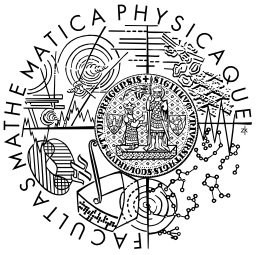


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Filip Lorenc

CRC-kódy

Katedra algebry

Vedoucí diplomové práce: doc. Mgr. et Mgr. Jan Žemlička, Ph.D.

Studijní program: Matematika

Studijní obor: Matematika pro informační technologie

Praha 2021

Prohlašuji, že jsem tuto diplomovou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

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 má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Chtěl by velice poděkovat svému vedoucímu diplomové práce doc. Mgr. et Mgr. Janu Žemličkovi, Ph.D. za jeho nápady, připomínky, cenné rady, trpělivost při odhalování chyb a hlavně za jeho čas, který mi věnoval při konzultacích. Dále bych chtěl poděkovat rodině a svým kamarádům, kteří mě při psaní této práce psychicky podporovali.

Název práce: CRC-kódy

Autor: Filip Lorenc

Katedra: Katedra algebry

Vedoucí diplomové práce: doc. Mgr. et Mgr. Jan Žemlička, Ph.D., Katedra algebry

Abstrakt: Diplomová práce se zabývá popisem CRC kódů, což je typ polynomiálních samoopravných kódů, a popisem protokolů CAN a CAN FD, které se používají hlavně v automobilech pro přenos dat mezi senzory. Jedním z bezpečnostních prvků protokolů je využití CRC kódů s Hammingovou vzdáleností 6. Naneštěstí oba protokoly obsahují chybu v návrhu, která způsobuje, že některé přijaté zprávy s jedním chybným bitem nemusí být protokolem odhaleny. Cílem práce bylo tuto chybu popsat a zjistit, zda je možné ji eliminovat použitím jiného CRC kódu. Podařilo se charakterizovat všechny zprávy, které nejsou při tomto typu chyby odhaleny CRC kódem a na základě toho bylo možné dokázat, že pravděpodobnost výskytu chyby v protokolu nezávisí na volbě CRC kódu pevně dané délky.

Klíčová slova: Polynomiální kód, Generující polynom, CRC, CAN, Kontrolní bity

Title: CRC-codes

Author: Filip Lorenc

Department: Department of Algebra

Supervisor: doc. Mgr. et Mgr. Jan Žemlička, Ph.D., Department of Algebra

Abstract: This thesis deals with description of CRC codes, which is a type of polynomial error correction codes, and description of CAN and CAN FD protocols used in automobiles for data transmission between sensors. One of the security elements is usage of the CRC codes with the Hamming distance 6. Unfortunately, both protocols contain a design vulnerability which causes that some received messages with one wrong bit do not have to be detected by the protocol. The aim of the thesis was to describe this vulnerability and found out, if it was possible to eliminate it by using different CRC code. It managed to characterize all messages, which are not during this vulnerability detected by CRC code and based on that it was possible to prove, that the probability of error does not depend on a CRC code choice of a fixed length.

Keywords: Polynomial code, Generator polynomial, CRC, CAN, Check bits

Obsah

Úvod

Přenos dat je v dnešní době neodmyslitelná součást našich životů. Každý se s ním denně setkává, když se kouká na televizi, telefonuje, surfuje na internetu, platí v obchodě, pracuje na počítači a při mnoha dalších činnostech. Bohužel bezdrátový i drátový přenos není bezchybný. Proto je potřeba data upravit tak, aby bylo možné chybu po přenosu odhalit a případně ji i opravit. K tomu se využívají samoopravné kódy.

CRC kódy jsou jedním typem samoopravných kódů, které obecně slouží hlavně k detekci chyb. Jejich podstata spočívá v tom, že zprávu vnímáme jako polynom a tento polynom vydělíme se zbytkem generujícím polynomem, který je charakteristický pro každý CRC kód. Zbytek po dělení reprezentuje kontrolní informaci, která se spolu se zprávou odešle. Pokud nebyla zpráva příliš poškozena, je příjemce schopen díky této kontrolní informaci rozhodnout, zda k chybě došlo nebo ne.

CRC kódy se v praxi hojně využívají. Hlavně je to využití v automobilovém průmyslu a v oblasti automatizace, kde jsou součástí síťového protokolu CAN. Většina čidel a senzorů uvnitř moderních aut pomocí tohoto protokolu přenáší data. Protokol je též součástí rozvíjejícího se autonomního řízení. Slouží například k ovládání tempomatu, protikolizního systému, parkovacích senzorů, start-stop systému a dalších nejen bezpečnostních systémů a funkcionalit.

CRC kódy jsou vhodná metoda na ochranu dat proti poškození při přenosu. V CAN a CAN FD protokolech nebylo použití CRC kódů vhodně navrhnuto, což způsobilo zranitelnost v detekci chyb. Přestože používané CRC kódy dokáží odhalit chybnou zprávu obsahující až pět chyb, samotnému protokolu může uniknout zpráva s jedinou chybou.

Obsahem této práce je teorie CRC kódů a popis CAN protokolu i jeho novější verze CAN FD včetně detailního popisu chyby, její opravy a zhodnocení řešení. Většina problémů spojených s použitím CRC kódů je primárně technických a matematický přístup není obvykle příliš využíván. Snahou této diplomové práce je tak spojit dva vzdálené světy - ten matematický a technický.

Hlavní cíle této práce jsou dva. Za prvé popsat CRC kódy matematicky a zasadit je do teorie polynomiálních kódů, protože žádný kvalitní matematický popis v dostupné literatuře neexistuje. CRC kódy jsou v dostupných publikacích popsány hlavně algoritmicky a technicky. Druhým cílem je zjistit více o chybě v CAN FD protokolu a její souvislosti s CRC kódy, pokusit se vytvořit matematický popis problému a na základě něho vyvodit důsledky. Tím je tato práce odlišná od valné většiny ostatních publikovaných článků zabývajících se tímto protokolem a jeho zranitelností. Zdá se, že se zatím nikdo nepokusil tuto chybu popsat matematicky.

V první kapitole uvedeme základní terminologii a pojmy z teorie samoopravných kódů potřebných pro popis CRC kódů. V druhé kapitole popíšeme teorii polynomiálních kódů a jejich detekční a opravovací schopnosti. Samotným CRC kódům, jakožto speciálnímu typu polynomiálních kódů, se budeme věnovat ve třetí kapitole. Kromě popisu CRC kódů budou v této kapitole nastíněny i metody realizace efektivního kódování v praxi na dnešních procesorech.

Ve čtvrté kapitole popíšeme CAN protokol a jeho novější verzi CAN FD, jakožto jeden významný příklad využití CRC kódů v praxi. Dále popíšeme chyby

obou verzí. Pokusíme se vytvořit matematický popis chyby v původní neopravené verzi CAN FD a pomocí tohoto popisu zkusíme charakterizovat všechny zprávy s jednou chybou, které této verzi protokolu mohou uniknout. Na závěr uvedeme analýzu používaných CRC polynomů založenou na teorii z předchozích tří kapitol. V praxi se vlastnosti těchto kódů ověřují hrubou silou.

1. Polynomiální a cyklické kódy

První kapitola popisuje z velké části základní terminologii a známá tvrzení z teorie samoopravných kódů. Ta vychází spolu s myšlenkami důkazů ze skript [19]. Druhá část pojednávající o polynomiálních a cyklických kódech a vztahu mezi nimi vychází navíc z [1] a [3].

1.1 Úvod do blokových kódů

Značení. *Znakem $\mathbb{F}_q$ rozumíme konečné těleso o q prvcích.*

Definice 1.1.1 (Slovo a jeho délka)**.** *Nechť $n \in \mathbb{N}$. **Slovo** $\boldsymbol{u} \in \mathbb{F}_q^n$ definujeme jako uspořádanou n-tici hodnot tělesa $\mathbb{F}_q$. **Délku** slova definujeme jako počet jeho souřadnic.*

Značení. *Slovo $\boldsymbol{u} \in \mathbb{F}_q^n$ délky n budeme zapisovat jako*

$$\boldsymbol{u} = u_{n-1} \dots u_0,$$

kde souřadnice slova označujeme u_i a jejich indexy jsou zleva seřazeny sestupně. Slova budou dále v textu značena tučným písmem. Speciálně slovo složené ze samých nul budeme značit $\boldsymbol{o}$.

Poznámka 1.1.2. *Dále v textu budeme slova a řádkové vektory ztotožňovat. Pokud budeme chtít zdůraznit, že se jedná o vektor, zapíšeme jeho souřadnice do závorek.*

Definice 1.1.3 (Q-ární blokový kód)**.** *Libovolnou nenulovou podmnožinu vektorového prostoru $\mathbb{F}_q^n$ nazýváme **(q-ární) blokový kód** délky n.*

Definice 1.1.4 (Kódové slovo)**.** *Každý prvek blokového kódu nazýváme **kódové slovo**.*

Definice 1.1.5 (Lineární kód)**.** *Blokový kód $\mathscr{C} \subseteq \mathbb{F}_q^n$, který je podprostorem vektorového prostoru $\mathbb{F}_q^n$, nazýváme **lineární**.*

Značení. *Lineární kód $\mathscr{C} \subseteq \mathbb{F}_q^n$, kde $k = \dim_{\mathbb{F}_q} \mathscr{C}$, budeme značit $[n, k]_q$-kód. Pokud bude hodnota q z kontextu zřejmá, zkráceně takový kód označíme*

$$[n, k]\text{-kód.}$$

Definice 1.1.6 (Generující matice)**.** *Nechť $\mathscr{C}$ je lineární kód. Matice, která má v řádcích bázové vektory kódu $\mathscr{C}$, se nazývá **generující matice** kódu $\mathscr{C}$ a budeme ji značit $\boldsymbol{C}$.*

Pozorování. *Jestliže $\mathscr{C}$ je $[n, k]_q$-kód, potom $\boldsymbol{C} \in \mathbb{F}_q^{k \times n}$.*

Generující matice neslouží pouze jako stručný zápis lineárního kódu, ale také se pomocí ní realizuje kódování. V $[n, k]_q$-kódu se slovo $\boldsymbol{u}$ délky k zakóduje na slovo $\boldsymbol{c}$ délky n pomocí vzorce

$$\boldsymbol{u} \cdot \boldsymbol{C} = \boldsymbol{c},$$

kde $\boldsymbol{u} \in \mathbb{F}_q^k$ a $\boldsymbol{c} \in \mathbb{F}_q^n$.

Zakódované slovo $\boldsymbol{c}$ potřebujeme umět dekódovat. Abychom toho byli schopni, potřebujeme mít jistotu, že je $\boldsymbol{c}$ slovo kódové. K tomuto ověření slouží kontrolní matice.

Definice 1.1.7 (Kontrolní matice)**.** *Nechť $\mathscr{C}$ je $[n, k]_q$-kód. Matice $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$ budeme nazývat* ***kontrolní matice*** *lineárního kódu $\mathscr{C}$, pokud platí*

$$\boldsymbol{u} \in \mathscr{C} \Leftrightarrow \boldsymbol{H}\boldsymbol{u}^T = \boldsymbol{o}^T.$$

Definice 1.1.8 (Generující matice ve standardním tvaru)**.** *Generující matice $[n, k]_q$-kódu je ve* ***standardním tvaru****, pokud*

$$\boldsymbol{C} = (\boldsymbol{I_k}|\boldsymbol{A}),$$

kde $\boldsymbol{A} \in \mathbb{F}_q^{k\times(n-k)}$ a $\boldsymbol{I_k}$ je jednotková matice řádu k.

Generující matice ve standardním tvaru je užitečná tím, že z kódového slova je okamžitě vidět, jak vypadá jeho dekódovaná verze. Tím je právě prvních k souřadnic zleva. Ostatní souřadnice obsahují tzv. kontrolní součet, který určuje, zda se jedná o kódové slovo.

Příklad. Matice

$$\boldsymbol{C} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

z předchozího příkladu je ve standardním tvaru.

□

Další výhodou generující matice ve standardním tvaru je rychlý výpočet příslušné kontrolní matice.

Tvrzení 1.1.9. *Je-li generující matice $[n, k]$-kódu ve tvaru $(\boldsymbol{I_k}|\boldsymbol{A})$, potom je jeho kontrolní matice ve tvaru $(-\boldsymbol{A}^T|\boldsymbol{I_{n-k}})$.*

Důkaz. Viz tvrzení 1.1 v [19].

□

Příklad. Má-li $[5, 3]_2$-kód generující matici ve tvaru

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

pak je jeho kontrolní matice tvaru

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

□

Definice 1.1.10 (Hammingova vzdálenost)**.** *Nechť* $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_q^n$*, potom*

$$d(\boldsymbol{u}, \boldsymbol{v}) = |\{i : u_i \neq v_i\}|$$

nazveme ***Hammingova vzdálenost*** *slov* $\boldsymbol{u}$ *a* $\boldsymbol{v}$*.*

Definice 1.1.11 (Hammingova váha slova)**.** *Nechť* $\boldsymbol{u} \in \mathbb{F}_q^n$*.* ***Hammingovou váhou slova*** $\boldsymbol{u}$ *rozumíme*

$$w(\boldsymbol{u}) = d(\boldsymbol{u}, \boldsymbol{o}).$$

Důležitou charakteristikou kódu je minimální vzdálenost dvou jeho libovolných kódových slov. Ta popisuje schopnost kódu odhalovat a opravovat chyby, došlo-li při přenosu k chybě.

Definice 1.1.12 (Vzdálenost kódu)**.** *Nechť* $\mathscr{C} \subseteq \mathbb{F}_q^n$*,* $|\mathscr{C}| > 1$ *je blokový kód. Potom*

$$d(\mathscr{C}) = \min\{d(\boldsymbol{u}, \boldsymbol{v}) : \boldsymbol{u}, \boldsymbol{v} \in \mathscr{C}, \boldsymbol{u} \neq \boldsymbol{v}\}$$

nazýváme ***(Hammingova) vzdálenost kódu*** $\mathscr{C}$*.*

Vzdálenost kódu lze v případě lineárních kódů spočítat jednodušeji. Je to totéž jako minimální váha kódového slova.

Poznámka 1.1.13. *Je-li* $\mathscr{C}$ $[n, k]_q$*-kód, potom platí, že*

$$d(\mathscr{C}) = \min\{d(\boldsymbol{u}, \boldsymbol{v}) : \boldsymbol{u}, \boldsymbol{v} \in \mathscr{C}, \boldsymbol{u} \neq \boldsymbol{v}\} = \min\{w(\boldsymbol{u}) : \boldsymbol{u} \in \mathscr{C} \setminus \{\boldsymbol{o}\}\}.$$

Důkaz. Plyne z toho, že

$$u_i \neq v_i \Leftrightarrow u_i - v_i \neq 0$$

a že rozdíl dvou kódových slov je v lineárním kódu opět kódové slovo.

□

Definice 1.1.14. *Řekneme, že kód* $\mathscr{C}$ ***odhaluje*** *(detekuje)* $r \in \mathbb{N}$ *chyb, jestliže*

$$d(\mathscr{C}) > r.$$

Řekneme, že kód $\mathscr{C}$ ***opravuje*** $r \in \mathbb{N}$ *chyb, jestliže*

$$d(\mathscr{C}) > 2r.$$

Poznámka 1.1.15. *Odhaluje-li / opravuje-li kód* r *chyb, pak odhaluje / opravuje i libovolný nižší počet chyb.*

Vzdálenost kódu nám umožňuje odpovědět na otázku, kolik chyb při přenosu kódového slova dokážeme odhalit a nebo opravit. Pokud je například $d(\mathscr{C}) = 3$, dokáže kód odhalit slovo, které má až dvě chyby, nebo opravit slovo s jednou chybou.

1.2 Polynomiální kódy

Na slova z $\mathbb{F}_q^n$ můžeme nahlížet též jako na polynomy nad tělesem $\mathbb{F}_q$. Přesněji slovo $\boldsymbol{u} = u_{n-1} \dots u_0 \in \mathbb{F}_q^n$ lze chápat jako polynom

$$\sum_{i=0}^{n-1} u_i x^i \in \mathbb{F}_q[x].$$

Značení. *Nadále budeme polynom s koeficienty* $\boldsymbol{u}$ *značit jako* $u(x)$*. Naopak koeficienty libovolného polynomu* $v(x)$ *budeme popisovat slovem* $\boldsymbol{v}$*. V další části textu budou pojmy slovo a k němu odpovídající polynom často zaměňovány podle kontextu.*

Definice 1.2.1 (MSbF a LSbF řazení)**.** *Řekneme, že slovo* $\boldsymbol{u} = u_{n-1} \dots u_0$ *má*

- ***MSbF*** *(Most significant bit first)* ***řazení*** *(budeme značit* $\overset{m}{\sim}$*), jestliže*

$$u(x) = \sum_{i=0}^{n-1} u_i x^i,$$

- ***LSbF*** *(Least significant bit first)* ***řazení*** *(budeme značit* $\overset{l}{\sim}$*), jestliže*

$$u(x) = \sum_{i=0}^{n-1} u_{n-i} x^i.$$

Poznámka 1.2.2. *Pokud nebude řečeno jinak, předpokládá se implicitně MSbF řazení.*

Příklad. Například

$$x^6 + x^4 + x^3 + x \overset{m}{\sim} 1011010,$$
$$x^6 + x^4 + x^3 + x \overset{l}{\sim} 0101101.$$

□

Důležitým pojmem celé této práce je definice polynomiálního kódu, což je typ lineárního kódu, jehož slova splňují speciální vlastnost jako polynomy.

Definice 1.2.3 (Polynomiální kód)**.** *Nechť* $0 \neq g(x) \in \mathbb{F}_q[x]$ *je polynom stupně menšího než* $n \in \mathbb{N}$*. Množinu slov*

$$\{\boldsymbol{c} \in \mathbb{F}_q^n \ : \ g(x) \mid c(x)\}$$

budeme nazývat ***polynomiální kód*** *délky* n *s* ***generujícím polynomem*** $g(x)$*.*

Příklad. Polynomiální kód délky 5 s generujícím polynomem

$$g(x) = x^2 + 1 \in \mathbb{F}_2[x]$$

obsahuje následující seznam slov:

$$\begin{aligned}
0 \cdot g(x) &= 0 && \overset{m}{\sim} 00000,\\
1 \cdot g(x) &= x^2 + 1 && \overset{m}{\sim} 00101,\\
x \cdot g(x) &= x^3 + x && \overset{m}{\sim} 01010,\\
(x+1) \cdot g(x) &= x^3 + x^2 + x + 1 && \overset{m}{\sim} 01111,\\
x^2 \cdot g(x) &= x^4 + x^2 && \overset{m}{\sim} 10100,\\
(x^2+1) \cdot g(x) &= x^4 + 1 && \overset{m}{\sim} 10001,\\
(x^2+x) \cdot g(x) &= x^4 + x^3 + x^2 + x && \overset{m}{\sim} 11110,\\
(x^2+x+1) \cdot g(x) &= x^4 + x^3 + x + 1 && \overset{m}{\sim} 11011
\end{aligned} \tag{1.1}$$

□

Pozorování. *Stupeň generujícího polynomu polynomiálního $[n, k]$-kódu je $n - k$.*

Tvrzení 1.2.4. *Nechť $\mathscr{C}$ je polynomiální kód délky n s generujícím polynomem*

$$g(x) = \sum_{i=0}^{n-k} g_i x^i \in \mathbb{F}_q[x]$$

stupně $n - k$. Potom $\mathscr{C}$ je $[n, k]_q$-kód s generující maticí $\boldsymbol{C} \in \mathbb{F}_q^{k \times n}$ ve tvaru

$$\begin{pmatrix}
g_{n-k} & \cdots & g_0 & & & \\
 & g_{n-k} & \cdots & g_0 & & \\
 & & \ddots & & \ddots & \\
 & & & g_{n-k} & \cdots & g_0
\end{pmatrix}$$

Důkaz. Protože je $g(x) \neq 0$, jsou i jednotlivé řádky matice $\boldsymbol{C}$ nenulové. Jelikož je matice v odstupňovaném tvaru bez nulových řádků, jsou všechny řádky lineárně nezávislé. Jejich počet je

$$n - (n - k + 1) + 1 = k.$$

Každý řádek matice odpovídá kódovému slovo kódu $\mathscr{C}$, neboť tato slova reprezentují polynomy

$$x^{k-1} \cdot g(x),\ x^{k-2} \cdot g(x), \ldots,\ x \cdot g(x),\ g(x), \tag{1.2}$$

která jsou v kódu z definice. Protože

$$g(x) \cdot \sum_{i=0}^{k-1} a_i x^i = \sum_{i=0}^{k-1} \left(a_i x^i \cdot g(x) \right),$$

kde $a_i \in \mathbb{F}_q$, můžeme každé kódové slovo z $\mathscr{C}$ rozepsat jako lineární kombinaci polynomů z (1.2) a naopak každá lineární kombinace řádků matice $\boldsymbol{C}$ tvoří opět kódové slovo. Jedná se tak o lineární kód s generující maticí $\boldsymbol{C}$.

□

Příklad. Polynomiální kód s generujícím polynomem $g(x) = x^2 + 1 \in \mathbb{F}_2[x]$ z předchozího příkladu má generující matici ve tvaru

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

Báze tohoto kódu je v řeči polynomů ve tvaru

$$\left\{x^4 + x^2,\, x^3 + x,\; x^2 + 1\right\}.$$

Jedná se o $[5,\, 3]_2$-kód. Bázi můžeme rozepsat následujícím způsobem:

$$\begin{aligned} x^4 + x^2 &= x^2 \cdot g(x), \\ x^3 + x &= x \cdot g(x), \\ x^2 + 1 &= 1 \cdot g(x). \end{aligned}$$

Díky tomu se dá na násobení polynomu stupně nejvýše $k - 1$ s generujícím polynomem $g(x)$ nahlížet jako na lineární kombinaci řádků matice, což je nad $\mathbb{F}_2$ součet. Například součin

$$(x^2 + x + 1)(x^2 + 1)$$

je ekvivalentní součtu všech tří řádků matice $\boldsymbol{C}$. Z toho jednoduše dostaneme

$$x^4 + x^3 + x + 1 \overset{m}{\sim} 11011.$$

□

1.3 Cyklické kódy

Cyklické kódy jsou speciálním příkladem blokových kódů.

Definice 1.3.1 (Cyklický kód)**.** *Blokový kód $\mathscr{C}$ délky n nazveme **cyklickým**, jestliže je invariantní vůči cyklickému posunu souřadnic, neboli*

$$\forall\; u_{n-1} \dots u_0 \in \mathscr{C} \;:\; u_{n-2} \dots u_0 u_{n-1} \in \mathscr{C}.$$

Značení. *Množina všech polynomů nad $\mathbb{F}_q$ stupně nejvýše $n - 1$ se sčítáním a odčítáním po složkách a násobením modulo $(x^n - 1)$ tvoří okruh, který budeme značit $\mathbb{F}_q[x]_n$.*

Pozorování.

$$\mathbb{F}_q[x]_n \simeq \mathbb{F}_q[x]/(x^n - 1).$$

V polynomiálním zápisu je cyklický posun ekvivalentní násobení polynomu hodnotou x v okruhu $\mathbb{F}_q[x]_n$. V cyklickém kódu délky n to znamená, že jestliže $u(x) \in \mathscr{C}$, pak

$$x \cdot u(x) \bmod (x^n - 1) \in \mathscr{C}.$$

Pokud je cyklický kód lineární, neboli každá lineární kombinace kódových slov je opět kódovým slovem, platí navíc, že

$$a(x) \cdot u(x) \bmod (x^n - 1) \in \mathscr{C},$$

kde $a(x) \in \mathbb{F}_q[x]$.

Pozorování. *Lineární cyklický kód $\mathscr{C} \subseteq \mathbb{F}_q^n$ můžeme též chápat jako podmnožinu $\mathbb{F}_q[x]_n$.*

Tvrzení 1.3.2. *$\mathscr{C} \subseteq \mathbb{F}_q[x]_n$ je lineární cyklický kód právě tehdy, když $\mathscr{C}$ je ideál okruhu $\mathbb{F}_q[x]_n$.*

Důkaz. Implikace $\Leftarrow$ plyne z definice ideálu, neboť ideál je uzavřený na součty prvků z ideálu a na násobky prvků z $\mathbb{F}_q[x]_n$.

Pro důkaz implikace $\Rightarrow$ stačí dokázat uzavřenost na násobení prvky z $\mathbb{F}_q[x]_n$, neboť ostatní vlastnosti plynou z linearity kódu. Po zbytek důkazu budeme znakem $\cdot$ rozumět násobení v okruhu $\mathbb{F}_q[x]_n$. Nechť $u(x) \in \mathscr{C}$. Ukážeme, že

$$u(x) \cdot a(x) \in \mathscr{C} \quad \forall\ a(x) \in \mathbb{F}_q[x]_n.$$

Z cykličnosti plyne, že $x \cdot u(x) \in \mathscr{C}$. To lze aplikovat dál, a proto i

$$x^2 \cdot u(x), \ldots, x^{n-1} \cdot u(x) \in \mathscr{C}.$$

Z linearity a výše uvedeného dostáváme

$$u(x) \cdot a(x) = u(x) \cdot \sum_{i=0}^{n-1} a_i x^i = \sum_{i=0}^{n-1} \Big(a_i x^i \cdot u(x)\Big) \in \mathscr{C},$$

kde $a_i \in \mathbb{F}_q$, z čehož vyplývá, že $\mathscr{C}$ je ideál.

□

Tvrzení 1.3.3. *Všechny ideály okruhu $\mathbb{F}_q[x]_n$ jsou hlavní a ve tvaru $([g(x)])$, kde $g(x) \mid x^n - 1$ v $\mathbb{F}_q[x]$.*

Důkaz. Viz tvrzení C.1 v [19].

□

Důsledek 1.3.4. *Cyklický kód je polynomiálním kódem.*

Tvrzení 1.3.5. *Nechť $\mathscr{C} \subseteq \mathbb{F}_q^n$ je lineární cyklický kód. Pak $\mathscr{C}$ je polynomiální kód a $g(x) \in \mathbb{F}_q[x]$ je jeho generující polynom, neboli*

$$u(x) \in \mathscr{C} \Leftrightarrow g(x) | u(x)$$

právě tehdy, když $0 \neq g(x) \in \mathscr{C}$ je nejmenšího stupně. Navíc generující monický polynom je určen jednoznačně.

Důkaz. Implikaci $\Leftarrow$ dokážeme pomocí obměny, tedy pokud $g(x)$ negeneruje celý kód $\mathscr{C}$, není $g(x)$ nejmenšího stupně. Protože $g(x)$ negeneruje $\mathscr{C}$, existuje kódové slovo $a(x) \in \mathscr{C}$, které není násobkem $g(x)$, neboli

$$g(x) \nmid a(x).$$

Polynomy $g(x)$ a $a(x)$ jsou nejvýše stupně $n-1$, protože jsou z kódu délky n. Polynom $a(x)$ vydělíme se zbytkem polynomem $g(x)$

$$a(x) = b(x) \cdot g(x) + r(x),$$

kde $\deg(g(x)) > \deg(r(x))$ a $r(x) \neq 0$. Protože je kód $\mathscr{C}$ polynomiální, tak

$$b(x) \cdot g(x) \in \mathscr{C}.$$

Z linearity kódu vyplývá, že

$$r(x) = a(x) - b(x) \cdot g(x) \in \mathscr{C}.$$

Našli jsme nenulový polynom $r(x) \in \mathscr{C}$, jehož stupeň je menší než stupeň polynomu $g(x)$.

Pro důkaz implikace $\Rightarrow$ si stačí uvědomit, že nemůže existovat nenulové kódové slovo stupně menšího, než je stupeň generujícího polynomu. V opačném případě by nemohlo platit, že generující polynom takové slovo dělí.

Zbývá jednoznačnost monického $g(x)$. Nechť existují dva monické polynomy nejmenšího stupně $g(x)$, $g'(x) \in \mathscr{C}$. Jejich rozdíl je také kódové slovo z linearity kódu, ale stupeň jejich rozdílu je ostře menší. Proto $g(x) = g'(x)$.

□

1.4 Vztah polynomiálních a cyklických kódů

Z předchozí sekce víme, že každý lineární cyklický kód je polynomiálním kódem. V této kapitole ukážeme, kdy platí i opačná implikace.

Lemma 1.4.1. *Nechť $f(x)$, $g(x)$, $h(x) \in \mathbb{F}_q[x]$. Pak*

$$f(x) \cdot g(x) \ mod\ (g(x) \cdot h(x)) = g(x) \cdot (f(x) \ mod\ h(x)).$$

Důkaz. Po vydělení polynomů $f(x) \cdot g(x)$ a $g(x) \cdot h(x)$ se zbytkem můžeme $f(x) \cdot g(x)$ zapsat jako

$$f(x) \cdot g(x) = a(x) \cdot g(x) \cdot h(x) + r(x),$$

kde $a(x)$, $r(x) \in \mathbb{F}_q[x]$ a $\deg(r(x)) < \deg(g(x) \cdot h(x))$. Z toho plyne $g(x) \mid r(x)$ a

$$\deg\left(\frac{r(x)}{g(x)}\right) = \deg(f(x) - a(x) \cdot h(x)) < \deg(h(x)),$$

a proto

$$\begin{aligned} f(x) \cdot g(x) \bmod (g(x) \cdot h(x)) = r(x) &= f(x) \cdot g(x) - a(x) \cdot h(x) \cdot g(x) \\ &= g(x) \cdot (f(x) - a(x) \cdot h(x)) \\ &= g(x) \cdot (f(x) \bmod h(x)). \end{aligned}$$

□

Tvrzení 1.4.2. *Nechť $\mathscr{C}$ je polynomiální kód délky n s generujícím polynomem $g(x) \in \mathbb{F}_q[x]$. $\mathscr{C}$ je cyklický právě tehdy, když*

$$g(x) \mid x^n - 1 \quad (nad\ \mathbb{F}_q).$$

Důkaz. Nejprve dokážeme implikaci $\Rightarrow$. Z tvrzení 1.3.5 víme, že stupeň polynomu $g(x)$ je nejmenší ze všech stupňů polynomů z kódu $\mathscr{C}$. Vydělíme-li $x^n - 1$ polynomem $g(x)$

$$x^n - 1 = a(x) \cdot g(x) + r(x),$$

kde $\deg(r(x)) < \deg(g(x)) < n$, dostaneme, že

$$r(x) \equiv -a(x) \cdot g(x) \pmod{(x^n - 1)},$$

a tedy $r(x) \in \mathscr{C}$. Protože $r(x)$ má menší stupeň než $g(x)$, je $r(x) = 0$ a dostáváme

$$x^n - 1 = a(x) \cdot g(x),$$

což je hledaná dělitelnost $g(x) \mid x^n - 1$.

Nyní dokážeme opačnou implikaci $\Leftarrow$. Předpokládáme, že $g(x) \mid x^n - 1$, tedy

$$x^n - 1 = a(x) \cdot g(x),$$

kde $a(x) \in \mathbb{F}_q[x]$ stupně $\leq n$. Mějme kódové slovo $u(x) \in \mathscr{C}$. Chceme ukázat, že i

$$x \cdot u(x) \bmod (x^n - 1) \in \mathscr{C},$$

což prokáže cykličnost kódu $\mathscr{C}$. Stupeň $u(x)$ je nejvýše $n - 1$ a navíc

$$u(x) = b(x) \cdot g(x)$$

pro nějaké $b(x) \in \mathbb{F}_q[x]$. Spojíme-li tyto informace dohromady, dostáváme za pomoci lemmatu 1.4.1

$$\begin{aligned} x \cdot u(x) \bmod (x^n - 1) &= x \cdot g(x) \cdot b(x) \bmod (x^n - 1) \\ &= x \cdot g(x) \cdot b(x) \bmod (a(x) \cdot g(x)) \\ &= g(x) \cdot (x \cdot b(x) \bmod a(x)). \end{aligned} \tag{1.3}$$

Protože $\mathscr{C}$ je polynomiální, je slovo z (1.3) prvkem $\mathscr{C}$. Tím je důkaz dokončen. □

Pozorování. *Nechť*

$$x^n - 1 = f_1(x)^{e_1} \cdots f_k(x)^{e_k},$$

kde $e_i \in \mathbb{N}$, $k \in \mathbb{N}$, je ireducibilní rozklad nad $\mathbb{F}_q$. Počet všech různých cyklických kódů $\mathscr{C} \subseteq \mathbb{F}_q^n$ je $2^{\sum_{i=1}^{k} e_i}$.

Příklad. Uvažujme polynomiální kód délky 5 s generujícím polynomem

$$g(x) = x^2 + 1 \in \mathbb{F}_2[x].$$

Protože

$$g(x) \nmid x^5 - 1$$

nad $\mathbb{F}_2$, není tento kód cyklický. Například $10100 \in \mathscr{C}$, ale $01001 \notin \mathscr{C}$.

Naopak polynomiální kód délky 4 se stejným generujícím polynomem již cyklický je, protože

$$(x^2 + 1) \mid (x^4 - 1)$$

nad $\mathbb{F}_2$, neboť $x^4 - 1 = (x^2 + 1)^2$. Kód obsahuje slova

$$\begin{aligned} 0 \cdot g(x) &= 0 && \overset{m}{\sim} 0000, \\ 1 \cdot g(x) &= x^2 + 1 && \overset{m}{\sim} 0101, \\ x \cdot g(x) &= x^3 + x && \overset{m}{\sim} 1010, \\ (x+1) \cdot g(x) &= x^3 + x^2 + x + 1 && \overset{m}{\sim} 1111. \end{aligned}$$

Za zmínku stojí to, že tato množina kódových slov vznikla z předchozího kódu délky 5. Uvážili jsme všechna kódová slova začínající nulou a tato nula se odřízla. □

Na konci předchozího příkladu byl nastíněn postup zkracování polynomiálního kódu. Chceme-li zkrátit délku o 1, vynecháme bázový polynom s nejvyšším stupněm. Pomocí bázových polynomů nižších stupňů nemůžeme dostat jedničku na první pozici zleva, proto bereme vždy jen slova, která začínají nulou. Všechna ostatní slova pomocí zbývajících bázových polynomů nagenerovat dokážeme.

Poznámka 1.4.3. *Nechť $\mathscr{C}$ je $[n, k]$-kód. Uvažujme množinu slov kódu $\mathscr{C}$, kde všechna slova na souřadnici i, kde $i < n$, mají nulu. Vypuštěním této nuly získáme množinu slov, kterou nazýváme zkráceným kódem kódu $\mathscr{C}$. Zkrácený kód dostáváme i v případě vícenásobného vypuštění souřadnic.*

Tvrzení 1.4.4. *Nechť $g(x), h(x) \in \mathbb{F}_q[x]$ jsou monické polynomy splňující*

$$g(x) \cdot h(x) = x^n - 1, \ \deg(h(x)) = k > 0, \ \deg(g(x)) = n - k,$$

kde $g(x)$ je generující polynom cyklického kódu délky n. Potom

$$\boldsymbol{C} = \begin{pmatrix} g_{n-k} & \cdots & g_1 & g_0 & & \\ & g_{n-k} & \cdots & g_1 & g_0 & \\ & & \ddots & & \ddots & \ddots \\ & & & g_{n-k} & \cdots & g_1 & g_0 \end{pmatrix}$$

$$\boldsymbol{H} = \begin{pmatrix} h_0 & h_1 & \cdots & h_k & & \\ & h_0 & h_1 & \cdots & h_k & \\ & & \ddots & \ddots & & \ddots \\ & & & h_0 & h_1 & \cdots & h_k \end{pmatrix}$$

Důkaz. Tvrzení C.3 v [19].

□

Pozorování. *Jestliže je $g(x) = x^n - 1$ generující polynom polynomiálního kódu $\mathscr{C}$ délky n, potom $\mathscr{C} = \{\boldsymbol{o}\}$.*

Poznámka 1.4.5. *Generující a kontrolní matice cyklického kódu v tvrzení 1.4.4 jsou uvedené v MSbF řazení, a proto se mírně liší oproti běžně uváděnému tvaru.*

Příklad. Uvažujme cyklický kód délky 5 s generujícím polynomem

$$g(x) = x + 1 \in \mathbb{F}_2[x].$$

Potom

$$h(x) = (x^5 - 1)/g(x) = x^4 + x^3 + x^2 + x + 1.$$

Generující a kontrolní matice tohoto kódu jsou

$$\boldsymbol{C} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}, \quad \boldsymbol{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Vidíme, že kód má $2^4 = 16$ kódových slov. Navíc z kontrolní matice vyplývá, že se jedná o všechna slova, která mají sudou váhu.

□

Pozorování. *Z každého cyklického kódu můžeme vyrobit polynomiální kód jiné délky se stejným generujícím polynomem.*

Příklad. Z cyklického kódu délky 5 s generující maticí nad $\mathbb{F}_2$

$$\boldsymbol{C} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

vytvoříme polynomiální kódy jiných délek se stejným generujícím polynomem. Stačí z matice $\boldsymbol{C}$ odebrat prvních d sloupců a řádků (ekvivalentně posledních d řádků a sloupců), nebo naopak řádky a sloupce přidat. Například generující matice polynomiálních kódů délek 3, 4 a 6 jsou

$$\boldsymbol{C_3} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \boldsymbol{C_4} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \boldsymbol{C_6} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Tyto kódy jsou v tomto případě všechny cyklické, protože $(x + 1)$ dělí nad $\mathbb{F}_2$ $x^3 - 1$, $x^4 - 1$ i $x^6 - 1$.

□

2. Detekce chyb pomocí polynomiálních kódů

Druhá kapitola se zabývá podrobnou analýzou polynomiálních kódů a jejich schopností detekovat chyby. Zdrojem informací jsou převážně [1], [2] a [5]. Začátek kapitoly obsahuje úvod do problematiky detekce chyb, následuje teorie primitivních polynomů. Dále se text zaměřuje na analýzu polynomiálních kódů s primitivním generujícím polynomem, neboť to jsou v jistém smyslu nejlepší polynomiální kódy. Druhá polovina celé kapitoly popisuje schopnost polynomiálních kódů detekovat chyby.

Připomínka. *Jestliže kód odhaluje (detekuje) r chyb, myslíme tím, že odhaluje (detekuje) i libovolný menší počet chyb.*

2.1 Reprezentace chyb v polynomiálních kódech

Vzdálenost kódu určuje, kolik je daný kód schopný odhalit nebo opravit chyb. To je v případě lineárního kódu podle poznámky 1.1.13 rovno nejmenší váze slova ze všech nenulových kódových slov. Bohužel tato informace není z generujícího polynomu zřejmá.

Poznámka 2.1.1. *Binárním polynomiálním kódem $\mathscr{C}$ délky n budeme rozumět polynomiální kód $\mathscr{C} \subseteq \mathbb{F}_2^n$.*

Příklad. Uvažujme binární polynomiální kód délky 5 s generujícím polynomem

$$g(x) = x^2 + x + 1 \in \mathbb{F}_2[x].$$

Příslušná generující matice je

$$\boldsymbol{C} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Váha $w(11100) = 3$, ale přitom

$$w(11100 + 01110) = w(10010) = 2.$$

Vzdálenost kódu není bezprostředně vidět z generujícího polynomu.

□

Tvrzení 2.1.2. *Nechť $\boldsymbol{H}$ je kontrolní matice $[n, k]_q$-kódu $\mathscr{C}$. Je-li $r \in \mathbb{N}_0$ největší číslo takové, že každých r sloupců matice $\boldsymbol{H}$ je lineárně nezávislých, pak*

$$d(\mathscr{C}) = r + 1.$$

Důkaz. Tvrzení 2.6 v [19].

□

Máme-li generující polynom $g(x)$ polynomiálního kódu $\mathscr{C}$, tak s jistotou víme, že

$$d(\mathscr{C}) \leq w(\boldsymbol{g}).$$

Spočítáme kontrolní matici kódu a najdeme nejmenší $r < w(\boldsymbol{g})$, které splňuje tvrzení 2.1.2. Tak získáme informaci o vzdálenosti kódu. I tato metoda je ale pro delší kódy výpočetně velmi náročná.

Příklad. Mějme binární polynomiální kód z předchozího příkladu, jehož generující matice je

$$\boldsymbol{C} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Příslušná kontrolní matice je ve tvaru

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

V kontrolní matici se nachází stejné sloupce, tedy existují dva, které jsou lineárně závislé. Proto je $r = 1$ a podle tvrzení 2.1.2 je

$$d(\mathscr{C}) = 2.$$

□

Předpokládejme, že máme polynomiální $[n, k]_q$-kód $\mathscr{C}$ s generujícím polynomem $g(x)$ stupně $n-k$. Pomocí tohoto kódu kódujeme zprávy délky k a dostáváme kódová slova délky n. Takové kódové slovo $\boldsymbol{c}$ je jako polynom ve tvaru

$$c(x) = a(x) \cdot g(x),$$

kde $a(x) \in \mathbb{F}_q[x]$, jehož stupeň je nejvýše $n - 1$.

Kódové slovo $\boldsymbol{c}$ se odešle a příjemce obdrží slovo $\boldsymbol{r}$, které je stejné délky jako odeslané slovo $\boldsymbol{c}$. Dále předpokládejme, že při přenosu bylo kódové slovo $\boldsymbol{c}$ poškozeno, a tak $r(x)$ můžeme zapsat ve tvaru

$$r(x) = c(x) + e(x) = a(x) \cdot g(x) + e(x),$$

kde polynom $e(x) \in \mathbb{F}_2[x]$, jehož stupeň je nejvýše $n - 1$, označuje chybu při přenosu.

Definice 2.1.3 (Chybový polynom)**.** *Polynom nesoucí chybu při přenosu nazýváme* ***chybový polynom*** *a značíme $e(x)$ (pokud není řečeno jinak).*

Nechť I je množina souřadnic, ve kterých došlo při přenosu k chybě. Chybový polynom má pak tvar

$$e(x) = \sum_{i \in I} a_i x^i,$$

kde $a_i \in \mathbb{F}_q$. Pokud byl přenos bezchybný, je $e(x) = 0$.

K ověření, že je přijaté slovo kódové, musíme toto slovo vydělit generujícím polynomem. Pokud byl přenos bezchybný, bude zbytek po dělení nulový a po dekódování dostaneme původní zprávu. V opačném případě

$$\frac{r(x)}{g(x)} = \frac{a(x) \cdot g(x) + e(x)}{g(x)} = a(x) + \frac{e(x)}{g(x)}.$$

Mohou nastat následující varianty v závislosti na hodnotě $e(x)$:

- $e(x)$ je násobkem $g(x) \rightarrow$ nebude zahlášena chyba, dekódování projde v pořádku, ale výsledek bude špatně.
- $e(x)$ není násobkem $g(x)$ a $\deg e(x) < \deg g(x) \rightarrow$ bude zahlášena chyba, chybový polynom je roven zbytku po dělení a díky tomu odhalíme indexy, které byly poškozeny.
- $e(x)$ není násobkem $g(x)$ a $\deg e(x) \geq \deg g(x) \rightarrow$ bude zahlášena chyba, ale nebudeme schopni odhalit chybné indexy.

Obecně se snažíme, aby situace z prvního bodu nastávala co nejméně.

Pozorování. *Druhý bod mimo jiné říká, že polynomiální kód s generujícím polynomem stupně $d > 0$ odhalí libovolný počet chyb, pokud všechny nastaly pouze na d pozicích zprava (v MSbF řazení).*

Důkaz. Takový chybový polynom má totiž stupeň menší než generující polynom a nemůže ho tak generující polynom dělit.

□

Pokud bychom měli v binárním případě zaručeno, že

$$x^i \bmod g(x) \quad \forall i \in \{0, \ldots, n-1\}$$

jsou po dvou různé a $e(x) = x^i$ (slovo má jednu chybu), pak bychom byli schopni opravit chybu z druhého a z třetího bodu. K tomu si stačí vytvořit tabulku dvojic hodnot

$$\left(x^i,\, x^i \bmod g(x)\right).$$

Přijaté slovo vydělíme se zbytkem polynomem $g(x)$. Spočtený zbytek nalezneme v tabulce a najdeme tak příslušný x^i. Exponent i určuje souřadnici, kde došlo k chybě a jelikož jsme v binárním případě, máme jednoznačně určenou opravu. Formálně viz věta 2.3.1.

2.2 Primitivní polynomy

Pro popis binárních polynomiálních kódů schopných opravit jednu chybu budeme potřebovat teorii primitivních polynomů, které se věnuje následující část textu.

Značení. *Znakem $\mathbb{F}_q^*$ rozumíme multiplikativní grupu tělesa $\mathbb{F}_q$.*

Definice 2.2.1 (Primitivní prvek tělesa)**.** *Libovolný generátor $\mathbb{F}_q^*$ nazýváme **primitivní prvek** tělesa $\mathbb{F}_q$.*

Definice 2.2.2 (Primitivní polynom)**.** *Nechť α je primitivní prvek tělesa $\mathbb{F}_{q^m}$. Minimální polynom prvku α nad tělesem $\mathbb{F}_q$ nazýváme **primitivním polynomem**.*

Tvrzení 2.2.3. *Nechť $f(x) \in \mathbb{F}_q[x]$ je ireducibilní polynom stupně m. Tento polynom má v $\mathbb{F}_{q^m}$ kořen α. Prvky*

$$\alpha,\, \alpha^q, \ldots,\, \alpha^{q^{m-1}} \in \mathbb{F}_{q^m}$$

jsou navzájem různé a tvoří množinu všech kořenů polynomu $f(x)$. Proto se $f(x)$ nad $\mathbb{F}_{q^m}$ rozkládá na lineární členy.

Důkaz. Věta 3.7 v [20].

□

Lemma 2.2.4. *Nechť $G = \mathbb{F}_{q^m}^*$ je grupa, $\alpha \in G$. Potom prvky*

$$\alpha,\ \alpha^q, \ldots,\ a^{q^{m-1}}$$

mají stejný řád.

Důkaz. Řád grupy G je $q^m - 1$ a protože $\mathrm{NSD}(q^m - 1,\ q^i) = 1$, pak

$$\mathrm{ord}(\alpha) = \mathrm{ord}(\alpha^{q^i}).$$

□

Značení. *Rozkladové třídy budeme značit hranatými závorkami [ ].*

Tvrzení 2.2.5. *Nechť $f(x) \in \mathbb{F}_q[x]$ je ireducibilní monický polynom stupně m. Pak $f(x)$ je primitivní právě tehdy, když $\alpha = [x]$ je primitivní prvek tělesa*

$$\mathbb{F}_q[\alpha] \simeq \mathbb{F}_q[x]/(f(x)).$$

Důkaz. Začneme implikací $\Leftarrow$. Polynom $f(x)$ má za kořen prvek α. Izomorfismus grup zachovává řády prvků, a proto primitivní prvek α tělesa $\mathbb{F}_q[\alpha]$ se izomorfismem převede na primitivní prvek tělesa $\mathbb{F}_{q^m}$, protože

$$\mathbb{F}_q[\alpha] \simeq \mathbb{F}_{q^m}.$$

Polynom $f(x)$ je tedy minimální polynom primitivního prvku $\mathbb{F}_{q^m}$ a podle definice je primitivní.

Nyní implikace $\Rightarrow$. Protože je polynom $f(x)$ ireducibilní, je $\mathbb{F}_q[x]/(f(x))$ těleso. Nyní ukážeme, že

$$\alpha = [x] \in \mathbb{F}_q[x]/(f(x))$$

je primitivní prvek. Z předpokladu primitivity víme, že $f(x)$ má za kořen primitivní prvek $\beta \in \mathbb{F}_{q^m}$. Všechny kořeny jsou podle tvrzení 2.2.3

$$\beta,\ \beta^q, \ldots,\ \beta^{q^{m-1}} \in \mathbb{F}_{q^m}$$

a jsou opět primitivními prvky, protože mají všechny stejný řád podle lemma 2.2.4. Navíc α je kořenem $f(x)$ též, tedy α je roven jednomu z primitivních prvků β^i.

□

Pozorování. *Minimální polynom primitivního prvku $\alpha \in \mathbb{F}_{q^m}$ je stupně m.*

Důkaz. Označme $f(x)$ minimální polynom prvku α. Potom

$$\mathbb{F}_q[\alpha] \simeq \mathbb{F}_q[x]/(f(x))$$

je jeho kořenové rozšíření. Navíc $\mathbb{F}_q[\alpha]$ je nejmenší těleso obsahující α a zároveň z předpokladu víme, že $\alpha \in \mathbb{F}_{q^m}$, takže

$$|\mathbb{F}_q[\alpha]| \leq q^m.$$

Na druhou stranu neexistuje podtěleso tělesa $\mathbb{F}_{q^m}$ obsahující α, protože je to primitivní prvek $\mathbb{F}_{q^m}$. Proto

$$|\mathbb{F}_q[\alpha]| = q^m$$

a stupeň $f(x)$ je m.

□

Existuje algoritmický způsob, jak ověřit, zda je daný polynom primitivní. To popisuje následující tvrzení.

Tvrzení 2.2.6. *Nechť $f(x) \in \mathbb{F}_q[x]$ je monický ireducibilní polynom stupně m. Pak $f(x)$ je primitivní právě tehdy, když jsou hodnoty*

$$x^i \bmod f(x),\ i \in \{0, \ldots, q^m - 2\}$$

po dvou různé.

Důkaz. Podle tvrzení 2.2.5 stačí ověřit, že $[x] = \alpha$ je primitivní prvek tělesa

$$\mathbb{F}_q[\alpha] \simeq \mathbb{F}_q[x]/(f(x)).$$

To nastává právě tehdy, když

$$\alpha^i,\ i \in \{0, \ldots, q^m - 2\}$$

jsou po dvou různé, neboť v takovém případě dostaneme všechny nenulové prvky tělesa. To je v řeči rozkladových tříd ekvivalentní tomu, že

$$\alpha^i = [x]^i = [x^i] = x^i \bmod f(x),\ i \in \{0, \ldots, q^m - 2\}$$

jsou po dvou různé.

□

Na první pohled by se mohlo zdát, že algoritmus na ověřování primitivnosti polynomu je exponenciální ve stupni polynomu. Ve skutečnosti nás ale zajímá, zda je $\alpha = [x]$ generátorem grupy $G = \mathbb{F}^*_{q^m}$, kde

$$\mathbb{F}_{q^m} \simeq \mathbb{F}_q[x]/(f(x)).$$

S jistotou víme, že $x^{q^m-1} = 1$ v G. Všechny prvky grupy G dělí její řád q^m-1. Stačí tedy ověřovat pouze mocniny, které jsou děliteli řádu grupy G. To je formálně uvedeno v následujícím důsledku.

Důsledek 2.2.7. *Nechť $f(x) \in \mathbb{F}_q[x]$ je monický ireducibilní polynom stupně m. Pak $f(x)$ je primitivní právě tehdy, když platí*

$$x^i \bmod f(x) \neq 1 \quad \forall\ i \mid q^m - 1,\ i \neq q^m - 1.$$

Příklad. Mějme ireducibilní polynomy (ověříme např. pomocí PC)

$$f(x) = x^8 + x^4 + x^3 + x^2 + 1,$$
$$g(x) = x^8 + x^5 + x^4 + x^3 + 1.$$

Ukážeme, že polynom $f(x)$ je primitivní a polynom $g(x)$ není primitivní nad $\mathbb{F}_2$. Pokud bychom výpočet prováděli hloupým způsobem, museli bychom spočítat

$$2^8 - 1 = 255$$

dělení se zbytkem, což s dnešními počítači sice není žádný problém, ale pro polynomy stupně 32 by to byla už zbytečná zátěž. Podle důsledku 2.2.7 nám stačí ověřit dělitele 255. Prvočíselný rozklad je

$$255 = 3 \cdot 5 \cdot 17$$

a všechny dělitelé jsou

$$1,\ 3,\ 5,\ 15,\ 17,\ 51,\ 85,\ 255.$$

Dělitelé 1 a 255 jsou mimo hru, zbytek vychází následovně:

$$\begin{aligned}
x^3 \bmod f(x) &= x^3,\\
x^5 \bmod f(x) &= x^5,\\
x^{15} \bmod f(x) &= x^5 + x^2 + x,\\
x^{17} \bmod f(x) &= x^7 + x^4 + x^3,\\
x^{51} \bmod f(x) &= x^3 + x,\\
x^{85} \bmod f(x) &= x^7 + x^6 + x^4 + x^2 + x.
\end{aligned}$$

To dokazuje, že $f(x)$ je primitivní.

$$\begin{aligned}
x^3 \bmod g(x) &= x^3,\\
x^5 \bmod g(x) &= x^5,\\
x^{15} \bmod g(x) &= x^6 + x^3 + x^2 + x,\\
x^{17} \bmod g(x) &= 1.
\end{aligned}$$

To naopak dokazuje, že $g(x)$ primitivní není.

□

Tvrzení 2.2.8. *Primitivních polynomů nad $\mathbb{F}_q$ stupně m existuje přesně*

$$\frac{\varphi(q^m - 1)}{m}$$

kde φ je Eulerova funkce. Pravděpodobnost, že náhodně zvolený monický polynom stupně m nad $\mathbb{F}_q$ bude primitivní, je

$$\frac{\varphi(q^m - 1)/m}{q^m}$$

za předpokladu, že polynomy volíme s rovnoměrným rozdělením.

Důkaz. Potřebujeme spočítat, kolik existuje různých minimálních polynomů generátorů grupy $G = \mathbb{F}_{q^m}^*$. Počet generátorů je

$$\varphi(q^m - 1),$$

neboť $|G| = q^m - 1$. Mějme minimální polynom $g(x)$ pro nějaký generátor $\alpha \in G$. Podle tvrzení 2.2.3 má $g(x)$ kromě α dalších $m-1$ různých kořenů

$$\alpha,\ \alpha^q, \ldots,\ \alpha^{q^{m-1}}.$$

Každý z těchto kořenů je opět generátorem G podle lemma 2.2.4. Navíc minimální polynom je jednoznačně určený. Celkový počet primitivních polynomů je proto

$$\frac{\varphi(q^m - 1)}{m}.$$

Pravděpodobnost plyne z toho, že počet všech monických polynomů stupně m nad $\mathbb{F}_q$ je q^m.

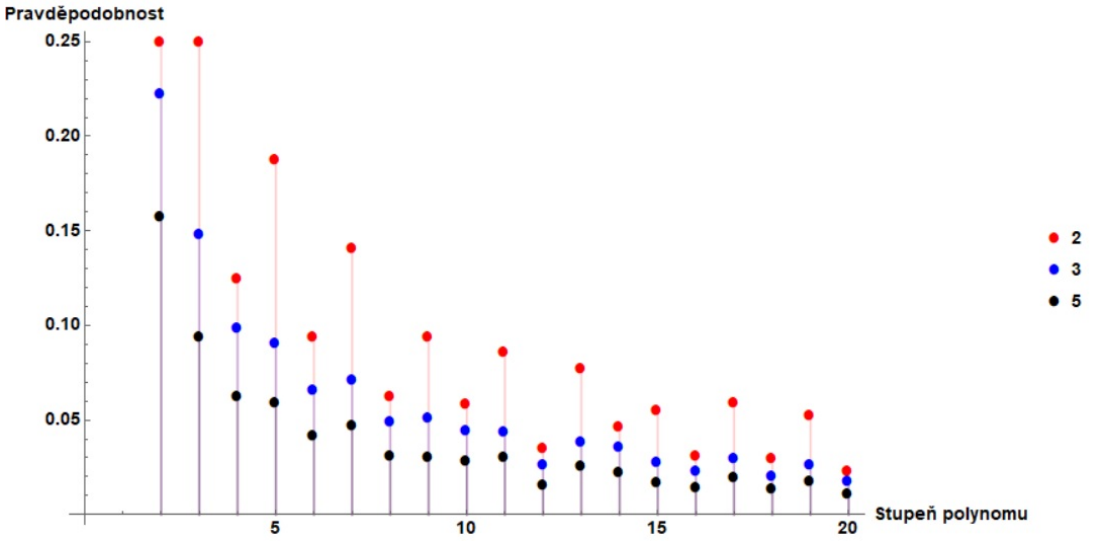
□

Poznámka 2.2.9. *Pravděpodobnost, že náhodně zvolený polynom stupně m nad $\mathbb{F}_q$ je primitivní, je možné shora odhadnout zlomkem $\frac{1}{m}$.*

Příklad. Pravděpodobnost, že je náhodně zvolený monický polynom stupně 5 a stupně 20 nad $\mathbb{F}_2$ primitivní, je

$$\frac{\varphi(2^5 - 1)/5}{2^5} = \frac{6}{32} \doteq 0{,}1875, \quad \frac{\varphi(2^{20} - 1)/20}{2^{20}} = \frac{375}{16384} \doteq 0{,}023.$$

□



Obrázek 2.1: Pravděpodobnost, že náhodně zvolený monický polynom stupně ≥ 2 je nad tělesy $\mathbb{F}_2$, $\mathbb{F}_3$ a $\mathbb{F}_5$ primitivní.

Pravděpodobnost nalezení primitivního polynomu se snižuje jak s rostoucím stupněm polynomu, tak s rostoucí velikostí tělesa, což ilustruje obrázek 2.1. Pravděpodobnost je ovlivněna hodnotou $\varphi(q^m - 1)$, která je maximální v případě, že $q^m - 1$ je prvočíslo. Nutnou podmínkou prvočíselnosti je $q = 2$ a takovým prvočíslům se říká Mersennova.

1	2	3	4	5
$x+1$	x^2+x+1	x^3+x+1	x^4+x+1	x^5+x^2+1
		x^3+x^2+1	x^4+x^3+1	x^5+x^3+1
				$x^5+x^3+x^2+x+1$
				$x^5+x^4+x^2+x+1$
				$x^5+x^4+x^3+x+1$
				$x^5+x^4+x^3+x^2+1$

Tabulka 2.1: Seznam primitivních polynomů stupňů 1, 2, 3, 4 a 5 nad $\mathbb{F}_2$.

Poznámka 2.2.10. *Existují 3 Mersennova prvočísla menší než 32 - 3, 7 a 31.*

Příklad. Tabulka 2.2 uvádí příklady primitivních polynomů nad $\mathbb{F}_2$. Primitivní polynom stupně jedna a dva je jen jeden, stupně tři a čtyři jsou dva a stupně pět je šest.

S rostoucím stupněm roste i počet primitivních polynomů. Například primitivních polynomů stupně 20 je 24000. Více příkladů viz [21].

□

2.3 Polynomiální kódy s primitivním generujícím polynomem

V této kapitole uvedeme nějaké vlastnosti polynomiálních kódů s primitivním generujícím polynomem, které nepřesáhnout jistou délku.

Věta 2.3.1. *Binární polynomiální kód $\mathscr{C}$ s primitivním generujícím polynomem $g(x) \in \mathbb{F}_2[x]$ stupně m délky $n \leq 2^m - 1$ opravuje jednu chybu, neboli*

$$d(\mathscr{C}) \geq 3.$$

Důkaz. Slovo délky $n \leq 2^m - 1$ s jednou chybou je ve tvaru

$$r(x) = a(x) \cdot g(x) + e(x),$$

kde $a(x) \in \mathbb{F}_2[x]$ a $e(x) = x^i$, $i \in \{0, \ldots, n-1\}$. Podle tvrzení 2.2.6 jsou zbytky po dělení

$$x^i \bmod g(x), \quad i = 0, \ldots, 2^{m-2}$$

po dvou různé. Z toho plyne, že

$$(x^i + x^j) \bmod g(x) = \left(x^i \bmod g(x)\right) + \left(x^j \bmod g(x)\right) \neq 0 \quad \forall i \neq j,$$

a tedy nemůže existovat kódové slovo váhy dva. Kódové slovo váhy jedna zřejmě nemůže existovat, a tak váha kódu je alespoň 3.

Každá z hodnot $x^i \bmod g(x)$ je jednoznačně určena monomem x^i. Podle zbytku

$$r(x) \bmod g(x) = e(x) \bmod g(x) = x^i \bmod g(x)$$

tak najdeme příslušný monom a jeho exponent i nám prozradí, která souřadnice je poškozená. Hodnotu na této pozici stačí změnit, protože v binárním případě máme pouze dvě možné hodnoty.

□

Pozorování. *Kód splňující předpoklady předchozí věty odhaluje dvě chyby.*

Poznámka 2.3.2. *Algoritmus zmíněný v důkazu věty 2.3.1 platí pouze pro binární kódy. Například v případě ternárních kódů ztrácíme jednoznačnost, neboť existují $i \neq j \in \mathbb{N}$, $i, j \leq 3^m - 2$ takové, že*

$$x^i \ mod\ g(x) = 2x^j \ mod\ g(x).$$

Proto nevíme, zda je chyba s hodnotou 1 na pozici i, nebo chyba s hodnotou 2 na pozici j.

Polynomiální binární kód s primitivním generujícím polynomem sice dokáže opravit jednu chybu, ale k tomu je potřeba tabulka, která sdružuje hodnoty x^i a $x^i \bmod g(x)$. Ta obsahuje $2^{\deg g(x)} - 1$ hodnot a roste exponenciálně ve stupni generujícího polynomu. Proto tato metoda není vhodná pro kódy s vysokým stupněm generujícího polynomu. Následující pozorování popisuje, jak takovou tabulku efektivně spočítat.

Pozorování. *Nechť $g(x) = \sum_{i=0}^{n-k} a_i x^i$ a $m_i(x) = x^i \ mod\ g(x)$. Hodnota $m_{i+1}(x)$ se spočítá následovně:*

$$m_{i+1}(x) = \begin{cases} x \cdot m_i(x), & \text{jestliže } \deg(m_i(x)) < \deg(g(x)) - 1, \\ x \cdot m_i(x) + g(x), & \text{jestliže } \deg(m_i(x)) = \deg(g(x)) - 1. \end{cases}$$

V počítači to lze realizovat pomocí bitového posunu a operace xor.

Příklad. Mějme polynom

$$g(x) = x^4 + x^3 + 1 \in \mathbb{F}_2[x]$$

a spočítáme prvních pár hodnot $x^i \bmod g(x)$ metodou z předchozího pozorování. Začneme hodnotou 0001. Následují hodnoty

$$0010,\ 0100,\ 1000.$$

Další hodnotu získáme tak, že provedeme bitový posun o jedna doleva jako v předchozích případech, ale nově jedničku odpovídající x^4 vypustíme a ke zbytku přičteme 1001 (operace xor), protože podle $g(x)$ je

$$x^4 = x^3 + 1 \overset{m}{\sim} 1001.$$

To odpovídá výpočtu $0000 + 1001 = 1001$. Takto pokračujeme dále a dostáváme 1011, 1111,...

□

Značení. *Znakem $\mathscr{C}_{hc}$ budeme značit binární polynomiální kód délky 15 s generujícím (ireducibilním) polynomem*

$$g(x) = x^4 + x^3 + 1 \in \mathbb{F}_2[x].$$

Příklad. Mějme polynomiální kód $\mathscr{C}_{hc}$. Vzdálenost kódu je ≤ 3, což vidíme ihned z generujícího polynomu. Tabulka hodnot $x^i \bmod g(x)$ je

$$\begin{aligned} 1 &\to 0001,\ x &&\to 0010,\ x^2 &&\to 0100,\\ x^3 &\to 1000,\ x^4 &&\to 1001,\ x^5 &&\to 1011,\\ x^6 &\to 1111,\ x^7 &&\to 0111,\ x^8 &&\to 1110,\\ x^9 &\to 0101,\ x^{10} &&\to 1010,\ x^{11} &&\to 1101,\\ x^{12} &\to 0011,\ x^{13} &&\to 0110,\ x^{14} &&\to 1100 \end{aligned} \tag{2.1}$$

V seznamu jsou všechny čtveřice různé, a proto je $g(x)$ primitivní polynom. Tabulku pro ověření primitivnosti není potřeba vytvářet celou podle důsledku 2.2.7, ale využijeme ji v další části textu. Generující matice tohoto kódu je ve tvaru

$$\boldsymbol{C} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & & & & & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Protože je generující polynom primitivní a délka kódu je ≤ 15, plyne z kombinace věty 2.3.1 a předchozího, že

$$d(\mathscr{C}_{hc}) = 3.$$

□

Tvrzení 2.3.3. *Nechť $\mathscr{C}$ je binární polynomiální kód délky $n \leq 2^m - 1$ s generujícím polynomem $g(x) \in \mathbb{F}_2[x]$. Sloupce kontrolní matice tohoto kódu odpovídají transponovaným slovům délky* $\deg(g(x))$

$$x^i \ mod\ g(x), \quad i = n-1, \ldots, 0$$

přesně v tomto pořadí.

Důkaz. Slovo $\boldsymbol{u} \in \mathbb{F}_2^n$ je v kódu $\mathscr{C}$ právě tehdy, když $u(x)$ je dělitelné generujícím polynomem $g(x)$ a to je právě tehdy, když

$$0 = u(x) \bmod g(x) = \left(\sum_{i=0}^{n-1} u_i x^i\right) \bmod g(x) = \sum_{i=0}^{n-1} u_i \left(x^i \bmod g(x)\right).$$

Pokud máme matici $\boldsymbol{H}$ ve tvaru ze znění tvrzení, pak $H\boldsymbol{u}^T = \boldsymbol{o}^T$ právě tehdy, když $u(x) \bmod g(x) = 0$.

□

Příklad. Kód $\mathscr{C}_{hc}$ má kontrolní kontrolní matici ve tvaru

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

protože sloupce odpovídají zbytkům po dělení z (2.1).

□

Poznámka 2.3.4. *Hammingův (binární) kód je* $[2^m - 1,\ 2^m - m - 1]_2$*-kód se vzdáleností 3. Kontrolní matice Hammingova kódu se skládá z* $2^m - 1$ *sloupců délky* m *s vlastností, že žádné dva sloupce nejsou stejné (tj. všechny různé kombinace).*

Pozorování. *Kontrolní matice z předchozího příkladu je kontrolní maticí Hammingova* $[15,\ 11]_2$*-kódu.*

Na závěr si můžeme všimnout, že

$$(x^4 + x^3 + 1) \mid (x^{15} - 1)$$

nad $\mathbb{F}_2$, takže $\mathscr{C}_{hc}$ je i cyklický kód. Protože

$$\frac{x^{15} - 1}{x^4 + x^3 + 1} = x^{11} + x^{10} + x^9 + x^8 + x^6 + x^4 + x^3 + 1,$$

je kontrolní matice kódu $\mathscr{C}_{hc}$ (podle tvrzení 1.4.4) též ve tvaru

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

To, že polynomiální kód $\mathscr{C}_{hc}$ je zároveň cyklický i Hammingův, není náhoda. Taková vlastnost platí obecně pro binární polynomiální kódy s primitivním generujícím polynomem stupně m a délky $2^m - 1$. Zbývá zformulovat požadavek na generující polynom polynomiálního kódu, který zajistí jeho cykličnost.

Tvrzení 2.3.5. *Je-li generující polynom* $g(x) \in \mathbb{F}_q[x]$ *stupně* m *ireducibilní, pak je příslušný polynomiální kód délky* $q^m - 1$ *cyklický.*

Důkaz. Podle tvrzení 1.4.2 stačí ukázat, že

$$g(x) \mid x^{q^m - 1} - 1$$

nad $\mathbb{F}_q$. V tělese $\mathbb{F}_q[x]/(g(x))$ má polynom $g(x)$ kořen $\alpha = [x]$. Protože

$$\alpha \in \mathbb{F}_q[x]/(g(x)) \simeq \mathbb{F}_{q^m},$$

je α též kořenem polynomu $x^{q^m - 1} - 1$. Jelikož je α společný kořen obou polynomů a $g(x)$ je ireducibilní, platí, že

$$g(x) \mid x^{q^m - 1} - 1.$$

□

Důsledek 2.3.6. *Binární polynomiální kód délky* $2^m - 1$ *s primitivním generujícím polynomem je cyklický a zároveň se jedná o Hammingův* $[2^m - 1,\ 2^m - m - 1]_2$*-kód.*

Na závěr uvedeme tvrzení, které popisuje, že každý polynomiální kód můžeme prodloužit na cyklický kód.

Tvrzení 2.3.7. *Ke každému polynomiálnímu kódu délky* n *existuje cyklický kód délky* $n' \geq n$ *se stejným generujícím polynomem* $g(x) \in \mathbb{F}_q[x]$.

Důkaz. Stačí ukázat, že existuje $n' \geq n$ takové, že

$$g(x) \mid x^{n'} - 1.$$

Nechť

$$g(x) = h_1(x)^{e_1} \cdots h_k(x)^{e_k},$$

kde $e_i \in \mathbb{N}$, $k \in \mathbb{N}$, $h_i \in \mathbb{F}_q[x]$, je ireducibilní rozklad. Každý z ireducibilních polynomů $h_i(x)$ tvoří konečné těleso

$$\mathbb{F}_q[x]/(h_i(x)) \simeq \mathbb{F}_{q^{\deg(h_i(x))}}.$$

Protože $h_i(x)$ je ireducibilní polynom s kořenem $\alpha = [x]$, který je též kořenem polynomu $x^{q^{\deg(h_i(x))}-1} - 1$, tak

$$h_i(x) \mid x^{q^{\deg(h_i(x))}-1} - 1.$$

Pro dostatečně velké $a_i \in \mathbb{N}$ platí, že

$$h_i(x)^{e_i} \mid \left(x^{q^{\deg(h_i(x))}-1} - 1\right)^{e_1} \mid \left(x^{q^{\deg(h_i(x))}-1} - 1\right)^{q^{a_i}}.$$

Využitím Frobeniova endomorfismu dostáváme

$$\left(x^{q^{\deg(h_i(x))}-1} - 1\right)^{q^{a_i}} = x^{(q^{\deg(h_i(x))}-1)q^{a_i}} - 1.$$

Nyní stačí zvolit n' tak velké, aby

$$\prod_i^k \left(x^{(q^{\deg(h_i(x))}-1)q^{a_i}} - 1\right) \mid \left(x^{n'} - 1\right).$$

Označme

$$m = \mathrm{NSN}((q^{\deg(h_i(x))} - 1)q^{a_i}).$$

Protože

$$(q^{\deg(h_i(x))} - 1)q^{a_i} \mid m \quad \forall i \in \{1, \ldots, k\},$$

tak i

$$\left(x^{(q^{\deg(h_i(x))}-1)q^{a_i}} - 1\right) \mid (x^m - 1) \quad \forall i \in \{1, \ldots, k\},$$

neboť platí

$$x^k - 1 \mid x^l - 1 \Leftrightarrow k \mid l.$$

Zbývá provést stejný trik s Frobeniovým endomorfismem na polynom $(x^m - 1)^k$, a tak najdeme hledané n'. Musíme umocnit na počet členů, protože polynomy $x^{(q^{\deg(h_i(x))}-1)q^{a_i}} - 1$ mohou být soudělné.

□

Příklad. Najdeme n' pro polynom

$$g(x) = (x+1)(x^2+x+1)^2(x^3+x+1)^3 \in \mathbb{F}_2[x].$$

Z postupu důkazu tvrzení 2.3.7 vidíme, že

$$\begin{aligned} x+1 &\mid x+1, \\ (x^2+x+1)^2 &\mid (x^3+1)^2 = x^6+1, \\ (x^3+x+1)^3 &\mid (x^7+1)^3 \mid (x^7+1)^4 = x^{28}+1. \end{aligned}$$

Spočítáme nejmenší společný násobek exponentů NSN(1, 6, 28) = 84. Proto

$$g(x) \mid (x^{84}+1)^3 \mid (x^{84}+1)^4 = x^{336}+1,$$

a tedy $n' = 336$.

Poznámka 2.3.8. *Metodou výše nalezneme běžně mnohem větší n', než by bylo potřeba.*

Pozorování. *Zkrácením cyklického kódu dostaneme kód polynomiální. Generující polynom zůstává zachován, ztrácí se pouze vlastnost cykličnosti (příklady viz závěr první kapitoly).*

Důsledek 2.3.9. *Každý polynomiální kód můžeme prodloužit na kód cyklický a každý cyklický kód můžeme zkrátit na kód polynomiální.*

2.4 Schopnost polynomiálních kódů detekovat chyby

Jak dobrý je polynomiální kód v odhalování a opravování chyb závisí na generujícím polynomu. V předchozí kapitole jsme odvodili, že pokud je takový polynom primitivní a binární kód není příliš dlouhý, dokáže kód opravit jednu a odhalit dvě chyby. Existují další kritéria, která ovlivňují detekční vlastnosti kódu. Více o nich v této kapitole.

Poznámka 2.4.1. *Od této chvíle budeme v celé druhé kapitole pracovat nad tělesem $\mathbb{F}_2$, nebude-li řečeno jinak.*

Definice 2.4.2 (Váha polynomu)**.** ***Váhu polynomu*** *definujeme jako počet jeho nenulových koeficientů.*

Pozorování. *Hammingova váha slova splývá s váhou odpovídajícího polynomu.*

Tvrzení 2.4.3. *Polynomiální kód libovolné délky s generujícím polynomem váhy alespoň dva odhaluje jednu chybu.*

Důkaz. Chyba na souřadnici i odpovídá chybovému polynomu x^i. Pro odhalení jedné chyby je důležité, aby generující polynom nedělil žádné z x^i. Jediný polynom, který dělí x^i, je ve tvaru

$$x^j,\ j \leq i,$$

a takový polynom je váhy jedna.

□

Nejjednodušší polynom váhy dva je $x + 1$. Generující polynomy váhy jedna nemá význam uvažovat (to jsou polynomy tvaru x^i, $i \in \mathbb{N}$), protože takové kódy přidávají ke zprávám pouze nulové bity.

Tvrzení 2.4.4. *Polynomiální kód libovolné délky s generujícím polynomem*

$$g(x) = x + 1$$

odhaluje libovolný lichý počet chyb.

Důkaz. Stačí si uvědomit, že libovolný polynom $h(x) \in \mathbb{F}_2[x]$ splňující

$$(x+1) \mid h(x),$$

má sudou váhu. Takový polynom má za kořen 1 a pokud by jeho váha byla lichá, nemohla by být 1 kořenem. Proto má každé kódové slovo sudý počet jedniček. Když se u takového slova změní lichý počet souřadnic, výsledné slovo bude mít lichý počet jedniček a nebude kódovým slovem.

□

Poznámka 2.4.5. *Mějme polynomiální kód $\mathscr{C}$ s generujícím polynomem $g(x)$ a polynomiální kód $\mathscr{C}'$, jehož generující polynom je násobkem $g(x)$. Pak*

$$\mathscr{C}' \subseteq \mathscr{C},\ d(\mathscr{C}') \geq d(\mathscr{C}).$$

Speciálně každý polynomiální kód s generujícím polynomem, který je dělitelný polynomem $x+1$, dokáže odhalit libovolný lichý počet chyb. To proto, že kód s generujícím polynomem $x+1$ obsahuje pouze slova sudé váhy.

Důsledek 2.4.6. *Polynomiální kód libovolné délky s generujícím polynomem*

$$g(x) = x^c + 1,\ c \in \mathbb{N},$$

detekuje libovolný lichý počet chyb.

Důkaz. Protože

$$x^c + 1 = (x+1)(x^{c-1} + \cdots + 1),$$

je polynom $x^c + 1$ dělitelný polynomem $x+1$. Zbytek plyne z tvrzení 2.4.4 a poznámky 2.4.5.

□

Definice 2.4.7 (Exponent polynomu)**.** *Nechť $f(x) \in \mathbb{F}_2[x]$. Nejmenší $e \in \mathbb{N}$ takové, že*

$$f(x) \mid x^e + 1,$$

definujeme jako ***exponent polynomu*** *$f(x)$. Jestliže žádné takové e neexistuje, řekneme, že exponent polynomu $f(x)$* ***neexistuje****.*

Pozorování. *Exponent polynomu, který má za kořen nulu, neexistuje. Takové polynomy jsou ve tvaru*

$$x^i \cdot h(x),\ i > 0.$$

Speciálně exponent neexistuje u polynomů $f(x) \neq 1$ váhy 1.

Tvrzení 2.4.8. *Nechť $g(x) \in \mathbb{F}_2[x]$ je polynom s exponentem $e \in \mathbb{N}$. Polynomiální kód s generujícím polynomem $g(x)$ délky $n \leq e$ odhaluje dvě chyby.*

Důkaz. Podle tvrzení 2.4.3 polynomiální kód odhaluje jednu chybu. Dvojitá chyba má chybový polynom ve tvaru $x^i + x^j$, kde $i \neq j < n$. Za předpokladu, že $i < j$, můžeme psát

$$x^i + x^j = x^i \cdot (1 + x^{j-i}).$$

Protože $x \nmid g(x)$, stačí pro detekci dvou chyb ověřit, že generující polynom nedělí žádný z polynomů

$$x^k + 1,\ k < n.$$

To platí z definice exponentu polynomu, protože $j - i < n \leq e$.

□

Pro každý ireducibilní polynom stupně m nad $\mathbb{F}_2$ platí, že

$$e \leq 2^m - 1.$$

To proto, že rozkladové nadtěleso takového polynomu je konečné těleso $\mathbb{F}_{2^m}$ (viz tvrzení 2.2.3) a všechny jeho kořeny jsou i kořeny polynomu $x^{2^m-1} + 1$.

Poznámka 2.4.9. *Exponent primitivního polynomu stupně m je roven $2^m - 1$.*

Důkaz. Primitivní polynom obsahuje kořen, jehož řád je $2^m - 1$ v grupě $\mathbb{F}^*_{2^m}$, a tedy nemůže dělit polynom $x^k + 1$, kde $k < 2^m - 1$.

□

Příklad. Spočítáme exponent polynomu

$$g(x) = x^4 + x^3 + x^2 + x + 1 \in \mathbb{F}_2[x],$$

který je ireducibilní (ověříme např. pomocí PC). Je zřejmé, že $e > 4$ a $e \leq 15$. Ve skutečnosti hledáme řád libovolného jeho kořenu (protože podle lemmatu 2.2.4 mají všechny stejný řád) v grupě

$$(\mathbb{F}_2[x]/(g(x)))^* \simeq \mathbb{F}^*_{16}.$$

Jedním z kořenů je $[x] = \alpha$. Řád prvku dělí řád grupy a zároveň $e > 4$, takže stačí ověřit pouze

$$x^5 + 1 \bmod g(x) = 0.$$

Z toho plyne, že exponent polynomu je 5, a tak nejde o primitivní polynom (α není generátor).

Podle tvrzení 2.4.8 polynomiální kód s generujícím polynomem $g(x)$ délky 5 odhalí dvě chyby (vzdálenost kódu je alespoň 3). Tento kód obsahuje pouze dvě kódová slova

$$11111,\ 00000.$$

Jak vidíme, tak jeho vzdálenost je dokonce 5, ale takový kód není moc užitečný. Naopak kód délky 6 už dvě chyby neodhalí (vzdálenost kódu je menší než 3). Obsahuje 4 kódová slova

$$111110,\ 011111,\ 000000,\ 100001.$$

Vzdálenost tohoto kódu je 2.

□

Kombinace poznámky 2.4.9 a tvrzení 2.4.8 poskytuje jiný důkaz toho, že polynomiální kódy délky $2^m - 1$ s primitivním generujícím polynomem stupně m dokáží odhalit dvě chyby. Původní důkaz je v rámci věty 2.3.1.

Tvrzení 2.4.10. *Mějme generující polynom polynomiálního kódu tvaru*

$$g(x) = (x+1) \cdot g_1(x) \in \mathbb{F}_2[x],$$

kde $g_1(x)$ je váhy alespoň dva s exponentem $e \in \mathbb{N}$. Je-li délka kódu $n \leq e$, pak kód odhaluje dvě a libovolný lichý počet chyb. Navíc je schopen opravit jednu chybu.

Důkaz. Protože

$$(x+1) \mid g(x),\ g_1(x) \mid g(x),$$

leží tento kód podle poznámky 2.4.5 v průniku množin slov kódů stejné délky s generujícími polynomy $g_1(x)$ a $x+1$.

Polynom $x+1$ generuje podle tvrzení 2.4.4 kód odhalující libovolný lichý počet chyb. Polynom $g_1(x)$ splňuje předpoklady tvrzení 2.4.8. Celkově tedy tento kód odhaluje dvě chyby a libovolný lichý počet chyb. Jeho vzdálenost je alespoň 4 (odhaluje alespoň 3 chyby), a proto dokáže opravit alespoň jednu chybu.

□

Důsledek 2.4.11. *Nechť $g_1(x) \in \mathbb{F}_2[x]$ je primitivní polynom stupně m. Polynomiální kód s generujícím polynomem*

$$g(x) = (x+1) \cdot g_1(x)$$

délky $n \leq 2^m - 1$ odhaluje dvě a libovolný lichý počet chyb. Navíc je schopen opravit jednu chybu.

Důkaz. Kombinace poznámky 2.4.9 a tvrzení 2.4.10.

□

Příklad. Polynomiální kód s generujícím polynomem

$$g(x) = (x+1)(x^3+x+1) = x^4+x^3+x^2+1 \in \mathbb{F}_2[x]$$

délky 7 splňuje předpoklady důsledku 2.4.11. Kód obsahuje následující kódová slova

$$0000000,\ 1110100,\ 0111010,\ 0011101$$
$$0100111,\ 1001110,\ 1101001,\ 1010011.$$

Všechna obsahují sudý počet jedniček, a proto kód odhalí lichý počet chyb. Navíc vidíme, že vzdálenost kódu je 4, proto navíc odhalí tři chyby a opraví jednu chybu.

Dostaneme-li nekódové slovo 1011101, opravíme ho tak, že buď najdeme nejbližší slovo z kódových slov a nebo použijeme následující postup. Jednu chybu je schopen opravit již kód s generujícím polynomem $g_1(x)$. Zbytek po dělení nekódového slova polynomem $g_1(x)$ je roven

$$(x^6+x^4+x^3+x^2+1) \bmod (x^3+x+1) = x^2+1.$$

Zbytek x^2+1 odpovídá monomu x^6, protože

$$x^6 \bmod (x^3+x+1) = x^2+1$$

(viz seznam (2.1)). Chyba je proto na první pozici a opravené slovo je ve tvaru

$$0011101.$$

□

Na závěr uvedeme tvrzení, které udává dolní odhad Hammingovy vzdálenosti polynomiálního kódu, jehož generující polynom je speciálního tvaru.

Tvrzení 2.4.12. *Nechť d, m, $n \in \mathbb{F}_2$, $n = 2^m - 1$, $d \leq n$, a nechť $\alpha \in \mathbb{F}_{2^m}$ je primitivní prvek. Označme $m_i(x) \in \mathbb{F}_2[x]$ minimální polynom prvku α^i, kde $i \in \{1, \ldots, d-1\}$. Definujme polynom*

$$g(x) = NSN(m_1(x), \ldots, m_{d-1}(x)) \in \mathbb{F}_2[x].$$

Polynomiální kód $\mathscr{C}$ délky n s generujícím polynomem $g(x)$ je cyklický a

$$d(\mathscr{C}) \geq d.$$

Důkaz. Protože jsou minimální polynomy ireducibilní, a tedy navzájem nesoudělné nebo stejné, je jejich nejmenší společný násobek roven jejich součinu bez duplicit. Pro důkaz cykličnosti stačí podle tvrzení 1.4.2 ukázat, že

$$g(x) \mid x^n - 1.$$

Protože $0 \neq \alpha \in \mathbb{F}_{2^m}$, je

$$(\alpha^i)^{2^m-1} = (\alpha^i)^n = 1 \quad \forall i \in \{1, \ldots, d-1\}.$$

Proto všechny minimální polynomy $m_i(x)$ dělí $x^n - 1$, a tak i $g(x) \mid x^n - 1$.

Odhad minimální váhy dokážeme sporem. Předpokládejme, že existuje kódové slovo váhy $b < d$ ve tvaru

$$c(x) = x^{k_1} + \cdots + x^{k_b}, \quad k_1 < \cdots < k_b.$$

Protože $\alpha, \ldots, \alpha^b$ jsou kořeny polynomu $g(x)$ a

$$c(x) = a(x) \cdot g(x), \quad a(x) \in \mathbb{F}_2[x],$$

tak pro $\forall i \in \{1, \ldots, b\}$ platí

$$c(\alpha^i) = \alpha^{ik_1} + \cdots + \alpha^{ik_b} = 0.$$

V maticovém zápisu dostáváme

$$\begin{pmatrix} \alpha^{k_1} & \alpha^{k_2} & \ldots & \alpha^{k_b} \\ \alpha^{2k_1} & \alpha^{2k_2} & \ldots & \alpha^{2k_b} \\ \vdots & \vdots & & \vdots \\ \alpha^{b \cdot k_1} & \alpha^{b \cdot k_2} & \ldots & \alpha^{b \cdot k_b} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2.2}$$

Nyní spočítáme determinant matice soustavy.

$$\begin{vmatrix} \alpha^{k_1} & \alpha^{k_2} & \ldots & \alpha^{k_b} \\ \alpha^{2k_1} & \alpha^{2k_2} & \ldots & \alpha^{2k_b} \\ \vdots & \vdots & & \vdots \\ \alpha^{b \cdot k_1} & \alpha^{b \cdot k_2} & \ldots & \alpha^{b \cdot k_b} \end{vmatrix} = \prod_{i=1}^{b} \alpha^{k_i} \begin{vmatrix} 1 & 1 & \ldots & 1 \\ \alpha^{k_1} & \alpha^{k_2} & \ldots & \alpha^{k_b} \\ \vdots & \vdots & & \vdots \\ \alpha^{(b-1)k_1} & \alpha^{(b-1)k_2} & \ldots & \alpha^{(b-1)k_b} \end{vmatrix}$$

$$= \left(\prod_{i=1}^{b} \alpha^{k_i} \right) \left(\prod_{1 \leq i < j \leq b} \left(\alpha^{k_j} - \alpha^{k_i} \right) \right),$$

kde poslední rovnost dostáváme z determinantu Vandermondovy matice. Protože je $\mathrm{ord}(\alpha) = n > b$, jsou α^i po dvou různé, a proto je tento determinant nenulový. Tím ale dostáváme spor, protože jediné řešení soustavy (2.2) je

$$\begin{pmatrix} 0 & 0 & \dots & 0 \end{pmatrix}^T \neq \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}^T$$

Proto $d(\mathscr{C}) \geq d$.

□

Poznámka 2.4.13. *Polynomiální kód s generujícím polynomem z tvrzení 2.4.12 se nazývá BCH kód v úzkém smyslu a odhadu se říká BCH mez. Toto tvrzení existuje v obecnější formě, ale pro naše účely stačí tato konkrétní formulace.*

2.5 Detekce shluku chyb

Definice 2.5.1 (Shluk chyb)**.** *Nechť $e(x) \in \mathbb{F}_2[x]$ je chybový polynom ve tvaru*

$$e(x) = x^i \cdot e_1(x),$$

*kde $i \in \mathbb{N}_0$ je největší možné. Potom slovo $\boldsymbol{e_1}$ nazveme **shlukem chyb** délky* $\deg(e_1(x)) + 1$.

Pozorování. *Shluk chyb začíná a končí jedničkou.*

Značení. *Znakem $\overset{e}{\sim}$ budeme rozumět převod mezi slovem reprezentující shluk chyb a chybovým polynomem (v MSbF řazení).*

Příklad.

$$e(x) = x^8 + x^4 + x^3 = x^3 \cdot (x^5 + x + 1) \overset{e}{\sim} 100011.$$

□

Definice 2.5.2 (Odhalit shluk chyb)**.** *Nechť $b \leq n$. Řekneme, že polynomiální kód délky n s generujícím polynomem $g(x)$ **odhalí všechny shluky chyb** délky b, jestliže*

$$g(x) \nmid x^i \cdot (1 + a_1 x + \cdots + a_{b-2} x^{b-2} + x^{b-1}) \quad \forall i \leq n - b,$$

kde $a_i \in \mathbb{F}_2$.

Tvrzení 2.5.3. *Polynomiální kód libovolné délky n s generujícím polynomem $g(x) \in \mathbb{F}_2[x]$ stupně m, $x \nmid g(x)$, odhalí všechny shluky chyb délky $b \leq m$.*

Důkaz. Stačí ověřit, že $g(x) \nmid e(x)$, kde

$$e(x) = x^i \cdot e_1(x) = x^i \cdot (1 + a_1 x + \cdots + a_{b-2} x^{b-2} + x^{b-1}), \tag{2.3}$$

$a_i \in \mathbb{F}_2$, $i \in \{0, \dots, n - b\}$. Protože $x \nmid g(x)$, stačí ověřit, že $g(x)$ nedělí žádný z $e_1(x)$ popsaných v (2.3). To je ale zřejmé ze stupňů polynomů, neboť

$$\deg(e_1(x)) = b - 1 < b \leq m = \deg(g(x)),$$

□

Tvrzení 2.5.4. *Mějme polynomiální kód libovolné délky n s generujícím polynomem $g(x) \in \mathbb{F}_2[x]$ stupně m, $x \nmid g(x)$. Dále předpokládejme uniformní pravděpodobnost výskytu shluku pevné délky. Pak pravděpodobnost, že shluk délky b, $n \geq b > m$, není odhalený, je*

$$\begin{aligned} 2^{-m}, &\text{ jestliže } b > m+1, \\ 2^{-(m-1)}, &\text{ jestliže } b = m+1. \end{aligned}$$

Důkaz. Chybový polynom shluku chyb je ve tvaru

$$e(x) = x^i \cdot e_1(x) = x^i \cdot (1 + a_1 x + \cdots + a_{b-2}x^{b-2} + x^{b-1}), \tag{2.4}$$

kde $a_i \in \mathbb{F}_2$, $i \in \{0, \ldots, n-b\}$. Protože $\deg(e_1(x)) = b-1$, je počet všech různých $e_1(x)$ roven 2^{b-2}, neboť $b-2$ koeficientů může mít hodnotu 0 nebo 1. Shluk není odhalený, pokud

$$e_1(x) = a(x) \cdot g(x),$$

kde $a(x) \in \mathbb{F}_2[x]$. Protože $\deg(g(x)) = m$ a $\deg(e_1(x)) = b-1$, pak

$$\deg(a(x)) = b - 1 - m.$$

Jestliže $b-1 = m$, pak $a(x) = 1$, a tedy existuje pouze jeden shluk, který není detekovaný. Ten je ve tvaru

$$e_1(x) = g(x).$$

Pravděpodobnost, že taková situace nastane, je

$$\frac{1}{2^{b-2}} = 2^{-(m-1)}.$$

Jestliže $b-1 > m$, pak $\deg(a(x)) > 0$ a navíc $a(x)$ obsahuje člen 1, protože $x \nmid e_1(x)$. Polynom $a(x)$ tedy obsahuje pevně dané dva členy 1 a x^{b-1-m} a ostatních $b-2-m$ členů je volitelných s koeficientem 0 nebo 1. Existuje zde proto 2^{b-2-m} různých voleb $a(x)$, a tedy

$$\frac{2^{b-2-m}}{2^{b-2}} = 2^{-m}$$

je pravděpodobnost, že shluk chyb není odhalen.

□

Definice 2.5.5. *Nechť $f(x)$, $g(x) \in \mathbb{F}_2[x]$. Řekneme, že polynom $f(x)$* ***nezasahuje*** *do polynomu $g(x)$, jestliže existuje $a \in \mathbb{N}_0$ takové, že buď*

$$f(x) = \sum_{i=0}^{a} f_i \cdot x^i, \; g(x) = \sum_{i=a+1}^{deg(g(x))} g_i \cdot x^i,$$

nebo

$$f(x) = \sum_{i=a+1}^{deg(f(x))} f_i \cdot x^i, \; g(x) = \sum_{i=0}^{a} g_i \cdot x^i,$$

kde f_i, $g_i \in \mathbb{F}_2$.

Příklad. Například polynom $x^3 + x$ nezasahuje do polynomu $x^5 + x^4$ a polynom $x^4 + x^3$ nezasahuje do polynomu $x^2 + x + 1$. Naopak polynom $x^6 + x^3$ zasahuje do polynomu $x^4 + x^2$.

□

Na jeden dlouhý shluk chyb se můžeme koukat jako na dva shluky kratší délky. Přesněji shluk

$$e(x) = x^k \cdot e_0(x)$$

můžeme rozepsat na dvě samostatné části následujícím způsobem

$$e(x) = x^i \cdot e_1(x) + x^j \cdot e_2(x), \tag{2.5}$$

kde polynom $x^i \cdot e_1(x)$ nezasahuje do polynomu $x^j \cdot e_2(x)$.

Poznámka 2.5.6. *Dále v textu budeme u zápisu dvou shluků z (2.5) automaticky předpokládat, že*

$$j > \deg(x^i \cdot e_1(x)) = i + \deg(e_1(x)),$$

tedy že $\boldsymbol{e_2}$ *je shluk ve slově nalevo od shluku* $\boldsymbol{e_1}$.

Příklad. Uvažujme chybový polynom

$$e(x) = x^8 + x^6 + x^3 + x^2 \overset{e}{\sim} 1010011.$$

Rozdělení na dva shluky není jednoznačné. Existují zde následující možnosti:

$$\begin{aligned} x^2 &\overset{e}{\sim} 1, & x^8 + x^6 + x^3 &\overset{e}{\sim} 101001 \\ x^3 + x^2 &\overset{e}{\sim} 11, & x^8 + x^6 &\overset{e}{\sim} 101 \\ x^6 + x^3 + x^2 &\overset{e}{\sim} 10011, & x^8 &\overset{e}{\sim} 1 \end{aligned}$$

□

Tvrzení 2.5.7. *Polynomiální kód délky n s generujícím polynomem ve tvaru*

$$g(x) = (x + 1) \cdot g_1(x) \in \mathbb{F}_2[x]$$

odhalí libovolnou kombinaci dvou shluků délek b_1, $b_2 \leq 2$, *jestliže* $n \leq e$, *kde e je exponent polynomu* $g_1(x)$.

Důkaz. Existují čtyři varianty dvojic shluků chyb délek b_1, $b_2 \leq 2$:

$$\begin{aligned} e(x) &= x^i + x^j \\ e(x) &= (x^i + x^{i+1}) + x^j \\ e(x) &= x^i + (x^j + x^{j+1}) \\ e(x) &= (x^i + x^{i+1}) + (x^j + x^{j+1}), \end{aligned}$$

kde $i \neq j$, $i \neq j+1$, $j \neq i+1$. Druhá a třetí varianta reprezentuje lichý počet chyb a ty jsou detekovány pomocí členu $(x + 1)$ (viz tvrzení 2.4.4). Čtvrtou variantu můžeme upravit do tvaru

$$e(x) = (x + 1) \cdot (x^i + x^j).$$

Člen $(x+1)$ se zkrátí se členem $x+1$ v $g(x)$. Pro zbytek důkazu čtvrté a důkazu první varianty stačí ukázat, že

$$g_1(x) \nmid (x^i + x^j).$$

To plyne z důkazu tvrzení 2.4.8.

□

Lemma 2.5.8. *Pro a, $c \in \mathbb{N}$ platí, že*

$$x^c + 1 \mid x^{ac} + 1.$$

Důkaz. Polynom $x^{ac}+1$ lze rozepsat do tvaru

$$x^{ac} + 1 = (x^c + 1) \cdot (x^{c(a-1)} + x^{c(a-2)} + \cdots + x^c + 1)$$

a z toho je znění zřejmé.

□

Tvrzení 2.5.9. *Mějme polynomiální kód délky n s generujícím polynomem ve tvaru*

$$g(x) = (x^c + 1) \cdot g_1(x) \in \mathbb{F}_2[x],$$

kde $g_1(x)$ je ireducibilní polynom stupně m_1 s exponentem e. Tento kód detekuje všechny kombinace dvou shluků chyb délek b_1 a b_2, jestliže

$$m_1 \geq min(b_1, b_2),\ n \leq NSN(c, e),\ c \geq b_1 + b_2 - 1.$$

Důkaz. Chybový polynom $e(x)$ dvou shluků přepíšeme do tvaru

$$e(x) = x^i \cdot (e_1(x) + x^{j-i} \cdot e_2(x)) = x^i \cdot e'(x), \tag{2.6}$$

kde $e_1(x)$ má stupeň $b_1 - 1$ a $e_2(x)$ má stupeň $b_2 - 1$. Potřebujeme ukázat, že

$$g(x) \nmid e'(x).$$

To stačí, protože $x \nmid g(x)$. Nejprve odvodíme tvar $e_1(x)$ a $e_2(x)$ za předpokladu, že $(x^c + 1) \mid e'(x)$. Označme

$$j - i = ac + r,\ r < c. \tag{2.7}$$

Nyní $e'(x)$ z (2.6) přepíšeme do tvaru

$$\begin{aligned} e'(x) &= e_1(x) + x^{ac+r} \cdot e_2(x) \\ &= e_1(x) + x^r \cdot e_2(x) + (x^{ac} + 1) \cdot (x^r \cdot e_2(x)). \end{aligned} \tag{2.8}$$

Podle lemmatu 2.5.8

$$(x^c + 1) \mid (x^{ac} + 1)(x^r \cdot e_2(x))$$

a protože předpokládáme, že $(x^c + 1) \mid e'(x)$, musí být i člen $e_1(x) + x^r \cdot e_2(x)$ dělitelný $x^c + 1$. Z toho dostáváme

$$e_1(x) + x^r \cdot e_2(x) = (x^c + 1) \cdot q(x). \tag{2.9}$$

Předpokládejme, že $q(x) \neq 0$ a označme $h = \deg(q(x))$. Srovnáním stupňů obou stran v (2.9) dostaneme

$$\begin{aligned} r + b_2 - 1 \geq c + h, \;\; \text{jestliže} \;\; \deg\left(x^r \cdot e_2(x)\right) \geq \deg\left(e_1(x)\right), \\ b_1 - 1 \geq c + h, \;\; \text{jestliže} \;\; \deg\left(x^r \cdot e_2(x)\right) \leq \deg\left(e_1(x)\right). \end{aligned} \tag{2.10}$$

Z předpokladu tvrzení víme, že $c \geq b_1 + b_2 - 1$, a protože

$$b_1 - 1 \geq c + h \geq b_1 + b_2 - 1 + h \implies 0 \geq b_2 + h,$$

může nastat pouze nerovnost (2.10)

$$r + b_2 - 1 \geq c + h,$$

neboť $b_2 > 0$ a $h \geq 0$. Použitím nerovnosti $c \geq b_1 + b_2 - 1$ na (2.10) dostáváme

$$b_1 + b_2 - 1 + h \leq c + h \leq r + b_2 - 1 \implies b_1 + h \leq r.$$

Protože je $b_1 > 0$ a $h \geq 0$, platí

$$h < r,\, b_1 \leq r.$$

Díky tomu je stupeň $e_1(x)$ a $q(x)$ menší než stupeň kteréhokoliv členu v $x^r \cdot e_2(x)$, a proto

$$x^r \cdot e_2(x) = x^c \cdot q(x). \tag{2.11}$$

Protože $e_2(x)$ obsahuje člen x^0, je v $x^r \cdot e_2(x)$ člen s nejnižší mocninou x^r. Naopak člen s nejnižší možnou mocninou v polynomu $x^c \cdot q(x)$ je x^c. Ale podle (2.7) je $r < c$, čímž dostáváme spor a rovnost v (2.11) nemůže nastat. Jedinou zbývající možností tak je, že v (2.9) je $q(x) = 0$, z čehož dostáváme

$$e_1(x) = x^r \cdot e_2(x).$$

Polynom $e_1(x)$ obsahuje člen x^0, proto musí být $r = 0$, což vede k rovnosti

$$e_1(x) = e_2(x).$$

Tu aplikujeme na (2.8) a dostaneme

$$e'(x) = e_2(x) \cdot (x^{ac} + 1). \tag{2.12}$$

To je tvar chybového polynomu, který je dělitelný polynomem $x^c + 1$. Zbývá dokázat, že tento chybový polynom není dělitelný polynomem $g_1(x)$. Polynom $g_1(x)$ je ireducibilní, tedy musí dělit buď $e_2(x)$, nebo $(x^{ac} + 1)$.

Nejprve začneme s $e_2(x)$. Protože $e_1(x) = e_2(x)$, tak i $b_1 = b_2$. Z předpokladu máme, že

$$\deg(g_1(x)) \geq \min(b_1,\, b_2) = b_1 = b_2$$

Polynom $g_1(x)$ má tedy větší stupeň než chybový polynom $e_2(x)$ (jeho stupeň je $b_2 - 1$), a proto ho nemůže dělit.

Zbývá ukázat, že $g_1(x)$ nemůže dělit ani $x^{ac}+1$. K tomu využijeme exponent polynomu $g_1(x)$. Rozepišme hodnotu ac do tvaru

$$ac = ue + v,$$

kde $v < e$. Platí, že $ac < n$, neboť stupeň chybového polynomu nemůže být větší nebo roven délce slova. Navíc $n \leq NSN(c,\, e)$, takže $e \nmid ac$, neboť nejmenší číslo, které je dělitelné c a zároveň e je právě $NSN(c,\, e)$. Proto je $v \neq 0$. Máme tak

$$x^{ac}+1 = x^{ue+v}+1 = x^v + 1 + x^v \cdot (x^{ue}+1).$$

Podle lemma 2.5.8 $(x^e+1) \mid (x^{ue}+1)$, tedy i

$$g_1(x) \mid (x^{ue}+1),$$

což plyne z jeho hodnoty exponentu. Protože je $0 \neq v < e$, tak

$$g_1(x) \nmid (x^v+1),$$

a proto $g_1(x) \nmid x^{ac}+1$. Tím je důkaz dokončen.

□

Příklad. Mějme polynomiální kód s generujícím polynomem

$$g(x) = (x+1)(x^3+x+1) = x^4+x^3+x^2+1 \in \mathbb{F}_2[x].$$

Exponent polynomu x^3+x+1 je 7, neboť se jedná o primitivní polynom. Podle tvrzení 2.5.3 odhalí polynomiální kód libovolné délky všechny shluky délky maximálně 4. Podle tvrzení 2.5.4 kód odhalí

$$(1-\frac{1}{8}) \cdot 100 \to 87{,}5\ \%\ \text{všech shluků délky } 5$$
$$(1-\frac{1}{16}) \cdot 100 \to 93{,}75\ \%\ \text{všech shluků délky větší než } 5.$$

Jestliže je kód délky maximálně 15, tak podle tvrzení 2.5.7 detekuje libovolnou kombinaci dvou shluků chyb délek menší než 2. Tvrzení 2.5.9 nám říká, že kód délky nejvýše NSN(1, 7) = 7 detekuje všechny dvojshluky délek $b_1 = 1$ a $b_2 = 1$. To je ekvivalentní tomu, že kód odhalí dvě chyby. To souhlasí se závěrem důsledku 2.4.11, který navíc tvrdí, že tento kód dokáže odhalit i libovolný lichý počet chyb.

□

Příklad. Mějme polynomiální kód s generujícím polynomem

$$g(x) = (x^5+1)(x^3+x+1).$$

Exponent polynomu x^3+x+1 je 7, NSN(5, 7) = 35, $m_1 = 3$, $c = 5$. Tvrzení 2.5.9 nám v tomto případě říká, že tento kód délky až 35 dokáže detekovat shluky délek b_1 a b_2 splňující

$$\min(b_1,\, b_2) \leq 3,\ \ b_1 + b_2 - 1 \leq 5.$$

Konkrétně kód s tímto generujícím polynomem délky 12 odhalí například chyby $e(x) = f(x) + g(x)$

$$\begin{aligned} f(x) &= x^2 \overset{e}{\sim} 1, & g(x) &= x^{11} + x^8 + x^7 \overset{e}{\sim} 10011, \\ f(x) &= x^4 + x^3 + x \overset{e}{\sim} 1101, & g(x) &= x^8 + x^7 \overset{e}{\sim} 11, \\ f(x) &= x^2 + 1 \overset{e}{\sim} 101, & g(x) &= x^5 + x^4 + x^3 \overset{e}{\sim} 111. \end{aligned}$$

Pro ilustraci to vyzkoušíme například na kódovém slovu

$$c(x) = (x^3 + 1)(x^5 + 1)(x^3 + x + 1) = x^{11} + x^9 + x^5 + x^4 + x + 1.$$

Po přičtení chybového polynomu dostáváme

$$\begin{aligned} c(x) + x^{11} + x^8 + x^7 + x^2 &= x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1, \\ c(x) + x^8 + x^7 + x^4 + x^3 + x &= x^{11} + x^9 + x^8 + x^7 + x^5 + x^3 + 1, \\ c(x) + x^5 + x^4 + x^3 + x^2 + 1 &= x^{11} + x^9 + x^3 + x^2 + x. \end{aligned}$$

Nyní ověříme, že to nejsou kódová slova.

$$\begin{aligned} x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \bmod g(x) &= x^3 + x, \\ x^{11} + x^9 + x^8 + x^7 + x^5 + x^3 + 1 \bmod g(x) &= x^7 + x^6 + x^5 + x^4 + x + 1, \\ x^{11} + x^9 + x^3 + x^2 + x \bmod g(x) &= x^5 + x^4 + x^3 + x^2 + 1, \end{aligned}$$

Podle tvrzení 2.5.3 a 2.5.4 tento kód odhalí všechny shluky délky 8, shluk délky 9 odhalí s pravděpodobností $1 - 2^{-7} \doteq 0{,}9922$ a shluk délky 10 a více odhalí s pravděpodobností $1 - 2^{-8} \doteq 0{,}9961$.

□

3. CRC kódy

Třetí kapitola popisuje speciální typ polynomiálních kódů - CRC kódy. Základní myšlenky první části kapitoly vychází z článku [5], ve kterém se pojem CRC kódu vyskytl poprvé. Druhá část, která pojednává o efektivní implementaci, je lehce inspirována textem [6].

Část kapitoly charakterizující CRC kódy pomocí lineárního zobrazení je originálním přínosem této práce. Totéž platí pro efektivní implementaci pomocí lineárního zobrazení. Existující teorie kolem CRC kódů obsahuje pouze algoritmus kódování popsaný v řeči dělení polynomů a drobné náznaky, že jde o polynomiální kódy. Žádný kvalitní matematický popis není v dostupné literatuře k dispozici. V této kapitole se povedlo existující algoritmus převést do řeči matic a zasadit do teorie polynomiálních kódů z předchozí kapitoly.

Poznámka 3.0.1. *V celé kapitole budeme pracovat nad tělesem* $\mathbb{F}_2$.

Připomínka. *Pokud není řečeno jinak, symbolem* $f(x)$ *značíme polynom a symbolem* $\boldsymbol{f}$ *značíme slovo reprezentující koeficienty polynomu* $f(x)$. *Délka slova* $\boldsymbol{f}$ *je*

$$k = \deg(f(x)) + 1,$$

pokud není uvedeno jinak. V opačném případě je vždy $\geq k$ *a chybějící souřadnice jsou doplněny zleva nulami. Slova uvažujeme v MSbF řazení a souřadnice zapisujeme zleva od nejvyššího indexu*

$$\boldsymbol{f} = f_{k-1} \dots f_0.$$

Pro zjednodušení zápisu budeme někdy symboly $\boldsymbol{f}$ *a* $f(x)$ *zaměňovat.*

Připomínka. *Polynomiální* $[n, k]$*-kód je kód délky* n *a dimenze* k, *jehož generující polynom je stupně* $n - k$.

3.1 Úvod do CRC kódů

Polynomiální kód je množina kódových slov a ke každému kódu existuje mnoho různých kódování, pomocí kterých zprávy informačního zdroje převádíme na kódová slova a naopak.

Mějme polynomiální $[n, k]$-kód $\mathscr{C}$ s generujícím polynomem $g(x)$ a zprávu $m(x)$ jako polynom stupně nejvýše $k-1$. Použijme takové kódovaní, které zprávě $m(x)$ přiřadí kódové slovo ve tvaru

$$m(x) \cdot g(x).$$

Dekódování pro toto konkrétní kódování funguje tak, že kódové slovo vydělíme generujícím polynomem a výsledkem je původní zpráva.

Značení. *Symbolem* $r_{m,g}(x)$ *budeme značit zbytek po dělení*

$$x^{n-k} \cdot m(x) \ mod \ g(x),$$

kde $n - k$ *je stupeň polynomu* $g(x)$.

Nyní si ukážeme jinou metodu kódování a dekódování pro stejný kód $\mathscr{C}$. Nejprve spočítáme

$$r_{m,g}(x) = x^{n-k} \cdot m(x) \bmod g(x).$$

Tento zbytek přičteme k polynomu $x^{n-k} \cdot m(x)$, a tak získáme kódové slovo kódu $\mathscr{C}$ ve tvaru

$$x^{n-k} \cdot m(x) + r_{m,g}(x) = a(x) \cdot g(x),$$

kde $a(x) \in \mathbb{F}_2[x]$.

Definice 3.1.1 (CRC kód)**.** *Nechť $\mathscr{C}$ je polynomiální $[n, k]$-kód s generujícím polynomem $g(x) \in \mathbb{F}_2[x]$ a*

$$C : \mathbb{F}_2^k \to \mathbb{F}_2^n$$

*je zobrazení (kódování). Dvojici $(\mathscr{C}, C)$ budeme nazývat **(n-bitový) CRC kód**, jestliže C je definované jako*

$$C(m(x)) = x^{n-k} \cdot m(x) + r_{m,g}(x).$$

*Zobrazení C tohoto tvaru budeme nazývat **CRC kódování**.*

Definice 3.1.2 (CRC polynom)**.** *Generující polynom kódu $\mathscr{C}$, kde $(\mathscr{C}, C)$ je CRC kód, budeme nazývat **CRC polynom**.*

Definice 3.1.3 (Kontrolní bity)**.** *Slovo $\boldsymbol{r_{m,g}}$ délky $\deg(g(x))$ budeme nazývat **kontrolní bity**.*

Poznámka 3.1.4. *CRC-m značí v literatuře CRC polynom stupně m.*

Značení. *CRC kód $(\mathscr{C}, C)$, kde $\mathscr{C}$ je polynomiální $[n, k]$-kód, budeme stručně označovat jako **CRC** $[n, k]$**-kód**.*

Zajímavé na CRC kódech je to, že dekódování je velice jednoduché. Zbytek po dělení $r_{m,g}(x)$ má totiž stupeň menší než $n - k$ a polynom $x^{n-k} \cdot m(x)$ má na $n - k$ pozicích zprava nulu. Proto je kódové slovo složeno z původní zprávy a $n-k$ kontrolních bitů. Pro dekódování stačí kontrolní bity odříznout. To můžeme udělat jen za předpokladu, že se jedná o kódové slovo. Slovo je kódové, jestliže je jako polynom dělitelné generujícím polynomem. V opačném případě o kódové slovo nejde.

Poznámka 3.1.5. *Protože jsou CRC kódy polynomiální kódy, týkají se jich všechna tvrzení odvozená v kapitole 2.*

Příklad. Mějme 10-bitový CRC kód s generujícím polynomem

$$g(x) = x^3 + x + 1 \in \mathbb{F}_2[x].$$

V tomto případě je $n = 10$, $k = 7$. Zakódujeme zprávu $\boldsymbol{m} = 0101101$, která je jako polynom ve tvaru

$$m(x) = x^5 + x^3 + x^2 + 1.$$

Nejprve spočítáme

$$x^{n-k} \cdot m(x) = x^3 \cdot (x^5 + x^3 + x^2 + 1) = x^8 + x^6 + x^5 + x^3 \overset{m}{\sim} 0101101000. \quad (3.1)$$

Nyní spočítáme zbytek po dělení generujícím polynomem a dostaneme tři kontrolní bity

$$r_{m,g}(x) = (x^8 + x^6 + x^5 + x^3) \bmod (x^3 + x + 1) = x + 1 \overset{m}{\sim} 011.$$

Tento zbytek přičteme k (3.1) a tím dostaneme výsledné kódové slovo

$$c(x) = x^{n-k} \cdot m(x) + r_{m,g}(x) = x^8 + x^6 + x^5 + x^3 + x + 1 \overset{m}{\sim} 0101101011.$$

□

Z úvah a příkladu výše plyne, že se bity zprávy při kódování nemění, pouze se posouvají. Díky tomu existuje popis CRC kódů jako kódu s lineárním kódováním.

Tvrzení 3.1.6. *Nechť $\mathscr{C}$ je polynomiální $[n, k]$-kód s generujícím polynomem $g(x)$ a kódováním $C : \mathbb{F}_2^k \to \mathbb{F}_2^n$. Potom $(\mathscr{C}, C)$ je CRC kód právě tehdy, když C je lineární zobrazení splňující*

$$C(\boldsymbol{m}) = \boldsymbol{m} \cdot (\boldsymbol{I_k}|\boldsymbol{A}),$$

kde $\boldsymbol{A} \in \mathbb{F}_2^{k \times n-k}$ je matice splňující

$$\begin{pmatrix} x^{n-1} & mod\ g(x) \\ \vdots & \\ x^{n-k+1} & mod\ g(x) \\ x^{n-k} & mod\ g(x) \end{pmatrix} = \boldsymbol{A} \cdot \begin{pmatrix} x^{n-k-1} \\ \vdots \\ x \\ 1 \end{pmatrix}$$

Důkaz. Protože $\deg(g(x)) = n - k$, jsou polynomy

$$x^{n-k+i} \bmod g(x), \;\; i = 0, \ldots, k-1$$

stupně menšího než $n - k$, a tak má matice $\boldsymbol{A}$ správný typ. Proto je matice

$$(\boldsymbol{I_k}|\boldsymbol{A}) \in \mathbb{F}_2^{k \times n}$$

a C je zobrazení z $\mathbb{F}_2^k$ do $\mathbb{F}_2^n$. Označíme-li

$$\begin{aligned} \boldsymbol{m} &= (m_{k-1} \;\cdots\; m_0) \in \mathbb{F}_2^k, \\ \boldsymbol{A} &= (\boldsymbol{a_{k-1}} \;\cdots\; \boldsymbol{a_0})^T \in \mathbb{F}_2^{k \times n-k}, \\ \boldsymbol{c} &= (c_{n-1} \;\cdots\; c_0) = \boldsymbol{m} \cdot (\boldsymbol{I_k}|\boldsymbol{A}) \in \mathbb{F}_2^n \end{aligned}$$

pak

$$\begin{aligned} &(c_{n-1} \;\cdots\; c_{n-k}) = (m_{k-1} \;\cdots\; m_0), \\ &(c_{n-k-1} \;\cdots\; c_0) = \boldsymbol{m} \cdot \boldsymbol{A} = \sum_{i=0}^{k-1} m_i \cdot \boldsymbol{a_i}. \end{aligned} \tag{3.2}$$

Nyní se podíváme, čemu odpovídá suma z (3.2). V řeči polynomů máme

$$\begin{aligned} \boldsymbol{m} \cdot \boldsymbol{A} \cdot (x^{n-k-1} \;\cdots\; x\ 1)^T &= \sum_{i=0}^{k-1} m_i \cdot \left(x^{n-k+i} \bmod g(x)\right) \\ &= \left(x^{n-k} \cdot \sum_{i=0}^{k-1} m_i \cdot x^i\right) \bmod g(x) \\ &= x^{n-k} \cdot m(x) \bmod g(x) = r_{m,g}(x), \end{aligned}$$

Přechodem od polynomům ke slovům dostáváme, že

$$\boldsymbol{m} \cdot \boldsymbol{A} = \boldsymbol{r_{m,g}},$$

a tedy

$$c(x) \overset{m}{\sim} (c_{n-1} \ \cdots \ c_0) = (m_{k-1} \ \cdots \ m_0 \ r_{n-k-1} \ \cdots \ r_0) \overset{m}{\sim} x^{n-k} \cdot m(x) + r_{m,g}(x),$$

což je přesně definice CRC kódu. Tím jsme dokázali implikaci $\Leftarrow$. Pro důkaz opačné implikace stačí nahlédnout, že předchozí úvahy byly ekvivalentní, a tedy stačí postupovat v důkazu od konce.

□

Z tvrzení 3.1.6 vyplývá, že pro kódování si stačí pamatovat pouze matici $\boldsymbol{A}$. Kontrolní bity dostaneme vynásobením zprávy zprava touto maticí.

Pozorování. *Generující matice* $(\boldsymbol{I_k}|\boldsymbol{A})$ *je ve standardním tvaru a platí pro ni tvrzení 1.1.9 o výpočtu kontrolní matice, která je ve tvaru* $(\boldsymbol{A}^T|\boldsymbol{I_{n-k}})$.

Příklad. Polynomiální [10, 7]-kód s generujícím polynomem

$$g(x) = x^3 + x + 1$$

z předchozího příkladu má podle tvrzení 1.2.4 generující matici

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

Spočítáme generující matici, která určuje kódování 10-bitového CRC kódu s generujícím polynomem $g(x)$. K tomu nám stačí určit matici $\boldsymbol{A}$ z tvrzení 3.1.6. Nejprve určíme zbytky po dělení monomů $x^3, \ldots, x^9$ polynomem $g(x)$. Dostáváme

$$\begin{aligned} x^3 \bmod g(x) &= x + 1, \\ x^4 \bmod g(x) &= x^2 + x, \\ x^5 \bmod g(x) &= x^2 + x + 1, \\ x^6 \bmod g(x) &= x^2 + 1, \\ x^7 \bmod g(x) &= 1, \\ x^8 \bmod g(x) &= x, \\ x^9 \bmod g(x) &= x^2. \end{aligned}$$

Koeficienty zbytků (odspoda) odpovídají hledané matici $\boldsymbol{A}$ a matice kódování je podle tvrzení 3.1.6 ve tvaru $(\boldsymbol{I_k}|\boldsymbol{A})$, což odpovídá matici

$$\boldsymbol{C'} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

V předchozím příkladu jsme kódovali zprávu 0101101. Nyní stačí pouze vynásobit zprávu maticí $\boldsymbol{C'}$ zprava, z čehož mimo jiné dostaneme tři kontrolní bity

$$011.$$

Kontrolní matice tohoto kódu je ve tvaru $(\boldsymbol{A}^T|\boldsymbol{I_3})$, což je konkrétně

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

□

Poznámka 3.1.7. *Generující matice, která určuje CRC kódování, popsaná v tvrzení 3.1.6 je v jednoznačně určeném redukovaném odstupňovaném tvaru. Z libovolného polynomiálního kódu dostaneme CRC kód také tak, že na jeho generující matici provedeme Gaussovu-Jordanovu eliminaci.*

Poznámka 3.1.8. *Díky důsledku 2.3.9 každý CRC kód můžeme prodloužit na kód cyklický a naopak každý CRC kód vznikl zkrácením nějakého CRC kódu.*

Nechť $(\mathscr{C}, C)$ je CRC-kód, kde $\boldsymbol{C} = (\boldsymbol{I_k}|\boldsymbol{A})$ (viz tvrzení 3.1.6). Zkrácení o $a < k$ souřadnic lze provést tak, že odebereme prvních k sloupců zleva a prvních k řádků matice $\boldsymbol{C}$. To je ekvivalentní tomu, že z delšího kódu vybereme kódová slova, která mají na prvních k pozicích nulu, a tyto nuly odřízneme. Tím získáme zkrácený kód.

3.2 Implementace kódování CRC kódů

Mějme n-bitový CRC kód s generujícím polynomem $g(x) \in \mathbb{F}_2[x]$, jehož stupeň je $n - k$. V předchozí sekci jsme v tvrzení 3.1.6 odvodili konkrétní tvar kódování C. Před výpočtem je vhodné si do paměti uložit matici $\boldsymbol{A}$, jejíž řádky odpovídají kontrolním bitům zpráv tvaru x^i, kde $i = k - 1, \ldots, 0$. Velikost této matice je

$$(n - k) \cdot k$$

bitů, což je součin počtu kontrolních bitů zprávy a délky zprávy, kterou kódujeme. Máme-li matici v paměti, pak samotné kódování je součtem nejvýše k hodnot délky $n - k$.

V další části textu rozebereme jiné způsoby implementace kódování. Z definice CRC kódu víme, že k výpočtu kontrolních bitů stačí spočítat

$$x^{n-k} \cdot m(x) \bmod g(x),$$

kde $\boldsymbol{m}$ je zpráva délky k, kterou chceme zakódovat.

Poznámka 3.2.1. *Sčítáme-li dvě slova různé délky, doplníme kratší z nich zleva nulami.*

Dělení se zbytkem polynomů nad $\mathbb{F}_2$ je jednodušší než dělení nad jinými okruhy. Navíc je velice jednoduché ho implementovat. Samotný výpočet se ve skutečnosti skládá pouze ze součtu slov.

Příklad. Ilustrujeme výpočet

$$(x^7 + x^3) \bmod (x^4 + x^2 + 1)$$

nad $\mathbb{F}_2$. Hodnotou $r^{(i)}(x)$ budeme značit zbytek v i-tém kroku.

$$\begin{aligned}
r^{(0)}(x) &= x^7 + x^3 \\
r^{(1)}(x) &= (x^7 + x^3) + (x^4 + x^2 + 1) \cdot x^3 = (x^7 + x^3) + (x^7 + x^5 + x^3) = x^5 \\
r^{(2)}(x) &= x^5 + 0 \cdot (x^4 + x^2 + 1) \cdot x^2 = x^5 \\
r^{(3)}(x) &= x^5 + (x^4 + x^2 + 1) \cdot x = x^5 + x^5 + x^3 + 1 = x^3 + 1 \\
r^{(4)}(x) &= x^3 + 1 + 0 \cdot (x^4 + x^2 + 1) = x^3 + 1.
\end{aligned}$$

Dostáváme tak, že

$$(x^7 + x^3) \bmod (x^4 + x^2 + 1) = x^3 + 1.$$

□

Výše uvedený příklad přepíšeme pomocí koeficientů v MSbF řazení.

$$\begin{aligned}
x^7 + x^3 &\overset{m}{\sim} 10001000 \\
(x^7 + x^3) + (x^7 + x^5 + x^3) \cdot x^3 &\overset{m}{\sim} 10001000 + 10101000 \\
x^5 &\overset{m}{\sim} 00100000 \\
x^5 + (x^5 + x^3 + x) &\overset{m}{\sim} 00100000 + 00101010 \\
x^3 + x &\overset{m}{\sim} 00001010
\end{aligned}$$

Vidíme, že k 10001000 přičítáme postupně 10101000 a 00101010. První hodnota reprezentuje polynom $f(x)$ a další hodnoty jsou bitově doprava posunuté zápisy polynomu $g(x)$. To zapsáno pod sebe vypadá následovně

$$\begin{array}{r}
1\,0\,0\,0\,1\,0\,0\,0 \\
1\,0\,1\,0\,1 \\
\quad 1\,0\,1\,0\,1 \\
\hline
\quad\quad 1\,0\,1\,0
\end{array}$$

Pozorování. *Algoritmus na výpočet zbytku po dělení polynomů nad $\mathbb{F}_2$ lze implementovat pouze za pomocí operace XOR a bitového posunu.*

Nevýhody tohoto postupu jsou minimálně dvě. Počítáme-li

$$f(x) \bmod g(x),$$

pak je počet součtů shora omezen hodnotou

$$\deg(f(x)) - \deg(g(x)),$$

kde $f(x)$ má typicky výrazně větší stupeň než $g(x)$. Druhá nevýhoda spočívá v tom, že vysoký stupeň polynomu $f(x)$ neumožní tento polynom reprezentovat v počítači běžnou celočíselnou proměnnou, jakou je například 32-bitový int. Z toho důvodu bude potřeba použít aritmetiku dlouhých čísel, se kterou budou procesory pracovat výrazně pomaleji.

V další části textu popíšeme modifikaci výpočtu, pomocí které se lze těmto dvěma nevýhodám vyhnout.

Definice 3.2.2 (Prefix a sufix slova)**.** *Nechť* $\boldsymbol{u} = u_{m-1} \dots u_0 \in \mathbb{F}_2^m$ *je slovo délky* $m \geq k \in \mathbb{N}$. ***Prefixem*** *slova* $\boldsymbol{u}$ *délky* k *rozumíme slovo*

$$u_{m-1} \dots u_{m-k}$$

a ***sufixem*** *slova* $\boldsymbol{u}$ *délky* k *rozumíme slovo*

$$u_{k-1} \dots u_0.$$

Definice 3.2.3. *Nechť* $\boldsymbol{u} = u_{m-1} \dots u_0 \in \mathbb{F}_2^m$, $u_0 = u_{m-1} = 1$, *je slovo délky* $m \geq k \in \mathbb{N}$. ***Bázi prefixů*** *slova* $\boldsymbol{u}$ *dimenze* k *definujeme jako množinu vektorů délky* k

$$\left\{ \begin{pmatrix} u_{m-1} \\ u_{m-2} \\ \vdots \\ u_{m-k} \end{pmatrix}, \begin{pmatrix} 0 \\ u_{m-1} \\ \vdots \\ u_{m-k+1} \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ \vdots \\ 0 \\ u_{m-1} \end{pmatrix} \right\}.$$

Bázi sufixů *slova* $\boldsymbol{u}$ *dimenze* k *definujeme jako množinu vektorů délky* k

$$\left\{ \begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} u_1 \\ u_0 \\ \vdots \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_0 \end{pmatrix} \right\}.$$

Pozorování. *Báze prefixů a sufixů dimenze* k *jsou báze vektorového prostoru* $\mathbb{F}_2^k$.

Pozorování. *Nechť*

$$\boldsymbol{g} = g_{m-1} \dots g_0,$$

kde $g(x) \in \mathbb{F}_2[x]$ *je CRC polynom. Hodnota* $g_{m-1} = 1$ *je zaručena vždy, neboť jde o vedoucí koeficient* $g(x)$*, a* $g_0 = 1$ *za předpokladu, že* $x \nmid g(x)$.

Poznámka 3.2.4. *V další části kapitoly předpokládáme, že* $\boldsymbol{g}$ *začíná a končí jedničkou, což je typicky u generujících polynomů splněno.*

Zapíšeme-li výpočet

$$(x^{11} + x^{10} + x^5) \bmod (x^4 + x^2 + x + 1)$$

nad $\mathbb{F}_2$ pomocí koeficientů v MSbF řazení pod sebe, dostaneme

$$\begin{array}{l} 1\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0 \\ 1\,0\,1\,1\,1 \\ \quad 1\,0\,1\,1\,1 \\ \quad\quad 1\,0\,1\,1\,1 \\ \quad\quad\quad 1\,0\,1\,1\,1 \\ \quad\quad\quad\quad 1\,0\,1\,1\,1 \\ \quad\quad\quad\quad\quad 1\,0\,1\,1\,1 \\ \hline \quad\quad\quad\quad\quad 0\,0\,0\,0\,1 \end{array}$$

To můžeme rozdělit do bloků, jejichž velikost je rovna stupni dělícího polynomu $x^4 + x^2 + x + 1$. V našem konkrétním případě je velikost bloku 4. Pokud by stupeň

děleného polynomu nebyl dělitelný stupněm dělícího polynomu, doplníme zleva potřebný počet nul. Rozdělením na bloky dostáváme následující části

1 1 0 0	0 0 1 0	0 0 0 0
1 0 1 1	1	
1 0 1	1 1	
1 0	1 1 1	
	1 0 1 1	1
	1 0	1 1 1
	1	0 1 1 1
	0	0 0 0 1

Nejprve se snažíme vynulovat první blok zleva přičtením vhodné lineární kombinace řádků z daného bloku, které obsahují části bitově posunutého dělícího polynomu. Hledáme tak koeficienty lineární kombinace $a_1, \ldots, a_4 \in \mathbb{F}_2$ splňující

$$1100 = a_1 \cdot 1011 + a_2 \cdot 0101 + a_3 \cdot 0010 + a_4 \cdot 0001.$$

To jsou ve skutečnosti souřadnice slova 1100 vzhledem k bázi prefixů slova 10111 dimenze 4, což je v tomto případě množina

$$\{1011,\ 0101,\ 0010,\ 0001\}.$$

Vynulování prvního bloku způsobí změnu druhého bloku. Tato změna odpovídá lineární kombinací vektorů báze sufixů slova 10111 dimenze 4 se stejnými koeficienty $a_1, \ldots, a_4$. Báze sufixů je v tomto konkrétním případě

$$\{1000,\ 1100,\ 1110,\ 0111\}.$$

Takovou změnu budeme nadále nazývat přenos.

Definice 3.2.5 (Přenos)**.** *Nechť* $\boldsymbol{u} \in \mathbb{F}_2^m$ *a* $\boldsymbol{g} \in \mathbb{F}_2^{m+1}$ *jsou slova a*

$$a_0, \ldots, a_{m-1} \in \mathbb{F}_2^n \tag{3.3}$$

jsou souřadnice slova $\boldsymbol{u}$ *vzhledem k bázi prefixů slova* $\boldsymbol{g}$ *dimenze* m*. Slovo délky* m*, jehož souřadnice vzhledem k bázi sufixů slova* $\boldsymbol{g}$ *dimenze* m *jsou jako v (3.3), nazveme* ***přenosem*** *(slova)* $\boldsymbol{u}$ *určeným (slovem)* $\boldsymbol{g}$*.*

Přenos přičteme k následujícímu bloku a celý proces opakujeme, dokud nevynulujeme předposlední blok. Protože je velikost bloku rovna stupni dělícího polynomu $g(x)$, zbytek po dělení nalezneme v posledním bloku.

Následující tvrzení popisuje, jak vypadají všechny přenosy určené slovem $\boldsymbol{g}$. Ptáme se, jakou hodnotu má přenos $d_{m-1} \ldots d_0$ slova (bloku) $c_{m-1} \ldots c_0$ určený slovem $\boldsymbol{g}$ délky $m + 1$.

Tvrzení 3.2.6. *Nechť* $m \in \mathbb{N}$ *je délka bloku a* $g(x) = \sum_{i=0}^{m} g_i x^i \in \mathbb{F}_2[x]$*. Dále mějme* $c_0, \ldots, c_{m-1}, d_0, \ldots, d_{m-1} \in \mathbb{F}_2$ *takové, že*

$$\left(\sum_{i=0}^{m-1} c_i x^i \right) \cdot x^m \ mod\ g(x) = \sum_{i=0}^{m-1} d_i x^i.$$

Hodnoty $d_0, \ldots, d_{m-1}$ *jsou určeny lineárním zobrazením* $f : \mathbb{F}_2^m \to \mathbb{F}_2^m$ *dané vztahem*

$$f\begin{pmatrix} c_{m-1} \\ c_{m-2} \\ \vdots \\ c_0 \end{pmatrix} = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_{m-1} \\ 0 & g_0 & g_1 & \ldots & g_{m-2} \\ \vdots & & & & \vdots \\ 0 & 0 & \ldots & 0 & g_0 \end{pmatrix} \begin{pmatrix} g_m & 0 & 0 & \ldots & 0 \\ g_{m-1} & g_m & 0 & \ldots & 0 \\ \vdots & \vdots & & & \\ g_1 & g_2 & \ldots & g_{m-1} & g_m \end{pmatrix}^{-1} \begin{pmatrix} c_{m-1} \\ c_{m-2} \\ \vdots \\ c_0 \end{pmatrix}$$

Důkaz. Lineární zobrazení f zapišme ve tvaru

$$f(x) = \boldsymbol{A}\boldsymbol{B}^{-1}x.$$

Sloupce matice $\boldsymbol{A}$ obsahují bázi A sufixů slova $\boldsymbol{g}$ dimenze m a sloupce matice $\boldsymbol{B}$ obsahují bázi B prefixů slova $\boldsymbol{g}$ dimenze m. Matice $\boldsymbol{B}$ je matice přechodu od báze B ke kanonické bázi, proto je $\boldsymbol{B^{-1}}$ matice přechodu od kanonické báze k bázi B. To znamená, že pro $\boldsymbol{u} \in \mathbb{F}_2^n$ jsou

$$\boldsymbol{B^{-1}u}$$

souřadnice slova $\boldsymbol{u}$ vzhledem k bázi B. Podle definice má přenos stejné souřadnice vzhledem k bázi A. Výsledek proto získáme vynásobením těchto souřadnic maticí $\boldsymbol{A}$ zleva.

□

Pozorování. *Pro určení matice lineárního zobrazení* f *z tvrzení 3.2.6 stačí určit obraz na vektorech kanonické báze. To dostaneme z výpočtu*

$$\begin{aligned} x^{m-1} \cdot x^m \ mod\ g(x) &= \sum_{i=0}^{m-1} d_{i,0}x^i \\ x^{m-2} \cdot x^m \ mod\ g(x) &= \sum_{i=0}^{m-1} d_{i,1}x^i \\ &\vdots \\ 1 \cdot x^n \ mod\ g(x) &= \sum_{i=0}^{m-1} d_{i,m-1}x^i. \end{aligned}$$

Zobrazení f *je potom ve tvaru*

$$f\begin{pmatrix} c_{m-1} \\ \vdots \\ c_0 \end{pmatrix} = \begin{pmatrix} d_{0,0} & \ldots & d_{0,m-1} \\ \vdots & & \vdots \\ d_{m-1,0} & \ldots & d_{m-1,m-1} \end{pmatrix} \begin{pmatrix} c_{m-1} \\ \vdots \\ c_0 \end{pmatrix}$$

Nyní zbývá předpočítat tabulku všech přenosů určených slovem $\boldsymbol{g}$, kde $g(x)$ je generující polynom CRC kódu stupně m. Spočítáme

$$x^i \cdot x^m \bmod g(x,) \ \ i = 0, \ldots, m-1$$

a z toho získáme předpis lineárního zobrazení f. Jednou možností je uložit jen předpis zobrazení a při výpočtu si hledanou hodnotu vždy dopočítat. Druhou možností je uložit množinu všech hodnot přenosů. Tuto množinu lze spočítat ze zobrazení f pro všechny vstupní vektory, což odpovídá výpočtu všech hodnot

$$\sum_{i=0}^{m-1} a_i x^i \cdot x^n \bmod g(x), \ \ a_i \in \mathbb{F}_2.$$

Pozorování. *Všechny polynomy nad $\mathbb{F}_2$ stupně menšího než $m \in \mathbb{N}$ odpovídají všem slovům délky m. Na taková slova se můžeme dívat jako na množinu všech m-bitových čísel, neboli na hodnoty*

$$0 \text{ až } 2^m - 1.$$

Tabulku všech přenosů můžeme tedy indexovat pomocí přirozených čísel, kde index značí vstupní hodnotu bloku.

Příklad. Mějmě polynom

$$g(x) = x^4 + x^2 + x + 1 \overset{m}{\sim} 10111.$$

Spočítáme tabulku přenosů určených slovem $\boldsymbol{g}$ pomocí příslušného lineárního zobrazení. Nejprve spočítáme

$$x^i \cdot x^4 \bmod g(x), \;\; i = 0, \ldots, 3.$$

Dělení zapsané pod sebe pomocí koeficientů je ve tvaru

$$\begin{array}{rl} 0001 & 0000 \\ 1 & 0111 \\ \hline & 0111 \end{array} \qquad \begin{array}{rl} 0010 & 0000 \\ 10 & 111 \\ \hline & 1110 \end{array} \qquad \begin{array}{rl} 0100 & 0000 \\ 101 & 11 \\ 1 & 0111 \\ \hline & 1011 \end{array} \qquad \begin{array}{rl} 1000 & 0000 \\ 1011 & 1 \\ 10 & 111 \\ 1 & 0111 \\ \hline & 0001 \end{array}$$

Lineární zobrazení $f : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ reprezentující přenosy slov určených slovem $\boldsymbol{g}$ je ve tvaru

$$f(x) = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} x$$

Pomocí něho už lze jednoduše dopočítat ostatní hodnoty.

$$\begin{aligned} &0000 \to 0000,\ 0001 \to 0111,\ 0010 \to 1110,\ 0011 \to 1001, \\ &0100 \to 1011,\ 0101 \to 1100,\ 0110 \to 0101,\ 0111 \to 0010, \\ &1000 \to 0001,\ 1001 \to 0110,\ 1010 \to 1111,\ 1011 \to 1000, \\ &1100 \to 1010,\ 1101 \to 1101,\ 1110 \to 0100,\ 1111 \to 0011. \end{aligned} \tag{3.4}$$

□

Když máme k dispozici tabulku všech přenosů, výpočet zbytku po dělení polynomem $g(x)$ je v takovém případě výrazně rychlejší. Ve verzi bez tabulky se pro každý nenulový bit počítá jeden součet, zatímco ve verzi s tabulkou se počítá jeden součet pro každý blok.

Příklad. Spočítáme

$$(x^{13} + x^{12} + x^{11} + x^{10} + x^5) \bmod (x^4 + x^2 + x + 1)$$

nad $\mathbb{F}_2$ pomocí tabulkové verze dělení.

Uvažujeme blok velikosti 4 a k výpočtu použijeme hodnoty z (3.4). Slovo odpovídající prvnímu z polynomů je v MSbF řazení ve tvaru 11110000100000, jeho délka je 14. Slovo doplníme zleva dvěma nulami, aby jeho délka byla dělitelná čtyřmi. Začínáme tak se slovem

$$0011|1100|0010|0000.$$

Vezmeme první blok 0011 a najdeme k němu odpovídající přenos v tabulce, kterým je 1001. Tento přenos přičteme k druhému bloku a proces opakujeme. Výpočet vypadá následovně

$$\begin{aligned} 0011 &\to 1001, \\ 1100 + 1001 = 0101 &\to 1100, \\ 0010 + 1100 = 1110 &\to 0100, \\ 0000 + 0100 = 0100. & \end{aligned}$$

Dostáváme tak, že

$$(x^{13} + x^{12} + x^{11} + x^{10} + x^5) \bmod (x^4 + x^2 + x + 1) = x^2 \overset{m}{\sim} 0100.$$

□

Technická poznámka 3.2.7. *V případě 32-bitových procesorů je ideální velikost bloku 8, 16 nebo 32. U 64-bitového procesoru lze navíc uvažovat velikost i 64. Většina dnešních procesorů má hardwarovou podporu pro výpočet se slovy těchto délek. Jedná se proto o jedny z nejrychlejších procesorových operací.*

Příklad. Pro zajímavost spočítáme přibližnou paměťovou náročnost výpočtu zbytku po dělení polynomů s blokem délky 32 při uložení celé tabulky přenosů. Tabulka obsahuje 2^{32} hodnot, kde každá z nich je 32-bitové bezznaménkové číslo. Na to je potřeba paměť o velikosti

$$2^{32} \cdot 32 \text{ b} = 16 \text{ GB}.$$

Výpočet je v takovém případě extrémně rychlý. Pokud má vstupní polynom stupeň 8192 (ekvivalentně velikost zprávy je 1 kB), provede se pouze 31 sčítání 32-bitových čísel, což je na 32-bitovém procesoru pouze 31 operací. Každopádně v průběhu výpočtu je potřeba číst data z paměti, které bude v tomto případě nejpomalejší část výpočtu. Každopádně i tak lze dosáhnout rychlosti řádově jednotek až desítek GB/s (v závislosti na rychlosti RAM paměti).

□

Z předchozího příkladu vyplývá, že realizace výpočtu pomocí celé tabulky a bloků délky 32 nebude na malých součástkách možná. Nabízí se otázka, zda by si nestačilo pamatovat pouze matici lineárního zobrazení přenosů. V takové situaci by bylo potřeba do paměti uložit pouze 32 32-bitových hodnot (1 kB). Při zpracování bloku by se nejprve musel daný přenos spočítat. To by znamenalo sečíst až 32 sloupců matice, což je sice 32 krát více operací než v předchozím případě, ale stále je vše realizováno pomocí standardních 32-bitových celočíselných proměnných. To je velká výhoda oproti obyčejné metodě bez tabulky.

Příklad. Na závěr srovnáme všechny čtyři uvedené způsoby výpočtu v celé této kapitole. Předpokládejme, že počítáme $f(x) \bmod g(x)$, kde

$$\deg(g(x)) = 32,\ \deg(f(x)) = 1024,\ f(x) = x^{32} \cdot f'(x).$$

U všech metod uděláme horní odhad počtu sčítání a spočítáme paměťovou náročnost. Pro každé sčítání bude navíc potřeba provést minimálně nějaké čtení z paměti, ale to je pro všechny metody podobné, takže ho nebudeme uvažovat.

Začneme se standardním dělením se zbytkem bez tabulky. V ideálním světě by bylo potřeba provést $992 \cdot 2$ sčítání 32-bitových čísel, protože polynom $g(x)$ zasahuje vždy do dvou bloků velikosti 32. V reálném světě bude potřeba použít speciální aritmetiku dlouhých čísel, která nebude implementována dokonale pro tento typ úlohy. Skutečné množství operací bude mnohonásobně vyšší.

Další uvedenou metodou je výpočet pomocí generující matice ve tvaru $(\boldsymbol{I_k}|\boldsymbol{A})$. Paměťová náročnost odpovídá velikosti matice $\boldsymbol{A}$, což je v tomto případě

$$992 \cdot 32 \text{ b } \doteq 4 \text{ kB}.$$

Výpočetní náročnost je 992 sčítání 32-bitových čísel.

Ve variantě výpočtu se zobrazením f a kompletní tabulkou všech přenosů bude potřeba provést pouze 31 sčítání, zato paměťová náročnost je extrémních 16 GB.

V posledním případě uvažujme výpočet se zobrazením f, ale pouze s uloženou maticí. Paměťová náročnost je

$$32 \cdot 32 \text{ b } \doteq 128 \text{ B}$$

a je potřeba provést $31 \cdot 32 = 992$ sčítání pro zisk hodnot přenosů a 31 sčítání pro samotný výpočet.

□

4. CAN

CAN (Controller Area Network) je sériový komunikační protokol vyvinutý v roce 1986 firmou Robert Bosch GmbH. Je to jeden z nejúspěšnějších síťových protokolů vůbec. V dnešní době obsahuje téměř každé auto vyrobené v Evropě alespoň jednu elektronickou součástku, která využívá tento protokol. CAN se nepoužívá pouze v automobilech, ale i v ostatních typech dopravních prostředků. Kromě dopravy se s ním můžeme setkat například i v průmyslové oblasti, kde se využívá k automatizaci výrobních procesů.

V roce 1991 byla představena verze CAN 2.0, která se dělí na verzi A a B. Zanedlouho vydala Mezinárodní organizace pro normalizaci CAN standard ISO 11898. V roce 2011 začal vývoj další verze CAN protokolu s názvem CAN FD, který byl představen o rok později. Bohužel při ISO certifikaci byla objevena závažná slabina v detekci chyb při přenosu. Z toho důvodu bylo potřeba protokol opravit, což se nakonec podařilo. Právě tato opravená verze je v dnešní době nejvíce rozšířená. Díky velké popularitě se od roku 2018 pracuje na vývoji třetí verze protokolu s označením CAN XL.

Jak už bylo řečeno, s CAN se setkáváme hlavně v automobilech. V dnešní době obsahuje automobil velké množství elektronických součástek, které mezi sebou komunikují právě pomocí CAN protokolu. Například se používá pro ovládání airbagů, zamykání dveří, ABS, posilovače řízení, start/stop systému, elektronické parkovací brzdy, parkovacích asistentů, protikolizního systému, kontrolu udržování pruhu na silnici a mnoho dalšího.

Některé systémy jsou pro bezpečnost posádky automobilu zásadní. Proto je důležité detekovat chyby při přenosu. Na to se při vývoji protokolu myslelo. Jednou z několika typů kontrol je kontrola pomocí CRC kódů, na kterou se v této kapitole zaměříme.

První polovina kapitoly popisuje základní i rozšířenou verzi protokolů CAN a CAN FD. Oba protokoly v původním návrhu obsahují zranitelnosti v detekci chyb, které jsou v textu podrobně popsány a vysvětleny. Samotný technický popis protokolů vychází hlavně z [9], [10], [13] a [15].

Veškerá matematická část okolo protokolů je originálním přínosem této práce. Důležitou částí celé kapitoly je vybudování matematického aparátu a samotná charakterizace všech zpráv, které uniknou CRC kontrole v CAN FD protokolu. Závěrem celé práce je analýza používaných CRC polynomů v CAN a CAN FD využívající teorii z prvních tří kapitol. Analýza podobných CRC kódů se v praxi provádí hrubou silou.

Poznámka 4.0.1. *Všechna slova (zprávy) uvažujeme v MSbF řazení a v celé kapitole pracujeme nad* $\mathbb{F}_2$.

Jak je popsáno dále, CAN protokol v libovolné variantě a verzi používá proměnnou délku zpráv, zatímco blokové (a speciálně CRC) kódy používají pevně danou délku. Protože výpočet kontrolních bitů nezávisí na délce slova, ale na hodnotě příslušného polynomu, můžeme všechny zprávy kratší délky pomyslně prodloužit zleva nulami na stejnou délku.

Pozorování. *Přidání nul zleva ke slovu v MSbF řazení neovlivní hodnotu příslušného polynomu.*

Poznámka 4.0.2. *V celé kapitole budeme používat následující úvahu. Mějme $[n, k]$-kód $\mathscr{C}$ a slovo (zprávu) $\boldsymbol{u}$ délky $l < k$. Kdykoliv řekneme, že počítáme kódové slovo (kontrolní bity) z kódu $\mathscr{C}$ ke slovu $\boldsymbol{u}$, doplňujeme pomyslně před výpočtem zleva nulami $\boldsymbol{u}$ na délku k. V případě polynomů se nic nemění a zprávou jsou všechny polynomy stupně menšího než k.*

V celé kapitole si tak lze spojení CRC kódů a CAN protokolu představovat tak, že odesíláme zprávy pevně dané délky k, ale před přenosem odřízneme nulové bity zleva. To hodnotu příslušného polynomu neovlivní, a tak ani hodnotu kontrolních bitů.

4.1 Protokol CAN

V této kapitole nejprve popíšeme protokol CAN 2.0 ve verzi A a B, poté se podíváme na metodu bit stuffingu a na konec popíšeme konkrétní postup výpočtu CRC.

4.1.1 Popis protokolu

Popisem protokolu budeme rozumět pouze popis struktury zprávy, která se přenáší v rámci jedné komunikace.

Definice 4.1.1 (Rámec)**.** *Jednotku datového přenosu protokolu nazýváme* ***rámec****.*

Technická poznámka 4.1.2. *Rámec kromě samotných informačních dat obsahuje také data důležité pro samotnou komunikaci - identifikace, kontrolní bity a další bity definované protokolem.*

Specifikace protokolu CAN definuje čtyři různé typy rámců - datová zpráva, žádost o data, zpráva o chybě a zpráva o přetížení. Pokud chce jedno zařízení informace od druhého, pošle žádost o data. Zařízení na druhé straně tuto žádost vyslyší a odešle datovou zprávu. Zpráva o chybě slouží k signalizaci, že došlo při přenosu k chybě. Zpráva o přetížení říká, že jedna ze stran nestíhá a přenos se pozdrží.

Nejprve začneme s popisem rámce datové zprávy ve standardním formátu (verze A). Struktura rámce je uvedena na obrázku 4.1 a popsána v tabulce 4.1.

Obrázek 4.1: Popis struktury standardního formátu datové zprávy protokolu CAN.

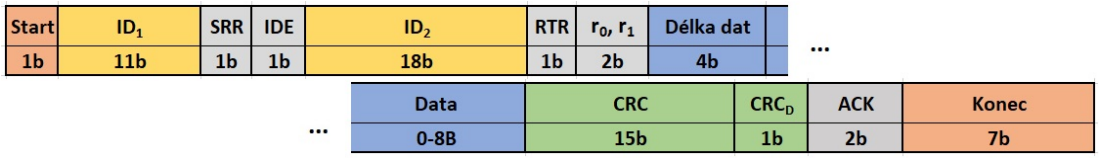
Technická poznámka 4.1.3. *ACK je informatická zkratka pro acknowledgement (potvrzovací) bit. Je to bit, který vždy odesílá přijímající strana a dává tak najevo, že danou zprávu přijala.*

Název	Popis	Hodnota
Start	Značí začátek zprávy	0
ID	Identifikátor, který určuje cíl a prioritu zprávy	
RTR	Rozlišuje datovou zprávu (0) a žádost o data (1)	0/1
IDE	Rozlišuje standardní (0) a rozšířený formát (1)	0
r_0	Rezervovaný bit bez účelu	0
Délka dat	Binárně zapsaný počet bajtů dat	
Data	Samotná přenášená data	
CRC	15 kontrolních bitů - detaily viz kapitola 4.1.2	
CRC_D	Oddělovač kontrolních bitů	0
ACK	Slouží k potvrzení přijmutí zprávy	0/1
Konec	Slouží k zakončení zprávy	1111111

Tabulka 4.1: Popis částí standardního formátu datové zprávy protokolu CAN.

CAN B na rozdíl od verze A používá standardní i rozšířenou verzi rámce datové zprávy. Rozdíl mezi standardním a rozšířeným formátem není v možnosti odesílat větší množství datových bajtů v jednom rámci, ale ve schopnosti odeslat delší identifikátor. Ve standardní verzi se používá 11 bitový identifikátor, zatímco v rozšířené 29 bitový.

Velká část rozšířeného formátu je shodná se standardním formátem a je uvedena na obrázku 4.2. V tabulce 4.2 jsou popsány pouze části, které se liší od standardní verze.



Obrázek 4.2: Popis struktury rozšířeného formátu datové zprávy protokolu CAN.

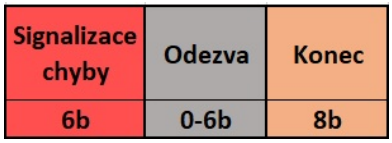
Pozorování. *Rámec typu žádost o data má stejnou strukturu jako datová zpráva s rozdílem, že v žádosti se odešle RTR bit s hodnotou jedna a datová část se ponechá prázdná.*

Definice 4.1.4 (Datová a kontrolní část)**.** *Část od začátku rámce datové zprávy (nebo žádosti o data) po konec bloku dat budeme nazývat* ***datová část*** *a část od začátku CRC bloku až po konec rámce budeme nazývat* ***kontrolní část****.*

Ještě se stručně zmíníme o struktuře zprávy o chybě, neboť ta částečně souvisí s bit stuffing procesem (viz kapitola 4.1.2). Ta je uvedena na obrázku 4.3 a popsána v tabulce 4.3.

Název	Popis	Hodnota
ID_1	Prvních 11 bitů identifikátoru	
SRR	Označuje, že se jedná o rozšířený formát	1
IDE	Rozlišuje standardní (0) a rozšířený formát (1)	1
ID_2	Zbývajících 18 bitů identifikátoru	
r_0, r_1	Rezervované bity bez účelu	00

Tabulka 4.2: Popis částí rozšířeného formátu datové zprávy protokolu CAN, které se liší od standardního formátu.



Obrázek 4.3: Popis struktury zprávy o chybě protokolu CAN.

Název	Popis	Hodnota
Signalizace chyby	Šest po sobě jdoucích nulových bitů	000000
Odezva	Až šest po sobě jdoucích nulových bitů	0...0
Konec	Simuluje poslední blok datové zprávy	01111111

Tabulka 4.3: Popis částí zprávy o chybě protokolu CAN.

4.1.2 Bit stuffing a výpočet CRC

Technická poznámka 4.1.5. *Technicky se přenos bitů provádí změnou napětí na vodiči, které se musí na dané hodnotě udržovat pevně určenou dobu. Každá ze stran musí vědět, kdy je ten správný okamžik k měření. Určení tohoto okamžiku se na CAN sběrnici realizuje pomocí hodin jak na straně odesílatele, tak na straně příjemce. Na začátku přenosu je ale potřeba hodiny synchronizovat. Bohužel svět není dokonalý a hodiny se postupem času rozcházejí. K tomu je potřeba provádět synchronizaci i v průběhu přenosu.*

Technická poznámka 4.1.6. *Clock recovery je proces synchronizace hodin pomocí přenášených bitů, který se mimo jiné používá v CAN (i CAN FD) protokolu. Například příjemce synchronizuje hodiny vždy v momentě, kdy se mění napětí odpovídající bitu 0 na napětí odpovídající bitu 1 (nebo naopak). Clock recovery má jednu zásadní nevýhodu. Nefunguje, pokud se delší dobu hodnoty přenášených bitů nemění. To se řeší v CANu pomocí bit stuffing metody.*

Bit stuffing je metoda vkládání bitů do odesílané zprávy tak, aby neobsahovala dlouhé řetězce bitů stejné hodnoty.

Definice 4.1.7 (Stuff bit)**.** *Nechť $a_1, \ldots, a_n \in \mathbb{F}_2$ je posloupnost bitů. Bit $b \in \mathbb{F}_2$ vložený za bit a_i, kde $1 \leq i \leq n$, s hodnotou $1 - a_i$ nazveme* ***stuff bitem****.*

Definice 4.1.8 (Bit stuffing)**.** *Proces vkládání stuff bitů do posloupnosti bitů nazýváme* ***bit stuffing****.*

Technická poznámka 4.1.9. *V CANu se používá bit stuffing na datovou část rámce a na CRC blok. Stuff bit se vkládá vždy po pěti po sobě jdoucích bitech stejné hodnoty (včetně stuff bitu).*

Pozorování. *Rámec chybové zprávy začíná šesti nulovými bity mimo jiné proto, že v běžném přenosu nemůže v CANu k takové situaci díky bit stuffingu dojít.*

V následujícím příkladu je ukázka bit stuffingu, který se používá v CANu (viz poznámka 4.1.9).

Příklad. Mějme následující posloupnosti bitů

$$0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,1\,1\,1\,1,$$
$$1\,1\,1\,1\,1\,0\,0\,0\,0\,0,$$

na které provedeme bit stuffing popsaný výše. Za každých pěti po sobě shodných bitů se přidá stuff bit. V druhé posloupnosti je potřeba započítat do pěti shodných bitů i nově přidaný stuff bit. Posloupnosti po bit stuffingu vypadají následovně

$$0\,1\,1\,1\,1\,1\,\mathbf{0}\,1\,0\,0\,0\,0\,0\,\mathbf{1}\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,1\,1\,\mathbf{0}\,1\,1,$$
$$1\,1\,1\,1\,1\,\mathbf{0}\,0\,0\,0\,0\,\mathbf{1}\,0,$$

kde tučně zvýrazněné hodnoty jsou stuff bity.

□

Zbývá popsat, jak probíhá výpočet CRC. V CANu se pro výpočet kontrolních bitů používá polynom

$$g_{15} = x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1.$$

Protože je jeho stupeň 15, počítá se 15 kontrolních bitů.

Poznámka 4.1.10. *Jak už bylo řečeno v úvodu kapitoly, přestože je délka kódových slov v CANu proměnná, můžeme je pomyslně zleva doplnit nulami na pevně danou délku, a tak získat množinu slov jednoho konkrétního CRC kódu.*

Pozorování. *Počet bitů datové části je maximálně* $39 + 64 = 103$.

Výpočet kontrolních bitů probíhá následovně. Vezmou se všechny bity datové části, které se uvažují v MSbF řazení, a z nich se spočítá kódové slovo CRC kódu s generujícím polynomem g_{15}. Délka kódu není ve specifikaci definována, ale můžeme brát maximální délku, při které má daný kód ještě vhodné vlastnosti (více viz kapitola 4.3). Tím se získají kontrolní bity, které se uloží do CRC bloku rámce. Poté se provede bit stuffing a výsledná zpráva se odešle. Příjemce z přijaté zprávy odebere stuff bity a opět spočítá kontrolní CRC bity. Pokud neodpovídají kontrolním bitům z CRC bloku, zahlásí chybu.

Pozorování. *Ověření, zda je obdržená zpráva kódové slovo, je ekvivalentní opětovnému výpočtu kontrolních bitů, jako se děje v CANu. Pokud přijatá zpráva není kódové slovo, nemůžeme opětovným výpočtem dostat stejný výsledek, neboť tento výpočet dává dohromady se zprávou kódové slovo. Pokud přijatá zpráva je kódové slovo, opětovný výpočet nám vrátí stejný výsledek, protože každé zprávě odpovídá právě jedno kódové slovo.*

4.1.3 Zranitelnost protokolu

Přestože má protokol několik bezpečnostních prvků, může se stát, že pouhé dvě chyby při přenosu mohou zprávu poškodit tak nešťastně, že nebude chyba protokolem odhalena. Odhaduje se, že podmíněná pravděpodobnost

$$P(\text{chyba nebude odhalena} \mid \text{při přenosu dojde ke dvě chybám})$$

je řádově 10^{-7}. To se může zdát jako zanedbatelné riziko, ale ve spojení s velkým množstvím automobilů, které CAN stále ještě využívá, to může být nebezpečný problém.

CAN využívá CRC kód s Hammingovou vzdáleností 6 (viz kapitola 4.3), proto je schopen odhalit až 5 chyb v rámci jedné zprávy. V CANu se zabezpečuje pomocí CRC samotná zpráva, která se ještě před odesláním doplní o stuff bity. To je bohužel nešťastná kombinace, protože chyby spolu s přidáváním a odebíráním stuff bitů dokáží vnést do původní zprávy velké množství chyb a to je problém, neboť CRC má jen omezené detekční schopnosti. Konkrétní chybovou situaci popíšeme na následujícím příkladu.

Příklad. Předpokládejme, že máme zprávu ve tvaru

$$1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1,$$

ke které byly již přidány kontrolní bity. Před odesláním se provede bit stuffing, který zprávu upraví do tvaru

$$1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1.$$

Ta se odešle a při přenosu dojde ke dvěma chybám vyznačených tučně

$$1\ 1\ 1\ \mathbf{0}\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ \mathbf{0}\ 1\ 1.$$

Ty jsou ale na takových místech, že z pohledu příjemce ovlivnily pozici stuff bitu. Po jeho odebrání příjemce dostává zprávu

$$1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1.$$

Jak můžeme vidět níže, zpráva před a po se liší na 7 pozicích, což překonává detekční schopnosti CRC kódu.

$$\begin{array}{c} 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1 \\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1 \\ \hline 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0 \end{array}$$

Při přenosu nastaly přitom pouze dvě chyby.

□

Problematická situace nastává ve chvíli, kdy chyby narušují části zprávy, které ovlivňují přítomnost stuff bitů. Protože stuff bity zasahují do délky zprávy, tak chyba, která způsobí přidání nebo odebrání stuff bitu, posune o bit celou část zprávy doleva nebo doprava. To způsobí, že se původní a přijatá zpráva liší na mnoha pozicích.

Konkrétně musí zároveň dojít ke dvěma typům chyb. Jedna, která zruší úsek pěti po sobě jdoucích shodných bitů a druhá, která naopak takovou posloupnost vytvoří. První z nich výslednou zprávu prodlouží a druhá z nich zprávu zkrátí. Je důležité, aby se oba typy chyb vyskytly ve stejném množství. V opačném případě by přijatá zpráva neměla původní délku a to by protokol vyhodnotil jako chybu. Ačkoliv i tak by taková zpráva mohla CRC kontrole uniknout.

Pozorování. *Nastane-li při přenosu libovolná chyba v části zprávy se stuff bity tvaru*

$$11111,\ 00000,$$

dojde ke ztrátě stuff bitu. Nastane-li taková chyba, že se z části zprávy se stuff bity tvaru

$$11110,\ 11101,\ 11011,\ 10111,\ 01111,\ 10000,\ 01000,\ 00100,\ 00010,\ 00001$$

stane pět shodných bitů, dojde k přidání stuff bitu. Pokud v obou skupinách dojde ke stejnému počtu chyb, bude mít výsledná zpráva stejnou délku jako na začátku a pozice stuff bitů se změní.

Výše popsanou chybu se pokusili opravit v protokolu CAN FD. Rozhodli se, že kontrolní bity nebudou přidávat ke zprávě bez stuff bitů, ale až ke zprávě se stuff bity. V takovém případě se kontroluje přesně to, co se přenáší, a tak je malý počet chyb při přenosu odhalen. To sice tuto konkrétní chybu vyřešilo, ale ani v takovém případě nebyl nový návrh bezchybný. Více o této chybě v kapitole 4.2.3.

4.2 Protokol CAN FD

Původní CAN protokol obsahuje maximálně 8 datových bajtů a maximální rychlost přenosu v dnešní době dosahuje 1 Mb/s. To začalo být v automobilovém průmyslu nedostačující. Bylo proto potřeba původní CAN protokol vylepšit.

CAN FD (CAN with Flexible Data-Rate) je protokol založený na původním CANu s několika vylepšeními. Zvýšit rychlost přenosu lze na úrovni protokolu bez potřeby změny hardwaru. Toho dosáhneme tak, že zvýšíme poměr užitečných dat vůči celé odesílané zprávě. CAN FD proto kromě základní délky dat 0-8 B nabízí další varianty délek a to 12, 16, 20, 24, 32, 48 a 64 B.

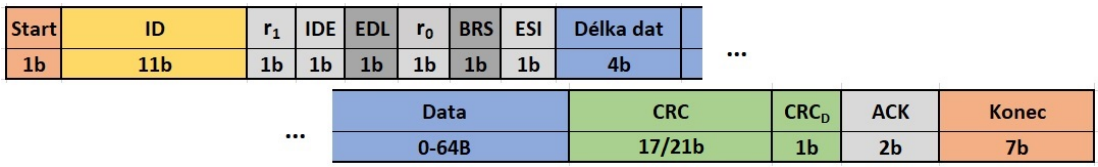
Dalšího zrychlení se dosáhlo přidáním možnosti odesílat data vyšší rychlostí, které se týká pouze samotných dat a CRC bitů. Ostatní části se odesílají vždy původní rychlostí. Díky výše uvedeným vylepšením se rychlost několikanásobně zvýšila.

4.2.1 Popis původního protokolu

V této kapitole je popsán protokol CAN FD z roku 2012, u kterého se při certifikaci objevila zásadní zranitelnost. Jeho upravená verze je rozebrána v kapitole 4.2.5.

Stejně jako CAN ve verzi B má i CAN FD dva formáty zpráv - standardní a rozšířený. Struktura zpráv je velice podobná struktuře v CANu. Standardní

formát datové zprávy je zobrazen na obrázku 4.4 a popisy nových polí, které se liší od CANu z kapitoly 4.1.1, jsou uvedeny v tabulce 4.4.



Obrázek 4.4: Popis struktury standardního formátu datové zprávy protokolu CAN FD.

Název	**Popis**	**Hodnota**
EDL	Odlišuje CAN formát (0) od CAN FD formátu (1)	1
BRS	Rozlišuje standardní (0) a vyšší rychlost (1) přenosu	0/1
ESI	Signalizuje, zda došlo k chybě (1), nebo nedošlo (0)	0/1

Tabulka 4.4: Popis částí standardního formátu datové zprávy protokolu CAN FD, které se liší od původního CANu.

Technická poznámka 4.2.1. *Po jedné komunikační lince je možné zároveň posílat CAN i CAN FD formát zpráv. Ty se odlišují právě pomocí EDL bitu. Příjemce, který podporuje CAN FD, tak dokáže tyto dva typy odlišit.*

Rozšířený formát zprávy, který je zobrazen na obrázku 4.5, je pouze kombinací rozšířeného formátu CANu a standardního formátu CAN FD.

Obrázek 4.5: Popis struktury rozšířeného formátu datové zprávy protokolu CAN FD.

Technická poznámka 4.2.2. *Hodnota ukládaná do pole délka dat se v CAN FD určuje pro délky větší než 8 B pomocí následující tabulky.*

Počet bajtů	*12*	*16*	*20*	*24*	*32*	*48*	*64*
Hodnota	*9*	*10*	*11*	*12*	*13*	*14*	*15*

Poznámka 4.2.3. *Pojmy datová a kontrolní část splývají s pojmy z kapitoly 4.1.1 o popisu protokolu CAN.*

4.2.2 Bit stuffing a výpočet CRC

Bit stuffing se provádí v CAN FD jiným způsobem než v CANu. Důvodem je, že původní návrh snižoval schopnost detekce chyb pomocí CRC (viz kapitola 4.1.3).

Nejprve se do datové části vloží stuff bit po každé pětici po sobě jdoucích bitů stejné hodnoty (včetně již přidaných stuff bitů). Z toho se poté spočítají kontrolní CRC bity, které se vloží do CRC bloku. Stuff bity se přidávají i do kontrolních bitů, ale jiným postupem. Jeden stuff bit se přidá na začátek CRC bloku a poté se přidává stuff bit po každé čtveřici bitů nezávisle na jejich hodnotě. Díky tomuto postupu jsou na rozdíl od CANu zabezpečeny přenášené bity.

Definice 4.2.4 (Statický a dynamický stuff bit)**.** *Stuff bity přidávané po čtyřech bitech budeme nazývat* ***statické*** *a stuff bity přidávané po pěti po sobě jdoucích bitech stejné hodnoty budeme nazývat* ***dynamické****.*

V CAN FD se používají dva CRC polynomy, které se vybírají podle délky odesílané zprávy. Je-li délka dat nejvýše 16 bajtů, volí se polynom g_{17}, jinak se volí g_{21}, kde

$$g_{17} = x^{17} + x^{16} + x^{14} + x^{13} + x^{11} + x^6 + x^4 + x + 1,$$
$$g_{21} = x^{21} + x^{20} + x^{13} + x^{11} + x^7 + x^4 + x^3 + 1.$$

Vlastnosti těchto polynomů jsou rozebrány v kapitole 4.3.

4.2.3 Zranitelnost protokolu v původním návrhu

Úpravou bit stuffingu (viz kapitola 4.2.2) se eliminovala chyba vyskytující se v CAN protokolu (viz kapitola 4.1.3). Tato úprava nebyla správně navržena a i v tomto případě způsobuje závažnou chybu. Přestože CAN FD využívá CRC kódy s Hammingovou vzdáleností 6 (viz kapitola 4.3), může se stát, že protokol nezahlásí chybu u zprávy s jednou chybou. Situaci, při které tohle nastává, popíšeme podrobněji.

Při přenosu dochází k synchronizaci hodin na straně příjemce (viz poznámky 4.1.5 a 4.1.6). Může se stát, že v průběhu přenosu dojde ve vodiči k výkyvu napětí tak nešťastně, že se synchronizace provede dříve nebo později, než by měla a zároveň je výkyv dostatečně malý, že neovlivní přijatou hodnotu bitu. Protože se synchronizace neprovedla ve správný čas, čte příjemce napětí na vodiči příliš brzy nebo příliš pozdě. To může mít za následek, že se naměří jeden bit navíc nebo se jeden bit ztratí.

Poznámka 4.2.5. *Chybě, při které dojde ke ztrátě bitu, se říká zkrácení bitové posloupnosti a chybě, při které dojde k přidání bitu, se říká prodloužení bitové posloupnosti. Takových chyb může při přenosu jednoho rámce nastat několik.*

V obecném případě to není problém, neboť rámec zprávy má danou strukturu s pevně danými konstantními bity a délkami. Příjemce tedy přesně ví, jak má být přijatý rámec po odebrání stuff bitů dlouhý. Obdrží-li se zpráva s nevalidním formátem, je zahlášena chyba.

Problém nastává, pokud k prodloužení nebo zkrácení posloupnosti dojde v místě, které určuje (ne)přítomnost stuff bitu. V takovém případě se délka přijaté zprávy neovlivní.

Pozorování. *Dojde-li ke zkrácení v úseku zprávy* 00000 *nebo* 11111*, nebude po odebrání stuff bitů délka zprávy ovlivněna. To samé platí pro prodloužení v úseku zprávy* 0000 *hodnotou 0 nebo* 1111 *hodnotou 1.*

Příklad. Prvně předpokládejme, že odesíláme zprávu 100000. Protože obsahuje 5 po sobě jdoucích nulových bitů, dojde k bit stuffingu a odešle se posloupnost 1000001. Při přenosu dojde k opožděné synchronizaci na prvním bitu, která způsobí, že dojde ke zkrácení a příjemce přečte 100001. Tato posloupnost neobsahuje stuff bit, takže se jedná i o obdrženou zprávu.

V druhém případě předpokládejme, že odesíláme zprávu 100001. Tato zpráva nepotřebuje přidat žádný stuff bit, takže se i v tomto tvaru odešle. Při odesílání dojde tentokrát k příliš brzké synchronizaci na prvním bitu, která způsobí prodloužení a příjemce přečte 1000001. Tato posloupnost obsahuje stuff bit, který se odebere. Výsledná zpráva je ve tvaru 100000.

	Zkrácení	Prodloužení
Původní zpráva	100000	100001
Odesílání	1000001	100001
Příjem	100001	1000001
Výsledek	100001	100000

V obou případech se přijatá zpráva liší od odeslané na jednom bitu.

□

Nyní zbývá objasnit, proč může tato chyba protokolu uniknout. Problém je v tom, že se CRC kontrolní bity počítají z posloupnosti, která už obsahuje stuff bity. To při zkrácení nebo prodloužení posloupnosti znamená, že se výpočet porovnává na dvou posloupnostech různé délky, které jsou navzájem posunuté.

Pozorování. *Mějme zprávu* $a_0 \dots a_{m-1} \in \mathbb{F}_2^m$*. Příjemce po prodloužení obdrží posloupnost*

$$a_0 \dots a_i \ a \ a_{i+1} \dots a_{m-1}$$

a po zkrácení obdrží posloupnost

$$a_0 \dots a_{j-1} \ a_{j+1} \dots a_{m-1},$$

kde $i, j \in \mathbb{N}$, $a \in \mathbb{F}_2$*. V obou případech zůstala pravá část od místa chyby zachována, levá část byla o bit posunuta doleva v případě prodloužení nebo doprava v případě zkrácení.*

Tyto zprávy můžeme doplnit zleva nulami tak, aby výsledná délka odpovídala délce používaného CRC kódu (viz poznámka 4.1.10). V obou případech dochází k velké Hammingově vzdálenosti mezi odeslanou a přijatou zprávou, což často překoná detekční schopnosti používaného CRC kódu. Pokud dorazí datová část rámce spolu s CRC blokem jako kódové slovo, je zpráva přijata jako platná. A to i přesto, že po odebrání stuff bitů může být chyba jen v jednom jediném bitu.

Příklad. Předpokládejme, že odesíláme zprávu ve tvaru

1 0 1 1 1 0 0 0 0 0.

Před odesláním se provede bit stuffing, přidá se statický stuff bit a (v tomto příkladě např. 3) kontrolní bity. Přenáší se tak zpráva ve tvaru

$$1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1.$$

Při přenosu dojde ke zkrácení, což znamená, že příjemce obdrží

$$1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1.$$

Následně se odeberou kontrolní a stuff bity a přijatá zpráva je ve tvaru

$$1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1.$$

Odeslaná a přijatá zpráva se liší pouze na jedné pozici, ale z pohledu CRC kontroly došlo ke 4 chybám, jak je vidět z

$$\begin{array}{r} 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1 \\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1 \\ \hline 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \end{array}$$

Pokud to CRC kód neodhalí, bude zpráva s jednou chybu považována za bezchybnou.

□

Pozorování. *Dojde-li ke zkrácení nebo prodloužení posloupnosti v místě kontrolních bitů, projeví se zkrácení nebo prodloužení i v posloupnosti po odebrání statických stuff bitů. Taková situace bude odhalena, protože se poškodí formát rámce.*

Poznámka 4.2.6. *Nadále budeme uvažovat pouze zkrácení nebo prodloužení posloupnosti v datové části, kterým se budeme zabývat v následující kapitole.*

4.2.4 Zprávy poškozené zkrácením nebo prodloužením

V této kapitole charakterizujeme zprávy, které při zkrácení nebo prodloužení o jeden bit projdou CRC kontrolou jako validní. Zprávy nebudeme nijak omezovat a budou tak moci nabývat libovolné hodnoty. To je odlišné od datové části rámce v CAN FD, kde je pevně určena struktura a hodnota některých bitů.

Poznámka 4.2.7. *Pojmy zpráva a slovo budeme v další části textu ztotožňovat. Přesněji zpráva délky $k \in \mathbb{N}$ je pro nás nadále to samé co slovo délky k.*

Tvrzení 4.2.8. *Mějme CRC $[n, k]$-kód s generujícím polynomem $g(x) \in \mathbb{F}_2[x]$, $x \nmid g(x)$, a nechť $m(x) \in \mathbb{F}_2[x]$ je zpráva délky k. Všechny zprávy $m'(x) \in \mathbb{F}_2[x]$ délky k, které mají stejné kontrolní bity jako zpráva $m(x)$, jsou ve tvaru*

$$m'(x) = m(x) + \sum_{i=0}^{2k-n-1} a_i x^i \cdot g(x), \tag{4.1}$$

kde $a_i \in \mathbb{F}_2$ a $2k > n$. Jestliže $2k \leq n$, potom žádná taková zpráva $m'(x) \neq m(x)$ neexistuje.

Důkaz. Nejprve ukážeme, že $m'(x)$ je zpráva délky k, neboli

$$\deg(m'(x)) \leq k-1.$$

To je vidět z následující nerovnosti

$$\deg(m'(x)) \leq 2k-n-1+\deg(g(x)) = 2k-n-1+(n-k) = k-1.$$

Nyní dokážeme, že $m(x)$ a $m'(x)$ mají stejné kontrolní bity. To nastává právě tehdy, když

$$x^{n-k} \cdot m(x) \bmod g(x) = x^{n-k} \cdot m'(x) \bmod g(x)$$

a protože $x \nmid g(x)$, a tedy $\mathrm{NSD}(x^{n-k}, g(x)) = 1$, nastává toto právě tehdy, když

$$m(x) \bmod g(x) = m'(x) \bmod g(x) \Leftrightarrow g(x) \mid (m(x)+m'(x)).$$

Dostáváme tak, že $m(x)$ a $m'(x)$ mají stejné kontrolní bity právě tehdy, když

$$m'(x)+m(x) = \sum_{i=0}^{2k-n-1} a_i x^i \cdot g(x).$$

Pokud je $2k \leq n$, je suma rovná nule, což implikuje, že

$$m'(x) = m(x).$$

□

Důsledek 4.2.9. *Mějme CRC $[n, k]$-kód s generujícím polynomem $g(x) \in \mathbb{F}_2[x]$, $x \nmid g(x)$. Zprávy $m(x) \neq m'(x) \in \mathbb{F}_2[x]$ délky k mají stejné kontrolní bity právě tehdy, když $2k > n$ a soustava o k rovnicích*

$$\left(\begin{array}{ccccc|c} g_{n-k} & & & & & \\ \vdots & g_{n-k} & & & & \\ g_0 & \vdots & \ddots & & & \\ & g_0 & & g_{n-k} & & (\boldsymbol{m}+\boldsymbol{m'})^T \\ & & \ddots & \vdots & & \\ & & & g_0 & & \end{array}\right)$$

má řešení nad $\mathbb{F}_2$.

Důkaz. V rovnosti (4.1) v tvrzení 4.2.8 stačí přesunout $m(x)$ na levou stranu a vše přepsat pomocí koeficientů.

□

Pozorování. *Nechť $r < s \in \mathbb{N}$. $(a_0 \ \ldots \ a_{r-1})^T \in \mathbb{F}_2^r$ je řešením soustavy*

$$\left(\begin{array}{ccc|c} g_{n-k} & & & b_0 \\ \vdots & \ddots & & \\ g_0 & & g_{n-k} & \vdots \\ & \ddots & \vdots & \\ & & g_0 & b_m \end{array}\right)$$

právě tehdy, když $(a_0 \ \dots \ a_{r-1} \ 0 \ \dots \ 0)^T \in \mathbb{F}_2^s$ *je řešením soustavy*

$$\left(\begin{array}{cccc|c} g_{n-k} & & & & b_0 \\ \vdots & \ddots & & & \vdots \\ g_0 & & \ddots & & b_m \\ & \ddots & & g_{n-k} & 0 \\ & & \ddots & \vdots & \vdots \\ & & & g_0 & 0 \end{array}\right).$$

Definice 4.2.10 (Odříznutí nulových bitů)**.** *Mějme slovo*

$$\boldsymbol{u} = u_j \dots u_i \ 0 \dots 0, \ \ j > i, \ u_i = 1.$$

Odříznutím nulových bitů *ze slova* $\boldsymbol{u}$ *rozumíme slovo ve tvaru*

$$u_j \dots u_i.$$

Částečným odříznutím nulových bitů *ze slova* $\boldsymbol{u}$ *rozumíme slovo ve tvaru*

$$u_j \dots u_i \ 0 \dots 0,$$

jehož délka je menší než délka $\boldsymbol{u}$*.*

Důsledek 4.2.11. *Mějme CRC* $[n, k]$*-kód s generujícím polynomem* $g(x) \in \mathbb{F}_2[x]$*,* $x \nmid g(x)$*. Dále uvažujme slova* $\boldsymbol{m} \neq \boldsymbol{m'}$ *délky* k *a nechť* $\boldsymbol{r}$ *je slovo délky* l*, které vznikne (částečným) odříznutím nulových bitů slova* $\boldsymbol{m} + \boldsymbol{m'}$*. Potom zprávy* $m(x)$ *a* $m'(x)$ *mají stejné kontrolní bity právě tehdy, když* $k \geq l > n - k$ *a soustava*

$$\left(\begin{array}{ccc|c} g_{n-k} & & & \\ \vdots & \ddots & & \\ g_0 & & g_{n-k} & \boldsymbol{r} \\ & \ddots & \vdots & \\ & & g_0 & \end{array}\right)$$

má řešení nad $\mathbb{F}_2$*.*

Důkaz. Kombinace předchozího pozorování a důsledku 4.2.9.

□

Naším cílem je najít takové zprávy, které mají stejné kontrolní bity a zároveň jedna vznikla z druhé zkrácením nebo prodloužením. Pokud totiž takovou zprávu odešleme a příjemce obdrží zkrácenou nebo prodlouženou verzi, projde skrz CRC kontrolu jako validní.

Definice 4.2.12 (Vypuštění a přidání souřadnice)**.** *Nechť* $u_{j-1} \dots u_0$ *je slovo délky* $j > i \in \mathbb{N}$*.* ***Vypuštěním i-té souřadnice (zleva)*** *rozumíme slovo délky* $j - 1$ *ve tvaru*

$$u_{j-1} \dots u_{j-i+1} \ u_{j-i-1} \dots u_0.$$

Přidáním i-té souřadnice (zleva) *(s hodnotou* $b \in \mathbb{F}_2$*) rozumíme slovo délky* $j + 1$ *ve tvaru*

$$u_{j-1} \dots u_{j-i+1} \ b \ u_{j-i} \dots u_0.$$

Definice 4.2.13 (Zranitelná zpráva)**.** *Zprávu, která má před i po vypuštění (přidání) i-té souřadnice stejné kontrolní bity, nazveme* ***zranitelnou při vypuštění (přidání) i-té souřadnice****. Obecně takovou zprávu nazveme* ***zranitelnou****.*

Značení. *Nechť $d \in \mathbb{N}$. Zobrazením F budeme rozumět $F : \mathbb{F}_2^d \to \mathbb{F}_2^d$ definované vztahem*

$$F\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{pmatrix} + \begin{pmatrix} 0 \\ a_0 \\ \vdots \\ a_{d-2} \end{pmatrix}.$$

Pozorování. *Zobrazení F je lineární a lze ho popsat jako násobení maticí typu $d \times d$ ve tvaru*

$$\begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{pmatrix}.$$

Definice 4.2.14 (Rozdíl prefixů)**.** *Nechť $i < j \in \mathbb{N}$. Mějme slova*

$$\begin{aligned} \boldsymbol{u} &= u_{j-1} \cdots u_{j-i+1} u_{j-i} u_{j-i-1} \cdots u_0, \\ \boldsymbol{u'} &= \quad\; u_{j-1} \cdots u_{j-i+1} u_{j-i-1} \cdots u_0, \end{aligned}$$

taková, že slovo $\boldsymbol{u'}$ vzniklo z $\boldsymbol{u}$ vypuštěním i-té souřadnice (nebo $\boldsymbol{u}$ vzniklo z $\boldsymbol{u'}$ přidáním i-té souřadnice). ***Rozdíl prefixů*** *slov $\boldsymbol{u}$ a $\boldsymbol{u'}$ definujeme jako hodnotu*

$$(u_{j-1}\ u_{j-2}\ \cdots\ u_{j-i})^T + (0\ u_{j-1}\ \cdots\ u_{j-i+1})^T.$$

Poznámka 4.2.15. *Pro ověření, že slova z předchozí definice mají stejné kontrolní bity, lze využít důsledek 4.2.11, kde na místo $\boldsymbol{r}$ dosadíme rozdíl prefixů slov $\boldsymbol{u} + \boldsymbol{u'}$, protože zbývající bity jsou nulové.*

Pozorování. *Rozdíl prefixů slov $\boldsymbol{u}$ a $\boldsymbol{u'}$ je obraz zobrazení $F((u_{j-1}\ \cdots\ u_{j-i})^T)$.*

Když spojíme dohromady rozdíl prefixů slov s charakterizací slov, které mají stejné kontrolní bity, dostaneme charakterizaci slov, které mají před i po vypuštění (přidání) konkrétní souřadnice stejné kontrolní bity. Tím získáme množinu slov, které jsou potenciálně nebezpečné pro CAN FD protokol. Tuto množinu popisuje následující klíčová věta této kapitoly.

Věta 4.2.16. *Mějme CRC $[n, k]$-kód s generujícím polynomem $g(x) \in \mathbb{F}_2[x]$, $x \nmid g(x)$ a nechť*

$$2k > n,\ \ n - k < i \leq j < k,\ \ i, j \in \mathbb{N},$$

kde j je délka zprávy. Dále mějme jádro matice typu $i \times (2i - n + k)$

$$K = Ker\begin{pmatrix} g_{n-k} & & & 1 & & & \\ \vdots & \ddots & & 1 & 1 & & \\ g_0 & & g_{n-k} & & 1 & 1 & \\ & \ddots & \vdots & & & \ddots & \ddots \\ & & g_0 & & & & 1 & 1 \end{pmatrix}.$$

Zpráva $\boldsymbol{u} = u_{j-1} \ldots u_0$ je zranitelná

- *při vypuštění i-té souřadnice (zleva) právě tehdy, když*

$$\{(a_0 \ \dots \ a_{i-1-n+k} \ u_{j-1} \ \dots \ u_{j-i})^T \mid a_i \in \mathbb{F}_2\} \cap K \neq \emptyset.$$

- *při přidání i-té souřadnice (zleva) s hodnotou b právě tehdy, když*

$$\{(a_0 \ \dots \ a_{i-1-n+k} \ u_{j-1} \ \dots \ u_{j-i+1} \ b)^T \mid a_i \in \mathbb{F}_2\} \cap K \neq \emptyset.$$

Jestliže $2k \leq n$ nebo $n - k \geq i$, žádná zranitelná zpráva neexistuje.

Důkaz. Mějme zprávy $\boldsymbol{m}$ a $\boldsymbol{m'}$, kde jedna vznikla z druhé vypuštěním (nebo přidáním) i-té souřadnice zleva a následně kratší z nich byla doplněna zleva nulou. Délka zprávy je buď j v případě vypuštění nebo $j+1$ v případě přidání - to označme obecně jako j'. Označme $\boldsymbol{r}$ rozdíl prefixů slov $\boldsymbol{m}$ a $\boldsymbol{m'}$. Máme tak rovnost

$$\boldsymbol{m} + \boldsymbol{m'} = \boldsymbol{r} \ 0 \dots 0.$$

BÚNO je $\boldsymbol{m}$ zpráva, která nebyla zleva doplněna nulou. Rozdíl prefixů je podle předchozího pozorování roven

$$\begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{pmatrix} \begin{pmatrix} m_{j'-1} \\ m_{j'-2} \\ \vdots \\ m_{j'-i} \end{pmatrix},$$

kde matice zobrazení F je typu $i \times i$. Podle poznámky 4.2.15 mají slova $\boldsymbol{m}$ a $\boldsymbol{m'}$ stejné kontrolní bity právě tehdy, když $i > n - k$ a soustava

$$\left(\begin{array}{cccc|c} g_{n-k} & & & & \\ \vdots & \ddots & & & \\ g_0 & & g_{n-k} & & \boldsymbol{r}^T \\ & \ddots & \vdots & & \\ & & g_0 & & \end{array}\right)$$

má řešení nad $\mathbb{F}_2$. Nyní výše uvedené aplikujeme na zprávu

$$\boldsymbol{u} = u_{j-1} \dots u_0.$$

Začneme s variantou zkrácení. Dostáváme, že zpráva $\boldsymbol{u}$ je zranitelná při zkrácení na i-té pozici zleva právě tehdy, když existuje řešení soustavy nad $\mathbb{F}_2$

$$\begin{pmatrix} g_{n-k} & & & \\ \vdots & \ddots & & \\ g_0 & & g_{n-k} & \\ & \ddots & \vdots & \\ & & g_0 & \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{i-1-n+k} \end{pmatrix} = \begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{pmatrix} \begin{pmatrix} u_{j-1} \\ u_{j-2} \\ \vdots \\ u_{j-i} \end{pmatrix}. \tag{4.2}$$

V případě prodloužení je soustava velice podobná. Jediný rozdíl je v tom, že i-tá souřadnice zleva u_{j-i} není tentokrát z původní zprávy, ale jde o nově přidaný bit s hodnotou b. Soustava je tak ve tvaru

$$\begin{pmatrix} g_{n-k} & & \\ \vdots & \ddots & \\ g_0 & & g_{n-k} \\ & \ddots & \vdots \\ & & g_0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{i-1-n+k} \end{pmatrix} = \begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{pmatrix} \begin{pmatrix} u_{j-1} \\ \vdots \\ u_{j-i+1} \\ b \end{pmatrix}. \tag{4.3}$$

Na závěr si všimneme, že $(a_0 \ \dots \ a_{i-1-n+k} \ u_{j-1} \ \dots \ u_{j-i})^T$ je řešením soustavy (4.2) právě tehdy, když

$$(a_0 \ \dots \ a_{i-1-n+k} \ u_{j-1} \ \dots \ u_{j-i})^T \in K$$

a $(a_0 \ \dots \ a_{i-1-n+k} \ u_{j-1} \ \dots \ u_{j-i+1} \ b)^T$ je řešením soustavy (4.3) právě tehdy, když

$$(a_0 \ \dots \ a_{i-1-n+k} \ u_{j-1} \ \dots \ u_{j-i+1} \ b)^T \in K.$$

Počet řádků matice je i a počet sloupců je roven

$$(i - (n-k)) + i = 2i - n + k.$$

Jestliže $2k \leq n$ nebo $n - k \geq i$, dané soustavy nelze sestavit, a tedy neexistuje žádné řešení. □

Poznámka 4.2.17. *V předpokladu věty 4.2.16 můžeme v případě zkrácení uvažovat i variantu, že $n - k < i \leq j \leq k$.*

Značení. *Matici ze znění věty 4.2.16 budeme značit $\boldsymbol{Z} = (\boldsymbol{Z_1}|\boldsymbol{Z_2})$.*

Pozorování. *Zprávy délky $j > i \in \mathbb{N}$ zranitelné při vypuštění i-té souřadnice tvoří podprostor aritmetického vektorového prostoru $\mathbb{F}_2^j$. Totéž platí i pro zprávy zranitelné při přidání i-té souřadnice.*

Důkaz. Plyne z věty 4.2.16 a tvaru báze jádra matice $\boldsymbol{Z}$.

□

Tvrzení 4.2.18. *Nechť $n \neq k$ jsou parametry kódu. Zpráva je zranitelná při přidání i-té souřadnice s hodnotou 1, nebo s hodnotou 0. Nikdy není zranitelná v obou případech zároveň.*

Důkaz. Pokud by zpráva byla zranitelná v obou případech, platilo by (s parametry z věty 4.2.16), že

$$(a_0 \ \dots \ a_{i-1-n+k} \ u_{j-1} \ \dots \ u_{j-i+1} \ 0)^T \in K,$$
$$(b_0 \ \dots \ b_{i-1-n+k} \ u_{j-1} \ \dots \ u_{j-i+1} \ 1)^T \in K,$$

a tedy i jejich součet

$$(c_0 \ \dots \ c_{i-1-n+k} \ 0 \ \dots \ 0 \ 1)^T \in K.$$

To by znamenalo, že existuje řešení soustavy

$$\begin{pmatrix} g_{n-k} & & \\ \vdots & \ddots & \\ g_0 & & g_{n-k} \\ & \ddots & \vdots \\ & & g_0 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{i-1-n+k} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

To ale není možné, což je vidět, pokud vyhodnocujeme proměnné od c_0. Matice $\boldsymbol{Z_1}$ by musela být jednotková a to nastane pouze v případě $n = k$ a $g_0 = 1$ a to vylučujeme.

□

Pozorování. *Pro každou i-tici $u_{j-i} \dots u_{j-i}$ existuje nejvýše jeden vektor*

$$(a_0 \ \dots \ a_{i-1-n+k} \ u_{j-1} \ \dots \ u_{j-i})^T \in K.$$

Důkaz. Plyne z toho, že $\boldsymbol{Z_1}$ obsahuje lineárně nezávislé sloupce.

□

Důsledek 4.2.19. *Mějme CRC $[n, k]$-kód s generujícím polynomem $g(x) \in \mathbb{F}_2[x]$, $x \nmid g(x)$. Jestliže*

$$2k > n, \ \ n - k < i \leq j < k, \ \ i, j \in \mathbb{N},$$

pak počet zranitelných zpráv délky j při vypuštění i-té souřadnice (zleva) je

$$2^{k-n+j}$$

a počet zranitelných zpráv délky j při přidání i-té souřadnice (zleva) s hodnotou 0 nebo 1, je

$$2^{k-n+j+1}.$$

Jestliže $2k \leq n$ nebo $i \leq n - k$, jsou obě hodnoty nulové.

Důkaz. Nejprve začneme s variantou vypuštění souřadnice. Podle věty 4.2.16 je zranitelnost zprávy při vypuštění i-té souřadnice zleva určena z jejích prvních i bitů zleva, které jsou součástí jádra matice $\boldsymbol{Z}$. Počet prvků v jádře je roven

$$2^{\dim(\mathrm{Ker}\ \boldsymbol{Z})}$$

a v kombinaci s předchozím pozorovaní, je počet zranitelných zpráv při vypuštění i-té souřadnice roven

$$2^{\dim(\mathrm{Ker}\ \boldsymbol{Z})} \cdot 2^{j-i},$$

protože na $j - i$ pozicích může zpráva nabývat libovolné hodnoty. Všechny řádky matice $\boldsymbol{Z}$ jsou lineárně nezávislé, proto

$$\dim(\mathrm{Ker}\ \boldsymbol{Z}) = (2i - n + k) - i = i - n + k.$$

Počet zranitelných zpráv délky j při vypuštění i-té souřadnice je

$$2^{\dim(\mathrm{Ker}\ \boldsymbol{Z})} \cdot 2^{j-i} = 2^{k-n+j}.$$

V případě prodloužení zprávy je situace trochu odlišná. Podle věty 4.2.16 je zranitelnost zprávy při přidání i-té souřadnice určena z prvních $i-1$ bitů zleva a hodnoty b přidaného bitu. Počet všech zpráv délky j zranitelných při přidání bitu 0 nebo 1 je tak roven

$$2^{\dim(\mathrm{Ker}\ \boldsymbol{Z})} \cdot 2^{j-(i-1)} = 2^{k-n+j+1}.$$

□

Poznámka 4.2.20. *Do konce kapitoly 4.2.4 implicitně předpokládáme, že*

- *hodnota slova délky $j \in \mathbb{N}$,*
- *hodnota přidaného bitu b při prodloužení,*
- *souřadnice, na které dojde k prodloužení / ke zkrácení (v povoleném rozsahu),*

mají uniformní rozdělení.

Důsledek 4.2.21. *Mějme stejné předpoklady jako v důsledku 4.2.19. Jestliže*

$$2k > n, \quad n - k < i \leq j < k,$$

pak pravděpodobnost, že je zpráva délky j zranitelná při vypuštění (přidání) i-té souřadnice (zleva), je

$$2^{k-n}.$$

Jestliže $2k \leq n$ nebo $i \leq n-k$, je pravděpodobnost nulová.

Důkaz. Počet všech zpráv délky j je 2^j. V případě vypuštění i-té souřadnice je pravděpodobnost rovna podílu počtu zranitelných zpráv (z důsledku 4.2.19) a počtu všech zpráv, neboli

$$\frac{2^{k-n+j}}{2^j} = 2^{k-n}.$$

V případě přidání i-té souřadnice musíme rozlišit hodnotu b. Výsledek dostáváme z následujícího výpočtu

$$\frac{1}{2} \cdot P(\text{zranitelnost} \mid b = 0) + \frac{1}{2} \cdot (\text{zranitelnost} \mid b = 1) = \frac{1}{2} \cdot \frac{2^{k-n+j+1}}{2^j} = 2^{k-n}.$$

□

Příklad. Hodnoty pravděpodobnosti 2^{n-k} pro používané kódy v CAN FD protokolu jsou:

$$\text{CRC-17: } 2^{-17} \doteq 7{,}63 \cdot 10^{-6},$$
$$\text{CRC-21: } 2^{-21} \doteq 4{,}77 \cdot 10^{-7}.$$

□

Věta 4.2.22. *Mějme CRC $[n,\ k]$-kód s generujícím polynomem $g(x) \in \mathbb{F}_2[x]$, $x \nmid g(x)$. Pravděpodobnost, že zpráva délky $j < k$, která byla při přenosu poškozena vypuštěním nebo přidáním jedné souřadnice, nebude odhalena CRC kontrolou, je*

$$\begin{aligned} \frac{j-n+k}{j} \cdot 2^{k-n}, &\text{ došlo-li ke zkrácení,} \\ \frac{j+1-n+k}{j+1} \cdot 2^{k-n}, &\text{ došlo-li k prodloužení,} \end{aligned}$$

jestliže $2k > n$. Pokud $2k \leq n$, jsou tyto pravděpodobnosti nulové.

Důkaz. Z důsledku 4.2.21 známe pravděpodobnosti chyb na jednotlivých souřadnicích. Stačí spočítat výslednou pravděpodobnost. Ke zkrácení může dojít na j pozicích a k prodloužení na $j+1$ pozicích. Označme pravděpodobnost zkrácení (přidání) i-té souřadnice (zleva) jako $P(i)$. Pravděpodobnost při zkrácení je rovna

$$\begin{aligned} \sum_{i=1}^{j} P(\text{zranitelnost} \mid i) \cdot P(i) &= \sum_{i=n-k+1}^{j} P(\text{zranitelnost} \mid i) \cdot P(i) \\ &= \frac{1}{j} \cdot \sum_{i=n-k+1}^{j} 2^{k-n} = \frac{j-n+k}{j} \cdot 2^{k-n}. \end{aligned}$$

Pravděpodobnost při prodloužení je rovna

$$\begin{aligned} \sum_{i=1}^{j+1} P(\text{zranitelnost} \mid i) \cdot P(i) &= \sum_{i=n-k+1}^{j+1} P(\text{zranitelnost} \mid i) \cdot P(i) \\ &= \frac{1}{j+1} \cdot \sum_{i=n-k+1}^{j+1} 2^{k-n} = \frac{j+1-n+k}{j+1} \cdot 2^{k-n}. \end{aligned}$$

□

Pozorování. *CRC $[2k,\ k]$-kód odhalí všechny chyby způsobené zkrácením nebo prodloužením na jedné pozici. Takový kód je ale dost nepraktický, protože zakódovaná zpráva je dvakrát delší než původní.*

Příklad. Mějme CRC [15, 11]-kód. Za stejných předpokladů jako ve větě 4.2.22 spočítáme pravděpodobnost, že slovo délky 10 poškozené vypuštěním nebo přidáním náhodné souřadnice (s uniformním rozdělením), nebude takovým kódem odhaleno. Dosazením do vzorce dostáváme

$$\begin{aligned} \frac{6}{10} \cdot 2^{-4} &= 0{,}0375 \to 3{,}75\ \%, \text{ v případě zkrácení,} \\ \frac{7}{11} \cdot 2^{-4} &\doteq 0{,}0398 \to 3{,}98\ \%, \text{ v případě prodloužení.} \end{aligned}$$

Máme-li například kód s parametry [255, 238] používaný v CAN FD (více viz kapitola 4.3) a slovo délky 160, vychází pravděpodobnosti následovně

$$\begin{aligned} \frac{143}{160} \cdot 2^{-17} &\doteq 6{,}819 \cdot 10^{-6} \to 0{,}000681\ \%, \text{ v případě zkrácení,} \\ \frac{144}{161} \cdot 2^{-17} &\doteq 6{,}824 \cdot 10^{-6} \to 0{,}000682\ \%, \text{ v případě prodloužení.} \end{aligned}$$

□

Důsledek 4.2.23. *Množina zranitelných zpráv není ovlivněna volbou generujícího polynomu za podmínky, že $x \nmid g(x)$. Bezpečnost je ovlivněna pouze počtem kontrolních bitů, neboli parametry n a k.*

Příklad. Mějme CRC [14, 11]-kód s generujícím polynomem

$$g(x) = x^3 + x + 1.$$

Určíme všechna zranitelná slova délky $j \geq 5$ na 5. souřadnici zleva. Nejprve podle věty 4.2.16 spočítáme jádro matice

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

Výpočtem dostaneme, že

$$\begin{aligned} \operatorname{Ker} \boldsymbol{Z} &= \left\langle (0\ 1\ 0\ 1\ 1\ 0\ 1)^T, (1\ 1\ 1\ 0\ 1\ 1\ 0)^T \right\rangle \\ &= \left\{ (0\ 0\ 0\ 0\ 0\ 0\ 0)^T, (0\ 1\ 0\ 1\ 1\ 0\ 1)^T, (1\ 1\ 1\ 0\ 1\ 1\ 0)^T, (1\ 0\ 1\ 1\ 0\ 1\ 1)^T \right\}. \end{aligned}$$

Vektory jádra jsou ve tvaru $(a_0\ a_1\ u_{j-1} \dots u_{j-5})^T$ v případě zkrácení a ve tvaru $(a_0\ a_1\ u_{j-1} \dots u_{j-4}\ b)^T$ v případě prodloužení. Hledané prefixy zpráv způsobující zranitelnost vypuštění 5. souřadnice zleva jsou ve tvaru

$$(u_{j-1} \dots u_{j-5})^T.$$

Všechny zprávy délky $5 \leq j \leq 11$, které začínají

$$00000,\ 01101,\ 10110 \text{ nebo } 11011$$

jsou zranitelné při vypuštění 5. souřadnice (zleva). Vyzkoušíme to na zprávě

$$11011111010 \overset{m}{\sim} x^{10} + x^9 + x^7 + x^6 + x^5 + x^4 + x^3 + x = g_1(x).$$

Po zkrácení 5. souřadnice dostáváme slovo

$$1101111010 \overset{m}{\sim} x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x = g_2(x)$$

a výpočtem ověříme, že platí

$$x^4 \cdot g_1(x) \bmod g(x) = x^4 \cdot g_2(x) \bmod g(x) = x^2 + 1 \overset{m}{\sim} 101.$$

Navíc první dvě souřadnice vektorů z jádra popisují, jak se slovo před a po zkrácení liší. V našem případě jde o hodnotu 10, což znamená, že

$$g_2(x) = g_1(x) + 1 \cdot x^7 \cdot g(x) + 0 \cdot x^7 \cdot g(x).$$

Zafixujeme-li délku slov, tak pravděpodobnost, že náhodně vybrané slovo bude zranitelné na 5. souřadnici při zkrácení, je

$$\frac{4}{2^5} = \frac{1}{8} = 2^{-3}.$$

Nyní se podíváme na případ prodloužení. Prefixy zpráv, které způsobují zranitelnost při přidání 5. souřadnice, jsou ve tvaru

$$(u_{j-1} \dots u_{j-4}\ b)^T.$$

Všechny zprávy délky $5 \leq j \leq 10$, které začínají

$$0000 \text{ nebo } 1011,$$

jsou zranitelné při přidání 5. souřadnice s hodnotou 0. S hodnotou 1 jsou zranitelná slova začínající

$$0110 \text{ nebo } 1101.$$

Vyzkoušíme to například na zprávě

$$0110111101 \overset{m}{\sim} x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + 1 = g_3(x),$$

která po přidání 5. souřadnice s hodnotou 1 vypadá následovně

$$01101111101 \overset{m}{\sim} x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + 1 = g_4(x).$$

Výpočtem ověříme, že platí

$$x^4 \cdot g_3(x) \bmod g(x) = x^4 \cdot g_4(x) \bmod g(x) = x^2 + x + 1 \bmod 111$$

Tentokrát první dvě souřadnice vybraného prefixu zprávy mají hodnotu 01, takže platí

$$g_4(x) = g_3(x) + x^6 \cdot g(x).$$

Zafixujeme-li délku slov, tak pravděpodobnost, že náhodně vybrané slovo bude zranitelné při přidání 5. souřadnice, je

$$\frac{1}{2} \cdot \frac{2}{2^4} + \frac{1}{2} \cdot \frac{2}{2^4} = \frac{1}{8} = 2^{-3}.$$

□

Není pravda, že by zranitelná zpráva, která má správnou strukturu definovanou protokolem, protokolu vždy unikla. Pokud nedojde ke zkrácení nebo prodloužení ve spojení se stuff bitem, bude takové poškození detekováno jiným mechanismem. Navíc při přenosu rámce může dojít ke zkrácení nebo prodloužení vícekrát. V této kapitole byl uveden pouze popis zranitelných zpráv při jedné chybě. Popis dalších variant by byl komplikovanější. Bohužel i v tomto případě jsme zjistili, že nelze snížit výskyt této chyby v původní verzi CAN FD vhodnou volbou generujícího polynomu.

4.2.5 Popis upraveného protokolu

Trvalo rok, než se podařilo dohodnout řešení, které se snaží opravit uvedenou chybu. K ISO certifikaci novější verze došlo v roce 2015 a tato verze se používá dodnes. Původní návrh se označuje také jako non-ISO CAN FD a upravená verze jako ISO CAN FD.

Pro eliminaci chyby způsobenou zkrácením nebo prodloužením zprávy v průběhu přenosu se autoři rozhodli do rámce přidat další čtyři bity, které říkají, kolik dynamických stuff bitů bylo do zprávy přidáno. V takovém případě víme, že jsme po přenosu obdrželi stejný počet stuff bitů a že se kontrolní CRC bity počítají na stejně dlouhé posloupnosti jako před přenosem.

Umístění bloku, který obsahuje zakódovanou informaci o počtu dynamických stuff bitů ve zprávě, je zobrazeno na obrázku 4.6 s označením SC. Ve standardní i rozšířené verzi datové zprávy je modifikace shodná.

Obrázek 4.6: Popis modifikované části datové zprávy protokolu CAN FD.

Počet stuff bitů se odesílá modulo 8. To znamená, že jsme schopni detekovat až 7 ztracených, nebo přidaných stuff bitů. To je dostatečné množství, protože se na ochranu ostatních částí využívá CRC kód schopný detekovat až 5 chyb (viz kapitola 4.3).

Tyto nově přidané bity je potřeba také zabezpečit. Pokud při přenosu dojde ke změně počtu stuff bitů a zároveň dojde k poškození SC bloku, nemusí se tato chyba opět odhalit. Proto kromě tří bitů nesoucí počet přidáváme ještě paritní bit. Navíc se počet nekóduje jako binární zápis, ale pomocí převodní tabulky 4.5. Na závěr se za tyto čtyři nové bity vkládá jeden statický stuff bit.

Počet mod 8	Kód	Parita
0	000	0
1	001	1
2	011	0
3	010	1
4	110	0
5	111	1
6	101	0
7	100	1

Tabulka 4.5: Podoba SC bloku nesoucí informaci o počtu dynamických stuff bitů v datové zprávě.

Poznámka 4.2.24. *Každé dva po sobě jdoucí kódy v tabulce 4.5 se liší vždy na jednom bitu. Takovému uspořádání se říká Grayův kód. Tato metoda kódování nemá vliv na zabezpečení přenosu, ale je zde z technických důvodů.*

Příklad. Otestujeme bezpečnost těchto nových pěti bitů (4 + 1 stuff bit). **Tučná** hodnota znamená pozici chyby a podtržená hodnota určuje bit, který danou chybu odhalí. Navíc vždy uvádíme pouze reprezentanta dané skupiny chyb. C určuje bit popisující hodnotu počtu, P značí paritní bit a S stuff bit.

Počet chyb	Odhaleno	Neodhaleno
1	**C**CCPS, CCC**P**S, CCCP**S**	
2	**C**CC**P**S, **C**CCP**S**, CCC**PS**	**CC**CPS
3	**CCC**PS, **CC**C**P**S, **CC**CP**S**	**C**CC**PS**
4	**CCCP**S, **CCC**P**S**, **CC**C**PS**	
5		**CCCPS**

Z tabulky vyplývá, že odhalíme

- všechny chyby, pokud je právě jedna chyba v paritním nebo stuff bitu.
- lichý počet chyb, pokud není chyba v paritním ani stuff bitu.
- sudý počet chyb, pokud je chyba v paritním i stuff bitu.

Naopak je například vidět, že zabezpečení nestačí v případě, kdy se objeví dvě chyby na prvních třech bitech.

□

Pozorování. *Ačkoliv ISO CAN FD využívá CRC kód s Hammingovou vzdáleností 6, nemusí protokol označit zprávu se* ***čtyřmi chybami*** *za chybnou.*

Důkaz. K takové situaci dojde v případě, že zároveň nastalo:

- v SC bloku dojde ke dvěma chybám, které jsou na prvních 3 bitech
- dojde ke ztrátě / přidání dynamických bitů v takovém množství, že výsledný počet je stejný jako poškozená hodnota v SC bloku.

Změna dvou bitů v SC bloku znamená odchylku od původní hodnoty alespoň o 2. Pokud nastane právě tato situace a navíc při přenosu vypadnou, nebo se přidají dva dynamické stuff bity, nemusí být taková chyba detekována.

□

Samotná kontrola se provádí tak, že příjemce spočítá počet přijatých dynamických stuff bitů a porovná toto číslo s přijatou hodnotou v SC bloku. Pokud se hodnoty liší, zachová se protokol stejným způsobem, jako by došlo k chybě při kontrole CRC kontrolních bitů.

Technická poznámka 4.2.25. *Při přenosu může dojít zároveň k prodloužení i ke zkrácení posloupnosti. Odesílatel a příjemce tak vidí stejný počet stuff bitů, ale ne všechny jsou na původních pozicích.*

Přestože se do protokolu přidaly další extra bity spolu s netriviálním mechanismem, které slouží ke zvýšení bezpečnosti při přenosu, tak ani tak není protokol zabezpečený proti zkracováním a prodlužováním posloupnosti.

Tvrzení 4.2.26. *Certifikovaná a aktuálně používaná modifikovaná verze CAN FD protokolu popsaná v této kapitole* ***je stále*** *zranitelná před zkracováním a prodlužováním posloupnosti. Pokud nastane případ z poznámky 4.2.25, je informace o počtu stuff bitů bezvýznamná a opakuje se situace z původní verze protokolu.*

Zranitelnost demonstrujeme na následujícím příkladu, kdy nemusí být přijatá zpráva se dvěma chybami označena jako chybná.

Příklad. Odešleme zprávu s jedním stuff bitem ve tvaru

$$1\ 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1.$$

Při přenosu dojde nejprve ke zkrácení a poté k prodloužení a příjemce obdrží zprávu

$$1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ \mathbf{0}\ 1.$$

Obě zprávy obsahují jeden stuff bit, takže informace z SC bloku chybu nezpůsobí. Navíc se liší na 8 pozicích, což nemusí CRC kód odhalit. Bez stuff bitů je rozdíl mezi zprávami pouze na dvou pozicích.

□

Jak je vidět, oprava funguje v případě, kdy nedošlo ke stejnému počtu prodloužení a zkrácení v rámci jedné zprávy. Pravděpodobnost chyby se snížila, ale stále není nulová, a tak protokolu může uniknout zpráva, která obsahuje dvě chyby. To je znepokojivé, neboť oprava trvala rok a ve skutečnosti tento typ chyby nebyl odstraněn.

O to horší je, že se na tuto chybu nepřišlo při certifikaci. Navíc v dostupných publikacích neexistuje zmínka, že by tato verze nebyla v pořádku. To je odlišné oproti původní verzi CAN FD, kde byla samotná chyba spolu s řešením popsána v oficiálním článku.

4.3 Analýza používaných CRC polynomů

Tvrzení 4.3.1. *Polynomiální kód $\mathscr{C}$ délky 127 s generujícím polynomem*

$$g_{15} = x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$$

je cyklický, odhalí všechny shluky chyb délky ≤ 15 a

$$d(\mathscr{C}) \geq 6,$$

neboli tento kód odhalí 5 chyb.

Důkaz. Ireducibilní rozklad g_{15} je

$$(x+1)(x^7 + x^3 + 1)(x^7 + x^3 + x^2 + x + 1).$$

Protože je 127 prvočíslo, jsou všechny prvky $\mathbb{F}_{128}^*$ primitivní, speciálně $\alpha = [x]$, a tedy jsou oba ireducibilní polynomy

$$x^7 + x^3 + 1,\ x^7 + x^3 + x^2 + x + 1$$

primitivní s exponenty 127. Kombinací tvrzení 2.4.4 a 2.4.8 dostáváme, že polynomiální kód délky maximálně 127 s generujícím polynomem g_{15} **odhaluje dvě a lichý počet chyb**. Z toho plyne, že takový kód odhaluje alespoň 3 chyby a

$$d(\mathscr{C}) \geq 4.$$

Na druhou stranu takovou vlastnost by měl i kód s generujícím polynom ve tvaru

$$(x+1)(x^7 + x^3 + 1).$$

Ve skutečnosti má tento generující polynom větší potenciál, který odhalíme pomocí tvrzení 2.4.12. Protože je $x^7 + x^3 + 1$ primitivní polynom, dostáváme okamžitě z tvrzení 2.2.5, že $\alpha \in \mathbb{F}_{2^7}$ splňující

$$\alpha^7 + \alpha^3 + 1 = 0,$$

je primitivní prvek $\mathbb{F}_{2^7}$.

Zvolíme $n = 127$ a spočítáme několik prvních minimálních polynomů $m_i(x)$ prvků α^i. Z definice α ihned dostáváme, že $m_1(x) = x^7 + x^3 + 1$. Z tvrzení 2.2.3 vyplývá, že

$$m_{2^k}(x) = x^7 + x^3 + 1,\ \ k \in \mathbb{N}.$$

Nyní spočítáme $m_3(x)$. Dosazením do obou polynomů z rozkladu a následnou úpravou výrazů zjišťujeme následující

$$\begin{aligned} (\alpha^3)^7 + (\alpha^3)^3 + 1 &= \alpha^2 + \alpha, \\ (\alpha^3)^3 + (\alpha^3)^3 + (\alpha^3)^2 + (\alpha^3) + 1 &= 0. \end{aligned}$$

Z toho plyne, že

$$m_3(x) = x^7 + x^3 + x^2 + x + 1.$$

Protože

$$\begin{aligned} (x^7 + x^3 + 1)(x^7 + x^3 + x^2 + x + 1) &= \operatorname{lcm}(m_1(x),\, m_2(x),\, m_3(x),\, m_4(x)) \\ &= m_1(x) \cdot m_3(x) \end{aligned}$$

dostáváme z tvrzení 2.4.12, že polynomiální kód délky 127 s generujícím polynomem

$$g(x) = (x^7 + x^3 + 1)(x^7 + x^3 + x^2 + x + 1)$$

má Hammingovu vzdálenost alespoň 5. Protože g_{15} obsahuje člen $x + 1$, je

$$d(\mathscr{C}) \geq 6.$$

Navíc se jedná o cyklický kód. Dělitelnost

$$(x^7 + x^3 + 1)(x^7 + x^3 + x^2 + x + 1) \mid (x^{127} - 1)$$

plyne z tvrzení 2.4.12 a dělitelnost $(x+1) \mid (x^{127} - 1)$ je zřejmá díky společnému kořenu nad $\mathbb{F}_2$, který není kořenem ostatních dvou ireducibilních polynomů. Na závěr podle tvrzení 2.5.3 tento kód odhalí všechny **shluky chyb délky** ≤ 15. □

Použitím algoritmu popsaného v tvrzení 2.1.2 na výpočet Hammingovy vzdálenosti lineárního kódu se dá zjistit následující výsledek.

Poznámka 4.3.2. *Polynomiální kód $\mathscr{C}$ z tvrzení 4.3.1 splňuje $d(\mathscr{C}) = 6$. To dokazuje například kódové slovo*

$$x^{81} + x^{17} + x^4 + x^2 + x + 1.$$

Tvrzení 4.3.3. *Polynomiální kód $\mathscr{C}$ délky 255 s generujícím polynomem*

$$g_{17} = x^{17} + x^{16} + x^{14} + x^{13} + x^{11} + x^6 + x^4 + x + 1$$

je cyklický, odhalí všechny shluky chyb délky ≤ 17 a

$$d(\mathscr{C}) \geq 4,$$

neboli tento kód odhalí alespoň 3 chyby.

Důkaz. Ireducibilní rozklad g_{17} je

$$(x+1)(x^8 + x^7 + x^2 + x + 1)(x^8 + x^7 + x^6 + x + 1).$$

Ze seznamu primitivních polynomů z [21] zjistíme, že oba polynomy

$$x^8 + x^7 + x^2 + x + 1,\ x^8 + x^7 + x^6 + x + 1$$

jsou primitivní, a proto jsou exponenty těchto polynomů

$$2^8 - 1 = 255.$$

Kombinace tvrzení 2.4.4 a 2.4.8 říká, že polynomiální kód délky maximálně 255 s generujícím polynomem g_{17} **odhaluje dvě a lichý počet chyb**. Proto

$$d(\mathscr{C}) \geq 4.$$

Použitím BCH meze z tvrzení 2.4.12 nedostaneme lepší odhad. Použijeme-li primitivní prvek $\alpha \in \mathbb{F}_{2^8}$ splňující

$$\alpha^8 = \alpha^7 + \alpha^2 + \alpha + 1$$

a zvolíme $n = 255$, dostaneme pouze

$$m_{2^k}(x) = x^8 + x^7 + x^2 + x + 1,\ \ k \in \mathbb{N}.$$

Stejný výsledek dostaneme i tehdy, když použijeme primitivní prvek příslušný druhému primitivnímu polynomu. Z tvrzení 2.4.12 ale ze stejného důvodu jako z předchozího tvrzí plyne, že je tento kód délky 255 cyklický. Navíc podle tvrzení 2.5.3 odhalí všechny **shluky chyb délky** ≤ 17.

□

Je zajímavé, že g_{17} nezvolili při návrhu CAN FD stejným způsobem jako g_{15} a g_{21}. Stačilo by použít polynom ve tvaru

$$(x+1)(x^8+x^7+x^2+x+1)(x^8+x^7+x^6+x^4+x^3+x^2+1),$$

který by ihned zajistil vzdálenost kódu alespoň 6 jako v ostatních případech. Naštěstí situace není v tomto případě horší, než u ostatních polynomů.

Poznámka 4.3.4. *Polynomiální kód z tvrzení 4.3.3 splňuje $d(\mathscr{C}) = 6$. To dokazuje například kódové slovo*

$$x^{118}+x^{32}+x^7+x^2+x+1.$$

Vzdálenost 4 lze vyloučit výpočtem hrubou silou pomocí tvrzení 2.1.2.

Tvrzení 4.3.5. *Polynomiální kód $\mathscr{C}$ délky 1023 s generujícím polynomem*

$$g_{21} = x^{21}+x^{20}+x^{13}+x^{11}+x^7+x^4+x^3+1$$

je cyklický, odhalí všechny shluky chyb délky ≤ 21 a $d(\mathscr{C}) \geq 6$, neboli tento kód odhalí alespoň 5 chyb.

Důkaz. Ireducibilní rozklad g_{21} je

$$(x+1)(x^{10}+x^3+1)(x^{10}+x^3+x^2+x+1).$$

Tentokrát je z polynomů

$$x^{10}+x^3+1,\ x^{10}+x^3+x^2+x+1$$

primitivní pouze první z nich a jeho exponent je proto

$$2^{10}-1 = 1023.$$

Kombinací tvrzení 2.4.4 a 2.4.8 dostáváme, že polynomiální kód délky maximálně 1023 s generujícím polynomem g_{21} **odhaluje dvě a lichý počet chyb**. Proto

$$d(\mathscr{C}) \geq 4.$$

I zde zkusíme využít BCH mez z tvrzení 2.4.12. Použijeme primitivní prvek $\alpha \in \mathbb{F}_2^{10}$ splňující

$$\alpha^{10} = \alpha^3 + 1$$

Zvolíme $n = 1023$ a budeme postupně počítat minimální polynomy $m_i(x)$ prvků α^i. Stejně jako v tvrzení 4.3.1 dostáváme

$$m_{2^k}(x) = x^{10}+x^3+1,\ \ k \in \mathbb{N}.$$

Nyní se podíváme na $m_3(x)$. Dosazením do obou polynomů z rozkladu a následné úpravě výrazu (na počítači) dostáváme

$$(\alpha^3)^{10} + (\alpha^3)^3 + 1 \neq 0,$$
$$(\alpha^3)^{10} + (\alpha^3)^3 + (\alpha^3)^2 + (\alpha^3) + 1 = 0,$$

z čehož vyplývá, že

$$m_3(x) = x^{10} + x^3 + x^2 + x + 1.$$

Tvrzení 2.4.12 nám proto říká, že polynomiální kód s generujícím polynomem g_{21} délky 1023 má vzdálenost alespoň 5 a díky členu $x + 1$ v rozkladu, je

$$d(\mathscr{C}) \geq 6.$$

Obdobně jako výše se i zde jedná o cyklický kód délky 1023. Navíc podle tvrzení 2.5.3 odhalí všechny **shluky chyb délky** ≤ 21.

□

Poznámka 4.3.6. *Polynomiální kód z tvrzení 4.3.5 splňuje $d(\mathscr{C}) = 6$. To dokazuje například kódové slovo*

$$x^{984} + x^{967} + x^7 + x^2 + x + 1.$$

Příklad. Pro zajímavost spočítáme exponent polynomu $x^{10} + x^3 + x^2 + x + 1$. Využijeme toho, že

$$x^{1023} \bmod x^{10} + x^3 + x^2 + x + 1 = 1,$$

což plyne z vlastnosti všech prvků grupy $\mathbb{F}^*_{2^{10}}$. Protože

$$1023 = 3 \cdot 11 \cdot 31,$$

stačí najít nejmenší $e \in \{3, 11, 31, 33, 93, 341\}$ takové, že

$$x^e \bmod x^{10} + x^3 + x^2 + x + 1 = 1.$$

Výpočtem dostaneme, že exponent je roven $e = 341$.

□

Závěr

Hlavním cílem práce bylo matematicky popsat CRC kódy, představit protokol CAN a jeho novější verzi CAN FD a popsat jejich chyby ve spojení s CRC kódy. Chyba v CAN FD protokolu měla vážnější důsledky, a proto dalším cílem práce bylo vytvořit její matematický popis, na základě něho odvodit významné vlastnosti a případně navrhnout vhodnější typ CRC kódu, který by tuto chybu omezil.

Samotný text je rozdělen do čtyř kapitol. V první kapitole jsme uvedli základní terminologii a pojmy z teorie samoopravných kódů potřebných pro popis CRC kódů. Druhá kapitola je věnována teorii polynomiálních kódů a jejich detekčním a opravovacím schopnostem. Zjistili jsme, že v jistém smyslu nejlepšími polynomiálními kódy jsou kódy s primitivním generujícím polynomem. Pokud délka takového kódu nepřesáhne exponent generujícího polynomu, je kód schopný detekovat až 2 chyby.

CRC kódům je věnována třetí kapitola. V první části jsme tyto kódy matematicky popsali. Jedná se o prosté polynomiální kódy s konkrétní podobou kódování. V druhé části jsme srovnali různé přístupy jejich implementace. Využitím paměti lze výpočet na dnešních procesorech výrazným způsobem urychlit.

V první části čtvrté kapitoly jsou uvedeny a porovnány CAN a CAN FD včetně popisu jejich chyb. V další části kapitoly se podařilo matematicky popsat chybu v necertifikované verzi CAN FD protokolu a to se využilo k charakterizaci všech zpráv, které při jednom zkrácení nebo prodloužení posloupnosti projdou CRC kontrolou bez chyby. To je zásadní chyba, neboť takto poškozená zpráva se od původní liší pouze na jedné pozici.

Všechny takto zranitelné zprávy dané délky lze získat vyřešením homogenní soustavy lineárních rovnic, konkrétně výpočtem jádra matice. Díky tomu se podařilo odvodit pravděpodobnost výskytu této chyby, což vedlo k negativnímu závěru - pravděpodobnost, že zpráva poškozená zkrácením nebo prodloužením unikne CRC kontrole, nezávisí na volbě generujícího polynomu. Z toho vyplývá, že sebelepší volbou generujícího polynomu nemůžeme chybovost v protokolu ovlivnit.

Pravděpodobnost výskytu chyby je závislá pouze na parametrech n a k daného polynomiálního kódu. Konkrétně jsme zjistili, že čím více kontrolních bitů přidáme, tím nižší je tato pravděpodobnost. Speciálně pokud dosáhneme hodnoty $n = 2k$, pak takový kód nezávisle na tvaru generujícího polynomu odhalí všechny zprávy poškozené zkrácením nebo prodloužením na jedné pozici. To jde ale proti efektivitě přenosu dat, protože kontrolní bity nenesou samotnou přenášenou informaci.

Závěrem čtvrté kapitoly je i popis upraveného protokolu CAN FD. Nepříjemným zjištěním v této práci je to, že i opravený protokol stále obsahuje stejný typ chyby, jenom se snížila pravděpodobnost jeho výskytu. Navíc tomuto faktu není na rozdíl od předchozí zranitelnosti věnována v dostupné literatuře žádná pozornost. Na závěr celé práce byla provedena analýza používaných CRC polynomů za pomocí teorie z prvních tří kapitol. Ve většině případů jsme byli schopni ověřit informace, které jsou známé z výpočtu hrubou silou a to, že používané kódy mají Hammingovu vzdálenost 6. Odhalují tak až 5 chyb.

Hlavní přínos této práce se dá shrnout do tří bodů. Za prvé jde o sjednocení

teorie kolem polynomiálních kódů z různých zdrojů, které jsou doplněny vlastními příklady, poznámkami, vysvětlivkami a některými vlastními důkazy. Za druhé je to vytvoření matematického popisu CRC kódů. V dostupných publikacích se na CRC kódy dívají z inženýrského pohledu, zatímco zde byly popsány matematicky a zasazeny do teorie polynomiálních kódů. Popis efektivní implementace CRC kódů založený na teorii lineárního zobrazení je též přínosem této práce.

Posledním a hlavním přínosem práce je matematický popis zranitelnosti v původní necertifikované verzi CAN FD spolu s charakterizací zranitelných zpráv, které mohou protokolu uniknout. Vlastním přínosem je též analýza používaných CRC kódů. Běžně se totiž vlastnosti polynomiálních kódů podobných parametrů testují hrubou silou.

Seznam použité literatury

[1] S. Ho. Polynomial codes. 2010. http://math.mit.edu/~shor/18.310/polycode.pdf.

[2] T. Radu. Error-correcting codes. 2013. http://math.ubbcluj.ro/~tradu/TI/erccodes.pdf.

[3] A. Ta-Shma and D. Doron. Polynomial codes and cyclic codes. 2017. http://www.cs.tau.ac.il/~amnon/Classes/2017-ECC/Lectures/Lecture2.pdf.

[4] S. O'Connor. Computing primitive polynomials - theory and algorithm. http://www.seanerikoconnor.freeservers.com/Mathematics/AbstractAlgebra/PrimitivePolynomials/theory.html. [13.01.2021].

[5] W. W. Peterson and D. T. Brown. Cyclic codes for error detection. 1960. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.680.3771&rep=rep1&type=pdf.

[6] P. Geremia. Cyclic redundancy check computation: An implementation using the TMS320C54x. 1999. https://www.ti.com/lit/an/spra530/spra530.pdf.

[7] CAN history. https://www.can-cia.org/can-knowledge/can/can-history/. [10.02.2021].

[8] CAN code. https://en.wikipedia.org/wiki/CAN_bus. [10.02.2021].

[9] K. Polák. Sběrnice CAN. 2003. http://www.elektrorevue.cz/clanky/03021/index.html.

[10] R. Bosch. CAN specification. 1991. https://www.kvaser.com/software/7330130980914/V1/can2spec.pdf.

[11] J. A. Cook and J. S. Freundenberg. Controller area network (CAN). 2008. https://www.eecs.umich.edu/courses/eecs461/doc/CAN_notes.pdf.

[12] M. Zavidčák. CAN - popis struktury. 2004. https://vyvoj.hw.cz/navrh-obvodu/rozhrani/can-popis-struktury.html.

[13] R. Bosch. CAN with flexible data-rate. 2012. https://can-newsletter.org/assets/files/ttmedia/raw/e5740b7b5781b8960f55efcc2b93edf8.pdf.

[14] BCH code. https://en.wikipedia.org/wiki/BCH_code, 2020. [25.03.2021].

[15] E. Tran. Multi-bit error vulnerabilities in the controller area network protocol. 1999. http://users.ece.cmu.edu/~koopman/thesis/etran.pdf.

[16] Polynomial code. https://en.wikipedia.org/w/index.php?title=Polynomial_code. [20.02.2021].

[17] Primitive polynomial (field theory). https://en.wikipedia.org/wiki/Primitive_polynomial_(field_theory). [27.02.2021].

[18] Mersenne prime. https://en.wikipedia.org/wiki/Mersenne_prime. [28.03.2021].

[19] A. Drápal. Samoopravné kódy. https://www2.karlin.mff.cuni.cz/~holub/soubory/drapal_kody.pdf.

[20] L. Barto and J. Tůma. Konečná tělesa. https://www2.karlin.mff.cuni.cz/~barto/student/SkriptaKonTel.pdf.

[21] P. Maurer. Primitive polynomials for the field GF(2). https://baylor-ir.tdl.org/bitstream/handle/2104/8792/GF2%20Polynomials.pdf?sequence=1&isAllowed=y.

Seznam obrázků

Seznam tabulek