita dapatkan $t_{1,1} = 1$. Selanjutnya untuk $t_{1,2}$ kita gunakan

$$(1 + 3 + 9t_{1,2})^2 \equiv 1042387 \pmod{3^3},$$

dan seterusnya hingga kita dapatkan $t_1 \equiv (210211)_3 \pmod{3^7}$ dan $t_2 = 3^7 - (210211)_3 = (2012012)_3$ jadi $t_2 \equiv (2012012)_3 \pmod{3^7}$. Namun tidak ada $t$ diantara 1021 dan 1520 inklusif yang mematuhi persamaan $t \equiv t_1 \pmod{3^7}$ atau $t \equiv t_2 \pmod{3^7}$. Jadi $\beta = 6$ dan kita ambil

$$\begin{aligned} t_1 &\equiv (210211)_3 \pmod{3^6} \\ &\equiv 1318 \pmod{3^6} \end{aligned}$$

dan $t_2 = 3^6 - 1318 = 140$, dan karena tidak ada $t \equiv t_2 \pmod{3^6}$ dengan $1021 \leq t \leq 1050$ maka

$$\begin{aligned} t_2 &\equiv 140 \pmod{3^5} \\ &\equiv 1112 \pmod{3^5}. \end{aligned}$$

Proses penyaringan dimulai dengan melakukan langkah-langkah berikut untuk setiap $t$ dengan $1021 \leq t \leq 1050$ dan $t - t_1 \equiv 0 \pmod{3}$:

1. taruh 1 di kolom $p = 3$,
2. bagi $x$ dengan 3,

lalu untuk setiap $t$ dengan $1021 \leq t \leq 1050$ dan $t - t_1 \equiv 0 \pmod{3^2}$ kita lakukan langkah-langkah berikut:

1. ganti 1 menjadi 2 di kolom $p = 3$,
2. bagi $x$ dengan 3,

dan seterusnya hingga untuk setiap $t$ dengan $1021 \leq t \leq 1050$ dan $t - t_1 \equiv 0 \pmod{3^6}$ kita lakukan langkah-langkah berikut:

1. ganti 5 menjadi 6 di kolom $p = 3$,
2. bagi $x$ dengan 3.

Proses dilanjutkan dengan melakukan langkah-langkah berikut untuk setiap $t$ dengan $1021 \leq t \leq 1050$ dan $t - t_2 \equiv 0 \pmod{3}$:

1. taruh 1 di kolom $p = 3$,

2. bagi $x$ dengan 3,

dan seterusnya hingga untuk setiap $t$ dengan $1021 \leq t \leq 1050$ dan $t - t_2 \equiv 0 \pmod{3^5}$ kita lakukan langkah-langkah berikut:

1. ganti 4 menjadi 5 di kolom $p = 3$,
2. bagi $x$ dengan 3.

Proses penyaringan dilakukan dengan setiap $p$ ganjil dalam *factor base*. Setelah kita buang setiap $t$ dengan $x \neq 1$ kita dapatkan tabel 14.1. Dari baris ke 5 dan

| $t$ | $t^2 - n$ | $x$ | 2 | 3 | 11 | 17 | 19 | 23 | 43 | 47 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1021 | 54 | 1 | 1 | 3 | – | – | – | – | – | – |
| 1027 | 12342 | 1 | 1 | 1 | 2 | 1 | – | – | – | – |
| 1030 | 18513 | 1 | – | 2 | 2 | 1 | – | – | – | – |
| 1061 | 83334 | 1 | 1 | 1 | – | 1 | 1 | – | 1 | – |
| 1112 | 194157 | 1 | – | 5 | – | 1 | – | – | – | 1 |
| 1129 | 232254 | 1 | 1 | 3 | 1 | 1 | – | 1 | – | – |
| 1148 | 275517 | 1 | – | 2 | 3 | – | – | 1 | – | – |
| 1175 | 338238 | 1 | 1 | 2 | – | – | 1 | 1 | 1 | – |
| 1217 | 438702 | 1 | 1 | 1 | 1 | 2 | – | 1 | – | – |
| 1390 | 889713 | 1 | – | 2 | 2 | – | 1 | – | 1 | – |
| 1520 | 1268013 | 1 | – | 1 | – | 1 | – | 2 | – | 1 |

Tabel 14.1: Tabel Quadratic Sieve untuk $n = 1042387$

baris terahir kita dapatkan

$$(1112 \cdot 1520)^2 \equiv (3^{(5+1)/2} \cdot 17^{(1+1)/2} \cdot 23^{2/2} \cdot 47^{(1+1)/2})^2 \pmod{1042387}$$

jadi

$$647853^2 \equiv 496179^2 \pmod{1042387}.$$

Kita dapatkan faktor $\gcd(647853 - 496179, 1042387) = 1487$.

Kompleksitas metode *quadratic sieve*, seperti halnya dengan metode *continued fraction* dan metode Dixon, diestimasikan adalah

$$O(e^{C\sqrt{\log n \log \log n}})$$

tetapi dengan konstan $C$ yang lebih kecil lagi. Untuk analisa yang cukup mendalam mengenai kompleksitas metode ini, silahkan membaca [pom82].

Dalam prakteknya, metode *quadratic sieve* merupakan cara tercepat untuk menguraikan bilangan besar sampai dengan 100 digit. Untuk menguraikan bilangan lebih dari 100 digit, metode *number field sieve* yang akan dibahas di bagian berikut, lebih cepat.

## 14.6 Metode Number Field Sieve

Yang dimaksud dengan metode *number field sieve* disini adalah metode *general number field sieve*. Ini berbeda dengan metode *special number field sieve* yang digunakan untuk mencari faktor dari bilangan-bilangan tertentu seperti *Fermat numbers*. Secara garis besar, metode *number field sieve* untuk menguraikan bilangan $n$ adalah sebagai berikut. Pilih *degree* $d$ untuk *polynomial* $f(x)$:

$$d \approx \left(\frac{3\log n}{2\log\log n}\right)^{\frac{1}{3}}.$$

Untuk $n$ 100 hingga 200 digit, biasanya dipilih $d = 5$. Berikutnya, dengan $m$ sebagai bagian bulat akar pangkat $d$ dari $n$:

$$m = \lfloor \sqrt[d]{n} \rfloor,$$

tuliskan $n$ sebagai bilangan dengan basis $m$:

$$n = m^d + c_{d-1}m^{d-1} + \ldots + c_1 m + c_0,$$

dimana $0 \leq c_i < m$. Maka *polynomial* $f(x)$ didefinisikan sebagai berikut:

$$f(x) = x^d + c_{d-1}x^{d-1} + \ldots + c_1 x + c_0,$$

jadi $n = f(m)$. Jika $f(x)$ *reducible*, maka $f(x)$ dapat diuraikan menjadi $f(x) = g(x)h(x)$, dimana $g(x)$ dan $h(x)$ merupakan *polynomial* non-konstan. Jadi

$$n = f(m) = g(m) \cdot h(m)$$

yang akan menghasilkan penguraian $n$ dan kita selesai. Jadi untuk selanjutnya kita umpamakan $f(x)$ *irreducible*. Algoritma untuk penguraian *polynomial*, contohnya yang dijelaskan di [len82], dapat digunakan untuk melakukan test apakah $f(x)$ *irreducible* (dan menguraikannya jika $f(x)$ *reducible*). Algoritma tersebut menguraikan *polynomial* dalam $\mathbf{Q}$, tetapi, seperti akan dijelaskan, *polynomial* yang *irreducible* dalam $\mathbf{N}$ berarti juga *irreducible* dalam $\mathbf{Q}$. Kompleksitas dari algoritma tersebut adalah *polynomial*.

Jika metode *quadratic sieve* menyaring kandidat $t$ dimana $t^2 - n$ *smooth over factor base* (dapat diuraikan menjadi produk elemen-elemen *factor base*), maka metode *number field sieve* menyaring kandidat pasangan $(a, b)$ dimana $a + mb$ dan $(-b)^d f(-\frac{a}{b})$ keduanya dapat diuraikan menjadi produk elemen-elemen *factor base*. Penyaringan dilakukan untuk memperkecil ruang pencarian subset kandidat yang menghasilkan kuadrat. Akan tetapi, untuk metode *number field sieve*, perlu ada penyaringan tambahan dengan berbagai alasan yang akan dibahas dalam penjelasan berikut.

Motivasi untuk metode *number field sieve* adalah pengamatan bahwa dalam metode *quadratic sieve*, fungsi

$$f(t) = t^2 - n,$$

menghasilkan *ring homomorphism* dari $\mathbf{Z}$ ke $\mathbf{Z}/n\mathbf{Z}$, yang memetakan dua kuadrat yang berbeda dalam $\mathbf{Z}$ (yaitu $\prod_{t_i \in U} f(t_i)$ dan $\prod_{t_i \in U} t_i^2$) ke kuadrat yang sama dalam $\mathbf{Z}/n\mathbf{Z}$. Yang menjadi pertanyaan adalah apakah $f$ harus berupa *polynomial* dengan *degree* 2 dan apakah *ring homomorphism* harus dari $\mathbf{Z}$ ke $\mathbf{Z}/n\mathbf{Z}$? Jawabannya ternyata tidak.

Mari kita mulai dengan $f(x)$ berupa *monic polynomial* yang *irreducible* sebagai *polynomial* bilangan bulat. Kita gunakan *Gauss's Lemma 2* untuk menunjukkan bahwa *polynomial* tersebut juga *irreducible* sebagai *polynomial* bilangan rasional.

**Teorema 107 (Gauss's Lemma 2)**

1. *Produk dari dua polynomial primitif adalah polynomial primitif.*
2. *Jika suatu polynomial irreducible sebagai polynomial bilangan bulat, maka polynomial tersebut juga irreducible sebagai polynomial bilangan rasional.*

Suatu *polynomial* disebut primitif jika setiap koefisien berupa bilangan bulat dan gcd (pembagi persekutuan terbesar) dari semua koefisien adalah 1. Pembuktian bagian pertama kita lakukan dengan dua cara: cara pertama adalah pembuktian kongkrit, sedangkan cara kedua lebih abstrak. Jika $f(x)$ dan $g(x)$ keduanya merupakan *polynomial* primitif, maka jelas bahwa setiap koefisien produk $f(x) \cdot g(x)$ merupakan bilangan bulat. Jika produk bukan merupakan *polynomial* primitif maka terdapat bilangan prima $p$ yang membagi setiap koefisien dari produk. Karena $f(x)$ dan $g(x)$ keduanya primitif, maka terdapat suku-suku dalam $f(x)$ dan $g(x)$ yang koefisiennya tidak dapat dibagi oleh $p$. Jika $a_r x^r$ merupakan suku pertama dalam $f(x)$ yang tidak dapat dibagi oleh $p$ dan $b_s x^s$ merupakan suku pertama dalam $g(x)$ yang tidak dapat dibagi oleh $p$, maka suku $x^{r+s}$ dalam produk mempunyai koefisien dengan rumus:

$$a_r b_s + a_{r+1} b_{s-1} + a_{r+2} b_{s-2} + \ldots + a_{r-1} b_{s+1} + a_{r-2} b_{s+2} + \ldots$$

Suku pertama dalam koefisien tidak dapat dibagi oleh $p$ (karena $a_r$ dan $b_s$ keduanya tidak dapat dibagi $p$), sedangkan suku-suku lainnya dapat dibagi $p$ (karena $b_{s-1}, b_{s-1}, \ldots$ dan $a_{r-1}, a_{r-2}, \ldots$ semua dapat dibagi $p$), jadi koefisien tidak dapat dibagi $p$. Jadi produk harus berupa *polynomial* primitif.

Cara pembuktian kedua lebih abstrak. Jika $f(x) \cdot g(x)$ tidak primitif, maka semua koefisien produk dapat dibagi oleh satu bilangan prima, sebut saja $p$. Berarti $f(x) \cdot g(x) = 0$ dalam *ring* $(\mathbf{Z}/p\mathbf{Z})[X]$. Karena $\mathbf{Z}/p\mathbf{Z}$ merupakan *integral domain*, maka $f(x)$ atau $g(x)$ harus 0 dalam $(\mathbf{Z}/p\mathbf{Z})[X]$, jadi $p$ harus membagi setiap koefisien $f(x)$ atau $g(x)$. Tetapi ini tidak mungkin karena $f(x)$ dan $g(x)$ keduanya primitif. Jadi produk juga harus primitif.

Mari kita buktikan bagian kedua dari teorema 107 secara kontra-positif menggunakan bagian pertama. Jika *polynomial* $f(x)$ tidak primitif, maka $f(x)$ dapat dibagi oleh gcd semua koefisien $f(x)$ menghasilkan $f'(x)$ yang primitif; dan, jika $f(x)$ dapat diuraikan menjadi produk dua *polynomial* bilangan bulat non-konstan, maka $f'(x)$ juga dapat diuraikan menjadi produk dua *polynomial* bilangan bulat non-konstan. Jadi untuk pembuktian, kita asumsikan bahwa $f(x)$ primitif. Jika $f(x)$ *reducible* sebagai *polynomial* bilangan rasional maka $f(x)$ dapat diuraikan menjadi produk dua *polynomial* bilangan rasional non-konstan $g(x)$ dan $h(x)$ ($f(x) = g(x) \cdot h(x)$). Terdapat $a, b \in \mathbf{Z}$ dimana $a \cdot g(x)$ dan $b \cdot h(x)$ keduanya primitif (dalam $\mathbf{Z}[X]$). Menggunakan bagian pertama, berarti

$$(a \cdot g(x)) \cdot (b \cdot h(x)) = (ab)f(x)$$

juga primitif. Berarti $ab$ merupakan unit dalam $\mathbf{Z}$, jadi $a$ dan $b$ masing-masing merupakan unit dalam $\mathbf{Z}$. Jadi $f(x)$ dapat diuraikan menjadi

$$f(x) = (b^{-1} \cdot g(x)) \cdot (a^{-1} \cdot h(x))$$

dimana koefisien-koefisien $(b^{-1} \cdot g(x))$ dan $(a^{-1} \cdot h(x))$ semua bilangan bulat, yang berarti $f(x)$ *reducible* dalam $\mathbf{Z}[X]$. Jadi jika $f(x)$ *reducible* dalam $\mathbf{Q}[X]$ (sebagai *polynomial* bilangan rasional) maka $f(x)$ juga *reducible* dalam $\mathbf{Z}[X]$ (sebagai *polynomial* bilangan bulat). Selesailah pembuktian kita.

Kembali ke $f(x)$ berupa *monic polynomial* dengan *degree* $d$ yang *irreducible* sebagai *polynomial* bilangan bulat (dan didefinisikan berdasarkan $m$ dan $d$). Menurut *Gauss's Lemma 2* maka $f(x)$ juga *irreducible* sebagai *polynomial* bilangan rasional. Berdasarkan *Fundamental Theorem of Algebra* (lihat [fin97] atau cari di internet), $f(x)$ dapat diuraikan sebagai berikut:

$$f(x) = (x - \alpha_1)(x - \alpha_2) \cdots (x - \alpha_d)$$

dimana $\alpha_i \in \mathbf{C}$ (bilangan kompleks). Kita dapat memilih $\alpha = \alpha_i$ akar mana saja dan membuat *field* $\mathbf{Q}(\alpha)$ (lihat bagian 10.6). Elemen-elemen dari $\mathbf{Q}(\alpha)$ dapat direpresentasikan oleh kombinasi linear elemen-elemen

$$S = \{1, \alpha, \alpha^2, \ldots, \alpha^{d-1}\}.$$

Kita akan fokus pada subset dari $\mathbf{Q}(\alpha)$ yang dapat direpresentasikan oleh kombinasi $\mathbf{Z}$-linear elemen-elemen $S$ (jadi semua koefisien bilangan bulat). Subset ini membentuk suatu *ring* $\mathbf{Z}[\alpha]$ dan merupakan *subring* dari $\mathfrak{O}$, algebraic integers dalam $\mathbf{Q}(\alpha)$ (lihat bagian 12.4). Lalu bagaimana kita mendapatkan perbedaan kuadrat dari $\mathbf{Z}[\alpha]$? Karena $f(\alpha) = 0$ dan $f(m) \equiv 0 \pmod n$ maka terdapat *ring homomorphism* $\varphi$ dari $\mathbf{Z}[\alpha]$ ke $\mathbf{Z}/n\mathbf{Z}$ dimana:

- $i \mapsto [i]$ untuk $i \in \mathbf{Z}$ dan
- $\alpha \mapsto m \pmod n$.

Kita buat $U$ berupa himpunan *finite* dari pasangan bilangan bulat $(a, b)$ dimana:

1. Produk dari $a + \alpha b$ untuk setiap pasangan $(a, b)$ dalam $U$ merupakan kuadrat dalam $\mathbf{Z}[\alpha]$, sebut saja $\gamma^2$.
2. Produk dari $a + mb$ untuk setiap pasangan $(a, b)$ dalam $U$ merupakan kuadrat dalam $\mathbf{Z}$, sebut saja $v^2$.

Karena $\gamma$ dapat direpresentasikan sebagai *polynomial* dalam $\alpha$, dan $\alpha$ dipetakan ke $m$ oleh $\varphi$, kita bisa dapatkan bilangan bulat $u$ dimana $\varphi(\gamma) \equiv u \pmod n$. Jadi

$$\begin{aligned} u^2 &\equiv \varphi(\gamma)^2 \pmod n \\ &\equiv \varphi(\gamma^2) \pmod n \\ &\equiv \varphi\left(\prod_{(a,b)\in U} (a + \alpha b)\right) \pmod n \\ &\equiv \prod_{(a,b)\in U} \varphi(a + \alpha b) \pmod n \\ &\equiv \prod_{(a,b)\in U} (a + mb) \pmod n \\ &\equiv v^2 \pmod n. \end{aligned}$$

Setelah kita dapatkan $u^2 \equiv v^2 \pmod n$, jika $u \not\equiv \pm v \pmod n$ maka kita tahu apa yang harus dilakukan untuk mendapatkan penguraian $n$.

Serupa dengan metode *quadratic sieve*, penyaringan harus dilakukan terhadap pasangan $(a, b)$. Akan tetapi penyaringan harus dilakukan dari dua sisi:

- Berdasarkan nilai $a + mb$ dalam $\mathbf{Z}$.
- Berdasarkan nilai $a + \alpha b$ dalam $\mathbf{Z}[\alpha]$.

Dari sisi nilai $a+mb$ dalam $\mathbf{Z}$, penyaringan untuk mendapatkan produk berupa kuadrat dapat dilakukan serupa dengan metode *quadratic sieve*, tetapi menggunakan dua variabel: $a$ dan $b$. Dari sisi nilai $a + \alpha b$ dalam $\mathbf{Z}[\alpha]$, penyaringan lebih rumit. Pada prinsipnya kita dapat saja melakukan komputasi menggunakan *ring* $\mathbf{Z}[\alpha]$. Akan tetapi ini akan sangat tidak efisien dan kita akan menghadapi banyak kesulitan lainnya. Kita akan jelaskan cara penyaringan yang lebih efisien, akan tetapi, sebelum melanjutkan, kita cari rumus *norm* untuk $a + \alpha b$:

$$\begin{aligned}
N(a + \alpha b) &= \sigma_1(a + \alpha b)\sigma_2(a + \alpha b) \cdots \sigma_d(a + \alpha b) \\
&= (a + \alpha_1 b)(a + \alpha_2 b) \cdots (a + \alpha_d b) \\
&= b^d \left[ (\frac{a}{b} + \alpha_1)(\frac{a}{b} + \alpha_2) \cdots (\frac{a}{b} + \alpha_d) \right] \\
&= (-b)^d \left[ (-\frac{a}{b} - \alpha_1)(-\frac{a}{b} - \alpha_2) \cdots (-\frac{a}{b} - \alpha_d) \right] \\
&= (-b)^d f(-\frac{a}{b}).
\end{aligned}$$

Sebetulnya *norm* berlaku pada *field extension*, jadi *norm* yang kita maksud adalah *norm* pada $\mathbf{Q}(\alpha)/\mathbf{Q}$ dengan notasi $N_{\mathbf{Q}}^{\mathbf{Q}(\alpha)}$. Kita gunakan notasi $N$ karena lebih ringkas dan cukup jelas apa yang dimaksud.

Sekarang kita jelaskan cara penyaringan dari sisi nilai $a + \alpha b$ dalam $\mathbf{Z}[\alpha]$. Karena $\mathbf{Z}[\alpha]$ bukan seluruh *algebraic integers* dalam $\mathbf{Q}(\alpha)/\mathbf{Q}$, maka kita tidak akan gunakan elemen-elemen prima dalam $\mathbf{Z}[\alpha]$ sebagai *factor base*. Untuk *factor base* kita akan gunakan *prime ideals* tertentu dalam $\mathbf{Z}[\alpha]$ yang dinamakan *first degree prime ideals*. Suatu *ideal* dalam $\mathbf{Z}[\alpha]$ adalah suatu *first degree prime ideal* jika *norm* dari *ideal* tersebut adalah bilangan prima.

**Teorema 108** *Terdapat suatu bijective mapping antara first degree prime ideals dari* $\mathbf{Z}[\alpha]$ *dengan pasangan-pasangan* $(r, p)$ *dimana* $p$ *adalah bilangan prima,* $r \in \mathbf{Z}/p\mathbf{Z}$*, dan* $f(r) = 0 \pmod{p}$*.*

Jika $\mathfrak{p}$ merupakan *first degree prime ideal* dalam $\mathbf{Z}[\alpha]$ maka $|\mathbf{Z}[\alpha]/\mathfrak{p}| = p$ untuk suatu bilangan prima $p$, jadi

$$\mathbf{Z}[\alpha]/\mathfrak{p} \simeq \mathbf{Z}/p\mathbf{Z}.$$

Terdapat *canonical homomorphism* $\varphi : \mathbf{Z}[\alpha] \longrightarrow \mathbf{Z}[\alpha]/\mathfrak{p}$ yang *surjective* (*homomorphism* yang *surjective* dinamakan *epimorphism*) dan $\ker(\varphi) = \mathfrak{p}$. Karena $\mathbf{Z}[\alpha]/\mathfrak{p} \simeq \mathbf{Z}/p\mathbf{Z}$ maka $\varphi$ dapat dipandang sebagai *epimorphism*:

$$\varphi : \mathbf{Z}[\alpha] \longrightarrow \mathbf{Z}/p\mathbf{Z}$$

dengan $\ker(\varphi) = \mathfrak{p}$, jadi $\varphi(a) = a \pmod{p}$ untuk setiap bilangan bulat $a$. Jika $r = \varphi(\alpha) \in \mathbf{Z}/p\mathbf{Z}$ maka

$$\begin{aligned}
0 &\equiv \varphi(f(\alpha)) \\
&\equiv \varphi(\alpha^d + c_{d-1}\alpha^{d-1} + \ldots + c_1\alpha + c_0) \\
&\equiv \varphi(\alpha)^d + c_{d-1}\varphi(\alpha)^{d-1} + \ldots + c_1\varphi(\alpha) + c_0 \\
&\equiv r^d + c_{d-1}r^{d-1} + \ldots c_1 r + c_0 \\
&\equiv f(r) \pmod{p}
\end{aligned}$$

jadi $r$ merupakan akar dari $f(x) \pmod p$ dan *ideal* $\mathfrak{p}$ menentukan pasangan unik $(r,p)$. Sebaliknya, jika $p$ adalah bilangan prima dan $r \in \mathbf{Z}/p\mathbf{Z}$ dengan $f(r) \equiv 0 \pmod p$ maka terdapat *ring epimorphism*

$$\varphi : \mathbf{Z}[\alpha] \longrightarrow \mathbf{Z}/p\mathbf{Z}$$

dimana

$$\varphi(a) \equiv a \pmod p$$

untuk setiap $a \in \mathbf{Z}$, dan

$$\varphi(\alpha) \equiv r \pmod p.$$

Jika kita buat $\mathfrak{p} = \ker(\varphi)$ maka $\mathfrak{p}$ merupakan *ideal* dalam $\mathbf{Z}[\alpha]$. Karena $\varphi$ *surjective* dan $\ker(\varphi) = \mathfrak{p}$ maka

$$\mathbf{Z}[\alpha]/\mathfrak{p} \simeq \mathbf{Z}/p\mathbf{Z},$$

yang berarti $|\mathbf{Z}[\alpha]/\mathfrak{p}| = p$, jadi $\mathfrak{p}$ merupakan *first degree prime ideal* dalam $\mathbf{Z}[\alpha]$. Jadi pasangan $(r,p)$ menentukan *first degree prime ideal* yang unik dan selesailah pembuktian teorema 108.

Untuk mencari kuadrat berdasarkan *factor base* dalam $\mathbf{Z}[\alpha]$ kita dapat menggunakan vektor pemangkatan untuk $N(\langle a+\alpha b\rangle)$. Akan tetapi meskipun ini menghasilkan produk kuadrat dalam $\mathbf{Z}$, produk dalam $\mathbf{Z}[\alpha]$ belum tentu kuadrat, jadi diperlukan mekanisme tambahan. Kita jelaskan terlebih dahulu teori dibalik penggunaan vektor pemangkatan untuk $N(\langle a+\alpha b\rangle)$. Jika $a,b \in \mathbf{Z}$ dan $a$ koprima dengan $b$ ($\gcd(a,b) = 1$) dan pasangan $(r,p)$ merepresentasikan *first degree prime ideal*, maka kita definisikan $e_{r,p}(a+\alpha b)$ sebagai berikut:

$$e_{r,p}(a+\alpha b) = \begin{cases} \mathrm{ord}_p(N(\langle a+\alpha b\rangle)) & \text{jika } a+rb \equiv 0 \pmod p \\ 0 & \text{jika } a+rb \not\equiv 0 \pmod p \end{cases}$$

dimana $\mathrm{ord}_p(k)$ adalah banyaknya $p$ sebagai faktor dalam $k$. Tentunya

$$N(\langle a+\alpha b\rangle) = \prod_{r,p} p^{e_{r,p}(a+\alpha b)}$$

dimana produk meliputi semua pasangan $(r,p)$. Penggunaan $e_{r,p}(a+\alpha b)$ didasarkan pada pengamatan bahwa jika $U$ merupakan himpunan *finite* berbagai pasangan bilangan bulat $(a,b)$ dimana $a$ koprima dengan $b$, dan

$$\prod_{(a,b)\in U} (a+\alpha b)$$

merupakan kuadrat dalam $\mathbf{Q}(\alpha)$, maka untuk setiap pasangan $(r,p)$ kita dapatkan

$$\sum_{(a,b)\in U} e_{r,p}(a+\alpha b) \equiv 0 \pmod 2. \qquad (14.2)$$

Kita akan jelaskan persamaan 14.2. Jika $\mathbf{Z}[\alpha]$ sama dengan $\mathfrak{O}$ maka jelas apa yang dimaksud dengan $e_{r,p}(a+\alpha b)$ karena setiap $\beta \in \mathfrak{O}$ dapat diuraikan sepenuhnya menjadi produk prima dalam $\mathfrak{O}$. Akan tetapi biasanya $\mathbf{Z}[\alpha]$ tidak sama dengan $\mathfrak{O}$ jadi kita perlukan teorema berikut.

**Teorema 109** *Untuk setiap ideal prima $P$ dalam $\mathbf{Z}[\alpha]$ terdapat suatu group homomorphism $l_P : \mathbf{Q}(\alpha)^* \longrightarrow \mathbf{Z}$ dimana*

1. *$l_P(x) \geq 0$ untuk setiap $x \in \mathbf{Z}[\alpha]$, $x \neq 0$.*

2. *Jika $x \in \mathbf{Z}[\alpha]$, $x \neq 0$, maka $l_P(x) > 0$ jika dan hanya jika $x \in P$.*

3. *Untuk setiap $x \in \mathbf{Q}(\alpha)^*$, $l_P(x) = 0$ kecuali untuk beberapa $P$ yang banyaknya finite, dan kita dapatkan*

$$\prod_P (N(P))^{l_P(x)} = N(\langle x \rangle)$$

*dimana $P$ meliputi semua ideal prima dalam $\mathbf{Z}[\alpha]$.*

Untuk membuktikan teorema 109, kita definisikan terlebih dahulu fungsi $l_p$. Jika $P$ merupakan *ideal* prima dalam $\mathbf{Z}[\alpha]$, $x \in \mathbf{Z}[\alpha]$ dan $x \neq 0$, maka karena $x\mathbf{Z}[\alpha]$ mempunyai *finite index* dalam $\mathbf{Z}[\alpha]$, terdapat *finite chain*

$$\mathbf{Z}[\alpha] = I_0 \supset I_1 \supset I_2 \cdots I_{t-1} \supset I_t = x\mathbf{Z}[\alpha]$$

terdiri dari *ideals* yang berbeda, dan untuk $1 \leq i \leq t$, tidak terdapat *ideal* $J$ dimana $I_{i-1} \supset J \supset I_i$. Kita definisikan $l_p(x)$ sebagai banyaknya $i \in \{1, 2, \ldots, t\}$ dimana

$$I_{i-1}/I_i \simeq \mathbf{Z}[\alpha]/P$$

sebagai $\mathbf{Z}[\alpha]$-*module*. Berdasarkan teorema Jordan-Hölder (lihat [wae66] bagian 51, atau cari di internet), $l_p(x)$ *well-defined* karena tidak tergantung pada bagaimana *finite chain* dipilih. Jika $0 \neq y \in \mathbf{Z}[\alpha]$ maka *finite chain* untuk $x$ dapat disambung dengan *finite chain* untuk $y$:

$$\mathbf{Z}[\alpha] = J_0 \supset J_1 \supset J_2 \cdots J_{u-1} \supset J_u = y\mathbf{Z}[\alpha]$$

dan kita dapatkan *finite chain* untuk $xy$:

$$\mathbf{Z}[\alpha] = I_0 \supset I_1 \cdots I_t = xJ_0 \supset xJ_1 \cdots xJ_u = xy\mathbf{Z}[\alpha].$$

Jadi $l_p(xy) = l_p(x) + l_p(y)$. Dengan mendefinisikan $l_p(x/z) = l_p(x) - l_p(z)$ untuk $x, z \in \mathbf{Z}[\alpha]$ dimana $x$ dan $z$ tidak sama dengan 0, kita dapat memperluas *domain* $l_p$ menjadi $\mathbf{Q}(\alpha)^*$. Sangat jelas bahwa bagian 1 dari teorema 109 berlaku. Untuk membuktikan $l_p(x) > 0 \Longleftarrow x \in P$ di bagian 2, kita buat $I_1 =$

$P$. Untuk membuktikan $l_p(x) > 0 \Longrightarrow x \in P$, kita lihat apa konsekuensinya jika $x \notin P$. Karena $P$ maksimal, maka *ideal* $x\mathbf{Z}[\alpha] + P = \mathbf{Z}[\alpha]$. Jadi

$$\exists y \in \mathbf{Z}[\alpha], z \in P : xy + z = 1.$$

Efeknya mengalikan dengan $z$ adalah *identity map* $\mathbf{Z}[\alpha]/x\mathbf{Z}[\alpha] \longrightarrow \mathbf{Z}[\alpha]/x\mathbf{Z}[\alpha]$. Akibatnya,

$$z \cdot (I_{i-1}/I_i) = (I_{i-1}/I_i),$$

dan karena $z \in P$ maka $I_{i-1}/I_i$ tidak bisa *isomorphic* dengan $\mathbf{Z}[\alpha]/x\mathbf{Z}[\alpha]$, jadi $l_p(x) = 0$. Untuk membuktikan bagian 3, karena $l_p(x) = 0$ jika $x \notin \mathbf{Z}[\alpha]$, maka kita tinggal menunjukkan persamaan untuk $0 \neq x \in \mathbf{Z}[\alpha]$. Karena

$$|N(x)| = |\mathbf{Z}[\alpha]/x\mathbf{Z}[\alpha]| = \prod_{i=1}^{t} |I_{i-1}/I_i|$$

maka kita tinggal tunjukkan bahwa untuk setiap $i$ terdapat *ideal* prima $P$ yang unik dimana $I_{i-1}/I_i \simeq \mathbf{Z}[\alpha]/P$. Kita pilih $y \in I_{i-1}$ dimana $y \notin I_i$. Karena tidak terdapat *ideal* $J$ dimana

$$I_{i-1} \supset J \supset I_i$$

maka $y\mathbf{Z}[\alpha]+I_i = I_{i-1}$, jadi efek dari perkalian dengan $y$ adalah suatu *surjective map* $\mathbf{Z}[\alpha] \longrightarrow I_{i-1}/I_i$. Jadi terdapat suatu *ideal* $P$ dimana

$$\mathbf{Z}[\alpha]/P \simeq I_{i-1}/I_i.$$

Karena $\mathbf{Z}[\alpha]/P$ tidak memiliki *non-trivial submodule* maka $P$ maksimal, yang berarti $P$ prima. Juga, karena $P$ merupakan *annihilator* untuk $I_{i-1}/I_i$ sebagai $\mathbf{Z}[\alpha]$-*module*:

$$P = \{r \in \mathbf{Z}[\alpha] \,|\, \forall m \in I_{i-1}/I_i : rm = 0\},$$

maka $P$ unik. Selesailah pembuktian teorema 109.

Sebagai konsekuensi dari teorema 109 kita dapatkan teorema berikut.

**Teorema 110** *Untuk $a$ dan $b$ dua bilangan bulat yang koprima dan $P$ suatu ideal prima dalam $\mathbf{Z}(\alpha)$:*

- *Jika $P$ bukan first degree prime ideal maka $l_P(a + \alpha b) = 0$.*
- *Jika $P$ adalah first degree prime ideal yang direpresentasikan dengan pasangan $(r, p)$ maka $l_P(a + \alpha b) = e_{r,p}(a + \alpha b)$.*

Mari kita buktikan teorema 110. Jika $P$ merupakan *ideal* prima dalam $\mathbf{Z}[\alpha]$ dengan $l_P(a + \alpha b) > 0$, maka berdasarkan teorema 109, $a + \alpha b$ dipetakan ke 0 oleh *canonical homomorphism* $\mathbf{Z}[\alpha] \longrightarrow \mathbf{Z}[\alpha]/P$. Terdapat bilangan prima $p$

dalam $P$. Jika $p$ membagi $b$, maka $\alpha b$ juga dipetakan ke 0, dengan demikian $a$ juga dipetakan ke 0, jadi $p$ membagi $a$, suatu kontradiksi dengan $\gcd(a, b) = 1$. Jadi $b$ dipetakan ke elemen $b' \neq 0$ dalam $\mathbf{Z}[\alpha]/P$. Bukan itu saja, $b' \equiv b \pmod{p}$ berada didalam *prime field* $F_p$ dan mempunyai *inverse* $b'^{-1}$. Demikian juga $a$ dipetakan ke elemen $a' \in F_p$. Karena $a + \alpha b$ dipetakan ke 0, maka $\alpha$ dipetakan ke $a'b'^{-1}$ yang merupakan elemen dari $F_p$. Jadi seluruh $\mathbf{Z}[\alpha]$ dipetakan ke $F_p$ yang berarti $P$ merupakan *first degree prime ideal.* Selesailah pembuktian bagian pertama. Untuk bagian kedua, kita gunakan

$$\prod_p (N(P))^{l_P(\beta)} = N(\langle\beta\rangle)$$

dari teorema 109 dan periksa pangkat dari $p$ di kedua sisi persamaan. Selesailah pembuktian teorema 110.

Sekarang kita bisa dapatkan persamaan 14.2. Kita buat

$$\prod_{(a,b)\in U} a + \alpha b = \gamma^2$$

dan $P$ merupakan *first degree prime ideal* dengan representasi $(r, p)$. Jadi

$$\begin{aligned}
\sum_{(a,b)\in U} e_{r,p}(a + \alpha b) &= \sum_{(a,b)\in U} l_P(a + \alpha b) \\
&= l_P\left(\prod_{(a,b)\in U} (a + \alpha b)\right) \\
&= l_P(\gamma^2) \\
&= 2l_P(\gamma) \\
&\equiv 0 \pmod{2}.
\end{aligned}$$

Teknik diatas membuat perumpamaan bahwa $\mathbf{Z}[\alpha]$ adalah suatu *Dedekind domain*, contohnya jika $\mathbf{Z}[\alpha] = \mathfrak{O}$. Biasanya $\mathbf{Z}[\alpha] \subset \mathfrak{O}$ dan $\mathbf{Z}[\alpha]$ bukan merupakan *Dedekind domain.* Jadi tidak ada jaminan bahwa produk merupakan kuadrat dalam $\mathbf{Z}[\alpha]$. Untuk lebih memberi jaminan, meskipun tidak 100 persen, digunakan *quadratic characters.* Sebagai motivasi untuk konsep *quadratic characters*, kita gunakan contoh yang sederhana yaitu kuadrat dalam $\mathbf{Z}$. Jika $x$ adalah kuadrat dalam $\mathbf{Z}$ ($x = y^2$, $x, y \in \mathbf{Z}$), maka $x$ juga merupakan *perfect square* modulo setiap bilangan prima. Jadi menggunakan simbol Legendre,

$$\left(\frac{x}{p}\right) = 1$$

untuk setiap bilangan prima $p$. Fakta ini dapat digunakan sebagai test untuk mengetahui apakah suatu bilangan bulat $z$ merupakan kuadrat dalam $\mathbf{Z}$. Jika kita test $\left(\frac{z}{p}\right)$ dengan beberapa bilangan prima dan hasilnya semua 1 maka besar kemungkinan bahwa $z$ merupakan kuadrat. Semakin banyak bilangan prima yang digunakan, semakin besar jaminan bahwa $z$ merupakan kuadrat. Konsep ini dapat digeneralisasi untuk test kuadrat dalam $\mathbf{Q}(\alpha)$.

**Teorema 111** *Jika $U$ merupakan himpunan pasangan $(a, b)$ dimana*

$$\prod_{(a,b)\in U} (a + \alpha b) = \beta^2$$

*untuk suatu $\beta \in \mathbf{Q}(\alpha)$, dan $\mathfrak{p}$ adalah suatu first degree prime ideal dengan representasi $(r, p)$ dimana*

$$a + rb \not\equiv 0 \pmod{p} \text{ untuk setiap } (a, b) \in U,$$
$$f'(r) \not\equiv 0 \pmod{p}$$

*maka*

$$\prod_{(a,b)\in U} \left(\frac{a + rb}{p}\right) = 1.$$

Pembuktian teorema 111 menggunakan fakta bahwa jika

$$\prod_{(a,b)\in U} (a + \alpha b) = \beta^2$$

dimana $\beta \in \mathbf{Q}(\alpha)$, maka $\beta \in \mathfrak{D}$ dan $\beta f'(\alpha) \in \mathbf{Z}[\alpha]$. Kita tidak akan buktikan fakta ini disini, pembaca yang berminat dipersilahkan membaca [wei63] (Proposition 3-7-14). Pertama, kita buat *canonical ring epimorphism* $\varphi$

$$\varphi : \mathbf{Z}[\alpha] \longrightarrow \mathbf{Z}/p\mathbf{Z}$$

dengan $\varphi(\alpha) \equiv r \pmod{p}$. Jadi $\mathfrak{p} = \ker(\varphi)$ adalah suatu *first degree prime ideal*. Karena $\varphi$ memetakan elemen-elemen diluar $\mathfrak{p}$ ke elemen-elemen yang bukan 0, kita dapat membuat pemetaan

$$\begin{aligned} \chi_{\mathfrak{p}} : \mathbf{Z}[\alpha] - \mathfrak{p} &\longrightarrow \{-1, 1\} \\ \delta &\mapsto \left(\frac{\varphi(\delta)}{p}\right). \end{aligned}$$

Menggunakan fakta diatas, terdapat suatu $\gamma = \beta f'(\alpha) \in \mathbf{Z}[\alpha]$ dimana

$$f'(\alpha)^2 \prod_{(a,b)\in U} (a + \alpha b) = \gamma^2.$$

Karena $\langle a + \alpha b\rangle$ tidak dapat dibagi oleh $\mathfrak{p}$ berarti $a + \alpha b \notin \mathfrak{p}$. Demikian juga, berdasarkan hipotesis dalam teorema, $f'(r)$ tidak dapat dibagi oleh $p$, jadi

$f'(\alpha)^2 \notin \mathfrak{p}$. Akibatnya $\langle\gamma^2\rangle$ tidak dapat dibagi dengan $\mathfrak{p}$, demikian juga $\langle\gamma\rangle$, jadi $\chi_\mathfrak{p}$ berlaku untuk $\gamma^2$ dan $\gamma$. Menggunakan simbol Legendre, kita dapatkan

$$\chi_\mathfrak{p}(\gamma^2) = \left(\frac{\varphi(\gamma^2)}{p}\right) = \left(\frac{\varphi(\gamma)\varphi(\gamma)}{p}\right) = \left(\frac{\varphi(\gamma)}{p}\right)^2 = 1.$$

Demikian juga kita dapatkan $\chi_\mathfrak{p}(f'(\alpha)^2) = 1$. Jadi, menggunakan simbol Legendre, kita dapatkan

$$\begin{aligned}
1 &= \chi_\mathfrak{p}(\gamma^2) \\
&= \chi_\mathfrak{p}\left(f'(\alpha)^2 \prod_{(a,b)\in U}(a+\alpha b)\right) \\
&= \left(\frac{\varphi(f'(\alpha)^2 \prod_{(a,b)\in U}(a+\alpha b))}{p}\right) \\
&= \left(\frac{\varphi(f'(\alpha)^2) \prod_{(a,b)\in U}\varphi(a+\alpha b)}{p}\right) \\
&= \left(\frac{\varphi(f'(\alpha)^2)}{p}\right)\left(\frac{\prod_{(a,b)\in U}\varphi(a+\alpha b)}{p}\right) \\
&= \chi_\mathfrak{p}(\varphi(f'(\alpha)^2))\left(\frac{\prod_{(a,b)\in U}(a+rb)}{p}\right) \\
&= \prod_{(a,b)\in U}\left(\frac{a+rb}{p}\right).
\end{aligned}$$

Selesailah pembuktian teorema 111. Test kuadrat sesuai dengan teorema 111 dapat digunakan untuk meningkatkan jaminan bahwa produk menghasilkan kuadrat. Tentunya jika ternyata kuadrat tidak ditemukan, produk lain harus dicari.

Mari kita ringkas bagaimana apa yang sudah dijelaskan mengenai metode *number field sieve* digunakan untuk menguraikan suatu bilangan $n$:

1. Berdasarkan nilai $n$, suatu *irreducible polynomial* dengan *degree* $d$ dipilih.

2. Penyaringan pertama menggunakan *rational factor base* yaitu sekumpulan bilangan prima.

3. Penyaringan kedua menggunakan *algebraic factor base* yaitu sekumpulan *first degree prime ideals* yang direpresentasikan menggunakan pasangan-pasangan $(r, p)$.

4. Penyaringan ketiga dengan *quadratic characters* sesuai dengan teorema 111 menggunakan sekumpulan *first degree prime ideals*.

5. Setelah kuadrat-kuadrat ditemukan, kedua faktor dari $n$ didapat menggunakan *difference of squares*.

Kompleksitas metode *number field sieve*, diestimasikan adalah

$$O(e^{C(\log n)^{1/3}(\log\log n)^{2/3}}).$$

Implementasi dari metode *number field sieve* tidak dijelaskan disini. Untuk pembaca yang ingin mengetahui lebih rinci mengenai implementasi dan estimasi kompleksitas dari metode ini dipersilahkan untuk membaca [buh93].

## 14.7 Ringkasan

Bab ini telah membahas penguraian bilangan bulat, topik yang sangat penting untuk kriptografi *public key*. Metode untuk menguraikan bilangan bulat dapat digolongkan menjadi dua kategori: metode yang bersifat Las Vegas dan metode yang menggunakan *factor base*. Contoh metode yang bersifat Las Vegas adalah metode Rho, sedangkan untuk yang menggunakan *factor base* telah dibahas metode Dixon, metode *continued fraction*, metode *quadratic sieve* dan metode *number field sieve*. Untuk menguraikan bilangan hingga 100 digit, metode *quadratic sieve* adalah yang tercepat, sedangkan untuk menguraikan bilangan lebih dari 100 digit hingga lebih dari 150 digit, metode *number field sieve* adalah yang tercepat. Secara umum, penguraian bilangan yang lebih besar dari 200 digit masih belum terjangkau. Akan tetapi pada tanggal 12 Desember 2009, sekelompok peneliti berhasil menguraikan kunci RSA sebesar 768 bit (232 digit) menggunakan metode *number field sieve* [kle10]. Penguraian tersebut memakan waktu $2\frac{1}{2}$ tahun menggunakan ratusan komputer yang tersebar di beberapa negara. Pencarian *polynomial* memakan waktu 6 bulan, *sieving* memakan waktu 2 tahun dan lainnya sekitar 2 minggu.

Banyak teknik-teknik implementasi yang tidak dibahas dalam bab ini. Untuk mengimplementasi metode penguraian bilangan bulat tentunya pembaca perlu mempelajari teknik-teknik tersebut. Misalnya untuk metode yang menggunakan *factor base*, algoritma *Block Lanczos* dapat digunakan untuk komputasi matrik.