Metoda index calculus

$GF(23)^*$ z generatorem 5.
Izberi bazo ‘majhnih’ faktorjev: $B = \{-1, 2, 3\}$
in sestavi tabelo njihovih logaritmov:

elt	-1	2	3
log	11	2	16

Iščemo logaritem elementa β (Las Vegas).
Poišči ‘gladko’ potenco elementa β,
tj. β^x, ki se da razstaviti na faktorje iz B.

Izračunaj log(13): $13^2 = 169 = 2^3 \iff$
$\log 13^2 = \log 2^3 \iff 2\log 13 \equiv 3\log 2 \iff$
$2\log 13 \equiv 6 \pmod{22}$

Sledi $\log 13 \equiv 3$ ali $14 \pmod{22}$.
Preverimo $\log 13 = 14$.

Izračunaj log(14):
$14^3 = 2^3 7^3 = 2^3 \cdot 21 = 2^3 \cdot (-2) = -2^4.$
$3\log 14 = \log(-2^4) = \log(-1) + \log 2^4 = 11 + 4 \cdot 2 = 19,$
$\log 14 = \frac{19}{3} = 19 \cdot (-7) = (-3)(-7) = 21.$

Izračunaj log(15):
$15^3 = 3^3 \cdot 5^3 = 3^3 \cdot 2 \cdot 5 = (-1) \cdot 2 \cdot 3$,
$3 \log 15 = \log(-1) + \log 2 + \log 3 = 11+2+16 = 29 = 7$,
$\log 15 = \frac{7}{3} = 7(-7) = -49 = -5 = 17$.

Izračunaj log(7):
$7^3 = 49 \cdot 7 = 3 \cdot 7 = 21 = (-1) \cdot 2$,
$3 \log 7 = \log(-1) + \log 2 = 11 + 2 = 13$,
$\log 7 = \frac{13}{3} = 13 \cdot (-7) = 63 = -3 = 19$.

Še en primer: $GF(89)^*$ z gen. 3.

tabela logaritmov:

elt	-1	2	3	5
log	44	16	1	70

Izračunaj log(7):
$7^3 = 76 = 2^2 \cdot 19,\;\; 7^5 = 3 \cdot 5^2,$
$5 \log 7 = \log 3 + 2 \log 5 = 1 + 2 \cdot 70 = 141 = 53,$
$\log 7 = \frac{53}{5} = 53 \cdot (-35) = 81.$

Izračunaj log(53):
$53^3 = 3 \cdot 23,\; 53^5 = 2^2 \cdot 17,\; 53^7 = 2 \cdot 3^2,$
$7 \log 53 = \log 2 + 2 \log 3 = 16 + 2 = 8,$
$\log 53 = \frac{18}{7} = 18 \cdot (-25) = 78.$

Metoda index calculus (splošno)

1. Izberi bazo faktorjev $\mathcal{B} = \{p_1, \ldots, p_t\}$, tako da se da dovolj veliko število elementov grupe G dovolj hitro razstaviti v $\mathcal{B}$.

2. Poišči $t + 10$ lineranih zvez z logaritimi elementov iz $\mathcal{B}$:

izberi število $k < n$, izračunaj α^k in ga poskusi zapisati kot

$$\alpha^k = \Pi_{i=1}^t \, p_i^{c_i} \quad \Longleftrightarrow \quad k \equiv \sum_{i=1}^{t} c_i \log p_i \bmod p-1.$$

3. Sestavi tabelo logaritmov elementov iz $\mathcal{B}$.

4. Izberi naključno število $k \in \{1, \ldots, n\}$, izračunaj $\beta\alpha^k$ in ga poskusi zapisati kot

$$\beta\alpha^k = \Pi_{i=1}^{t} \, p_i^{d_i}.$$

Končno dobimo

$$\log_\alpha \beta = (\sum_{i=1}^{t} \, d_i \log_\alpha p_i - k) \quad \mod \ n.$$

Obstajajo različni slučajni algoritmi za metodo Index calculus. Ob sprejemljivih predpostavkah je njihova časovna zahtevnost za pripravljalno fazo

$$\mathcal{O}\left(e^{1+o(1))\sqrt{\log p \log \log p}}\right),$$

za izračun vsakega posameznega logaritma pa

$$\mathcal{O}\left(e^{1/2+o(1))\sqrt{\log p \log \log p}}\right).$$

Varnost bitov pri diskretnem log.

Podatki: (p, α, β, i),
kjer je p praštevilo, α primitiven element grupe $\mathbb{Z}_p^*$ in i poljubno naravno število, ki je manjše ali enako $\log_2(p-1)$.

Cilj: izračunaj i-ti bit (oznaka: $L_i(\beta)$) logaritma $\log_\alpha \beta$ za fiksna α in p (začnemo šteti z desne).

$L_1(\beta)$ lahko najdemo s pomočjo Eulerjevega kriterija za kvadratne ostanke po modulu p:

Ker je $w^2 \equiv x^2 \pmod{p} \iff p|(w-x)(w+x)$ oziroma $w \equiv \pm x \pmod{p}$, velja

$$\{x^2 \bmod p \mid x \in \mathbb{Z}_p^*\} = \left\{\alpha^{2i} \bmod p \;\middle|\; 0 \leq i \leq \frac{p-3}{2}\right\}.$$

Od tod pa sledi

$$\beta \text{ kvadratni ostanek} \iff 2|\log_\alpha \beta \quad \text{tj.} \quad L_1(\beta) = 0,$$

element β pa je kvadratni ostanek če in samo če je

$$\beta^{(p-1)/2} \equiv 1 \pmod{p}.$$

Sedaj pa si oglejmo še primer, ko je $i > 1$.

Naj bo $p - 1 = 2^s t$, kjer je t liho število.
Potem za $i \leq s$ ni težko izračunati $L_i(\beta)$,
verjetno pa je težko izračunati $L_{s+1}(\beta)$,
kajti v nasprotnem primeru bi bilo možno uporabiti hipotetični podprogram za rešitev DLP v $\mathbb{Z}_p$.

Zgornjo trditev bomo dokazali za $s = 1$ oziroma $p \equiv 3 \pmod 4$. Tedaj sta kvadratna korena iz β po modulu p števili $\pm\beta^{(p+1)/4} \bmod p$.

Za $\beta \neq 0$ velja $\quad L_1(\beta) \neq L_1(p-\beta), \quad$ saj iz

$$\alpha^a \equiv \beta \pmod{p} \implies \alpha^{a+(p-1)/2} \equiv -\beta \pmod{p},$$

ker je $(p-1)/2$ liho število.

Če je $\beta = \alpha^a$ za neko sodo potenco a, potem je

$$\alpha^{a/2} \equiv \beta^{(p+1)/4} \text{ ali } -\beta^{(p+1)/4} \pmod{p}.$$

Katera izmed teh dveh možnosti je pravilna lahko ugotovimo iz $L_2(\beta)$, saj velja

$$L_2(\beta) = L_1(\alpha^{a/2}).$$

Algoritem za računanje diskretnega logaritma v $\mathbb{Z}_p$ za $p \equiv 3 \pmod 4$:

1. $x_0 = L_1(\beta),\ \ \beta = \beta/\alpha^{x_0} \mod p, \quad i := 1$
2. **while** $\beta \neq 1$ **do**
3. $\qquad x_i = L_2(\beta)$
4. $\qquad \gamma = \beta^{(p+1)/4} \mod p$
5. $\qquad$ **if** $L_1(\gamma) = x_i$ **then** $\beta = \gamma$
6. $\qquad$ **else** $\beta = p - \gamma$
7. $\qquad \beta = \beta/\alpha^{x_i} \mod p,\ \ i := i + 1$

Dokaz pravilnosti zgornjega algoritma:

Naj bo

$$x = \log_\alpha \beta = \sum_{i \geq 0} x_i 2^i$$

in definirajmo za $i \geq 0$:

$$Y_i = \left\lfloor \frac{x}{2^{i+1}} \right\rfloor$$

in naj bo β_0 vrednost β v koraku 1.
Za $i \geq 1$, pa naj bo β_i vrednost β v zadnjem koraku pri i-ti iteraciji **while** zanke.

Z indukcijo pokažemo za vsak $i \geq 0$:

$$\beta_i \equiv \alpha^{2Y_i} \pmod{p}.$$

Iz $2Y_i = Y_{i-1} - x_i$ sledi $x_{i+1} = L_2(\beta_i)$ za $i \geq 0$ ter končno še $x_0 = L_1(\beta)$. $\blacksquare$

Končni obsegi

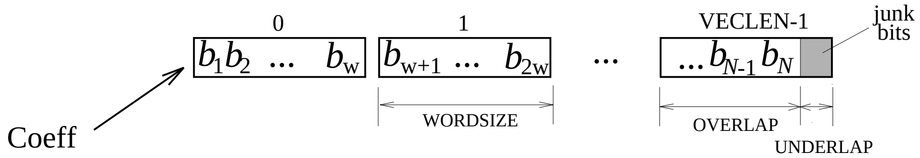
Primer končnega obsega: $\mathrm{GF}(2^4)$

Izberimo $f(x) = 1 + x + x^4 \in \mathrm{GF}(2)[x]$.
Naj bo $a_0 + a_1x + a_2x^2 + a_3x^3 = (a_0, a_1, a_2, a_3)$.
Elementi obsega $\mathrm{GF}(2^4)$ so:

(1000)	(1100)	(1010)	(1111)
(0100)	(0110)	(0101)	(1011)
(0010)	(0011)	(1110)	(1001)
(0001)	(1101)	(0111)	(0000)

Element končnega obsega v predstavimo kot vektor. V hardwaru ponavadi delamo v GF(2), torej je v 01-vektor, ki ga hranimo v registru dolžine n, in je vsota vektorjev enaka XOR po koordinatah.

V softwaru pa hranimo binarni vektor v v besedah (npr. long integers)



V splošnem obstaja veliko število različnih baz za GF(q^m) nad GF(q).

Definirajmo operaciji '+' in '×' v $\mathrm{GF}(p^n)$:

$$(a_0, \ldots, a_{n-1}) + (b_0, \ldots, b_{n-1}) = (c_0, \ldots, c_{n-1}),$$

kjer je $c_i = a_i + b_i \pmod{p}$.

$$(a_0, \ldots, a_{n-1}) \times (b_0, \ldots, b_{n-1}) = (r_0, \ldots, r_{n-1}),$$

kjer je $(r_0, \ldots, r_{n-1})$ ostanek produkta $(a_0, \ldots, a_{n-1}) \times (b_0, \ldots, b_{n-1})$ pri deljenju z nerazcepnim polinomom $f(x)$ stopnje n.

Primer: $(1011)+(1001)=(0010)$

$$
\begin{aligned}
&(1011)\times(1001)\\
&=(1+x^2+x^3)(1+x^3)=1+x+x^5+x^6\\
&=(x^4+x+1)(x^2+x)+(1+x+x^2+x^3)\\
&=(1111)
\end{aligned}
$$

Končni obseg $\mathrm{GF}(2^4)^*$: izberemo $f(x) = 1+x+x^4$. $\mathrm{GF}(2^4)^*$ je generiran z elementom $\alpha = x$.

$\alpha_0 = (1000)$	$\alpha_8 = (1010)$
$\alpha_1 = (0100)$	$\alpha_9 = (0101)$
$\alpha_2 = (0010)$	$\alpha_{10} = (1110)$
$\alpha_3 = (0001)$	$\alpha_{11} = (0111)$
$\alpha_4 = (1100)$	$\alpha_{12} = (1111)$
$\alpha_5 = (0110)$	$\alpha_{13} = (1011)$
$\alpha_6 = (0011)$	$\alpha_{14} = (1001)$
$\alpha_7 = (1101)$	$\alpha_{15} = \alpha^0 = 1$

Log tabela

log	elt		log	elt
0	(1000)		8	(1010)
1	(0100)		9	(0101)
2	(0010)		10	(1110)
3	(0001)		11	(0111)
4	(1100)		12	(1111)
5	(0110)		13	(1011)
6	(0011)		14	(1001)
7	(1101)			

Zech log tabela

$1 + \alpha^i = \alpha^{z(i)}$				
i	$z(i)$		i	$z(i)$
∞	0		7	9
0	∞		8	2
1	4		9	7
2	8		10	5
3	14		11	12
4	1		12	11
5	10		13	6
6	13		14	3

Računanje v **polinomski bazi** je odvisno od izbire polinoma $f(x)$.

Da bi pospešili redukcijo (po množenju ali kvadriranju), si ponavadi izberemo za $f(x)$ nerazcepni **trinom** (to je $x^n + x^m + 1$).

Na žalost nerazcepni trinomi ne obstajajo za poljubno velikost končnega obsega. V tem primeru uporabljamo *pentonome* ali *helptonome*.

Znano je, da ima vsak končni obseg $\mathrm{GF}(p^n)$ bazo nad podobsegom $\mathrm{GF}(p)$ naslednje oblike:

$$B = \{\beta, \beta^p, \cdots, \beta^{p^{n-1}}\}.$$

V praksi so takšne baze, ki jih imenujemo **normalne**, zelo praktične za hardwersko implementacijo množenja v obsegu $\mathrm{GF}(p^n)$, še posebej, kadar je $p = 2$.

Implementacija

Potenciranje opravimo z algoritmom kvadriraj in množi:

$$\alpha^{21} = (\alpha)(\alpha^4)(\alpha^{16})$$

Najprej izračunamo faktorje α, α^2, α^4, α^8, α^{16}, in jih nato zmnožimo.

Namesto 20 množenj smo jih potrebovali le 6.

Ali je lahko kvadriranje hitrejše od (splošnega) množenja?

NE!

$$ab = \frac{(a+b)^2 - a^2 - b^2}{2}.$$

Če je kvadriranje 'lahko', potem tudi splošno množenje ni dosti težje od seštevanja.

DA!

V normalni bazi končnega obsega $\mathrm{GF}(2^n)$
je kvadriranje ciklični zamik,
množenje pa ostane težko v splošnem.

V praksi so normalne baze zelo praktične za hardwarsko implementacijo množenja v obsegu $\mathrm{GF}(p^n)$, še posebej, kadar je $p = 2$. in je kvadriranje *cikličen zamik*.

S tem namenom so Mullin, Onyszchuk, Vanstone in Wilson [MOVW88] definirali **optimalne normalne baze** (ONB) kot tiste baze, katerih število koeficientov v reprezentaciji elementov β^{p^i+1}, $i = 0, \cdots, n-1$ glede na bazo B je natanko $2n-1$. Z drugimi besedami $n \times n$-razsežna matrika $T = (t_{mk})$, definirana z $\beta\beta^{p^m} = \sum_{k=0}^{n-1} \beta^{p^k} t_{mk}$, vsebuje natanko $2n-1$ neničelnih elementov.

Ni težko preveriti, da je število $2n-1$ absolutna spodnja meja (DN).

Izrek (Mullin et al. [MOVW]):

Obseg $\mathrm{GF}(p^n)$ *vsebuje optimalno normalno bazo v naslednjih primerih*

(i) $n+1$ *je praštevilo in* p *primitiven element obsega* $\mathrm{GF}(n+1)$,

(ii) $p=2$, $2n+1$ *je praštevilo in bodisi*
2 je primitiven element obsega $\mathrm{GF}(2n{+}1)$ *bodisi*
n *je lih in 2 generira kvadratne ostanke obsega* $\mathrm{GF}(2n{+}1)$.

Mullin et al. [MOVW] so postavili hipotezo, da za $p = 2$ obstajajo optimalne normalne baze natanko tedaj kadar velja en izmed pogojev (i) in (ii).

Hipotezo sta leta 1992 dokazala Gao in Lenstra.