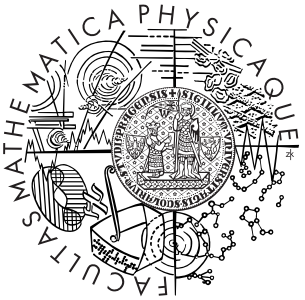
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Ludmila Divišová

Dekódování Reed-Solomonových a BCH kódů

Katedra Algebry

Vedoucí bakalářské práce: Mgr. Jan Šťovíček Ph.D

Studijní program: matematika
Studijní obor: obecná matematika

Praha 2011

Děkuji vedoucímu své práce panu Mgr. Janu Šťovíčkovi Ph.D. za kontrolu a konzultace.

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 24. května 2011 Ludmila Divišová

Název práce: Dekódování Reed-Solomonových a BCH kódů
Autor: Ludmila Divišová
Katedra (ústav): Katedra Algebry
Vedoucí bakalářské práce: Mgr. Jan Šťovíček Ph.D.
e-mail vedoucího: <stovicek@karlin.mff.cuni.cz>

Abstrakt: V předložené práci se zabýváme dvěma třídami samoopravných kódů, a to Reed-Solomonovými a BCH. Kromě definic a základních vlastností se na začátku textu nachází také shrnutí základních pojmů z teorie kódování. Reed-Solomonovy kódy jsou definovány pomocí evaluací polynomů a jejich vztah k BCH kódům je dokázán prostřednictvím Mattson-Solomonových polynomů a inverzní formule. V kapitole věnované dekódování se seznamujeme s lokačním polynomem, evaluačním polynomem a Newtonovými identitami, jak v obecné tak v binární formě. Uvádíme Petersonův a Berlekamp-Masseyův dekódovací algoritmus, druhý jmenovaný také v rychlejší mutaci pro binární kódy a doplněný ilustračním příkladem.

Klíčová slova: samoopravné kódy, BCH kódy, Reed-Solomonovy kódy, Berlekamp-Masseyův algoritmus

Title: Decoding of Reed-Solomon and BCH codes
Author: Ludmila Divišová
Department: Department of Algebra
Supervisor: Mgr. Jan Šťovíček Ph.D.
Supervisor's e-mail address: <stovicek@karlin.mff.cuni.cz>

Abstract: The dissertation studies two classes of error-correcting codes - Reed-Solomon and BCH. The first two chapters introduce the definitions and basic properties and summarize the basic terms of the coding theory. Subsequently, Reed-Solomon codes are defined by evaluations of polynomials. Their relationship to BCH codes is proved by Mattson-Solomon polynomials and the inversion formula. The next chapter deals with a locator polynomial, evaluator polynomial and Newton identities in their generalized and usual form. Finally, Peterson and Berlekamp-Massey decoding algorithm are presented of which the latter one also appears in its faster mutation for binary codes and is complemented with an illustrative example.

Keywords: error-correcting codes, BCH codes, Reed-Solomon codes, Berlekamp-Massey algorithm

Obsah

Úvod

Svět okolo nás je plný jedniček a nul. Vlní se prostorem mezi vysílačem a přijímačem a přenášejí data - obrázky, texty, telefonní hovory. Stává se, že se cestou jejich posloupnost zkomolí, nastane chyba. A to je místo, kde vstupují na scénu samoopravné kódy. Pokud totiž data před vysláním vhodně upravíme, můžeme na konci jejich putování informačním kanálem zjistit, jestli nastala chyba, najít kde a konečně ji opravit.

V této bakalářské práci představím dvě často používané rodiny kódů, Reed-Solomonovy kódy a BCH kódy. Obě skupiny se, jak se ukáže, navzájem překrývají. Hlavním cílem potom je opravování chyb, tedy popis dekódovacích algoritmů.

BCH - kódy objevil v roce 1959 A. Hocquenhem a v následujícím roce nezávisle na něm R. C. Bose a D. K. Ray-Chaudhuri. Zkratka BCH vznikla z prvních písmen jejich příjmení. Reed-Solomonovy kódy vznikly rovněž v roce 1960.

Pro některé termíny z teorie kódů není v češtině ustálený překlad, proto je budu užívat společně s anglickým názvoslovím.

Kapitola 1

Základní pojmy z teorie kódování

1.1 Samoopravný kód a jeho parametry

Co je tedy samoopravný kód? Vraťme se k situaci naznačené v úvodu. Odesílatel, příjemce a mezi nimi komunikační kanál a data, která po něm chtějí poslat. Zjednodušeně si můžeme představit, že každý znak se při průchodu kanálem poruší s pravděpodobností p. Jak odesílatel, tak příjemce, by rádi měli jistotu, že odeslaná data jsou stejná, jako příchozí. Proto jsou ochotni poslat více dat, když se zvýší pravděpodobnost, že zpráva došla správně. Odeslaná posloupnost znaků tedy bude delší, ale za to bude mít takovou vnitřní strukturu, která nám umožní odstranit chyby vzniklé při přenosu.

Samoopravný kód se obvykle značí symbolem C. Kód tvoří *kódová slova*, neboli vybrané n-tice znaků nad danou abecedou. Zapsáno symbolicky $C \subset \Sigma^n$. Abeceda, značená jako Σ, je množina znaků, které se v kódu používají a nemusí být jenom binární. Kód nad abecedou tvořenou q znaky se nazývá *q-ární*. ($|\Sigma| = q$)

Důležitým parametrem kódu je jeho *délka* n. Délkou kódu rozumíme délku libovolného kódového slova $c \in C$, které můžeme také zapsat jako vektor $c = (c_0, c_1, \cdots c_{n-1})$.

Další charakteristikou je počet kódových slov, neboli *velikost* kódu$|C|$. Nemůžeme totiž použít všechna slova z Σ^n. Základní myšlenkou samoopravného kódu je, že jednotlivá kódová slova od sebe budou "dost daleko". A pokud nenastane při přenosu příliš mnoho chyb, budeme moci z přijatého slova vyčíst, které z kódových slov mu je nejblíž a tedy můžeme předpokládat, že bylo odesláno. Analogii k této situaci najdeme v běžném

používání jazyka. Jen málo kombinací písmen tvoří slovo, které má v daném jazyce svůj význam. Je-li slovo dostatečně dlouhé, domyslí si čtenář jeho smysl navzdory případnému překlepu.

Vzdálenost dvou slov z Σ^n, $x = (x_0 x_1 \cdots x_{n-1})$ a $y = (y_0 y_1 \cdots y_{n-1})$, definujeme jako počet souřadnic, kde se od sebe hodnoty x_i a y_i liší, kde $x_i \neq y_i$ pro $i = 0, ..., n-1$. Také se jí říká Hammingova vzdálenost a značí se $d(x, y)$.

Příklad Slova $x = (11110000)$ a $y = (11000011)$ se od sebe liší na čtyřech pozicích, pro $i = 3, 4, 7, 8$. Tedy $d(x, y) = 4$.

Tak se dostáváme k nejdůležitějšímu parametru kódu, k jeho *minimální vzdálenosti* d, nejmenší vzdálenosti mezi dvěma různými kódovými slovy, $d = \min_{x,y \in C} d(x, y)$, $x \neq y$.

Nyní můžeme říci přesně, co znamená, že dvě slova jsou "dost daleko" od sebe a že nenastalo "příliš mnoho" chyb. Pokud $d = 2t + 1$ a při přenosu nastalo nejvýše t chyb, umíme je opravit. Základní předpoklad, který máme, je, že pravděpodobnost chyby je malá, proto opravujeme na nejbližší kódové slovo (v angličtině maximum likehood decoding). Pokud tento předpoklad není splněn, dekódování selže.

O kódu s uvedenými parametry se mluví jako o (n, k, d) kódu, kde $k = \log_q |C|$.

Pokud má množina Σ strukturu tělesa, $\Sigma = F_q$, je kód C podprostorem vektorového prostoru F_q^n. Takovému kódu říkáme *lineární*. Má-li být F_q těleso, musí být q nutně mocninou nějakého prvočísla.

Parametr k pak vyjadřuje dimenzi C nad Σ, protože $|C| = q^k$.

1.2 Generující a paritní matice lineárního kódu

Abychom kompletně popsali libovolný kód, museli bychom vypsat všechna jeho slova. Pro lineární kódy existuje daleko snažší reprezentace a to pomocí *generující matice* G (generator matrix). Totiž matice, jejíž řádky tvoří bázi kódu C nad F_q. Všechna ostatní kódová slova můžeme dostat jako jejich lineární kombinaci a naopak, pokud se dá slovo napsat jako lineární kombinace bázových vektorů, pak určitě patří do kódu. Generující matice má rozměr $k \times n$.

Pro lineární kódy je také oproti obecným kódům snažší určit minimální vzdálenost. Platí totiž tvrzení[1], že pro lineární kódy se minimální vzdálenost rovná minimální váze nenulového kódového slova,

[1] viz. [4], strana 11.

$d = \min_{0 \neq c \in C} w(c)$. Váha slova $w(c)$ je počet jeho nenulových souřadnic.

Příklad Máme slovo $x = (11101001)$. To má 5 nenulových souřadnic, tedy $w(x) = 5$.

Pro lineární kódy existuje ještě jedna důležitá matice, *paritní matice* (parity-check matrix) značená H. Cesta k ní vede přes duální kód $C^\perp$. Definujme nejprve skalární součin slov x a y nad F_q^n jako $\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$. Duální kód ke kódu C je $C^\perp = \left\{ y \in F_q^n | \forall_{x \in C} \langle x, y \rangle = 0 \right\}$. Jako ortogonální doplněk v F_q^n je $C^\perp$ také lineární kód a jeho dimenze je $n - k$. Generující matice kódu $C^\perp$ je pro kód C paritní matice H.

Proč je paritní matice důležitá? Vidíme, že $GH^T = 0$. Tato vlastnost platí pro každé kódové slovo, tedy $c \in C \Leftrightarrow cH^T = 0$.

Navíc, jak záhy uvidíme, existuje vztah mezi paritní maticí a minimální vzdáleností kódu, který se nám později bude hodit.

Lemma Je-li C lineární kód a H jeho paritní matice, potom minimální vzdálenost d kódu je rovna nejmenšímu počtu sloupců H, které jsou lineárně závislé. Respektive je rovna d, pokud je každých $d-1$ sloupců H lineárně nezávislých.

Důkaz Napišme si $H = (sl_0, \cdots, sl_{n-1})$ pomocí jejích sloupcových vektorů. Vezmeme kódové slovo $c = (c_0, \cdots, c_{n-1})$. Při násobení cH^T dostáváme lineární kombinaci sloupců $\sum_{i=0}^{n-1} c_i sl_i$. Přitom vlastně sčítáme jen ty sloupce sl_i, kde je složka c_i nenulová. Chceme získat nulový vektor. To se nám může podařit jedině pro lineárně závislou množinu, což je alespoň d vektorů. Slovo c tedy musí mít nejméně d nenulových složek. Víme, že pro lineární kódy je minimální váha táž hodnota jako minimální vzdálenost.

1.3 Syndrom a dekódování lineárních kódů

Vraťme se opět k úvodní situaci. Slovo c se při průletu kanálem změnilo na jiné slovo y. Můžeme si vyjádřit chybový vektor $e = y - c$. Pomocí paritní matice budeme s chybovým vektorem pracovat dál.

Definice Buď C lineární kód s paritní maticí H. Potom pro každé $x \in F_q^n$ říkáme součinu xH^T *syndrom* slova x. Označíme ho písmenem s.

Paritní matice H má $n-k$ řádků, tedy $s = (s_0 \dots s_{n-k-1})$.

Víme, že nulový syndrom charakterizuje kódová slova. Tedy pokud je syndrom slova y, které jsme obdrželi, roven nule, bude přijaté slovo pravděpodobně správně. Vlivem chyb, které na kanál působí ovšem můžeme jako y, dostat téměř cokoli. V dalším odstavci si ukážeme jednoduchý způsob, jak pomocí syndromu dekódovat.

Je-li C je lineární kód, je to zároveň také vektorový podprostor F_q^n, můžeme tedy podle něj faktorizovat jako podle podgrupy. Dvě slova leží ve stejné rozkladové třídě, pokud mají stejný syndrom. $yH^T = (c+e)H^T = 0+eH^T = s$. V každé třídě určíme jejího reprezentanta, jako slovo, které má v dané třídě nejmenší váhu. Různé syndromy a reprezentanty jim náležejících tříd uspořádáme do tabulky. Dekódování se provádí jako $c = y - e$, kde e je reprezentant odpovídající syndromu a zároveň předpokládaný chybový vektor. Tato strategie odpovídá našemu předpokladu, že pravděpodobnost chyby je malá, tudíž odeslané slovo bylo spíše zkomoleno na menším počtu pozic. V kapitole věnované dekódování se seznámíme se sofistikovanějšími prostředky, ovšem stále budeme hledat řešení, které předpokládá nejmenší počet chyb.

Příklad Vezměme binární lineární $[5, 2, 3]$-kód s generující maticí $G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$, a paritní maticí $H = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$. Vidíme, že kód C se skládá ze čtyř kódových slov

$$\{(11101), (01011), (10110), (00000)\}.$$

Násobením cH^T můžeme dostat 8 různých syndromů, viz tabulka. Krom toho je z tabulky vidět, že syndromy (101) a (111) nemají jednoznačného reprezentanta. V příslušné třídě neexistuje slovo váhy 1, a chybu s větší vahou neumí náš kód opravit.

syndrom	**reprezentant**	**ostatní slova**
(000)	(00000)	(11101), (01011), (10110)
(001)	(00001)	(01010), (11100), (10111)
(010)	(00010)	(01001)**,**(10100)**,**(11111)
(100)	(00100)	(11001), (01111), (10010)
(011)	(01000)	(00011), (10101), (11110)
(101)	(00101)**nebo** (11000)	(10011), (01110)
(110)	(10000)	(00110), (01101), (11011)
(111)	(10001)**nebo** (01100)	(00111), (11010)

Kapitola 2

Reed-Solomonovy a BCH-kódy

2.1 Zavedení Reed-Solomonových kódů přes evaluace polynomů

V literatuře jsou Reed-Solomonovy kódy často zaváděny jako speciální podskupina BCH-kódů[1]. Existuje však ještě jiný,ekvivalentní způsob, jak se na ně dívat.

Definice Mějme těleso F_q. Všechny jeho nenulové prvky seřadíme do libovolné pevné posloupnosti $\beta_1, \beta_2, \ldots, \beta_{q-1}$. Dále pak vezmeme polynom f nad F_q. Jeho evaluací rozumíme vektor hodnot polynomu na nenulových prvcích z F_q ve zvoleném pořadí,

$$[f] = (f(\beta_1), f(\beta_2), \ldots, f(\beta_{q-1})).$$

Vezmeme $k \geq 1$. Reed-Solomonův kód $RS_{q,k}$ je lineární kód délky $q-1$, jehož slova tvoří evaluace všech polynomů nad F_q až do stupně $k-1$.

Proč je RS kód lineární? Platí totiž, že $[f] + [g] = [f+g]$ a $\alpha[f] = [\alpha f]$, $\alpha \in F_q$, $f, g \in F_q[x]$.

Minimální vzdálenost určíme následující úvahou: Dva různé polynomy f, g stupně $\leq t$ nemůžou nabývat tutéž hodnotu ve více než t bodech. Kdyby tomu tak bylo, měl by polynom $f-g$ více než t kořenů a stupeň $\leq t$. To nejde. Stupně f a g jsou podle definice $\leq k-1$, tedy f se od g liší nejméně na $q-1-(k-1) = q-k$ prvcích z F_q. Najít dvě slova přesně ve vzdálenosti $q-k$ už je snadné, je to například vzdálenost evaluace polynomu s $k-1$ kořeny od slova tvořeného samými nulami.

[1] např. v [2] na str. 85

Tvrzení Kód $RS_{q,k}$ má minimální vzdálenost $q-k$.

Jaká je dimenze RS kódu? Polynomy stupně $\leq k-1$ tvoří lineární prostor dimenze k nad F_q. Jako bázi můžeme vzít množinu $\left\{1, x, x^2, \ldots, x^{k-1}\right\}$. Očekáváme, že prostor jejich evaluací, tedy náš kód, má rovněž dimenzi k. Evaluace polynomu $f = \sum \alpha_i x^i$, který je lineární kombinací polynomů x^i, se rovná $[f] = \sum \alpha_i [x^i]$ podle definice. To znamená, že dimenze $RS_{q,k}$ nemůže být větší než k.

Abychom odhadli dimenzi RS kódu z druhé strany, vezměme matici $M(RS_{q,k})$ s rozměry $k \times (q-1)$, která má v řádcích evaluace

$$[1], [x], \left[x^2\right], \ldots, \left[x^{k-1}\right],$$

tedy $M(RS_{q,k}) = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \beta_1 & \beta_2 & \cdots & \beta_{q-1} \\ \beta_1^2 & \beta_2^2 & \cdots & \beta_{q-1}^2 \\ \vdots & \vdots & & \vdots \\ \beta_1^{k-1} & \beta_2^{k-1} & \cdots & \beta_{q-1}^{k-1} \end{pmatrix}$. Pokud má tato matice hodnost k máme vyhráno. Vybereme prvních k sloupců matice a získáme tak Vandermondovu matici $V(\beta_1, \beta_2, \ldots, \beta_k)$ a ta má nenulový determinant, tedy hodnost k. Navíc můžeme vzít $M(RS_{q,k})$ jako generující matici kódu $RS_{q,k}$.

Tvrzení Dimenze kódu $RS_{q,k}$ je rovna k.

Shrneme poznatky o parametrech kódu $RS_{q,k}$:

Věta Kód $RS_{q,k}$ má parametry $[q-1, k, q-k]$.

Příklad Nad tělesem F_5 lze zkonstruovat následující $[4,2,3]$ RS kód. Délka kódu je určená z předchozí věty a zvolíme-li si minimální vzdálenost 3, dostaneme z téže věty dimenzi 2. Nenulové prvky F_5 seřadíme do posloupnosti , např.$1, 2, 4, 3$. Generující matice je evaluace polynomů 1 a x, tedy $G = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \end{pmatrix}$.

$$\begin{aligned} C = \{ \quad & (0000), (1111), (2222), (3333), (4444), \\ & (1243), (2304), (3410), (4021), (0132), \\ & (3042), (4103), (0214), (1320), (2431), \\ & (4230), (0341), (1402), (2013), (3124), \\ & (0423), (1034), (2140), (3201), (4312) \quad \} \end{aligned}$$

2.2 Singletonova nerovnost a MDS kódy

Abychom si mohli plně uvědomit přednosti parametrů RS kódu, povíme si nyní něco o omezeních parametrů kódů obecně. Přirozeně taková omezení existují. Ideálně bychom si představovali kód s který zvládne opravit co nejvíc chyb, prodlužuje odesílanou zprávu co nejméně a má co nejvíc kódových slov. Uvedené požadavky však nutně jdou proti sobě.

Pro počet kódových slov při dané délce kódu n a minimální vzdálenosti d existuje horní odhad. Označme jej $A(n,d) = \max_{C} \log_q |C|$, neboli maximální počet kódových slov mezi všemi kódy C nad q-prvkovou abecedou, které mají délku n a minimální vzdálenost alespoň d.

Věta (Singletonův odhad). Pro každé $d \leq n$ je $A(n,d) \leq n - d + 1$.[2]

Pro lineární kód platí: $k \leq n - d + 1$. Pro RS kód zde platí rovnost $k = q-1-(q-k)+1$. Je tedy nejlepší z hlediska Singletonovy nerovnosti. Lineární kódy, které splňují Singletonovu nerovnost s rovností se nazývají MDS (Maximum distance separable). RS kódy patří do skupiny MDS kódů.

2.3 Cyklické kódy a jejich základní vlastnosti

Dříve než nadefinujeme BCH kódy, podíváme se na obecnější třídu cyklických kódů. Uvidíme, že BCH kódy do této třídy rovněž patří.

Definice Lineární kód C se nazývá cyklický, pokud do něj s každým slovem $c = (c_0, c_1, \ldots, c_{n-1})$ patří také jeho cyklický posun $(c_{n-1}, c_0, c_1, \ldots, c_{n-2})$.

Pro naše další úvahy bude vhodné reprezentovat kódová slova a ostatní prvky F_q^n jako polynomy. Slovo $a = (a_0, a_1, \ldots, a_{n-1}) \in F_q^n$ vyjádříme jako polynom $a(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1} = \sum_{i=0}^{n-1} a_i x^i$, protože mezi vektorovým prostorem F_q^n a okruhem $R = {}^{F_q[x]}\!/_{(x^n-1)}$ existuje bijekce která zobrazuje slovo a na polynom $a(x)$. V okruhu R, okruhu polynomů s koeficienty z F_q až do stupně $n-1$, počítáme modulo rovnost $x^n = 1$. Není těžké si uvědomit, že cyklický posun slov z F_q^n odpovídá v R násobení polynomem x. Platí totiž

$$\begin{aligned} xa(x) &= x(a_0 + a_1 x + \ldots + a_{n-1}x^{n-1}) \\ &= a_0 x + a_1 x^2 + \cdots + a_{n-2}x^{n-1} + a_{n-1}x^n \\ &= a_{n-1} + a_0 x + a_1 x^2 + \cdots + a_{n-2}x^{n-1} \end{aligned} \tag{2.1}$$

[2] Důkaz věty lze vyhledat např v [4], str. 7.

Tato rovnost hodně napovída o postavení cyklických kódů v okruhu R.

Tvrzení Lineární kód C v F_q^n je cyklický právě tehdy, když C je ideálem okruhu $R = {}^{F_q[x]}\!/_{(x^n-1)}$.

Důkaz $\Leftarrow$: Je- li C ideál v R a $c(x)$ kódové slovo, je $xc(x)$ také kódové slovo a tedy je kód C cyklický podle rovnosti 2.1 .

$\Rightarrow$: Je-li C cyklický, patří do něj spolu s $c(x)$ také $xc(x)$ a $x^i c(x)$ pro každé i. Díky linearitě C sem patří také $a(x)c(x)$ pro každý polynom $a(x)$. To odpovídá definici ideálu.

Tvrzení Cyklický kód C se skládá z násobků polynomu $g(x)$, nenulového polynomu nejnižšího stupně v C. Říkáme, že $g(x)$ je generující polynom, tj $C = (g(x))$. Navíc $g(x)$ dělí $x^n - 1$ v okruhu $F_q[x]$. Tedy R je okruh hlavních ideálů.

Kořeny polynomu $g(x)$ jsou tedy kořeny všech polynomů tvořících kódová slova. Můžeme říci, že jsou to kořeny kódu C.

Důkaz Mějme polynom $c(x)$ v C. Protože F_q je těleso, můžeme v $F_q[x]$ dělit se zbytkem, tedy dostáváme $c(x) = h(x)g(x) + d(x)$, kde $h(x)$, $d(x)$ jsou polynomy a $\deg d(x) < \deg g(x)$. Víme, že $d(x) \in C$, protože $d(x) = c(x) - h(x)g(x)$ a to je kódové slovo. Ovšem předpokládáme, že stupeň $g(x)$ je minimální v kódu C. Má-li tedy být stupeň $d(x)$ nižší, musí být nulový, tedy $d(x) = 0$. Odtud $c(x) \in (g(x))$.

Podobně dostaneme $x^n - 1 = h(x)g(x) + d(x)$, kde $\deg d(x) < \deg g(x)$. V okruhu R tedy $h(x)g(x) + d(x) = 0$. Vídíme, že $d(x)$ patří do kódu, protože $d(x) = -h(x)g(x)$. Z minimality $g(x)$ plyne stejným způsobem jako výše, že $d(x) = 0$.

Skutečnost, že je nějaký kód C cyklický pro nás znamená zjednodušení při hledání jeho generující a paritní matice.

Je-li délka kódu n a vezmeme-li generující polynom $g(x)$ stupně k, vidíme, že slova $g(x), xg(x), \ldots, x^{n-k-1}g(x)$ tvoří bázi kódu. Po složkách zapíšeme $g(x) = \sum_{i=0}^{k} g_i x^i$. Proto je generující matice

$$G = \begin{pmatrix} g_0 & g_1 & \cdots & g_k & 0 & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_{k-1} & g_k & 0 & \cdots & 0 \\ 0 & 0 & \cdots & & & & \cdots & 0 \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & \cdots & g_k \end{pmatrix}.$$

Z matice vyčteme také, že dimenze kódu je $n - k$.

Při hledání paritní matice využijeme polynom $h(x) = \sum_{i=0}^{n-k} h_i x^i$, pro který v $F_q[x]$ platí $h(x)g(x) = x^n - 1$ a v R je $h(x)g(x) = 0$, respektive $\sum_{i=0}^{n-1} \sum_{j=0}^{i} g_j h_{i-j} x^i = 0$. Rozepsáno po složkách to znamená

$$g_0 h_i + g_1 h_{i-1} + \cdots + g_k h_{i-k} = 0 \text{ pro } i = 0, 1, \ldots, n-1. \tag{2.2}$$

Přidáme koeficienty $h_{n-k+1}, \ldots, h_{n-1}$ a $g_{k+1}, \ldots, g_{n-1}$, které jsou všechny nulové a na celkovém součtu nic nezmění. Přesto součiny $h_{n-k+i} g_{k+j} x^{n+i+j}$ bereme modulo n. Představíme-li si, jak bychom mohli součty 2.2 získat v maticovém násobení GH^T, dostaneme matici

$$H = \begin{pmatrix} h_{n-k} & h_{n-k-1} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\ 0 & h_{n-k} & \cdots & h_1 & h_0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & & & & \cdots & 0 \\ 0 & 0 & \cdots & 0 & h_{n-k} & h_{n-k-1} & \cdots & h_0 \end{pmatrix}.$$

Vlastnosti cyklických kódů se příjemně projeví i v dekódování. Umožnují totiž reprezentovat syndrom pomocí polynomu $s(x)$, což je jednodušší na počítání. Syndromový polynom $s(x)$ získáme jako zbytek po dělení přijatého slova $y(x) = c(x) + e(x)$ generujícím polynomem $g(x)$,

$$s(x) = y(x) \bmod g(x) = e(x) \bmod g(x), \tag{2.3}$$

protože $g(x)$ dělí všechna kódová slova.

2.4 Minimální polynomy a další pojmy z konečných těles

Definice Mějme prvek γ v tělese F_{q^m}. Minimální polynom $M_\gamma(x)$ prvku γ je monický polynom v $F_q[x]$ nejnižšího stupně splňující $M_\gamma(\gamma) = 0$.

Zrekapitulujme v krátkosti, co víme z teorie konečných těles. Konečné těleso může mít q prvků, kde q je nějaká mocnina prvočísla. Prvky tělesa F_q jsou zároveň všechny kořeny polynomu $x^q - x$. Důležitý poznatek z teorie konečných těles říká, že multiplikativní grupa tělesa F_q je cyklická, tudíž jí lze nagenerovat jedním prvkem.[3]

Definice Prvek $\xi \in F_q$, který generuje multiplikativní grupu F_q^*, se nazývá primitivní prvek tělesa F_q.

[3] Důkaz tvrzení lze najít např. v [2], str. 9.

Definice Prvek $\alpha \in F_q$ takový, že $\alpha^k = 1$, ale $\alpha^l \neq 1$ pro $0 < l < k$ se nazývá primitivní k-tá odmocnina z jednotky.

Primitivní prvek ξ tělesa F_q je tedy zároveň primitivní $(q-1)$-tá odmocnina z jednotky, protože prvek $1 = \xi^{q-1}$ musí být nagenerovaný jako poslední. Pokud k dělí $q-1$, pak je $\xi^{\frac{q-1}{k}}$ primitivní k-tá odmocnina z jednotky.

V pozdějších důkazech se nám bude hodit také následující jednoduché lemma.

Lemma Buď $\alpha \in F_{q^m}$ nějaký kořen polynomu $x^n - 1$, tedy n-tá odmocnina z jednotky. Potom

$$\sum_{i=0}^{n-1} \alpha^i = \begin{cases} 0 & \alpha \neq 1 \\ n & \alpha = 1 \end{cases}$$

Důkaz Pokud $\alpha = 1$ je součet jistě roven n. Ve druhém případě využijeme $\sum_{i=0}^{n-1} \alpha^i = \frac{1-\alpha^n}{1-\alpha} = 0$.

2.5 BCH-kódy

O cyklických kódech jsme dokázali, že všechna kódová slova jsou násobkem generujícího polynomu $g(x)$. Odtud je už jen krůček ke konstrukci kódu tak, že si generující polynom speciálně vybereme, respektive si speciálně vybereme jeho kořeny.

Stále se pohybujeme nad tělesem F_q. Kódová slova ztotožnujeme s polynomy v okruhu $R = {}^{F_q[x]}\!/_{(x^n-1)}$. Jejich kořeny, to znamená všechny potenciální kořeny kódu, najdeme v rozkladovém nadtělese polynomu $x^n - 1$ nad F_q, které budeme značit F_{q^m}. Kvůli jednoznačnosti je pro nás důležitá podmínka, aby $\mathrm{NSD}(n, q) = 1$. Jako m pak stačí vzít nejmenší celé kladné číslo s vlastností $q^m \equiv 1 \pmod{n}$, protože pak víme, že n dělí $q^m - 1$. Tedy můžeme v F_{q^m} najít cyklickou podgrupu řádu n generovanou prvkem $\alpha = \xi^{\frac{q^m-1}{n}}$. Tedy α je n-tá odmocnina z jedné v F_{q^m}, stejně jako všechny mocniny α, protože $\left(\alpha^k\right)^n = \left(\alpha^n\right)^k = 1$. Každá z nich je proto také kořenem $x^n - 1$. Našli jsme rozklad $x^n - 1 = (x-1)(x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{n-1})$ nad F_{q^m}. Odtud je také vidět, že $x^n - 1$ nemá žádné vícenásobné kořeny.

Věta (Hranice BCH) Mějme cyklický kód C generovaný polynomem $g(x)$, který mezi svými kořeny obsahuje posloupnost $\delta - 1$ po sobě

jdoucích mocnin nějakého prvku α, jenž je primitivní n-tou odmocninou z jednotky. Tedy pro jistá $b \geq 0$, $\delta \geq 0$, $b, \delta \in \mathbb{Z}$ máme

$$g(\alpha^b) = g(\alpha^{b+1}) = \cdots = g(\alpha^{b+\delta-2}) = 0.$$

Potom je minimální vzdálenost kódu alespoň δ.

Důkaz Máme-li kódové slovo $c(x) = \sum_{i=0}^{n-1} c_i x^i$, pak pro něj platí, že $c\left(\alpha^b\right) = c\left(\alpha^{b+1}\right) = \ldots = c\left(\alpha^{b+\delta-2}\right) = 0$.

Tedy $cH^T = 0$, kde $H = \begin{pmatrix} 1 & \alpha^b & \alpha^{2b} & \ldots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \ldots & \alpha^{(n-1)(b+1)} \\ \ldots & & & & \ldots \\ 1 & \alpha^{b+\delta-2} & \alpha^{2(b+\delta-2)} & \ldots & \alpha^{(n-1)(b+\delta-2)} \end{pmatrix}$.
To platí i tehdy, když v H vynecháme některé řádky. V důkazu využijeme lemma 1.2 a ukážeme, že jakýchkoli $\delta-1$ nebo méně sloupců H je lineárně nezávislých nad F_{q^m}. Vybereme $w \leq \delta - 1$ sloupců z H a označíme je indexy $\{a_1, a_2, \ldots, a_w\}$. Zbyla nám tedy matice

$$\begin{pmatrix} \alpha^{a_1 b} & \ldots & \alpha^{a_w b} \\ \alpha^{a_1(b+1)} & \ldots & \alpha^{a_w(b+1)} \\ \ldots & & \ldots \\ \alpha^{a_1(b+w-1)} & \ldots & \alpha^{a_w(b+w-1)} \end{pmatrix}$$

a my ukážeme , že její determinant je nenulový. Z matice vytkneme $\alpha^{(a_1+a_2+\cdots+a_w)b}$ a zůstane Vandermondův determinant

$$\det \begin{pmatrix} 1 & \cdots & 1 \\ \alpha^{a_1} & \cdots & \alpha^{a_w} \\ \vdots & & \vdots \\ \alpha^{a_1(w-1)} & \ldots & \alpha^{a_w(w-1)} \end{pmatrix}.$$

Věta Minimální vzdálenost cyklického kódu délky n s kořeny

$$\alpha^b, \alpha^{b+r}, \alpha^{b+2r}, \ldots, \alpha^{b+(\delta-2)r}$$

je, pokud jsou čísla n a r nesoudělná, alespoň δ.

Důkaz Vezměme $\beta = \alpha^r$. Jsou-li r a n nesoudělná, víme, že β je také primitivní n-tá odmocnina z 1. Obě generují stejnou multiplikativní grupu, takže existuje t splňující $\alpha^b = \beta^t$. Tímto způsobem se dají kořeny zapsat jako $\beta^t, \beta^{t+1}, \cdots, \beta^{t+\delta-2}$ a to je situace kterou řeší předchozí věta.

Zjistili jsme, že volbou kořenů generujícího polynomu jako ve větách výše, dostaneme kód, jehož minimální vzdálenost určitě nebude menší, než námi zvolené číslo δ.

Definice Cyklický kód délky n nad tělesem F_q se nazývá BCH kód se zadanou vzdáleností δ, je-li generován polynomem

$$g(x) = nsn\left(M_{\alpha^b}(x), M_{\alpha^{b+1}}(x), \ldots, M_{\alpha^{b+\delta-2}}(x)\right),$$

kde $b \geq 0$ celé a α je primitivní n-tá odmocnina z jednotky v F_{q^m} a $M_{\alpha^b}(x)$ je minimální polynom prvku α^b.

To znamená, že $g(x)$ má kořeny $\alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+\delta-2}$ a zároveň je monický a má nejnižší možný stupeň. Slovo c patří do kódu, právě když $c\left(\alpha^b\right) = c\left(\alpha^{b+1}\right) = \ldots = c\left(\alpha^{b+\delta-2}\right) = 0$.

Chceme-li zdůraznit, který kónkrétní BCH kód máme na mysli, píše se také $\mathrm{BCH}_{q,m,\delta}$.

Je-li $b = 1$, mluvíme obvykle o BCH kódech v užším smyslu. V případě, že $n = q^m - 1$ není α v tělese F_{q^m} pouze primitivní n-tou odmocninou z jednotky, ale přímo primitivním prvkem. Proto takové kódy nazýváme primitivní.

Prvky z tělesa F_{q^m} lze reprezentovat jako vektor pomocí m-tice prvků z tělesa F_q, pokud si vhodně zvolíme bázi. Nabízí se nám třeba $1, \alpha, \ldots, \alpha^{m-1}$.

Touto cestou získáme z $H = \begin{pmatrix} 1 & \alpha^b & \alpha^{2b} & \ldots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \ldots & \alpha^{(n-1)(b+1)} \\ \ldots & & & & \ldots \\ 1 & \alpha^{b+\delta-2} & \alpha^{2(b+\delta-2)} & \ldots & \alpha^{(n-1)(b+\delta-2)} \end{pmatrix}$ paritní matici. Každý prvek matice nahradíme m-prvkovým sloupcovým vektorem z F_q. Tak získáme $m(\delta - 1)$ řádků, které ovšem nemusí být lineárně nezávislé. Odtud také získáme odhad pro dimenzi kódu, $k = n -$ hodnost matice H. Tedy $k \geq n - m(\delta - 1)$.

Poznatky o BCH kódech shrneme ve větě:

Věta Máme BCH kód délky n nad tělesem F_q se zadanou vzdáleností δ. Jeho minimální vzdálenost je $d \geq \delta$ a dimenze $k \geq n - m(\delta - 1)$.

Příklad Nad tělesem $F_5 = \{0, 1, 2, 2^2 = 4, 2^3 = 3\}$ najdeme BCH kód délky 4 se zadanou vzdáleností 3. Rozkladové nadtěleso polynomu $x^4 - 1$ je opět F_5 a jako primitivní prvek můžeme vzít $\alpha = 2$. Generující polynom vezmeme například pro $b = 1$

$$g(x) = \mathrm{nsn}\left(M_\alpha, M_{\alpha^2}\right) = nsn(x+3, x+1) = x^2 + 4x + 3.$$

Generující matice je pak $G = \begin{pmatrix} 3 & 4 & 1 & 0 \\ 0 & 3 & 4 & 1 \end{pmatrix}$ a dimenze kódu 2.

2.6 Mattson-Solomonovy polynomy

Slovo $a = (a_0, a_1, \ldots, a_{n-1})$ jsme dosud reprezentovali, jako polynom $a(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1} = \sum_{i=0}^{n-1} a_i x^i$. Nyní si představíme jiný polynom spřízněný s a, a to Mattson - Solomonův polynom $A(z)$.

Vektor a v definici je vzat obecněji z F_{q^m}, aby zůstalo zachováno značení, nicméně vektory kódových slov bereme nad F_q. Stále platí, že n dělí $q^m - 1$, tedy α je primitivní n-tá odmocnina z jednotky v F_{q^m}.

Definice Mattson-Solomonův (MS) polynom asociovaný s vektorem $a = (a_0, a_1, \ldots, a_{n-1})$, $a_i \in F_{q^m}$, respektive s polynomem $a(x)$, je polynom $A(z) = \sum_{j=0}^{n-1} A_j z^{n-j}$ v $F_{q^m}[z]$, kde členy

$$A_j = a\left(\alpha^j\right) = \sum_{i=0}^{n-1} a_i \alpha^{ij}.$$

Vzhledem k tomu, že α je n-tá odmocnina z jednotky, můžeme koeficienty A_j uvažovat modulo n, protože $A_n = a(\alpha^n) = a(1) = a(\alpha^0) = A_0$, a tak dále.

Poznamenejme ještě, že v případech kdy $a_i \in F_q$, platí

$$(A_j)^q = A_{jq} \text{ pro všechny } j = 0, \ldots, n-1, \tag{2.4}$$

protože v aritmetice nad F_q platí

$$(A_j)^q = \left(\sum_{i=0}^{n-1} a_i \alpha^{ij}\right)^q = \sum_{i=0}^{n-1} a_i^q \alpha^{ijq} = \sum_{i=0}^{n-1} a_i \alpha^{ijq} = A_{jq}.$$

Tato nevinná poznámka hraje klíčovou roli při zjednodušování algoritmů pro binární kódy, ale o tom až později.

Také jsme získali novou možnost, jak vyjádřit BCH kódy v užším smyslu se zadanou vzdáleností δ, totiž jako vektory a pro které je $A_1 = A_2 = \cdots = A_{\delta-1} = 0$.

Pro zobrazení přiřazující $a(x)$ polynom $A(z)$ se používá také název diskrétní Fourierova transformace.

V další větě zjistíme, jak se od MS polynomu $A(z)$ vrátit zpět k $a(x)$.

Věta (Inverzní formule) MS polynom $A(z)$ převedeme na vektor a pomocí vztahů $a_i = \frac{1}{n} A(\alpha^i)$ pro $i = 0, \ldots, n-1$, tedy

$$a = \frac{1}{n}\left(A(1), A(\alpha), \ldots, A\left(\alpha^{n-1}\right)\right) \tag{2.5}$$

a polynom $a(x) = \frac{1}{n} \sum_{i=0}^{n-1} A(\alpha^i) x^i$.

Důkaz Provedeme následující úpravy

$$A\left(\alpha^i\right) = \sum_{j=0}^{n-1} A_j \left(\alpha^i\right)^{n-j} = \sum_{j=0}^{n-1} A_j \alpha^{i(n-j)} = \sum_{j=0}^{n-1} A_j \alpha^{-ij},$$

protože $\alpha^n = 1$. Členy A_j ve výrazu rozepíšeme podle definice. Dostaneme

$$\begin{aligned}\sum_{j=0}^{n-1} a\left(\alpha^j\right)\alpha^{-ij} &= \sum_{j=0}^{n-1}\sum_{k=0}^{n-1} a_k \left(\alpha^j\right)^k \alpha^{-ij} = \\ &= \sum_{k=0}^{n-1} a_k \sum_{j=0}^{n-1} \alpha^{j(k-i)} = \sum_{k=0}^{n-1} a_k \sum_{j=0}^{n-1} \left(\alpha^{k-i}\right)^j .\end{aligned}$$

Na tomto místě použijeme lemma v kapitole 2.4, čímž vypadnou všechny sčítance, kde $k \neq i$. Máme tedy $\sum_{j=0}^{n-1}\left(\alpha^{k-i}\right)^j = n$ pokud $k = i$ a druhá sumace se rovněž zredukovala na jediný člen, což znamená $\sum_{k=0}^{n-1} a_k \sum_{j=0}^{n-1}\left(\alpha^{k-i}\right)^j = n a_i$. Tedy $A\left(\alpha^i\right) = n a_i$. To stačí vydělit n a je dokázáno.

2.7 Vztah BCH a RS kódů

Ačkoli jsme obě rodiny kódů definovali různým způsobem, dostali jsme v ilustračním příkladu pokaždé stejný kód. To není náhoda. V MS polynomech jsme získali nástroj, jak určit jejich vzájemný vztah.

Věta Vezměme q nějakou mocninu prvočísla, $m \geq 1$, $n = q^m - 1$, $1 \leq \delta \leq n$, primitivní prvek $\alpha \in F_{q^m}$ a kódy s těmito parametry

$$RS_{q^m, q^m-\delta} = \left\{\left(f(1), f(\alpha), \ldots, f\left(\alpha^{n-1}\right)\right)\right\} \subseteq \left(F_{q^m}\right)^n$$

$$BCH_{q,m,\delta} = \left\{(a_o, \ldots, a_{n-1}) \in F_q | a(\alpha) = \cdots = a\left(\alpha^{\delta-1}\right) = 0\right\}$$

Potom

$$BCH_{q,m,\delta} = RS_{q^m,q^m-\delta} \cap \left(F_q\right)^n . \tag{2.6}$$

Ve speciálním případě, kdy $m = 1$, platí

$$BCH_{q,1,\delta} = RS_{q,q-\delta} \tag{2.7}$$

Důkaz Začneme speciálním případem. Nenulové prvky tělesa F_q seřadíme do posloupnosti $\alpha, \alpha^2, \ldots, \alpha^{n-1=q-2}$, v tomto případě $\alpha \in F_q$.

Slovo $(f(1), f(\alpha), \ldots, f(\alpha^{n-1}))$ z kódu $RS_{q,q-\delta}$ označíme

$$(a_o, \ldots, a_{n-1}) = \left(f(1), f(\alpha), \ldots, f\left(\alpha^{n-1}\right)\right)$$

a budeme se snažit dokázat, že patří do kódu $BCH_{q,1,\delta}$. Musí tedy splňovat, že $a(\alpha^1) = \cdots = a\left(\alpha^{\delta-1}\right) = 0$. Ve vyjádření pomocí MS polynomů tedy $A_1 = \cdots = A_{\delta-1} = 0$, protože $A_i = a(\alpha^i)$ podle definice. Pomocí inverzní formule dostáváme zpátky $a_i = \frac{1}{n}A(\alpha^i)$, a také vztah $f(\alpha^i) = \frac{1}{n}A(\alpha^i)$. Počítáme nad tělesem F_q, tedy $\frac{1}{n} = -1$, protože $n = q - 1$. Najdeme vztah přímo mezi jednotlivými koeficienty. Platí rovnost lineárních kombinací

$$-A\left(\alpha^i\right) = -\sum_{j=0}^{n-1} A_j \left(\alpha^i\right)^{n-j} = f\left(\alpha^i\right) = \sum_{j=0}^{n-1} f_j \left(\alpha^i\right)^j$$

a navíc matice

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha^n & \alpha^{n-1} & \cdots & \alpha^1 \\ \alpha^{2n} & \alpha^{2(n-1)} & \cdots & \alpha^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{(n-1)n} & \alpha^{(n-1)(n-1)} & \cdots & \alpha^{(n-1)} \end{pmatrix},$$

kterou používáme, je Vandermondova a má tedy nenulový determinant. Tedy také $f_j = -A_{n-j}$, pro $j = 0, \ldots, n-1$.

Podle definice RS kódů je $\deg f \leq q - \delta - 1$. Proto také koeficienty $f_{q-\delta} = f_{q-\delta+1} = \cdots = f_{q-2} = 0$, $q = n-1$ a s nimi i $A_{1=q-1-(q-2)} = \cdots = A_{\delta-1=q-1-(q-\delta)} = 0$, což jsme potřebovali dokázat. Tím jsme dostali $BCH_{q,1,\delta} \supseteq RS_{q,q-\delta}$.

Podíváme se na dimenze obou kódů. Víme, že $\dim RS_{q,q-\delta} = q - \delta$. Kód $BCH_{q,1,\delta}$ je cyklický a generovaný polynomem stupně $\delta - 1$ (Má $\delta - 1$kořenů). Tedy

$$\dim BCH_{q,1,\delta} = n - (\delta - 1) = q - 1 - (\delta - 1) = q - \delta.$$

Dimenze obou kódů jsou stejné, musí se tedy shodovat.

V obecném případě použijeme, co jsme dokázali pro speciální případ, totiž že $BCH_{q^m,1,\delta} = RS_{q^m,q^m-\delta}$. Nyní je $\alpha \in F_{q^m}$. Uvědomíme

si, že

$$\begin{aligned} BCH_{q^m,1,\delta} &= \left\{(a_o,\ldots,a_{n-1}) \in F_{q^m} | a(\alpha) = \cdots = a\left(\alpha^{\delta-1}\right) = 0\right\} \\ BCH_{q,m,\delta} &= \left\{(a_o,\ldots,a_{n-1}) \in F_q | a(\alpha) = \cdots = a\left(\alpha^{\delta-1}\right) = 0\right\} \end{aligned}$$

takže abychom získali kód $BCH_{q,m,\delta}$ z $BCH_{q^m,1,\delta}$ musíme vybrat slova ležící nad F_q. Tedy

$$BCH_{q,m,\delta} = BCH_{q^m,1,\delta} \cap (F_q)^n = RS_{q^m,q^m-\delta} \cap (F_q)^n .$$

Z této věty mimo jiné vyplývá, že RS kódy jsou cyklické a že druhou možností jak RS kódy nadefinovat je právě jako podtřídu BCH kódů délky $n = q - 1$.

Kapitola 3

Dekódování

3.1 Lokační polynom a Newtonovy identity

Pro dekódování je klíčové určit pozici, kde nastala chyba. Nadefinujeme si polynom, který nám později bude pro tento účel sloužit.

Představme si, že vektor $a = (a_0, a_1, \ldots, a_{n-1})$, $a_i \in F_q$ má váhu $w(a) = w$, a označme si jeho nenulové složky jako $a_{i_1}, a_{i_2}, \ldots, a_{i_w}$. Vyjádříme si a jako seznam dvojic $(X_1, Y_1), \ldots, (X_w, Y_w)$, kde X-ové pozice ponesou unformaci o umístění nenulových složek a a Y-ové o jejich číselné hodnotě. Tedy k $a_{i_1}, a_{i_2}, \ldots, a_{i_w}$ přiřadíme $X_1 = \alpha^{i_1}, \ldots, X_w = \alpha^{i_w}$, $X_1, \ldots, X_w \in F_{q^m}$ takzvané lokátory (locators) a $Y_1 = a_{i_1}, \ldots, Y_w = a_{i_w}$, $Y_1, \ldots, Y_w \in F_q$, α nadále značí primitivní n-tou odmocninu z jednotky v F_{q^m}. V případě, že je a binární vektor, je hodnota všech Y_i rovna 1.

Nepřehlédněme skrytou souvislost s MS polynomy, totiž že

$$\sum_{k=1}^{w} Y_k X_k^j = \sum_{k=1}^{w} a_{i_k} \left(\alpha^{i_k}\right)^j = \sum_{i=0}^{n-1} a_i \alpha^{ij} = a(\alpha^j) = A_j,$$

protože všechny ostatní sčítance jsou nulové.

Definice Lokační polynom (locator polynomial) vektoru a definujeme jako

$$\sigma(z) = \prod_{i=1}^{w} (1 - X_i z) = \sum_{i=0}^{w} \sigma_i z^i,$$

poslední člen $\sigma_0 = 1$.

Z předpisu vídíme, že jako kořeny $\sigma(z)$ se objeví převrácené hodnoty lokátorů $\frac{1}{X_i}$. Jednotlivé koeficienty σ_i si můžeme vyjádřit pomocí hodnot

X_i následovně

$$\begin{aligned}\sigma_1 &= -(X_1 + \cdots + X_w)\\ \sigma_2 &= X_1X_2 + X_1X_3 + \cdots + X_{w-1}X_w\\ \cdots &\ \cdots\ \cdots\\ \sigma_w &= (-1)^w X_1 \cdots X_w\end{aligned}$$

U koeficitentu σ_k sčítáme součiny k různých X_i, při lichém k, je znaménko záporné.

Nepřekvapí nás závislost mezi koeficienty σ_i a MS koeficienty A_i. Ukážeme, že je lze najít pomocí soustavy lineárních rovnic.

Věta (Obecný tvar Newtonových identit) MS koeficienty A_j splňují pro každé $j = 0, \ldots, n-1$ rekurentní předpis

$$A_{j+w} + \sigma_1 A_{j+w-1} + \cdots + \sigma_w A_j = 0. \tag{3.1}$$

V maticovém zápisu pro $j = 0, \ldots, w-1$ má

$$\begin{pmatrix} A_{w-1} & A_{w-2} & \cdots & A_0 \\ A_w & A_{w-1} & \cdots & A_1 \\ \cdots & \cdots & \cdots & \cdots \\ A_{2w-2} & A_{2w-3} & \cdots & A_{w-1} \end{pmatrix} \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_w \end{pmatrix} = - \begin{pmatrix} A_w \\ A_{w+1} \\ \vdots \\ A_{2w-1} \end{pmatrix}. \tag{3.2}$$

Důkaz Do rovnice $\prod_{i=1}^{w}(1 - X_i z) = 1 + \sigma_1 z + \cdots + \sigma_w z^w$ dosadíme za z postupně všechny $\frac{1}{X_i}$ pro $i = 1, \ldots, w$ a přenásobíme ji členem $Y_i X_i^{j+w}$. Tím dostaneme w rovnic

$$Y_i X_i^{j+w} + \sigma_1 Y_i X_i^{j+w-1} + \cdots + \sigma_w Y_i X_i^j = 0.$$

Sečteme-li jednotlivé rovnice máme

$$\sum_{i=1}^{w} Y_i X_i^{j+w} + \sum_{i=1}^{w} Y_i X_i^{j+w-1}\sigma_1 + \cdots + \sum_{i=1}^{w} Y_i X_i^j \sigma_w = 0,$$

což je požadovaná rekurence, neboť $\sum_{i=1}^{w} Y_i X_i^j = A_j$.

Pro A_{w+j}-tý MS koeficient máme tedy vyjádření $A_{w+j} = -\sum_{i=1}^{w} \sigma_i A_{j+w-i}$, $j = 1, \ldots, n-w$.

Pro binární případy platí jednodušší verze této věty, takzvaná běžná forma Newtonových identit. Zjednodušení spočívá v tom, že můžeme zapomenout na členy Y_i, protože nabývají pouze hodnot 0 a 1. Místo

$A_j = \sum_{i=1}^{w} Y_i X_i^j$ nám tedy zbyde pouze $P_k = \sum_{r=1}^{w} X_r^k$. Běžná forma Newtonových identit vyplyne z následujících pozorování[1].

Lemma Označíme si $P_k = \sum_{r=1}^{w} X_r^k$ pro všechna k a dále $P(z) = \sum_{k=1}^{\infty} P_k z^k$. Pak platí $\sigma(z)P(z) + z\sigma\prime(z) = 0$.

Důkaz Nejprve si vyjádřím $z\sigma\prime(z)$. Derivace součinu $(fg)\prime = fg\prime + f\prime g$ mi dává $\sigma\prime(z) = \sum_{i=1}^{w} \prod_{j=1,j\neq i}^{w} -X_i(1 - X_j z)$ respektive

$$z\sigma\prime(z) = \sum_{i=1}^{w} \prod_{j=1,j\neq i}^{w} -X_i z(1 - X_j z).$$

S výrazem $\sigma(z)P(z)$ pracuji jako s formální mocninnou řadou.

$$\begin{aligned}
\sigma(z)P(z) &= (\prod_{j=1}^{w}(1 - X_j z))(\sum_{k=1}^{\infty} P_k z^k) = \sum_{k=1}^{\infty} P_k z^k \prod_{j=1}^{w}(1 - X_j z) = \\
&= \sum_{k=1}^{\infty}(\sum_{r=1}^{w} X_r^k) z^k \prod_{j=1}^{w}(1 - X_j z) = \sum_{k=1}^{\infty}\sum_{r=1}^{w}(X_r^k z^k \prod_{j=1}^{w}(1 - X_j z)) = \\
&= \sum_{r=1}^{w}\sum_{k=1}^{\infty}(X_r^k z^k \prod_{j=1}^{w}(1 - X_j z)) = \sum_{r=1}^{w}\prod_{j=1}^{w}(1 - X_j z)\sum_{k=1}^{\infty}(X_r^k z^k)
\end{aligned}$$

Sumu podle k umíme sečíst jako geometrickou řadu $\sum_{k=1}^{\infty} q^k = \frac{q}{1-q}$ a dostáváme

$$\sum_{r=1}^{w}\prod_{j=1}^{w}(1 - X_j z)\frac{X_r z}{1 - X_r z} = \sum_{r=1}^{w}\prod_{j=1,j\neq r}^{w} X_r z(1 - X_j z),$$

odkud už je vidět, žeσ$(z)P(z) + z\sigma\prime(z) = 0$.

Lemma Dále pak platí

$$\begin{aligned}
P_1 + \sigma_1 &= 0 \\
P_2 + \sigma_1 P_1 + 2\sigma_2 &= 0 \qquad (3.3) \\
&\cdots \\
P_w + \sigma_1 P_{w-1} + \cdots + \sigma_{w-1} P_1 + w\sigma_w &= 0
\end{aligned}$$

a pro $i > w$

$$P_i + \sigma_1 P_{i-1} + \cdots + \sigma_w P_{i-w} = 0 \qquad (3.4)$$

[1] Důkazy následujících dvou lemmat jsou řešením problému (52) z knihy [1] na str. 245

Důkaz Z minulého lemmatu víme $\sigma(z)P(z) + z\sigma\prime(z) = 0$. Tuto rovnost rozepíšeme jako součet řad.

Máme $\sigma(z) = \sum_{i=0}^{w} \sigma_i z^i$, položme $\sigma_i = 0$ pro $i > w$, čímž rozšíříme sumu do ∞. Dále $P(z) = \sum_{i=1}^{\infty} P_i z^i$, položme $P_0 = 0$. Pomocí součinu řad:

$$P(z)\sigma(z) = \sum_{i=0}^{\infty} \sum_{j=0}^{i} P_j \sigma_{i-j} z^i = \sum_{i=1}^{\infty} \sum_{j=1}^{i} P_j \sigma_{i-j} z^i$$

Druhý člen

$$z\sigma\prime(z) = \sum_{i=1}^{w} i\sigma_i z^i$$

Tedy

$$\sigma(z)P(z) + z\sigma\prime(z) = \sum_{i=1}^{\infty} \sum_{j=1}^{i} P_j \sigma_{i-j} z^i + \sum_{i=1}^{w} i\sigma_i z^i = 0,$$

což nastane právě když bude nulový koeficient u každého ze sčítanců z^i, to jest

$$\sum_{j=1}^{i} P_j \sigma_{i-j} + i\sigma_i = 0$$

pro $i \leq w$ a

$$\sum_{j=1}^{i} P_j \sigma_{i-j} = 0$$

pro $i > w$.

Tvrzení (Newtonovy identity - běžná forma) V binárním případě, tedy nad tělesem F_2, se maticový zápis Newtonových identit zjednoduší na

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ A_2 & A_1 & 1 & 0 & 0 & \cdots & 0 \\ A_4 & A_3 & A_2 & A_1 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ A_{2w-2} & A_{2w-3} & \cdots & \cdots & \cdots & \cdots & A_{w-1} \end{pmatrix} \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \vdots \\ \sigma_w \end{pmatrix} = \begin{pmatrix} A_1 \\ A_3 \\ A_5 \\ \vdots \\ A_{2w-1} \end{pmatrix}.$$

Důkaz [2]Vybereme liché řádky ze soustavy rovnic 3.3, 3.4 a osamocený člen P_i převedeme na pravou stranu. Víme už, že členy P_i a A_i

[2]Důkaz tvrzení je řešením problému (54) z knihy [1], str 245.

si v binárním případě odpovídají. Navíc se poslední sčítanec na levé straně $i\sigma_i$ zredukuje v lichých řádcích pouze na σ_i. Zbyde tedy soustava

$$\begin{aligned} \sigma_1 &= A_1 \\ \sigma_1 A_2 + \sigma_2 A_1 + \sigma_3 &= A_3 \\ \sigma_1 A_4 + \sigma_2 A_3 + \sigma_3 A_2 + \sigma_4 A_1 + \sigma_5 &= A_5 \\ &\dots \\ \sigma_1 A_{2w-2} + \dots + \sigma_w A_{w-1} &= A_{2w-1}, \end{aligned}$$

jejíž každý řádek je produktem maticového násobení jako ve znění tvrzení.

Co se stalo se sudými řádky v soustavě rovnic? Členy $A_2, A_4, \dots$ nemusíme počítat pomocí rekurze, ale díky postřehu 2.4 je dostaneme rovnou jako druhou mocninu A_i, protože $A_{2i} = (A_i)^2$ nad F_2.

Předešlé úvahy budeme pro účely dekódování aplikovat na chybový vektor e a jeho MS polynom $E(z)$.

Dekódování založené na lokačním polynomu (locator decoding) a Newtonových identitách není jedinou možností, jak k celému procesu přistupovat[3]. Nabízí se otázka, proč je v případě BCH a RS kódů výhodné použít právě tento způsob. Volbu ozřejmí připomeneme-li, že obě rodiny kódů obsahují mezi svými kořeny posloupnost $\delta - 1$ kořenů. Až doplníme mezikrok 3.5, zjistíme, že v rovnicích 3.2 známe právě $\delta - 1 = 2t$ koeficientů A_j respektive E_j. A to je akorát ten správný počet, abychom mohli soustavu vyřešit a najít lokační polynom, protože $t \geq w$ je maximální možný počet chyb. U BCH kódů může být sice skutečná minimální vzdálenost $d \geq \delta$, ovšem to už je nad síly lokačního polynomu.

3.2 Evaluační polynom

V předchozím odstavci jsme reprezentovali vektor $a = (a_0, a_1, \dots, a_{n-1})$, $a_i \in F_q$ pomocí dvojic $(X_1, Y_1), \dots, (X_w, Y_w)$, kde $a_{i_1}, a_{i_2}, \dots, a_{i_w}$ jsou nenulové složky a. Nyní zavedeme polynom, který využijeme při dekódování $Y-$ových složek.

Definice Evaluační polynom (evaluator polynomial) je definován jako

$$\omega(z) = \sigma(z) + \sum_{i=1}^{w} z X_i Y_i \prod_{j=1, j\neq i}^{w} (1 - X_j z).$$

[3] Více v knize [3]

Jak z takového polynomu vyčteme Y-ové složky?

Dosadíme - li za z hodnotu $\frac{1}{X_k}$, $1 \leq k \leq w$ dostaneme

$$\begin{aligned}\omega\left(\frac{1}{X_k}\right) &= \sigma\left(\frac{1}{X_k}\right) + \sum_{i=1}^{w} \frac{1}{X_k} X_i Y_i \prod_{j=1, j\neq i}^{w} \left(1 - X_j \frac{1}{X_k}\right) = \\ &= 0 + \frac{1}{X_k} X_k Y_k \prod_{j=1, j\neq k}^{w} \left(1 - X_j \frac{1}{X_k}\right).\end{aligned}$$

Všechny ostatni sčítance jsou nulové. Y_k tedy vypočítáme jako

$$Y_k = \frac{\omega\left(\frac{1}{X_k}\right)}{\prod_{j=1, j\neq k}^{w} \left(1 - X_j \frac{1}{X_k}\right)}.$$

Výraz pod zlomkovou čárou se dá dále zjednodušit, použijeme-li $\sigma\prime(z) = \sum_{i=1}^{w} -X_i \prod_{j=1, j\neq i}^{w}(1 - X_j z)$, což se po dosazení $\frac{1}{X_k}$ zredukuje na $-X_k \prod_{j=1, j\neq k}^{w}(1 - X_j \frac{1}{X_k})$. Máme tedy $Y_k = -\frac{X_k \omega\left(\frac{1}{X_k}\right)}{\sigma\prime(\frac{1}{X_k})}$.

Věta Mějme formální mocninnou řadu $Q(z) = \sum_{i=1}^{\infty} A_i z^i$. Potom $\omega(z) = (1 + Q(z))\,\sigma(z)$.

Z koeficientů A_i jsme utvořili formální mocninnou řadu, abychom mohli v důkazu využít vlastnosti jejího součtu, ovšem k určení $\omega(z)$ stačí pouze koeficienty $A_1, \ldots, A_w$, protože $\deg \omega(z) \leq \deg \sigma(z) = w$, kde w stále označuje počet chyb, které nastaly při přenosu.

Důkaz

$$\begin{aligned}\frac{\omega(z)}{\sigma(z)} &= 1 + \sum_{i=1}^{w} \frac{z X_i Y_i}{1 - z X_i} = 1 + \sum_{i=1}^{w} Y_i \sum_{j=1}^{\infty} (z X_i)^j = \\ &= 1 + \sum_{j=1}^{\infty} z^j \sum_{i=1}^{w} Y_i X_i^j = 1 + Q(z)\end{aligned}$$

3.3 Jednotlivé fáze dekódování

Proces dekódování můžeme pro přehlednost rozdělit do několika postupových cílů. Na začátku máme přijaté slovo $y = c + e$, neboli součet nějakého kódového slova a neznámého chybového vektoru.

1. Výpočet syndromu - z kapitoly 1.3 víme, že informace o chybovém vektoru e zůstane obsažená v syndromu s.

2. Hledání lokačního polynomu $\sigma(z)$ - využijeme znalost Newtonových identit a získáme tak polynom, který svými kořeny lokalizuje nenulové složky chybového vektoru e. V této části dekódování je největší prostor pro invenci a zlepšování a také se jí explicitně algoritmy nejvíce věnují, ostatní části mohou mít společné.

3. Určování kořenů lokačního polynomu a tedy chybových pozic

4. Hledání evaluačního polynomu a s ním také velikosti chyb, čímž dáme dohromady celý chybový vektor e.

Konkrétně si všemi těmito kroky projdeme s Petersonovým algoritmem v následující kapitole.

3.4 Petersonův algoritmus

Přijali jsme slovo $y = c + e$ a naším cílem je odhalit chybový vektor e.

V popisu budeme zmíněné vektory reprezentovat pomocí polynomů $y(x), c(x), e(x)$, ale hlavně pomocí jejich MS polynomů $V(z), C(z), E(z)$.

Syndrom se dá vypočítat klasicky podle definice 1.3. Místo paritní matice můžeme vzít

$$H = \begin{pmatrix} 1 & \alpha^{b} & \alpha^{2b} & \dots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \dots & \alpha^{(n-1)(b+1)} \\ \dots & & & & \dots \\ 1 & \alpha^{b+\delta-2} & \alpha^{2(b+\delta-2)} & \dots & \alpha^{(n-1)(b+\delta-2)} \end{pmatrix},$$

která sice není nad F_q, ale přesto platí pro všechna kódová slova $cH^T = 0$.

$$\begin{aligned} s &= yH^T = (y_o, \dots y_{n-1}) \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha^{b} & \alpha^{b+1} & \dots & \alpha^{b+\delta-2} \\ \alpha^{2b} & \alpha^{2(b+1)} & \dots & \alpha^{2(b+\delta-2)} \\ \dots & & & \dots \\ \alpha^{(n-1)b} & \alpha^{(n-1)(b+1)} & \dots & \alpha^{(n-1)(b+\delta-2)} \end{pmatrix} = \\ &= \left(\sum_{i=0}^{n-1} y_i \left(\alpha^{b}\right)^{i}, \sum_{i=0}^{n-1} y_i \left(\alpha^{b+1}\right)^{i}, \dots, \sum_{i=0}^{n-1} y_i \left(\alpha^{b+\delta-2}\right)^{i} \right) = \\ &= \left(y\left(\alpha^{b}\right), y\left(\alpha^{b+1}\right), \dots, y\left(\alpha^{b+\delta-2}\right) \right) = (V_b, V_{b+1}, \dots, V_{b+\delta-2}) \end{aligned}$$

Druhou možností, kterou máme, je využít skutečnosti, že jak BCH tak RS kódy jsou cyklické a vzpomenout si na vzorec 2.3.

Transformace do MS polynomů je lineární, tedy platí $V_i = C_i + E_i$ pro $j = 0, \ldots, n-1$. V zápisu pomocí MS polynomů jsou koeficienty $C_j = c(\alpha^j) = 0$, pro všechny kořeny kódu C, tedy pro $j = b, b+1, \ldots b+\delta-2$.

Obdobně platí pro koeficienty syndromu odpovídající kořenům kódu $j = b, b+1, \ldots b+\delta-2$, že

$$s_{j-b} = s(\alpha^j) = y(\alpha^j) = e(\alpha^j) = E_j. \tag{3.5}$$

Označíme $s_{j-b} = S_j$ a z linearity pak $S_j = E_j = V_j$.

Pro další úvahy by se nám hodilo předpokládat, že $b = 0$ a můžeme to také bez újmy na obecnosti udělat. Cyklické posunutí koeficientů C_j o b míst totiž odpovídá pouze přenásobení kódu C konstantou α^{-b}, protože $C_j = c\,(\alpha^j)$. Takový kód bude pouze jiný RS kód.

Máme syndrom a s ním dokonce část MS transformace chybového vektoru. Dalším postupovým cílem je nalezení lokačního polynomu. V tuto chvíli je třeba algoritmicky vyřešit, jak to udělet co nejjednoduššeji a jednoznačně. Přímočaré řešení, zvané Petersonův algoritmus vychází ze vztahu 3.2, který aplikujeme na chybový vektor e.

$$\begin{pmatrix} E_{w-1} & E_{w-2} & \cdots & E_0 \\ E_w & E_{w-1} & \cdots & E_1 \\ \cdots & \cdots & \cdots & \cdots \\ E_{2w-2} & E_{2w-3} & \cdots & E_{w-1} \end{pmatrix} \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_w \end{pmatrix} = - \begin{pmatrix} E_w \\ E_{w+1} \\ \vdots \\ E_{2w-1} \end{pmatrix}. \tag{3.6}$$

To, co dopředu neznáme, je počet nastalých chyb w a s ním ani velikost matice. Pokud by byl počet chyb maximální opravitelný, tedy $t = \left\lfloor \frac{\delta-1}{2} \right\rfloor$, dá nám rovnice jediné řešení a bude mít nenulový determinant. S menším počtem chyb, řekněme $\nu < w$, ovšem budou sloupce matice lineárně závislé, protože sloupec $\begin{pmatrix} E_\nu \\ E_{\nu+1} \\ \vdots \\ E_{2\nu-1} \end{pmatrix}$ se dá rekurentně vyjádřit pomocí sloupců $\begin{pmatrix} E_{\nu-1} & \cdots & E_0 \\ E_\nu & \cdots & E_1 \\ \vdots & \cdots & \vdots \\ E_{2\nu-2} & \cdots & E_{\nu-1} \end{pmatrix}$ a determinant bude nulový. Řešíme tedy rovnice pro maximální možný počet chyb a vyjde-li nulový determinant, víme, že jsme počet chyb nadhodnotili, zmenšíme matici o poslední řádek a první sloupec a zkusíme to znovu.

Se znalostí lokačního polynomu se dají dopočítat všechny koeficienty $E(z)$ pomocí rekurze 3.1 a odtud pak pomocí inverzní formule 2.5 získáme chybový vektor e.

Jinou možností je vyjít z definice lokačního polynomu a chybné pozice najít jako jeho kořeny. Pro polynomy vyššího stupně je výhodné hledat kořeny tak, že vyzkoušíme všechny možnosti a postupně dosazujeme do $\sigma(z)$ různá α^{-i}, $i = 0, \ldots, n-1$, tzv. Chienovo prohledávání (Chien search). Chyba nastala na těch indexech i, pro které je $\sigma(\alpha^{-i}) = \sigma\left(\frac{1}{X_k}\right) = 0$. Velikost chyby dopočítáme dosazením do evaluačního polynomu získaného podle věty 3.2.

Pro výpočet lokačního polynomu existuje ovšem i rychlejší a efektivnější algoritmus viz další kapitola.

3.5 Berlekamp-Masseyův algoritmus

B-M algoritmus je další cestou pro hledání lokačního polynomu, který splňuje Newtonovy identity, vzorec 3.1. Zbytek dekódování můžeme provést stejně, jako v předchozí kapitole. Vychází ze známých koeficientů syndromu $S_b, ..., S_{b+2t-1}$. Pro začátek předpokládejme, že $b = 0$ a tedy známe $E_0, ..., E_{2t-1}$ a hledáme $\sigma(z)$. Algoritmus postupuje iterativně, v r-tém kroku počítá polynom $\sigma^{(r)}(z)$ takový, že rekurze 3.1 nejnižší možné délky L_r platí pro prvních r členů posloupnosti E. Tuto dílčí posloupnost označíme $E^r = E_0, \ldots, E_{r-1}$. Výstupem po r-tém kroku algoritmu je tedy dvojice $\left(\sigma^{(r)}(z), L_r\right)$ splňující rekurzi

$$E_j = -\sum_{k=1}^{L_r} \sigma_k^{(r)} E_{j-k} \text{ pro každé } j = L_r, \ldots, r-1.$$

Délka rekurze L_r není, ač by se to tak na první pohled mohlo zdát, synonymem pro stupeň polynomu $\sigma^{(r)}(z)$. Ten může být nižší, když jsou koeficienty $\sigma_{L_r}^{(r)}(z), \ldots, \sigma_{L_{r-i}}^{(r)}(z)$ nulové[4].

Dříve než se budeme zaobírat zákulisím, proč BM algoritmus funguje, ukážeme si, jak probíhá.

Jako inicializační předkrok položme $\left(\sigma^{(0)}(z), L_0\right) = (1, 0)$. Vlastní inicializací budiž dvojice $\left(\sigma^{(i+1)}(z), L_{i+1}\right) = (z^{i+1}, i+1)$, kde i je index prvního nenulového koeficientu E.

Začíná-li posloupnost E samými nulami a první nenulový prvek je až na i-té pozici, musí mít $\sigma^{(i+1)}$ stupeň $L_{i+1} = i+1$, protože jinak bychom z $\sigma^{(i+1)}(z)$ nemohli v rekurzi získat nic jiného než samé nuly.

[4]konkrétní případ lze najít např. v [5] na str. 124

Klíčovou otázkou zůstává, jak posloupnost E^r prodloužit, tedy jak najít $\sigma^{(r+1)}(z)$ známe-li už $\sigma^{(r)}(z)$. Přirozená lenost velí nejprve vyzkoušet, jestli už $\sigma^{(r+1)}(z)$ náhodou neznáme, totiž neplatí-li že

$$E_r + \sum_{k=1}^{L_r} \sigma_k^{(r)} E_{r-k} = 0,$$

což by znamenalo, že $\left(\sigma^{(r+1)}(z), L_{r+1}\right) = \left(\sigma^{(r)}(z), L_r\right)$. Obecně ovšem dostaneme

$$E_r + \sum_{k=1}^{L_r} \sigma_k^{(r)} E_{r-k} = d_r$$

a získáním nenulového d_r jsme ověřili, že polynom $\sigma^{(r)}(z)$ z předchozího kroku už pro delší posloupnost nelze použít. Hodnota d_r se nazývá diskrepance (discrepancy).

Není těžké si všimnout, že mezi délkami rekurzí platí nerovnost $L_{r+1} \geq L_r$. Kdyby tomu tak nebylo, nemohl by polynom $\sigma^{(r+1)}(z)$ splňovat rekurzi pro E_r, protože délka L_r je nejkratší možná. Z našich úvah vyplyne dokonce, že $L_{r+1} = \max[L_r, r+1-L_r]$.

V případě, že jsme nuceni hledat nové $\sigma^{(r+1)}(z)$, postupujeme následovně. Najdeme poslední případ v posloupnosti, kdy se délka rekurze změnila, tedy takové $\sigma^{(m)}(z)$, že $L_m < L_{m+1} = L_r$. Pro takové $\sigma^{(m)}(z)$ totiž platí $E_m + \sum_{k=1}^{L_m} \sigma_k^{(m)} E_{m-k} = d_m \neq 0$. Zvolíme-li

$$\sigma^{(r+1)}(z) = \sigma^{(r)}(z) - \frac{d_r}{d_m} z^{r-m} \sigma^{(m)}(z) \tag{3.7}$$

dostaneme

$$\begin{aligned} E_r + \sum_{k=1}^{L_{r+1}} \sigma_k^{(r+1)} E_{r-k} &= E_r + \sum_{k=1}^{L_r} \sigma_k^{(r)} E_{r-k} - \frac{d_r}{d_m} \left(E_m + \sum_{k=1}^{L_m} \sigma_k^{(m)} E_{m-k} \right) \\ &= d_r + \frac{d_r}{d_m} d_m = 0 \end{aligned}$$

a pro všechny nižší indexy $j = L_{r+1}, \ldots, r-1$ rekurze také platí, protože

$$\begin{aligned} E_j + \sum_{k=1}^{L_r} \sigma_k^{(r)} E_{j-k} - \frac{d_r}{d_m} \left(E_{j-r+m} + \sum_{k=1+r-m}^{L_m+r-m} \sigma_{k+r-m}^{(m)} E_{j-k} \right) = \\ E_j + \sum_{k=1}^{L_r} \sigma_k^{(r)} E_{j-k} - \frac{d_r}{d_m} \left(E_{j-r+m} + \sum_{k=1}^{L_m} \sigma_k^{(m)} E_{j-k-r+m} \right) = 0. \end{aligned}$$

Zkontrolujme ještě, jestli má polynom 3.7 minimální možný stupeň. Dáme-li dohromady naše předpoklady o délkách L_m a L_{m+1}, totiž že $L_{m+1} = \max[L_m, m+1-L_m]$ a $L_m < L_{m+1}$, vidíme, že $L_{m+1} = m+1-L_m = L_r$. Ze vzorce 3.7 je jasné, že $L_{r+1} = \max[L_r, r-m+L_m]$ a dosadíme-li v souladu s předchozí úvahou $m+1-L_r$ místo L_m, je podmínka minimality splněna.

Aby byl polynom $\sigma^{(n)}(z)$ skutečně lokačním polynomem, musí splňovat ještě jednu podnímku. Nestačí nám to, že rekurze $E_j = -\sum_{i=0}^{L_r} \sigma_i^{(r)} E_{j-i}$ produkuje posloupnost $(E_o, \ldots, E_{n-1})$. Chceme, aby produkoval posloupnost $(E_o, \ldots, E_{n-1}, E_o, \ldots, E_{n-1}, \ldots)$ která se cyklicky opakuje.

Tady se už ovšem nevyhneme několika obecným poznatkům o posloupnostech.

Věta Pokud $(\sigma(z), L)$ a $(\sigma'(z), L')$ obě splňují rekurentní předpis pro $E_0, \ldots E_{n-1}$ a $L+L' \leq n$, pak $E_n = -\sum_{k=1}^{L} \sigma_k E_{n-k} = -\sum_{k=1}^{L'} \sigma'_k E_{n-k}$ a oba polynomy nadále generují tutéž posloupnost.

Důkaz Z předpokladů víme, že $E_j = -\sum_{i=1}^{L} \sigma_i E_{j-i}$ pro $i = L, \ldots, n-1$ a $E_j = -\sum_{i=1}^{L'} \sigma'_i E_{j-i}$ pro $i = L', \ldots, n-1$. Využijeme-li, že $L+L' \leq n$, můžeme rekurzi zapsat také jako $E_{n-k} = -\sum_{i=1}^{L} \sigma_i E_{n-k-i}$ pro $k = 1, \ldots, L'$ a rovněž $E_{n-k} = -\sum_{i=1}^{L'} \sigma'_i E_{n-k-i}$. Zkombinujeme-li různá vyjádření, vidíme, že

$$\begin{aligned} E_n = -\sum_{k=1}^{L} \sigma_k E_{n-k} &= \sum_{k=1}^{L} \sigma_k \sum_{i=1}^{L'} \sigma'_i E_{n-k-i} = \\ &= \sum_{i=1}^{L'} \sigma'_i \sum_{k=1}^{L} \sigma_k E_{n-k-i} = -\sum_{k=1}^{L'} \sigma'_k E_{n-k}. \end{aligned}$$

Odtud také plyne slíbená rovnost $L_{r+1} = \max[L_r, r+1-L_r]$.

Důsledek I Pro iterace $\left(\sigma^{(r)}(z), L_r\right)$ a $\left(\sigma^{(r+1)}(z), L_{r+1}\right)$ v popisu BM algoritmu platí $L_{r+1} = \max[L_r, r+1-L_r]$.

Důkaz Pokud polynom $\left(\sigma^{(r)}(z), L_r\right)$ generuje posloupnost $E^r = E_o, \ldots, E_{r-1}$, ale posloupnost $E^{r+1} = E_o, \ldots, E_r$ už ne, potom je $L_{r+1} \geq r+1-L_r$, protože kdyby snad existoval nějaký polynom $\left(\sigma'^{(r)}(z), L'_r\right)$, $L'_r < r+1-L_r$, který by zvládl nagenerovat i E^{r+1}, generoval by i E_r. Tím by oba polynomy splňovaly předpoklady předchozí věty, protože $L_r + L'_r \leq r$ a museli by se tedy shodovat i na E^{r+1}, což by byl spor. Máme $L_{r+1} \geq L_r$ a $L_{r+1} \geq r+1-L_r$, tedy

$L_{r+1} \geq \max[L_r, r+1-L_r]$. Rovnost plyne z toho, že ji algoritmus skutečně najde.

Důsledek II Je-li délka rekurze $L \leq \frac{n}{2}$, pak je polynom $(\sigma(z), L)$ jediným polynomem nejnižšího stupně, který splňuje pro posloupnost E délky n rekurenci $E_n = -\sum_{k=1}^{L} \sigma_k E_{n-k}$.

To je také důvod proč jsme v úvahách o BM algoritmu věnovali tolik prostoru délce rekurze L_r.

Nyní shrmeme BM algoritmus ve schématické podobě. Roli $\sigma^{(m)}(z)$ převezme pomocný polynom $B^{(m)}(z)$ (scratch polynomial). Pokud se lokační polynom změní z $\sigma^{(m)}(z)$ na $\sigma^{(m+1)}(z)$, uloží se do $B^{(m)}(z)$ jeho předchozí hodnota $\sigma^{(m)}(z)$. Jinak se pouze v každém kroku přenásobí z, takže při případné další změně z $\sigma^{(r)}(z)$ na $\sigma^{(r+1)}(z)$ obsahuje $z^{r-m}\sigma^{(m)}(z)$, což přesně potřebujeme podle 3.7.

Algoritmus (Berlekamp - Masseyův)

vstup: $S_b, \ldots, S_{b+2t-1}$ MS koeficienty syndromu

výstup: $\sigma(z) := \sigma^{(b+2t)}(z)$

Inicializace : $\sigma^{(b)}(z) = 1, B^{(b)}(z) = 1, L_b = 0$.

for $r := b$ to $b + 2t - 1$ do begin

1. $d_r := S_r + \sum_{i=1}^{L_r} \sigma_i^{(r)} S_{r-i}$
2. if $d_r \neq 0$ and $2L_r \leq r - b$ then $\Delta_r := 1$ else $\Delta_r := 0$
3. $L_{r+1} := \Delta_r (r + 1 - b - L_r) + (1 - \Delta_r) L_r$
4. $$\begin{pmatrix} \sigma^{(r+1)}(z) \\ B^{(r+1)}(z) \end{pmatrix} = \begin{pmatrix} 1 & -d_r z \\ \frac{\Delta_r}{d_r} & (1-\Delta_r) z \end{pmatrix} \begin{pmatrix} \sigma^{(r)}(z) \\ B^{(r)}(z) \end{pmatrix}$$

end

$\sigma(z) := \sigma^{(b+2t)}(z)$.

Složitost výpočtů BM algoritmu je nejvýše $6t^2$, protože má $2t$ iterací a v každé z nich počítáme $\sigma^{(r+1)}(z)$, $B^{(r+1)}(z)$, d_r , což u jednoho z nich zabere nejvýše t operací násobení.

Věta Známe-li $S_b, \ldots, S_{b+2t-1}$ a počet chyb je $\leq t$, pak umíme pomocí BM algoritmu jednoznačne spočítat lokační polynom $\sigma(z)$.

Důkaz Z popisu algoritmu je jasné, že polynom, který dostaneme splňuje rekurzi a zároveň má minimální stupeň. Obojí předpokládáme o lokačním polynomu. Navíc je-li počet chyb $w \leq t$ a minimální vzdálenost $d = 2t + 1$, máme podle důsledku II jednoznačnost.

Výše uvedený algoritmus si ukážeme v praxi v následujícím příkladu.

Příklad Pracujeme s kódem $\mathrm{BCH}_{4,2,3}$ délky 7 generovaným polynomem

$$g(x) = x^3 + \beta x^2 + \beta x + \beta^2 = \left(x^2 + \beta\right)(x + \beta)$$

nad tělesem $F_4 = {}^{F_2}\!/_{x^2+x+1} = \{0, 1, \beta, \beta^2\}$. Prvek α je primitivním prvkem v $F_8 = {}^{F_4}\!/_{x^2+\beta} = \{0, 1, \alpha, \alpha^2 = \beta, \alpha^3, \alpha^4 = \beta^2, \alpha^5, \alpha^6\}$, protože $n = q^m - 1$.

Přijali jsme slovo $v = (00\beta^2\beta 010)$, neboli $v(x) = x^5 + \beta x^3 + \beta^2 x^2$. Vidíme, že $g(x) = \mathrm{nsn}\ (M_\alpha(x), M_{\alpha^2}(x))$ a vypočítáme S_1, S_2.

$$\begin{aligned} S_1 &= v(\alpha) = \alpha^5 + \beta\alpha^3 + \beta^2\alpha^2 = \alpha^6 \\ S_2 &= v\left(\alpha^2\right) = \alpha^{10} + \beta\alpha^6 + \beta^2\alpha^4 = \alpha^3 \end{aligned}$$

Hledání lokačního polynomu $\sigma(z)$ pomocí BM algoritmu znázorníme v tabulce.

i	$\sigma^{(i)}(z)$	$B^{(i)}(z)$	L_i	d_i	$2L_i \leq i-1$	Δ_i
1	1	1	0	α^6	OK	1
2	$1 + \alpha^6 z$	$\frac{1}{\alpha^6} = \alpha$	1	α^2	ne	0
3	$1 + \alpha^4 z$		1			

$$\begin{aligned} d_1 &= S_1 = \alpha^6 \\ d_2 &= S_2 + \sigma_1^{(2)} S_1 = \alpha^3 + \alpha^{12} = \alpha^2 \\ \sigma(z) = \sigma^{(3)}(z) &= 1 + \alpha^6 z + \alpha\alpha^2 z = 1 + \alpha^4 z \end{aligned}$$

Vyšlo nám tedy, že pozice, na níž došlo k první a jediné chybě, má index 4 (pozor, indexujeme od 0), protože $\sigma\left(\frac{1}{\alpha^4}\right) = 1 + \alpha^4 \frac{1}{\alpha^4} = 0$. Tedy $X_1 = \alpha^4$. Zbývá dopočítat velikost chyby, což uděláme pomocí evaluačního polynomu. Podle věty 3.2 je $\omega(z) = \left(1 + \sum_{i=1}^{\infty} A_i z^i\right)\sigma(z)$ a navíc platí $\deg\omega(z) \leq \deg\sigma(z) = w = 1$. Z celého výrazu $\left(1 + \alpha^6 z + \alpha^3 z^2 + \cdots\right)(1 + \alpha^4 z)$ stačí vzít pouze členy do první mocniny z, tedy $\omega(z) = 1 + \left(\alpha^6 + \alpha^4\right) z$. Velikost chyby je

$$Y_1 = -\frac{X_1 \omega\left(\frac{1}{X_1}\right)}{\sigma\prime\left(\frac{1}{X_1}\right)} = \frac{\alpha^4\left(1 + \frac{\left(\alpha^6 + \alpha^4\right)}{\alpha^4}\right)}{\alpha^4} = \alpha^2 = \beta.$$

$$\text{Odeslané slovo bylo tedy } \begin{array}{r} (\ 0 \ 0 \ \beta^2 \ \beta \ 0 \ 1 \ 0 \ ) \\ - \ (\ 0 \ 0 \ 0 \ 0 \ \beta \ 0 \ 0 \ ) \\ \hline (\ 0 \ 0 \ \beta^2 \ \beta \ \beta \ 1 \ 0 \ ) \end{array}.$$

Pro binární kódy je situace ještě o trochu jednodušší. Při počítání podle klasického BM algoritmu si můžeme všimnout, že všechny sudé diskrepanční bity vycházejí nulové. To není náhoda, jak se dá nahlédnout pomocí 3.3 a 3.4. Proto lze upravit BM algoritmus pro binární kódy tak, aby sudé kroky, kde se $\sigma^{(r)}(z)$ nemění, přeskakoval a pracoval rychleji. Navíc odpadá starost s výpočtem velikosti chyby, to je vždy 1.

Algoritmus (Berlekamp-Masseyův pro binární kódy)

vstup: $S_b, \ldots, S_{b+2t-1}$ MS koeficienty syndromu

výstup: $\sigma(z) := \sigma^{(b+2t)}(z)$

Inicializace : $\sigma^{(b)}(z) = 1, B^{(b)}(z) = 1, L_b = 0.$

for $r := b$ to $b + 2t - 1$ do

if $r \bmod 2 \neq 0$ then begin

1. $d_r := S_r + \sum_{i=1}^{L_r} \sigma_i^{(r)} S_{r-i}$
2. if $d_r \neq 0$ and $2L_r \leq r - b$ then $\Delta_r := 1$ else $\Delta_r := 0$
3. $L_{r+2} := \Delta_r (r + 2 - b - L_r) + (1 - \Delta_r) L_r$
4. $$\begin{pmatrix} \sigma^{(r+2)}(z) \\ B^{(r+2)}(z) \end{pmatrix} = \begin{pmatrix} 1 & -d_r z^2 \\ \frac{\Delta_r}{d_r} & (1 - \Delta_r) z^2 \end{pmatrix} \begin{pmatrix} \sigma^{(r)}(z) \\ B^{(r)}(z) \end{pmatrix}$$

end

$\sigma(z) := \sigma^{(b+2t)}(z)$.

Závěr

Naše exkurze do světa samoopravných kódů je u konce. Získali jsme dvojí náhled na Reed-Solomonovy kódy, jako na evaluace polynomů až do určitého stupně a jako podtřídy BCH kódů, kde $n = q - 1$. Seznámili jsme se s lokačním a evaluačním polynomem a způsobem, jak je použít v dekódování. Pomocí Berlekamp-Masseyho algoritmu jsme naučili, jak lokační polynom najít.

Tím ovšem téma dekódování Reed-Solomonových a BCH není zcela vyčerpáno. Jsou známé další, zde neuvedené algoritmy, například Berlekampův a Welch-Berlekampův, s nimiž se čtenář může seznámit např. v knize [3]. Zůstává otázka, jak opravit větší množství chyb, než je polovina minimální vzdálenosti, kterou řeší Sudanův algoritmus (více v [3])a která by stačila na samostatnou práci.

Literatura

[1] F.J. MacWilliams, N.J.A. Sloane : *The theory of error-correcting codes*, North-Holland, Amsterdam, 1977.

[2] J.H. van Lint: *Introduction to Coding Theory*, Springer-Verlag, New York 1982.

[3] Richard E. Blahut *Handbook of coding theory, Decoding of Cyclic Codes and Codes on Curves*, Elsevier Science, 1998.

[4] Tomáš Kaiser *skripta k přednášce ze Samoopravných kódů na MFF UK 2009*, http://home.zcu.cz/~kaisert/kody/.

[5] James L. Massey : *Shift-Register Syntesis and BCH Decoding*, IEEE Transactions on information theory, Vol. It-15, January 1969.

Seznam použitých zkratek

BCH	kód, jehož název vznikl jako zkratka počátečních jmen jeho objevitelů: A. Hocquenhem, R. C. Bose a D. K. Ray-Chaudhuri.
BM	Berkelamp-Masseyův algoritmus
MS	Mattson-Solomonův polynom
RS	Reed-Solomonův kód