Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Sabina Marancina

Kriptografija na osnovi kodiranja

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentorica: izr. prof. dr. Arjana Žitnik

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kriptografija na osnovi kodiranja

Tematika naloge:

Kriptografija in še posebej kriptosistemi z javnim ključem igrajo v informacijski dobi pomembno vlogo. Pri kriptosistemih z javnim ključem je ključ za šifriranje javno objavljen in vsakdo lahko šifrira. Varnost kriptosistema pa temelji na tem, da iz javnega ključa ne moremo v doglednem času izračunati zasebnega, s katerim je mogoče šifrirano sporočilo odšifrirati. Da bi lahko iz javnega ključa izračunali zasebni, bi morali rešiti nek težek matematični problem; običajno je to problem diskretnega logaritma ali problem razcepa števila, na katerem temelji danes najširše uporabljani kriptosistem RSA.

Vendar pa zgoraj omenjena problema lahko rešimo v polinomskem času z uporabo kvantnega računalnika. Čeprav danes izgleda, da kvantnih računalnikov še nekaj časa ne bo, se že išče kriptosisteme, ki bi temeljili na problemih, ki se jih ne da hitro rešiti s kvantnim računalnikom. Eden od takšnih problemov je dekodiranje linearnih kodov za popravljanje napak.

V diplomskem delu predstavite Goppove kode za popravljanje napak in opišite učinkovit algoritem za dekodiranje teh kodov. Nato predstavite McEliececov kriptosistem z javnim ključem, ki temelji na Goppovem kodu, in analizirajte njegovo varnost.

Kazalo

Povzetek

Naslov: Kriptografija na osnovi kodiranja

Avtor: Sabina Marancina

Z naraščajočo grožnjo izgradnje kvantnega računalnika, ki bi razbil večino uporabljanih šifer, se povečuje zanimanje za alternativno rešitev, na katero kvantna tehnologija ne bi imela večjega vpliva. V prvem delu diplome predstavimo Goppov kod za popravljanje napak, ki se uporablja v nekaterih post-kvantnih kriptosistemih. Pokažemo, da ima veliko razmaknjenost in opišemo učinkovit algoritem za dekodiranje. V nadaljevanju predstavimo McElieceov kriptosistem z javnim ključem, ki temelji na Goppovem kodu, in na kratko analiziramo varnost kriptosistema, ki ostaja visoka tudi po štiridesetih letih analiz.

Ključne besede: McElieceov kriptosistem, Goppov kod, post-kvantna kriptografija.

Abstract

Title: Code-based cryptography

Author: Sabina Marancina

As the threat of building quantum computers which would break most of the used ciphers increases, so does the interest in an alternative solution. In the first part of this thesis, we consider Goppa codes which are error-correcting codes used in some post-quantum cryptosystems. We show that they have a large minimum distance and describe an efficient decoding algorithm. Furthermore, we consider McEliece public-key cryptosystem based on Goppa codes and we give a short analysis of its security which remains high even after forty years of analysis.

Keywords: McEliece cryptosystem, Goppa code, post-quantum cryptography.

Poglavje 1

Uvod

Živimo v informacijski dobi. Dnevno se po svetovnem spletu prenaša nepredstavljiva količina podatkov, do katerih lahko dostopa marsikdo. Že pri preprosti elektronski pošti si pogosto želimo, da njena vsebina ostane tajna in nespremenjena. Prav tako želimo biti prepričani, da sporočilo prihaja od prave osebe. Dandanes, ko se vse več storitev izvaja elektronsko, kot na primer spletne banke in spletne trgovine, je kriptografija ključnega pomena.

Zelo se je razširila kriptografija z javnim ključem, saj ni potrebna izmenjava ključev. Varnost kriptosistemov z javnim ključem temelji na tem, da iz javnega ključa ne moremo izračunati zasebnega, saj bi morali za to rešiti nek težek matematični problem. Za zdaj se zdi spletna varnost ob primerni uporabi sodobnih kriptografskih postopkov, kot na primer RSA, zagotovljena. Kljub temu da so računalniki vsako leto hitrejši, lahko varnost pogosto zagotovimo samo s podaljšanjem ključa. Vendar temu ne bi bilo več tako ob izgradnji kvantnega računalnika, saj obstajajo algoritmi, s katerimi bi bilo mogoče na kvantnem računalniku razbiti RSA in ostale kriptosisteme v širši uporabi v polinomskem času [21, 40].

Zamislimo si, da čez nekaj let nekdo naznani uspešno izgradnjo kvantnega računalnika in se po medijih razširi novica, da skoraj nobena uporabljana šifra ni več varna. Po svetu kar naenkrat zavlada panika. Marsikdo bi lahko pomislil, da je s tem konec kriptografije in bo mogoče varno hranjenje in prenos

informacij le z uporabo drage fizične zaščite, kot na primer skrivanje USB-jev v zaklenjenem kovčku priklenjenem na zapestje zaupanja vrednega kurirja. Na drugi strani pa bi se začelo mrzlično iskanje novega kriptosistema, ki ga kvantni računalnik ne bi mogel streti. A zakaj bi čakali na katastrofo? Iskanje novih sistemov namreč niti ni potrebno, saj obstajajo številni razredi kriptosistemov, na katere izgradnja kvantnega računalnika ne bi imela velikega vpliva; pravimo jim tudi post-kvantni kriptosistemi [32]. Primeri takih razredov so kriptosistemi na osnovi celoštevilskih mrež, na osnovi zgoščevalnih funkcij in na osnovi kodiranja.

Kod za popravljanje napak lahko predstavlja dobro podlago za kriptosistem, saj bi morebitni prisluškovalec imel težave pridobiti poslano sporočilo iz prejetega v primeru, da bi ga prestregel. Pri kodih za popravljanje napak imamo problem dekodiranja, saj v splošnem kodu iz prejete besede ne znamo na hitro dobiti kodne besede. Ta problem je NP-težek [3]. Za nekatere kode pa poznamo učinkovite dekodirne algoritme, zato bi lahko prejemnik, ki ima dekodirni algoritem, sporočilo hitro prebral.

Kod za popravljanje napak tako lahko uporabimo v kriptosistemu z javnim ključem, kjer lahko vsi kodirajo sporočila, dekodira pa jih lahko samo prejemnik. Lastnik ključa objavi javno kodirno matriko, dekodirno metodo pa ohrani skrivno. Da zagotovimo, da iz javne kodirne matrike ni moč dobiti dekodirnega postopka, moramo kodirno matriko tako spremeniti, da izgleda povsem naključno. Poleg tega mora izbrani kod pripadati velikemu razredu kodov, tako da je nemogoče z izčrpnim preiskovanjem ključev ugotoviti točno kateri kod iz tega razreda je bil izbran.

Prvi tak kriptosistem je McElieceov kriptosistem [27]. Zanj so čez leta predstavili številne napade, ki pa se jim lahko izognemo s primerno izbiro parametrov. Zakaj se torej ta sistem še ne uporablja namesto RSA? Glavni razlog je nepraktičnost zaradi velikega javnega ključa. Poleg tega McElieceovega kriptosistema ni mogoče tako naravno uporabiti za podpisovanje kot RSA. S temi pomanjkljivostmi se ta kriptosistem ob svoji pojaviti enostavno ni razširil zaradi popolnoma varnih preprostejših alternativnih kriptosiste-

mov. Da se pripravimo na vse možne prihodnje scenarije, bi bilo smiselno preučiti dobro sisteme, za katere vemo, da kvantni računalniki ne bodo imeli večjega vpliva.

V tem diplomskem delu se bomo zato posvetili McEliecejevemu sistemu, ki temelji na Goppovem kodu. V 2. poglavju bomo najprej na kratko predstavili, kaj sploh je kriptografija in samo idejo kriptosistemov z javnim ključem. V 3. poglavju se bomo posvetili osnovam teorije končnih obsegov, ki so ključne za razumevanje kasnejših poglavij. V 4. poglavju bomo definirali kode za popravljanje napak in obravnavali njihove najbolj pomembne lastnosti. Goppov kod, ki se uporablja v nekaterih post-kvantnih kriptosistemih, bomo predstavili v 5. poglavju. Pokazali bomo, da ima veliko razmaknjenost in predstavili učinkovit algoritem za dekodiranje. V 6. poglavju pa bomo predstavili McElieceov kriptosistem in analizo varnosti. V zaključku bomo podali prednosti in slabosti uporabe McElieceovega kriptosistema.

Poglavje 2

Kriptografija

Potreba po ohranjanju tajnosti sega daleč v zgodovino. Da bi to dosegli, so se razvili številni postopki za šifriranje, ki tako spremenijo sporočilo, da postane nerazumljivo. V tem poglavju bomo sledili knjigama [28] in [43].

Eden izmed ciljev kriptografije je izdelati kriptosistem, ki zagotavlja varen prenos sporočil med dvema ali več osebami tako, da je zadovoljeno neki podmnožici naslednjih štirih ciljev:

- *Zaupnost:* želimo zagotoviti, da lahko do podatkov dostopajo samo tisti, ki so jim ti namenjeni. Predstavlja glavni fokus kriptografije.
- *Celovitost podatkov:* želimo zagotoviti, da prvotno sporočilo ni bilo spremenjeno z nedovoljenimi sredstvi, vključno z vstavljanjem, brisanjem ali zamenjavo sporočila.
- *Avtentikacija:* želimo zagotoviti možnost preverjanja določenih lastnosti sporočila, na primer pristnosti pošiljatelja, prejemnika ali pa časa, ko je bilo sporočilo poslano.
- *Preprečevanje nepriznavanja:* želimo zagotoviti, da ni moč zanikati dane obljube ali storjenih dejanj. Potrebno je v primerih, kjer bi lahko prišlo do sporov, na primer pri spletnem nakupovanju.

V tem delu bomo obravnavali samo zagotavljanje zaupnosti, kar naredimo

s pomočjo *šifriranja*, kjer sporočilo tako spremenimo, da ni moč razbrati sporočila. Prejemnik pa ga lahko razbere tako, da sporočilo *dešifrira*.

Definicija 2.1. *Kriptosistem* je peterka $(\mathcal{B}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, za katero velja:

- $\mathcal{B}$ je končna množica *besedil* (angl. *plaintext*),
- $\mathcal{C}$ je končna množica *kriptogramov* (angl. *ciphertext*),
- $\mathcal{K}$ je končna množica *ključev*,
- $\mathcal{E} = \{E_k : \mathcal{B} \to \mathcal{C}; k \in \mathcal{K}\}$ je družina *kodirnih funkcij*,
- $\mathcal{D} = \{D_k : \mathcal{C} \to \mathcal{B}; k \in \mathcal{K}\}$ je družina *dekodirnih funkcij*,
- Za vsak $e \in \mathcal{K}$ obstaja $d \in \mathcal{K}$, da velja

$$D_d(E_e(x)) = x \text{ za vsak } x \in \mathcal{B}.$$

Šifriranje tako izvedemo s kodirno funkcijo in dobimo kriptogram, ki ga lahko dešifriramo z ustrezno dekodirno funkcijo, da dobimo prvotno besedilo. Pri klasičnih kriptosistemih sta bila ponavadi kodirni in dekodirni ključ enaka, zato je bila zelo pomembna tajnost samega ključa. Kodirni in dekodirni ključ pa nista nujno enaka.

Definicija 2.2. Kriptosistem $\mathcal{S}$ je *simetričen*, če poznamo učinkovit algoritem, ki za vsak kodirni ključ izračuna pripadajoči dekodirni ključ. Če kriptosistem ni simetričen, pravimo, da je *asimetričen*.

Čez leta so se razvili številni kriptosistemi, ki so bili bolj ali manj varni. Vsem je bila skupna ena velika težava, in sicer kako izmenjati kodirni ključ, da ta ne bi prišel v roke nepooblaščenim. Najvarnejši način, a tudi zelo zamuden, je bil osebna izročitev oziroma z uporabo zaupanja vrednega kurirja. Prva sta idejo asimetričnih kriptosistemov, kjer naj bi bila rešena težava izmenjave ključev, predstavila Diffie in Hellman [12]. Za razliko od ostalih kriptosistemov si uporabniki pri asimetričnem kriptosistemu ne delijo več skupnega skrivnega ključa, ampak ima vsak uporabnik svoj ključ, ki je sestavljen iz dveh komponent:

- kodirni ključ e, ki je *javen*: vsakdo lahko šifrira,
- dekodirni ključ d, ki je *tajen*: dešifrira lahko samo lastnik ključa.

Pri asimetričnem kriptosistemu imenujemo kodirni ključ tudi *javni ključ*, dekodirni ključ pa *zasebni ključ*. Sam kriptosistem imenujemo tudi *kriptosistem z javnim ključem*.

Varnost asimetričnih kriptosistemov temelji na tem, da samo iz javnega ključa e ne moremo v sprejemljivem času izračunati zasebnega ključa d. To ponavadi pomeni, da je za izračun zasebnega ključa treba rešiti kak težek problem, kot na primer:

- problem razcepa sestavljenega števila ali
- problem diskretnega logaritma.

Na problemu razcepa sestavljenega števila temelji sistem z javnim ključem RSA, ki je danes množično v uporabi. Kriptosistem so predstavili Rivest, Shamir in Adleman leta 1978 [38]. Drugi znan kriptosistem je ElGamalov kriptosistem [15], ki temelji na problemu diskretnega algoritma.

Kakor nas uči zgodovina, ima še tako na videz nezlomljiv kriptosistem, lahko šibke točke, ki jih bomo odkrili šele s časom. Za zdaj so trenutno uporabljani algoritmi dovolj varni, kar bi se pa lahko hitro spremenilo z izgradnjo kvantnega računalnika. Leta 1994 je Shor predstavil algoritem za kvantni računalnik , ki reši problem razcepa sestavljenega števila v polinomskem času [40]. Ta algoritem bi torej lahko uporabili za razbitje določenih kriptosistemov, med katerimi je tudi RSA. Dve leti kasneje je Grover predstavil nov kvantni algoritem, ki omogoča hitro preiskovanje velikih neurejenih tabel [21]. S pomočjo tega algoritma lahko razbijemo kriptosisteme, ki temeljijo na problemu diskretnega algoritma s pregledovanjem vseh možnih ključev.

Kljub prepričanju, da so kvantni računalniki oddaljeni še mnoga desetletja, pa je odkritje novih algoritmov, ki bi uspešno razbili trenutno uporabljane algoritme, sprožilo iskanje kriptosistema, na katerega kvantni računalnik ne bi imel bistvenega vpliva. Ugotovili so, da takšni sistemi tudi

že obstajajo, saj na kriptosisteme, ki uporabljajo kode za popravljanje napak, kvantni algoritmi ne vplivajo. V 6. poglavju bomo opisali enega izmed teh kriptosistemov, in sicer McElieceov kriptosistem, ki temelji na težkem problemu dekodiranja splošnih linearnih kodov.

Poglavje 3

Končni obsegi

Teorija končnih obsegov je ena izmed vej moderne algebre, katere začetki segajo v 17. in 18. stoletje. Prvi je pomembnejše objekte iz tega področja vpeljal Evariste Galois, zato ji včasih pravimo tudi *Galoisova teorija*. V zadnjih desetletjih je postala bolj zanimiva za praktično uporabo, predvsem v računalništvu. Dandanes je zelo pomembna v kombinatoriki in teoriji kodiranja. Definirali bomo nekatere algebraične strukture in predstavili nekatere izreke, ki jih bomo potrebovali v nadaljevanju pri Goppovih kodih. Prav tako bomo opisali Evklidov algoritem za polinome. V tem poglavju bomo sledili knjigama [25] in [48].

Definicija 3.1. Naj bo G množica in $\circ$ dvočlena operacija na G, kjer za vsak par $a, b \in G$ velja $a \circ b \in G$. Potem je $(G, \circ)$ *grupa*, če velja

1. operacija $\circ$ je *asociativna*:

$$(a \circ b) \circ c = a \circ (b \circ c) \text{ za vse } a, b, c \in G,$$

2. obstaja *enota* $e \in G$, za katero velja

$$e \circ a = a \circ e \text{ za vsak } a \in G,$$

3. vsak element ima *inverz*, torej za vsak $a \in G$ obstaja $b \in G$, da velja $a \circ b = b \circ a = e$.

Inverz elementa a označimo a^{-1}.

Grupa $(G, \circ)$ je *komutativna* ali *Abelova*, če za vsak par $a, b \in G$, velja $a \circ b = b \circ a$. Ponavadi bomo grupo označevali samo z G namesto $(G, \circ)$, če bo jasno, katero operacijo uporabljamo.

Abelovo grupo pogosto pišemo aditivno, torej operacija $\circ$ predstavlja seštevanje. Enoto potem označimo z 0, inverzni element pa je nasprotni element in ga pišemo $-a$.

Grupa G ima lahko končno ali neskončno število elementov. V prvem primeru pravimo, da je grupa *končna*. Naj bo G končna grupa in $\alpha \in G$. Definiramo $\alpha^0 = e$ in $\alpha^i = \underbrace{\alpha \circ \alpha \circ \cdots \circ \alpha}_{i \text{ - krat}}$ za naravno število i. *Red elementa* α je najmanjše naravno število s, da je $\alpha^s = e$ in ga označimo z $|\alpha|$. *Red grupe* G je število elementov G. Označimo ga z $|G|$.

Definicija 3.2. Naj bo $(G, \circ)$ grupa in H neprazna podmnožica G. Potem je H *podgrupa* G, če je $(H, \circ)$ grupa.

O moči grupe in njenih podgrup govori naslednji izrek, ki mu pogosto pravimo Lagrangeev izrek, glej [29].

Izrek 3.3. *Naj bo G končna grupa in H podgrupa G. Potem red podgrupe H deli red grupe G.*

Najbolj preproste grupe so ciklične grupe.

Definicija 3.4. Grupa G je *ciklična*, če obstaja tak $\alpha \in G$, da za vsak $b \in G$ obstaja nenegativno celo število i, da velja $b = \alpha^i$. Takemu elementu α pravimo *generator grupe*.

Vsak element grupe lahko generira novo grupo.

Trditev 3.5. *Če je G grupa in $\alpha \in G$, potem množica vseh potenc α tvori ciklično podgrupo G. Pravimo, da je podgrupa generirana z α in jo označimo z $\langle \alpha \rangle$.*

Povezavo med redom grupe in redom elementa podaja naslednji izrek.

Izrek 3.6. *Naj bo G končna grupa in $g \in G$. Potem red elementa g deli red grupe G.*

Dokaz. Naj bo g element grupe G. Ker je grupa G končna, je tudi red elementa g končen. Tedaj je $\langle g \rangle = \{e, g, g^2, \cdots, g^{|g|-1}\}$ podgrupa grupe G in velja $|\langle g \rangle| = |g|$. Po Lagrangeevem izreku red podgrupe $\langle g \rangle$ deli red grupe G. Torej red elementa g res deli red grupe G.

□

Oznaka 3.7. Za množico K je $K^* = K \backslash \{0\}$ množica neničelnih elementov množice K.

V grupi imamo samo eno operacijo, pri običajnem računanju pa ponavadi prav prideta dve operaciji - seštevanje in množenje.

Definicija 3.8. Naj bo K množica z dvema operacijama. Potem je $(K, +, \circ)$ *kolobar*, če velja

- $(K, +)$ je Abelova grupa,
- operacija $\circ$ je asociativna: za vsako trojico $a, b, c \in K$ je

$$(a \circ b) \circ c = a \circ (b \circ c).$$

- operaciji $+$ in $\circ$ povezuje distributivnost: za vsako trojico $a, b, c \in K$ je

$$a \circ (b + c) = (a \circ b) + (a \circ c) \text{ in}$$

$$(a + b) \circ c = (a \circ c) + (b \circ c).$$

Zgled 3.9. Pogosto imamo opravka z naslednjimi kolobarji.

1. Vzemimo množico celih števil $\mathbb{Z}$ in operaciji seštevanja in množenja. Potem je $(\mathbb{Z}, +, \cdot)$ kolobar.
2. Vzemimo množico $\mathbb{Z}_n = \{0, \ldots, n-1\}$ in operaciji seštevanja in množenja po modulu n, potem je $(\mathbb{Z}_n, +, \cdot)$ kolobar.

3. Vzemimo množico polinomov $\mathbb{Z}[x]$ s koeficienti v $\mathbb{Z}$ in operaciji seštevanja in množenja polinomov. Potem je $(\mathbb{Z}[x], +, \cdot)$ kolobar.

Definicija 3.10. Naj bo $(K, +, \circ)$ kolobar. Potem je $(K, +, \circ)$ *obseg*, če je $(K^*, \circ)$ grupa.

Obseg je *komutativen*, če je grupa $(K, \circ)$ komutativna. Obseg je *končen*, če je množica K končna. Končen obseg imenujemo tudi *Galoisov obseg* in za končni obseg s q elementi uporabljamo oznako $GF(q)$.

Definicija 3.11. Naj bo K obseg. Število p imenujemo *karakteristika obsega*, če za vsak $x \in K$ velja

$$px = \underbrace{x + \cdots + x}_{p \text{ - krat}} = 0$$

.

Definicija 3.12. Naj bosta $(G_1, +_1, \cdot_1)$ in $(G_2, +_2, \cdot_2)$ obsega. Preslikava $\varphi : G_1 \to G_2$ je *homomorfizem obsegov*, če velja

- $\varphi(a +_1 b) = \varphi(a) +_2 \varphi(b)$ za vse $a, b \in G_1$ in
- $\varphi(a \cdot_1 b) = \varphi(a) \cdot_2 \varphi(b)$ za vsaka $a, b \in G_1$.

Če je φ bijektivna preslikava, potem je φ *izomorfizem*. Če med G_1 in G_2 obstaja izomorfizem, potem sta obsega G_1 in G_2 *izomorfna*.

Za Galoisove obsege lahko izpeljemo številne lastnosti.

Izrek 3.13. *Za vsak $n \in \mathbb{N}$ veljajo naslednje trditve:*

- *Če n ni potenca praštevila, potem ne obstaja obseg moči n.*
- *Če $n = p^m$, kjer je p praštevilo in $m > 0$, potem obstaja natanko en obseg moči n do izomorfizma natančno. Označimo ga z $\mathbb{F}_n = GF(n)$. Če je $m = 1$, potem je $\mathbb{F}_p = \mathbb{Z}_p$. Če pa je $m > 1$, potem je $\mathbb{F}_{p^m} \neq \mathbb{Z}_{p^m}$.*
- *Obseg $GF(p^m)$ ima karakteristiko enako p.*
- *Za vsak $a \in GF(q)$ velja $a^q = a$.*

- $(GF(q)^*, \circ)$ *je ciklična grupa reda* $q-1$. *Njene generatorje imenujemo* primitivni elementi.

Pogosto se za predstavitev elementov obsega $GF(q)$, kjer je $q = p^m$ in p praštevilo, uporablja predstavitev s polinomi. Če je $m = 1$, potem imamo obseg $\mathbb{Z}_p$ in za računanje uporabljamo operacije po modulu p. V enačbah bomo za modul uporabljali oznako mod. Za $m \geq 2$ bomo računali po modulu $f(x)$, kjer je f izbrani nerazcepni polinom stopnje m.

Pri množenju in seštevanju polinomov po modulu $f(x)$ najprej polinoma seštejemo oziroma zmnožimo in dobimo $g(x)$. Pri tem koeficiente seštevamo oziroma množimo po modulu p. Če je stopnja $g(x)$ manjša od stopnje $f(x)$ smo dobili iskani polinom, sicer pa $g(x)$ še delimo s $f(x)$ in za rešitev vzamemo ostanek pri deljenju.

Zgled 3.14. Pogosto imamo opravka z naslednjimi obsegi.

1. Naj bo p praštevilo in operaciji seštevanje po modulu p in množenje po modulu p. Potem je $(\mathbb{Z}_p, +, \cdot)$ obseg.

2. Naj bo $\mathbb{Z}_p[x]/f(x) = \{r(x) : \deg r < \deg f = m\}$, kjer je $f(x)$ nerazcepen polinom stopnje m, in operaciji seštevanje in množenje polinomov po modulu $f(x)$. Potem je $\mathbb{Z}_p[x]/f(x)$ končen obseg reda p^m. V primeru, da $f(x)$ ni nerazcepen, je $\mathbb{Z}_p[x]/f(x)$ kolobar z enoto za množenje.

Da bo $\mathbb{Z}_p/f(x)$ res obseg, mora imeti vsak njegov neničelni element tudi inverz. Izračunali ga bomo s pomočjo razširjenega Evklidovega algoritma. Ključen pri izpeljavi tega algoritma bo naslednji izrek.

Izrek 3.15 (Osnovni izrek o deljenju polinomov)**.** *Naj bosta* $f(x), g(x) \in \mathbb{F}[x], g(x) \neq 0$. *Tedaj obstajata enolično določena polinoma* $q(x), r(x) \in \mathbb{F}[x]$, *kjer je* $\deg(r) < \deg(g)$, *da velja*

$$f(x) = q(x) \cdot g(x) + r(x). \tag{3.1}$$

Dokaz. Recimo, da pri deljenju $f(x)$ z $g(x)$ količnik in ostanek nista enolično določena. Potem veljata enakosti

$$\begin{aligned} f(x) &= q_1(x)g(x) + r_1(x) \\ f(x) &= q_2(x)g(x) + r_2(x), \end{aligned}$$

kjer je $q_1(x) \neq q_2(x)$ ali $r_1(x) \neq r_2(x)$ Tedaj za vsak $x \in \mathbb{F}$ velja enakost

$$q_1(x)g(x) + r_1(x) = q_2(x)g(x) + r_2(x),$$

kar lahko preuredimo v

$$g(x)(q_1(x) - q_2(x)) = r_2(x) - r_1(x). \tag{3.2}$$

Stopnja polinoma na levi strani enačbe (5.10) je večja ali enaka stopnji $g(x)$, razen če je $q_1(x) = q_2(x)$ za vsak $x \in \mathbb{F}$. Stopnja polinoma na desni strani pa je manjša od stopnje $g(x)$, saj imata oba ostanka stopnjo manjšo od stopnje $g(x)$. Iz tega sledi, da morata imeti polinoma na obeh straneh stopnjo 0, torej mora veljati $q_1(x) = q_2(x)$ za vsak $x \in \mathbb{F}$ in $r_1(x) = r_2(x)$. Prišli smo do protislovja, torej sta količnik $q(x)$ in ostanek $r(x)$ enolično določena.

□

Polinom $g(x)$ deli $f(x)$, če je ostanek $r(x)$ enak 0. Deljenje polinomov lahko predstavimo tudi kot ulomek

$$f(x) : g(x) = \frac{f(x)}{g(x)},$$

kjer je $g(x) \neq 0$. Pogosto želimo okrajšati ulomek, zato nas zanima, če imata polinoma skupne faktorje.

Definicija 3.16. Naj bosta $g(x), h(x) \in \mathbb{Z}_p[x]$ in vsaj eden od njiju različen od nič. Potem je *največji skupni delitelj* $g(x)$ in $h(x)$ moničen polinom največje stopnje v $\mathbb{Z}_p$, ki deli oba polinoma. Označimo ga z $\gcd(g(x), h(x))$.

Največji skupni delitelj dveh polinomov lahko izračunamo s pomočjo Evklidovega algoritma za polinome.

Algoritem 3.1. (Evklidov algoritem za $\mathbb{F}[x]$)
Vhod: *polinoma* $g(x), h(x) \in \mathbb{F}[x]$.
Izhod: *največji skupni delitelj* $g(x)$ *in* $h(x)$.

1. *Določimo* $r_0(x) = g(x)$ *in* $r_1(x) = h(x)$.
2. *Dokler je ostanek* $r_i(x) \neq 0$ *ponavljamo* $r_{i+1} = r_{i-1} \mod r_i$.
3. *Naj bo* k *najmanši indeks, da je* $r_k = 0$. *Potem vrnemo* r_{k-1}.

Zaporedje ostankov $r_i(x)$ strogo pada. Torej po največ $\deg(h(x))$ korakih dobimo ničelni ostanek; označili smo ga z r_k. Ker imajo $g(x), h(x)$ in $h(x), g(x) \mod h(x)$ enake delitelje, se množica skupnih deliteljev ni spremenila z enim korakom Evklidovega algoritma. Vsi pari (r_i, r_{i+1}) imajo tako enako množico skupnih deliteljev. Iz tega sledi, da so skupni delitelji $g(x)$ in $h(x)$ enaki kot skupni delitelji r_{k-1} in 0. Torej je r_{k-1} res največji skupni delitelj $g(x)$ in $h(x)$.

Ni težko videti, da je število korakov Evklidovega algoritma logaritemsko. Skupno časovno zahtevnost nam poda naslednji izrek, glej [28].

Trditev 3.17. *Naj bo stopnja polinomov* $g(x)$ *in* $h(x)$ *največ* m. *Potem ima algoritem 3.1 časovno zahtevnost* $O(m^2)$ *operacij v* $\mathbb{Z}_p$ *ali ekvivalentno* $O(m^2(\log p)^2)$ *bitnih operacij.*

Evklidov algoritem lahko tudi razširimo tako, da ne izračuna samo največjega skupnega delitelja.

Algoritem 3.2. (Razširjeni Evklidov algoritem za $\mathbb{Z}_p[x]$)
Vhod: *polinoma* $g(x), h(x) \in \mathbb{Z}_p[x]$.
Izhod: $d(x) = gcd(g(x), h(x))$ *in polinoma* $s(x), t(x) \in \mathbb{Z}_p[x]$, *ki zadoščata enačbi* $s(x)g(x) + t(x)h(x) = d(x)$.

1. *Če* $h(x) = 0$, *potem priredimo* $d(x) = g(x), s(x) = 1, t(x) = 0$ *in vrnemo* $(d(x), s(x), t(x))$.
2. *Nastavimo* $s_2(x) = 1, s_1(x) = 0, t_2(x) = 0, t_1(x) = 1$.

3. *Dokler* $h(x) \neq 0$ *ponavljamo:*

   3.1. *Naj bo* $q(x)$ *kvocient pri deljenju* $g(x)$ *s* $h(x)$ *in*
   $r(x) = g(x) - h(x)q(x)$.

   3.2. $s(x) = s_2(x) - q(x)s_1(x)$ *in*
   $t(x) = t_2(x) - q(x)t_1(x)$.

   3.3. $g(x) = h(x), h(x) = r(x)$.

   3.4. $s_2(x) = s_1(x), s_1(x) = s(x)$ *in*
   $t_2(x) = t_1(x), t_1(x) = t(x)$.

4. *Priredimo* $d(x) = g(x), s(x) = s_2(x), t(x) = t_2(x)$.

5. *Vrnemo* $(d(x), s(x), t(x))$.

Število korakov algoritma 3.2 je enako kot število korakov algoritma 3.1. Skupno časovno zahtevnost podaja spodnji izrek, glej [28].

Trditev 3.18. *Naj bo stopnja polinomov* $g(x)$ *in* $h(x)$ *največ* m. *Potem ima algoritem 3.2 ob primerni implementaciji časovno zahtevnost* $O(m^2)$ *operacij v* $\mathbb{Z}_p$ *ali ekvivalentno* $O(m^2(\log p)^2)$ *bitnih operacij.*

Iz razširjenega Evklidovega algoritma sledi naslednji izrek.

Izrek 3.19. *Naj bosta* $f(x), g(x) \in \mathbb{F}[x]$ *in naj bo* $d(x) = \gcd(f(x), g(x))$. *Tedaj obstajata polinoma* $s(x)$ *in* $t(x)$, *da velja*

$$d(x) = f(x) \cdot s(x) + g(x) \cdot t(x).$$

S pomočjo razširjenega Evklidovega algoritma lahko izračunamo inverz polinoma $a(x)$ v $\mathbb{F}_p/f(x)$, kjer je $f(x)$ nerazcepen polinom. Inverz elementa $a(x)$ označimo z $a(x)^{-1}$. Ker je $f(x)$ nerazcepen polinom, je $a(x)$ tuj z $f(x)$. Zato je enačba

$$1 = f(x) \cdot s(x) + a(x) \cdot t(x)$$

rešljiva po izreku 3.19. Gornjo enačbo lahko zapišemo kot

$$a(x) \cdot t(x) \equiv 1 \mod f(x).$$

Torej je $t(x) = a(x)^{-1}$ iskani inverz.

Poglavje 4

Kodi za popravljanje napak

Eden izmed ciljev kodiranja je zagotoviti zanesljiv prenos sporočil po komunikacijskem kanalu tako, da je mogoče (do neke mere) popraviti napake, ki so nastale med prenosom. To dosežemo s kodi za popravljanje napak, kjer prvotnemu sporočilu priredimo daljše sporočilo, ki nam pomaga popraviti morebitne nastale napake. V nadaljevanju bomo sledili knjigi [49].

4.1 Bločni kodi

Poseben razred kodov za popravljanje napak so bločni kodi, pri katerih sporočilo zakodiramo v kodne besede iste dolžine.

Definicija 4.1. *Bločni kod* $\mathcal{C}$ nad abecedo Σ je končna podmnožica Σ^n. Vsakemu elementu $\mathcal{C}$ pravimo *kodna beseda*.

Če je $\Sigma = GF(2)$, imenujemo kod nad Σ *dvojiški kod*.

Za merjenje razdalj med kodnimi besedami uporabljamo Hammingovo razdaljo.

Definicija 4.2. Naj bosta $X, Y \in \Sigma^n$. *Hammingova razdalja* med x in y je

$$d(x, y) = |\{i; x_i \neq y_i\}|.$$

Ni težko preveriti, de je Hammingova razdalja res metrika, saj velja:

1. $d(x, y) \geq 0$ *(nenegativnost)*,
2. $d(x, y) = 0$ natako tedaj, ko je $x = y$,
3. $d(x, y) = d(y, x)$ *(simetričnost)*,
4. $d(x, y) \leq d(x, z) + d(z, y)$ *(trikotniška neenakost)*.

Za kodne besede pogosto uporabljamo tudi pojem teže.

Definicija 4.3. *Teža* besede x je

$$t(x) = |\{i; x_i \neq 0\}|.$$

Razdaljo in težo med seboj povezuje naslednji izrek.

Izrek 4.4. *Če sta v Σ definirani operaciji seštevanja in odštevanja, velja:*

$$d(x, y) = t(x - y).$$

Dokaz. Naj bosta $x, y \in \Sigma^n$.

$$\begin{aligned} d(x, y) &= |\{i; x_i \neq y_i\}| \\ &= |\{i; x_i - y_i \neq 0\}| \\ &= t(x - y) \end{aligned}.$$

□

Zelo pogosto nas pri kodih zanima tudi, kako daleč sta najbližji kodni besedi.

Definicija 4.5. *Razmaknjenost koda $\mathcal{C}$* (angl. *minimum distance*) je

$$d(\mathcal{C}) = \min_{x,y \in \mathcal{C}, x \neq y} d(x, y).$$

Oznaka 4.6. Kod $\mathcal{C}$ je (n, M, d)-kod nad abecedo Σ, če ima bločno dolžino n, $\mid \mathcal{C} \mid = M$ in $d(\mathcal{C}) = d$.

Recimo, da imamo sporočilo s, ki ga želimo poslati tako, da ga bo mogoče prebrati, tudi če pri prenosu pride do napak. Sporočilu priredimo kodno besedo x iz izbranega koda $\mathcal{C}$. Pravimo, da sporočilo *zakodiramo*. Kodno besedo x potem pošljemo po nezanesljivem komunikacijskem kanalu. Prejemnik prejme besedo $y = x + e$, kjer je e *napaka*. Prejemnik poišče kodno besedo $x' \in \mathcal{C}$, ki je najbližje y glede na Hammingovo razdaljo. Pravimo, da besedo y *dekodiramo*. Iz x' potem pridobi sporočilo s'. Želimo si, da je $x = x'$ oziroma $s = s'$, kar je res, če je teža napake $t(e)$ dovolj majhna.

Definicija 4.7. Kod $\mathcal{C}$ dolžine n *odkrije* r napak, če $x + e \notin \mathcal{C}$ za vse $x \in \mathcal{C}$ in $e \in \Sigma^n$:

$$1 \leq t(e) \leq r.$$

Kod $\mathcal{C}$ *popravi* r napak, če

$$d(x + e, x) < d(x + e, y)$$

za vsaka $x, y \in \mathcal{C}$ in vsak $e \in \Sigma^n : t(e) \leq r$.

Na sposobnost koda, da odkrije in popravi določeno število napak, vpliva razmaknjenost koda.

Izrek 4.8. *Naj bo $\mathcal{C}$ (n, M, d)-kod. Potem velja:*

1. *kod $\mathcal{C}$ odkrije $d - 1$ napak,*
2. *kod $\mathcal{C}$ popravi $\lfloor \frac{d-1}{2} \rfloor$ napak.*

Dokaz. 1. Naj bo $e \in \Sigma^n$ tak, da je $1 \leq t(e) \leq d - 1$. Izberemo poljuben $x \in \mathcal{C}$. Potem za razdaljo med x in $x + e$ velja

$$d(x, x + e) = t(x + e - x) = t(e) \leq d - 1,$$

iz česar sledi, da $x + e \notin \mathcal{C}$, ker za poljubni kodni besedi iz $\mathcal{C}$ velja, da je razdalja enaka vsaj d. Torej kod res odkrije $d - 1$ napak.

2. Naj bo $e \in \Sigma^n$ tak, da je $t(e) \leq \lfloor \frac{d-1}{2} \rfloor$. Izberemo poljubna $x, y \in \mathcal{C}$, kjer $x \neq y$. Potem za razdaljo med x in $x + e$ velja

$$d(x, x + e) = t(e) \leq \lfloor \frac{d-1}{2} \rfloor \leq \frac{d-1}{2}.$$

To je ekvivalentno $2d(x, x + e) \leq d - 1 \leq d(x, y) - 1$. Z uporabo trikotniške neenakosti dobimo neenakost

$$2d(x, x + e) \leq d(x, x + e) + d(x + e, y) - 1.$$

Na obeh straneh neenakosti odštejemo $d(x, x + e)$ in dobimo $d(x, x + e) \leq d(x + e, y) - 1$. Sledi, da je $d(x, x + e) < d(x + e, y)$, torej kod res popravi $\lfloor \frac{d-1}{2} \rfloor$ napak.

□

Dobro je, da je razmaknjenost koda velika, saj bo tako kod lahko popravil veliko napak.

4.2 Linearni kodi

V praksi se najpogosteje uporabljajo linearni kodi. Ker so tudi Goppovi kodi linearni, se bomo v tem delu omejili le na linearne kode. Naj bo v nadaljevanju $\Sigma = GF(q)$, kjer je q potenca praštevila.

Definicija 4.9. Kod $\mathcal{C} \subseteq \Sigma^n$ je *linearen*, če je vektorski podprostor Σ^n. Torej velja: za vsak $c_1, c_2 \in \mathcal{C}$ in $\alpha, \beta \in \Sigma$ je tudi $\alpha c_1 + \beta c_2 \in \mathcal{C}$.

Pri linearnih kodih je število kodnih besed povezano z dimenzijo koda.

Trditev 4.10. *Naj bo $\mathcal{C}$ linearen (n, M, d)-kod nad Σ dimenzije k. Potem je $M = q^k$.*

Oznaka 4.11. Linearen (n, q^k, d)-kod $\mathcal{C}$ nad Σ označimo z $[n, k, d]$.

Za linearen kod lahko razmaknjenost izračunamo tudi s pomočjo teže.

Izrek 4.12. *Naj bo $\mathcal{C}$ linearen kod. Potem je*

$$d(\mathcal{C}) = \min_{x \in \mathcal{C}, x \neq 0} t(x).$$

Dokaz. Razmaknjenost koda je enaka

$$d(\mathcal{C}) = \min_{x,y \in \mathcal{C}, x \neq y} d(x,y) = \min_{x,y \in \mathcal{C}, x-y \neq 0} t(x-y).$$

Ker je kod linearen, $x - y$ preteče vse kodne besede, zato je

$$d(\mathcal{C}) = \min_{z \in \mathcal{C}, z \neq 0} t(z).$$

□

Lažje je izračunati težo M besed kot razdaljo $\binom{M}{2}$ besed, zato za računanje razmaknjenosti linearnih kodov uporabljamo težo, kar pa je še vedno zelo zamudno, če je besed veliko.

Definicija 4.13. *Generatorska matrika* G koda $\mathcal{C}$ je matrika velikosti $k \times n$, katere vrstice sestavljajo bazo koda $\mathcal{C}$.

Generatorska matrika seveda ni enolično določena.

S pomočjo generatorske matrike lahko izvedemo *kodiranje*. Naj bo $\mathcal{C}$ linearen $[n,k,d]$-kod nad Σ, kjer je $\{x^{(1)}, x^{(2)}, \ldots, x^{(k)}\}$ baza za $\mathcal{C}$. Ena izmed generatorskih matrik je potem

$$G = \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(k)} \end{pmatrix}.$$

Sporočilo $s \in \Sigma^k$, kjer je $s = s_1 s_2 \ldots s_k$ za $s_i \in \Sigma$, lahko zakodiramo tako

$$y = s \cdot G = s_1 \cdot x^{(1)} + s_2 \cdot x^{(2)} + \cdots + s_k \cdot x^{(k)}.$$

Definicija 4.14. Koda $\mathcal{C}_1$ in $\mathcal{C}_2$ sta *ekvivalentna*, če lahko $\mathcal{C}_2$ dobimo iz $\mathcal{C}_1$ z zaporedjem transformacij $\tau_1, \ldots, \tau_s$, kjer je vsak τ_i

- permutacija vrstic kodne matrike,
- permutacija stolpcev kodne matrike,
- permutacija simbolov v izbranem stolpcu kodne matrike.

Oznaka 4.15. Če sta koda $\mathcal{C}_1$ in $\mathcal{C}_2$ ekvivalentna, pišemo $\mathcal{C}_1 \sim \mathcal{C}_2$

Ekvivalentna koda lahko popravita isto število napak, saj velja naslednja trditev.

Trditev 4.16. *Če sta koda $\mathcal{C}_1$ in $\mathcal{C}_2$ ekvivalentna, potem velja $d(\mathcal{C}_1) = d(\mathcal{C}_2)$.*

Za vsak kod $\mathcal{C}$ lahko definiramo njegov *dualni kod* $\mathcal{C}^\perp$.

Definicija 4.17. Naj bo $\mathcal{C}$ $[n, k, d]$-kod. Potem je

$$\mathcal{C}^\perp = \{x \in \Sigma^n; c^T x = 0 \text{ za vsak } c \in \mathcal{C}\}$$

dualni kod koda $\mathcal{C}$. Tudi $\mathcal{C}^\perp$ je linearen kod. Generatorsko matriko dualnega koda $\mathcal{C}^\perp$ imenujemo tudi *nadzorna matrika* koda $\mathcal{C}$. Običajno jo bomo označevali s H.

Opazimo, da je $\mathcal{C}^\perp$ $(n-k)$-dimenzionalen podprostor Σ^n, ortogonalen na $\mathcal{C}$. Vsak kod lahko podamo z generatorsko matriko ali nadzorno matriko, saj lahko eno izračunamo iz druge s pomočjo naslednje trditve.

Izrek 4.18. *Naj bosta $G \in GF(q)^{k\times n}$ in $H \in GF(q)^{(n-k)\times n}$, kjer sta matriki ranga k oziroma $n-k$. Potem je G je generatorska matrika in H je nadzorna matrika nekega linearnega koda natako tedaj, ko velja $GH^T = 0$.*

Dokaz. ($\Rightarrow$) Naj bo $Gx^T = 0$ za vsak $x \in \mathcal{C}^\perp$. Ker so vrstice H iz $\mathcal{C}^\perp$ je $GH^T = 0$.
($\Leftarrow$) Naj bo $\mathcal{C}$ kod, katerega bazo sestavljajo vrstice G. Potem je $\mathcal{C}^\perp$ ničelni podprostor matrike G, iz česar sledi, da ima dimenzijo $n -$ rang$(G) = n - k$. Torej vrstice H pripadajo $\mathcal{C}^\perp$, so linearno neodvisne in jih je $n - k$. Sledi, da sestavljajo bazo $\mathcal{C}^\perp$, torej je H nadzorna matrika za $\mathcal{C}$.

□

S pomočjo nadzorne matrike lahko preprosto preverimo, ali je dana beseda kodna beseda.

Posledica 4.19. *Naj bo $\mathcal{C}$ linearen kod in H njegova nadzorna matrika. Potem je $x \in \mathcal{C}$ natanko tedaj, ko je $xH^T = 0$.*

Poglejmo, kako si lahko pri dekodiranju linearnih kodov pomagamo z nadzorno matriko. Naj bo $\mathcal{C}$ linearen $[n, k, d]$-kod, $x \in \mathcal{C}$ poslana beseda, $y \in \Sigma^n$ prejeta beseda in $s = \lfloor \frac{d-1}{2} \rfloor$. Potem je napaka enaka $e = y - x$, kjer je $e \in \Sigma^n$.

Definicija 4.20. *Sindrom* besede y je Hy^T.

Torej je $Hy^T = H(x + e)^T = Hx^T + He^T$. Ker je $x \in \mathcal{C}$, je po posledici **??** Hx^T enak nič in dobimo $Hy^T = He^T$. Torej sindrom prejete besede primerjamo s sindromi napak in poiščemo ujemanje. V težave bi prišli, če bi bili sindromi različnih napak enaki. Spodnja trditev dokaže, da so sindromi napak, ki jih je z danim kodom mogoče popraviti, med seboj različni.

Trditev 4.21. *Naj bodo $s = \lfloor \frac{d-1}{2} \rfloor$ in $e_1, e_2 \in \Sigma^n$ taki, da velja $t(e_1) \leq s$, $t(e_2) \leq s$ in $e_1 \neq e_2$. Potem je $He_1^T \neq He_2^T$.*

Dokaz. Recimo, da velja $He_1^T = He_2^T$. Potem je $He_1^T - He_2^T = 0$ in $H(e_1 - e_2)^T = 0$, iz česar po posledici 4.19 sledi, da je $e_1 - e_2 \in \mathcal{C}$. Poglejmo si zdaj težo $e_1 - e_2$, za katero velja $t(e_1 - e_2) = d(e_1, e_2)$. Če uporabimo trikotniško neenakost, dobimo $t(e_1, e_2) \leq d(e_1, 0) + d(0, e_2) = t(e_1) + t(e_2)$. Torej je $t(e_1, e_2)$ največ $2s$ in je manjši od d. Ker je $e_1 - e_2$ kodna beseda in mora imeti neničelna kodna beseda težo vsaj d, sledi da je $e_1 - e_2 = 0$. Torej sta e_1 in e_2 enaka, kar je v protislovju s predpostavko. Torej velja $He_1^T \neq He_2^T$. □

Za dekodiranje prejete besede y najprej sestavimo urejeno tabelo parov (He^T, e) za $e \in \Sigma^n$, kjer je $t(e) \leq s$. Izračunamo sindrom Hy^T in ga poiščemo v tabeli. Če ga v tabeli ni, lahko zahtevamo ponoven prenos besede, saj je število napak večje od s. Sicer vzamemo pripadajoči e in izračunamo $x = y - e$.

Zgled 4.22. Vzemimo linearni $[5, 2, 3]$-kod nad $GF(2)$, podan z nadzorno matriko

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Ni težko preveriti, da je razmaknjenost enaka 3, zato lahko kod popravi $\lfloor \frac{3-1}{2} \rfloor = 1$ napako. Sestavimo tabelo parov.

e	$(He^T)^T$
00000	000
00001	001
00010	010
00100	100
01000	110
10000	101

Recimo, da smo prejeli besedo $y = (1, 1, 1, 1, 1)$. Izračunamo sindrom $Hy^T = (1, 0, 0)^T$ in ga poiščemo v tabeli. Pripadajoča napaka je $e = (0, 0, 1, 0, 0)$, torej je poslana beseda enaka $x = y - e = (1, 1, 0, 1, 1)$.

Zanima nas, kako hitro je dekodiranje s pomočjo sindromov. Če imamo $[n, k, d]$-kod nad $GF(q)$, ki lahko popravi $s = \lfloor \frac{d-1}{2} \rfloor$ napak, je število vseh možnih vektorjev napak enako

$$N = 1 + n \cdot (q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{s}(q-1)^s.$$

Za vsak vektor napak moramo izračunati sindrom, za kar potrebujemo n^2 operacij. Torej za izračun elementov tabele potrebujemo Nn^2 operacij. Za izračun sindroma prejete besede pa potrebujemo še $O(n^2)$ operacij. Če smo sindrome shranjevali v zgoščeno tabelo, lahko iskanje v tabeli izvedemo v času $O(1)$. Torej ima dekodiranje s sindromi časovno zahtevnost vsaj $O(Nn^2)$. Vidimo, da je dekodiranje s sindromi v primeru majhne razmaknjenosti lahko še dovolj dobro, v primeru večje razmaknjenosti pa postane prezahtevno, saj je N lahko eksponentne velikosti glede na generatorsko matriko velikosti $k \times n$,

s katero podamo kod. Prav tako ne pride v poštev neposredno dekodiranje, kjer iščemo najbližjo kodno besedo, za kar potrebujemo $O(q^k \cdot n)$ časa. Izkaže se, da je v splošnem dekodiranje linearnega koda NP-težek problem [3]. Torej ne poznamo nobenega algoritma, ki bi ta problem rešil v polinomskem času.

Tudi računanje razmaknjenosti koda ni lahek problem [46]. Zato so bile izpeljane razne zgornje meje za razmaknjenost. Ena izmed njih je Singletonova meja.

Izrek 4.23 (Singletonova meja)**.** *Naj bo $\mathcal{C}$ $[n, k, d]$-kod nad Σ. Potem velja $d \leq n - k + 1$.*

Dokaz. Naj bo $\mathcal{C}$ $[n, k, d]$-kod nad Σ. Zložimo vse besede iz $\mathcal{C}$ v matriko K velikosti $q^k \times n$. Če prečrtamo $d - 1$ stolpcev matrike K, so vse besede še vedno različne med seboj, saj se razlikujejo na vsaj d mestih. Vseh besed dolžine $n - d + 1$ nad Σ je q^{n-d+1}. Torej je $M = q^k \leq q^{n-d+1}$, iz česar sledi, da je $d \leq n - k + 1$.

□

Posledica 4.24. *Linearen $[n, k, d]$-kod popravi največ $\lfloor \frac{n-k}{2} \rfloor$ napak.*

Dokaz. Na podlagi izreka 4.8 lahko kod popravi $s = \lfloor \frac{d-1}{2} \rfloor$ napak. Ob upoštevanju Singletonove meje (izrek 4.23) lahko kod popravi $s \leq \lfloor \frac{n-k}{2} \rfloor$ napak.

□

V praksi je lahko postopek za popravljanje napak lahko zelo kompleksen. Kot smo videli je neposredno dekodiranje ali s pomočjo sindromov prezahtevno v primeru večje razmaknjenosti koda. Na srečo pa obstajajo določeni razredi linearnih kodov, za katere se da dekodiranje izvesti dokaj učinkovito. Kot primer takega koda bo v nadaljevanju predstavljen Goppov kod.

Poglavje 5

Goppov kod

Leta 1970 je V. D. Goppa predstavil nov razred linearnih kodov za popravljanje napak [20]. Goppovi kodi so posplošitev bolj znanih kodov BCH [8, 22]. Goppovi kodi so pomembni, ker imajo veliko razmaknjenost in zanje poznamo učinkovit algoritem za dekodiranje, ki temelji na algoritmu za dekodiranje kodov BCH. V nadaljevanju bomo sledili knjigama [29] in [44].

Kljub temu, da se v praksi najpogosteje uporablja dvojiški Goppov kod, definirajmo najprej splošni Goppov kod. Izberimo praštevilo q. Naj bosta $L = \{\alpha_1, ..., \alpha_n\} \subseteq GF(q^m)$ in $g(z)$ polinom stopnje t nad $GF(q^m)$,

$$g(z) = \sum_{i=0}^{t} g_i z^i, \tag{5.1}$$

taka, da velja $g(\alpha_i) \neq 0$ za vsak $\alpha_i \in L$.

Trditev 5.1. *V kolobarju $GF(q^m)[z]/g(z)$ ima $z - \alpha_i$ enoličen inverz, ki je enak*

$$(z - \alpha_i)^{-1} = -\frac{g(z) - g(\alpha_i)}{z - \alpha_i} g(\alpha_i)^{-1}.$$

Dokaz. Element $a(x) \in GF(q^m)[z]/g(z)$ ima inverz $a(x)^{-1}$, če je $a(x)$ tuj z $g(z)$, kar sledi iz izreka 3.19. Ker velja $g(\alpha_i) \neq 0$ za vsak $\alpha_i \in L$, sledi da α_i ni ničla polinoma $g(z)$. Torej sta $z - \alpha_i$ in $g(z)$ tuja in inverz $(z - \alpha_i)^{-1}$ obstaja.

Pokažimo najprej, da je $\frac{g(z)-g(\alpha_i)}{z-\alpha_i}$ polinom. Če uporabimo enačbo (5.1), lahko vidimo, da velja naslednje.

$$\begin{aligned}\frac{g(z)-g(\alpha_i)}{z-\alpha_i} &= \frac{g_t z^t + \cdots + g_1 z + g_0 - g_t\alpha_i^t - \cdots - g_1\alpha_i - g_0}{z-\alpha_i} \\ &= \frac{g_t(z^t-\alpha_i^t) + \cdots + g_1(z-\alpha_i)}{z-\alpha_i} \\ &= g_t(z^{t-1} + z^{t-2}\alpha_i + \cdots + z\alpha_i^{t-2} + \alpha_i^{t-1}) \\ &+ g_{t-1}(z^{t-2} + z^{t-3}\alpha_i + \cdots + z\alpha_i^{t-3} + \alpha_i^{t-2}) \\ &+ \cdots + g_2(z+\alpha_i) + g_1\end{aligned} \tag{5.2}$$

Preverimo, da je inverz v kolobarju $GF(q^m)[z]/g(z)$ enak $(z-\alpha_i)^{-1} = -\frac{g(z)-g(\alpha_i)}{z-\alpha_i}g(\alpha_i)^{-1}$. Potem mora veljati $(z-\alpha_i)\cdot(z-\alpha_i)^{-1} \equiv 1 \mod g(z)$. Preverimo, če je temu res tako.

$$\begin{aligned}(z-\alpha_i)\cdot(z-\alpha_i)^{-1} &= (z-\alpha_i)\cdot(-\tfrac{g(z)-g(\alpha_i)}{z-\alpha_i}g(\alpha_i)^{-1}) \\ &= (-g(z)+g(\alpha_i))\cdot g(\alpha_i)^{-1} \\ &= -g(z)\cdot g(\alpha_i)^{-1} + g(\alpha_i)\cdot g(\alpha_i)^{-1} \\ &\equiv -g(z)\cdot g(\alpha_i)^{-1} + 1 \\ &\equiv 1 \mod g(z).\end{aligned}$$

□

Oznaka 5.2. Inverz $(z-\alpha_i)^{-1}$ bomo včasih označili tudi z $\frac{1}{z-\alpha_i}$.

Za vsak vektor $c = (c_1, ..., c_n)$ nad $GF(q)$ lahko definiramo polinom

$$R_c(z) = \sum_{i=1}^{n} c_i(z-\alpha_i)^{-1}. \tag{5.3}$$

Definicija 5.3. *Goppov kod* $\Gamma(L, g(z))$, kjer sta L in $g(z)$ definirana kot prej, je množica vseh vektorjev $c \in GF(q)^n$, za katere velja

$$R_c(z) \equiv 0 \ (\mathrm{mod}\ g(z)). \tag{5.4}$$

Polinomu $g(z)$ rečemo tudi *generatorski polinom.* Če je $g(z)$ nerazcepen polinom, potem je $\Gamma(L, g(z))$ *nerazcepen Goppov kod.*

Neposredno iz definicije sledi naslednja trditev.

Trditev 5.4. *Goppov kod* $\Gamma(L, g(z))$ *je linearen.*

5.1 Parametri Goppovega koda

Ker je Goppov kod linearen, bomo zanj uporabljali standardne oznake: n je bločna dolžina, k dimenzija in d razmaknjenost koda. Prvi parameter predstavlja dolžino kodnih besed in je zato natanko določen z L. Za druga dva parametra lahko izpeljemo spodnje meje. V celotnem razdelku bo t predstavljal stopnjo generatorskega polinoma $g(z)$.

Izrek 5.5. *Za Goppov kod $\Gamma(L, g(z))$ nad $GF(q^m)$ dolžine n velja:*

- *dimenzija koda k zadošča*

$$k \geq n - mt, \tag{5.5}$$

- *razmaknjenost koda d zadošča*

$$d \geq t + 1. \tag{5.6}$$

Dokaz. Najprej bomo pokazali spodnjo mejo za dimenzijo koda. Označimo s $p_i(z)$ polinom $(z - \alpha_i)^{-1}$ iz trditve 5.1.

$$(z - \alpha_i)^{-1} \equiv p_i(z) = p_{i1} + p_{i2}z + ... + p_{it}z^{t-1} \pmod{g(z)}.$$

Enačbo (5.4) lahko zapišemo kot

$$\sum_{i=1}^{n} c_i p_i(z) \equiv 0 \pmod{g(z)}.$$

Če pogledamo enačbo za vsak koeficient pri z^j posebej, dobimo sistem enačb

$$\sum_{i=1}^{n} c_i p_{ij} = 0, \text{ za } 1 \leq j \leq t.$$

Torej je $\Gamma(L, g(z))$ definiran s t linearnimi enačbami nad $GF(q^m)$, kar lahko spremenimo na največ mt linearnih enačb nad $GF(q)$, tako da elemente iz obsega $GF(q^m)$ zapišemo v bazi $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$, kjer je α primitivni element obsega. Torej mora biti dimezija k vsaj $n - mt$.

Pokažimo še drugo neenakost. Ker je kod linearen, vemo, da je razmaknjenost enaka minimalni teži neničelne kodne besede. Naj bo c neničelna kodna beseda teže w, kjer je $c_i \neq 0$ za vse $i \in \{i_1, ..., i_w\}$. Potem lahko vsoto (5.3) preuredimo v

$$\sum_{i=1}^{n} \frac{c_i}{z - \alpha_i} = \frac{\sum_{j=1}^{w} c_{i_j} \prod_{1 \leq k \leq w, k \neq j} (z - \alpha_{i_k})}{\prod_{j=1}^{w} (z - \alpha_{i_j})}.$$

Ker imenovalec nima nobenega skupnega faktorja z $g(z)$, mora $g(z)$ deliti števec, saj mora veljati enačba (5.4). Števec ima stopnjo največ $w - 1$, iz česar sledi, da velja $w - 1 \geq t$ in je razmaknjenost d vsaj $w \geq t + 1$.

□

5.2 Parametri v primeru, da je generatorski polinom separabilen

Želimo, da je razmaknjenost koda d čim večja, saj lahko kod popravi r napak, če je $2r + 1 \leq d$. V posebnem primeru lahko spodnjo mejo za razmaknjenost Goppovega koda zvišamo, in sicer če vzamemo binarni Goppov kod $\Gamma(L, g(z))$, kjer je $g(z)$ separabilen polinom stopnje t nad $GF(2^m)$.

Definicija 5.6. Polinom $g(z)$ nad obsegom K je *separabilen*, če v nobeni razširitvi obsega K nima večkratnih ničel.

Ta pogoj je enostavno preveriti s pomočjo naslednje trditve.

Trditev 5.7. *Naj bo K obseg. Neničelen polinom $f(z) \in K[z]$ je separabilen, natanko tedaj, ko sta $f(z)$ in njegov odvod $f'(z) \in K[z]$ tuja.*

Izrek 5.8. *Naj bo $\Gamma(L, g(z))$ dvojiški Goppov kod, kjer je $g(z)$ separabilen polinom stopnje t. Potem je razmaknjenost tega koda enaka vsaj $2t + 1$.*

Dokaz. Vzemimo dvojiški Goppov kod $\Gamma(L, g(z))$, kjer je $g(z)$ separabilen polinom stopnje t. Naj bo $L = \{\alpha_1, \ldots, \alpha_n\}$ in $c = c_1 c_2 \ldots c_n$ neničelna

kodna beseda s težo w, ki ima enice na mestih $i_1, \ldots, i_w$. Vemo, da velja

$$\sum_{i=1}^{n} \frac{c_i}{z-\alpha_i} = \frac{\sum_{j=1}^{w} c_{i_j} \prod_{1\leq k\leq w, k\neq j}(z-\alpha_{i_k})}{\prod_{j=1}^{w}(z-\alpha_{i_j})}.$$

Označimo števec z

$$f(z) = \sum_{j=1}^{w} c_{i_j} \prod_{1\leq k\leq w, k\neq j} (z-\alpha_{i_k}).$$

Ker imamo dvojiški kod, so vsi c_{i_j} enaki 1 in dobimo

$$f(z) = \sum_{j=1}^{w} \prod_{1\leq k\leq w, k\neq j} (z-\alpha_{i_k}). \tag{5.7}$$

Kot v dokazu izreka 5.5 vidimo, da velja

$$\sum_{i=1}^{n} \frac{c_i}{z-\alpha_i} \equiv 0 \pmod{g(z)} \text{ natanko tedaj, ko } g(z) \text{ deli } f(z).$$

Opazimo, da je $f(z)$ ravno odvod polinoma $\prod_{j=1}^{w}(z-\alpha_{i_j})$.

Pri odvajanju polinomov v dvojiškem obsegu lahko dobimo samo sode eksponente, ker se lihi eksponenti izničijo zaradi sodega koeficienta. V primeru sode stopnje polinoma je najvišji možni eksponent $w-2$, v lihem primeru pa $w-1$. Torej je dobljeni odvod oblike:

$$f'(z) = f_0 + f_2 z^2 + ... + f_{2u} z^{2u}, \text{kjer je } 2u \leq w-1.$$

V $GF(2^m)$ je to ekvivalentno:

$$f'(z) = (k_0 + k_2 z + ... + k_{2u} z^u)^2 = (k(z))^2, \text{ kjer je } k_i^2 = f_i \text{ in } 2u \leq w-1.$$

Torej $g(z)$ deli $(k(z))^2$, kjer je $k(z)$ polinom stopnje u in $2u \leq w-1$. Ker $g(z)$ nima večkratnih nerazcepnih faktorjev, mora $g(z)$ deliti tudi $k(z)$. Torej $t \leq u$ in $w-1 \geq 2u \geq 2t$. Iz tega sledi, da je razmaknjenost d vsaj $w \geq 2t+1$. □

Posledica 5.9. *Naj bo $\Gamma(L, g(z))$ dvojiški Goppov kod, kjer je $g(z)$ separabilen polinom. Potem je $\Gamma(L, g(z)) = \Gamma(L, g^2(z))$.*

Opomba 5.10. Iz tega ne sledi $\Gamma(L, g(z)) = \Gamma(L, g^4(z))$, ker $g^2(z)$ ni separabilen.

V nadaljevanju bomo ob omembi Goppovega koda imeli v mislih dvojiški Goppov kod, kjer je generatorski polinom separabilen.

5.3 Nadzorna in generatorska matrika

Za dekodiranje koda je koristno, če poznamo nadzorno matriko. Po definiciji je c kodna beseda, če in samo če velja

$$\sum_{i=1}^{n} c_i p_i(z) \equiv 0 \pmod{g(z)},$$

kjer je $p_i(z) = p_{i1} + p_{i2}z + ... + p_{it}z^{t-1}$ ter velja

$$p_i(z) \cdot (z - \alpha_i) = 1 \pmod{g(z)}.$$

Če velja $cH^T = 0$ za vsak $c \in \mathcal{C}$ in je H polnega ranga, potem je nadzorna matrika koda $\mathcal{C}$ po izreku 4.18. Z uporabo te enakosti lahko dobimo matriko

$$\bar{H} = \begin{pmatrix} p_{11} & \cdots & p_{n1} \\ \vdots & \ddots & \vdots \\ p_{1t} & \cdots & p_{nt} \end{pmatrix}. \tag{5.8}$$

To še ni prava nadzorna matrika, saj je $\bar{H}$ matrika velikosti $t \times n$ nad $GF(q^m)$, glede na definicijo 4.17 pa je iskana nadzorna matrika $H \in GF(q)^{(n-k)\times n}$. Da določimo koeficiente p_{ij} preuredimo $p_i(z)$ v

$$p_i(z) \equiv (z - \alpha_i)^{-1} \equiv -\frac{g(z) - g(\alpha_i)}{z - \alpha_i} \cdot g(\alpha_i)^{-1}. \tag{5.9}$$

Sedaj definiramo $h_i := g(\alpha_i)^{-1}$ in upoštevamo enakost (5.2), da iz enačbe (5.9) dobimo:

$$p_i(z) = -(g_t(z^{t-1} + ... + \alpha_i^{t-2}) + ... + g_2(z + \alpha_i) + g_1) \cdot h_i. \tag{5.10}$$

Če sedaj nadomestimo $p_i(z) = p_{i1} + p_{i2}z + ... + p_{it}z^{t-1}$, dobimo naslednji sistem enačb za p_{ij} :

$$\begin{aligned} p_{i1} &= -(g_t\alpha_i^{t-1} + g_{t-1}\alpha_i^{t-2} + ... + g_2\alpha_i + g_1) \cdot h_i \\ p_{i2} &= -(g_t\alpha_i^{t-2} + g_{t-1}\alpha_i^{t-3} + ... + g_2) \cdot h_i \\ &\vdots \\ p_{i(t-1)} &= -(g_t\alpha_i + g_{t-1}) \cdot h_i \\ p_{it} &= -g_t \cdot h_i \end{aligned} \tag{5.11}$$

Če združimo (5.8) in (5.11), dobimo da je $\bar{H} = CXY$ za

$$C = \begin{pmatrix} -g_t & -g_{t-1} & -g_{t-2} & \cdots & -g_1 \\ 0 & -g_t & -g_{t-1} & \cdots & -g_2 \\ 0 & 0 & -g_t & \cdots & -g_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -g_t \end{pmatrix}, \tag{5.12}$$

$$X = \begin{pmatrix} \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \\ \alpha_1^{t-2} & \alpha_2^{t-2} & \cdots & \alpha_n^{t-2} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ 1 & 1 & \cdots & 1 \end{pmatrix}, \tag{5.13}$$

$$Y = \begin{pmatrix} h_1 & 0 & 0 & \cdots & 0 \\ 0 & h_2 & 0 & \cdots & 0 \\ 0 & 0 & h_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & h_n \end{pmatrix}. \tag{5.14}$$

Iz $\bar{H}$ lahko dobimo matriko velikosti $tm \times n$ nad $GF(q)$, tako da elemente predstavimo kot vektorje v bazi $\{1, \alpha, \cdots, \alpha^{m-1}\}$, kjer je α generator $GF(q^m)$. Dobljena matrika H ni nujno polnega ranga, vendar za vsak $c \in \mathcal{C}$ velja $cH^T = 0$. Če je rang H enak v, potem je dimenzija koda $\Gamma(L, g(z))$ enaka $n - v$.

Nadzorno matriko uporabljamo pri popravljanju napak, vendar potrebujemo tudi generatorsko matriko za kodiranje sporočil. Generatorsko matriko lahko dobimo iz nadzorne s pomočjo enačbe $GH^T = 0$ (izrek 4.18), ki nam poda sistem linearnih enačb, ki ga lahko rešimo na primer z Gaussovo metodo.

5.4 Primer binarnega Goppovega koda

Konstruirali bomo Goppov kod nad obsegom $GF(2^4)$, generiranim z nerazcepnim polinomom $f(x) = x^4 + x^3 + 1$. Torej je $GF(2^4)^* = \langle\alpha\rangle$ oziroma $GF(2^4) = \{0, \alpha, \alpha^2, \ldots, \alpha^{14}\}$, kjer je α primitvni element. Elemente $GF(2^4)^*$ lahko predstavimo kot potence α, če uporabimo enakost $\alpha^4 = \alpha^3 + 1$. Element 0 predstavimo z vektorjem $(0,0,0,0)^T$.

$$
\begin{array}{lcllll}
1 & = & 1 & & & = (1,0,0,0)^T, \\
\alpha & = & & \alpha & & = (0,1,0,0)^T, \\
\alpha^2 & = & & & \alpha^2 & = (0,0,1,0)^T, \\
\alpha^3 & = & & & & \alpha^3 = (0,0,0,1)^T, \\
\alpha^4 & = & 1 + & & & \alpha^3 = (1,0,0,1)^T, \\
\alpha^5 & = & 1 + & \alpha + & & \alpha^3 = (1,1,0,1)^T, \\
\alpha^6 & = & 1 + & \alpha + & \alpha^2 + & \alpha^3 = (1,1,1,1)^T, \\
\alpha^7 & = & 1 + & \alpha + & \alpha^2 & = (1,1,1,0)^T, \\
\alpha^8 & = & & \alpha + & \alpha^2 + & \alpha^3 = (0,1,1,1)^T, \\
\alpha^9 & = & 1 + & & \alpha^2 & = (1,0,1,0)^T, \\
\alpha^{10} & = & & \alpha + & & \alpha^3 = (0,1,0,1)^T, \\
\alpha^{11} & = & 1 + & & \alpha^2 + & \alpha^3 = (1,0,1,1)^T, \\
\alpha^{12} & = & 1 + & \alpha & & = (1,1,0,0)^T, \\
\alpha^{13} & = & & \alpha + & \alpha^2 & = (0,1,1,0)^T, \\
\alpha^{14} & = & & & \alpha^2 + & \alpha^3 = (0,0,1,1)^T.
\end{array}
\tag{5.15}
$$

Poglejmo si zdaj Goppov kod $\Gamma(L, g(z))$ nad $GF(2^4)$ definiran z

$$g(z) = (z+\alpha)(z+\alpha^{14}) = z^2 + \alpha^8 z + 1 \text{ in}$$
$$L = \{\alpha^i \mid 2 \leq i \leq 13\}.$$

Za ta kod je $q = 2, m = 4, n = 12$ in $t = 2$. Po izreku 5.5 mora biti dimenzija $k \geq 12 - 4 \cdot 2 = 4$. Ker je $g(z)$ separabilen polinom nad dvojiškim obsegom, lahko uporabimo izrek 5.8, da dobimo $d \geq 2 \cdot 2 + 1 = 5$. Torej imamo $[12, \geq 4, \geq 5]$ Goppov kod.

Da najdemo nadzorno matriko, lahko uporabimo obliko $\bar{H} = CXY$, kjer so matrike C, X, Y iz (5.12)-(5.14). Pri tem je $g_1 = \alpha^8$, $g_2 = 1$ in $\alpha_1 = \alpha^2, \alpha_2 = \alpha^3, \ldots, \alpha_{12} = \alpha^{13}$. Koeficiente $h_i = g(\alpha_i)^{-1}$ za $1 \leq i \leq 12$ izračunamo takole:

$$\begin{aligned} h_1 &= g(\alpha^2)^{-1} = ((\alpha^2))^2 + \alpha^8 \cdot \alpha^2 + 1)^{-1} \\ &= ((1,0,0,1)^T + (0,1,0,1)^T + (1,0,0,0)^T)^{-1} \\ &= ((0,1,0,0)^T)^{-1} = \alpha^{-1} = \alpha^{14}, \end{aligned}$$

$$\begin{aligned} h_2 &= g(\alpha^3)^{-1} = (\alpha^6 + \alpha^{11} + 1)^{-1} \\ &= ((1,1,1,1)^T + (1,0,1,1)^T + (1,0,0,0)^T)^{-1} \\ &= ((1,1,0,0)^T)^{-1} = \alpha^{-12} = \alpha^3. \end{aligned}$$

Ostale h_i izračunamo podobno. Zdaj je potrebno samo vstaviti izračunane h_i v enačbo $\bar{H} = CXY$, za katero velja:

$$\bar{H} = \begin{pmatrix} -(\alpha^2 + \alpha^8)h_1 & \cdots & -(\alpha^{13} + \alpha^8)h_{12} \\ -h_1 & \cdots & -h_{12} \end{pmatrix}.$$

Negativne predznake izpustimo, ker imamo dvojiški kod. Dobimo neke vrste nadzorno matriko:

$$\bar{H} = \begin{pmatrix} \alpha^9 & \alpha & \alpha^8 & \alpha^{13} & \alpha^7 & \alpha^5 & 0 & \alpha^9 & \alpha & \alpha^6 & \alpha^5 & \alpha^6 \\ \alpha^{14} & \alpha^3 & \alpha & \alpha^4 & \alpha^7 & \alpha & 1 & \alpha^4 & \alpha^{14} & \alpha^9 & \alpha^9 & \alpha^3 \end{pmatrix}.$$

Preverimo, če za prvi stolpec velja $(z-\alpha^2)^{-1} \equiv \alpha^9 + \alpha^{14}z \pmod{g(z)}$, kjer uporabimo enakosti (5.15):

$$\begin{aligned}(z-\alpha^2)\cdot(\alpha^9+\alpha^{14}z) &= \alpha^{14}z^2+(\alpha^9+\alpha)z+\alpha^{11}\\ &\equiv \alpha^{14}z^2+\alpha^7 z+\alpha^{11}+\alpha^{14}(z^2+\alpha^8 z+1)\\ &= \alpha^7 z+\alpha^{11}+\alpha^7 z+\alpha^{14}\\ &= \alpha^{11}+\alpha^{14}\\ &= 1.\end{aligned}$$

Ostale stolpce lahko preverimo na podoben način. Matriko $\bar{H}$ z uporabo (5.15) zapišemo v binarni obliki.

$$H = \left(\begin{array}{cccccccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1\\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1\\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1\\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1\\
\hdashline
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0\\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0\\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1
\end{array}\right)$$

Matrika H ni prava nadzorna matrika, saj ni težko preveriti da ni polnega ranga. Matrika H ima rang 8, iz česar sledi, ga je dimenzija koda enaka 4 in imamo Goppov kod s parametri $[12, 4, \geq 5]$. Generatorsko matriko lahko zdaj dobimo iz nadzorne matrike tako, da uporabimo zvezo $GH^T = 0$. Dobimo sistem 12 linearnih enačb, ki ga rešimo z Gaussovo eliminacijo. Ena izmed možnih generatorskih matrik je:

$$G = \begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0\\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0\\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1
\end{pmatrix}.$$

Ni nujno, da za $g(z)$ izberemo separabilen polinom. Prednost drugačne izbire je, da je lahko računanje po modulu $g(z)$ lažje, saj lahko izberemo na

primer polinom kot z^t. Slabost pa je, da bo spodnja meja razmaknjenosti le $t+1$, namesto $2t+1$, kar pomeni, da bo lahko izbrani kod popravil manjše število napak.

5.5 Kodiranje in dekodiranje Goppovega koda

Naj bo $\Gamma(L, g(z))$ Goppov kod definiran s polinomom $g(z)$ stopnje t nad $GF(q^m)$ in množico $L \subseteq GF(q^m)$ velikosti n. Naj bo dimenzija koda enaka k in razmaknjenost d. Potem sporočilo s kodiramo tako, da ga razdelimo v bloke velikosti k in vsak blok pomnožimo z generatorsko matriko G:

$$(s_1, \ldots, s_k) \cdot G = (c_1, \ldots, c_n).$$

Dekodiranje je precej bolj zapleteno. V nadaljevanju bomo orisali algoritma za dekodiranje splošnega Goppovega koda in dvojiškega nerazcepnega koda. Oba je prvi opisal Patterson v [33] in uporabljata znan Berlekampov algoritem za dekodiranje kodov BCH [2], ki ga danes poznamo kot Berlekamp-Masseyjev algoritem. Pattersonov algoritem lahko uporabimo tudi za dekodiranje kodov BCH in Reed-Solomonovih kodov.

5.5.1 Popravljanje napak

Naj bo c poslana kodna beseda in y prejeta beseda, ki vsebuje r napak, kjer velja $2r + 1 \leq d$, v primeru, da je $d = t + 1$. Potem je

$$(y_1, \ldots, y_n) = (c_1, \ldots, c_n) + (e_1, \ldots, e_n),$$

kjer $e \neq 0$ na natanko r mestih. Če želimo popraviti besedo in najti pravilno kodno besedo c, moramo najti vektor napake e. Torej moramo odkriti:

- množico mest B, kjer se nahajajo napake, $B = \{i \mid e_i \neq 0\}$,
- pripadajoče vrednosti napak e_i za $i \in B$.

Definicija 5.11. *Polinom za iskanje napak* $\sigma(z)$ je

$$\sigma(z) := \prod_{i \in B}(z - \alpha_i). \tag{5.16}$$

Polinom za ocenitev napake $\omega(z)$ je

$$\omega(z) := \sum_{i \in B} c_i \prod_{j \in B, j \neq i} (z - \alpha_j). \tag{5.17}$$

Iz definicije je jasno, da položaje napak dobimo neposredno iz ničel $\sigma(z)$, torej je $B = \{i \mid \alpha_i \text{ je ničla } \sigma(z)\}$. Kasneje bomo videli še, kako s pomočjo polinoma $\omega(z)$ izračunamo še vrednosti napak.

Definicija 5.12. *Sindrom prejete besede* y je:

$$s(z) := \sum_{i=1}^{n} \frac{y_i}{z - \alpha_i}.$$

Opazimo, da je

$$\begin{aligned} s(z) &= \sum_{i=1}^{n} \frac{c_i + e_i}{z - \alpha_i} = \\ &= \sum_{i=1}^{n} \frac{c_i}{z - \alpha_i} + \sum_{i \in B} \frac{e_i}{z - \alpha_i} = \\ &\equiv \sum_{i \in B} \frac{e_i}{z - \alpha_i} \pmod{g(z)}. \end{aligned} \tag{5.18}$$

Izrek 5.13. *Naj bo* e *napaka s težo* r*, pri čemer je* $r \leq \lfloor \frac{t}{2} \rfloor$*. Naj bodo* $\sigma(z), \omega(z)$ *in* $s(z)$ *definirani kot v definicijah 5.11 in 5.12. Potem veljajo naslednje lastnosti:*

1. $\deg(\sigma(z)) = r$,
2. $\deg(\omega(z)) \leq r - 1$,
3. $\gcd(\sigma(z), \omega(z)) = 1$,
4. $e_k = \frac{\omega(\alpha_k)}{\sigma'(\alpha_k)}, k \in B$,

5. $\sigma(z)s(z) \equiv \omega(z) \pmod{g(z)}$.

Dokaz. Prve tri lastnosti sledijo neposredno iz definicije 5.11. Da dokažemo četrto lastnost, najprej izračunajmo $\sigma'(z)$.

$$\sigma'(z) = \sum_{i \in B} \prod_{j \in B, j \neq i} (z - \alpha_j).$$

Torej za $k \in B$ velja

$$\frac{\omega(\alpha_k)}{\sigma'(\alpha_k)} = \frac{\sum_{i \in B} e_i \prod_{j \in B, j \neq i} (\alpha_k - \alpha_j)}{\sum_{i \in B} \prod_{j \in B, j \neq i} (\alpha_k - \alpha_j)} = e_k.$$

Da dokažemo peto lastnost, uporabimo definiciji 5.11 in 5.12:

$$\begin{aligned} \sigma(z)s(z) &\equiv \prod_{i \in B}(z - \alpha_i) \cdot \sum_{i \in B} \frac{e_i}{z - \alpha_i} \\ &= \sum_{i \in B} e_i \prod_{j \in B, j \neq i} (z - \alpha_j) \\ &= \omega(z). \end{aligned}$$

□

Enačbi iz točke 5 v izreku 5.13 pravimo *ključna enačba.* Da bomo lahko popravili napake v kodni besedi, moramo rešiti prav ključno enačbo:

$$\sigma(z)s(z) \equiv \omega(z) \pmod{g(z)}. \tag{5.19}$$

Ker je $g(z)$ poznan in sindrom $s(z)$ lahko izračunamo, je potrebno rešiti sistem t enačb z $2r$ neznankami $\sigma_0, \sigma_1, \ldots, \sigma_{r-1}, \omega_0, \omega_1, \ldots, \omega_{r-1}$, kjer $\sigma(z) = \sigma_0 + \sigma_1 z + \cdots + \sigma_{r-1} z^{r-1}$ in $\omega(z) = \omega_0 + \omega_1 z + \cdots + \omega_{r-1} z^{r-1}$. Ker je $2r \leq t$, vemo, da obstaja enolična rešitev.

Zdaj lahko opišemo algoritem za popravljanje $r \leq \lfloor \frac{t}{2} \rfloor$ napak v Goppovem kodu.

Algoritem 5.1 (Patterson [33])**.**
Naj bo $y = (y_1, \ldots, y_n)$ *prejeta kodna beseda, ki vsebuje* r *napak, kjer je* $2r \leq t$.

1. Izračunamo sindrom

$$s(z) = \sum_{i=1}^{n} \frac{y_i}{z - \alpha_i}.$$

2. *Rešimo ključno enačbo*

$$\sigma(z)s(z) \equiv \omega(z) \ (mod\ g(z)),$$

tako da upoštevamo

$$\sigma(z) = \sigma_0 + \sigma_1 z + \cdots + \sigma_{r-1} z^{r-1},$$

$$\omega(z) = \omega_0 + \omega_1 z + \cdots + \omega_{r-1} z^{r-1},$$

in rešimo pomožni sistem s t enačbami in 2r neznankami. Če imamo dvojiški kod, lahko vzamemo $\omega(z) = \sigma`(z)$.

3. *Določimo množico pozicij napak* $B = \{i \mid \sigma(\alpha_i) = 0\}$.

4. *Izračunamo vrednosti napak*

$$e_i = \begin{cases} \frac{\omega(\alpha_i)}{\sigma'(\alpha_i)} & ;\ \check{c}e\ je\ i \in B \\ 0 & ;\ sicer \end{cases}.$$

5. *Vektor napak je* $e = (e_1, ..., e_n)$.

6. *Poslana kodna beseda je* $c = y - e$.

Če imamo dvojiški kod, za odvod polinoma za iskanje napak velja

$$\sigma'(z) = \sum_{i \in B} \prod_{j \in B, j \neq i} (z - \alpha_j).$$

Prav tako so v (5.17) vsi c_i enaki 1, torej je $\omega(z) = \sigma'(z)$.

5.5.2 Popravljanje napak v primeru separabilnega generatorskega polinoma

Naj bo $\Gamma(L, g(z))$ dvojiški Goppov kod z generatorskim polinomom $g(z)$ nad $GF(2^m)$, kjer je $g(z)$ separabilen polinom stopnje t. Potem ima glede na izrek 5.8 kod razmaknjenost vsaj $2t+1$. Torej lahko popravimo vsaj t napak in ne

moremo biti zadovoljni z algoritmom, ki popravi samo $\lfloor \frac{t}{2} \rfloor$ napak. Algoritem 5.1 lahko prilagodimo tako, da v tem primeru popravi t napak.

Iz posledice 5.9 vemo, da $\Gamma(L, g(z)) = \Gamma(L, g^2(z))$, in ker je $g^2(z)$ stopnje $2t$, lahko algoritem 5.1 popravi $\lfloor \frac{2t}{2} \rfloor = t$ napak. Slaba stran tega pristopa je, da moramo določiti nadzorno matriko H Goppovega koda $\Gamma(L, g^2(z))$, da lahko izračunamo sindrom prejetega vektorja, za kar potrebujemo nekaj dodatnega dela.

Obstaja še drug pristop, ki ne uporablja $g^2(z)$, vendar deluje samo v primeru, ko je $\gcd(s(z), g(z)) = 1$. Temu pa v splošnem ni zadoščeno za poljuben separabilen polinom $g(z)$. Če vzamemo nerazcepen $g(z)$, je to vedno res. V nadaljevanju bomo izpeljali algoritem za popravljanje napak, če imamo nerazcepen generatorski polinom.

Ker imamo dvojiški kod, lahko ključno enačbo (5.19) zapišemo kot

$$\sigma(z)s(z) \equiv \sigma'(z) \pmod{g(z)}. \tag{5.20}$$

Polinom $\sigma(z)$ stopnje največ t lahko razdelimo glede na sode in lihe potence in dobimo

$$\sigma(z) = a^2(z) + b^2(z)z,$$

kjer je $\deg(a(z)) \leq \frac{t}{2}$ in $\deg(b(z)) \leq \frac{t-1}{2}$. To velja, ker je

$$\begin{aligned} a^2(z) + b^2(z)z &= (a_0 + a_1 z + \cdots + a_k z^k)^2 + (b_0 + b_1 z + \cdots + b_l z^l)^2 \cdot z \\ &= a_0^2 + b_0^2 z + a_1^2 z^2 + b_1^2 z^3 + \cdots + a_k^2 z^{2k} + b_l^2 z^{2l+1} \end{aligned}$$

za $2k \leq t$ in $2l + 1 \leq t$. Potem je odvod enak

$$\sigma'(z) = 2a(z)a'(z) + 2b(z)b'(z)z + b^2(z) = b^2(z).$$

Če uporabimo enačbo (5.20), dobimo:

$$(a^2(z) + b^2(z)z) \cdot s(z) \equiv b^2(z) \pmod{g(z)}. \tag{5.21}$$

Ker je $g(z)$ nerazcepen, je $s(z)$ tuj $g(z)$. Torej nam razširjeni Evklidov algoritem da tak polinom $h(z)$, da velja

$$h(z)s(z) \equiv 1 \pmod{g(z)}. \tag{5.22}$$

Če združimo (5.21) in (5.22) dobimo

$$a^2(z) + b^2(z)z \equiv b^2(z)h(z) \pmod{g(z)}$$

ali ekvivalentno

$$b^2(z)(z + h(z)) \equiv a^2(z) \pmod{g(z)}. \tag{5.23}$$

Če je $h(z) = z$, iz (5.23) sledi, da je $a(z) = 0$ in torej $\sigma(z) = zb^2(z)$. Ker $\sigma(z)$ nima večkratnih ničel, je $b^2(z) = 1$ in $\sigma(z) = z$ rešitev. Sicer obstaja enolični polinom $d(z) \pmod{g(z)}$, tak da je

$$d^2(z) = z + h(z) \pmod{g(z)}.$$

To lahko storimo, saj je kvadriranje polinomov po modulu $g(z)$ linearna preslikava in je karakteristika $GF(2^m)$ enaka 2. Iz (5.23) vemo, da je

$$d^2(z) \cdot b^2(z) \equiv a^2(z) \pmod{g(z)},$$

kar je ekvivalentno

$$d(z) \cdot b(z) \equiv a(z) \pmod{g(z)},$$

ker smo v dvojiškem obsegu. Gornji razmislek nam poda algoritem za iskanje $\sigma(z)$ stopnje $\leq \deg(g(z)) = t$, ki lahko popravi največ t napak.

Opišimo algoritem, ki popravi $r \leq t$ napak za nerazcepen $g(z)$ nad $GF(2^m)$.

Algoritem 5.2 (Patterson[33])**.**
Naj bo $y = (y_1, \ldots, y_n)$ prejeta kodna beseda, ki vsebuje r napak, kjer je $r \leq t$.

1. *Izračunamo sindrom*
$$s(z) = \sum_{i=1}^{n} \frac{y_i}{z - \alpha_i}.$$

2. *Določimo $\sigma(z)$ v naslednjih štirih korakih:*

   - *Uporabimo razširjen Evklidov algoritem (algoritem 3.2), da najdemo $h(z)$, tak da*
   $$s(z)h(z) \equiv 1 \ (mod\ g(z)).$$
   Če je $h(z) = z$, končamo in je rešitev enaka $\sigma(z) = z$.

- *Izračunamo* $d(z)$ *tako da velja* $d^2(z) \equiv h(z) + z \pmod{g(z)}$.
- *Najdemo* $a(z)$ *in* $b(z)$ *najmanjše možne stopnje, da zadoščata*

$$d(z)b(z) \equiv a(z) \pmod{g(z)}.$$

- *Izračunamo* $\sigma(z) = a^2(z) + b^2(z)z$.

3. *Določimo množico položajev napak* $B = \{i \mid \sigma(\alpha_i) = 0\}$.
4. *Vektor napak* $e = (e_1, \ldots, e_n)$ *je definiran z* $e_i = 1$ *za* $i \in B$ *in* $e_i = 0$ *sicer.*
5. *Poslana kodna beseda je* $c = y - e$.

Ko smo popravili morebitne napake v kodni besedi, lahko dobimo poslano sporočilo, če se spomnimo na to, kako kodiramo:

$$(s_1, \ldots, s_k) \cdot G = (c_1, \ldots, c_n),$$

kar je ekvivalentno

$$G^T \cdot \begin{pmatrix} s_1 \\ \vdots \\ s_k \end{pmatrix} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}.$$

Da najdemo sporočilo s, rešimo gornji sistem z uporabo Gaussove metode. Najprej zapišemo razširjeno matriko sistema

$$\left(\begin{array}{c|c} G^T & \begin{matrix} c_1 \\ \vdots \\ c_n \end{matrix} \end{array}\right).$$

Z elementarnimi transformacijami vrstic poskušamo dobiti v zgornjem levem kotu identiteto velikosti $k \times k$. Potem prvih k bitov v zadnjem stolpcu predstavlja našo rešitev. Dobimo

$$\left(\begin{array}{c|c} G^T & \begin{matrix} c_1 \\ \vdots \\ c_n \end{matrix} \end{array}\right) \sim \cdots \sim \left(\begin{array}{c|c} I_k & \begin{matrix} s_1 \\ \vdots \\ s_k \end{matrix} \\ \hline \multicolumn{2}{c}{P} \end{array}\right), \tag{5.24}$$

kjer je I_k identiteta velikosti $k \times k$ in P $(n-k) \times (k+1)$ matrika.

5.5.3 Primer kodiranja in dekodiranja Goppovega koda

Naj bo $\Gamma(L, g(z))$ Goppov kod iz razdelka 5.4. Recimo, da želimo poslati sporočilo $(1,1,1,1)$. Najprej ga pomnožimo z generatorsko matriko G, da ga zakodiramo.

$$c = (1,1,1,1) \cdot \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$
$$= (0,0,1,0,1,0,1,0,1,1,1,1).$$

Parametri koda so $[12, 4, \geq 5]$, torej lahko naredimo $r \leq 2$ napak. Recimo, da se pri prenosu pokvari tretje in zadnje mesto. Prejeta beseda bo

$$y = (0,0,0,0,1,0,1,0,1,1,1,0).$$

Ker je $g(z)$ separabilen polinom, lahko prejemnik uporabi dejstvo, da je $\Gamma(L, g(z)) = \Gamma(L, g^2(z))$, v kombinaciji z algoritmom 5.1, da popravi napake. Najprej moramo najti nadzorno matriko koda $\Gamma(L, g^2(z))$. Postopek je podoben kot pri primeru v razdelku 5.4. Uporabimo naslednji polinom

$$\hat{g}(t) = g^2(t) = (z^2 + \alpha^8 z + 1)^2 = z^4 + \alpha z^2 + 1$$

kot generatorski polinom. Torej so $\hat{g_4} = 1, \hat{g_3} = 0, \hat{g_2} = \alpha, \hat{g_1} = 0, \hat{g_0} = 1$. Nove koeficiente $\hat{h_i}$ lahko dobimo tako, da kvadriramo stare koeficiente h_i, ker velja

$$\hat{h_i} = (g^2(\alpha_i))^{-1} = (g(\alpha_i)^{-1})^2 = h_i^2.$$

Iz enačb (5.12)-(5.14) vemo, da je nadzorna matrika enaka

$$\hat{H} = \begin{pmatrix} ((\alpha^2)^3 + \alpha\alpha^2)h_1^2 & \cdots & ((\alpha^{13})^3 + \alpha\alpha^{13})h_{12}^2 \\ ((\alpha^2)^2 + \alpha)h_1^2 & \cdots & ((\alpha^{13})^2 + \alpha)h_{12}^2 \\ \alpha^2 h_1^2 & \cdots & \alpha^{13} h_{12}^2 \\ h_1^2 & \cdots & h_{12}^2 \end{pmatrix}.$$

Po daljšem računanju dobimo matriko

$$\hat{H} = \begin{pmatrix} \alpha^5 & \alpha^5 & \alpha^5 & \alpha & \alpha^5 & \alpha^2 & 0 & \alpha^{12} & \alpha^{12} & \alpha^8 & \alpha^7 & \alpha^{10} \\ \alpha^3 & \alpha^2 & \alpha & \alpha^{11} & \alpha^{14} & \alpha^{10} & 0 & \alpha^3 & \alpha^2 & \alpha^{12} & \alpha^{10} & \alpha^{12} \\ 1 & \alpha^9 & \alpha^6 & \alpha^{13} & \alpha^5 & \alpha^9 & \alpha^8 & \alpha^2 & \alpha^8 & \alpha^{14} & 1 & \alpha^4 \\ \alpha^{13} & \alpha^6 & \alpha^2 & \alpha^8 & \alpha^{14} & \alpha^2 & 1 & \alpha^8 & \alpha^{13} & \alpha^3 & \alpha^3 & \alpha^6 \end{pmatrix}.$$

Preverimo, če velja $\frac{1}{z-\alpha^2} \equiv \alpha^5 + \alpha^3 z + z^2 + \alpha^{13} z^3 \pmod{z^4 + \alpha z^2 + 1}$ za prvi stolpec.

$$\begin{aligned} (z-\alpha^2)(\alpha^5 + \alpha^3 z + z^2 + \alpha^{13} z^3) &= \alpha^{13} z^4 + (\alpha^2 + \alpha^3) z^2 + \alpha^7 \\ &\equiv \alpha^{13} z^4 + \alpha^{14} z^2 + \alpha^7 + \alpha^{13}(z^4 + \alpha z^2 + 1) \\ &= \alpha^7 + \alpha^{13} \\ &= 1. \end{aligned}$$

Podobno lahko preverimo še ostale stolpce. Zdaj, ko imamo nadzorno matriko, lahko popravimo prejeto besedo $y = (0,0,0,0,1,0,1,0,1,1,1,0)$ z algoritmom 5.1.

1. Izračunamo sindrom.

$$\begin{aligned} s(z) &= \sum_{i=1}^{12} \frac{y_i}{z - \alpha_i} \\ &= \frac{1}{z-\alpha^6} + \frac{1}{z-\alpha^8} + \frac{1}{z-\alpha^{10}} + \frac{1}{z-\alpha^{11}} + \frac{1}{z-\alpha^{12}} \\ &\equiv (\alpha^5 + 0 + \alpha^{12} + \alpha^8 + \alpha^7) + \\ &\quad (\alpha^{14} + 0 + \alpha^2 + \alpha^{12} + \alpha^{10}) z + \\ &\quad (\alpha^5 + \alpha^8 + \alpha^8 + \alpha^{14} + 1) z^2 + \\ &\quad (\alpha^{14} + 1 + \alpha^{13} + \alpha^3 + \alpha^3) z^3 \\ &= \alpha^5 z^3 + \alpha^{13} z^2 + z + 1. \end{aligned}$$

2. Izračunamo $\sigma(z)s(z) \pmod{z^4 + \alpha z^2 + 1}$.

$$\begin{aligned}
\sigma(z)s(z) &= (z^2 + \sigma_1 z + \sigma_0)(\alpha^5 z^3 + \alpha^{13} z^2 + z + 1) \\
&= \alpha^5 z^5 + (\alpha^{13} + \sigma_1\alpha^5)z^4 + (1 + \sigma_1\alpha^{13} + \sigma_0\alpha^5)z^3 + \\
&\quad (1 + \sigma_1 + \sigma_0\alpha^{13})z^2 + (\sigma_1 + \sigma_0)z + \sigma_0 \\
&\equiv (\alpha^{13} + \sigma_1\alpha^5)z^4 + (\alpha^8 + \sigma_1\alpha^{13} + \sigma_0\alpha^5)z^3 + \\
&\quad (1 + \sigma_1 + \sigma_0\alpha^{13})z^2 + (\alpha^5 + \sigma_1 + \sigma_0)z + \sigma_0 \\
&\equiv (\alpha^8 + \sigma_1\alpha^{13} + \sigma_0\alpha^5)z^3 + (\alpha^{11} + \sigma_1\alpha^8 + \sigma_0\alpha^{13})z^2 + \\
&\quad (\alpha^5 + \sigma_1 + \sigma_0)z + (\alpha^{13} + \sigma_1\alpha^5 + \sigma_0) \pmod{z^4 + \alpha z^2 + 1}.
\end{aligned}$$

V tretjem koraku smo dodali $\alpha^5 z \cdot \hat{g}(z)$ in v četrtem $(\alpha^{13} + \sigma_1\alpha^5) \cdot \hat{g}(z)$. Ker je kod dvojiški, imamo enačbo:

$$\sigma(z)s(z) \equiv \sigma'(z) \pmod{z^4 + \alpha z + 1},$$

torej moramo rešiti naslednji sistem enačb:

$$\begin{aligned}
\sigma_1 &= \alpha^{13} + \sigma_1\alpha^5 + \sigma_0 \\
0 &= \alpha^5 + \sigma_1 + \sigma_0 \\
0 &= \alpha^{11} + \sigma_1\alpha^8 + \sigma_0\alpha^{13} \\
0 &= \alpha^8 + \sigma_1\alpha^{13} + \sigma_0\alpha^5
\end{aligned}$$

Izračunamo, da je $\sigma_0 = \alpha^2$ in $\sigma_1 = \alpha^6$, torej je

$$\sigma(z) = z^2 + \alpha^6 z + \alpha^2 = (z + \alpha^4)(z + \alpha^{13}).$$

3. Ker je $\alpha_3 = \alpha^4$ in $\alpha_{12} = \alpha^{13}$, je množica položajev napak enaka

$$B = \{i \mid \sigma(\alpha_i) = 0\} = \{3, 12\}.$$

4. Vektor napak je $e = (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)$.

5. Pripadajoča kodna beseda je

$$y - e = (0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1).$$

Iz kodne besede, lahko dobimo prvotno sporočilo s, če uporabimo enačbe (5.24). Zamenjamo prve štiri vrstice z zadnjimi štirimi in dobimo rešitev v petem stolpcu.

$$\left[\begin{array}{c|c} G^T & c^T \end{array}\right] = \left(\begin{array}{cccc|c} 1&0&0&1&0\\ 1&0&1&0&0\\ 0&0&0&1&1\\ 0&1&0&1&0\\ 0&1&1&1&1\\ 0&1&1&0&0\\ 1&0&1&1&1\\ 1&1&1&1&0\\ 1&0&0&0&1\\ 0&1&0&0&1\\ 0&0&1&0&1\\ 0&0&0&1&1 \end{array}\right) \sim \left(\begin{array}{cccc|c} 1&0&0&0&1\\ 0&1&0&0&1\\ 0&0&1&0&1\\ 0&0&0&1&1\\ \hline 0&1&1&1&1\\ 0&1&1&0&0\\ 1&0&1&1&1\\ 1&1&1&1&0\\ 1&0&0&1&0\\ 1&0&1&0&0\\ 0&0&0&1&1\\ 0&1&0&1&0 \end{array}\right).$$

Prenesena informacija je bila $s = (1, 1, 1, 1)$.

Poglavje 6

McElieceov kriptosistem

S pojmom *kriptografija na osnovi kodiranja* (angl. *code-based cryptography*) zajemamo vse kriptosisteme, ki temeljijo na nekem kodu za popravljanje napak $\mathcal{C}$. Šifriranje izvedemo tako, da sporočilu dodamo napako. Kriptosistemi, ki spadajo v to vejo kriptografije, so večinoma že precej stari, ampak dandanes niso v uporabi [32]. Razlog za to je njihova nepraktičnost, saj so javni ključi zelo veliki. Ob obstoječem kriptosistemu s krajšim ključem, kot je RSA, se preprosto niso razširili.

Prvi takšen kriptosistem je bil sistem z javnim ključem, ki ga je leta 1978 predlagal Robert J. McEliece [27], kjer je zasebni ključ generatorska matrika za naključni dvojiški nerazcepni Goppov kod in je javni ključ dobljen iz zasebnega z naključno permutacijo vrstic in stolpcev. Kriptogram je kodna beseda, ki so ji bile dodane napake in samo lastnik zasebnega ključa jih lahko odstrani, saj za Goppov kod obstaja učinkovit algoritem za dekodiranje. Po štirih desetletjih ni znan noben napad, ki bi predstavljal resno grožnjo. Poleg tega, da je varen za klasične računalnike, je ogromna prednost tega kriptosistema, da bi morebitna izgradnja kvantnega računalnika na varnost kriptosistema imela minimalen učinek. Obstoječi kvantni algoritmi namreč ne izboljšajo stopnje kompleksnosti dekodiranja. Podobne ideje so bile uporabljene za konstrukcijo drugih sistemov, med katerimi so:

- Niederreiterjeva šifrirna shema [31], za katero se je izkazalo, da je var-

nostno ekvivalentna McElieceovemu predlogu,

- CFS shema digitalnega podpisa [11],
- identifikacijske sheme [42, 47], generatorji naključnih števil [17, 18] in kriptografske zgoščevalne funkcije [1].

Izvajanje kriptografije na podlagi kodov predstavlja nek kompromis med varnostjo in učinkovitostjo. Ta problematika je bila čez leta dobro preučena za McElieceov kriptosistem. Kljub temu ta kriptosistem ni v širši uporabi, na kar delno vpliva velikost javnega ključa (od 100 kB pa do nekaj MB), deloma pa pomanjkanje promocije, ker alternativna rešitev za zdaj ni nujno potrebna. McElieceov kriptosistem pa ima tudi prednosti, in sicer je zelo hiter, saj imata tako šifriranje kot dešifriranje nizko kompleksnost.

McElieceov kriptosistem ostaja nerazbit v svoji originalni različici, z leti je bilo treba samo prilagoditi parametre za zagotovitev varnosti. Za kod, na katerem temelji McElieceov kriptosistem se je uveljavil Goppov kod, vendar je bilo v originalni različici predvideno, da je lahko zasebni ključ iz katerega koli podrazreda razreda alternantnih kod (npr. kod BCH, Goppov kod, Strivastavov kod). Izbira drugega alternantnega koda pa lahko ne bi zagotovila želene varnosti, saj jih enostavno ni dovolj.

6.1 Opis kriptosistema

McElieceov kriptosistem je kriptosistem z javnim ključem, ki temelji na zahtevnosti dekodiranja pri splošnih linearnih kodih. Kod, ki se uporablja v originalni različici je dvojiški nerazcepni Goppov kod. Uporabnik najprej izbere želene vrednosti n, m in t, kjer je $mt < n$. S tem definira tudi $k = n - mt$. V naslednjem koraku naključno izbere n različnih elementov iz $GF(2^m)$, ki tvorijo množico $L = \{\alpha_1, \ldots, \alpha_n\}$. Potem naključno izbere nerazcepen polinom $g(z)$ stopnje t nad $GF(2^m)$ in izračuna $k \times n$ generatorsko matriko G za kod $\mathcal{G} = \Gamma(g(z), L)$. Po generiranju G mora uporabnik premešati G,

tako da izbere naključno nesingularno matriko S velikosti $k \times k$ in naključno permutacijsko matriko P velikosti $n \times n$.

Potem izračuna $G' = SGP$, ki generira linearen kod $\mathcal{G}'$ z isto razmaknjenostjo kot kod $\mathcal{G}$. Matriki G' rečemo *javna generatorska matrika*. Uporabnik objavi G' in t, ostale informacije pa zadrži zase.

Če želi nekdo poslati sporočilo uporabniku, najprej poišče njegov javni ključ. Množica besedil so nizi dolžine k. Sporočilo mora torej razdeliti na zaporedje besedil in za vsako besedilo s izbere naključno napako e dolžine n s težo največ t. Niz s šifrira kot $y = sG' + e$ in ga pošlje. Za šifriranje besedila potrebujemo približno $\frac{k}{2} \cdot n + t$ binarnih operacij in čas, ki je potreben za generiranje vektorja napak.

Prejemnik lahko izračuna sporočilo s pomočjo svoje skrivne matrike P. Najprej izračuna $y' = yP^{-1}$, za katerega velja

$$y' = yP^{-1} = sG'P^{-1} + eP^{-1} = sSGPP^{-1} + e' = (sS)G + e',$$

kjer je $t(e') \leq t$.

Za dešifriranje prejetega besedila prejemnik naprej z uporabo algoritma za dekodiranje Goppovega koda 5.2 dekodira y' v kodno besedo $u = sSG$ tako, da najde e'. Od tod s pomočjo generatorske matrike G pridobi $s' = sS$. Ker prav tako pozna S^{-1} lahko pridobi poslano besedilo $s = s'S^{-1}$. Dešifriranje dvojiškega Goppovega koda zahteva $O(ntm^2)$ binarnih operacij [16].

6.2 Generiranje ključa

Generiranje ključa je najpomembnejši del pri implementaciji, saj je od tega odvisna varnost kriptosistema. Če namreč izberemo premajhne parametre, smo lahko izpostavljeni napadom. Prav tako veliko vlogo igra naključnost. Če imamo slabe generatorje naključnih števil, lahko pri generiranju ključev dobimo kakšne vzorce, ki jih lahko napadalci izkoristijo v svojo korist.

Pri generiranju ključa za McElieceov kriptosistem, ki temelji na dvojiškem Goppovem kodu izberemo parametre (n, k, t) tako, da ustrezajo kate-

remu izmed podanih predlogov (razdelek 6.4.2). Parameter m potem ustreza enakosti $k = n - mt$.

Generatorski polinom $g(z)$ nad $GF(2^m)$ stopnje t izberemo tako, da generiramo naključne monične polinome $g(z) = z^t + a_{t-1}z^{t-1} + \cdots + a_1 z + a_0$, kjer so $a_i \in GF(2^m)$ za vsak i in $a_0 \neq 0$. Potem preverimo, ali je izbrani polinom nerazcepen z algoritmom za preverjanje nerazcepnosti. Če je razcepen, postopek ponovimo. Obstajajo številni algoritmi za preverjanje nerazcepnosti polinomov, med katerimi je tudi Rabinov test nerazcepnosti polinomov.

Algoritem 6.1 (Rabinov test nerazcepnosti [35])**.**
Vhod: *moničen polinom $f \in \mathbb{F}_q[x]$ stopnje t in $p_1, \ldots, p_k$ različni prafaktorji števila t.*
Izhod: *da, če je f nerazcepen in ne, če je f razcepen*

1. *Za j od 1 do k izračunaj $t_j = n/p_j$.*
2. *Za i od 1 do k ponovi*
   - 2.1. *Izračunamo $h = x^{q_i^t} - x \mod f$.*
   - 2.2. *Izračunamo $g = \gcd(f, h)$.*
   - 2.3. *Če velja $g \neq 1$, potem vrnemo, da f ni nerazcepen in končamo.*
3. *Izračunamo še $g = x^{q^t} - x \mod f$.*
4. *Če velja $g = 0$ vrnemo, da je f nerazcepen. Sicer vrnemo, da je razcepen.*

Algoritem temelji na naslednjem izreku, glej [35].

Izrek 6.1. *Naj bodo $p_1, \ldots, p_k$ vsi prafaktorji števila t in $m_i = t/p_i$ za vsak $i = 1, \ldots, k$. Polinom $f(x) \in \mathbb{F}_q[x]$ stopnje t je nerazcepen v $\mathbb{F}_q[x]$, natanko tedaj, ko $f(x)$ deli $(x^{q^t} - x)$ in velja $\gcd(f(x), x^{q^{m_i}} - x) = 1$ za vsak $i = 1, \ldots, k$.*

Poleg nerazcepnega polinoma potrebujemo še naključni matriki P in S. Permutacijsko matriko P lahko generiramo na različne načine. Eden izmed načinov je z uporabo Durstenfeldove implementacije Fisher-Yatesovega mešanja [13]. Ideja algoritma je, da vzamemo poljubno permutacijsko matriko velikosti $n \times n$ (na primer identiteto) in potem za vsak stolpec $i \in \{1, 2, \ldots, n-1\}$ trenutni stolpec i zamenjamo z naključnim stolpcem na položaju od i do n. Vseh možnih permutacij je $n!$. Vidimo lahko, da je časovna zahtevnost algoritma $O(n)$.

Generiranje nesingularne matrike S je bolj zapleteno. Najpreprostejša metoda je, da generiramo naključno matriko velikosti $k \times k$ nad $GF(2)$ in preverimo, če je njena determinanta neničelna. Če matrika ni obrnljiva, jo zavrnemo in postopek ponovimo. Ta metoda je lahko zelo neučinkovita. Dana Randall je v [36] predlagala učinkovitejši algoritem, ki rekurzivno zgradi dve matriki velikosti $k \times k$, katerih produkt je naključna nesingularna matrika. Časovna zahtevnost algoritma je $M(k) + O(k^2)$, kjer je $M(k)$ čas, ki ga potrebujemo za zmnožek dveh matrik velikosti $k \times k$.

Po uspešnem generiranju ključev dobimo naslednji kriptosistem.

Povzetek McElieceovega kriptosistema

Parametri sistema: $n, t \in \mathbb{N}$, kjer $t \ll n$

Javni ključ: (G', t), kjer je G' matrika velikosti $k \times n$ in kod $\Gamma(L, g(z))$ lahko popravi t napak

Zasebni ključ: (S, G, P), kjer je G originalna generatorska matrika velikosti $k \times n$, S obrnljiva matrika velikosti $k \times k$ in P permutacijska matrika velikosti $n \times n$, za katere velja $G' = SGP$.

Sporočilo: $s \in \{0,1\}^k$.

Kodiranje: $y = sG' + e$, kjer je $e \in \{0,1\}^n$ in $t(e) \leq t$.

Dekodiranje:

Najprej izračunamo $y' = yP^{-1}$.

Dekodiramo y' v $s' = sS$.

Izračunamo $s'S^{-1} = s$.

6.3 Primer uporabe McElieceovega kriptosistema

Naj bo primer Goppovega koda iz razdelka 5.4 podlaga za naš kriptosistem. Spomnimo se, da ta kod lahko popravi 2 napaki, če uporabimo algoritem prikazan v primeru iz razdelka 5.5.3. Za McElieceov kriptosistem moramo najprej izbrati matriki S in P, s katerima bomo spremenili generatorsko matriko

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix},$$

kjer je S obrnljiva in P permutacijska matrika.
Vzemimo naslednji matriki

$$S = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

V tem primeru je javna generatorska matrika G' enaka

$$G' = SGP = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

S to matriko lahko sedaj kdorkoli kodira informacije. Recimo, da želimo zakodirati sporočilo $s = (1, 1, 0, 0)$.
Najprej izračunamo

$$sG' = (0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0)$$

in prištejemo nek naključni vektor e s težo 2. Izberemo si, da bomo pokvarili mesti tri in deset. Pošljemo naslednje zakodirano sporočilo

$$y = sG' + e = (0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1).$$

Prejemnik želi dobiti sporočilo s iz prejetega y. Najprej izračuna yP^{-1} s svojo zasebno matriko P,

$$yP^{-1} = sSG + e' = (0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0).$$

Kot rezultat te permutacije sta se napaki premaknili na drugo in četrto mesto. Z algoritmom za popravljanje napak (algoritem 5.1) ju popravimo in dobimo

$$sSG = (0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0).$$

Iz dobljene kodne besede lahko glede na (5.24) z elementarnimi transformacijami vrstic dobimo sS.

$$\left[\; G^T \;\middle|\; (sS)^T \;\right] = \left(\begin{array}{cccc|c} 1&0&0&1&0\\ 1&0&1&0&1\\ 0&0&0&1&0\\ 0&1&0&1&1\\ 0&1&1&1&0\\ 0&1&1&0&0\\ 1&0&1&1&1\\ 1&1&1&1&0\\ 1&0&0&0&0\\ 0&1&0&0&1\\ 0&0&1&0&1\\ 0&0&0&1&0 \end{array}\right) \sim \left(\begin{array}{cccc|c} 1&0&0&0&0\\ 0&1&0&0&1\\ 0&0&1&0&1\\ 0&0&0&1&0\\ \hline 0&1&1&1&0\\ 0&1&1&0&0\\ 1&0&1&1&1\\ 1&1&1&1&0\\ 1&0&0&1&0\\ 1&0&1&0&1\\ 0&0&0&1&0\\ 0&1&0&1&1 \end{array}\right).$$

Torej je $sS = (0, 1, 1, 0)$. To še pomnožimo s S^{-1}, da dobimo

$$s = (0, 1, 1, 0) \cdot \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} = (1, 1, 0, 0),$$

kar je res naše poslano sporočilo.

6.4 Varnost kriptosistema

Tudi če je nek kriptosistem na videz še tako nezlomljiv, obstaja možnost, da bo v prihodnosti razbit, saj kriptoanalitiki vedno znova odkrivajo nove napade, ki lahko ogrozijo varnost uporabljanih šifer. Zaupanje v varnost kriptosistema pogosto narašča s časom, namreč če po mnogih letih analiz še niso odkrili napada, ki bi ogrozil šifro, je večja verjetnost, da tega ne bo mogoče storiti v prihodnosti.

Stopnja varnosti McElieceovega sistema je ostala presenetljivo visoka, kljub temu da so v zadnjih štiridesetih letih predstavili številne napade na sistem. Poleg varnosti za napade klasičnih računalnikov se zdi odporen tudi na napade s kvantnim računalnikom, saj poznani kvantni algoritmi ne vplivajo na težavnost problema dekodiranja linearnih kodov. V nadaljevanju bomo opisali bomo nekatere znane napade na kriptosisteme in navedli predloge za primerno izbiro parametrov. V nadaljevanju bomo sledili [4], [16] in [37].

Varnost McElieceovega kriptosistema temelji na dveh predpostavkah:

1. Težko je za podani vektor najti najbližjo kodno besedo iz poljubnega linearnega koda. Temu pravimo tudi problem dekodiranja linearnih kodov, za katerega vemo, da je NP-težek [3].

2. Ne obstaja tak algoritem za dekodiranje kodne besede iz koda $\mathcal{G}'$ (če ne poznamo zasebnega ključa), ki bi bil bolj učinkovit kot algoritem za poljuben linearen kod.

Za drugo predpostavko se zdi, da ne drži za poljubno izbiro generatorske matrike. Originalna različica McElieceovega kriptosistema temelji na dvojiških Goppovih kodih, za katere pa se zdi, da druga predpostavka drži. Vendar pa ti dve predpostavki nista dovolj za varen kriptosistem. Kljub temu, da je splošni problem dekodiranja linearnih kodov NP-težek, je za zagotovitev želene varnosti kriptosistema potrebna pazljiva izbira parametrov sistema.

6.4.1 Znani napadi

Napade na kriptosisteme lahko ločimo glede na namen napadalca. Njegov cilj je lahko le dešifrirati prestreženo sporočilo, najti ključ ali pa izvedeti nekaj o besedilu. Naš napadalec ima vedno na voljo javni ključ, pogosto pa tudi prestreženo sporočilo.

Po letu 1978 so bili objavljeni številni algoritmi, ki jih lahko uporabimo za napad na McElieceov kriptosistem, glej [4]. Vsem tem napadom pa se lahko izognemo s spremembo parametrov. Poleg varnosti pred klasičnimi napadi je varen tudi pred kvantnimi, saj od poznanih kvantnih napadov lahko uporabimo le Groverjev algoritem, ki pa le zmanjša konstanto pri časovni zahtevnosti.

Do sedaj najuspešnejša metoda za dekodiranje naključnega koda je dekodiranje z množico informacij (angl. *information-set decoding*). Tak algoritem je za splošni kod prvi predstavil Prange v [34]. Ker je javna generatorska matrika na videz naključna, je takšna metoda dekodiranja najuspešnejša tudi za McElieceov kriptosistem.

Napad z uporabo ISD

Poznamo več različic napada na McElieceov kriptosistem, ki uporabljajo metodo ISD. Osnovno različico je predstavil že McEliece v svojem originalnem članku [27].

Recimo, da imamo kriptosistem s parametri (n, k, t) in želimo dešifrirati prestreženo sporočilo $c = sG' + e$, kjer je e napaka s težo t. Ideja napada z

uporabo ISD je, da izberemo k izmed n koordinat v c. Če pri izbranih k v c ni nobene napake, lahko določimo del sporočila s. Če iz G' vzamemo izbranih k stolpcev, dobimo matriko $G_{[k]}$, kateri ustreza sistem $c_{[k]} = s_{[k]}G_{[k]}$, kjer sta $c_{[k]}$ in $s_{[k]}$ vektorja c in s omejena na izbranih k koordinat. Dobimo sistem k linearnih enačb s k neznankami, ki ga lahko rešimo z Gaussovo metodo.

Verjetnost, da izberemo k koordinat brez napake je $\binom{n-t}{k}/\binom{n}{k}$. Ker Gaussova eliminacija potrebuje k^3 korakov, je pričakovana zahtevnost izvajanja napada $O(k^3\binom{n-t}{k}/\binom{n}{k})$. Lahko vidimo, da je to za predlagane parametre $(3178, 2384, 68)$ odločno preveč. Pričakovana zahtevnost je namreč $1.148 \cdot 10^{52} \approx 2^{172}$.

Lee in Brickell [23] sta opazila, da ni nujno, da mora biti izbranih k koordinat brez napake. Če je število napak v izbranih k koordinatah največ j, kjer je j neka nizka meja, lahko implementiramo bolj učinkovit splošni dekodirni algoritem. Njun predlog je bil da vzamemo $j = 2$. Časovna zahtevnost napada je odvisna od j. Za $j = 0$ dobimo klasični napad ISD, če pa povečamo j na t, kjer je t število napak, dobimo napad z grobo silo, ki preveri vse vektorje napak, s časovno kompleksnostjo $O(n^t)$. Obstajajo še številne izboljšane različice napada ISD, ki uporabljajo razne verjetnostne algoritme za izbiro k koordinat [24], [45], [9], [10].

Sternov napad

Stern je napad z ISD zastavil malo drugače [41]. Vzemimo kriptogram $c = mG' + e$. Vemo, da je beseda c od kodne besede iz $\mathcal{G}'$ oddaljen za t. Ker je kod $\mathcal{G}'$ ekvivalenten kodu $\mathcal{G}$, ki lahko popravi t napak, je njegova razmaknjenost vsaj $2t + 1$. Sestavimo najprej novo matriko

$$A = \begin{pmatrix} G' \\ e \end{pmatrix}.$$

Novi kod $\mathcal{A}$ z generatorsko matriko A ima razmaknjenost enako t in edini vektor s težo t v $\mathcal{A}$ je neznani vektor e, ki ga potrebujemo za dekodiranje skritega sporočila m. Cilj napada je najti najkrajši vektor v $\mathcal{A}$ s težo t.

Sternov napad deluje nad nadzorno matriko H, ki jo lahko izračunamo iz generatorske matrike. Če lahko napadalec najde t stolpcev iz H, ki se seštejejo v 0, lahko takoj izračuna e. Primerno množico stolpcev lahko najde tako, da najprej normalizira H in izračuna vsote poljubnih j stolpcev matrike H, potem pa išče trke. Naj bodo X, Y, Z tri disjunktne podmnožice stolpcev H velikosti j. Do trka pride, če imata X in Y natanko p enic v svojih stolpcih, Z nima nobene enice v svojih stolpcih in imamo v preostalih stolpcih natanko $t - 2p$ enic. Če ne najde nobenega trka, premeša stolpce H in ponovi napad.

Večkratno šifriranje istega sporočila

Ena izmed šibkih točk McElieceovega kriptosistema je, da ni varno večkrat šifrirati isto sporočilo z isto šifrirno matriko G'. Bolj kot napad je to primer napačne uporabe sistema. Poglejmo si to na primeru. Imamo dve različni enkripciji sporočila s,

$$\begin{aligned} y &= sG' + e \quad \text{in} \\ y' &= sG' + e'. \end{aligned}$$

Na koordinatah i, kjer sta y in y' različna vemo, da je natanko eden izmed e_i in e'_i enak 1. Na koordinatah j, kjer sta enaka pa imamo dve možnosti: ali velja $e_j = e'_j = 0$ ali $e_j = e'_j = 1$. Za naključno izbrana e in e' pričakujemo, da bo

- $e_i = e'_i = 0$ na
$$\frac{(n-t)^2}{n} \text{ mestih},$$
- $e_i = 0$ in $e'_i = 1$ ali $e_i = 1$ in $e'_i = 0$ na
$$\frac{2t(n-t)}{n} \text{ mestih},$$
- $e_i = e'_i = 1$ na
$$\frac{t^2}{n} \text{ mestih}.$$

Poglejmo, kaj to pomeni za priporočene parametre $(3178, 2384, 68)$. Pričakujemo, da je $e_i = e'_i = 1$ na

$$\frac{68^2}{3178} \approx 1.45500314663$$

koordinatah. Vseh koordinat, kjer sta y in y' enaka, je

$$\frac{(3178-68)^2}{3178} + \frac{68^2}{3178} \approx 3045.$$

Torej lahko pričakujemo, da med 3045 koordinatami pri katerih sta y in y' enaka samo dve vsebujeta napako. Da najdemo sporočilo s, potrebujemo množico k koordinat, kjer sta y in y' brez napake. V tem primeru je verjetnost, da izberemo tako množico enaka

$$\frac{\binom{3178-2}{2384}}{\binom{3178}{2384}} \approx 0.06236236575.$$

Torej pričakujemo, da bo napad uspešen po samo

$$\frac{1}{0.06236236575} \approx 16.0353121305$$

poskusih.

Ostali napadi

Metode ISD poskušajo dekodirati eno samo kodirano sporočilo in napadalec pri tem ne odkrije nobene informacije o zasebnem ključu. Obstajajo pa tudi nekateri prispevki [19, 26], ki preučujejo napade, ki poskušajo pridobiti zasebni ključ iz javne generatorske matrike G.

Matriki S in P sta bistvenega pomena za varnost kriptosistema, namreč če napadalec pridobi originalno generatorsko matriko G, potem mu ni težko najti generatorski polinom $g(z)$. Če pa pozna generatorski polinom, lahko za dekodiranje prestreženega sporočila uporabi učinkovit dekodirni algoritem za Goppove kode.

Pri McElieceovem kriptosistemu sta skrivni kod $\mathcal{G}$ z učinkovitim dekodirnim algoritmom in javni kod $\mathcal{G}'$ ekvivalentna; biti kodnih besed so le permutirani za neko neznano permutacijo P. Če napadalec pozna skrivni kod, lahko

permutacijo P dobi z ločevalnim algoritmom (angl. *support splitting algorithm*, SSA) [39] polinomske časovne zahtevnosti. Ta algoritem lahko uporabimo tudi za napad z grobo silo, če izbrani kod pripada majhnemu naboru dopustnih kodov. Na primer pri klasičnem McElieceovem kriptosistemu, ki temelji na Goppovem kodu, je dekodirni algoritem določen z generatorskim polinomom $g(z)$. Če generatorski polinom omejimo na množico nerazcepnih polinomov z dvojiškimi koeficienti, dobimo premajhno množico ključev, da bi kriptosistem zdržal napad s preizkušanjem vseh mogočih ključev s pomočjo SSA.

6.4.2 Izbira parametrov

Stopnja varnosti je določena glede na to, koliko korakov potrebujejo za napade najboljši znani algoritmi. Pri simetričnih kriptosistemih je zaželeno, da je to napad s preverjanjem vseh ključev in ima za dvojiški ključ dolžine n časovno zahtevnost $O(2^n)$. Za tak kriptosistem pravimo, da ima n-bitno varnost.

Za kriptosisteme z javnim ključem pa poznamo boljše napade kot je izčrpno preiskovanje ključev. Če najboljši napad potebuje $O(2^n)$, je to podobno kot, da bi pri simetričnem kriptosistemu pregledali 2^n ključev. Za trenutno stopnjo tehnologije zahtevamo 128-bitno varnost za kratkoročno komunikacijo in 256-bitno za pomembnejša sporočila.

Originalni parametri $n = 1024$, $k = 524$ in $t = 50$, ki jih je predlagal McEliece v [27], niso dovolj varni, saj zagotavljajo približno 50-bitno varnost. Na žalost pa optimalne izbire parametrov za določeno stopnjo varnosti za McElieceov kriptosistem ne moremo zapisati v obliki zaprte formule. Mnenja o velikosti parametrov za doseganje tipičnih ravni varnosti so različna, ponavadi pa jih izberemo tako, da je razmerje $R = k/n$ približno 0.8. Predloge lahko najdemo v številnih analizah varnosti in pa tudi raznih implementacijah sistema [14], [7].

V spodnji tabeli smo navedli nekatere predloge za parametre, ki zagotavljajo različne stopnje varnosti. Vsi predlogi upoštevajo neko različico na-

pada z uporabo ISD. Velikost ključa je približno enaka velikost generatorske matrike, torej $k \cdot n$.

Stopnja varnosti	Ref.	(n,k,t)	Zasebni ključ [kB]
50	[27]	(1023,524,50)	66
80	[5]	(2048,1782,27)	438
80	[30]	(1702,1219,45)	254
80	[7]	(2048,1696,32)	424
128	[6]	(3178,2384,68)	925
128	[7]	(4096,3604,41)	1802
256	[6]	(6944,5208,136)	4415

Bernstein et al so v [6] ocenili varnost kriptosistema z izboljšano različico Sternovega napada. Ob upoštevanju njihovih predlogov dobimo za 128-bitno varnost ključ velikosti približno 925 kB, kar je precej več kot 348 B za RSA primerljive varnosti. Za 256-bitno varnosti pa potrebujemo ključ velikosti približno 4415 kB v primerjavi s 1920 B za RSA.

Poglavje 7

Zaključek

Izgradnja kvantnega računalnika se zdi vse bolj verjetna, zato je iskanje kriptosistema, ki bi bil odporen na napade kvantnega računalnika zaželeno. Kljub temu, da za trenutno tehnologijo uporabljani sistemi pogosto zagotavljajo dovolj visoko varnost, pa je potrebno iskanje novih alternativ. Predvsem zato, ker ni dovolj samo odkritje sistema, ki je domnevno odporen, ampak je potreben čas, da se pridobi zaupanje v kriptosistem in njegovo domnevno nezlomljivost. Poleg tega je potreben tudi čas za primerno implementacijo kriptosistema.

Prav zato mislimo, da je McElieceov kriptosistem dobra izbira za nadaljnje preiskovanje, saj je že nekaj desetletij pod drobnogledom določenih strokovnjakov, ki še niso odkrili nobenega napada na klasičnem ali kvantnem računalniku, ki bi resno ogrožal kriptosistem. Poleg tega lahko šifriranje in dešifriranje izvedemo zelo hitro. McElieceov kriptosistem v originalu temelji na nerazcepnem Goppovem kodu, kar ostaja najbolj primerna različica kriptosistema do sedaj, kljub številnim predlogom za uporabo drugih podrazredov Goppovih kodov ali drugi alternantih kodov.

Glavni razlog, da McElieceov kriptosistem še ni zamenjal RSA, je njegova nepraktičnost, saj ima zelo velik javni ključ, in pa tudi to, da ga ni mogoče uporabiti za podpisovanje na tako naraven način kot pri RSA. Druga težava je delno že rešena, saj že obstajajo sheme za digitalno podpisovanje, ki te-

meljijo na McElieceovem kriptosistemu. Prav tako pa velikost ključa z napredkom tehnologije postaja vse manjša težava, zato predstavlja McElieceov kriptosistem vse boljši kandidat, da počasi zamenja RSA. Z ozirom na prihodnost je zato nadaljnje raziskovanje varnosti in izboljšava implementacij kriptosistema smiselna.

Literatura

[1] D. Augot, M. Finiasz in N. Sendrier. A family of fast syndrome based cryptographic hash functions. V *Progress in Cryptology – Mycrypt 2005*, strani 64–83. Springer Berlin, Heidelberg, 2005.

[2] E. Berlekamp. *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.

[3] E. R. Berlekamp, R. J. McEliece in H. C. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[4] D. J. Bernstein, T. Chou, T. Lange, R. Misoczki, R. Niederhagen, E. Persichetti, P. Schwabe, J. Szefer in W. Wang. Classic McEliece: conservative code-based cryptography 29 November 2017. 2017.

[5] D. J. Bernstein, T. Lange in C. Peters. Attacking and defending the McEliece cryptosystem. V *Post-quantum cryptography*, zvezek 5299 iz *Lecture Notes in Comput. Sci.*, strani 31–46. Springer, Berlin, 2008.

[6] D. J. Bernstein, T. Lange in C. Peters. Smaller decoding exponents: ball-collision decoding. V *Advances in cryptology—CRYPTO 2011*, zvezek 6841 iz *Lecture Notes in Comput. Sci.*, strani 743–760. Springer, Heidelberg, 2011.

[7] B. Biswas in N. Sendrier. McEliece cryptosystem implementation: theory and practice. V *Post-quantum cryptography*, zvezek 5299 iz *Lecture Notes in Comput. Sci.*, strani 47–62. Springer, Berlin, 2008.

[8] R. C. Bose in D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3:68–79, 1960.

[9] A. Canteaut in F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Trans. Inform. Theory*, 44(1):367–378, 1998.

[10] A. Canteaut in N. Sendrier. Cryptanalysis of the original McEliece cryptosystem. V *Advances in cryptology—ASIACRYPT'98 (Beijing)*, zvezek 1514 iz *Lecture Notes in Comput. Sci.*, strani 187–199. Springer, Berlin, 1998.

[11] N. T. Courtois, M. Finiasz in N. Sendrier. How to achieve a McEliece-based digital signature scheme. V *Advances in cryptology—ASIACRYPT 2001 (Gold Coast)*, zvezek 2248 iz *Lecture Notes in Comput. Sci.*, strani 157–174. Springer, Berlin, 2001.

[12] W. Diffie in M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[13] R. Durstenfeld. Algorithm 235: random permutation. *Communications of the ACM*, 7(7):420, 1964.

[14] T. Eisenbarth, T. Güneysu, S. Heyse in C. Paar. MicroEliece: McEliece for embedded devices. V *Cryptographic Hardware and Embedded Systems-CHES 2009*, strani 49–64. Springer, 2009.

[15] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31(4):469–472, 1985.

[16] D. Engelbert, R. Overbeck in A. Schmidt. A summary of McEliece-type cryptosystems and their security. *J. Math. Cryptol.*, 1(2):151–199, 2007.

[17] J. Fischer in J. Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. V *Advances in Cryptology — EUROCRYPT '96*, strani 245–255. Springer Berlin Heidelberg, 1996.

[18] P. Gaborit, C. Lauradoux in N. Sendrier. Synd: a fast code-based stream cipher with a security reduction. V *2007 IEEE International Symposium on Information Theory*, strani 186–190, 2007.

[19] J. K. Gibson. Equivalent Goppa codes and trapdoors to McEliece's public key cryptosystem. V *Advances in cryptology—EUROCRYPT '91 (Brighton, 1991)*, zvezek 547 iz *Lecture Notes in Comput. Sci.*, strani 517–521. Springer, Berlin, 1991.

[20] V. D. Goppa. A new class of linear correcting codes. *Problemy Peredači Informacii*, 6(3):24–30, 1970.

[21] L. K. Grover. A fast quantum mechanical algorithm for database search. V *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, strani 212–219. ACM, New York, 1996.

[22] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2:147–156, 1959.

[23] P. J. Lee in E. F. Brickell. An observation on the security of McEliece's public-key cryptosystem. V *Advances in cryptology—EUROCRYPT '88 (Davos, 1988)*, zvezek 330 iz *Lecture Notes in Comput. Sci.*, strani 275–280. Springer, Berlin, 1988.

[24] J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Trans. Inform. Theory*, 34(5):1354–1359, 1988. Coding techniques and coding theory.

[25] R. Lidl in H. Niederreiter. *Finite fields*, zvezek 20 iz *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, druga izdaja, 1997.

[26] P. Loidreau in N. Sendrier. Weak keys in the McEliece public-key cryptosystem. *IEEE Transactions on Information Theory*, 47(3):1207–1211, March 2001.

[27] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. V *DNS Progress Report 42-44*, strani 114–116. Jet Propulsion Laboratory, Pasadena, 1978.

[28] A. J. Menezes, P. C. van Oorschot in S. A. Vanstone. *Handbook of applied cryptography*. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, Boca Raton, FL, 1997.

[29] T. K. Moon. *Error Correction Coding, Mathematical Methods and Algorithms*. Wiley, 2005.

[30] R. Niebuhr, M. Meziani, S. Bulygin in J. Buchmann. Selecting parameters for secure McEliece-based cryptosystems. *International Journal of Information Security*, 11(3):137–147, 2012.

[31] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Probl. Control and Inform. Theory*, 15:19–34, 1986.

[32] R. Overbeck in N. Sendrier. Code-based cryptography. V D.J. Bernstein, J. Buchmann in E. Dahmen, ur., *Post-Quantum Cryptography*, strani 95–145. Springer, 2009.

[33] N. Patterson. The algebraic decoding of goppa codes. *IEEE Transactions on Information Theory*, 21(2):203–207, 1975.

[34] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[35] M. O. Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on computing*, 9(2):273–280, 1980.

[36] D. Randall. Efficient generation of random nonsingular matrices. *Random Structures & Algorithms*, 4(1):111–118, 1993.

[37] M. Repka in P. Zajac. Overview of the McEliece cryptosystem and its security. *Tatra Mt. Math. Publ.*, 60:57–83, 2014.

[38] R. L. Rivest, A. Shamir in L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21(2):120–126, 1978.

[39] N. Sendrier. Finding the permutation between equivalent linear codes: the support splitting algorithm. *IEEE Trans. Inform. Theory*, 46(4):1193–1203, 2000.

[40] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. V *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*, strani 124–134. IEEE Comput. Soc. Press, Los Alamitos, CA, 1994.

[41] J. Stern. A method for finding codewords of small weight. V *Coding theory and applications (Toulon, 1988)*, zvezek 388 iz *Lecture Notes in Comput. Sci.*, strani 106–113. Springer, New York, 1989.

[42] J. Stern. A new identification scheme based on syndrome decoding. V *Advances in Cryptology — CRYPTO' 93*, strani 13–21. Springer Berlin Heidelberg, 1994.

[43] D. R. Stinson. *Cryptography*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, tretja izdaja, 2006.

[44] H. C. A. van Tilborg. *Coding theory, a first course*. Citeseer, 1993.

[45] J. van Tilburg. On the McEliece public-key cryptosystem. V *Advances in cryptology—CRYPTO '88 (Santa Barbara, CA, 1988)*, zvezek 403 iz *Lecture Notes in Comput. Sci.*, strani 119–131. Springer, Berlin, 1990.

[46] A. Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, 1997.

[47] P. Véron. Improved identification schemes based on error-correcting codes. *Appl. Algebra Engrg. Comm. Comput.*, 8(1):57–69, 1997.

[48] I. Vidav. *Algebra*. DMFA, Ljubljana, 1987.

[49] D. Welsh. *Codes and cryptography*. Oxford Science Publications. The Clarendon Press, Oxford University Press, New York, 1988.