Bab 15

Matematika VIII - Logaritma Diskrit

Jika $\mathbf{GF}(q)$ merupakan finite field dan $u, g \in \mathbf{GF}(q)^*$ (g biasanya *generator* untuk $\mathbf{GF}(q)^*$), maka logaritma diskrit dari u dengan basis g dalam $\mathbf{GF}(q)^*$ adalah bilangan bulat k, $0 \leq k < q-1$, dimana

$$u = g^k.$$

Sukarnya mengkalkulasi logaritma diskrit jika *finite field* sangat besar merupakan kunci dari keamanan berbagai algoritma kriptografi *public key* seperti Diffie-Hellman, DSA dan ElGamal. Sukarnya mengkalkulasi logaritma diskrit mirip dengan sukarnya menguraikan bilangan bulat yang sangat besar, bahkan perkembangan metode untuk mengkalkulasi logaritma diskrit saling mempengaruhi dengan perkembangan metode penguraian. Kita akan bahas tiga metode untuk kalkulasi logaritma diskrit yaitu metode Silver-Pohlig-Hellman, metode *baby steps - giant steps* dan metode *index calculus*.

15.1 Metode Silver-Pohlig-Hellman

Untuk bilangan prima p, $\mathbf{GF}(p)$ merupakan suatu *finite field*. Jika $p-1$ dapat diuraikan menjadi produk bilangan-bilangan prima yang kecil, maka logaritma diskrit dalam $\mathbf{GF}(p)^*$ dapat dikalkulasi dengan cepat menggunakan metode Silver-Pohlig-Hellman. Metode ini menggunakan fakta bahwa karena $\mathbf{GF}(p)^*$ mempunyai $p-1$ elemen, maka aritmatika *multiplicative group* untuk pangkat suatu elemen dari $\mathbf{GF}(p)^*$ dapat dilakukan modulo $p-1$. Berikut adalah bagaimana metode Silver-Pohlig-Hellman mencari solusi untuk k, $0 \leq k < p-1$, jika g *generator* untuk $\mathbf{GF}(p)^*$ dan $u = g^k$.

Langkah pertama dalam metode Silver-Pohlig-Hellman adalah mengkomputasi, untuk setiap bilangan prima p_i yang membagi $p-1$, akar-akar pangkat p_i dari 1,

$$r_{p_i,j} = g^{j(p-1)/p_i},$$

dimana g adalah *generator* untuk $\mathbf{GF}(p)^*$ dan $j = 0, 1, \ldots, p_i - 1$. Kita simpan $\{r_{p_i,j}\}_{0 \le j < p_i}$ dalam sebuah tabel.

Langkah berikutnya adalah mencari solusi untuk k modulo $p_i^{\alpha_i}$ untuk setiap bilangan prima p_i yang membagi $p-1$, dimana α_i adalah banyaknya p_i (pangkat dari p_i) dalam uraian $p-1$. Kita dapat tuliskan

$$k \equiv k_0 + k_1 p_i + k_2 p_i^2 + \ldots + k_{\alpha_i - 1} p_i^{\alpha_i - 1} \pmod{p_i^{\alpha_i}}.$$

Jika kita bisa temukan $k_0, k_1, \ldots, k_{\alpha_i - 1}$ maka kita akan dapatkan nilai untuk $k \pmod{p_i^{\alpha_i - 1}}$. Kita akan gunakan aritmatika $\mathbf{GF}(p)$ dalam mencari k_0. Pertama, kita kalkulasi $u^{(p-1)/p_i}$ untuk mendapatkan akar pangkat p_i dari 1 (karena $u^{p-1} = 1$). Karena $u = g^k$ dan $(k_1 p_i + k_2 p_i^2 + \ldots + k_{\alpha_i - 1} p_i^{\alpha_i - 1})/p_i$ adalah suatu bilangan bulat, kita dapatkan

$$\begin{aligned} u^{(p-1)/p_i} &= g^{k(p-1)/p_i} \\ &= g^{(k_0 + k_1 p_i + k_2 p_i^2 + \ldots + k_{\alpha_i - 1} p_i^{\alpha_i - 1})(p-1)/p_i} \\ &= g^{k_0 (p-1)/p_i} \\ &= r_{p_i, k_0}. \end{aligned}$$

Jadi kita dapat periksa tabel $\{r_{p_i,j}\}_{0 \le j < p_i}$ untuk mencari m dimana $r_{p_i,m} = r_{p_i,k_0}$ dan mendapatkan $k_0 = m$. Untuk mencari k_1, kita buat $u_1 = u/g^{k_0}$, jadi logaritma diskrit dari u_1 adalah

$$k - k_0 \equiv k_1 p_i + k_2 p_i^2 + \ldots + k_{\alpha_i - 1} p_i^{\alpha_i - 1} \pmod{p_i^{\alpha_i}}.$$

Karena u_1 adalah pemangkatan dengan p_i, $u_1^{(p-1)/p_i} = 1$ maka kita dapatkan

$$\begin{aligned} u_1^{(p-1)/p_i} &= g^{(k-k_0)(p-1)/p_i} \\ &= g^{(k_1 + k_2 p_i + \ldots + k_{\alpha_i - 1} p_i^{\alpha_i - 2})(p-1)/p_i} \\ &= g^{k_1 (p-1)/p_i} \\ &= r_{p_i, k_1}. \end{aligned}$$

Lagi, kita gunakan tabel $\{r_{p_i,j}\}_{0 \le j < p_i}$ untuk mendapatkan k_1. Jadi pola untuk mencari $k_0, k_1, \ldots, k_{\alpha_i - 1}$ sudah jelas. Untuk setiap $j = 1, 2, \ldots, \alpha_i - 1$, kita buat

$$u_j = u/g^{k_0 + k_1 p_i + \ldots + k_{j-1} p_i^{j-1}},$$

yang mempunyai logaritma diskrit ekuivalen dengan $k_j p_i^j + \ldots + k_{\alpha_i - 1} p_i^{\alpha_i - 1} \pmod{p_i^{\alpha_i}}$. Karena u_j merupakan pemangkatan dengan p_i^j, maka $u_j^{(p-1)/p_i^j} = 1$ dan kita dapatkan

$$\begin{aligned} u_j^{(p-1)/p_i^{j+1}} &= g^{(k_j + k_{j+1} p_i + \ldots + k_{\alpha_i - 1} p_i^{\alpha_i - j - 1})(p-1)/p_i} \\ &= g^{k_j (p-1)/p_i} \\ &= r_{p_i, k_j}. \end{aligned}$$

Kita cari k_j menggunakan tabel. Kita gabungkan hasilnya untuk mendapatkan

$$k \equiv k_0 + k_1 p_i + k_2 p_i^2 + \ldots + k_{\alpha_i - 1} p_i^{\alpha_i - 1} \pmod{p_i^{\alpha_i}}.$$

Langkah terahir setelah mendapatkan k modulo $p_i^{\alpha_i}$ untuk setiap p_i yang membagi p adalah menggabungkan hasil menggunakan *Chinese Remainder Theorem* (lihat bagian 10.2) untuk mendapatkan k modulo p.

Sebagai contoh penggunaan metode Silver-Pohlig-Hellman mari kita cari logaritma diskrit dari 28 dengan basis 2 dalam $\mathbf{GF}(37)^*$ (2 adalah *generator* untuk $\mathbf{GF}(37)^*$). Jadi kita ingin solusi untuk k dimana

$$28 \equiv 2^k \pmod{37}.$$

Karena $p = 37$ maka

$$p - 1 = 37 - 1 = 2^2 \cdot 3^2.$$

Jadi $p_1 = 2$, $\alpha_1 = 2$, $p_2 = 3$ dan $\alpha_2 = 2$. Untuk langkah pertama, kita dapatkan

$$\begin{aligned} 2^0 &\equiv 1 \pmod{37}, \\ 2^{36/2} &\equiv 36 \pmod{37}, \\ 2^0 &\equiv 1 \pmod{37}, \\ 2^{36/3} &\equiv 26 \pmod{37}, \\ 2^{36 \cdot 2/3} &\equiv 10 \pmod{37}. \end{aligned}$$

Jadi langkah pertama menghasilkan tabel

$$\begin{aligned} r_{2,0} &= 1, \\ r_{2,1} &= 36, \\ r_{3,0} &= 1, \\ r_{3,1} &= 26, \\ r_{3,2} &= 10. \end{aligned}$$

Berikutnya untuk $p_1 = 2$ kita kalkulasi $28^{36/2} \equiv 1 \pmod{37}$. Dari tabel kita dapatkan $r_{2,0}$, jadi $k_0 = 0$. Kemudian kita kalkulasi $28^{36/4} \equiv 36 \pmod{37}$, dan dari tabel kita dapatkan $k_1 = 1$. Jadi $k = 0 + 1 \cdot 2 = 2$ dan kita dapatkan

$$k \equiv 2 \pmod{4}.$$

Berikutnya untuk $p_2 = 3$ kita kalkulasi $28^{36/3} \equiv 26 \pmod{37}$. Dari tabel kita dapatkan $k_0 = 1$. Kemudian kita kalkulasi $14^{36/9} \equiv 10 \pmod{37}$, dan dari tabel kita dapatkan $k_1 = 2$. Jadi $k = 1 + 2 \cdot 3 = 7$ dan kita dapatkan

$$k \equiv 7 \pmod{9}.$$

Menggunakan *Chinese Remainder Theorem* kita dapatkan $k = 34$, jadi

$$28 \equiv 2^{34} \pmod{37}.$$

Tentunya contoh diatas tidak mencerminkan potensi metode karena hanya melibatkan bilangan-bilangan yang kecil. Metode Silver-Pohlig-Hellman cukup efisien jika setiap bilangan prima yang membagi $p-1$ relatif kecil, meskipun p sendiri sangat besar. Akan tetapi jika ada bilangan prima yang besar (contohnya 50 digit) yang membagi $p-1$ maka metode Silver-Pohlig-Hellman tidak akan berdaya karena perlu membuat tabel yang sangat besar untuk langkah pertama.

Ada varian metode Silver-Pohlig-Hellman yang menggunakan metode *baby steps - giant steps* untuk mencari $k_0, k_1, \ldots, k_{\alpha_i - 1}$. Untuk mencari k_0, kita buat

$$v = u^{p-1/p_i} = (g^{p-1/p_i})^{k_0} = h^{k_0}$$

dimana $h = g^{p-1/p_i}$. Kita pecahkan $v = h^{k_0}$ menggunakan metode *baby steps - giant steps* dimana *order* dari h adalah $n = (p-1)/p_i$.

15.2 Metode Baby Steps - Giant Steps

Jika $v, h \in \mathbf{GF}(p)^*$, h mempunyai *order* n dan $v \equiv h^a \pmod{p}$, maka metode *baby steps - giant steps* mencari a dimana $0 \leq a < n$ sebagai berikut.

- Kita buat $m = \lceil \sqrt{n} \rceil$. Maka terdapat bilangan-bilangan bulat a_0 dan a_1 yang menjadikan $a = a_0 + ma_1$ dimana $0 \leq a_0, a_1 < m$ (a_0 adalah *remainder* dan a_1 adalah *quotient*, lihat teorema 4).
- Komputasi *baby steps* $h^i \pmod{p}$ untuk $i = 0, 1, \ldots, m-1$. Hasil komputasi disimpan dalam struktur yang memudahkan pencarian nilai.
- Komputasi $u \equiv h^{-m} \pmod{p}$.
- Komputasi *giant steps* $vu^j \pmod{p}$ untuk $j = 0, 1, \ldots, m-1$. Periksa nilai apakah ada dalam hasil *baby steps*. Jika ada maka kita selesai.

Jika nilai *giant step* ditemukan dalam *baby steps* maka kita dapatkan

$$h^i \equiv v/h^{mj} \pmod{p}$$

jadi kita dapatkan nilai a yaitu $a = i + mj$.

Sebagai contoh penggunaan metode *baby steps - giant steps*, ambil contoh dari bagian 15.1 dan gunakan metode ini dalam mencari k_1 untuk $p_2 = 3$, dimana $v = 14^4 \equiv 10 \pmod{37}$, $h = 2^{12} \equiv 26 \pmod{37}$ dan $n = 2$ (jadi $m = 2$). Kita harus mencari solusi k_1 untuk

$$10 \equiv 26^{k_1} \pmod{37}.$$

Komputasi *baby steps* $h^i \pmod{37}$ menghasilkan

$$\begin{aligned} h^0 &\equiv 1 \pmod{37}, \\ h^1 &\equiv 26 \pmod{37}. \end{aligned}$$

Komputasi $u \equiv h^{-m} \pmod{37}$ menghasilkan

$$u \equiv 10^{-1} \equiv 26 \pmod{37}.$$

Komputasi *giant steps* vu^j menghasilkan

$$\begin{aligned} vu^0 &\equiv 10 \pmod{37}, \\ vu^1 &\equiv 1 \pmod{37}. \end{aligned}$$

Dari *baby step* pertama dan *giant step* terahir kita dapatkan $i = 0$ dan $j = 1$. Jadi

$$k_1 = 0 + 2 \cdot 1 = 2.$$

Hasil ini sesuai dengan yang kita dapatkan di bagian 15.1.

Metode *baby steps - giant steps* tidak akan efektif jika *order* dari basis (n) sangat besar. Contohnya jika n besarnya melebihi 200 digit (jadi $\sqrt{n}$ melebihi 100 digit), maka meskipun setiap partikel fundamental dapat digunakan untuk menyimpan data, jumlah partikel fundamental di seantero jagat raya (estimasi antara 10^{72} sampai dengan 10^{87}) tidak akan cukup untuk menyimpan *baby steps*.

15.3 Metode Index Calculus

Metode *index calculus* untuk mengkalkulasi logaritma diskrit sangat mirip dengan metode-metode untuk penguraian bilangan bulat yang menggunakan *factor base*. Karena kemiripan ini, kita hanya akan bahas "kerangka" dari metode ini dan fokus pada perbedaannya. Perbedaan utamanya adalah jika dalam penguraian menggunakan *factor base* kita menggunakan matrik dimana setiap baris merepresentasikan produk pemangkatan elemen-elemen *factor base*, dalam metode *index calculus* setiap baris dalam matrik merepresentasikan relasi

antara produk pemangkatan elemen-elemen *factor base* dengan pemangkatan basis. Sebagai contoh, baris

$$[1 \quad 0 \quad 1 \quad 2 \quad 17]$$

merepresentasikan relasi

$$p_1 p_3 p_4^2 \equiv g^{17},$$

dimana p_1, p_2, p_3, p_4 adalah elemen-elemen *factor base* dan g adalah basis untuk logaritma diskrit.

Untuk memudahkan pembahasan, kita akan fokus pada $\mathbf{GF}(p)$ dengan p bilangan prima, sehingga setiap elemen *factor base* adalah bilangan prima (kadang -1 diperbolehkan). Untuk $\mathbf{GF}(p^n)$, kita dapat menggunakan *irreducible polynomials* sebagai elemen-elemen *factor base*.

Kita mulai penjelasan metode *index calculus*, dimana

$$y \equiv g^x \pmod{p}$$

dan x harus dicari untuk suatu y. Langkah pertama yang harus dilakukan adalah menentukan *factor base*. Contohnya, kita dapat memilih himpunan n bilangan prima pertama $\{p_1, p_2, \ldots, p_n\}$ sebagai *factor base*. Komputasi logaritma diskrit terdiri dari dua tahap yaitu tahap prekomputasi dan tahap komputasi.

Pada tahap pertama, data logaritma diskrit untuk elemen-elemen *factor base* dihimpun dalam sebuah matrik. Ini mirip dengan metode Silver-Pohlig-Hellman dimana data dihimpun dalam suatu tabel. Tahap ini diawali dengan matrik yang kosong. Untuk beberapa k, dengan $1 \leq k < p$ (k dapat dipilih secara sembarang atau kita dapat menggunakan $k = 1, 2, \ldots$), kita ulang langkah-langkah berikut hingga banyaknya baris dalam matrik sama dengan banyaknya bilangan prima dalam *factor base* (r):

- Coba uraikan $g^k \pmod{p}$ menggunakan *factor base* menjadi $p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$.
- Jika berhasil diuraikan dan hasilnya *linearly independent* dengan matrik, maka tambahkan hasil sebagai baris dalam matrik.
- Jika banyaknya baris sama dengan r, kita selesai.

Penguraian menggunakan *factor base* adalah hal yang mudah karena hanya perlu membagi dengan elemen-elemen *factor base*. Setelah itu matrik dibuat menjadi *reduced echelon form*. Sebagai contoh, berikut adalah suatu matrik dalam *reduced echelon form*:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 & 9 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 & 6 \end{bmatrix}.$$

Elemen pertama pada kolom terahir adalah $\log_g(p_1)$ (logaritma diskrit untuk p_1), elemen kedua adalah $\log_g(p_2)$, dan seterusnya. Selesailah tahap pertama. Tahap ini hanya perlu dilakukan satu kali untuk berbagai logaritma diskrit dengan basis dan modulus yang sama.

Pada tahap kedua, logaritma diskrit dari y yaitu x dicari. Kita cari menggunakan beberapa s, dengan $0 \leq s < p$ (s dapat dipilih secara sembarang atau kita dapat menggunakan $s = 0, 1, 2, \ldots$) sehingga $g^s y$ dapat diuraikan sebagai berikut:

$$g^s y \equiv p_1^{f_1} p_2^{f_2} \cdots p_r^{f_r}.$$

Kita dapatkan x:

$$x = f_1 \log_g(p_1) + f_2 \log_g(p_2) + \ldots f_r \log_g(p_r) - s.$$

Sebagaimana halnya dengan penguraian menggunakan *factor base*, tidak ada jaminan secara umum bahwa logaritma diskrit dapat dicari dengan efisien menggunakan metode *index calculus*.

15.4 Ringkasan

Bab ini telah membahas logaritma diskrit, topik yang sangat penting untuk kriptografi *public key*. Sukarnya mengkalkulasi logaritma diskrit mirip dengan sukarnya menguraikan bilangan bulat. Bahkan teknik-teknik yang digunakan untuk mengkalkulasi logaritma diskrit diadaptasi untuk penguraian bilangan bulat dan sebaliknya. Logaritma diskrit untuk $\mathbf{GF}(2^n)$ lebih mudah untuk dikalkulasi dibandingkan $\mathbf{GF}(p^n)$ dimana p adalah bilangan prima ganjil (lihat [cop84]). Oleh sebab itu penggunaan $\mathbf{GF}(2^n)$ dalam kriptografi *public key* tidak direkomendasikan kecuali n sangat besar (> 2000).

Banyak teknik-teknik implementasi yang tidak dibahas dalam bab ini. Untuk mengimplementasi metode logaritma diskrit tentunya pembaca perlu mempelajari teknik-teknik tersebut. Misalnya untuk metode *index calculus*, algoritma *Block Lanczos* dapat digunakan untuk komputasi matrik.