5. poglavje

Drugi javni kriptosistemi

- ElGamalovi kriptosistemi in Massey-Omura shema
- Problem diskretnega logaritma in napadi nanj
- Metoda velikega in malega koraka
- Pohlig-Hellmanov algoritem
- Index calculus
- Varnost bitov pri diskretnem logaritmu
- Končni obsegi in eliptične krivulje
- Eliptični kriptosistemi
- Merkle-Hellmanov sistem z nahrbtnikom
- Sistem McEliece

Javna kriptografija

L. 1976 sta Whit **Diffie** in Martin **Hellman** predstavila koncept kriptografije z javnimi ključi.

Le-ta za razliko od simetričnega sistema uporablja dva različna ključa, **zasebnega** in **javnega**.
V prejšnjem poglavju smo spoznali RSA (1978).

Taher ElGamal (1985): enkripcije z javnimi ključi in sheme digitalnih podpisov.

Varianta: algoritem za digitalni podpis (**Digital Signature Algorithm – DSA**), ki ga je prispevala vlada ZDA.

V razvoju javne kriptografije je bilo razbitih veliko predlaganih sistemov.

Le tri vrste so se ohranile in jih danes lahko smatramo za varne in učinkovite.

Glede na matematični problem, na katerem temeljijo, so razdeljene v tri skupine:

- **Sistemi faktorizacije celih števil** (Integer Factorization Systems) z RSA (Rivest-Adleman-Shamir) kot najbolj znanim predstavnikom,
- **Sistemi diskretnega logaritma** (Discrete Logarithem Systems), kot na primer DSA,
- **Kriptosistemi z eliptičnimi krivuljami** (Elliptic Curve Cryptosystems).

Problem diskretnega logaritma v grupi G

za dana $\alpha, \beta \in G$, kjer je red elementa α enak n, najdi $x \in \{0, \dots, n-1\}$, tako da je $\alpha^x = \beta$.

Število x se imenuje **diskretni logaritem** osnove α elementa β.

Medtem ko je diskretni logaritem (verjetno) težko izračunati (v splošnem), lahko potenco izračunamo hitro (primer enosmerne funkcije).

Problem diskretnega logaritma v grupi $\mathbb{Z}_p$

Trenutno ne poznamo nobenega polinomskega algoritma za DLP.

Praštevilo p mora imeti vsaj 150 mest (500 bitov), $p-1$ pa mora imeti vsaj en "velik" prafaktor.

ElGamalovi protokoli

Delimo jih v tri razrede:

1. protokoli za izmenjavo ključev,
2. sistemi z javnimi ključi,
3. digitalni podpisi.

Te protokole lahko uporabimo s poljubno končno grupo G.

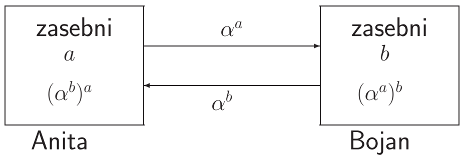
Osnovna razloga za uporabo različnih grup:

- operacije v nekaterih grupah so izvedene enostavneje v programih (software) in programski opremi (hardware) kot v drugih grupah,
- problem diskretnega logaritma je lahko v določeni grupi zahtevnejši kot v drugi.

Naj bo $\alpha \in G$ in naravno število n red tega elementa (t.j., $\alpha^n = 1$ in $\alpha^k \neq 1$ za vsak $k < n$).

1. Izmenjava ključev

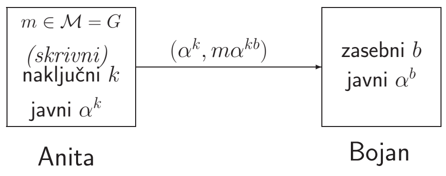
(Diffie-Hellman)



Anita in Bojan si delita skupni element grupe: $(\alpha^a)^b = (\alpha^b)^a = \alpha^{ab}$.

2. ElGamalov kriptosistem javnih ključev

(dva ključa, asimetrični sistem)



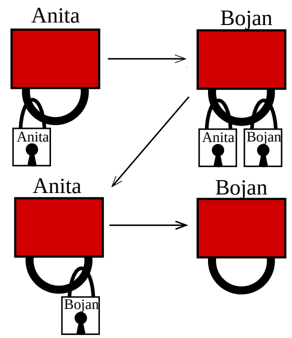
Če je $(y_1, y_2) = e_K(m, k) = (\alpha^k, m\alpha^{kb})$, potem je odšifriranje definirano z $d_K(y_1, y_2) = y_2(y_1^b)^{-1}$.

Sporočilo m lahko prebere le Bojan (s svojim b), ni pa nikjer rečeno, da mu ga je res poslala Anita (saj ni uporabila svojega zasebnega ključa).

V javni kriptografiji smatramo, da nam javni del (npr. α^k, α^b) v ničemer ne pomaga pri iskanju skrivnega/zasebnega dela (npr. k, b).

(Digitalni podpis bo obravnavan v 6. poglavju.)

Massey-Omura shema



Zgled:
za G si izberemo grupo $GF(23)^*$.

Elementi obsega $GF(23)$ so: $0, 1, \ldots, 22$.

Definirajmo:
$a + b = r_1$, kjer je r_1 vsota $a + b$ mod 23.
$ab = r_2$, kjer je r_2 produkt ab mod 23.

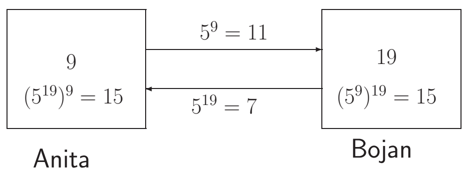
Primer: $12 + 20 = 32 = 9$, $8 \cdot 9 = 72 = 3$.

Multiplikativna grupa $GF(23)^*$

Elementi $GF(23)^*$ so elementi $GF(23)\backslash\{0\}$ in jih lahko generiramo z enim elementom:

$5^0 = 1$	$5^8 = 16$	$5^{16} = 3$
$5^1 = 5$	$5^9 = 11$	$5^{17} = 15$
$5^2 = 2$	$5^{10} = 9$	$5^{18} = 6$
$5^3 = 10$	$5^{11} = 22$	$5^{19} = 7$
$5^4 = 4$	$5^{12} = 18$	$5^{20} = 12$
$5^5 = 20$	$5^{13} = 21$	$5^{21} = 14$
$5^6 = 8$	$5^{14} = 13$	$5^{22} = 1$
$5^7 = 17$	$5^{15} = 19$	

Diffie–Hellmanov protokol v $GF(23)^*$



Anita in Bojan si sedaj delita skupen element $5^{9 \cdot 19} = 15$.

Log tabela

log	elt	log	elt	log	elt
0	1	8	16	16	3
1	5	9	11	17	15
2	2	10	9	18	6
3	10	11	22	19	7
4	4	12	18	20	12
5	20	13	21	21	14
6	8	14	13		
7	17	15	19		

Grupo G in generator α si izberemo tako, da je red elementa α velik (s tem pa je velika tudi log tabela).

Antilog tabela

elt	log	elt	log	elt	log
1	0	9	10	17	7
2	2	10	3	18	12
3	16	11	9	19	15
4	4	12	20	20	5
5	1	13	14	21	13
6	18	14	21	22	11
7	19	15	17		
8	6	16	8		

Algoritmi za računanje diskretnega logaritma

- Shankov algoritem (veliki korak – mali korak),
- Pollardov ρ-algoritem,
- Pohlig-Hellmanov algoritem,
- metoda "index calculus".

Danes si bomo ogledali samo prvega in zadnja dva.

Metoda veliki korak – mali korak:

$GF(23)^*$ z gen. 5: sestavi tabelo elementov $5^0, 5^5, 5^{10}, 5^{15}, 5^{20}$ in njihovih logaritmov.

element	1	20	9	19	12
logaritem	0	5	10	15	20

Izračunaj log(18): računaj $5 \times 18, 5^2 \times 18, \cdots$, vse dokler ne dobiš elementa iz tabele.
$5 \times 18 = 21, \quad 5^2 \times 18 = 13, \quad 5^3 \times 18 = 19.$
Iz tabele dobimo $\log(5^3 \times 18) = \log 19 = 15$.
Sledi $3 + \log 18 = 15$ oziroma $\log 18 = 12$.

$GF(89)^*$ z generatorjem 3: sestavi tabelo elementov $3^0, 3^{10}, 3^{20}, \cdots, 3^{80}$ in njihovih algoritmov.

elt	1	42	73	40	78	72	87	5	32
log	0	10	20	30	40	50	60	70	80

Izračunaj log(36): računaj $3 \times 36, 3^2 \times 36, \cdots$, vse dokler ne dobiš elementa iz tabele.
$3 \times 36 = 19$, $3^3 \times 36 = 82$, $3^5 \times 36 = 26$, $3^2 \times 36 = 57$, $3^4 \times 36 = 68$, $3^6 \times 36 = 78$.
Iz tabele preberemo $\log(3^6 \times 36) = \log 78$. Sledi $6 + \log 36 = 40$ oziroma $\log 36 = 34$.

Čim daljša je tabela, ki jo sestavimo, tem dlje časa jo je treba računati (enkratni strošek), po drugi strani pa hitreje naletimo na element v krajši tabeli.

Običajno sestavimo tabelo velikosti $m = \lfloor \sqrt{|G|} \rfloor$ in za iskanje potrebujemo $O(m)$ časa.

Pollardov ρ algoritem (s Floydovim algoritmom za iskanje ciklov)

Ima isto časovno zahtevnost kot metoda veliki korak – mali korak, porabi pa le malo spomina.

Pohlig-Hellmanov algoritem

$$p - 1 = \Pi_{i=1}^{k} p_i^{c_i}$$

za različna praštevila p_i. Vrednost $a = \log_\alpha \beta$ je natanko določena po modulu $p - 1$.

Najprej izračunamo $a \bmod p_i^{c_i}$ za vsak $i = 1, \ldots, k$ in nato izračunamo $a \bmod (p - 1)$ po kitajskem izreku o ostankih.

Predpostavimo, da je q praštevilo in c največje naravno število, za katero velja

$$p - 1 \equiv 0 \pmod{q^c}.$$

Kako izračunamo

$$x = a \bmod q^c, \quad \text{kjer je } 0 \leq x \leq q^c - 1?$$

Zapišimo x v številskem zapisu z osnovo q:

$$x = \sum_{i=0}^{c-1} a_i q^i, \quad \text{kjer je } 0 \leq a_i \leq q - 1.$$

Od tod dobimo

$$a = a_0 + a_1 q + \cdots + a_{c-1} q^{c-1} + s q^c,$$

kjer je s neko naravno število in $a = a_0 + Kq$.
a_0 izračunamo iz naslednje identitete

$$\beta^{(p-1)/q} \equiv \alpha^{a_0 (p-1)/q} \bmod p.$$

Dokažimo slednjo kongruenco:

$$\begin{aligned}\beta^{(p-1)/q} &\equiv (\alpha^a)^{(p-1)/q} \bmod p\\ &\equiv (\alpha^{a_0+Kq})^{(p-1)/q} \mod p\\ &\equiv \alpha^{a_0(p-1)/q}\,\alpha^{(p-1)K} \mod p\\ &\equiv \alpha^{a_0(p-1)/q} \mod p.\end{aligned}$$

Najprej torej izračunamo

$$\beta^{(p-1)/q} \mod p.$$

Če je $\beta^{(p-1)/q} \equiv 1 \mod p$, je $a_0 = 0$, sicer pa zaporedoma računamo

$$\gamma = \alpha^{(p-1)/q} \mod p, \quad \gamma^2 \mod p, \quad \dots,$$

vse dokler ne dobimo

$$\gamma^i \mod p = \beta^{(p-1)/q} \mod p$$

in je $a_0 = i$.

Sedaj moramo določiti $a_1, \dots, a_{c-1}$ (če je $c > 1$). Naj bo

$$\beta_j = \beta\alpha^{a_0+a_1q+\dots+a_{j-1}q^{j-1}} \mod p,$$

za $0 \le j \le c-1$. Tokrat velja splošnejša identiteta

$$(\beta_j)^{(p-1)/q^{j+1}} \equiv \alpha^{a_j(p-1)/q} \mod p,$$

ki jo dokažemo na enak način kot prejšnjo.

Za dani β_j ni težko izračunati a_j, omenimo pa še rekurzijo

$$\beta_{j+1} = \beta_j\alpha^{-a_jq^j} \mod p.$$

Za dano faktorizacijo števila n je časovna zahtevnost Pohlig-Hellmanovega algoritma $O(\sum_{i=0}^k c_i(\log n + \sqrt{p_i}))$ grupnih multiplikacij.

Primer: naj bo $p = 251$, potem je

$$n = p - 1 = 250 = 2{\cdot}5^3.$$

Naj bo $\alpha = 71$ in $\beta = 210$, torej želimo izračunati $a = \log_{71} 210$.

Modul 2: $\gamma_0 = 1$,

$$\gamma_1 \equiv \alpha^{250/2} \equiv 250 \mod p$$

in

$$\beta^{250/2} \equiv 250 \mod p,$$

torej $a_0 = 1$ in $\log_{71} 210 \equiv 1 \pmod 2$.

Modul 5: $\gamma_0 = 1$,

$$\gamma_1 \equiv \alpha^{250/5} \equiv 20 \bmod p$$

in

$$\beta^{250/5} \equiv 149 \mod p,$$

torej $a_0 = 2$. ...
$a_1 = 4 = \log_{20} 113$ in $a_2 = 2 = \log_{20} 149$,

$\log_{71} 210 \equiv 2 + 4{\cdot}5 + 2{\cdot}5^2 \equiv 72 \pmod{125}$.

Končno nam CRT da $\log_{71} 210 = 197$.

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 $\mathcal{B} = \{p_1, \dots, p_t\}$, 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

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

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

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

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

Končno dobimo

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

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

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

Od tod pa sledi

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

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

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 $x_0 = L_1(\beta)$. ■

Končni obsegi

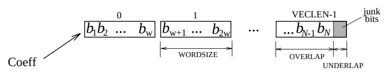
Primer končnega obsega: $\mathrm{GF}(2^4)$

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 $\mathrm{GF}(2)$, 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 $\mathrm{GF}(q^m)$ nad $\mathrm{GF}(q)$.

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

$(1011) \times (1001)$
$= (1 + x^2 + x^3)(1 + x^3) = 1 + x + x^5 + x^6$
$= (x^4 + x + 1)(x^2 + x) + (1 + x + x^2 + x^3)$
$= (1111)$

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.

Grupa na eliptični krivulji

Za kriptografijo sta jo leta 1985 prva
predlagala Neal Koblitz in Victor Miller.

Eliptična krivulja E nad obsegom $\mathbb{Z}_p$
je definirana z Weierstrassovo enačbo:

$$y^2=x^3+ax+b \qquad (1)$$

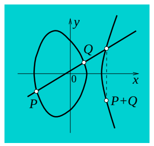
kjer sta $a,b\in\mathbb{Z}_p$ in $4a^3+27b^2\not\equiv 0 \pmod p$
($GF(2^m)$: $y^2+xy=x^3+ax^2+b$).

$$E(\mathbb{Z}_p):=\{(x,y)\,|\,x,y\in\mathbb{Z}_p,\ \text{ki ustrezajo (1)}\}\cup\mathcal{O}.$$

Pravilo za seštevanje

1. $P=(x_1,y_1)$, $Q=(x_2,y_2)\in E(\mathbb{Z}_p)$,
   kjer $P\neq -Q:=(x_2,-y_2)$.
   Potem je $P+Q=(x_3,y_3)$, kjer je
   $x_3=\lambda^2-x_1-x_2$, $\quad y_3=\lambda(x_1-x_3)-y_1$, in

$$\lambda=\begin{cases}\dfrac{y_2-y_1}{x_2-x_1} & ;\ \text{za } P\neq Q\\[2mm] \dfrac{3x_1^2+a}{2y_1} & ;\ \text{za } P=Q.\end{cases}$$



2. $P+\mathcal{O}=\mathcal{O}+P=P \quad$ in $\quad P+(-P)=\mathcal{O}$
   za vsak $P\in E(\mathbb{Z}_p)$.

Množica $E(\mathbb{Z}_p)$ je sestavljena iz točk (x,y), $x,y\in\mathbb{Z}_p$, ki ustrezajo zgornji enačbi, vključno s točko neskončno $\mathcal{O}$.

Izrek (Hasse).

$$p+1-2\sqrt{p}\le|E|\le p+1+2\sqrt{p}$$

Schoofov algoritem izračuna $|E|$ v $O((\log p)^8)$ bitnih operacijah.

Grupa E je izomorfna $\mathbb{Z}_{n_1}\times\mathbb{Z}_{n_2}$, kjer je $n_2|n_1$ in $n_2|(p-1)$, tako da lahko najdemo ciklično podgrupo $\mathbb{Z}_{n_1}$, ki jo uporabimo za ElGamalov kriptosistem.

Podeksponentno metodo **index calculus** zaenkrat ne znamo uporabiti pri DLP na eliptični grupi (razen če ni eliptična krivulja supersingularna).

Zato si lahko izberemo eliptično krivuljo s
ciklično podgrupo velikosti (samo) okoli 2^{160}.

Primer: EC nad $\mathrm{GF}(2^4)$

- Naj bo $\mathrm{GF}(2^4)$ generiran s korenom $\alpha = x$ nerazcepnega polinoma $f(x) = 1 + x + x^4$.
- $E_1(\mathrm{GF}(2^4))$
$= \{(x, y) : y^2 + xy = x^3 + \alpha^4 x^2 + 1\} \cup \{\mathcal{O}\}$.
- $E_1(\mathrm{GF}(2^4))$ tvori grupo za seštevanje z $\mathcal{O}$ kot identiteto.

Rešitve enačbe: $y^2 + xy = x^3 + \alpha^4 x^2 + 1$ nad $\mathrm{GF}(2^4)$

$(0, 1)$	
$(1, \alpha^6)$	$(1, \alpha^{13})$
(α^3, α^8)	(α^3, α^{13})
(α^5, α^3)	(α^5, α^{11})
(α^6, α^8)	(α^6, α^{14})
(α^9, α^{10})	(α^9, α^{13})
(α^{10}, α^1)	(α^{10}, α^8)
$(\alpha^{12}, 0)$	$(\alpha^{12}, \alpha^{12})$

Primer seštevanja v $E_1(\mathrm{GF}(2^4))$:

Naj bo $P_1 = (\alpha^6, \alpha^8)$, $P_2 = (\alpha^3, \alpha^{13})$.

- $P_1 + P_2 = (x_3, y_3)$:

$$\begin{aligned} x_3 &= \left(\frac{\alpha^8 + \alpha^{13}}{\alpha^6 + \alpha^3}\right)^2 + \frac{\alpha^8 + \alpha^{13}}{\alpha^6 + \alpha^3} + \alpha^6 + \alpha^3 + \alpha^4 \\ &= \left(\frac{\alpha^3}{\alpha^2}\right)^2 + \frac{\alpha^3}{\alpha^2} + \alpha^2 + \alpha^4 = 1 \end{aligned}$$

$$\begin{aligned} y_3 &= \left(\frac{\alpha^8 + \alpha^{13}}{\alpha^6 + \alpha^3}\right)(\alpha^6 + 1) + 1 + \alpha^8 \\ &= \left(\frac{\alpha^3}{\alpha^2}\right)\alpha^{13} + \alpha^2 = \alpha^{13} \end{aligned}$$

- $2P_1 = (x_3, y_3)$:

$$\begin{aligned} x_3 &= (\alpha^6)^2 + \frac{1}{(\alpha^6)^2} \\ &= \alpha^{12} + \alpha^3 = \alpha^{10} \end{aligned}$$

$$\begin{aligned} y_3 &= (\alpha^6)^2 + \left(\alpha^6 + \frac{\alpha^8}{\alpha^6}\right)\alpha^{10} + \alpha^{10} \\ &= \alpha^3 + (\alpha^6 + \alpha^2)\alpha^{10} = \alpha^8 \end{aligned}$$

Še en primer EC nad $\mathrm{GF}(2^4)$

- Naj bo $\mathrm{GF}(2^4)$ generiran s korenom $\alpha = x$ nerazcepnega polinoma $f(x) = 1 + x + x^4$.
- $E_2(\mathrm{GF}(2^4))$
$= \{(x, y) : y^2 + \alpha^6 y = x^3 + \alpha^3 x + 1\} \cup \{\mathcal{O}\}$.
- $E_2(\mathrm{GF}(2^4))$ tvori grupo za seštevanje z $\mathcal{O}$ kot identiteto.

Iščemo rešitve enačbe

$$y^2 + \alpha^6 y = x^3 + \alpha^3 x + 1$$

nad $\mathrm{GF}(2^4)$. Ta enačba ima samo 8 rešitev:

(α^2, α^8)	(α^2, α^{14})
$(\alpha^{10}, 1)$	$(\alpha^{10}, \alpha^{13})$
$(\alpha^{11}, 0)$	(α^{11}, α^6)
(α^{13}, α^5)	(α^{13}, α^9)

Primer: EC nad GF(23)

- Naj bo $p = 23$.
- $y^2 = x^3 + x + 1$, (i.e., $a = 1$, $b = 1$).
Velja: $27a^3 + 16b^2 = 3 \cdot 1^3 + 16 \cdot 1^3 = 19 \neq 0$ v GF(23).
- $E_3(\mathrm{GF}(23)) = \{(x, y) : y^2 = x^3 + x + 1\} \cup \{\mathcal{O}\}$.
- $E_3(\mathrm{GF}(23))$ tvori grupo za seštevanje z $\mathcal{O}$ kot identiteto.

Rešitve enačbe $y^2 = x^3 + x + 1$ nad $\mathbb{Z}_{23}$:

(0, 1)	(6, 4)	(-11,-4)
(0,-1)	(6,-4)	(-10, 7)
(1, 7)	(7, 11)	(-10,-7)
(1,-7)	(7,-11)	(-6, 3)
(3, 10)	(9, 7)	(-6,-3)
(3,-10)	(9,-7)	(-5, 3)
(4, 0)	(11, 3)	(-5,-3)
(5, 4)	(11,-3)	(-4, 5)
(5,-4)	(-11,4)	(-4,-5)

Primera seštevanja na $E_3(\mathrm{GF}(23))$

1. $P_1 = (3,10)$, $P_2 = (9,7)$, $P_1 + P_2 = (x_3, y_3)$.

   $\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} = 11 \in \mathbb{Z}_{23}$.

   $x_3 = 11^2 - 3 - 9 = 6 - 3 - 9 = -6$,
   $y_3 = 11(3-(-6)) - 10 = 11(9) - 10 = 89 = 20 = -3$.
   Sledi $P_1 + P_2 = (-6,-3)$.

2. $P_1 = (3,10)$, $2P_1 = (x_3, y_3)$,

   $\lambda = \frac{3(3^2)+1}{20} = \frac{5}{20} = \frac{1}{4} = 6$.

   $x_3 = 6^2 - 6 = 30 = 7$,
   $y_3 = 6(3-7) - 10 = -24 - 10 = -11$.

   Sledi $2P_1 = (7,-11)$.

$P = (0,1)$ je generator:

P=(0, 1)	
$2P$=(6,-4)	$15P$=(9, 7)
$3P$=(3,-10)	$16P$=(-6,3)
$4P$=(-10,-7)	$17P$=(1,7)
$5P$=(-5,3)	$18P$=(12,-4)
$6P$=(7,11)	$19P$=(-4,5)
$7P$=(11,3)	$20P$=(5,4)
$8P$=(5,-4)	$21P$=(11,-3)
$9P$=(-4,-5)	$22P$=(7 -11)
$10P$ =(12,4)	$23P$=(-5,-3)
$11P$=(1,-7)	$24P$=(-10,7)
$12P$=(-6,-3)	$25P$=(3,10)
$13P$=(9,-7)	$26P$=(6,4)
$14P$=(4,0)	$27P$=(0,-1)

Log – antilog tabela

log	elt	log	elt
0	$\mathcal{O}$	14	(4,0)
1	(0,1)	15	(9,7)
2	(6,-4)	16	(-6,3)
3	(3,-10)	17	(1,7)
4	(-10,-7)	18	(-11,-4)
5	(-5,3)	19	(-4,5)
6	(7,11)	20	(5,4)
7	(11,3)	21	(11,-3)
8	(5,-4)	22	(7,-11)
9	(-4,-5)	23	(-5,-3)
10	(-11,4)	24	(-10,7)
11	(1,-7)	25	(3,10)
12	(-6,-3)	26	(6,4)
13	(9,-7)	27	(0,-1)

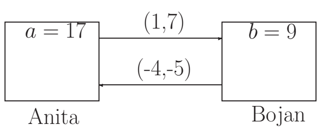
Antilog – log tabela

elt	log	elt	log
$\mathcal{O}$	0	(9,7)	15
(0,1)	1	(9,-7)	13
(0,-1)	27	(11,3)	7
(1,7)	17	(11,-3)	21
(1,-7)	11	(-11,4)	10
(3,10)	25	(-11,-4)	18
(3,-10)	3	(-10,7)	24
(4,0)	14	(-10,-7)	4
(5,4)	20	(-6,3)	16
(5,-4)	8	(-6,-3)	12
(6,4)	26	(-5,3)	5
(6,-4)	2	(-5,-3)	23
(7,11)	6	(-4,5)	19
(7,-11)	22	(-4,-5)	9

Diffie–Hellmanov protokol nad $E(\mathrm{GF}(23))$

Javni parametri:

$$y^2 = x^3 + x + 1$$
$$P = (0,1)$$



- Anita izračuna $17P = (1,7)$,
- Bojan izračuna $9P = (-4,-5)$,
- Anita izračuna $17(-4,-5) = (6,4)$,
- Bojan izračuna $9(1,7) = (6,4)$.

Anita in Bojan imata skupno točko $(6,4)$.

Računanje logaritmov

Izračunaj $\log_P(9,7)$**.**

Izračunaj naslednjo tabelo:

elt	(0,1)	(7,11)	(-6,-3)	(12,-4)	(-10,7)
log	1	6	12	18	24

Če je $k = \log_P(9,7)$, potem velja $kP = (9,7)$.

- Računamo
  $(9,7)+P,\ \ (9,7)+2P,\ \ \ (9,7)+3P, \ldots,$
  vse, dokler ne dobimo element iz tabele.
- Tako dobimo: $(9,7)+3P=(12,-4)$.
- Iz tabele preberemo $(12,-4)=18P$.
- Sledi $(9,7)+3P=18P$
  oziroma $(9,7)=15P$, torej $k=15$.

- Če je $|E(\mathrm{GF}(q))|=n$, lahko posplošimo metodo za $E(\mathrm{GF}(23))$ na naslednji način:
  - naredi tabelo (i, iP) velikosti $\sqrt{n}$,
  - za iskanje logaritma elementa v tej tabeli potrebujemo največ $\sqrt{n}$ seštevanj točk.
- Če je $q \approx 10^{40}$, potem je $|E(\mathrm{GF}(q))| \approx 10^{40}$ in ima tabela 10^{20} vrstic.

  To je očitno popolnoma nedosegljivo.

Merkle-Hellmanov sistem z nahrbtnikom

Merkle in Hellman sta leta 1978 predlagala ta sistem, že leta 1980 pa ga je razbil Shamir s pomočjo Lenstrinega algoritma za celoštevilčno programiranje (angl. integer programming).

Njegovo iterativno varianto pa je razbil malo kasneje Brickell.

Drugačen sistem z nahrbtnikom je predlagal Chor, razbil pa ga je Rivest.

Problem “podmnožica za vsoto”

Podatki: $I=(s_1,\ldots,s_n,T)$, $\quad$ *T je* **ciljna vsota**, *naravna števila s_i pa so* **velikosti**.

Vprašanje: *Ali obstaja tak binarni vektor* $\underline{x}=(x_1,\ldots,x_n)$, *za katerega velja* $\displaystyle\sum_{i=1}^{n} x_i s_i = T$?

Ta odločitveni problem je NP-poln:

– polinomski algoritem ni znan,
– isto velja tudi za ustrezen iskalni problem.

Ali za kakšno podmnožico problemov morda obstaja polinomskim algoritem?

Zaporedje $(s_1,\ldots,s_n)$ je **super naraščajoče**, če velja

$$s_j > \sum_{i=1}^{j-1} s_i \qquad \text{za } \ 2 \leq j \leq n.$$

Če je seznam velikosti super naraščajoč, potem lahko iskalno varianto zgornjega problema rešimo v času $O(n)$, rešitev $\underline{x}$ (če obstaja) pa je enolična.

Opišimo tak algoritem:

1. **for** $i=n$ **downto** 1 **do**
2. $\quad$ **if** $T \geq s_i$ **then**
3. $\quad\quad$ $T=T-s_i,\ \ x_i=1$
4. $\quad$ **else** $\ x_i=0$
5. **if** $T=0$ **then** $\underline{x}=(x_1,\ldots,x_n)$ je rešitev
6. **else** ni rešitve.

Naj bo $\underline{s}=(s_1,\ldots,s_n)$ super naraščajoč in

$$e_{\underline{s}}\ :\ \{0,1\}^n \longrightarrow \left\{0,\ldots,\sum_{i=1}^{n} s_i\right\}$$

funkcija, definirana s pravilom

$$e_{\underline{s}}(x_1,\ldots,x_n)=\sum_{i=1}^{n} x_i s_i.$$

Ali lahko to funkcijo uporabimo za enkripcijo?

Ker je $\underline{s}$ super naraščajoče zaporedje, je $e_{\underline{s}}$ injekcija, zgoraj opisani algoritem pa lahko uporabimo za dekripcijo.

Sistem **ni varen**, saj dekripcijo lahko opravi prav vsak.

Morda pa lahko transformiramo super naraščajoče zaporedje tako, da izgubi to lastnost in edino Bojan lahko opravi inverzno operacijo, da dobi super naraščajoče zaporedje.

Če napadalec Oskar ne pozna te transformacije, ima pred seboj primer (na videz) splošnega problema, ki ga mora rešiti, če hoče opraviti dekripcijo.

En tip takih transformacij se imenuje **modularna transformacija**. Izberemo si tak praštevilski modul p, da je

$$p > \sum_{i=1}^{n} s_i$$

ter število a, $1 \leq a \leq p-1$. Naj bo

$$t_i = \ a s_i \bmod p, \qquad \text{za} \ \ 1 \leq i \leq n.$$

Seznam $\underline{t} = (t_1, \ldots, t_n)$ je javni ključ, ki ga uporabimo za enkripcijo, vrednosti a in p, ki definirata modularno transformacijo, pa sta tajni.

Zakaj smo si izbrali za p praštevilo?

Zakaj je bil ta sistem sploh zanimiv?

Primer: Naj bo

$$s = (2, 5, 9, 21, 45, 103, 215, 450, 946)$$

tajni super naraščajoči seznam velikosti.

Za $p = 2003$ in $a = 1289$ dobimo javni seznam velikosti

$$\underline{t} = (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570).$$

Anita zašifrira sporočilo $\underline{x} = (1, 0, 1, 1, 0, 0, 1, 1, 1)$:

$$y = 575 + 1586 + 1030 + 721 + 1183 + 1570 = 6665$$

ter ga pošlje Bojanu, ki najprej izračuna
$z = a^{-1} y \bmod p = 1643$ in nato
reši problem podmnožice zaporedja $\underline{s}$ za vsoto z.