Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Fakulteta za matematiko in fiziko

Lea Vohar

Kode QR

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentorica: izr. prof. dr. Arjana Žitnik

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kode QR

Tematika naloge:

Koda QR je dvodimenzionalna črtna koda, namenjena strojnemu branju podatkov. Dandanes so kode QR takorekoč nepogrešljive, najdemo jih na plakatih, oglasih, vstopnicah, v časopisih, na izdelkih. Pogosto so v njih zapisani spletni naslovi, na katerih izvemo več o dogodkih oziroma izdelkih.

V diplomskem delu opišite kode QR: zgradbo simbolov kode QR za različne verzije, zapis različnih tipov podatkov v obliki kode QR in branje kod QR. Podatki so zaradi večje zanesljivosti branja zakodirani z Reed-Solomonovimi kodami za popravljanje napak. Zato podrobneje predstavite tudi Reed-Solomonove kode in potrebno matematično ozadje, predvsem končne obsege. Obravnavajte tudi varnost uporabe kod QR. Opišite na primer možne zlorabe v primeru, ko je v simbolu QR shranjen spletni naslov.

Iskreno se zahvaljujem mentorici, izr. prof. dr. Arjani Žitnik za potrpežljivost, prizadevnost, strokovnost in usmerjanje pri izdelavi diplomskega dela. Posebna hvala velja mojim staršem in teti Nadi, ki ste verjeli vame in me spodbujali.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
BCH	Bose-Chaudhuri-Hocquenghem algorithm	algoritem Bose-Chaudhuri-Hocquenghem
PGZ	Peterson-Gorenstein-Zierler algorithm	algoritem Peterson-Gorenstein-Zierler
QR	quick response	hitra odzivnost
RS	Reed-Solomon	Reed-Solomon
SQL	structured query language	strukturirani povpraševalni jezik
UPC	universal product code	univerzalna koda izdelka

Povzetek

Naslov: Kode QR

Avtor: Lea Vohar

V diplomskem delu obravnavamo kode QR. Koda QR je dvodimenzionalna črtna koda, namenjena strojnemu branju podatkov. Podatki so zaradi večje zanesljivosti branja zakodirani z Reed-Solomonovimi kodami za popravljanje napak. V prvem delu diplomskega dela najprej ponovimo končne obsege, ki jih potrebujemo v nadaljevanju. Nato obravnavamo kode za popravljanje napak. Definiramo Reed-Solomonove kode, ki dosežejo Singletonovo mejo in tako popravijo največje možno število napak glede na število simbolov, ki jih dodamo sporočilu. Predstavimo tudi algoritma za kodiranje in dekodiranje Reed-Solomonovih kod, ki sta uporabljena pri kodah QR. V drugem delu podrobneje predstavimo zgradbo simbola kode QR. S primerom razložimo, kako se sporočilo zapiše in zakodira v kodo QR. Na koncu pogledamo varnost kod QR in primere njihovih zlorab.

Ključne besede: kode QR, Reed-Solomonove kode, kode za popravljanje napak.

Abstract

Title: QR codes

Author: Lea Vohar

In this thesis QR codes are considered. A QR code is a two-dimensional bar code for machine-readable data. Data are first encoded using Reed-Solomon error correction codes to ensure readability even if the QR code is damaged. In the first part of the thesis we first review finite fiels that are needed in the sequel. Next we introduce error correction codes and define Reed-Solomon codes. Reed-Solomon codes attain the Singleton bound which means that they are capable of correcting the maximum possible number of errors with respect to to the number of symbols that are added to the message. We also present algorithms for coding and decoding Reed-Solomon codes, which are used for QR codes. In the second part of the thesis we describe the structure of the QR code symbol in detail. We explain how the message is encoded in the QR code and how a QR code is decoded. We illustrate these procedures with examples. Finally, we discuss the security of QR codes and possibilities of their abuse.
Keywords: QR codes, Reed-Solomon codes, error correcting codes.

Poglavje 1

Uvod

Koda QR. Neuporabna kradljivka prostora ali inovativna oblika oglaševanja? Kako lahko v črno-bele kvadratke shranimo besedilo, povezavo do spletne strani ali podatke potrebne za plačilo bančnih računov? Kaj nam zagotavlja zanesljivost prenosa podatkov, ko skeniramo kodo QR?

Koda QR (angl. *quick response*) je hitro odzivna dvodimenzionalna črtna koda, sestavljena iz kvadratnih črno-belih modulov in treh pozicijskih vzorcev. Leta 1994 jo je na Japonskem razvilo podjetje Denso Wave [5]. Želeli so razviti črtno kodo, s katero bi lahko zapisali več informacij na dani površini. Z zapisom podatkov v horizontalni in vertikalni smeri jim je to uspelo. S kodo QR porabimo za zapis enake količine podatkov približno eno desetino prostora enodimnzionalne črtne kode UPC. Primerjavo lahko vidimo na sliki 1.1. Z dodano informacijo o položaju kode v simbol - s tremi pozicijskimi vzorci - koda QR omogoča hitro zaznavanje in branje, kar razkriva njeno ime. Sprva so jo uporabljali za sledenje avtomobilskih delov pri proi-

Slika 1.1: Primerjava velikosti črtnih kod z isto vsebino.

zvodnji Toyotinih vozil. Razvoj mobilnih naprav je pripomogel k temu, da se je uporaba razširila izven industrijskih objektov. Dandanes so kode QR ključno marketinško orodje. Najdemo jih na plakatih, oglasih, vstopnicah, v časopisih, na izdelkih. Uporabnik s kamero na mobilni napravi skenira kodo s pomočjo prosto dostopne aplikacije. S skeniranjem kode QR lahko izvemo podrobnejše informacije o oglaševanem dogodku, o sestavinah izdelka, v muzeju pa lahko dostopamo do dodatnih vsebin. Najpogosteje je v kodi QR zapisan spletni naslov. Uporabljajo se tudi za sledenje poštnih paketov, prtljage na letališču, pri nadzoru prevozu tovora. Skoraj ni stvari, ki ne bi imela kode QR. Zaželene so zaradi poenostavljanja nalog in nevsiljivosti. Poglejmo si še nekaj primerov uporabe [2, 18]:

- Med čakanjem na letališču v Denverju si lahko v digitalni knjižnici s skeniranjem kode QR na mobilno napravo prenesemo knjigo.

- V kodo QR lahko shranimo naše pomembne zdravstvene podatke in jo namestimo na čelado, kolo, uro, hrbtno stran mobilnega telefona. V primeru nezgode lahko zdravstveno osebje z varno aplikacijo Code d'Urgence s skeniranjem hitro dostopa do naših podatkov in se ustrezno odzove.

- Zraven modela za šivanje je v reviji Burda objavljena koda QR, ki nas poveže do video vsebin za učenje postopka šivanja in dodatne razlage.

Kodo QR lahko izdelamo s prosto dostopnim generatorjem kod QR na osebnem računalniku. V program vnesemo podatke, ki jih želimo zakodirati, in ta nam zgenerira našo kodo QR.

Koda QR je ponavadi natisnjena na medij, ki se lahko poškoduje ali umaže. Tako je lahko koda neuporabna. A kode QR imajo možnost obnovitve podatkov. Uporaba Reed-Solomonovih kod za popravljanje napak omogoča, da lahko vsebino kode QR preberemo, če je poškodovanih do 30% podatkov. Temu ključnemu delu, ki kodo QR dela zanesljivo, se posvetimo v diplomskem delu.

Diplomsko delo je strukturirano na naslednji način. V drugem poglavju na kratko vpeljemo končne obsege in povzamemo njihove pomembnejše lastnosti. V tretjem poglavju obravnavamo kode za popravljanje napak. Definiramo linearne in ciklične kode ter izpeljemo Singletonovo mejo, ki pove, koliko najmanj je treba podaljšati sporočilo, da popravimo dano število napak. Četrto poglavje namenimo opisu Reed-Solomonovih kod in njihovih lastnosti. Opišemo učinkovit algoritem za dekodiranje Reed-Solomonovih kod. V petem poglavju opišemo zgradbo simbola kode QR in postopek zapisovanja podatkov v simbol. Na primeru si ogledamo generiranje kontrolnih simbolov za popravljanje napak, ki omogočijo zanesljivo branje poškodovanih delov simbola. Na koncu pogledamo primere zlorab kod QR.

Poglavje 2

Matematične osnove

Šele s prihodom računalnikov so se končni obsegi uveljavili v uporabni matematiki. Dandanes so v kriptografiji in teoriji kodiranja ključnega pomena. Računanje v obsegu $GF(2^n)$ se učinkovito izvaja na računalnikih, kjer so podatki zapisani binarno. Najpogosteje uporabljen končen obseg je $GF(2^8)$ z 2^8 elementi, kjer vsak element predstavlja en bajt informacije. Reed-Solomonove kode, ki se uporabljajo v kodah QR, so definirane nad obsegom $GF(2^8)$, kakor tudi operacije kriptosistema AES.

V tem poglavju obnovimo nekatere algebraične strukture, s katerimi računamo v nadaljevanju. Pri tem sledimo knjigam [1, 10, 20].

Definicija 2.1. Množica G z binarno operacijo $\circ$ je *grupa* $(G, \circ)$, če veljajo naslednje lastnosti:

- Množica G je zaprta za operacijo $\circ$: za vsak $a, b \in G$, velja $a \circ b \in G$.
- Asociativnost: $a \circ (b \circ c) = (a \circ b) \circ c$ za poljubne $a, b, c \in G$.
- Obstoj enote: za vsak $a \in G$, obstaja element $e \in G$, da velja

$$e \circ a = a \circ e = a.$$

- Obstoj inverznih elementov: za vsak $a \in G$ obstaja tak element $a^{-1} \in G$, da je

$$a \circ a^{-1} = a^{-1} \circ a = e.$$

Grupa $(G, \circ)$ je *komutativna* ali *Abelova*, če za poljubna $a, b \in G$, velja $a \circ b = b \circ a$. Operacijo $\circ$ v Abelovi grupi ponavadi pišemo kot $+$, kjer je enota 0 in inverzni element elementa a označimo z $-a$.

V nadaljevanju bomo namesto $(G, \circ)$ pisali krajše kar G. Če je število elementov grupe končno, pravimo, da je grupa *končna*. Število elementov v končni grupi G imenujemo *moč grupe* in označimo z $|G|$.

Naj bo G končna grupa in α element G. Naj bo i naravno število. Definirajmo $\alpha^0 = e$ in $\underbrace{\alpha^i = \alpha \circ \alpha \circ \cdot \circ \alpha}_{i-krat}$. Najmanjše naravno število s, da velja $\alpha^s = e$, imenujemo *red elementa*. Označimo ga z $\text{red}(\alpha)$. Če tak s ne obstaja, pravimo, da ima α neskončen red: v tem primeru je grupa G neskončna grupa. Poglejmo nekaj osnovnih lastnosti grup.

Trditev 2.1. *Naj bo G končna grupa in naj bo n moč grupe G. Potem velja:*

1. *red(α) deli n,*
2. $\alpha^n = e$.

Grupa G je *ciklična*, če obstaja $\alpha \in G$, da je vsak element grupe G enak neki potenci elemeta α. Element α imenujemo *generator* grupe G. Ciklična grupa G torej vsebuje element reda $|G|$. Ciklično grupo G lahko zapišemo kot

$$G = \{\alpha, \alpha^2, \ldots, \alpha^{|G|} = e\}.$$

Oznaka 2.2. Z $G^* = G \backslash \{0\}$ označimo množico neničelnih elementov množice G.

Poglejmo si algebrsko strukturo, v kateri sta definirani dve operaciji.

Definicija 2.3. Množica G z dvema binarnima operacijama $+$, $\circ$ je *obseg* če velja:

- $(G, +)$ je Abelova grupa; naj bo 0 enota v tej grupi,
- $(G \backslash \{0\}, \circ)$ je grupa,

- Distributivnost: za poljubne $a, b, c \in G$ velja $a \circ (b + c) = (a \circ b) + (a \circ c)$ in $(a + b) \circ c = (a \circ c) + (b \circ c)$.

Operaciji $+$ in $\circ$ v obsegu imenujemo *seštevanje* in *množenje*. Če je v obsegu množenje komutativno, potem je obseg *komutativen*.
V obsegu lahko definiramo tudi operaciji odštevanje in deljenje, kar lahko zapišemo kot $a - b = a + (-b)$ in $a/b = a \cdot b^{-1}$ za $b \neq 0$. Pri tem $(-b)$ pomeni inverzni element b glede na operacijo $+$.

Zgled 2.1. Množica $\mathbb{Z}_2$ je komutativen obseg ostankov po modulu 2, $\mathbb{Z}_2 = \{0, 1\}$. Operacijo seštevanja po modulu 2 označimo z $+_2$, operacijo moženja po modulu 2 pa z $\circ_2$. Poglejmo, da je $(\mathbb{Z}_2, +_2, \circ_2)$ res obseg. Po definiciji je grupa $(\mathbb{Z}_2, +_2)$ Abelova grupa, kar ni težko preveriti. Množica $\mathbb{Z}_2$ je zaprta za seštevanje po modulu 2, operacija seštevanja je asociativna, enota za seštevanje je 0 in vsak element ima svoj inverz. Inverz za 1 je 1 in inverz za 0 je 0. Operacija $+_2$ je komutativna, torej je grupa $(\mathbb{Z}_2, +_2)$ res Abelova.

Množica $\mathbb{Z}_2^* = \{1\}$ z asociativno operacijo $\circ_2$, enoto 1 in inverzom 1 je grupa. Ker za operaciji $+_2$ in $\circ_2$ velja zakon distributivnosti, je $(\mathbb{Z}_2, +_2, \circ_2)$ res obseg. Poglejmo si tabeli za operaciji množenja in seštevanja.

$+$	0	1
0	0	1
1	1	1

$\circ$	0	1
0	0	0
1	0	1

Definicija 2.4. Naj bosta $(G_1, +_1, \circ_1)$ in $(G_2, +_2, \circ_2)$ obsega. Preslikava $\varphi : G_1 \to G_2$, je *homomorfizem*, če za vsak $a, b \in G_1$ velja

1. $\varphi(a +_1 b) = \varphi(a) +_2 \varphi(b)$ in
2. $\varphi(a \circ_1 b) = \varphi(a) \circ_2 \varphi(b)$.

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

Obseg $(G, +, \circ)$ je *končen*, če je množica G končna. Končni obseg s q elementi imenujemo *Galoisov obseg* moči q. Označimo ga z $GF(q)$.

Izrek 2.2. *Naj bo G končen obseg. Potem veljajo naslednje lastnosti.*

1. *Obseg G je komutativen.*
2. *Moč obsega G je potenca praštevila.*
3. *Grupa $(G^*, \circ)$ je ciklična.*
4. *Poljubna dva končna obsega iste moči sta izomorfna.*

Zgled 2.2. Naj bo p poljubno praštevilo. Naj bosta operaciji seštevanje po modulu p in množenje po modulu m. Potem je $(\mathbb{Z}_p, +_p, \circ_p)$ končen obseg.

Neničelni elementi iz $GF(q)$ tvorijo grupo $GF(q)^*$, v kateri so vsi elementi obrnljivi. Po 3. točki izreka 2.2 velja, da je grupa $GF(q)^*$ ciklična. Za vsak element $\beta \in GF(q)^*$ velja $\beta^{q-1} = 1$. Generatorje ciklične grupe imenujemo *primitivni elementi obsega*. Naj bo α primitivni element obsega. Potem velja, da $\alpha^{q-1} = 1$ in $\alpha^i \neq 1$ za vsak $i \in \{1, \ldots, n-1\}$.

Vsi končni obsegi z enakim številom elementov so med seboj izomorfni. Obstaja več načinov za predstavitev elementov končnega obsega. V nadaljevanju jih bomo predstavili s polinomi. Poglejmo si postopek.

Naj bo p poljubno praštevilo, n poljubno naravno število in $q = p^n$. Množica $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ je končen obseg z operacijama $+_p$ in $\circ_p$, kjer je operacija $+_p$ seštevanje po modulu p in operacija $\circ_p$ množenje po modulu p. Nad $\mathbb{Z}_p$ poiščemo nerazcepen polinom $f(x)$ stopnje n, katerega koeficienti so iz obsega $\mathbb{Z}_p$. Elementi obsega $GF(p^n)$ so polinomi stopnje največ $n-1$, in jih lahko interpretiramo kot ostanke pri deljenju s polinomom $f(x)$. Polinome seštevamo po modulu $f(x)$ kot običajno; produkt dveh polinomov pa dobimo tako, da najprej polinoma zmnožimo, nato pa dobljeni polinom delimo s $f(x)$. Za rešitev vzamemo ostanek pri deljenju. Koeficiente polinoma seštevamo in množimo po modulu p (v enačbah bomo označili računanje po modulu

označili z mod). Tako konstruirana množica vseh polinomov stopnje manjše od n s koeficienti iz $\mathbb{Z}_p$ in z operacijama $+_f$, $\circ_f$, je končen obseg $GF(q^n)$.

Polinom $a_p x^p + \ldots + a_1 x + a_0$ lahko zapišemo bolj kompaktno tako, da po vrsti naštejemo njegove koeficiente, najprej koeficient na najvišji potenci: $a_p \ldots a_1 a_0$.

V zgledu si poglejmo generiranje elementov obsega.

Zgled 2.3. Naj bo $GF(2^4)$ končen obseg generiran z nerazcepnim polinomom $f(x) = x^4 + x + 1$. Elemente obsega izračunamo po zgoraj opisanem postopku in jih zapišimo v tabelo. Seštevamo in množimo polinome $1, x, x+1$. Zmnožimo npr. polinom x in $x+1$ in dobimo $x^2 + x$, kar lahko zapišemo kot 0110. Nato množimo naprej z x in dobimo $x^3 + x^2$, kar je enako 1100 in tako naprej. Opazimo, da je element x generator $GF(2^4)^*$. Vse elemente obsega lahko predstavimo s primitivnim elementom x (levi stolpec).

Tabela 2.1: Elementi končnega obsega $GF(2^4)$, generiranega s polinomom $p(x) = x^4 + x + 1$

potenca x	*polinom*	$a_3a_2a_1a_0$	potenca x	*polinom*	$a_3a_2a_1a_0$
	0	0000	x^7	$x^3 + x + 1$	1011
x^0	1	0001	x^8	$x^2 + 1$	0101
x^1	x	0010	x^9	$x^3 + x$	1010
x^2	x^2	0100	x^{10}	$x^2 + x + 1$	0111
x^3	x^3	1000	x^{11}	$x^3 + x^2 + x$	1110
x^4	$x + 1$	0011	x^{12}	$x^3 + x^2 + x + 1$	1111
x^5	$x^2 + x$	0110	x^{13}	$x^3 + x^2 + 1$	1101
x^6	$x^3 + x^2$	1100	x^{14}	$x^3 + 1$	1001

Naj bo α primitivni element obsega $GF(q)$. Elemente obsega $GF(q)$ lahko zapišemo kot $0, \alpha, \alpha^2, \ldots, \alpha^{q-1}$. Pri predstavitvi elementov obsega s primitivnim elementom je množenje elementov enostavnejše. Velja $\alpha^{n_1} \cdot \alpha^{n_2} = \alpha^{n_1+n_2 \mod (q-1)}$. V potencah seštevamo po modulu $q-1$ po 2. točki trditve

2.1. Seštevanje elementov pa je potem malo zahtevnejše. Pri tem si pomagamo z polinomsko predstavitvijo potenc primitivnega elementa, kot smo opisali zgoraj. Za $n_1 \leq n_2$ je $\alpha^{n_1} + \alpha^{n_2} = \alpha^{n_1}(1 + \alpha^{n_2 - n_1})$. Seštevanje z izpostavitvijo člena z manjšo potenco prevedemo na množenje. Dobimo produkt potence primitivnega elementa α in polinoma. Polinom nato zapišemo kot potenco α in zmnožimo, kot je opisano zgoraj. Poglejmo si primer.

Zgled 2.4. Naj bo α primitivni element obsega $GF(2^4)$ generiranega s polinomom $x^4 + x + 1$. S pomočjo tabele 2.1 si poglejmo primer seštevanja in množenja elementov obsega:

$$\alpha^5 \cdot \alpha^{12} = \alpha^{17 \mod 15} = \alpha^2,$$
$$\alpha^5 + \alpha^6 = \alpha^5(1 + \alpha) = \alpha^5\alpha^4 = \alpha^9.$$

Poglavje 3

Kode za popravljanje napak

Med pošiljanjem podatkov po komunikacijskem kanalu lahko pride do motenj, ki spremenijo poslano sporočilo. Dodajanje kontrolnih simbolov sporočilu pred pošiljanjem omogoči, da sistem zazna napake. „Če zna računalnik sam odkriti napako, zakaj ne zna najti tudi njenega mesta in je odpraviti?“, je dejal Richard Hamming pred več kot 65 leti in začel se je razvoj kod za popravljanje napak, s katerimi se seznamimo v tem poglavju [7].

Definiramo bločne kode dolžine n in njihove lastnosti. Opišemo linearne in ciklične kode ter izpeljemo Singletonovo mejo, ki nam pove, koliko najmanj simbolov moramo dodati sporočilu, da bo koda popravila želeno število napak. Sledimo knjigi [22].

Definicija 3.1. Naj bo Σ končna množica, imenujemo jo *kodna abeceda*. Elemete Σ imenujemo *kodni simboli*. S Σ^n označimo množico vseh besed dolžine n iz Σ. *Bločna koda za popravljanje napak* dolžine n nad abecedo Σ je podmnožica Σ^n. Elementi kode so *kodne besede*.

Definicija 3.2. Za x in $y \in \Sigma^n$ definirajmo $d(x, y) = |\{i; x_i \neq y_i\}|$, ki označuje število mest, kjer se x in y razlikujeta. Številu $d(x, y)$ pravimo *Hammingova razdalja* med besedami x in y.

Ni težko preveriti, da je Hammingova razdalja res metrika.

Definicija 3.3. *Razdalja kode* C je $d(C) = \min\limits_{\substack{x,y\in C\\ x\neq y}} d(x,y)$.

Razdalja kode je torej minimalna razdalja med različnima kodnima besedama. Kot bomo videli v nadaljevanju, ima razdalja ključno vlogo pri dekodiranju, saj določa koliko napak lahko popravimo. Večja je razdalja, več napak lahko odkrijemo in popravimo. V splošnem jo je težko določiti.

Definicija 3.4. *Teža besede* x je $t(x) = |\{i; x_i \neq 0\}|$. Teža označuje število neničelnih mest v besedi.

Velja, da je $d(x,0) = t(x)$.

Oznaka 3.5. Koda C je (n, M, d)-koda, če je n dolžina kodnih besed, M število besed v kodi in d razdalja kode.

Recimo, da imamo sporočilo $m = m_0 \dots m_{k-1}$ dolžine k iz abecede Σ^k, ki ga želimo poslati po nezanesljivem kanalu. Naj velja $n \geq k \geq 0$. Sporočilu m priredimo kodno besedo $c = c_0 \dots c_{n-1}$ iz izbrane kode C. Pravimo, da sporočilo *zakodiramo*. Kodno besedo nato pošljemo. Pri prenosu se lahko pojavi napaka $e = e_0 \dots e_{n-1}$ in prejemnik prejme besedo $y = c + e$. Besedo y *dekodira* v tisto kodno besedo $\bar{c} \in \Sigma^n$, pri kateri je razdalja $d(y, \bar{c})$ najmanjša. Temu pravimo *dekodiranje po pravilu najbližjega soseda*. Iz kodne besede $\bar{c}$ prejemnik izračuna sporočilo $\bar{m}$. Če je bila napaka e dovolj majhna, je $c = \bar{c}$ in zato $m = \bar{m}$.

Naj bo $r \in \mathbb{N}$. Če za vsak $c \in C$ velja: $c + e \notin C$ in za vse $e \in \Sigma^n$ velja: $1 \leq t(e) \leq r$, potem pravimo, da koda C *odkrije* r napak. Koda C *popravi* r napak, če je $d(c+e, c) \leq d(c+e, \bar{c})$ za vsak $\bar{c}, c \in C$ in $\bar{c} \neq c$ ter za vsak $e \in \Sigma^n : t(e) \leq r$. S pravilom najbližjega soseda v tem primeru prejeto besedo pravilno dekodiramo.

Kode za popravljanje napak imajo torej nalogo, da odkrijejo in popravijo napake, ki so nastale med pošiljanjem po komunikacijskem kanalu. To smo dosegli tako, da smo sporočilu dodali kontrolne simbole, ki nosijo informacijo za popravljanje napak. V splošnem velja: več simbolov kot dodamo, več

napak lahko kode popravijo. A s tem, ko dodajamo kontrolne simbole, lahko po istem kanalu pošljemo manj sporočil, saj se vsako sporočilo podaljša in prenos sporočila zaseda kanal dalj časa. Cilj je najti ravnovesje med velikostjo poslanega sporočila in dolžino prirejene kodne besede.

Trditev 3.1. *Naj bo C (n,M,d)-koda. Velja, da koda odkrije $d-1$ napak in popravi $\lfloor \frac{d-1}{2} \rfloor$ napak.*

Dokaz. Naj bo $e \in \Sigma^n$ tak da velja $1 \leq t(e) \leq d-1$. Naj bo $c \in C$. Velja, da je $d(c, c+e) = t(c+e-c) = t(e) \leq d-1$. Vidimo, da beseda e ni v kodi C, saj je njena teža manjša od razdalje koda in zato tudi $c+e \notin C$.

Pokažimo še drugi del, da koda popravi $\lfloor \frac{d-1}{2} \rfloor$ napak. Naj bosta $c, \bar{c} \in C$ in $\bar{c} \neq c$. Naj bo $e \in \Sigma^n$ tak, da velja $t(e) \leq \lfloor \frac{d-1}{2} \rfloor$. Potem velja:

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

Pomnožimo obe strani enačbe z 2 in dobimo: $2d(c, c+e) \leq d-1 \leq d(c, \bar{c})-1$. Z uporabo trikotniške neenakosti za c, $\bar{c}$ in $c+e$ dobimo neenakost

$$2d(c, c+e) \leq d(c, c+e) + d(c+e, \bar{c}) - 1.$$

To je enako $d(c, c+e) \leq d(c+e, \bar{c}) - 1$. Od tod sledi, da kod res popravi $\lfloor \frac{d-1}{2} \rfloor$ napak, saj je $d(c, c+e) < d(c+e, \bar{c})$. □

Definirajmo poseben razred bločnih kod, ki se v praksi najpogosteje uporabljajo. To so linearne kode.

Definicija 3.6. Naj bo Σ končen obseg. Koda $C \subseteq \Sigma^n$ je *linearna*, če je vektorski podprostor prostora Σ^n. Torej velja: za vsak $c_1, c_2 \in C$ in $\alpha, \beta \in \Sigma$ je tudi $\alpha c_1 + \beta c_2 \in C$. Dimenzija vektorskega podprostora je *dimenzija* kode.

V nadaljevanju bo končen obseg $\Sigma = GF(q)$. Moč obseg $GF(q)$ je q. Poglejmo si kako to vpliva na število kodnih besed v linearnih kodah.

Trditev 3.2. *Naj bo C linearna (n, M, d)-koda nad končnim obsegom $GF(q)$ dimenzije k. Potem velja, da je $M = q^k$.*

Oznaka 3.7. Linearno (n, q^k, d) kodo nad $GF(q)$ označimo z $[n, k, d]$.

Kot smo prej omenili, z dodajanjem kontrolnih simbolov med kodiranjem dosežemo možnost odkrivanja in popravljanja napak, ki nastanejo med prenosom. Zanima nas razmerje, koliko najmanj simbolov moramo dodati sporočilu, da bo dobljena koda popravila zahtevano število napak. Eno izmed zgornjih mej za število dodanih simbolov, imenujemo *Singletova meja.* Zanimivo je, da jo Reed-Solomonovi kodi, ki jih bomo spoznali v nadaljevanju, dosežejo.

Trditev 3.3. (Singletonova meja) *Naj bo C (n,M,d)-koda nad abecedo Σ s q simboli. Potem velja $M \leq q^{n-d+1}$.*

Dokaz. Vse besede v kodi C so dolžine n in so med seboj različne. V vseh M besedah prečrtamo $d-1$ simbolov na istih mestih. V dobljeni novi kodi, v kateri so besede dolžine $n-d+1$, so vse besede še vedno različne, saj je razdalja kode C enaka d. Vseh besed v Σ^{n-d+1} je q^{n-d+1}, torej velja, da je število vseh besed v kodi C največ q^{n-d+1}. □

Posledica 3.4. *Naj bo C [n,k,d]-koda nad $GF(q)$. Potem velja $d \leq n-k+1$.*

Dokaz. Število besed v kodi linearni kodi C je enako q^k, torej je $q^k \leq q^{n-d+1}$, kar pomeni $d \leq n-k+1$. □

Posledica 3.5. *Linearna [n,k,d]-koda lahko odkrije n-k napak in popravi največ $\lfloor \frac{n-k}{2} \rfloor$ napak.*

Poglejmo si poseben razred linearnih kod, ciklične kode.

Definicija 3.8. Linearna koda C nad $GF(q)$ je *ciklična*, če velja, da za poljubno besedo $c = c_0 c_1 \dots c_{n-1} \in C$, tudi $\hat{c} = c_{n-1} c_0 \dots c_{n-2} \in C$. Besedo $\hat{c}$ imenujemo *ciklični pomik* besede c.

Opomba 3.9. Z $GF(q)[x]$ označimo množico vseh polinomov s koeficienti iz $GF(q)$. Z $GF(q)[x]/(x^n-1)$ bomo označili polinomski kolobar nad $GF(q)$, glej [10].

Kodno besedo $c = c_0c_1 \dots c_{n-1} \in \Sigma^n$ identificiramo s polinom $c(x) = c_0 + c_1x + \dots + c_{n-1}x^{n-1} \in GF(q)[x]/(x^n - 1)$. Kodni besedi $\hat{c}$ pa ustreza polinom

$$\begin{aligned}\hat{c}(x) &= c_{n-1} + c_0x + c_1x^2 + \dots + c_{n-2}x^{n-1} \\ &= xc(x) - c_{n-1}(x^n - 1) \equiv xc(x) \pmod{x^n - 1}.\end{aligned}$$

V kolobarju polinomov $GF(q)[x]$ / $(x^n - 1)$ torej dobimo ciklični pomik $c(x)$ tako, da množimo polinom $c(x)$ z x. Zaradi linearnosti ciklične kode, potem za vsak $c(x) \in C$ velja: $c(x)p(x) \in C$ za vsak polinom $p(x)$. Poglejmo si nekaj pomembnejših lastnosti cikličnih kod.

Trditev 3.6. *Naj bo C ciklična koda dolžine n in dimenzije k nad $GF(q)$. Naj bo $g(x)$ neničelni polinom najmanjše stopnje v C. Potem velja:*

1. *Polinom $g(x)$ generira kodo C, torej*

$$C = \{g(x)m(x); m(x) \in GF(q)[x], \textit{ stopnja m(x) je največ } k\}.$$

2. *Polinom $g(x)$ deli $x^n - 1$.*

3. *Dimenzija kode k= $n-$stopnja polinoma $g(x)$.*

Dokaz. Pokažimo, da polinom $g(x)$ generira kodo C. Izberemo poljubno besedo $p(x) \in C$. Za nek polinom $f(x)$ kodno besedo lahko zapišemo kot $p(x) = f(x)g(x) + r(x)$, kjer je stopnja $f(x)$ največ k in stopnja $r(x)$ manjša od stopnje $g(x)$. Ker je kod C linearen, in ker je $p(x) \in C$ in $g(x) \in C$, mora veljati $r(x) \in C$. Ker je $g(x)$ neničelni polinom najmanjše stopnje v kodi C, sledi $r(x) = 0$. Velja, da je $p(x) = f(x)g(x)$ in je $g(x)$ generator kode.

Za dokaz druge točke zapišimo $x^n - 1 = f(x)g(x) + r(x)$, kjer je stopnja $r(x)$ manjša od stopnje $g(x)$. Enakovredno lahko zapišemo $r(x) = f(x)g(x) + (x^n - 1)$. Dobimo $r(x) = f(x)g(x) \equiv 0 \pmod{x^n - 1}$. Ker velja, da je stopnja $r(x) <$ stopnje $g(x)$, sledi, da je $r(x) = 0$ in polinom $g(x)$ deli $x^n - 1$.

Da dokaz tretje točke si pomagajmo s pomožno trditvijo. Naj bo $g(x) \in C$ in $k = n-$ stopnja $g(x)$. Trdimo, da je množica $B = \{g(x), xg(x), \ldots, x^{k-1}g(x)\}$ baza kode C. Da bo trditev veljala, moramo pokazati, da je B ogrodje in da je množica B linearno neodvisna. Recimo, da je B linearno neodvisna, torej nobenega polinoma iz B ne moremo napisati kot linearno kombinacijo drugih polinomov iz B. Naj velja $\sum_{i=0}^{k-1} \lambda_i \; x_i g(x) = 0$, kar lahko zapišemo kot $g(x) = \sum_{j=0}^{n-k} g_j \; x_j$. Analiziramo koeficiente, ki jih dobimo pri posameznem členu polinoma. Pri x^{n-1} vidimo, da je $\lambda_{k-1} \; g_{n-k} = 0$, torej je $\lambda_{k-1} = 0$, saj $g_{n-k} \neq 0$. Za koeficient pri x^{n-2} dobimo $\lambda_{k-2} \; g_{n-k} + \lambda_{k-1} g_{n-k-1} = 0$. Ker je $\lambda_{k-1} = 0$, velja $\lambda_{k-2} \; g_{n-k} = 0$. Sledi, da je $\lambda_{k-2} = 0$. Postopek ponavljamo in vidimo, da so vsi koeficienti $\lambda_0, \ldots, \lambda_{k-1}$ enaki 0. Torej je B linearno neodvisna.

Dokažimo še, da je množica B ogrodje. Naj bo $p(x) \in C$. Potem velja $p(x) = a(x)g(x)$, kjer je $g(x)$ stopnje k in $a(x)$ stopnje $\leq k-1$. Polinom $a(x)$ lahko zapišemo kot $a(x) = \sum_{i=0}^{k-1} a_i x^i$. Sledi, da je $p(x) = \sum_{i=0}^{k-1} a_i \underbrace{x^i g(x)}_{\in B}$. Ker je $x^i g(x) \in B$, je B res ogrodje. □

Definicija 3.10. Če polinom $g(x)$ generira kodo C ga imenujemo *generatorski polinom* kode C.

Opazimo, da so vse kodne besede v ciklični kodi deljive z generatorskim polinomom.

Poglavje 4

Reed-Solomonove kode

Reed-Solomonove kode (krajše RS kode) spadajo v družino cikličnih BCH-kod. Široko se uporabljajo v napravah za shranjevanje podatkov in drugod, od mobilne, digitalne in satelitske komunikacije, do širokopasovnega interneta, CD, DVD in črtnih kod. Ključni lastnosti, ki izpostavljata RS kode, je dosežena Singletonova meja in učinkoviti algoritmi za dekodiranje. Za algoritem rečemo da je učinkovit, kadar je polinomske časovne zahtevnosti glede na velikost podatkov. Sledimo knjigam [3, 14].

Abeceda, nad katero definiramo RS kode, bo v celotnem poglavju končni obseg $GF(q)$, kjer je $q = 2^r$ za nek $r > 1$. Vpeljimo Reed-Solomonove kode kot linearne ciklične kode.

Trditev 4.1. *Naj bodo $\alpha_1, \alpha_2, \ldots, \alpha_p$ med seboj različni neničelni elementi končnega obsega $GF(q)$. Potem polinom $g(x) = (x - \alpha_1)(x - \alpha_2)\ldots(x - \alpha_p)$ deli $x^{q-1} - 1$.*

Dokaz. Po trditvi 2.1 velja za vsak $\alpha_i^{q-1} = 1$, ker je moč ciklične grupe $GF(q)^*$ enaka $q - 1$. Elementi α_i so torej ničle polinoma $x^{q-1} - 1$. Ker so α_i različni, $(x - \alpha_1)(x - \alpha_2)\ldots(x - \alpha_p)$ deli polinom $x^{q-1} - 1$.

□

Po 2. točki trditve 3.6 dobimo naslednjo posledico.

Posledica 4.2. *Naj bodo $\alpha_1, \alpha_2, \ldots, \alpha_p$ različni neničelni elementi obsega $GF(q)$. Potem polinom $g(x) = (x-\alpha_1)(x-\alpha_2)\ldots(x-\alpha_p)$ generira linearno ciklično kodo dolžine $n = q-1$ nad obsegom $GF(q)$ dimenzije $n-p$.*

Definicija 4.1. Naj bo $n = 2^r - 1$, $0 \leq j < d \leq n$ in naj bo α primitivni element končnega obsega $GF(2^r)$. *Reed-Solomonova koda* RS(n, k) je ciklična linearna koda dolžine n in dimenzije $k = n - d + 1$ nad $GF(2^r)$, generirana z generatorskim polinomom $g(x) = (x - \alpha^j)(x - \alpha^{j+1})\ldots(x - \alpha^{j+d-2})$.

Opomba 4.2. V nadaljevanju naj velja $j = 0$, torej generatorski polinom je enak $g(x) = (x - \alpha^0)(x - \alpha^1)...(x - \alpha^{d-2})$. Tako se uporablja pri kodah QR.

Izrek 4.3. *Naj bo* C *Reed-Solomonova koda $RS(n,k)$ dolžine $n = 2^r - 1$ in dimenzije k. Potem je razdalja kode enaka $n - k + 1$.*

Dokaz izreka najdemo [3, stran 144]. S primerjavo trditve 3.3 in izreka 4.3 vidimo, da Reed-Solomonove kode dosežejo Singletonovo mejo. Tako kode popravijo največje število napak glede na število simbolov, ki jih dodamo sporočilu. Kodo $RS(n,k)$ lahko zapišemo tudi kot ciklično $[n, k, n-k+1]$ kodo nad abecedo $GF(2^r)$, kjer velja $0 \leq k \leq n$ in $n = 2^r - 1$. Trditev 3.2 nam pove, da je število besed RS kode 2^{rk}. Iz trditve 3.1 dobimo naslednjo trditev.

Trditev 4.4. *Koda $RS(n,k)$ odkrije $n-k$ napak in odpravi $\lfloor (n-k)/2 \rfloor$ napak.*

Sporočilu dolžine k priredimo kodno besedo dolžine n. Število dodanih kontrolnih simbolov je $n-k$. Iz trditve 4.4 vidimo, da RS koda odkrije toliko napak, kot ji dodamo kontrolnih simbolov in popravi polovico toliko napak, kot smo sporočilu dodali kontrolnih simbolov. Poglejmo si nekaj zgledov RS kod.

Zgled 4.1. Naj bo $n = 2^4 - 1 = 15$. Sestavimo generatorski polinom za $RS(n,k)$, ki lahko popravi do 3 napake nad obsegom $GF(2^4)$, generiranim z nerazcepnim polinomom $x^4 + x + 1$. Da lahko popravi 3 napake, mora biti

razdalja $d = 6$, kar določa tudi stopnjo generatorskega polinoma. Naj bo α primitivni element obsega. Pri računanju z elementi obsega, kot je opisano v zgledu 2.4, si pomagamo s tabelo 2.1 na strani 9. Izračunajmo generatorski polinom za kodo $RS(15, 9)$.

$$\begin{aligned} g(x) &= (x - \alpha^0)(x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)(x - \alpha^5) \\ &= x^6 + \alpha^9 x^5 + \alpha^{12} x^4 + \alpha x^3 + \alpha^2 x^2 + \alpha^4 x + 1. \end{aligned}$$

Zgled 4.2. Naj bo dolžina kode $n = 255$ nad obsegom $GF(2^8)$, generiranim z nerazcepnim polinomom $x^8 + x^4 + x^3 + x^2 + 1$. Naj bo α primitivni element obsega. Število napak, ki jih lahko popravi, naj bo 5, potem je stopnja generatorskega polinoma enaka 10. V tabeli A.1 v dodatku A, so zapisane vse potence primitivnega elementa v obsegu $GF(2^8)$, s pomočjo katerih izračunamo generatorski polinom:

$$\begin{aligned} g(x) = x^{10} &+ \alpha^{251} x^9 + \alpha^{67} x^8 + \alpha^{46} x^7 + \alpha^{61} x^6 + \alpha^{118} x^5 \\ &+ \alpha^{70} x^4 + \alpha^{64} x^3 + \alpha^{94} x^2 + \alpha^{32} x + \alpha^{45}. \end{aligned}$$

Dobimo kodo $RS(255, 245)$ z $2^{8 \cdot 245}$ kodnimi besedami.

RS kode so dobre pri popravljanju napak, ki se pojavljajo v skupkih. Naj bo p praštevilo. Vsak kodni simbol je element obsega $GF(p^r)$ in je zapisan z r zaporednimi biti. Če pride do napake na r zaporednih bitih, je to zgolj en ali največ dva simbola in zato samo ena ali dve napaki. Zanesljivost prenosa je lastnost, da po dekodiranju kodne besede ni napake v sporočilu. Ker RS kode dosežejo Singletovo mejo, tako popravijo največje število napak glede na število dodanih kontrolnih simbolov. Uvrščamo jih med zanesljivejše kode. A ne smemo jih kar slepo uporabljati. Kode za popravljanje napak je potrebno izbrati glede na komunikacijski kanal po katerem pošiljamo sporočilo in vrsto napak, ki se pojavljo. Če so napake enakomerno porazdeljene, neodvisne in razpršene, so konvolucijske, turbo in BCH kode bojša izbira. Večina sistemov zakodira podatke dvojno. Najprej jih zakodirajo z RS kodami, nato pa s konvolucijskimi kodami in tako lahko popravijo tudi do 4000 zaporednih napačnih bitov.

V naslednjem razdelku si pogledamo kodiranje sporočila z uporabo RS kod.

4.1 Sistematično kodiranje

V splošnem sporočilo kodiramo tako, da ga pomnožimo z generatorskim polinomom. Vsaka kodna beseda je namreč večkratnik generatorskega polinoma $g(x)$. Iz tako zakodirane kodne besede na prvi pogled ne dobimo informacije o prvotnem sporočilu. Sporočilo lahko zakodiramo tudi tako, da lahko iz kodne besede direktno preberemo izvirno sporočilo. Tako kodiranje imenujemo *sistematično kodiranje*. Poglejmo si postopek.

Najprej si izberemo obseg $GF(2^r)$ in naj bo α primitivni element obsega. Imamo sporočilo $m = (m_{k-1}, \ldots, m_1, m_0)$ dolžine k, kjer je $m_i \in GF(2^r)$. Sporočilo zapišemo v obliki polinoma. Sporočilo je zaporedje koeficientov za polinom. Prva beseda je koeficient pri členu z najvišjo stopnjo, zadnja beseda je prosti člen polinoma sporočila:

$$m(x) = m_{k-1}x^{k-1} + \ldots + m_2x^2 + m_1x + m_0.$$

Recimo, da želimo popraviti $t/2$ napak. Naj bo $n = k + t$. Polinom sporočila najprej pomnožimo z x^t in ga nato delimo z generatorskim polinomom $g(x) = (x - \alpha^0)(x - \alpha^1)...(x - \alpha^{t-1})$. Z množenjem z x^t smo naredili prostor za t kontrolnih simbolov, ki jih dodamo sporočilu na koncu. S tem tudi zagotovimo, da eksponent sporočilnega polinoma med deljenjem ne postane premajhen in res dobimo ostanek dolžine t. Ostanek pri deljenju $m(x)x^t$ z $g(x)$ je enolično določen. Označimo ga z $b(x)$,

$$b(x) = b_{t-1}x^{t-1} + \ldots + b_2x^2 + b_1x + b_0.$$

Ker je stopnja ostanka $b(x)$ manjša od t, odštevanje ostanka sporočilu $m(x)\cdot x^t$ ne vpliva na noben koeficient sporočilnega polinoma, tako da je $m(x)x^t - b(x)$ kot izvirno sporočilo z dodanimi kontrolnimi simboli. Koeficienti izračunanega polinoma ostanka $b(x)$ so naši kontrolni simboli. Kodna beseda je torej oblike

$$c = (m_{k-1}, \ldots, m_1, m_0, b_{t-1}, \ldots, b_1, b_0). \tag{4.1}$$

Zapišimo opisani kodirni postopek:

$$\begin{aligned} b(x) &= m(x)x^t \mod g(x), \\ c(x) &= m(x)x^t - b(x). \end{aligned} \tag{4.2}$$

Opomba 4.3. Ker računamo v obsegu s karakteristiko 2, je seštevanje enako odštevanju, torej je $c(x) = m(x)x^t + b(x)$.

Zgled 4.3. Sporočilo $m = (\alpha^2, \alpha^{12}, \alpha^9, \alpha^4, \alpha^3, \alpha^5, 1, \alpha, \alpha^8)$ zakodirajmo s kodo $RS(15, 9)$ iz zgleda 4.1. Ker je $t = n - k$ enako 6, polinom sporočila $m(x) = \alpha^2x^8 + \alpha^{12}x^7 + \alpha^9x^6 + \alpha^4x^5 + \alpha^3x^4 + \alpha^5x^3 + x^2 + \alpha x + \alpha^8$ pomnožimo z x^6 in delimo z generatorskim polinomom $g(x)$ iz zgleda 4.1. Dobimo polinom ostanka $b(x) = \alpha x^5 + \alpha^5x^4 + \alpha^3x^3 + \alpha^{14}x^2 + \alpha^2x + \alpha^8$, katerega koeficienti predstavljajo kontrolne simbole za odkrivanje in popravljanje napak. Sistematično zakodiran polinom iz katerega je prvih k koeficientov naše sporočilo je:

$$\begin{aligned} c(x) = \alpha^2x^{14} + \alpha^{12}x^{13} + \alpha^9x^{12} + \alpha^4x^{11} + \alpha^3x^{10} + \alpha^5x^9 + x^8 + \alpha x^7 + \alpha^8x^6 \\ + \alpha x^5 + \alpha^5x^4 + \alpha^3x^3 + \alpha^{14}x^2 + \alpha^2x + \alpha^8. \end{aligned}$$

Kodna beseda je $c = (\alpha^2, \alpha^{12}, \alpha^9, \alpha^4, \alpha^3, \alpha^5, 1, \alpha, \alpha^8, \alpha, \alpha^5, \alpha^3, \alpha^{14}, \alpha^2, \alpha^8)$.

4.2 Dekodiranje

Dekodiranje prejetega sporočila je v splošnem je NP-težek problem [4]. Za Reed-Solomonove kode pa obstajajo učinkoviti dekodirni algoritmi. Poznamo več učinkovitih algoritmov za dekodiranje, npr. Berlekamp-Masseyev algoritem, Guruswami-Sudanov algoritem, Evklidov algoritem in algoritem Peterson-Gorenstein-Zierler (PGZ) [21]. Algoritem PGZ podrobneje opišemo v nadaljevanju. Uporablja se za dekodiranje v kodah QR [5, Priloga B] in je enostaven za razumevanje in implementacijo [5, Priloga B]. Naj bo n dolžina kodiranega sporočila. Potem je časovna zahtevnosti algoritma PGZ $O(n^3)$.

Naj bo α primitiven element obsega $GF(q)$ in $n = q-1$. Velja, da je razdalja $RS(n,k)$ kode enaka $d = n-k+1$, kjer je $1 \leq k < n$. Poglejmo si postopek dekodiranja $RS(n,k)$ kode, generirane z $g(x) = (x-\alpha^0)(x-\alpha)\dots(x-\alpha^{d-2})$. Naj bo

$$r(x) = c(x) + e(x)$$

polinom prejete kodne besede, kjer je $c(x) = m(x)x^t - b(x)$ polinom poslane kodne besede (glej razdelek 4.1), $m(x)$ polinom sporočila in $e(x)$ *polinom napake*. Vemo, da so vsi polinomi kodnih besed iz $RS(n,k)$ večkratniki generatorskega polinoma $g(x)$ (trditev 3.6). Ta lastnost nam omogoča, da lahko hitro dekodiramo kodno besedo, če pri prenosu ni prišlo do napake. Pravilnost prenosa preverimo tako, da $r(x)$ delimo z generatorskim polinomom. Če je ostanek enak nič, je tudi polinom napake enak nič. Tako dobimo polinom sporočila $m(x)$ kar z deljenjem polinoma $r(x)$ s polinomom $g(x)$. Poglejmo si zgled dekodiranja, ko je polinom napake enak nič.

Zgled 4.4. Naj bo $RS(7,5)$ nad obsegom $GF(2^3)$, ki je konstruiran s polinomom x^3+x+1. Generatorski polinom za kodo je $g(x) = (x-\alpha^0)(x-\alpha^1) = x^2 + \alpha^3 x + \alpha$. Recimo, da smo prejeli sistematično kodirano sporočilo $r = (\alpha^3,\ 0,\ 0,\ \alpha,\ 1,\ \alpha,\ \alpha)$. Dolžina izvornega sporočila je $k = 5$. Če ni prišlo do napake med prenosom, je naše sporočilo kar prvih 5 členov, $m = (\alpha^3,\ 0,\ 0,\ \alpha, 1)$. Z deljenjem polinoma prejetega sporočila $r(x) = \alpha^3x^6 + \alpha x^3 + x^2 + \alpha^6 x^1 + \alpha^5$ z generatorskim polinomom $g(x)$ dobimo 0, zato gornja enačba velja in je prejeto sporočilo enako izvornemu.

V primeru, da je med prenosom prišlo do napake, je dekodiranje zahtevnejše. Opišimo dekodirni algoritem PGZ, ki dekodiranje prevede na reševanje posebega sistema enačb. Predpostavimo, da pri prenosu ni prišlo do več kot $t \leq \lfloor \frac{d-1}{2} \rfloor$ napak. Naj $0 \leq p_1, \dots, p_t \leq n-1$ označujejo mesta napak v kodni besedi. Števila $\alpha^{p_1}, \dots \alpha^{p_t}$ imenujmo *lokatorji napak*. *Velikost napake* na mestu lokatorja označimo z e_{p_i}. Polinom napake tako lahko zapišemo kot

$$e(x) = \sum_{i=1}^{t} e_{p_i} x^{p_i}.$$

V nadaljevanju označimo lokatorje napak z $X_i = \alpha^{p_i}$ in velikost napake z $Y_i = e_{p_i}$. Definirajmo *sindrome* S_j za $j = 1, \ldots, d-1$. Ker so $\alpha, \alpha^2, \ldots, \alpha^{d-1}$ ničle polinoma $g(x)$, so tudi ničle polinoma $c(x)$ in velja

$$S_j = r(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j) = \sum_{i=1}^{t} e_{p_i}(\alpha^j)^{p_i} = \sum_{i=1}^{t} Y_i X_i^j.$$

Dobljeni sistem enačb zapišimo v matrični obliki:

$$\begin{pmatrix} X_1^1 & X_2^1 & \ldots & X_t^1 \\ X_1^2 & X_2^2 & \ldots & X_t^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{d-1} & X_2^{d-1} & \ldots & X_t^{d-1} \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_t \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{d-1} \end{pmatrix}. \tag{4.3}$$

Sedaj smo problem prevedli na sistem $d-1$ nelinearnih enačb z $d-1$ neznankami $X_1, \ldots, X_t, Y_1, \ldots, Y_t$. Ker sistem ni linearen, je zahteven za računanje. Obstaja več različnih načinov za reševanje. Ogledali si bomo enega izmed njih [21]. Cilj je določiti vrednosti X_i za $i = 1, \ldots, t$, da dobimo linearni sistem enačb, nato pa iz sistema (4.3) določimo še velikosti napak Y_i. Poglejmo si postopek.

Definirajmo *polinom lokatorjev napak*

$$\Lambda(x) = \prod_{i=1}^{t}(1 - xX_i) = 1 + \lambda_1 x + \ldots + \lambda_t x^t.$$

Vidimo, da so ničle polinoma lokatorjev napak X_i^{-1} ravno inverzne vrednosti lokatorjev polinoma, zato za $i = 1, \cdots, t$ velja

$$\Lambda(X_i^{-1}) = 1 + \lambda_1 X_i^{-1} + \ldots + \lambda_t X_i^{-t} = 0.$$

Pomnožimo obe strani z $Y_i X_i^{j+t}$ in dobimo

$$Y_i X_i^{j+t}\Lambda(X_i^{-1}) = Y_i X_i^{j+t} + \lambda_1 Y_i X_i^{j+t-1} + \ldots + \lambda_t Y_i X_i^j = 0. \tag{4.4}$$

Za $i = 1, \ldots, t$ seštejemo enačbe (4.4)

$$\sum_{i=1}^{t}(Y_i X_i^{j+t}) + \lambda_1 \sum_{i=1}^{t}(Y_i X_i^{j+t-1}) + \ldots + \lambda_t \sum_{i=1}^{t}(Y_i X_i^j) = 0. \tag{4.5}$$

Za $j = 1, \ldots, t$ združimo sistem (4.3) in gornjo enačbo (4.5) in dobimo sistem linearnih enačb za $\lambda_i, i = 1, \ldots, t$

$$S_j\lambda_t + S_{j+1}\lambda_{t-1} + \ldots + S_{j+t-1}\lambda_1 + S_{j+t} = 0,$$

ki ga matrično zapišemo kot

$$\begin{pmatrix} S_1 & S_2 & \ldots & S_t \\ S_2 & S_3 & \ldots & S_{t+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_t & S_{t+1} & \ldots & S_{2t-1} \end{pmatrix} \begin{pmatrix} \lambda_t \\ \lambda_{t-1} \\ \vdots \\ \lambda_1 \end{pmatrix} = \begin{pmatrix} -S_{t+1} \\ -S_{t+2} \\ \vdots \\ -S_{2t} \end{pmatrix}. \tag{4.6}$$

Če je leva matrika obrnljiva, lahko določimo velikost napake λ_k.

Največkrat ni znano, koliko napak se je zgodilo med prenosom. Zato za reševanje sistema (4.6) za t vzamemo kar zgornjo mejo $\lfloor \frac{d-1}{2} \rfloor$. Izkaže se, da je rang matrike kar enak številu napak, glej [3, stran 149]. Ko poznamo število napak, izračunamo še koeficiente $\lambda_1, \ldots, \lambda_t$ polinoma lokatorjev napak. Za izračun lokatorjev napak moramo najprej poiskati njegove ničle in nato inverze, ki določijo mesto napake. Izračunamo še velikost napak z reševanjem linearnega sistema (4.3). Izračunan polinom napak $e(x)$, odštejemo od polinoma prejete kodne besede, da dobimo polinom poslane kodne besede

$$c(x) = r(x) - e(x).$$

Zgled 4.5. Oglejmo si primer $RS(15, 11)$ kode nad obsegom $GF(2^4)$, generiranim z nerazcepnim polinomom $x^4 + x + 1$. Razdalja kode je enaka $d = 5$, tako da koda popravi do dve napaki. Stopnja $g(x)$ je $n - k = 4$. Izračunajmo generatorski polinom $g(x) = (x - \alpha^0)(x - \alpha^1)(x - \alpha^2)(x - \alpha^3) = x^4 + \alpha^{12}x^3 + \alpha^4x^2 + x + \alpha^6$. Sistematično kodiramo sporočilo $m = (0, 0, 1, 1, 1, \alpha^{14}, 0, 0, \alpha^{10}, 0, 0)$ in ji priredimo polinom kodne besede $c(x) = m(x)x^4 - b(x) = x^{12} + x^{11} + x^{10} + \alpha^{14}x^9 + \alpha^{10}x^6 + \alpha^3x^3 + \alpha^5x^2 + \alpha^6x + \alpha^{13}$ oziroma $c = (0, 0, 1, 1, 1, \alpha^{14}, 0, 0, \alpha^{10}, 0, 0, \alpha^3, \alpha^5, \alpha^6, \alpha^{13})$.

Poglejmo si še dekodiranje. Recimo, da smo prejeli besedo r s prirejenim polinomom $r(x) = x^{12} + x^{11} + x^{10} + \alpha^{14}x^9 + \alpha x^8 + \alpha^4x^7 + \alpha^{10}x^6 + \alpha^3x^3 +$

$\alpha^5x^2 + \alpha^6x + \alpha^{13}$. Polinom $r(x)$ ni deljiv z $g(x)$, torej je med prenosom prišlo do napake. Izračunajmo sindrome $S_i = s(\alpha^i)$ za $i = 1, \ldots, 4$ in dobimo naslednje vredosti

$$\begin{aligned} S_1 &= r(\alpha^0) = \alpha^0, \\ S_2 &= r(\alpha^1) = \alpha^2, \\ S_3 &= r(\alpha^2) = \alpha^6, \\ S_4 &= r(\alpha^3) = 0. \end{aligned}$$

Sestavimo matriko iz sistema (4.3) in rešimo sistem dveh enačb z dvema neznankama

$$\begin{pmatrix} 1 & \alpha^2 \\ \alpha^2 & \alpha^6 \end{pmatrix} \begin{pmatrix} \lambda_2 \\ \lambda_1 \end{pmatrix} = \begin{pmatrix} \alpha^6 \\ 0 \end{pmatrix}, \tag{4.7}$$

ki nam da rešitev $\lambda_1 = \alpha^{11}$ in $\lambda_2 = \alpha^{15} = 1$. Dobimo polinom lokatorjev napak $\Lambda(x) = x^2 + \alpha^{11}x + 1$. Z računanjem njegovih vrednosti v vseh elementih obsega $GF(2^4)$ preverimo, da sta njegovi ničli α^7 in α^8. Njuna inverza α^8 in α^7 povesta, da sta napaki pri prejeti besedi na sedmem in osmem mestu. Izračunati moramo še velikost teh napak. S sistemom (4.3), dobimo

$$\begin{aligned} 1 &= \lambda_1\alpha^8 + \lambda_2\alpha^7 \\ \alpha^2 &= \lambda_1(\alpha^8)^2 + \lambda_2(\alpha^7)^2. \end{aligned}$$

Velikost napak je $\lambda_1 = \alpha$ in $\lambda_2 = \alpha^4$. Polinom napake $e(x) = \alpha x^8 + \alpha^4x^7$ odštejemo od $r(x)$ in dobimo polinom poslane kodne besede $c(x) = x^{12} + x^{11} + x^{10} + \alpha^{14}x^9 + \alpha^{10}x^6 + \alpha^3x^3 + \alpha^5x^2 + \alpha^6x + \alpha^{13}$.

Poglavje 5

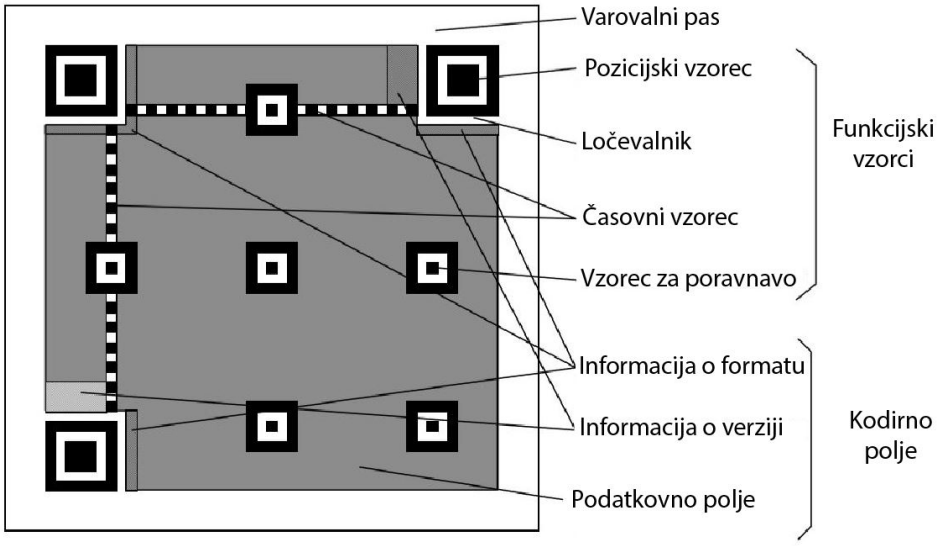
Kode QR

Kode QR so dvodimenzionalne črtne kode. So strojno berljiva predstavitev podatkov. Specifikacija kode QR je bila leta 2000 potrjena kot mednarodni standard ISO/IEC 18004. V našem delu se sklicujemo na specifikacijo simbologije kode QR 2005, opisane v prosto dostopnem standardu [5]. Trenutno je v veljavi novejši standard [6], dopolnjen leta 2015 in je plačljiv. Glavne lastnosti kot so karakteristika simbolov, metode kodiranja podatkov in dimenzijske značilnosti ostajajo enaki.

V tem poglavju podrobneje predstavimo zgradbo simbola (matrike) kode QR in opišemo pomen sestavnih delov. Skozi primer pogledamo, kako se podatki zapišejo v simbol QR. Najprej podatke ustrezno pretvorimo v dvojiški zapis, nato pa zakodiramo z ustreznim kodirnim algoritmom. Za kodiranje podatkov se uporabljajo Reed-Solomonove kode. Slike in tabele so iz standarda [5].

5.1 Zgradba simbola kode QR

Simbol je sestavljen iz črno-belih kvadratnih modulov, razporejenih v kvadratno matriko. Črni modul prestavlja število 1, beli pa število 0. En modul predstavlja en bit podatka. Simbol je zgrajen iz *funkcijskih vzorcev* in *kodirnega polja*. Funkcijski vzorci tvorijo okvir kode in ne kodirajo sporočila.



Slika 5.1: Zgradba simbola kode QR.

Čitalcu črtne kode (v nadaljevanju: čitalec) omogočijo, da kodo prepozna in prebere. Na sliki 5.1 so označeni sestavni deli simbola. Vsak ima svojo funkcijo, ki je podrobneje opisana v nadaljevanju.

Pozicijski vzorec (angl. *Finder pattern*). Trije pozicijski vzorci so ključni za prepoznavanje kode QR. Izdelani so tako, da je verjetnost, da bi se vzorec ponovil v podatkovnem delu, majhna. Razporejeni so v treh vogalih kode in definirajo položaj, velikost in rotacijsko usmerjenost simbola. Omogočajo hitro branje iz vseh smeri (360°), ne glede na lego kode ali čitalca črtnih kod.

Vzorec za poravnavo (angl. *Alignment patterns*). Koda QR se popači, ko je pritrjena na ukrivljeno površino ali je čitalec nagnjen. V primeru zmernega izkrivljenja vzorec za poravnavo omogoča berljivost simbola. Število vzorcev je odvisno od verzije simbola. Verzija 1 nima vzorca za poravnavo. Z večanjem verzije se poveča število vzorcev. Položaji in število vzorcev so v naprej definirani in si jih lahko ogledamo v [5, priloga E].

Časovni vzorec (angl. *Timing pattern*). Izmenjujoče se zaporedje črnobelih modulov pomaga čitalcu določiti vrstice in stolpce v matriki. Vzorec je razporejen vertikalno in horizontalno med pozicijskimi vzorci.

Varovalni pas (angl. *Quiet zone*). Prazno območje širine štirih modulov okrog simbola. Kodo loči od okolice in pripomore k prepoznavnosti kode QR.

Ločevalnik (angl. *Separator*). Funkcijski vzorec, ki ločuje pozicijske vzorce od podatkovnega dela in izboljša njihovo prepoznavnost. Ima širino enega modula.

Informacija o formatu. Kodirana informacija, ki vsebuje podatke o značilnosti simbola, bistvenih za dekodiranje sporočila. Vsebuje podatke o izbrani ravni popravljanja napak in podatkovni maski. Več o tem v razdelkih 5.2.1 in 5.3.4.

Informacija o verziji. Kodirana informacija, ki vsebuje podatek o verziji simbola.

Podatkovno polje. Osrednji del podatkovnega polja je zakodirano sporočilo z dodanimi kontolnimi simboli za popravljanje napak. Podatki so kodirani v binarnem številskem sistemu, kjer je 1 zapisana s črnim in 0 z belim modulom. Zakodirana je tudi vrsta podatkov in dolžina besedila, kar olajša branje kode QR. Postopek kodiranja si pogledamo v nadaljevanju.

Informacija o verziji in o formatu je v simbolu zapisana dvakrat, saj je bistvena za dekodiranje simbola.

5.2 Lastnosti

V podatkovno polje lahko zapišemo numerične, alfanumerične, bajtne ali kanji znake. Najmanjša koda QR je sestavljena iz 21 stolpcev in vrstic (mo-

dulov) – verzija 1, največja pa iz 177 x 177 modulov – verzija 40. Zaporedni verziji se med seboj razlikujeta za 4 module. Verzije označimo v obliki npr. 1-M, kjer število identificira verzijo, črka pa raven popravljanja napak. V največji simbol lahko zakodiramo do

- 7089 numeričnih znakov,
- 4296 alfanumeričnih znakov,
- 2953 bajtnih znakov ali
- 1817 kanji znakov.

5.2.1 Raven popravljanja napak

Sporočilo, ki je zapisano v podatkovnem polju kode QR, je zakodirano. S kodiranjem – kot smo opisali v razdelku 4.1 – se sporočilu dodajo kontrolni simboli za odkrivanje in popravljanje napak. Za popravljanje napak se uporabljajo Reed-Solomonove kode. Koliko napak lahko popravimo, je odvisno od števila dodanih kontrolnih simbolov. Ceno zaščite, ki nam jo kontrolni simboli nudijo „plačamo“ s prostorom. Ker je velikost kode QR v naprej določena, se z večjo zaščito zmanjša količina podatkov, ki jih lahko shranimo znotraj simbola. Če je koda QR premajhna, moramo izbrati večjo verzijo.

V kodi QR so na voljo štiri *ravni popravljanja napak*, ki omogočajo od 7% do 30% obnovljivost poškodovanih podatkov. Označimo jih z L, M, Q, H v naraščujočem vrstnem redu po zmogljivosti popravljanja napak (tabela 5.1). Odvisno od okolja uporabe kode QR izberemo ustrezno raven. Če bo koda QR uporabljena v digitalni obliki: ne bo ne mokra, ne poškodovana, je raven popravljanja napak lahko manjša. Ravni L in M sta primerni za splošno tržno uporabo, ravni Q in H pa za industrijo. Tabela 5.7, ki jo najdemo na koncu poglavja (stran 46), prikazuje dolžino sporočila, ki jo lahko zapišemo v kodo QR. Dolžina sporočila je odvisna od vrste vhodnih podatkov in ravni popravljanja napak. Celotna tabela je na voljo v standardu [5, stran 33]. Poglejmo si primer.

Zgled 5.1. Sporočilo PIKA NOGAVICKA je dolgo 14 alfanumeričnih znakov. V tabeli 5.7 vidimo, da ga lahko zakodiramo v kodo QR verzije 1-Q. Če želimo večjo zaščito, moramo izbrati večjo verzijo 2-H.

Tabela 5.1: Koliko podatkov lahko obnovimo glede na raven popravljanja napak.

Raven L	7%
Raven M	15%
Raven Q	25%
Raven H	30%

5.3 Ustvarjanje kode QR

Sporočilo, ki ga lahko zapišemo v kodo QR, je lahko numerično, alfanumerično, bajtno ali iz kanji znakov. Najprej ga ustrezno pretvorimo v zaporedje 8 bitnih besed in zakodiramo z Reed-Solomonovimi kodami. Generirano kodno besedo vložimo na ustrezno mesto v kodi QR in jo zaščitimo s podatkovno masko. Tako dosežemo enakomerno porazdelitev črnih in belih modulov v simbolu kode QR. Na ustrezno mesto zapišemo še informacije, ki določajo lastnosti simbola.

5.3.1 Analiza vhodnih podatkov/znakov

Glede na vrsto vhodnih podatkov izberemo ustrezen način pretvarjanja znakov sporočila v binarni številski sistem. V tabeli 5.2 vidimo, da vsak način potrebuje različno število bitov za predstavitev enega znaka. S 13 biti lahko zakodiramo 1 kanji znak ali skoraj 4 numerične. Ker se nabori znakov vsakega načina lahko prekrivajo, npr. številski znaki se lahko kodirajo v numeričnem, alfanumeričnem ali bajtnem načinu, je potrebno izbrati najprimernejši način za kodiranje podatkovnih znakov, ki se pojavljajo v več kot enem načinu.

Cilj je zapisati čim več podatkov na čim manjšem prostoru. Izbrani način označimo s 4 bitnim *indikatorjem načina* (tabela 5.3). Nato izberemo želeno raven popravljanja napak in določimo verzijo simbola. Če je ne izberemo, se določi najmanjša glede na vrsto in število podatkov (tabela 5.7). V nadaljevanju si poglejmo različne načine pretvorbe vhodnih podatkov.

Tabela 5.2: Načini pretvorbe.

	Max. znakov	Bit/znak	Možni znaki
Numerični	7089	$3\frac{1}{3}$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Alfanumerični	4296	$5\frac{1}{2}$	0-9, A-Z, presledek, $, %, *, +, -, ., /, :
Bajtni	2953	8	standard ISO/IEC 8859-1 [5, stran 28]
Kanji znaki	1817	13	JIS X 0208 [5, stran 92]

Tabela 5.3: Indikator načina pretvorbe.

Način	Indikator načina
Numerični	0001
Alfanumerični	0010
Bajtni	0100
Kanji	1000

Pretvorba numeričnega načina

Vhodne podatke razdelimo na trimestna števila in njihovo vrednost zapišemo v 10 bitnem številu. Če število znakov ni večkratnik števila 3, se končni znak ali dva pretvorita v 4 ali 7 bitov.

Zgled 5.2. Pretvorimo sporočilo 01234: $012 \rightarrow 0000001100$ in $34 \rightarrow 0100010$. Pretvorjeno sporočilo je 0000001100 0100010.

Pretvorba alfanumeričnega načina

Vhodne podatke razdelimo v pare znakov. Glede na tabelo 5.6 znaku priredimo številsko vrednost. Število, ki pripada prvemu znaku v paru, pomnožimo s 45 in prištejemo število, ki pripada drugemu znaku. Vsoto pretvorimo v 11 bitno število. Postopek ponovimo za vsak par. Če število vhodnih podatkov ni večkratnik števila 2, vrednost končnega znaka pretvorimo s 6 biti.

Zgled 5.3. Pretvorimo sporočilo ABC:

$AB : 10 \cdot 45 + 11 = 461 \rightarrow 00111001101$; C: $12 \rightarrow 000110$. Pretvorjeno sporočilo je 00111001101 000110.

Pretvorba bajtnega načina

Vhodnim podatkom priredimo bajtno vredost s pomočjo ASCII tabele, ki jo najdemo v [5, stran 28] in jo pretvorimo v bitni zapis.

Pretvorba kanji načina

Kanji znaki so pretvorjeni v skladu s sistemom Shift JIS, ki temelji na japonskem standardu JIS X 0208 [5, stran 92]. Kanji znak pretvorimo s 13 biti. Več o pretvorbi kanji načina, glej [5, stran 29].

Na začetek pretvorjenega bitnega sporočila dodamo indikator načina in *števec znakov*, ki nosi informacijo o dolžini vhodnega podatka (sporočila). S koliko biti zapišemo števec, je odvisno od verzije simbola in vrste podatkov. V večje verzije zapisujemo več podatkov, zato števec zapišemo z več biti (tabela 5.4).

Dobljeni bitni niz razdelimo v 8 bitna števila, ki jih imenujemo *podatkovni simboli*, in ga zaključimo z nizom štirih ničel imenovanim *terminator*. Za vsako verzijo je v odvisnosti od izbrane ravni popravljanja napak, določeno

Tabela 5.4: Število bitov za zapis števca podatkov glede na način kodiranja.

Verzija	Numerični	Alfanumerični	Bajtni	Kanji
1-9	10	9	8	8
10-26	12	11	16	10
27-40	14	13	16	12

število podatkovnih simbolov. V primeru, da smo s sporočilom zapolnili zmogljivost simbola, se terminator skrajša ali izpusti. Sicer:

1) če dolžina binarnega sporočila ni deljiva z 8, ji dodamo potrebno število ničelnih zapolnjevalnih bitov (angl. *padding bits*),

2) nato izmenično dodajamo zapolnjevalni besedi 11101100 in 00010001, dokler se prostor ne zapolni.

Poglejmo si na zgledu.

Zgled 5.4. Pretvorimo sporočilo: PIKA NOGAVICKA. Lahko bi pretvarjali v alfanumeričnem ali bajtnem načinu. Ker težimo k temu, da porabimo čim manj prostora, izberemo alfanumeričnega. Izberemo verzijo 1-M. Znake razdelimo v pare in jim priredimo številsko vrednost glede na tabelo 5.6: P $\rightarrow$ 25, I $\rightarrow$ 18. Vrednost prvega znaka pomnožimo s 45, drugega prištejemo. Dobimo $25 \cdot 45 + 18 = 1143$. Vsoto pretvorimo v binarno število (pretvarjamo tudi presledek):

$$\begin{aligned} PI &\rightarrow 10001110111, \\ KA &\rightarrow 01110001110, \\ \_N &\rightarrow 11001101011, \\ OG &\rightarrow 10001001000, \\ AV &\rightarrow 00111100001, \\ IC &\rightarrow 01100110110, \\ KA &\rightarrow 01110001110. \end{aligned}$$

Združimo pare in dobimo:

10001110111 01110001110 11001101011 10001001000 00111100001
00111100001 01100110110 01110001110.

Bitnemu sporočilu na začetku dodamo indikator za alfanumerični način 0010 in števec podatkov 14 $\rightarrow$ 000001110. Na koncu dodamo terminator 0000. Dobimo:

0010 000001110 10001110111 01110001110 11001101011
10001001000 00111100001 01100110110 01110001110 0000.

Zaporedje razdelimo v 8 bitne podatkovne simbole. Na koncu dodamo bite za zapolnjevanje (prikazani so podčrtano) in dobimo zaporedje:

00100000 01110100 01110111 01110001 11011001 10101110 00100100
00011110 00010110 01101100 11100011 10000000.

Skupno število podatkovnih simbolov, ki jih lahko zapišemo v izbrano verzijo 1-M, je 26. Od tega je 16 podatkovnih simbolov in 10 kontrolnih simolov za popravljanje napak. V našem primeru imamo že 12 podatkovnih simbolov. Da zapolnimo kapaciteto kode QR, dodamo še 4 zapolnjevalne simbole (prikazani so podčrtano). Dobimo zaporedje:

00100000 01110100 01110111 01110001 11011001 10101110 00100100
00011110 00010110 01101100 11100011 10000000 11101100 00010001
11101100 00010001.

Pretvorjeno sporočilo zapišimo še v desetiškem sistemu: 32, 166, 119, 113, 217, 174, 36, 30, 22, 108, 227, 128, 236, 17, 236, 17.

Če sporočilo vsebuje več vrst podatkov, lahko uporabimo več načinov pretvarjanja v enem simbolu. Tabela 5.5 prikazuje zgradbo takšnega pretvorjenega sporočila. Najpogosteje se izbere mešani način, kadar pretvarjamo sporočilo s kanji znaki, saj bi zapis v drugem načinu zahteval preveč prostora. Več o optimizaciji zapisa je dostopno v [5, priloga J].

Tabela 5.5: Zgradba sporočila pri pretvarjanju z n načini pretvarjanja.

Del 1			...	Del n			Terminator
Indikator načina 1	Števec znakov	Podatki	...	Indikator načina n	Števec znakov	Podatki	

Tabela 5.6: Alfanumerične vrednosti.

Znak	Vred.	Znak	Vred.	Znak	Vred.	Znak	Vred.	Znak	Vred.
0	0	9	9	I	18	R	27	presledek	36
1	1	A	10	J	19	S	28	$	37
2	2	B	11	K	20	T	29	%	38
3	3	C	12	L	21	U	30	*	39
4	4	D	13	M	22	V	31	+	40
5	5	E	14	N	23	W	32	-	41
6	6	F	15	O	24	X	33	.	42
7	7	G	16	P	25	Y	34	/	43
8	8	H	17	Q	26	Z	35	:	44

Kode QR lahko za isto sporočilo izgledajo drugače, saj je verzija kode sorazmerno odvisna od dolžine sporočila, vrste podatkov in od ravni popravljanja napak.

5.3.2 Kontrolni simboli za popravljanje napak

Da povečamo zanesljivost podatkov in zmožnost popravljanja in odkrivanja napak v kodi QR, sporočilo zakodiramo z RS kodami. Generiramo jih v skladu s kodirnim algoritmom opisanim v razdelku 4.1. Poglejmo, kako deluje algoritem pri kodah QR.

Opomba 5.1. Vsa aritmetika v kodah QR je nad končnim obsegom $GF(2^8)$, generiranim z nerazcepnim polinomom $x^8+x^4+x^3+x^2+x+1$. Polinom lahko bolj kompaktno zapišemo tako, da samo po vrsti naštejemo njegove koeficiente, najprej koeficient pri najvišji potenci. Za naš polinom je to 10001111_2

oziroma 285_{10}. Podatkovni simboli so dolžine 8. Spomimo se, da lahko vsa neničelna števila v obsegu predstavimo kot potenco primitivnega elemeta, kar nam poenostavi množenje z velikimi števili. V obsegu $GF(2^8)$ je primitivni element $\alpha = 2$, kar ustreza polinomu x. Vsa števila lahko tako predstavimo v obiki 2^k, kjer je $0 \leq k \leq 255$.

Do sedaj smo pretvorili sporočilo v 8 bitne podatkovne simbole. Odvisno od izbrane ravni popravljanja napak in verzije kode QR je določeno, koliko kontolnih simbolov za popravljanje napak potrebujemo (tabela 5.7 na strani 46). Recimo, da imamo k podatkovnih simbolov in potrebujemo t kontolnih simbolov. Podatkovni simboli $m = (m_{k-1}, \ldots, m_1, m_0)$ so koeficienti polinoma sporočila $m(x) = m_{k-1}x^{k-1} + \ldots + m_2x^2 + +m_1x + m_0$. Število kontrolnih simbolov t določa stopnjo generatorskega polinoma $g(x)$, ki ga izračunamo v skadu z definicijo 4.1. Vsak polinom $g(x)$ je produkt polinomov prve stopnje $x - \alpha^0, x - \alpha^1, \ldots, x - \alpha^{t-1}$; kjer je α primitivni element obsega. Tabela 5.8 prikazuje nekaj generatorskih polinomov, ki se uporabljo v kodah QR. Poglejmo si postopek kodiranja. Polinom sporočila pomnožimo z x^t in delimo z generatorskim polinomom $g(x)$. Zanima nas ostanek $b(x)$, kar lahko zapišemo kot $b(x) = m(x)x^t \pmod{g(x)}$. Koeficienti ostanka pri deljenju $b = (b_{t-1}, \ldots, b_o)$ so kontolni simboli za popravljanje napak in jih dodamo k podatkovnim simbolom. Podatkovne simbole in kontrolne simbole za popravljanje napak skupaj imenujemo kodna beseda. Sporočilu tako s kodiranjem priredimo kodno besedo $c = (m_{k-1}, \ldots, m_1, m_0, b_{t-1}, \ldots, b_1, b_0)$.

Da je možnost popravljanja napak učinkovitejša, se daljše zaporedje podatkovnih simbolov razdeli na manjše *bloke* za popravljanje napak. Kodiramo vsak blok posebej. Velikost bloka je omejena na 30 kontrolnih simbolov za popravljanje napak, kar pomeni, da je mogoče popraviti do 15 napak na blok. V kodi QR verzije 2-L, kjer imamo 10 kontolnih simbolov, je en blok za popravljanje napak dovolj. V verziji 3-Q s 36 kontrolnimi simboli pa potrebujemo dva bloka za popravljanje napak. Bloki se prepletejo v skladu s specifikacijami kode QR.

Opomba 5.2. Več o prepletanju si lahko pogledamo v [5, stran 46]. Za naš zgled je dovolj en blok.

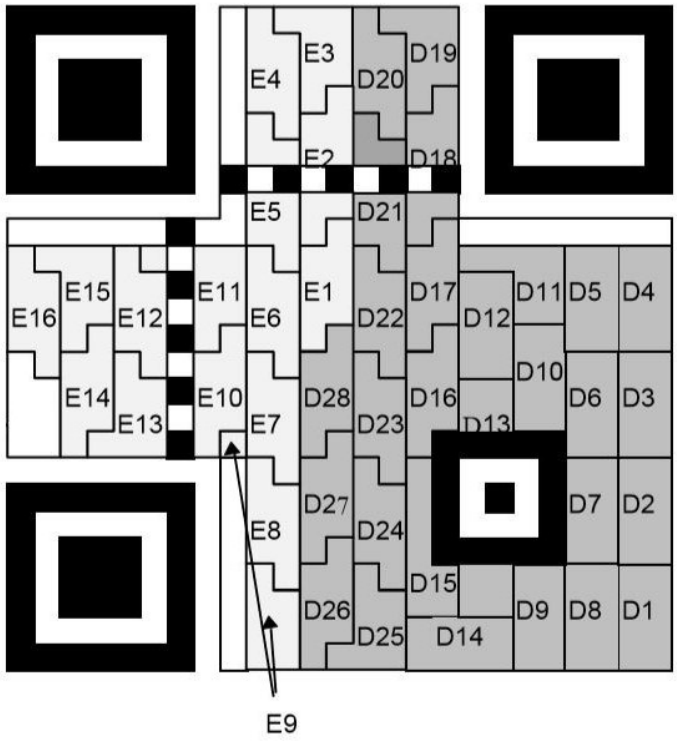
Tabela 5.8: Generatorski polinomi za generiranje RS kontrolnih simbolov za popravljanje napak [5, Tabela A.1]. Koeficienti so predstavljeni s potenco primitivnega polinoma $\alpha = 2$. S t je označeno število kontrolnih simbolov.

t	Generatorski polinom $g(x)$
2	$x^2 + \alpha^{25}x + \alpha$
5	$x^5 + \alpha^{113}x^4 + \alpha^{164}x^3 + \alpha^{166}x^2 + \alpha^{119}x + \alpha^{10}$
6	$x^6 + \alpha^{116}x^5 + x^4 + \alpha^{134}x^3 + \alpha^5x^2 + \alpha^{176}x + \alpha^{15}$
7	$x^7 + \alpha^{87}x^6 + \alpha^{229}x^5 + \alpha^{146}x^4 + \alpha^{149}x^3 + \alpha^{238}x^2 + \alpha^{102}x + \alpha^{21}$
8	$x^8 + \alpha^{175}x^7 + \alpha^{238}x^6 + \alpha^{208}x^5 + \alpha^{249}x^4 + \alpha^{215}x^3 + \alpha^{252}x^2 + \alpha^{196}x + \alpha^{28}$
10	$x^{10} + \alpha^{251}x^9 + \alpha^{67}x^8 + \alpha^{46}x^7 + \alpha^{61}x^6 + \alpha^{118}x^5 + \alpha^{70}x^4 + \alpha^{64}x^3 + \alpha^{94}x^2$ $+\alpha^{32}x + \alpha^{45}$
13	$x^{13} + \alpha^{74}x^{12} + \alpha^{152}x^{11} + \alpha^{176}x^{10} + \alpha^{100}x^9 + \alpha^{86}x^8 + \alpha^{100}x^7 + \alpha^{106}x^6$ $+\alpha^{104}x^5 + \alpha^{130}x^4 + \alpha^{218}x^3 + \alpha^{206}x^2 + \alpha^{140}x + \alpha^{78}$

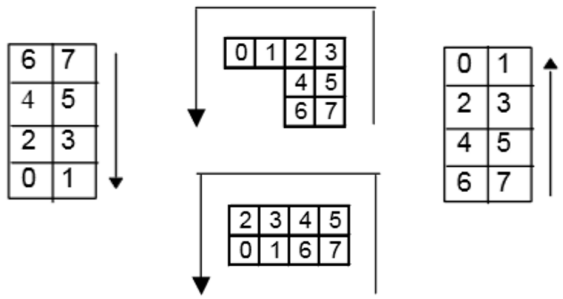
5.3.3 Vstavljanje kodne besede v podatkovno polje

Po ustvarjanju podatkovnih simbolov in kontolnih simbolov za popravljanje napak moramo kodno besedo pravilno vstaviti v matriko. Kodno besedo razčlenimo in zapisujemo v manjše 8 bitne simbole, ki tvorijo *kodni blok* velikosti 2x4 module. Kodni bloki skupaj tvorijo stolpce, z začetkom v desnem spodnjem kotu simbola kode QR in se izmenično linijsko razvrstijo navzgor in navzdol od desne proti levi. Če v stolpcu naletimo na funkcijski vzorec, se kodni blok razporedi okrog ali ga preskoči. Taki blok imenujemo *nepravilni kodni blok*. V zgledu na sliki 5.2 kodni blok D1 predstavlja prvi podatkovni simbol, kodni blok D2 drugega in tako naprej. S kodnim blokom E1 je označen prvi kontrolni simbol za popravljanje napak, S kodnim blokom E2 drugi in tako naprej. Zaporedje postavitve bitov v kodnem bloku je prav tako

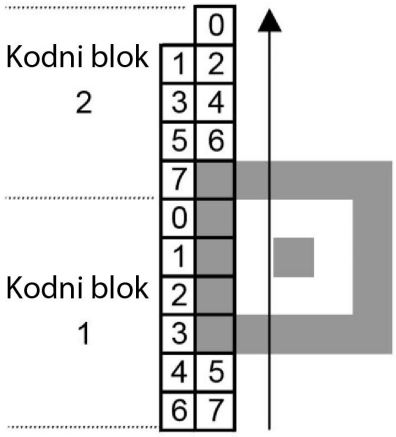
od desne proti levi in bodisi navzgor ali navzdol, odvisno od smeri postavitve bloka. Najpomembnejši (prvi) bit kodnega bloka (prikazan s številom 7 na sliki 5.3) se zapiše v prvi razpoložljivi modul. Najpomembnejši bit tako zavzema spodnji desni modul kodnega bloka, kadar je smer postavitve navzgor, in zgornji desni modul, kadar je smer postavitve navzdol. Lahko pa zasede spodnji levi modul nepravilnega kodnega bloka, če se je prejšnji kodni blok zaključil v desnem stolpcu modula (glej sliko 5.4).



Slika 5.2: Razporeditev podatkovnih simbolov in kontolnih simbolov za popravljanje napak po podatkovnem polju. Koda QR verzije 2-M.



Slika 5.3: Kodni blok in orientacija modulov glede na položaj v matriki.



Slika 5.4: Položaj kodnega bloka in orientacija modulov ob vzorcu poravnave.

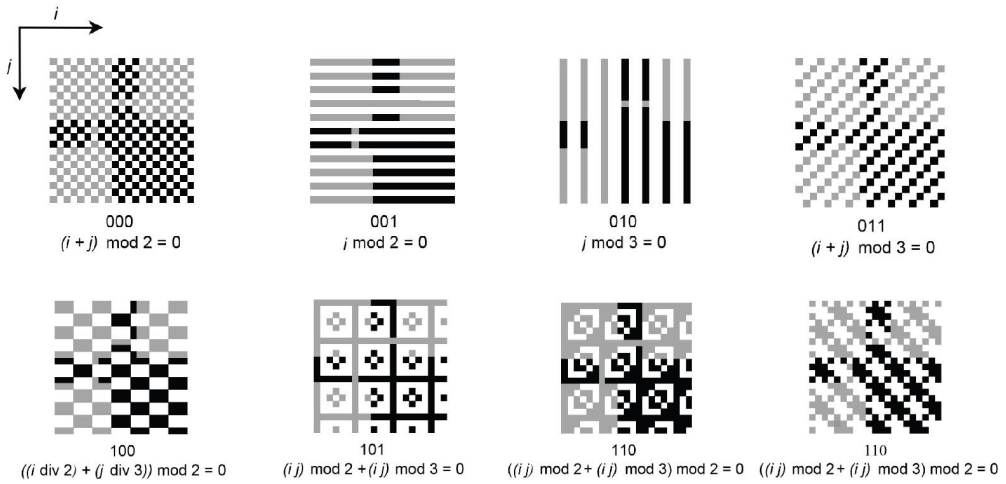
5.3.4 Zaščita kodne besede (angl. *Data Masking*)

Postavitev kodne besede v podatkovno polje lahko tvori vzorce podobne funkcijskim, kar bi čitalcu oteževalo hitro obdelavo kode. Da bi to preprečili, specifikacja kode QR določa osem vzorcev *podatkovnih mask* na podatkovnem polju. Izbere se tista, ki najbolj enakomerno porazdeli število črnih in belih modulov. Izbrana maska se označi s 3 bitnim *indikatorjem zaščite*. Na sliki 5.5, so prikazane verzije podatkovnih mask s pogojem za njihovo generiranje in njen indikator. Za izbran modul se pogoj poračuna glede na vrstico in stolpec v kateri se nahaja. Oznaka i se nanaša na vrstico, j pa

na stolpec, pri čemer $(i, j) = (0, 0)$ označuje zgornji levi modul. Modul, ki ustreza pogoju, je obarvan črno. Ker se funkcijski vzorci ter informacija o formati in verziji ne „maskirajo“ so na sliki 5.5 obarvani sivo.

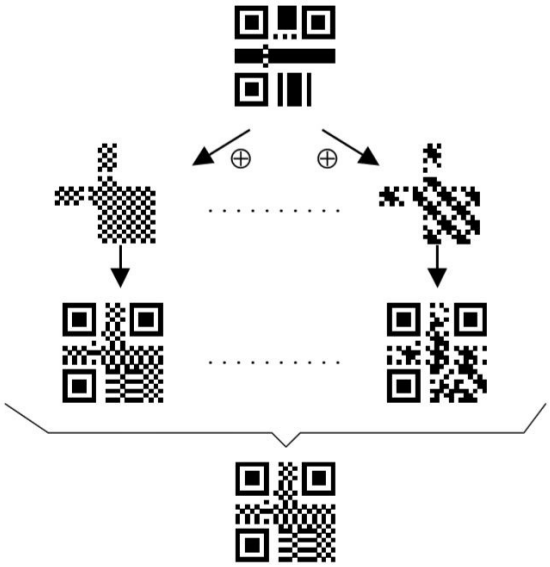
Maskiranje se enostavno uporablja (ali odstrani) z uporabo operacije XOR. Poglejmo si primer maskiranja dveh podatkovnih simbolov s podano masko.

Podatki: 10110110 10010110
Maska: 10101000 00100100
Rezultat: 00011110 10110010



Slika 5.5: Vzorci podatkovne maske na kodi QR verzije 1 z indikatorjem zaščite in pogojem za generiranje.

Na sliki 5.6 je predstavljena simulacija izbire ustrezne podatkovne maske. Vse možnosti se preverijo in nato izbere tista, ki najbolj enakomerno porazdeli črne in bele module po matriki.



Slika 5.6: Simulacija izbire ustrezne podatkovne maske podatkov na kodi QR verzije 1.

5.3.5 Informacija o formatu in verziji

Na koncu vstavimo na določeno mesto še zakodirano informacijo o formatu in verziji kode QR. V informaciji o formatu je zapisana raven popravljanja napak in podatkovna maska. Ker se informacija razporedi okrog pozicijskih vzorcev, kot lahko vidimo na sliki 5.1 na začetku poglavja, je verjetnost napake v skupkih manjša. Za kodiranje informacije o formatu in verziji se uporabljajo splošnejše BCH(15,5) kode [15], ki so učinkovitejše za popravljanje naključnih bitnih napak. Po kodiranju se podatki zaščitijo z masko 101010000010010, ki zagotavlja, da nobena kombinacija ravni popravljanja napak in podatkovne maske ne povzroči ničelnega niza. Modul na položaju $(4V + 9, 8)$, kjer je V verzija kode QR je vedno črn in ni del informacije o formatu.

Informacija o verziji nosi podatek o verziji QR matrike in se zapiše le od verzije 7 naprej. V manjših verzijah je ničelna. Zakodirana je z Golaye-

vimi $(18, 6)$ kodami in nima maske.

Zakodirajmo naše sporočila PIKA NOGAVICKA iz zgleda 5.4 z RS kodi kot smo opisali zgoraj, in dobljeno kodno besedo vstavimo v kodo QR.

Zgled 5.5. Podatkovni simboli 32, 166, 119, 113, 217, 174, 36, 30, 22, 108, 227, 128, 236, 17, 236, 17 so koeficienti členov polinoma sporočila.

$$m(x) = 32x^{15} + 166x^{14} + 119x^{13} + \ldots + 236x + 17.$$

Podatkovnim simbolom moramo dodati 10 kontrolnih simbolov za popravljanje napak, kar nam pove, da je stopnja generatorskega polinoma enaka 10. Polinom sporočila pomnožimo z x^{10}, da naredimo prostor kontrolnim simbolom za popravljanje napak. Dobljeni polinom delimo z generatorskim polinomom iz zgleda 4.2, na strani 19. Koeficienti ostanka $b(x)$, ki ga dobimo pri deljenju, so iskani kodni simboli za popravljanje napak.

$$b(x) = 9x^9 + 20x^8 + 49x^7 + 159x^6 + 171x^5 + 18x^4 + 99x^3 + 21x^2 + 96x + 25.$$

Kontrolne simbole za popravljanje napak: $9, 20, 49, 159, 171, 18, 99, 21, 96, 25$ pretvorimo v bitni zapis in jih dodamo na koncu sporočila (prikazani so podčrtano):

00100000 01110100 01110111 01110001 11011001 10101110 00100100
00011110 00010110 01101100 11100011 10000000 11101100 00010001
11101100 00010001 <u>00001001</u> <u>00010100</u> <u>00110001</u> <u>10011111</u> <u>10101011</u>
<u>00010010</u> <u>01100011</u> <u>00010101</u> <u>01100000</u> <u>00011001</u>.

V skladu s poglavjem 5.3.3 se v podatkovni polje zapiše dobljena kodna beseda in se zaščiti z ustrezno masko. Z BCH kodami se zakodira informacija o formatu. Zgled je dostopen v standardu [5, priloga C]. Končen rezultat je prikazan na sliki 5.7. Generirala sem ga z uporabo generatorja kode QR, dosegljivega na strani [17].

Celoten proces ustvarjanja kode QR lahko ponazorimo z diagramom na sliki 5.8.

Slika 5.7: Koda QR, verzije 1-M z vsebino PIKA NOGAVICKA.

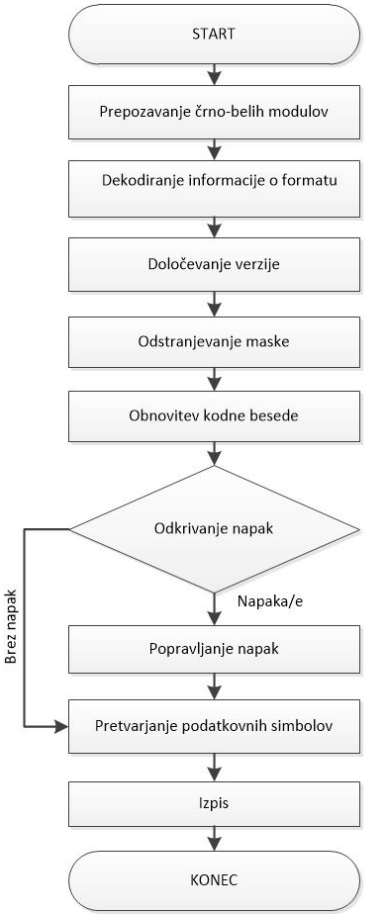
Slika 5.8: Diagram prikazuje proces kodiranja kode QR.

5.4 Branje kode QR

Postopek dekodiranja vsebine kode QR je ponazorjen z diagramom na sliki 5.9. Poteka v obratnem vrstnem redu kot kodiranje.

Najprej čitalec, ko mu približamo kodo QR, prepozna pozicijske vzorce za usmerjenost, vzorce za poravnavo, da izniči kot skeniranja in s pomočjo časovnega vzorca določi širino modula. Čitalec dobi sliko simbola. Dekodiranje podatkov se začne z branjem informacije o formatu, ki je bilo zakodirano z BCH kodami. Primerja oba zapisa informacije v kodi QR, ki morata hra-

niti enak zapis in po potrebi popravi napake. Tako določi raven popravljanja napak za podatkovno polje in podatkovno masko. Z branjem informacije o verziji, se določi verzija kode QR. Naslednji korak je odstranitev podatkovne maske z enako operacijo, kot se je izvedla v postopku kodiranja. Koda QR je pripravljena za dekodiranje. Branje kodne besede se bere v skladu z pravili zapisovanja v simbol (razdelek 5.3.3). Z dekodirnim algoritmom iz razdeleka 4.2 se izvede postopek odkrivanja in popravljanja morebitnih napak. Iz dobljenih podatkovnih simbolov se najprej prebere indikator načina pretvorbe in števec znakov. V skladu z vrsto podatkov, ki jo določi indikator načina, se prebere vsebino in jo izpiše.



Slika 5.9: Diagram prikazuje proces dekodiranja kode QR.

Tabela 5.7: Velikost vhodnih podatkov in število kontrolnih simbolov za popravljanje napak (*ecc pomeni error correction codewords*) posamezne verzije kode QR, glede na vrsto podatkov in raven popravljanja napak [5].

Verzija	Raven p.n	Numerični	Alfanumerični	Bajtni	Kanji	Število ecc
1	L	41	25	17	10	7
	M	34	20	14	8	10
	Q	27	16	11	7	13
	H	17	10	7	4	17
2	L	77	47	32	20	10
	M	63	38	26	16	16
	Q	48	29	20	12	22
	H	34	20	14	8	28
3	L	127	77	53	32	15
	M	101	61	42	26	26
	Q	77	47	32	20	36
	H	58	35	24	15	44
4	L	187	114	78	48	20
	M	149	90	62	38	36
	Q	111	67	46	28	52
	H	82	50	34	21	64
5	L	255	154	106	65	26
	M	202	122	84	52	48
	Q	144	87	60	37	72
	H	106	64	44	27	88
6	L	322	195	134	82	36
	M	255	154	106	65	64
	Q	178	108	74	45	96
	H	139	84	58	36	112
7	L	370	224	154	95	40
	M	293	178	122	75	72
	Q	207	125	86	53	108
	H	154	93	64	39	130
8	L	461	279	192	118	48
	M	365	221	152	93	88
	Q	259	157	108	66	132
	H	202	122	84	52	156
9	L	552	335	230	141	60
	M	432	262	180	111	110
	Q	312	189	130	80	160
	H	235	143	98	60	192
10	L	652	395	271	167	72
	M	513	311	213	131	130
	Q	364	221	151	93	192
	H	288	174	119	74	224

Poglavje 6

Varnost

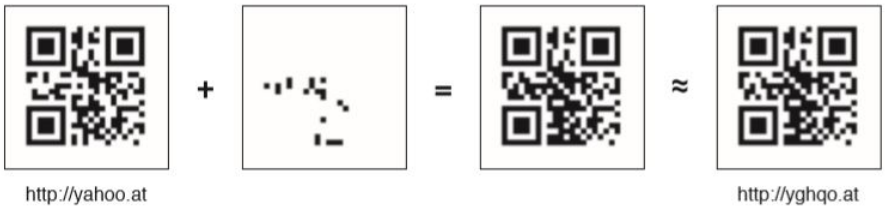
Svet je obdan s kodami QR. Raven uporabnosti je široka, tudi na resnih področjih kot so denarne transakcije. Ali smo preveč zaupljivi?

S premišljenim spreminjanjem črnih modulov v bele in obratno, se lahko spremeni vsebina kode QR. To imenujemo *napad* na kodo QR. Napadena koda QR najpogosteje kodira zlonamerno povezavo, ki nas poveže na lažno spletno mesto. Ker je koda QR samo strojno berljiva, ne moremo razlikovati med veljavno in spremenjeno kodo QR. To s pridom izkoriščajo napadalci. Poglejmo si nekaj najpogostejših napadov.

Lažno predstavljanje s kodo QR (angl. *QRishing*). QRishing je razširitev lažnega predstavljanja (angl. *phishing*) preko kode QR [16]. S skeniranjem napadene kod QR napadalec poskuša na naši programski opremi obiskati zlonamerno spletno stran ali pridobiti zaupne podatke uporabnika. Ranljivi so mobilni telefoni, saj z njimi beremo kodo. Cilj napadov je pridobiti podatke uporabnika, ki so zaupne narave, kot so številke kreditnih kartic, vstopna gesla, kraje osebnih podatkov, neželena sporočila, dostop do zasebnih omrežij. Preko kode QR se lahko prenašajo računalniški virusi in črvi. Na Kitajskem se je preko kod QR preneslo 23% virusov [12].

Zgled 6.1. Slika 6.1 iz [9] prikazuje primer napada z lažnim predstavljanjem. V kodi QR je bila shranjena povezava do spletne strani http://yahoo.at.

Sprememba 20 belih modulov v črne je povzročila, da se je vsebina spremenila tako, da so ob skeniranju kode QR RS kodi za popravljanje napak „popravili“ vsebino na http://yghqo.at.



Slika 6.1: Napad lažnega predstavljanja s kodo QR.

Vrivanje SQL (angl. *SQL injection*). SQL (angl. *Structured Query Language*) je standardariziran jezik za upravljanje zbirk podatkov. Napad, kjer napadalec skuša v obstoječo programsko kodo vriniti dodatne ukaze, da bi lahko dostopal do zaupnih informacij imenujemo, vrivanje SQL (angl. *SQL injection*). Bolj specifični napadi lahko vključujejo dodajanje uporabnika, izvajanje sistemskih ukazov ali spreminjanje podatkov, kot so cene ali gesla znotraj zbirke podatkov. V kodi QR je lahko zakodiran tak ukaz [9].

Ugrabitev prijave s QR (angl. *QRL Jacking - QR Code Login Jacking*). Nekatere aplikacije kot so WhatsApp, WeChat, Line, uporabnikom omogočajo prijavo na splet s skeniranjem kode QR [8]. Uspešni napadalec nastavi lažno kodo QR. S tem pridobi podatke trenutne lokacije, kodo naprave, podatke o SIM kartici in druge občutljive podatke, ki jih aplikacija odjemalca predstavi pri postopku prijave. Tako napadalec lahko prevzame nadzor nad komunikacijo.

Kode QR so samo strojno berljive in kar tako ne moremo zaznati, ali je vsebina škodljiva. Rešitev pred napadalnimi kodami QR je v prvi vrsti v ozaveščanju in izobraževanju ljudi. Da smo previdnejši pri skeniranju in uporabljamo primerne aplikacije za branje. Uporabljamo take aplikacije, ki nas po skeniranju kode QR ne povežejo neposredno na spletno stran, ampak

nam najprej izpišejo vsebino. Napadi so sprožili razvoj novih, varnejših kod, z dodatno zaščito. Varna koda QR (angl. *QR Code Secure*) uporablja AES šifriranje in digitalni podpis za preverjanje pristnosti izvora in celovitosti kode QR [13, 19].

Poglavje 7

Zaključek

V diplomskem delu smo se seznanili z zgradbo kode QR in njeno zanesljivostjo. Skozi proces kodiranja smo spoznali lastnosti in način delovanja. Podrobneje smo si pogledali kjučen element - Reed-Solomove kode, ki omogočajo obnovitev do 30% poškodovanih podatkov.

Zaradi različnih potreb po zapisovanju podatkov so se razvile dodatne različice kode QR:

Mikro kode QR. Uporabljamo jih, kadar želimo zapisati malo podatkov, npr. ID – številke, na manjši površini. Imajo samo en pozicijski vzorec.

iQR. Leta 2008 je bila izdana koda iQR, ki je pravkotne oblike. Ima večjo zmogljivost, vanjo lahko zapišemo do 40.000 znakov, in omogoča do 50% obnovljivost podatkov.

Tajna koda QR (angl. *Secret-function-equipped QR code*). Po videzu se ne razlikuje od običajne kode QR, ima pa dodatno funkcijo omejevanja branja. Vsebuje javne in zasebne podatke. Zasebni podatki so berljivi le z namenskimi čitalci, ki imajo kriptografski ključ. Uporablja se za shranjevanje zasebnih podatkov ali za upravljanje notranjih informacij podjetja.

Poglejmo si še nekaj zanimivosti v svetu uporabe kod QR. Prisotnost Reed-Solomonovih kod nekateri izkoristijo za umetniške namene. Napake povzročijo načrtno, da je koda privlačnejša za oko. Do 30% površine simbola

Slika 7.1: Koda QR iz 130.000 dreves, ki jo lahko skeniramo iz zraka.

kode QR lahko zavzame logotip podjetja, brez vpliva na berljivost. Lahko se poigramo tudi z barvami modulov. Ključna stvar je, da izberemo ustrezen kontrast. Kombinacija rumenega ozadja in modrih ali zelenih modulov bi bila berljiva, zeleno ozadje in modri moduli pa ne. Da je koda QR lahko umetniška, vidimo tudi na sliki 7.1, ki je dostopna na strani [11].

Ker koda QR hrani binarne podatke, vanje lahko zapišemo tudi slike in zvok. Pri tem je količina omejena na 3kB.

Dodatek A

Elementi kodne besede so v kodi QR definirani nad končnim obsegom $GF(2^8)$, generiranim z nerazcepnim polinomom $x^8+x^4+x^3+x^2+x+1$. Število vseh elementov obsega je 256. Elementi so v tabeli A.1 predstavljeni s potenco primitivnega elementa α^i, ki ustreza polinomu x in z 8 bitnim zapisom. Bitni zapis predstavlja koeficiente polinoma nad obsegom $GF(2^8)$, kot smo opisali v poglavju 2.

Tabela A.1: Elementi končnega obsega $GF(2^8)$ generiranega s polinomom $x^8+x^4+x^3+x^2+x+1$.

α^i	α^i	α^i	α^i	α^i
α 00000010	α^2 00000100	α^3 00001000	α^4 00010000	α^5 00100000
α^6 01000000	α^7 10000000	α^8 00011101	α^9 00111010	α^{10} 01110100
α^{11} 11101000	α^{12} 11001101	α^{13} 10000111	α^{14} 00010011	α^{15} 00100110
α^{16} 01001100	α^{17} 10011000	α^{18} 00101101	α^{19} 01011010	α^{20} 10110100
α^{21} 01110101	α^{22} 11101010	α^{23} 11001001	α^{24} 10001111	α^{25} 00000011
α^{26} 00000110	α^{27} 00001100	α^{28} 00011000	α^{29} 00110000	α^{30} 01100000
α^{31} 11000000	α^{32} 10011101	α^{33} 00100111	α^{34} 01001110	α^{35} 10011100
α^{36} 00100101	α^{37} 01001010	α^{38} 10010100	α^{39} 00110101	α^{40} 01101010
α^{41} 11010100	α^{42} 10110101	α^{43} 01110111	α^{44} 11101110	α^{45} 11000001
α^{46} 10011111	α^{47} 00100011	α^{48} 01000110	α^{49} 10001100	α^{50} 00000101
α^{51} 00001010	α^{52} 00010100	α^{53} 00101000	α^{54} 01010000	α^{55} 10100000
α^{56} 01011101	α^{57} 10111010	α^{58} 01101001	α^{59} 11010010	α^{60} 10111001
α^{61} 01101111	α^{62} 11011110	α^{63} 10100001	α^{64} 01011111	α^{65} 10111110
α^{66} 01100001	α^{67} 11000010	α^{68} 10011001	α^{69} 00101111	α^{70} 01011110
α^{71} 10111100	α^{72} 01100101	α^{73} 11001010	α^{74} 10001001	α^{75} 00001111
α^{76} 00011110	α^{77} 00111100	α^{78} 01111000	α^{79} 11110000	α^{80} 11111101
α^{81} 11100111	α^{82} 11010011	α^{83} 10111011	α^{84} 01101011	α^{85} 11010110
α^{86} 10110001	α^{87} 01111111	α^{88} 11111110	α^{89} 11100001	α^{90} 11011111
α^{91} 10100011	α^{92} 01011011	α^{93} 10110110	α^{94} 01110001	α^{95} 11100010
α^{96} 11011001	α^{97} 10101111	α^{98} 01000011	α^{99} 10000110	α^{100} 00010001

α^{101}	00100010	α^{102}	01000100	α^{103}	10001000	α^{104}	00001101	α^{105}	00011010
α^{106}	00110100	α^{107}	01101000	α^{108}	11010000	α^{109}	10111101	α^{110}	01100111
α^{111}	11001110	α^{112}	10000001	α^{113}	00011111	α^{114}	00111110	α^{115}	01111100
α^{116}	11111000	α^{117}	11101101	α^{118}	11000111	α^{119}	10010011	α^{120}	00111011
α^{121}	01110110	α^{122}	11101100	α^{123}	11000101	α^{124}	10010111	α^{125}	00110011
α^{126}	01100110	α^{127}	11001100	α^{128}	10000101	α^{129}	00010111	α^{130}	00101110
α^{131}	01011100	α^{132}	10111000	α^{133}	01101101	α^{134}	11011010	α^{135}	10101001
α^{136}	01001111	α^{137}	10011110	α^{138}	00100001	α^{139}	01000010	α^{140}	10000100
α^{141}	00010101	α^{142}	00101010	α^{143}	01010100	α^{144}	10101000	α^{145}	01001101
α^{146}	10011010	α^{147}	00101001	α^{148}	01010010	α^{149}	10100100	α^{150}	01010101
α^{151}	10101010	α^{152}	01001001	α^{153}	10010010	α^{154}	00111001	α^{155}	01110010
α^{156}	11100100	α^{157}	11010101	α^{158}	10110111	α^{159}	01110011	α^{160}	11100110
α^{161}	11010001	α^{162}	10111111	α^{163}	01100011	α^{164}	11000110	α^{165}	10010001
α^{166}	00111111	α^{167}	01111110	α^{168}	11111100	α^{169}	11100101	α^{170}	11010111
α^{171}	10110011	α^{172}	01111011	α^{173}	11110110	α^{174}	11110001	α^{175}	11111111
α^{176}	11100011	α^{177}	11011011	α^{178}	10101011	α^{179}	01001011	α^{180}	10010110
α^{181}	00110001	α^{182}	01100010	α^{183}	11000100	α^{184}	10010101	α^{185}	00110111
α^{186}	01101110	α^{187}	11011100	α^{188}	10100101	α^{189}	01010111	α^{190}	10101110
α^{191}	01000001	α^{192}	10000010	α^{193}	00011001	α^{194}	00110010	α^{195}	01100100
α^{196}	11001000	α^{197}	10001101	α^{198}	00000111	α^{199}	00001110	α^{200}	00011100
α^{201}	00111000	α^{202}	01110000	α^{203}	11100000	α^{204}	11011101	α^{205}	10100111
α^{206}	01010011	α^{207}	10100110	α^{208}	01010001	α^{209}	10100010	α^{210}	01011001
α^{211}	10110010	α^{212}	01111001	α^{213}	11110010	α^{214}	11111001	α^{215}	11101111
α^{216}	11000011	α^{217}	10011011	α^{218}	00101011	α^{219}	01010110	α^{220}	10101100
α^{221}	01000101	α^{222}	10001010	α^{223}	00001001	α^{224}	00010010	α^{225}	00100100
α^{226}	01001000	α^{227}	10010000	α^{228}	00111101	α^{229}	01111010	α^{230}	11110100
α^{231}	11110101	α^{232}	11110111	α^{233}	11110011	α^{234}	11111011	α^{235}	11101011
α^{236}	11001011	α^{237}	10001011	α^{238}	00001011	α^{239}	00010110	α^{240}	00101100
α^{241}	01011000	α^{242}	10110000	α^{243}	01111101	α^{244}	11111010	α^{245}	11101001
α^{246}	11001111	α^{247}	10000011	α^{248}	00011011	α^{249}	00110110	α^{250}	01101100
α^{251}	11011000	α^{252}	10101101	α^{253}	01000111	α^{254}	10001110	α^{255}	00000001

Literatura

[1] M. W. Baldoni, C. Ciliberto, and G. M. Piacentini Cattaneo. *Elementary Number Theory, Cryptography and Codes*. Springer, Bonn, 2008.

[2] M. Ebling and R. Cáceres. Bar codes everywhere you look. *IEEE Pervasive Computing*, 9:4–5, 2010.

[3] D. G. et al. Hoffman. *Coding Theory: The Essentials*. Marcel Dekker, Inc., New York, 1991.

[4] M. Hsieh and F. Le Gall. Np-hardness of decoding quantum error-correction codes. *Physical Review A*, 83, 2011.

[5] ISO/IEC 18004:2006. *Information technology - Automatic identification and data capture techniques - QR Code 2005 bar code symbology specification*. International Organization for Standardization, Geneva, 2006.

[6] ISO/IEC 18004:2015. *Information technology - Automatic identification and data capture techniques - QR Code bar code symbology specification*. International Organization for Standardization, Geneva, 2015.

[7] A. Jurišić. Napake niso večne. *Presek*, 30(6):361–366, 2002/2003.

[8] S. Khandelwal. QRLJacking - Hacking Technique to Hijack QR Code Based Quick Login System. Dosegljivo: `https://thehackernews.com/2016/07/qrljacking-hacking-qr-code.html`, 2016. [Dostopano: 6. 6. 2018].

[9] P. Kieseberg et al. Malicious pixels using QR codes as attack vector. In *Trustworthy Ubiquitous Computing*, pages 21–38. Atlantis Press, Paris, 2012.

[10] R. Škrekovski. *Diskretne strukture II*. Samozaložba, Ljubljana, 2010.

[11] M. L. Major. This Massive QR Code Made of 130,000 Trees Can Only Be Scanned From the Sky. Dosegljivo: `https://interestingengineering.com/this-massive-qr-code-made-of-130000-trees-can-only-be-scanned-from-the-sky`, 2017. [Dostopano: 1. 9. 2018].

[12] D. Mandal. QR Codes - Security Challenges: Did We All Jump The Gun Here? Dosegljivo: `https://gomedici.com/qr-codes-security-challenges/`, 2017. [Dostopano: 1. 9. 2018].

[13] V. Mavroeidis and M. Nicho. Quick response code secure: A cryptographically secure anti-phishing tool for QR code attacks. In *Computer Network Security*, pages 313–324. Springer, Cham, 2017.

[14] T. K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, New York, 2005.

[15] R. H. Morelos-Zaragoza. *The Art of Error Correcting Coding*. John Wiley & Sons, USA, 2006.

[16] Phishing. Dosegljivo: `https://www.cert.si/si/varnostne-groznje/phishing/`. [Dostopano: 23. 1. 2019].

[17] QR code 2D barcode generator. Dosegljivo: `https://racoindustries.com/barcodegenerator/2d/qr-code/`. [Dostopano: 1. 9. 2018].

[18] QR codes provide medical data for emergency responders. Dosegljivo: `https://designtoimprovelife.dk/codeurgencedesigntoimprovelifeapp/`, 2012. [Dostopano: 1. 9. 2018].

[19] T. Satyanarayana and G. Swathi. Secure QR code for anti-phishing system using mobile. *IRJES*, 2(12):78–81, 2013.

[20] I. Vidav. *Algebra*. DMFA-Založništvo, Ljubljana, 2003.

[21] Y. Wang. Decoding generalized Reed-Solomon codes and its application to RLCE encryption schemes. *CoRR*, abs/1702.07737, 2017.

[22] D. Welsh. *Codes and Cryptography*. Oxford University Press, Oxford, 1988.