ARITMETIKA DVOJIŠKIH KONČNIH OBSEGOV

ALEKSANDAR JURIŠIĆ in MARUŠA STANEK,
Fakulteta za računalništvo, Univerza v Ljubljani

Math. Subj. Class. (2000): 11T{06, 22, 55, 71}, 68R05

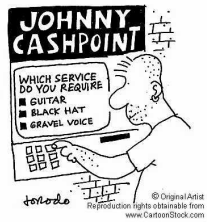
V članku predstavimo končne obsege in aritmetiko v končnih obsegih karakteristike 2. Slednji igrajo pomembno vlogo v implementaciji številnih kriptosistemov in kod za odpravljanje napak. Opišemo učinkovite algoritme za računanje v polinomskih bazah končnih obsegov, ki se pogosto uporabljajo v kriptografskih aplikacijah.

ARITHMETIC OF BINARY FINITE FIELDS

We introduce finite fields and arithmetic in finite fields of characteristic 2 which play an important role in implementation of many cryptosystems and error-correcting codes. We describe efficient arithmetic algorithms in polynomial bases for finite fields which are often used in cryptographic applications.

1. Uvod

S končnimi obsegi so se ukvarjali ugledni matematiki, kot so Fermat, Euler, Lagrange in Legendre, ki so prispevali k razvoju teorije praštevilskih obsegov $\mathbb{Z}_p$. Splošno teorijo končnih obsegov sta začela graditi Gauss in Galois. Vendar pa se je le-ta uveljavila v uporabni matematiki šele s prihodom računalnikov, kjer ne gre brez diskretnih matematičnih struktur. Spomnimo se, da je obseg najpreprostejša algebraična struktura, v kateri lahko izvajamo vse elementarne aritmetične operacije, tj. seštevamo, odštevamo, množimo in delimo (v resnici množimo z multiplikativnim inverzom). Z razvojem teorije kodiranja, kriptografije in številnih kriptosistemov, ki uporabljajo končne obsege, se je pojavila potreba po izboljšavi algoritmov za aritmetiko nad končnimi obsegi. Pri izvajanju kriptografskih aplikacij se osnovne aritmetične operacije v obsegih izvršijo zelo velikokrat, zato je hitrost ključnega pomena. Seštevanje elementov je običajno hitro, zato pa sta množenje in še posebej računanje inverza časovno zahtevnejši operaciji.



Slika 1: Če bi nam bankomat odvrnil, da po njegovem izračunu morda le ni gotovo, da smo pravi lastnik bančne kartice, bi znali biti nejevoljni. Pričakujemo natančen odgovor DA oziroma NE.

Računalniki skoraj vedno računajo s števili, predstavljenimi v dvojiškem sistemu. Naravno število $k \in [2^n, 2^{n+1})$, kjer $n \in \mathbb{N}$, lahko zapišemo kot

$$k = 2^n k_n + 2^{n-1} k_{n-1} + \cdots + 2k_1 + k_0, \quad \text{kjer } k_i \in \mathbb{Z}_2 \ \text{ in } \ k_n \neq 0. \tag{1}$$

V računalniku shranimo le $(n+1)$-terico $k_n k_{n-1} \ldots k_1 k_0$. Če je $\boldsymbol{s} = s_{n+1} s_n \ldots s_1 s_0$ vsota števil $\boldsymbol{a} = a_n a_{n-1} \ldots a_1 a_0$ in $\boldsymbol{b} = b_n b_{n-1} \ldots b_1 b_0$, manjših od 2^{n+1}, potem za $i \in \{0, 1, \ldots, n\}$ velja

$$d_i = a_i + b_i + c_{i-1}, \qquad s_i = d_i \bmod 2, \qquad c_i = d_i \operatorname{div} 2 \qquad \text{in} \qquad s_{n+1} = c_n. \tag{2}$$

Pri tem privzamemo $c_{-1} = 0$ in opomnimo, da je število s_{n+1} lahko tudi enako 0.

Tudi polinom $p(x)$ stopnje n s koeficienti iz $\mathbb{Z}_2$ je primeren za hranjenje v računalniškem pomnilniku, saj za

$$p(x) = \sum_{i=0}^{n} p_i x^i, \qquad \text{kjer } p_i \in \mathbb{Z}_2 \text{ in } p_n \neq 0, \tag{3}$$

spet shranimo samo $(n+1)$-terico $p_n p_{n-1} \dots p_1 p_0$. Če predstavlja $s_n s_{n-1} \dots s_1 s_0$ vsoto polinomov stopnje kvečjemu n, ki sta predstavljena z $a_n a_{n-1} \dots a_1 a_0$ in $b_n b_{n-1} \dots b_1 b_0$, potem je za vsak $i \in \{0, 1, \dots, n\}$

$$s_i = a_i + b_i \bmod 2. \tag{4}$$

Med zapisoma (1) in (3) na prvi pogled ni velike razlike, le število "2" zamenjamo s spremenljivko "x". Vendar pa je računalniško vezje za seštevanje, ki ga predstavlja enačba (4), en sam "ekskluzivni ali" (XOR), za izračun izrazov v (2) pa potrebujemo zaradi morebitnega prenosa dvakrat toliko vezja. V namenskih tiskanih vezjih se tako namesto praštevilskih obsegov večinoma uporabljajo razširitve dvojiškega obsega. Drugi razlog za uporabo razširitev dvojiškega obsega izhaja iz vprašanja: "ali je možno kvadrirati bistveno hitreje od množenja?" Naslednja identiteta

$$a \cdot b = \frac{(a+b)^2 - (a-b)^2}{4}$$

nakazuje, da je odgovor najbrž NE. Kajti če bi znali "zelo" hitro kvadrirati, potem bi kvečjemu dvakrat počasneje lahko izračunali tudi produkt. Ta enačba seveda velja le, če karakteristika obsega ni enaka 2. V razširitvah dvojiških obsegov je nekoliko drugače. V tem primeru namreč obstajajo t.i. normalne baze, v katerih je kvadriranje povsem enostavno. Izvedemo ga le s cikličnim zamikom, množenje pa ostane težko (kvadratne zahtevnosti glede na dolžino zapisa). Zato se bomo osredotočili na aritmetiko dvojiških obsegov, na katere bodo vezali tudi vsi naši primeri. Elemente razširitve dvojiškega obsega zapišemo kot dvojiške vektorje. Sodobni mikroprocesorji pa ne računajo z dolgimi vektorji, ampak uporabljajo krajše enote podatkov, imenovane besede. Tako izvedejo vsako operacijo v nekaj urinih ciklih.

V nadaljevanju najprej predstavimo osnove končnih obsegov, nato pa se posvetimo aritmetiki v razširitvah dvojiškega obsega. Seštevanje je običajno relativno hitra operacija (linearna glede na dolžino zapisa podatkov), kar potem ne velja za množenje. Obstaja pa tudi takšna predstavitev elementov, da je množenje hitro in seštevanje počasno, kot bomo videli v naslednjem razdelku. Nato bomo vpeljali polinomske baze ter opisali metode množenja, kvadriranja in redukcije v njih. Posebej obravnavamo deljenje, katerega izvajamo s pomočjo razširjenega Evklidovega algoritma. Nato opišemo elegantno Berlekampovo izvedbo razširjenega Evklidovega algoritma, katera je še posebej uporabna za računalnike z zelo malo pomnilnika. V zaključku predstavimo vlogo učinkovite aritmetike končnih obsegov v kriptografiji, kjer je le-ta pogosto ključnega pomena pri reševanju matematično zahtevnih problemov. Eden takih je problem diskretnega logaritma po modulu praštevila p, pri katerem za dani naravni števili $g, y \in [1, p-1]$ iščemo tako število x, da je $y = g^x \mod p$.

2. Končni obsegi

Spomnimo se, da je obseg tak kolobar z enoto za množenje, v katerem je vsak neničeln element obrnljiv. Podobseg pa je podmnožica obsega, ki je za isti operaciji tudi obseg. Za vsak končen obseg obstaja tako najmanjše naravno število p, za katerega velja $a+a+\dots+a = p \cdot a = 0$, kjer je a poljuben element danega obsega. Tako število p imenujemo **karakteristika**

končnega obsega in mora biti zaradi minimalnosti praštevilo. Množica ostankov pri deljenju s praštevilom p, skupaj z običajnim seštevanjem in množenjem po modulu p tvori obseg $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, ki mu pravimo **praobseg** in ga spoznamo že pri študiju deljivosti naravnih števil. Le-ta nima prav nobenega netrivialnega podobsega!

Izrek 1. *Moč poljubnega končnega obsega je enaka p^n, kjer je p neko praštevilo in n neko naravno število.*

Skica dokaza. Naj bo p karakteristika nekega obsega F. V tem obsegu enota za množenje generira podobseg, ki je izomorfen praobsegu $\mathbb{Z}_p$. Obseg F je vektorski prostor nad tem podobsegom $\mathbb{Z}_p$, saj je za seštevanje komutativna grupa, skalarno množenje pa je kar običajno množenje. Ker je obseg končen, ima kot vektorski prostor tudi končno razsežnost. Označimo jo z n. Število elementov tega obsega je torej enako p^n. □

V nadaljevanju bomo videli, kako skonstruiramo končni obseg s p^n elementi za poljubno praštevilo p in poljubno naravno število n. Ker so vsi končni obsegi z enakim številom elementov med seboj izomorfni, bomo obseg s p^n elementi označili z $\mathrm{GF}(p^n)$, kjer je GF okrajšava za Galoisov obseg (angl. *Galois field*), in ga smatrali za vektorski prostor nad obsegom $\mathbb{Z}_p$. Če so elementi $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ baza prostora $\mathrm{GF}(p^n)$ nad obsegom $\mathbb{Z}_p$, lahko vsak element obsega $\mathrm{GF}(p^n)$ zapišemo v obliki vsote

$$\sum_{i=0}^{n-1} a_i \alpha_i, \quad \text{kjer} \quad a_i \in \mathbb{Z}_p,$$

ali krajše kot vektor $(a_{n-1}, a_{n-2}, \ldots, a_1, a_0)$. Krajši uvod v grupe in končne obsege najdemo v [6], bolj obširen pa v [12]. Končni obsegi so podrobno predstavljeni v [8], s kriptografskega stališča pa v [9].

Množica neničelnih elementov obsega $\mathrm{GF}(p^n)$ tvori za množenje grupo moči $p^n - 1$, ki jo označimo z $\mathrm{GF}(p^n)^*$ in imenujemo **multiplikativna grupa** končnega obsega. Če potenciramo nek element poljubne končne grupe na moč te grupe, vedno dobimo enoto (glej npr. [12, str. 57]). Torej vsak element $x \in \mathrm{GF}(p^n)^*$ reši enačbo

$$x^{p^n-1} - 1 = 0. \tag{5}$$

Naj bo $q = p^n$. Če enačbo (5) pomnožimo z x, dobimo naslednjo enačbo

$$x^q - x = 0, \tag{6}$$

kateri zadošča tudi element $x = 0$, torej so njene rešitve ravno vsi elementi obsega $\mathrm{GF}(q)$. Od tod sledi, da je leva stran enačbe (6) enaka $(x - a_{q-1})(x - a_{q-2}) \cdots (x - a_1)(x - a_0)$, kjer so $a_{q-1}, a_{q-2}, \ldots, a_1, a_0$ vsi različni elementi obsega $\mathrm{GF}(q)$. Rešitve enačbe (5) pa so $(q-1)$-vi koreni enote v obsegu $\mathrm{GF}(q)$. Spomnimo se, da je **primitivni $(q-1)$-vi koren enote** tako kompleksno število z multiplikativnega reda q–1, da velja $|z| = 1$. Potem so elementi z^i za $i \in \{1, \ldots, q-1\}$ prav tako $(q-1)$-vi koreni enote in tvorijo ciklično grupo za množenje reda $q-1$, tj. multiplikativno grupo $\mathrm{GF}(q)^*$. Ker je vsaka ciklična grupa komutativna (glej npr. [12, str. 56]), so tudi končni obsegi komutativni. Generator grupe $\mathrm{GF}(q)^*$ imenujemo **primitivni element** obsega $\mathrm{GF}(q)$ in se po pričakovanju ujema z definicijo primitivnega $(q-1)$-vega korena enote. Obseg $\mathrm{GF}(q)$ torej sestavljajo elementi $0, z, z^2, \ldots, z^{q-1}$, kjer je z primitivni $(q-1)$-vi koren enote. Pri takem načinu predstavitve elementov obsega je množenje hitro,

saj v pomnilnik namesto elementa z^k zapišemo le eksponent k. Ker velja $z^{k_1} \cdot z^{k_2} = z^{k_1+k_2}$, produkta ni težko izračunati. Vendar pa v tej predstavitvi ne znamo hitro in enostavno izračunati vsote. Za $k_1 \leq k_2$ je $z^{k_1} + z^{k_2} = z^{k_1}(1 + z^{k_2-k_1})$, zato zadošča za vsak $k \in \mathrm{GF}(q)^*$ poznati eksponent ℓ, za katerega velja $z^\ell = z^k + 1$. Za velike končne obsege izračun vseh parov (k, ℓ) ni enostaven, njihovo hranjene pa zavzame tudi veliko računalniškega pomnilnika. Tabela vseh takih parov je poznana pod imenom ZechLog tabela.

Najbolj znana konstrukcija razširitev končnih obsegov sloni na nerazcepnih polinomih. Iskanje nerazcepnega polinoma stopnje m iz kolobarja $\mathrm{GF}(q)[x]$ je v splošnem težko. Konkretno konstrukcijo nerazcepnega polinoma bralec lahko najde v [10, pogl. 3].

Izrek 2. *Naj bo $m \in \mathbb{N}$ in $\mathrm{GF}(q^m)$ končen obseg, ki vsebuje podobseg $\mathrm{GF}(q)$. Potem v kolobarju polinomov $\mathrm{GF}(q)[x]$ obstaja nerazcepen polinom $f(x)$ stopnje m, za katerega je*

$$\mathrm{GF}(q^m) \cong \mathrm{GF}(q)[x]/f(x),$$

kjer je $\mathrm{GF}(q)[x]/f(x)$ obseg polinomov nad $\mathrm{GF}(q)$, reduciranih po modulu polinoma $f(x)$. □

Elementi obsega $\mathrm{GF}(q^m)$ tako ustrezajo polinomom nad $\mathrm{GF}(q)$ stopnje manj kot m. Seštevanje polinomov iz kolobarja $\mathrm{GF}(q)[x]$ se enostavno prevede v ustrezno število seštevanj elementov obsega $\mathrm{GF}(q)$. Pri množenju pa ne gre tako zlahka, saj lahko stopnja produkta dveh polinomov doseže oz. preseže stopnjo nerazcepnega polinoma $f(x)$ in je zato potrebna redukcija po modulu polinoma $f(x)$.

Primer. Konstruirajmo obseg $\mathrm{GF}(2^3)$. Najprej moramo najti nerazcepen polinom stopnje 3 nad $\mathbb{Z}_2$. Najbolj preprosto bi bilo poskusiti kar z binomom. Vendar pa za polinom $f(x)$ s sodo mnogo neničelnimi členi nad $\mathbb{Z}_2$ velja $f(1) = 0$, kar pomeni, da je $f(x)$ razcepen. Zato mora imeti nerazcepen polinom iz kolobarja $\mathbb{Z}_2[x]$ liho število neničelnih členov. Njegov konstantni člen mora biti neničeln, sicer je polinom deljiv z x. Tako sta edina kandidata $f_1(x) = x^3 + x + 1$ in $f_2(x) = x^3 + x^2 + 1$. Oba sta nerazcepna nad $\mathbb{Z}_2$, saj 0 in 1 nista ničli nobenega izmed njiju. Naj bo μ ničla polinoma $f_1(x)$. Potem je $\mu^3 = \mu + 1$ in elementi obsega $\mathrm{GF}(2^3)$ so

$$0,\ \mu,\ \mu^2,\ \mu^3 = \mu + 1,\ \mu^4 = \mu^2 + \mu,\ \mu^5 = \mu^2 + \mu + 1,\ \mu^6 = \mu^2 + 1,\ \mu^7 = 1.$$

Za ničlo ν polinoma $f_2(x)$ pa velja $\nu^3 \equiv \nu^2 + 1$ in dobimo naslednjo upodobitev:

$$0,\ \nu,\ \nu^2,\ \nu^3 \equiv \nu^2 + 1,\ \nu^4 \equiv \nu^2 + \nu + 1,\ \nu^5 \equiv \nu + 1,\ \nu^6 \equiv \nu^2 + \nu,\ \nu^7 \equiv 1,$$

kar lahko zapišemo kot trojice $a_2a_1a_0$, kjer $a_i \in \mathbb{Z}_2$ ustreza koeficientu pri ν^i:

$$000, 010, 100, 101, 111, 011, 110, 001.$$ ◇

3. Polinomske baze

Naj bo α ničla nerazcepnega polinoma $f(x)$ stopnje m iz kolobarja $\mathrm{GF}(q)[x]$. **Minimalni polinom** elementa α je tak polinom $r(x)$, za katerega velja $r(\alpha) = 0$, ima vodilni koeficient enak 1 in ima med vsemi polinomi s pravkar omenjenima lastnostima najmanjšo stopnjo. Vsak polinom $g(x) \in \mathrm{GF}(q)[x]$, ki ima element α za ničlo, je zato deljiv z minimalnim polinomom $r(x)$ elementa α. Ker pa je polinom $f(x)$ nerazcepen, mora biti kar enak skalarnemu večkratniku minimalnega polinoma $r(x)$. Zato element α ni ničla nobenega

neničelnega polinoma iz kolobarja $\mathrm{GF}(q)[x]$ stopnje manjše od m. Potem pa morajo biti elementi $1, \alpha, \ldots, \alpha^{m-1}$ linearno neodvisni in zato sestavljajo bazo. Imenujemo jo **polinomska baza** vektorskega prostora $\mathrm{GF}(q^m)$ nad obsegom $\mathrm{GF}(q)$. Torej vsaka ničla nerazcepnega polinoma stopnje m iz kolobarja $\mathrm{GF}(q)[x]$ določa polinomsko bazo v $\mathrm{GF}(q^m)$. Za različne nerazcepne polinome dobimo različne baze istega vektorskega prostora.

Računanje v polinomskih bazah je bolj učinkovito, če je nerazcepni polinom $f(x)$ iz izreka 2 enostavnejši. V praksi izberemo take $f(x)$, ki imajo malo neničelnih koeficientov. Polinome s tremi neničelnimi členi imenujemo **trinomi**. Le-ti omogočajo hitrejšo implementacijo aritmetike končnih obsegov. V tabeli 1 so našteti nerazcepni trinomi nad obsegom $\mathbb{Z}_2$ stopnje $m \in [150, 250]$, kateri se pogosto uporabljajo v kriptosistemih z eliptičnimi krivuljami. V tabeli je zapisano število m, za katerega nerazcepen trinom obstaja, ter najmanjše število k, za katerega je polinom $x^m + x^k + 1$ nerazcepen nad $\mathbb{Z}_2$.

m	k	m	k	m	k	m	k	m	k	m	k	m	k	m	k
150	53	161	18	174	13	185	24	201	14	214	73	228	113	241	70
151	3	162	27	175	6	186	11	202	55	215	23	231	26	242	95
153	1	166	37	177	8	191	9	204	27	217	45	233	74	244	111
154	15	167	6	178	31	193	15	207	43	218	11	234	31	247	82
155	62	169	34	180	3	196	3	209	6	220	7	236	5	249	35
156	9	170	11	182	81	198	9	210	7	223	33	238	73	250	103
159	31	172	1	183	56	199	34	212	105	225	32	239	36	252	15

Tabela 1: Nerazcepni trinomi nad $\mathbb{Z}_2$. Do te tabele lahko pridemo na požrešen način (podobno kot npr. pri Eratostenovem rešetu) tako, da najprej zmnožimo vse nerazcepne linearne polinome, tj. x in $x+1$. Njihovi produkti x^2, x^2+x, x^2+1 očitno ne "pokrijejo" vseh (štirih) polinomov stopnje 2. Zato je polinom x^2+x+1 nerazcepen. Sedaj množimo x in $x+1$ s polinomi stopnje 2 in ugotovimo, da pri stopnji 3 tudi ne "pokrijemo" vseh polinomov. Z nadaljevanjem tega postopka lahko poiščemo vse nerazcepne polinome željene stopnje. Kadar za nek m ne obstaja nerazcepen trinom, iščemo nerazcepen pentanom ali heptanom. Za vse $m \leq 10.000$ obstaja vsaj nerazcepen pentanom, v kolikor ni že nerazcepnega trinoma stopnje m, glej [11].

Primer. Že v prejšnjem primeru smo se prepričali, da je polinom $f(x) = x^3 + x^2 + 1$ nerazcepen nad $\mathbb{Z}_2$. Torej je $\mathrm{GF}(2^3) \cong \mathbb{Z}_2[x]/f(x)$, množica $\{x^2, x, 1\}$ pa je polinomska baza obsega $\mathrm{GF}(2^3)$. Vsak element zapišemo kot $a_2x^2 + a_1x + a_0$, kar v tem primeru okrajšamo v trojico $a_2a_1a_0$. Tabela 2 prikaže produkte elementov v $\mathrm{GF}(2^3)$.

*	000	001	010	011	100	101	110	111
000	000	000	000	000	000	000	000	000
001	000	001	010	011	100	101	110	111
010	000	010	100	110	101	111	001	011
011	000	011	110	101	001	010	111	100
100	000	100	101	001	111	011	010	110
101	000	101	111	010	011	110	100	001
110	000	110	001	111	010	100	011	101
111	000	111	011	100	110	001	101	010

Tabela 2: Množenje v obsegu $\mathrm{GF}(2^3)$.

Za zgled zmnožimo npr. elementa $x+1$ in x^2+x:

$$(x+1)(x^2+x) = x^3 + x^2 + x^2 + x = x^3 + x = (x^2+1) + x = x^2 + x + 1. \qquad \diamond$$

Preprosta metoda množenja elementov obsega GF(2^m), predstavljenih v polinomski bazi $\{x^{m-1}, x^{m-2}, \ldots, x, 1\}$ z nerazcepnim polinomom $f(x)$ stopnje m, temelji na zvezi

$$a(x) \cdot b(x) = a_0\, b(x) + a_1\, xb(x) + \cdots + a_{m-1}\, x^{m-1} b(x).$$

Po vrsti rekurzivno računamo $x^i b(x) \pmod{f(x)}$ za $i \in \{1, 2, \ldots, m-1\}$ in prištevamo tiste člene, pri katerih je a_i enak 1. Produkt $x \cdot b(x) \pmod{f(x)}$ izračunamo z zamikom koordinat vektorja $b(x)$. Spravimo ga kar v $b(x)$ ter mu na vsakem koraku prištejemo polinom $f(x)$, če je pred zamikom b_{m-1} enak 1. Tak pristop se dobro obnese v strojni opremi, kjer delamo na bitnem nivoju in lahko izvedemo zamik vektorja v enem ciklu. V programski implementaciji pa elemente običajno shranjujemo kot vektorje w-bitnih besed $A = (A[t-1], \ldots, A[1], A[0])$, kjer je $t = \lceil m/w \rceil$, zato pri takem pristopu potrebujemo $m-1$ zamikov besed. Množenje lahko hitreje izvedemo, če najprej zmnožimo elemente brez sprotnega reduciranja (kot običajno množimo polinome) in na koncu napravimo redukcijo po modulu nerazcepnega polinoma $f(x)$. Pri tem izračunanemu $x^k b(x)$ za $k \in \{0, 1, \ldots, w-1\}$ dodamo na konec j ničelnih besed in dobimo element $x^{wj+k} b(x)$. Na ta način za množenje porabimo samo $w-1$ vektorskih zamikov (množenj z x). Podrobnejši opis algoritma za množenje in njegovih izboljšav se nahaja v [4, pogl. 2].

Posebej omenimo kvadriranje polinomov v obsegih karakteristike 2, ki je mnogo hitrejše od množenja dveh različnih polinomov. Kvadriranje dvojiških polinomov je linearna operacija, kvadrat polinoma $a(x) = \sum_{i=0}^{m-1} a_i x^i$ je namreč kar $a(x)^2 = \sum_{i=0}^{m-1} a_i x^{2i}$. Kvadrat elementa dobimo z vstavitvijo ničelnih bitov med bite elementa $a(x)$:

$$(a_{m-1}, a_{m-2}, \ldots, a_1, a_0) \longmapsto (a_{m-1}, 0, a_{m-2}, 0, \ldots, 0, a_1, 0, a_0).$$

Tako pri množenju kot tudi pri kvadriranju polinomov lahko stopnja produkta $c(x)$ doseže ali preseže stopnjo m izbranega nerazcepnega polinoma $f(x)$. Stopnja produkta je lahko največ $2m-2$, zmanjšamo pa jo z redukcijo po modulu nerazcepnega polinoma $f(x)$. Vzemimo vektor w-bitnih besed $C = (C[n], C[n-1], \ldots, C[1], C[0])$ in opišimo algoritem, ki temelji na dejstvu, da za $i \geq m$ velja $x^i = x^{i-m} f_1(x) \pmod{f(x)}$, kjer je $f_1(x) = f(x) - x^m$. Če je $f(x)$ trinom, potem je $x^m = x^k + 1$ in je redukcija učinkovitejša, saj moramo pri reduciranju člena x^i za $i \geq m$ popraviti vektor C le na dveh mestih: pri $x^{i-(m-k)}$ in x^{i-m}. Ob redukciji naslednjega člena x^{i-1} popravljamo sosednje bite prej omenjenih mest. Zato lahko redukcijo izvajamo nad besedami namesto nad biti in s tem izboljšamo algoritem.

4. Evklidov algoritem

Za invertiranje elementov obsega GF(2^m) lahko uporabimo razširjeni Evklidov algoritem [1, razdelek 2.1], ki ga bomo podrobneje opisali v nadaljevanju. Morda je sicer na prvi pogled bolj naravna metoda potenciranja, saj za vsak $\alpha \in \mathrm{GF}(2^m)$ velja $\alpha^{2^m} = \alpha$, od koder za $\alpha \neq 0$ sledi $\alpha^{-1} = \alpha^{2^m - 2}$. Tak pristop se dobro obnese v teoriji, vendar pa se razširjeni Evklidov algoritem v praksi izkaže za dosti hitrejšega. Za $m = 173$ pri računanju inverza s potenciranjem porabimo 10 množenj, medtem ko lahko z razširjenim Evklidovim algoritmom inverzni element dobimo s 3-4 množenji.

Za dani polinom $a(x) \in \mathbb{Z}_2[x]$ stopnje manj od m iščemo tak polinom $p(x) \in \mathbb{Z}_2[x]$ stopnje manj od m, da velja $a(x)p(x) \equiv 1 \pmod{f(x)}$. To pomeni, da obstaja tak polinom $q(x)$, da v kolobarju polinomov $\mathbb{Z}_2[x]$ velja

$$a(x)p(x) + q(x)f(x) = 1. \tag{7}$$

Ker je polinom $f(x)$ nerazcepen, je največji skupni delitelj polinomov $f(x)$ in $a(x)$ enak 1. V splošnem je enačbo (7) težko rešiti, zato si pomagamo z enostavnejšimi primeri. Poznamo namreč rešitev enačbe (7), kadar na desni strani stoji bodisi $f(x)$ ali $a(x)$. V prvem primeru je rešitev kar $p(x) = 0$ in $q(x) = 1$. V drugem primeru, ko je na desni $a(x)$, pa enačbo reši par $p(x) = 1$ in $q(x) = 0$. Zdaj lahko z iterativno metodo poiščemo zaporedji polinomov p_i in q_i, za kateri velja $ap_i + fq_i = r_i$. Upoštevamo omenjena posebna primera in začnemo z $r_{-2} = f$, $r_{-1} = a$, $p_{-2} = 0$, $p_{-1} = 1$, $q_{-2} = 1$ in $q_{-1} = 0$. Na vsakem koraku iščemo taka polinoma s_i in r_i, da velja

$$r_{i-2} = s_i r_{i-1} + r_i,$$

pri čemer je stopnja polinoma r_i manjša od stopnje polinoma r_{i-1}. Postavimo še

$$q_i = s_i q_{i-1} + q_{i-2} \quad \text{in} \quad p_i = s_i p_{i-1} + p_{i-2}$$

ter ponavljamo postopek, dokler na nekem koraku ne dobimo $r_k = 0$. Iskana rešitev je potem $q(x) = q_{k-1}$ in $p(x) = p_{k-1}$, kjer je stopnja q manjša od stopnje a, stopnja p pa manjša od stopnje f. Potrebujemo le polinom $p(x)$, zato zaporedja q_i sploh ne računamo. Za lažjo predstavo si oglejmo primer.

Primer. Naj bo $a(x) = x^4 + x + 1$ in $f(x) = x^5 + x^2 + 1$. Poiščimo inverz polinoma $a(x)$.

$$\begin{array}{rcl@{\qquad}rcl}
r_{-2} = x^5+x^2+1 & = & x(x^4+x+1)+x+1 & x \cdot 1+0 & = & x = p_0(x) \\
r_{-1} = x^4+x+1 & = & (x^3+x^2+x) \cdot (x+1)+1 & (x^3+x^2+x) \cdot x+1 & = & x^4+x^3+x^2+1 = p_1(x) \\
r_0 = x+1 & = & (x+1) \cdot 1+0 & (x+1) \cdot (x^4+x^3+x^2+1)+x & = & x^5+x^2+1 = p_2(x)
\end{array}$$

V zadnji vrstici postane ostanek r_2 enak 0, zato je inverz polinoma $a(x)$ enak $p_1(x)$. $\diamond$

Računalnik polinome deli postopoma, tako da množi polinom nižje stopnje z ustrezno potenco elementa x. To pomeni, da izvaja ciklični zamik, vse dokler nimata oba polinoma enake stopnje. Nato zamenja polinom, ki je na začetku večje stopnje, z njuno razliko. Postopek ponavlja, vse dokler se stopnja tistega polinoma, ki ima večjo ali enako stopnjo, ne zmanjša pod stopnjo drugega polinoma. Tako se po delih izračuna vsak polinom s_i. Oglejmo si pravkar opisani postopek na prejšnjem primeru. Za $i = 0$ se v prvem koraku najprej pomnoži r_{-1} z elementom x. Razlika med r_{-2} in $x \cdot r_{-1}$ je linearni polinom $r_0 = x+1$ in takoj se lahko izračuna $p_0 = s_0 \cdot p_{-1} + p_{-2} = x \cdot 1 + 0 = x$. Nato se za $i = 1$ (pri deljenju polinoma r_{-1} z r_0) ostanek iz prejšnjega koraka r_0 najprej množi z ustrezno potenco elementa x, ki predstavlja prvi sumand v s_1, recimo mu s_{11}:

$$r_{-1} = x^4 + x + 1 = x^3 \cdot (x + 1) + x^3 + x + 1.$$

Zdaj pa je razlika med r_{-1} in $x^3 \cdot r_0$ polinom $x^3 + x + 1$, ki je višje stopnje od r_0. Zato se ju zamenja in ponovi postopek. Pri tem se spet množi z ustrezno potenco elementa x, ki zdaj predstavlja drugi del polinoma s_1, tj. s_{12}:

$$x^3 + x + 1 = x^2 \cdot (x + 1) + x^2 + x + 1.$$

Ker je razlika med $x^3 + x + 1$ in $x^2 \cdot r_0$ še vedno višje stopnje od r_0, se ju spet zamenja in množi r_0 z elementom x, kateri sedaj ustreza tretjemu sumandu polinoma s_1, tj. s_{13}:

$$x^2 + x + 1 = x \cdot (x + 1) + 1.$$

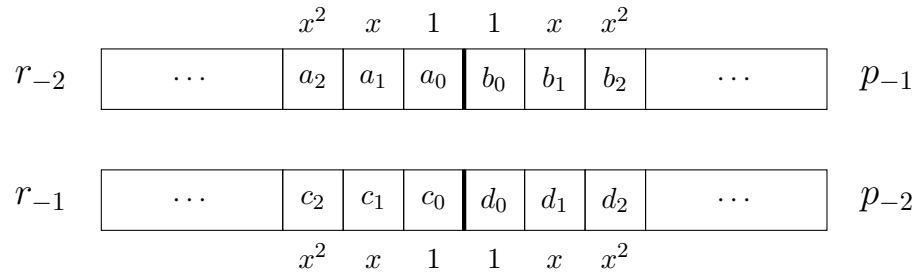
Sedaj je ostanek enak 1 in je nižje stopnje od linearnega polinoma r_0. Tako se po delih izračuna $s_1 = s_{11} + s_{12} + s_{13}$. Ob tem se vzporedno računa polinom $p_1 = s_1 p_0 + p_{-1}$ kot

$$((s_{11}p_0 + p_{-1}) + s_{12}p_0) + s_{13}p_0 = ((x^3 \cdot x + 1) + x^2 \cdot x) + x \cdot x = x^4 + x^3 + x^2 + 1.$$

Podobno v tretjem koraku ($i = 2$) iz $r_0 = x + 1 = (x + 1) \cdot 1 + 0$ sledi, da je $s_2 = x + 1$, $r_1 = 1$ in $r_2 = 0$. Potem pa polinoma p_2 sploh ni potrebno računati, saj je inverz elementa a enak polinomu p_1.

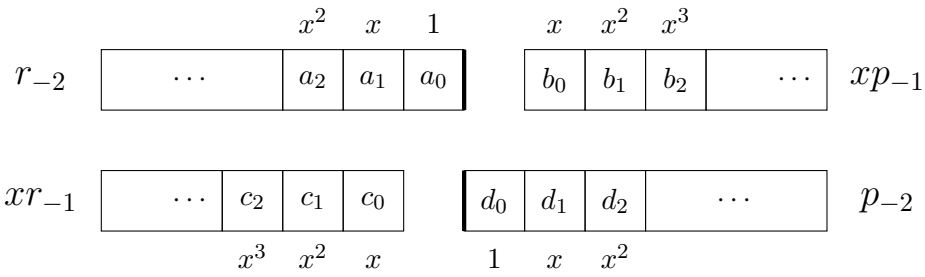
5. Berlekampov algoritem

Predstavili bomo Berlekampovo realizacijo razširjenega Evklidovega algoritma [1, razdelek 2.3], ki je posebej prilagojena za računalnike z zelo malo pomnilnika, kot so na primer pametne kartice. Potrebujemo namreč le dva registra z $m + 2$ biti, kjer je m stopnja nerazcepnega polinoma, opravimo pa najmanjše možno število logičnih operacij. Ideja je v zamikanju parov polinomov. Koeficiente polinomov r in p postavimo skupaj tako, da najbolj levi bit predstavlja vodilni koeficient polinoma r, najbolj desni bit pa vodilni koeficient polinoma p. Na začetku so v zgornjem registru koeficienti polinoma r_{-2}, katerim sledi vejica in za njo enica, ki predstavlja p_{-1}. V spodnji register pa na začetek zložimo koeficiente polinoma r_{-1}, katerim sledi vejica in za njo p_{-2}. Takšen zapis je najprimernejši, saj polinomom r stopnja pada, polinomom p pa se veča. Začetno stanje prikazuje slika 2.



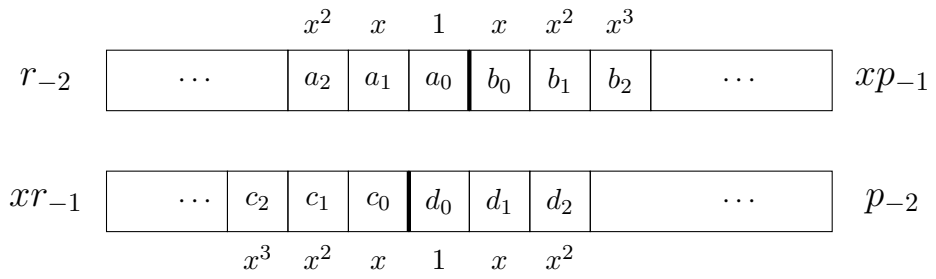
Slika 2: Začetno stanje registrov.

Če množimo polinom r_{-1} z elementom x, zaradi povezave med enačbama $r_{-2} = s_0 r_{-1} + r_0$ in $p_0 = s_0 p_{-1} + p_{-2}$ hkrati množimo tudi p_{-1} z elementom x. Pri tem se levi del spodnjega registra, v katerem so zapisani koeficienti polinoma r_{-1}, zamakne z vejico vred za eno mesto v levo. Desni del zgornjega registra pa se zaradi množenja p_{-1} z x zamakne za eno mesto v desno. Zamika sta prikazana na sliki 3.



Slika 3: Zamik registrov pri množenju z x.

Ta dva zamika pa sta ekvivalentna temu, da le zgornji register zamaknemo za eno mesto v desno, kot prikazuje slika 4. Potem seveda ne moremo med seboj primerjati bitov, ki ležijo med obema vejicama. Lahko pa primerjamo tiste, ki ležijo na levi strani bolj leve vejice, saj le-ti ustrezajo enakim potencam elementa x. Podobno lahko med seboj primerjamo bite, ki ležijo na desni strani bolj desne vejice in ustrezajo naraščajočim potencam elementa x.



Slika 4: Kompaktna oblika registrov.

Algoritem je sestavljen le iz zamikov in seštevanj. Skupno število zamikov (Z) je $2m+1$. Ker vsakemu seštevanju (S) sledi zamik, ne more biti več kot $2m$ seštevanj in celoten algoritem ne zahteva več kot $4m+1$ operacij. Oglejmo si postopek na primeru iz prejšnjega razdelka.

Primer.

I. $\begin{pmatrix}1\,0\,0\,1\,0\,1,\ 1\\ 0\,1\,0\,0\,1\,1,\ 0\end{pmatrix} = \begin{pmatrix}r_{-2},\ p_{-1}\\ r_{-1},\ p_{-2}\end{pmatrix} = \begin{pmatrix}f(x),1\\ a(x),0\end{pmatrix}$

Z: $\begin{pmatrix}1\,0\,0\,1\,0\,1,\ 1\\ 1\,0\,0\,1\,1,\ 0\,0\end{pmatrix} = \begin{pmatrix}r_{-2},\ xp_{-1}\\ xr_{-1},\ p_{-2}\end{pmatrix}$

S: $\begin{pmatrix}0\,0\,0\,0\,1\,1,\ 1\\ 1\,0\,0\,1\,1,\ 0\,1\end{pmatrix} = \begin{pmatrix}r_{-2}+xr_{-1},\ xp_{-1}\\ xr_{-1},\ p_{-2}+xp_{-1}\end{pmatrix}$

II. Z: $\begin{pmatrix}0\,0\,0\,1\,1,\ 1\,0\\ 1\,0\,0\,1\,1,\ 0\,1\end{pmatrix} = \begin{pmatrix}r_0,\ p_{-1}\\ r_{-1},\ p_0\end{pmatrix}$

$3\times$Z: $\begin{pmatrix}1\,1,\ 1\,0\,0\,0\,0\\ 1\,0\,0\,1\,1,\ 0\,1\end{pmatrix} = \begin{pmatrix}x^3r_0,\ p_{-1}\\ r_{-1},\ x^3p_0\end{pmatrix}$

S: $\begin{pmatrix}1\,1,\ 1\,0\,0\,0\,1\\ 0\,1\,0\,1\,1,\ 0\,1\end{pmatrix} = \begin{pmatrix}x^3r_0,\ p_{-1}+x^3p_0\\ r_{-1}+x^3r_0,\ x^3p_0\end{pmatrix}$

Z: $\begin{pmatrix}1\,1,\ 1\,0\,0\,0\,1\\ 1\,0\,1\,1,\ 0\,1\,0\end{pmatrix} = \begin{pmatrix}x^2r_0,\ p_{-1}+x^3p_0\\ r_{-1}+x^3r_0,\ x^2p_0\end{pmatrix}$

S: $\begin{pmatrix}1\,1,\ 1\,0\,0\,1\,1\\ 0\,1\,1\,1,\ 0\,1\,0\end{pmatrix} = \begin{pmatrix}x^2r_0,\ p_{-1}+(x^3+x^2)p_0\\ r_{-1}+(x^3+x^2)r_0,\ x^2p_0\end{pmatrix}$

Z: $\begin{pmatrix}1\,1,\ 1\,0\,0\,1\,1\\ 1\,1\,1,\ 0\,1\,0\,0\end{pmatrix} = \begin{pmatrix}xr_0,\ p_{-1}+(x^3+x^2)p_0\\ r_{-1}+(x^3+x^2)r_0,\ xp_0\end{pmatrix}$

S: $\begin{pmatrix}1\,1,\ 1\,0\,1\,1\,1\\ 0\,0\,1,\ 0\,1\,0\,0\end{pmatrix} = \begin{pmatrix}xr_0,\ p_{-1}+(x^3+x^2+x)p_0\\ r_{-1}+(x^3+x^2+x)r_0,\ xp_0\end{pmatrix}$

III. Z: $\begin{pmatrix}1\,1,\ 1\,0\,1\,1\,1\\ 0\,1,\ 0\,1\,0\,0\,0\end{pmatrix} = \begin{pmatrix}r_0,\ p_1\\ r_1,\ p_0\end{pmatrix}$

Z: $\begin{pmatrix}1\,1,\ 1\,0\,1\,1\,1\\ 1,\ 0\,1\,0\,0\,0\,0\end{pmatrix} = \begin{pmatrix}r_0,\ xp_1\\ xr_1,\ p_0\end{pmatrix}$

$\diamond$

6. Končni obsegi v kriptografiji

Področje končnih obsegov je polno raziskovalnih problemov, tako teoretičnih kakor tudi tistih, ki so povezani z njihovo uporabo. Mnogo slednjih izvira iz kriptografije in teorije kodiranja, kjer je aritmetika končnih obsegov velikokrat bistvenega pomena. V algoritmih pogosto uporabljamo razširitve dvojiških obsegov ali praštevilske obsege. Prednost prvih je v prilagodljivosti dvojiškemu zapisu v računalnikih, prednost drugih pa v že vgrajeni aritmetiki v sodobnih procesorjih. Primeri kriptografskih shem, ki temeljijo na končnih obsegih, so

- klasični Diffie-Hellmanov protokol za dogovor o ključu [2], [5],
- ElGamalove sheme za digitalni podpis [3] ter
- kriptosistemi z javnimi ključi, ki uporabljajo eliptične krivulje [7].

Učinkovitost teh računsko zahtevnih shem je odvisna od hitrosti izvajanja aritmetičnih operacij. Zahtevnost računskih operacij pa je odvisna od izbire baze. Idealno bi bilo, če bi lahko vsako operacijo opravili v tisti bazi, v kateri jo znamo najbolj učinkovito izvesti. Potem bi lahko kombinirali najboljše iz različnih svetov. Izbor baze je odvisen od tega, katere operacije z elementi najpogosteje izvajamo. V kriptografiji so najbolj uveljavljene polinomske in normalne baze. Kot smo videli, polinomske baze odigrajo pomembno vlogo že pri vpeljavi

končnih obsegov. Njihova praktična prednost se pokaže predvsem v učinkovitem invertiranju, ki ga izvajamo z razširjenim Evklidovim algoritmom. Normalne baze pa so zanimive tako s stališča matematične teorije kot tudi zaradi praktične uporabe, zato jih bomo podrobneje opisali v posebnem sestavku. Če namreč predstavimo elemente končnega obsega karakteristike 2 v normalni bazi, lahko kvadriranje izvedemo z enostavnim cikličnim zamikom. To je še posebej uporabno v kriptografskih aplikacijah, ki zahtevajo mnogo kvadriranj elementov končnega obsega, kot so npr. kriptosistemi z eliptičnimi krivuljami, ki temeljijo na problemu diskretnega logaritma.

Literatura

[1] E. Berlekamp, Algebraic Coding Theory, *Aegean Park Press; Revised edition*, 1984.

[2] W. Diffie in M. E. Hellman, New directions in Cryptography, *IEEE Transactions on Information Theory* **IT-22**, 6, str. 644–654 (1976).

[3] ElGamal, A public-key cryptosystem and signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory* **IT-21**, 4, str. 469–472 (1985).

[4] D. Hankerson, A. Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, 2004.

[5] A. Jurišić, Diffie-Hellmanov dogovor o ključu, *Presek* **34**, 1, str. 25–30 (2006/2007).

[6] A. Jurišić, Računala nove dobe, *Presek* **30**, 4 in 5, str. 226–231 in 290–296 (2002/2003).

[7] N. Koblitz, Elliptic curve cryptosystems, *Mathematics of Computation* **48**, str. 203–209 (1987).

[8] R. Lidl in H. Niederreiter, Encyclopedia of Mathematics and Its Applications: Finite Fields, *Cambridge University Press*, 1987.

[9] A. J. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[10] A. J. Menezes, I. F. Blake, S. Gao, R. C. Mullin, S. A. Vanstone and T. Yaghoobian, *Applications of Finite Fields*, Kluwer Academic Publishers, 1993.

[11] G. Seroussi, Table of Low-Weight Binary Irreducible Polynomials, *Hewlett-Packard Laboratories Technical Report*, HPL-98-135, 1998.

[12] I. Vidav, Algebra, *DMFA-založništvo*, 4. natis, 1989.