

Bezpečnostní kódy v prostředí Mathematica

The error – correcting codes in Mathematica environment

Jiří Facuna

Bakalářská práce
2012



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2011/2012

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: Jiří FACUNA
Osobní číslo: A09014
Studijní program: B 3902 Inženýrská informatika
Studijní obor: Informační a řídicí technologie

Téma práce: Bezpečnostní kódy v prostředí Mathematica

Zásady pro vypracování:

1. Seznamte se s problematikou efektivních a bezpečnostní kódů, zaměřte se na Reedovy – Solomonovy kódy.
2. Zpracujte literární řešení na dané téma.
3. Navrhněte názorné příklady algoritmu kódování a dekódování RS kódů.
4. Na základě teorie naprogramujte v prostředí Mathematica algoritmus kódování a dekódování RS kódu.
5. Zhodnoťte dosažené výsledky.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. JAN MAREŠ. Teorie kódování. Praha: ČVUT, 2008. ISBN 978-80-01-04203-8.
2. KAREL VLČEK. Teorie informace a kódování. Ostrava: VŠB, 1998. ISBN 80-7078-494-6.
3. KAREL VLČEK. Teorie Informace, kódování a kryptografie. Ostrava: VŠB, 1999. ISBN 80-7078-614-0.
4. JIŘÍ, Šicner. Srovnání algoritmů dekódování Reed - Solomonova kódu. Brno, 2011. Diplomová práce. VUT Brno. Vedoucí práce Pavel Šilhavý.
5. C. K. P. CLARK. Reed - Solomon error correction [online]. BBC, 2002 [cit. 2012-01-30]. Dostupné z: <http://downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHPO31.pdf>

Vedoucí bakalářské práce:

Ing. Bronislav Chramcov, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

24. února 2012

Termín odevzdání bakalářské práce:

8. června 2012

Ve Zlíně dne 24. února 2012

prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Bakalářská práce se zabývá bezpečnostními kódy, zejména Reed - Solomonovy kódy. Teoretická část nás zavádí do oblasti bezpečnostních kódů a především vysvětluje princip kódování a dekódování RS kódu. Praktická část je zaměřena na vytvoření simulace v prostředí Mathematica pro kódování a dekódování RS kódu i s názorným příkladem.

Klíčová slova: Reed - Solomonův kód, Galoisova pole, kódování, dekódování, Mathematica

ABSTRACT

This bachelor thesis deals with safety codes, especially Reed - Solomon codes. Theoretical part us introduces in to the area of safety codes and especially explanation of the principle of coding and decoding RS code. The practical part is focused on created simulation in environment Mathematica for coding and decoding RS code and with example.

Keywords: Reed - Solomon code, Galois field, coding, decoding, Mathematica

Děkuji vedoucímu bakalářské práce panu Ing. Bc. Bronislavu Chramcovovi, Ph.D. za vedení, cenné rady a pomoc při zpracování bakalářské práce. Děkuji také své rodinně za podporu mého studia.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	8
I TEORETICKÁ ČÁST	9
1 LITERÁRNÍ REŠERŠE	10
1.1 OBECNĚ PSANÁ LITERATURA.....	10
1.2 OBECNÁ LITERATURA SE SIMULACÍ.....	10
2 BEZPEČNOSTNÍ KÓDY	12
2.1 CYKICKÉ KÓDY	13
2.1.1 RS kód.....	15
2.1.2 RS – kódování.....	16
2.1.3 RS – dekódování.....	16
3 MATEMATICKÝ ZÁKLAD	20
3.1 KONEČNÉ TĚLESO.....	20
3.2 GALOISOVA POLE	20
4 PROSTŘEDÍ MATHEMATICA	22
II PRAKTICKÁ ČÁST	24
5 PŘÍKLAD REED – SOLOMONOVA KÓDOVÁNÍ	25
5.1 RS – KÓDOVÁNÍ	25
5.2 RS – DEKÓDOVÁNÍ.....	26
6 RS V PROSTŘEDÍ MATHEMATICA	30
6.1 STRUKTURA PROGRAMU	32
6.2 POPIS PROGRAMU	33
ZÁVĚR	36
ZÁVĚR V ANGLIČTINĚ	37
SEZNAM POUŽITÉ LITERATURY	38
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	40
SEZNAM OBRÁZKŮ	42
SEZNAM TABULEK	43
SEZNAM PŘÍLOH	44

ÚVOD

Postupnou modernizací technologie stoupají nároky na komunikační systém. Je snaha, aby byl přenos dat co nejrychlejší a bezchybný, ale žádný přenos dat není absolutně odolný vůči chybám. Informace, které se posílají po komunikačním kanálu, jsou s velkou pravděpodobností chybné.

Řešení existuje v podobě bezpečnostních kódů, které dokáží detekovat a opravovat způsobené chyby. Výběr vhodného bezpečnostního kódu se odráží na jeho jednoduchosti a stupně požadovaného zabezpečení. Je potřeba co nejvíce zmenšit poměr celkového počtu kontrolních symbolů k celkovému počtu symbolů.

Zmenšením počtu symbolů se zabývá studie cyklických kódů. Jedním z těchto cyklických kódů, který tento problém řeší je Reed - Solomonův kód. Patří do symbolově orientovaných kódů s detekčními a zároveň i korekčními schopnostmi. Kódování probíhá po symbolech a pro aritmetické operace mezi symboly se využívá binární operace XOR.

První část práce seznamuje s problematikou bezpečnostních kódů, pozornost je především věnována hlavní skupině cyklických kódů, kterou tvoří RS kódy. Tato část dále podrobně popisuje algoritmy pro kódování a dekódování RS kódů.

Cílem praktické části je na základě poznatků z teorie naprogramovat algoritmus pro kódování a dekódování RS kódů v prostředí Mathematica.

I. TEORETICKÁ ČÁST

1 LITERÁRNÍ REŠERŠE

Literární rešerše je zaměřena na knihy, bakalářské nebo diplomové práce, které obsahují popis cyklických kódů. Hlavní pozornost této kapitoly je věnovaná Reed – Solomonovým (RS) kódům, ale jsou zde zmíněny také práce týkající se Bose – Chaudhuriho – Hocquenghemových (BCH) kódů. RS kódy jsou speciálním případem BCH kódů.

1.1 Obecně psaná literatura

Bakalářská práce [7] na téma bezpečnostní kódy a algoritmy jejich zabezpečení od kolegy Janíka se zabývá problematikou týkající se přenosem zpráv informačním kanálem, opravou chyb, vzniklých při přenosu informací a bezpečnostními kódy. Obsahuje informace o chybách a jejich vzniku a zároveň i ochraně vůči nim.

Problematika RS kódů se ve valné většině vyskytuje v zahraničních knihách. Detailní popis RS nebo i BCH kódů můžete najít např. v literatuře [5][9][11][12]. Najdete zde několik dekódovacích algoritmů pro oba kódy i jejich srovnání a popis Galoisova pole (GF) nebo konečných těles. Obsahují i vzorové příklady pro lepší pochopení.

Do české literatury se tato problematika moc nedostala. Nejobsáhlejší popis RS kódů v češtině je [2], která je napsaná z pohledu matematiky. Dále najdeme i zmínku v [17][18][19], ale v těchto knížkách se nevyskytuje detailní popis pro větší zpracování nebo pro pochopení funkce kódování a dekódování RS kódů.

1.2 Obecná literatura se simulací

Bakalářská práce [8] na téma kódování – cyklické kódy od kolegy Kettnera, se zabývá oblastí bezpečnostních lineárních kódů, zejména cyklických kódů. Praktická část se věnuje vykreslením obvodu kodéru a dekodéru na základě zadaných dat a vyobrazení jednotlivých fází při kódování, dekódování a opravením cyklických kódů s jednou chybou. Aplikace je vytvořena v programu Java a umístěná na webu s popisem ovládání.

Další bakalářská práce [16] na téma kódování – BCH kódy od kolegy Večerky nahlíží do problematiky BCH kódů, vysvětlení principů kódování a dekódování. Na problematiku BCH kódů vytvořil simulaci v prostředí Java.

V bakalářské práci [1] na téma cyklické bezpečnostní kódy se kolega Adamec zabývá hlavně BCH a RS kódy. Teoretická část obsahuje základní informace o kódování a dekódování. V praktické části se zaměřuje o doplnění stávající elektronické příručky

a vytvoření webové aplikace pro ověření teorie na určených kódech. Elektronická příručka byla vhodně doplněna o informace týkající se cyklických kódů. Webová aplikace pro BCH a RS kódy je vytvořena pomocí vývojového prostředí NetBeans. U aplikace by byla vhodná i ukázka kódování v polynomiálním tvaru, kde se v binárním tvaru může začátečník hůře orientovat.

Diplomová práce [12] na téma srovnání algoritmů dekodování Reed – Solomonova kódu napsal kolega Šicner v Brně na VUT. V jeho práci je obecně popsáno algebraické kódování

a dekodování RS kódů s několika druhy algoritmů pro dekodování. K tomuto tématu je vytvořen výukový program v prostředí Matlab. Vytvořená aplikace slouží jako výuková pomůcka pro kódování i dekodování, obsahuje prezentované algoritmy a umožňuje porovnání uvedených algoritmů.

2 BEZPEČNOSTNÍ KÓDY

Vlivem rušení na přenosový kanál dochází u přenášených zpráv k chybám. Ke snížení těchto chyb a zvýšení odolnosti zprávy je snaha o kódové zabezpečení. Toto zabezpečení se řeší vhodným kódováním zprávy před přenosovým kanálem, např.: kóduje se zpráva tak, aby byl použit co nejmenší počet znaků pro přenos zprávy, a přidají se doplňující zabezpečující znaky (pro zjištění chyby, popřípadě i pro jejich opravu). Po přenosu zprávy se zpráva dekóduje (zjistí chyby, popřípadě opraví) a převede zprávu do požadované podoby.

Bezpečnostní kódy dělíme podle několika kategorií. Rozdělení je znázorněno na Obr. 1.



Obr. 1. Rozdělení bezpečnostních kódů

Bezpečnostní kódy se dají ještě rozdělit na detekční a korekční.

Detekční kódy jsou takové, u kterých lze detekovat chybu. Korekční jsou detekční, ale dokážou tyto chyby opravit.

Spojité

Ve spojitých kódech se bezpečnostní prvky vkládají spojitě do posloupnosti informačních prvků.

Blokové

Přenášená zpráva se dělí do několika navzájem nezávislých bloků. Zabezpečení blokových kódů vzniká pouze v jediném bloku.

Systematické

Rozložení bezpečnostních a informačních znaků je dána podle určitého systému a jsou vždy stejná. Známe tedy znaky, které jsou bezpečnostní, a které jsou informační.

Lineární

Kterákoli lineární kombinace kódových slov je opět kódové slovo.

Nelineární

Nelineární kódy nemají vlastnosti lineárních kódů. Způsob jejich vzniku se významně liší. Vznikají nejčastěji za pomoci kódovací tabulky, která obsahuje všechny kombinace.

Nesystematické

U nesystematických kódů není známo rozložení informačních a bezpečnostních znaků.

S vnitřním zákonem

U těchto kódů nelze zjistit rozložení informačních a bezpečnostních znaků, ale zjistíme, jestli daná zpráva byla přijata bez chyby, tedy podle nějakého splněného vnitřního „zákona“ (např. počet jedniček v bloku).

Bez vnitřního zákona

Ověřit správnost přijaté kódové zprávy lze pouze porovnáním s dekodovací tabulkou.

Více informací najdete v [20].

2.1 Cyklické kódy

Snaha o kódy s co nejmenším počtem obvodových prvků vedlo k tzv. cyklickým kódům. Sériová komunikace je posloupnost symbolů a cyklické kódy jsou pro tuto oblast nejuspěšnějším řešením.

Hlavní podmínka cyklických kódů je definována tak, že cyklickým posuvem kódového slova vzniká opět kódové slovo. Pro účely posuvu se využívá posuvného registru. Cyklické kódy patří do kategorie lineárních kódů a nachází uplatnění i v dobré schopnosti detekovat chyby.

Pracují na principu výpočetních operací s mnohočleny a na popisu vlastností *konečných těles*.

Konstrukce cyklických kódů

Cyklický kód je podle podmínky takový kód, jestliže cyklický posuv kódového slova $a_0a_1a_2 \dots a_{n-1}$ vznikne kódové slovo $a_{n-1}a_0a_1 \dots a_{n-2}$.

Pro popis z matematického hlediska se kódové slovo zapisuje pomocí polynomu (mnohočlenu) ve tvaru (1).

Nyní se může použít operace sčítání, odčítání, násobení a dělení mnohočlenů podle početních pravidel platných pro danou číselnou soustavu. Např. pro mnohočleny $a(x)$ a $b(x)$ zapsané ve tvaru (2) můžeme definovat operace sčítání a násobení, které lze zapsat ve tvaru (3) a (4).

$$a_0a_1a_2 \dots a_{n-1} \leftrightarrow a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1} \quad (1)$$

$$\begin{aligned} a(x) &= a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1} \\ b(x) &= b_0 + b_1x + b_2x^2 + \dots + b_{n-1}x^{n-1} \end{aligned} \quad (2)$$

Sčítání:

$$a(x) + b(x) = (a_0 + b_0) + (a_1 + b_1)x + (a_2 + b_2)x^2 + \dots \quad (3)$$

Násobení:

$$a(x)b(x) = a_0b_0 + (a_1b_0 + a_0b_1)x + (a_2b_0 + a_1b_1 + a_0b_2)x^2 + \dots \quad (4)$$

Každý cyklický kód stupně $n - k$ obsahuje kódové slovo $g(x)$, které jako mnohočlen je stupně $n - k$. Generující mnohočlen $g(x)$ dělí beze zbytku dvojčlen $x^n - 1$. Tento podíl se nazývá kontrolní mnohočlen $h(x)$, který je znázorněn ve vztahu (5). Cyklickým posuvem koeficientů generujícího mnohočlenu vznikne generující matice G ve tvaru (6) podle [17][20] a cyklickým posuvem koeficientů kontrolního polynomu, čteného od nevyšší mocniny, vznikne kontrolní matice cyklického kódu H ve tvaru (7). Pro zabezpečené kódové slovo $c(x)$ potom platí rovnice (8) a kontrolní matice splňuje podmínku (9).

$$h(x) = \frac{x^n - 1}{g(x)} \quad (5)$$

$$G = \begin{bmatrix} g(x) \\ x \cdot g(x) \\ x^2 \cdot g(x) \\ \dots \\ x^{k-1} \cdot g(x) \end{bmatrix} = \begin{bmatrix} g_0 & g_1 & & g_{n-k} & 0 & 0 & 0 & 0 \\ 0 & g_0 & & \dots & g_{n-k} & 0 & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & & g_0 & g_1 & \dots & \dots & g_{n-k} \end{bmatrix} \quad (6)$$

$$H = \begin{bmatrix} 0 & 0 & & 0 & h_k & h_{k-1} & & h_1 & h_0 \\ 0 & 0 & & h_k & h_{k-1} & h_{k-2} & & h_0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ h_k & h_{k-1} & & h_2 & h_1 & h_0 & & 0 & 0 \end{bmatrix} \quad (7)$$

$$c(x) \cdot h(x) = 0 \quad (8)$$

$$H \cdot c^T = 0^T \quad (9)$$

Binární cyklické kódy mají jednoduché kódování informačních bitů. Metody pro kódování:

Nesystematické kódování

Pracuje na principu násobení generujícího mnohočlenu. Výsledkem je kódové slovo, u kterého není vidět, které prvky jsou informační a které jsou zabezpečující.

Systematické kódování

U této metody se využívá dělení generujícího mnohočlenu. Ve výsledku je vidět informační část nazývaná také někdy jako *Cyclic Redundancy Check* (CRC) a zabezpečující část.

Více informací v [10][17][18][19][20]. V této práci se využívá systematického kódování.

2.1.1 RS kód

Reed - Solomonovy kódy jsou korekční, cyklické, lineární kódy se symboly a patří do kategorie BCH kódu. RS kód je tvořen polynomy, kde koeficienty polynomů jsou prvky Galoisova pole $GF(2^m)$. Symboly jsou tvořeny m – bitovými posloupnostmi, kde m znamená libovolné kladné číslo s hodnotou větší než 2. Kód se běžně zadává parametry (n, k) . Parametr k stanoví, kolik symbolů vstupuje do kodéru a parametr n určí počet symbolů vystupujících z kodéru.

Pro $RS(n, k)$ podle [5][14] platí rovnice (10). Nejběžnější $RS(n, k)$ forma je dána rovnicí (11), kde t je maximální počet symbolů (13), které má schopnost opravit dekodér. Počet symbolů pro zabezpečení je $n - k = 2t$. Minimální kódová (Hammingova) vzdálenost se počítá podle vztahu (12).

$$0 < k < n < 2^m + 2 \quad (10)$$

$$(n, k) = (2^m - 1, 2^m - 1 - 2t) \quad (11)$$

$$d_{min} = n - k + 1 \quad (12)$$

$$t = \frac{d_{min} - 1}{2} = \frac{n - k}{2} \quad (13)$$

Před kódováním se musí určit generující polynom, který znázorňuje vztah (14), kde a, a^2, \dots, a^{2^t} jsou prvky Galoisova pole.

$$g(x) = (x - a)(x - a^2) \dots (x - a^{2^t}) = \prod_{i=1}^{2^t} (x - a^i) \quad (14)$$

2.1.2 RS – kódování

Zabezpečená zpráva $c(x)$ se vytvoří pomocí vztahu (15). Zpráva $i(x)$, která se má zakódovat, se vynásobí $x^{(n-k)}$ proto, aby se oddělily zabezpečující znaky od informačních znaků. Pro vytvoření zabezpečujících symbolů se vypočítá zbytek po (binárním) dělení $p(x)$ (16) nebo podle [5] jak lze vidět v rovnici (17), kde Rem znamená zbytek po binárním dělení. Zabezpečující symboly dosadíme do (15) nebo podle [5] vznikne rovnice (18).

$$c(x) = i(x) \cdot x^{(n-k)} + p(x) \quad (15)$$

$$p(x) = \frac{i(x) \cdot x^{(n-k)}}{g(x)} \quad (16)$$

$$Rem \left\{ \frac{i(x) \cdot x^{(n-k)}}{g(x)} \right\} = p(x) \quad (17)$$

$$c(x) = i(x) \cdot x^{(n-k)} + Rem \left\{ \frac{i(x) \cdot x^{(n-k)}}{g(x)} \right\} \quad (18)$$

2.1.3 RS – dekodování

U vyhledávání lokalizačního mnohočlenu RS kódu bylo vyvinuto několik algoritmů jako např.: Berlekamp – Masseyův (BM) algoritmus, Euklidův algoritmus, Peterson – Gorenstein – Zierlův algoritmus atd.

Obecně lze rozdělit dekódování do několika kroků:

- Stanovení syndromů (výpočet jestli nastala chyba nebo ne a pro určení lokalizačního mnohočlenu)
- Výpočet chybového polynomu (lokalizační mnohočlen), který nám pomůže v dalším kroku při nalezení chyb. Použije se BM algoritmus.
- Nalezení pozice chyb pomocí Chienova vyhledávání.
- Pomocí Forneyova algoritmu stanovit hodnoty chyb.
- Opravit chybu.

Výpočet syndromů

Při dekódování se jako první zjistí, zda nastala nebo nenastala chyba pomocí výpočtu syndromů. Předpokládáme chybu $e(x)$ v přijatém polynomu $r(x)$ (19).

$$r(x) = c(x) + e(x) \quad (19)$$

Za x se dosadí prvky GF dle (20), kde $j = 1, 2, \dots, 2t$. Dostáváme tak syndromy přijatých dat S_1, S_2, \dots, S_{2t} zapsané ve tvaru (21).

$$S_j = x \Rightarrow (a^j) \quad (20)$$

$$S_j = r(a^j) \quad (21)$$

Vypočítáme binární XOR a pokud všechny syndromy = 0, tak kódové slovo je bez chyby. Jinak se pokračuje výpočtem lokátoru chyb.

Lokátor chyb

Tento výpočet lokátoru chyb se liší několika algoritmy. Obecně lze lokalizační mnohočlen popsat podle [4][11] rovnicí (22), kde v znamená počet chyb, $L_0 = 1$ a ostatní koeficienty je nutno dopočítat pomocí algoritmu.

$$L(x) = L_v x^v + L_{v-1} x^{v-1} + \dots + L_1 x + L_0 \quad (22)$$

Výpočet lokátoru chyb pomocí BM algoritmu

U tohoto algoritmu se využívá postupná iterace. Je rozdělen do několika kroků podle [5]:

1) Nastavení počátečních podmínek:

- index iterace $i = 0$
- pomocná proměnná $l = 0$
- chybový mnohočlen $L^{(l)}(x) = 0$
- pomocný chybový mnohočlen $A^{(l)}(x) = 0$

2) Kontrola $i = 2t$, jestli jsme dosáhli konce iterace. Jestli ano, algoritmus je u konce.

3) Odhad chyby je popsán vztahem (23).

$$d^i = \sum_{n=0}^{l^i} L_n^{(i)} \cdot S_{i+1-n} \quad (23)$$

4) Kontrola $d^i = 0$, pokud ne, přejdeme na krok 6, jinak pokračujeme krokem 5.

5) Změníme pomocný chybový mnohočlen $A^{(i)}(x)$ (24) a dále pokračujeme na krok 12.

$$A^{(i)}(x) = x \cdot A^{(i)}(x) \quad (24)$$

6) Upravíme dočasný chybový mnohočlen $T^{(i)}(x)$ (25).

$$T^{(i)}(x) = L^{(i)}(x) - d^{(i)} \cdot x \cdot A^{(i)}(x) \quad (25)$$

7) Kontrola jestli $2l \leq i$, pokud ano, přejdeme na krok 9.

8) Obnovení chybového mnohočlenu (26) a vrátíme se na krok 5.

$$L^{(i)}(x) = T^{(i)}(x) \quad (26)$$

9) Změníme pomocný chybový mnohočlen $A^{(i)}(x)$ podle rovnice (27).

$$A^{(i)}(x) = \frac{L^{(i)}(x)}{d^{(i)}} \quad (27)$$

10) Obnovení chybového mnohočlenu (28).

$$L^{(i)}(x) = T^{(i)}(x) \quad (28)$$

11) Zvětšíme pomocnou proměnnou l (29).

$$l^{i+1} = i + 1 - l^i \quad (29)$$

12) Inkrementujeme index iterace i a vrátíme se do kroku 2.

Chienovo vyhledávání

Lokalizační mnohočlen je stanovený a v tomto kroku prověříme všechny prvky GF, jestli nejsou kořenem mnohočlenu. Dosadíme prvky GF do mnohočlenu podle vztahu (22) a dostáváme lokalizační mnohočlen ve tvaru (30), kde q je prvočíslo z $GF(q^m)$.

$$\begin{aligned} L(a^0) &= L_v(a^0)^v + L_{v-1}(a^0)^{v-1} + \dots + L_1 a^0 + L_0 \\ L(a) &= L_v(a)^v + L_{v-1}(a)^{v-1} + \dots + L_1 a + L_0 \\ &\vdots \\ L(a^{q^m-2}) &= L_v(a^{q^m-2})^v + L_{v-1}(a^{q^m-2})^{v-1} + \dots + L_1(a^{q^m-2}) + L_0 \end{aligned} \quad (30)$$

Pokud $L(x) = 0$ pro některý z prvků GF, je nalezena pozice chyby $P(x)$. Ale pokud nejsou nalezeny žádné kořeny ($L(x) \neq 0$ pro všechny prvky z GF), tak to znamená neodstranitelné chyby.

Forneyův algoritmus

Jestliže jsou nalezeny kořeny chyb, tak vypočítáme podle [5][11] polynom pro určení hodnot chyb $E(x)$ dle rovnice (31), kde $\text{mod } x^{2t}$ je operace, která zajišťuje zrušení všech členů mnohočlenu, kteří mají x^{2t} nebo vyšší. $S(x)$ představuje syndromový mnohočlen, který vytvoříme podle rovnice (32). Vypočítáme hodnotu chyb $M(x)$ podle rovnice (33), kde l = počet chyb $P(x)$, L' je derivace lokalizačního mnohočlenu.

$$E(x) = S(x) \cdot L(x) \pmod{x^{2t}} \quad (31)$$

$$S(x) = \sum_{j=0}^{2t-1} S_{j+1} \cdot x^j \quad (32)$$

$$M_l = \frac{E(P_l^{-1})}{L'(P_l^{-1})} \quad (33)$$

Oprava chyby

V tomto kroku už je známa pozice chyby $P(x)$ a hodnota chyby $M(x)$. Z toho se vytvoří chybový mnohočlen $e(x)$ a provede se operace binární XOR s přijatým mnohočlenem $r(x)$. Touto operací vznikne opravená zabezpečená zpráva $c(x)$, kterou lze zapsat ve tvaru (34).

$$c(x) = r(x) + e(x) \quad (34)$$

3 MATEMATICKÝ ZÁKLAD

3.1 Konečné těleso

Algebraická struktura, která je tvořena zbytky po dělení celých kladných čísel prvočíslem q , je nazývána konečné těleso Z_q . Konečné těleso Z_2 je v praxi nejvíce využíváno kvůli tomu, že je tvořené pomocí množiny $\{0, 1\}$. Algebraická operace součtu je nahrazena logickou operací součtu XOR podle [10][12]. Algebraické násobení je nahrazeno logickou operací součinu AND. Logické operace XOR a AND jsou vidět v Tab. 1.

Tab. 1. Tabulka pro XOR a AND v konečném tělese Z_2

XOR		\oplus	AND		\cdot
0	0	0	0	0	0
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	1

3.2 Galoisova pole

Rozšířením konečného tělesa Z_q o stupeň m se vytvoří Galoisovo těleso (q^m). Vznikne množina vektorů o m prvcích, kde každý prvek je z tělesa Z_q . Galoisovo těleso $GF(2^m)$ se vytvoří rozšířením tělesa Z_2 , které se používá pro binární kódy. Toto těleso lze vyjádřit pomocí zbytků (vytvářejícím mnohočlenem) po dělení mnohočlenu Z_2 určitým mnohočlenem, pro který platí:

- je nerozložitelný v uvažovaném okruhu mnohočlenů
- je stupně m
- dělení beze zbytku ($x^n - 1$) a žádný jiný dvojčlen nižšího stupně tohoto tvaru

Pomocí vytvářejícího mnohočlenu se vytvoří pole prvků (Galoisova pole). První prvek pole je nula, která musí být v Galoisovu poli. Ostatní prvky se vyjadřují pomocí exponentů a^x . Při vyjádření jednotlivých prvků pomocí mnohočlenu musí být řádu $< m$. Proto tedy pro prvky pole musí být použita operace modulo s vytvářejícím mnohočlenem [13][17][18][19]. V Tab. 2. jsou uvedené prvky $GF(2^3)$, který je tvořen vytvářejícím mnohočlenem $q(x)$ ve tvaru (35).

$$q(x) = 1 + x + x^3 \quad (35)$$

Tab. 2. Galoisovo pole generované polynomem $q(x) = 1 + x + x^3$

Exponenciální tvar	Polynomiální tvar	Binární tvar
0	0	000
a^0	1	100
a^1	a^1	010
a^2	a^2	001
a^3	$1 + a$	110
a^4	$a + a^2$	011
a^5	$1 + a + a^2$	111
a^6	$1 + a^2$	101

4 PROSTŘEDÍ MATHEMATICA

Společnost Wolfram Research, Inc. byla založena Stephenem Wolframem v roce 1987. Program Mathematica byl poprvé vydán v roce 1988. Byl jedinečný v tom, že jako jeden z prvních nabízel více sad úloh v jednom softwaru (např. numerické, algebraické, grafické úlohy).

Tento software má symbolické programování a tedy umožňuje do jednoho symbolu uložit data, různé grafy i celé programy. Mathematica je vhodná jak pro úplného nováčka ve „světě“ programování, tak i pro zkušeného programátora.

Software Mathematica má velmi přívětivou nápovědu.

Count

`Count`[*list*, *pattern*]
gives the number of elements in *list* that match *pattern*.

`Count`[*expr*, *pattern*, *levelspec*]
gives the total number of subexpressions matching *pattern* that appear at the levels in *expr* specified by *levelspec*.

MORE INFORMATION

EXAMPLES

Basic Examples (1)

Count how many times `b` occurs:

```
In[1]:= Count[{a, b, a, a, b, c, b}, b]
```

```
Out[1]= 3
```

Scope (3)

Generalizations & Extensions (1)

Applications (1)

SEE ALSO

[BinCounts](#) · [Cases](#) · [FreeQ](#) · [MemberQ](#) · [Select](#) · [Position](#) · [StringCount](#) · [Tally](#) · [Total](#) · [LeafCount](#)

Obr. 2. Nápověda v program Mathematica

Stačí, aby uživatel znal pouze anglický název funkce (např. počítat – `Count`), kterou si si uživatel může nechat někde přeložit a nápověda mu podle názvu vyhledá několik

alternativních možností. Každá funkce je detailně popsána i s několika příklady, které se dají zkopírovat a okamžitě použít. Na Obr. 2. lze vidět příklad, kde uživatel chce vědět, kolik b je v určitém zadaném poli. Dále obsahují i možnosti využití více funkcí najednou nebo další funkce s podobným využitím (vidíme na Obr. 2. jako SEE ALSO).

Mathematica také obsahuje mnoho druhů grafů jako např. 3D, ekonomické nebo statické grafy atd. Lze vytvářet i vlastní animace nebo zvukové záznamy.

Software má implementované funkce pro počítání matematických příkladů, ale i různé procedury jako je seřazení nebo vyhledávání. Data se zadávají různými způsoby, můžeme je uložit jako tabulku, string, pole, list atd.

Největší využití tohoto softwaru je v oborech Matematika, Fyzika a Inženýrství.

Pro více informací pro začátečníky k tomuto prostředí doporučuji literaturu [6].

II. PRAKTICKÁ ČÁST

5 PŘÍKLAD REED – SOLOMONOVA KÓDOVÁNÍ

Pro názornou ukázkou byl vybrán kód $RS(7, 3)$ pro kódování a dekódování, který využívá $GF(2^3)$ generované polynomem $q(x) = 1 + x + x^3$ (Tab. 2.). Znamená to, že v zabezpečené kódové zprávě $c(x)$ o délce 7 symbolů je nezabezpečená kódová zpráva $i(x)$ o 3 symbolech a 4 zabezpečující symboly $p(x)$.

5.1 RS – kódování

Nezabezpečenou kódovou zprávu uvažujeme ve tvaru $i = (111\ 010\ 001)$ a ve vztahu (36) existuje zpráva přeepsaná prvky GF. Podle vztahu (13) se vypočítá $t = 2$. Vytvoříme tedy podle vztahu (14) generující polynom $g(x)$ rovnici (37). Dále se pokračuje podle vztahu (16), kde se vypočítá zbytek po dělení $p(x)$ pomocí binární operace XOR (38).

$$i(x) = a^5 + ax + a^2x^2 \quad (36)$$

$$\begin{aligned} g(x) &= (x - a)(x - a^2) \dots (x - a^{2^t}) \\ g(x) &= (x - a)(x - a^2)(x - a^3)(x - a^4) \\ g(x) &= a^3 + ax + x^2 + a^3x^3 + x^4 \end{aligned} \quad (37)$$

$$\begin{array}{r} a^2x^6 + ax^5 + a^5x^4 : x^4 + a^3x^3 + x^2 + ax + a^3 = a^2x^2 + a^6x + a^5 \\ -(a^2x^6 + a^5x^5 + a^2x^4 + a^3x^3 + a^5x^2) \\ \hline 0 + a^6x^5 + a^3x^4 + a^3x^3 + a^5x^2 \\ -(a^6x^5 + a^2x^4 + a^6x^3 + a^0x^2 + a^2x) \\ \hline 0 + a^5x^4 + a^4x^3 + a^4x^2 + a^2x \\ -(a^5x^4 + ax^3 + a^5x^2 + a^6x + a) \\ \hline 0 + a^2x^3 + a^0x^2 + a^0x + a \\ \hline p(x) = a^2x^3 + a^0x^2 + a^0x + a \end{array} \quad (38)$$

Podle (15) vznikne zabezpečená zpráva ve tvaru (39) a zprávu převedenou do binárního tvaru znázorňuje zápis (40).

$$c(x) = a^2x^6 + ax^5 + a^5x^4 + a^2x^3 + a^0x^2 + a^0x + a \quad (39)$$

$$c = (001\ 010\ 111\ 001\ 100\ 100\ 010) \quad (40)$$

5.2 RS – dekódování

Pro dekódování zprávy se použije zabezpečená zpráva z minulé podkapitoly, ale vytvoří se zde 2 chyby.

Zabezpečená zpráva z předchozí podkapitoly a vytvořená chyba jsou popsány vztahy (41) a (42). Zprávu k dekódování $r(x)$ pak dostáváme pomocí (19) přes binární operaci XOR. Výsledkem je tedy kódové slovo zapsané ve tvaru (43) respektive (44).

$$c(x) = a^2x^6 + ax^5 + a^5x^4 + a^2x^3 + a^0x^2 + a^0x + a \quad (41)$$

$$e(x) = a^2x^5 + a^4x^3 \quad (42)$$

$$r(x) = c(x) + e(x) = a^2x^6 + a^4x^5 + a^5x^4 + ax^3 + a^0x^2 + a^0x + a \quad (43)$$

$$r(x) = (001\ 011\ 111\ 010\ 100\ 100\ 010) \quad (44)$$

Výpočet syndromů

Nyní vypočítáme syndromy přes XOR, jestli je chyba v zabezpečené zprávě, pomocí (20), (21) a vznikne vztah (45).

$$\begin{aligned} S_1(a) &= r(a) = a + a^2 + a^2 + a^4 + a^2 + a + a = 0 \\ S_2(a^2) &= r(a^2) = a^0 + a^0 + a^6 + a^0 + a^4 + a^2 + a = a^2 \\ S_3(a^3) &= r(a^3) = a^6 + a^5 + a^3 + a^3 + a^6 + a^3 + a = a^4 \\ S_4(a^4) &= r(a^4) = a^5 + a^3 + a^0 + a^6 + a + a^4 + a = a^4 \end{aligned} \quad (45)$$

Lokalizační mnohočlen

Všechny syndromy nevyšli nulové, proto se bude pokračovat k určení chybě pomocí BM algoritmu, který je uveden v podkapitole 2.1.3.

Jako první se nastaví počáteční podmínky podle (46). Zkontroluje se, jestli není konec iterace $i = 2t \Rightarrow 0 \neq 4$ a proto se může pokračovat odhadem chyby (47).

$$i = 0; l = 0; A(x) = 1; L(x) = L_0 = 1 \quad (46)$$

$$d = \sum_{n=0}^l L_n \cdot S_{i+1-n} = L_0 \cdot S_1 = 1 \cdot 0 = 0 \quad (47)$$

Dále se provede kontrola, jestli se $d = 0$ a jelikož je to pravda, změní se pomocný chybový mnohočlen $A(x)$ podle (48). Zvýšíme iteraci $i = i + 1 = 1$ a pro přehlednost se vypíší proměnné (49).

$$A(x) = A(x) \cdot x = 1 \cdot x = x \quad (48)$$

$$i = 1; l = 0; A(x) = x; L(x) = 1 \quad (49)$$

Následuje test iterace $l \neq 4$, pokračuje se odhadem chyby (50). V tomto případě se už $d \neq 0$ a aktualizuje se dočasný mnohočlen $T(x)$ (51). Jelikož v binárním tvaru se píše -1 a $+1$ stejně, může se v těchto případech $T(x)$ napsat jako ve vztahu (52). Zkontroluje se jestli $2l \leq i \Rightarrow$ změna $A(x)$ podle (53) a aktualizace dočasného mnohočlenu $L(x)$ (54). Zvýší se proměnná $l = i + 1 - l = 1 + 1 - 0 = 2$ a zvýší se index inkrementace $i = i + 1 = 2$.

$$d = \sum_{n=0}^l L_n \cdot S_{i+1-n} = L_0 \cdot S_2 = 1 \cdot a^2 = a^2 \quad (50)$$

$$T(x) = L(x) - d \cdot x \cdot A(x) = 1 - a^2 \cdot x \cdot x = 1 - a^2 x^2 \quad (51)$$

$$T(x) = 1 + a^2 x^2 \quad (52)$$

$$A(x) = \frac{L(x)}{d} = \frac{1}{a^2} = a^5 \quad (53)$$

$$L(x) = T(x) = 1 + a^2 x^2 \quad (54)$$

$$i = 2; l = 2; A(x) = a^5; L(x) = 1 + a^2 x^2 \quad (55)$$

Po vypísání proměnných se provede kontrola konce iterace $2 \neq 4$ a pokračuje se odhadem chyby (56). Proměnná $d \neq 0$ a pokračuje se tedy rovnicí (57). $2l \leq i$ neplatí, proto následuje (58) a (59). Zvýší se iterace $i = i + 1$ a vypíšou se aktuální podmínky (60).

$$d = \sum_{n=0}^l L_n \cdot S_{i+1-n} = L_0 \cdot S_3 + L_1 \cdot S_2 + L_2 \cdot S_1 = 1 \cdot a^4 + 0 \cdot a^4 + a^2 \cdot 0 = a^4 \quad (56)$$

$$T(x) = L(x) - d \cdot x \cdot A(x) = 1 + a^2 x^2 - a^4 \cdot x \cdot a^5 = 1 + a^2 x + a^2 x^2 \quad (57)$$

$$L(x) = T(x) = 1 + a^2 x + a^2 x^2 \quad (58)$$

$$A(x) = A(x) \cdot x = a^5 x \quad (59)$$

$$i = 3; l = 2; A(x) = a^5 x; L(x) = 1 + a^2 x + a^2 x^2 \quad (60)$$

Nastane kontrola podmínky $3 \neq 4 \Rightarrow$ odhad chyby vztahem (61). Proměnná $d \neq 0$ a tak se vypočítá vztah (62). Podmínka $2l \leq i$ neplatí a vykoná se tedy (63) a (64).

$$d = \sum_{n=0}^l L_n \cdot S_{i+1-n} = L_0 \cdot S_4 + L_1 \cdot S_3 + L_2 \cdot S_2 = 1 \cdot a^4 + a^2 \cdot a^4 + a^2 \cdot a^2 = a^6 \quad (61)$$

$$T(x) = L(x) - d \cdot x \cdot A(x) = 1 + a^2x + a^2x^2 - a^6 \cdot x \cdot a^5x = 1 + a^2x + a^2x^2 - a^4x^2$$

$$T(x) = 1 + a^2x + ax^2 \quad (62)$$

$$L(x) = T(x) = 1 + a^2x + ax^2 \quad (63)$$

$$A(x) = A(x) \cdot x = a^5x^2 \quad (64)$$

Zvýší se inkrementace $i = i + 1$, a jelikož $4 = 4$, dosáhlo se tedy ke konečnému lokalizačnímu mnohočlenu popsaneho vztahem (65).

$$L(x) = 1 + a^2x + ax^2 \quad (65)$$

Chienovo vyhledávání

Stanovil se lokalizační mnohočlen, který se následně využije v Chienovém vyhledávání. Podle (22) se dosadí do lokalizačního mnohočlenu a jak je vidět ve tvaru (66), vyjdou 2 pozice chyb.

$$\begin{aligned} L(a^0) &= 1 + a^2 + a = a^5 \\ L(a^1) &= 1 + a^3 + a^3 = a^0 \\ L(a^2) &= 1 + a^4 + a^5 = 0 \Leftarrow \\ L(a^3) &= 1 + a^5 + a^0 = a^5 \\ L(a^4) &= 1 + a^6 + a^2 = 0 \Leftarrow \\ L(a^5) &= 1 + a^0 + a^4 = a^4 \\ L(a^6) &= 1 + a + a^6 = a^4 \end{aligned} \quad (66)$$

Pozice, které jsou nalezeny, se musí invertovat podle (67) a (68).

$$P_1 = (a^2)^{-1} = \frac{a^0}{a^2} = a^5 \quad (67)$$

$$P_2 = (a^4)^{-1} = \frac{a^0}{a^4} = a^3 \quad (68)$$

Forneyův algoritmus

Nyní se pokračuje pomocí Forneyova algoritmu. Vypočítá se syndromový polynom $S(x)$ podle (32) a dosadí se do polynomu pro určení hodnoty chyb $E(x)$ (31) a vznikne (69). Rovnice (70) a (71) pak určují hodnoty chyb podle (33).

$$\begin{aligned} E(x) &= S(x) \cdot L(x) \pmod{x^{2t}} \\ E(x) &= (1 + a^2x + ax^2) \cdot (a^2x^2 + a^4x^3 + a^4x^4) \pmod{x^4} \\ E(x) &= (a^2x^2 + a^4x^3 + a^4x^3 + a^3x^4 + a^4x^4 + a^6x^4 + a^5x^5 + a^6x^5 + a^5x^6) \pmod{x^4} \\ E(x) &= a^2x^2 \end{aligned} \quad (69)$$

$$M_1 = \frac{E(P_1^{-1})}{(1 - P_2P_1^{-1})} = \frac{a^2(a^2)^2}{a^0 + a^3a^2} = \frac{a^6}{a^4} = a^2 \quad (70)$$

$$M_2 = \frac{E(P_2^{-1})}{(1 - P_1P_2^{-1})} = \frac{a^2(a^4)^2}{a^0 + a^5a^4} = \frac{a^3}{a^6} = a^4 \quad (71)$$

Hodnota chyb a jejich pozice jsou známy a může se tedy vytvořit chybový mnohočlen $e(x)$ (72).

$$e(x) = M_1P_1 + M_2P_2 = a^2x^5 + a^4x^3 \quad (72)$$

Chybový mnohočlen $e(x)$ ve tvaru (72) se shoduje s vytvořenou chybou na začátku (42). Detekce chyby proběhla tedy úspěšně.

Oprava chyb

Podle (34) se vytvoří mnohočlen zabezpečené zprávy s vypočítanou chybou (73) a provede se operace XOR. Vznikne opravená zabezpečená zpráva, kterou můžeme zapsat ve tvaru (74) respektive v binární podobě (75).

$$c(x) = a^2x^6 + a^2x^5 + a^4x^5 + a^5x^4 + ax^3 + a^4x^3 + a^0x^2 + a^0x + a \quad (73)$$

$$c(x) = a^2x^6 + ax^5 + a^5x^4 + a^2x^3 + a^0x^2 + a^0x + a \quad (74)$$

$$c(x) = (001\ 010\ 111\ 001\ 100\ 100\ 010) \quad (75)$$

Další příklady ke kódování a dekodování jsou v příloze.

6 RS V PROSTŘEDÍ MATHEMATICA

Na základě prostudované teorie kódování a dekodování problematiky Reed Solomonových kódů byl vytvořen program v prostředí Mathematica. Teorie RS a prostředí Mathematica je probraná v předchozích kapitolách.

Program Mathematica byla vybraná pro její vestavěné funkce, počítání s polynomy, různé procedury jako např. seřazení a obsahuje i funkci pro Galoisova pole, která ale nebyla využita. RS kód může být naprogramován v binární kombinaci nebo polynomiální, ale kvůli vybranému prostředí byla zvolena polynomiální kombinace. Největším problémem byl přístup k jednotlivým členům polynomu (mnohočlenu) kódového slova. Pro tyto účely byla využita funkce *Length*. Tato funkce vypíše délku zadaného polynomu, kde délka znamená počet členů polynomu. Na Obr. 3. je znázorněn příklad při použití polynomu ve tvaru (76). Vztah (77) vypíše délku zadaného polynomu (délka = 2), takže polynom je rozdělen do 2 členů. První člen a^3 se dá rozdělit podle vztahu (79) na další 2 členy a a 3 , ale ve vztahu (82) už ne. U druhého členu nastává jiná situace, protože obsahuje více znaků. Vztah (84) rozděluje na členy a^4 a x^2 a podle vztahu (87) lze ještě rozdělit a^4 na znaky a a 4 , jak je znázorněno v (88) a (89).

Protože Mathematica dokáže ukládat do jednoho symbolu celé funkce nebo grafy, vzniká další problém při vyhledávání určitého symbolu.

Např.: Když přes funkci *If*, která porovnává 2 hodnoty, se vyhledává symbol a , tak podmínka může vrátit 3 hodnoty (TRUE, FALSE, NULL). Pokud byla porovnávaná hodnota jakákoliv kromě a , tak se vrátí hodnota *NULL* a výsledek je tedy stále neznámý. Proto se musí vnořovat více funkcí do jedné a vzniká tak zbytečně dlouhý kód, kde větší část tvoří podmínky.

$$\begin{aligned} \text{In}[1] := & \mathbf{c = a^3 + a^4 x^2} \\ \text{Out}[1] = & a^3 + a^4 x^2 \end{aligned} \quad (76)$$

$$\begin{aligned} \text{In}[2] := & \mathbf{Length[c]} \\ \text{Out}[2] = & 2 \end{aligned} \quad (77)$$

$$\begin{aligned} \text{In}[3] := & \mathbf{c[[1]]} \\ \text{Out}[3] = & a^3 \end{aligned} \quad (78)$$

$$\begin{aligned} \text{In}[4] := & \mathbf{Length[c[[1]]]} \\ \text{Out}[4] = & 2 \end{aligned} \quad (79)$$

$$\begin{aligned} \text{In}[5] := & \mathbf{c[[1, 1]]} \\ \text{Out}[5] = & a \end{aligned} \quad (80)$$

$$\begin{aligned} \text{In}[6] := & \mathbf{c[[1, 2]]} \\ \text{Out}[6] = & 3 \end{aligned} \quad (81)$$

$$\begin{aligned} \text{In}[7] := & \mathbf{Length[c[[1, 1]]]} \\ \text{Out}[7] = & 0 \end{aligned} \quad (82)$$

$$\begin{aligned} \text{In}[8] := & \mathbf{c[[2]]} \\ \text{Out}[8] = & a^4 x^2 \end{aligned} \quad (83)$$

$$\begin{aligned} \text{In}[9] := & \mathbf{Length[c[[2]]]} \\ \text{Out}[9] = & 2 \end{aligned} \quad (84)$$

$$\begin{aligned} \text{In}[10] := & \mathbf{c[[2, 1]]} \\ \text{Out}[10] = & a^4 \end{aligned} \quad (85)$$

$$\begin{aligned} \text{In}[11] := & \mathbf{c[[2, 2]]} \\ \text{Out}[11] = & x^2 \end{aligned} \quad (86)$$

$$\begin{aligned} \text{In}[12] := & \mathbf{Length[c[[2, 1]]]} \\ \text{Out}[12] = & 2 \end{aligned} \quad (87)$$

$$\begin{aligned} \text{In}[13] := & \mathbf{c[[2, 1, 1]]} \\ \text{Out}[13] = & a \end{aligned} \quad (88)$$

$$\begin{aligned} \text{In}[14] := & \mathbf{c[[2, 1, 2]]} \\ \text{Out}[14] = & 4 \end{aligned} \quad (89)$$

Obr. 3. Popis funkce Length

6.1 Struktura programu

Grafický vzhled je konstruován pomocí funkce Manipulate. Uvnitř funkce Manipulate jsou volány funkce, které byly definovány za účelem kódování a dekódování a jejich výsledky jsou znázorněny na Obr. 4.

```

Generující polynom
 $a^2 + ax + x^2 + a^3 x^3 + x^4$ 
Zakódovaná zpráva
 $a + x + x^2 + a^2 x^3 + a^5 x^4 + ax^5 + a^2 x^6$ 
Zpráva k dekódování
 $a + x + x^2 + a^2 x^3 + a^5 x^4 + ax^5 + a^4 x^6$ 
Hodnoty Syndromů
 $S_1: a^0$ 
 $S_2: a^6$ 
 $S_3: a^5$ 
 $S_4: a^4$ 
BM algoritmus
 $1 + a^6 x$ 
Pozice chyby:
 $P_1: a^6$ 
Hodnota chyby:
 $M_1: a$ 
Opravená zpráva
 $a + x + x^2 + a^2 x^3 + a^5 x^4 + ax^5 + a^2 x^6$ 

```

Obr. 4. Zobrazení výsledků přes Manipulate

Program obsahuje funkce pro knihovnu – převod binárního na polynomiální zobrazení a naopak. Dále funkce pro podmínku velikosti zprávy pro kódování a dekódování. A nakonec jsou funkce rozdělené podle kroku, co se má zobrazit (Např.: Chienovo vyhledávání, Syndromy, Lokalizační mnohočlen, Generující polynom, atd.).

Na Obr. 5. je vyobrazena funkce pro vytváření generujícího mnohočlenu.

```
(*Vytvoření generujícího polynomu*)
fgenpol[x1_] := Module[{g = x1}, konec = Expand[(x + a) (x + a^2) (x + a^3) (x + a^4)];
  For[i = 1, i ≤ Length[konec], i++, If[NumberQ[konec[[i, 1]]] == True, konec[[i, 1]] = 0];
  g = Collect[konec, x]; g[[1]] = ToExpression[ftext[g[[1]]]];
  For[j = 1, j ≤ Length[g] - 2, j++,
    For[i = 1, i ≤ Length[g[[j + 1, 1]]], i++, g[[j + 1, 1, i]] = ftext[g[[j + 1, 1, i]]]];
  k = 0; For[j = 1, j ≤ Length[g] - 2, j++,
    (For[i = 1, i ≤ Length[g[[j + 1, 1]]], i++, k = BitXor[ToExpression[g[[j + 1, 1, i]]], k]);
    g[[j + 1, 1]] = k; k = 0];
  For[i = 1, i ≤ Length[g], i++,
    If[Length[g[[i]]] > 0, If[NumberQ[g[[i, 1]]] == True, g[[i, 1]] = ftextzpetg[g[[i, 1]]],
      If[NumberQ[g[[i]]] == True, g[[i]] = ftextzpetg[g[[i]]]]]; g];
```

Obr. 5. Funkce pro generující mnohočlen

Jako první se vytvoří podle vztahu (16) generující polynom. Mathematica stejné členy jako např. ax spojí dohromady a vznikne $2ax$, kdy při binární operaci XOR je výsledek 0, takže člen vypadne. Přes funkci `Collect` se členy rozdělí podle stupně proměnné x , což lze demonstrovat ve tvaru (90).

$$a^3 + (a + a^4 + a^5)x + (a^2 + a^4 + a^6)x^2 + (a + a^2 + a^5)x^3 \quad (90)$$

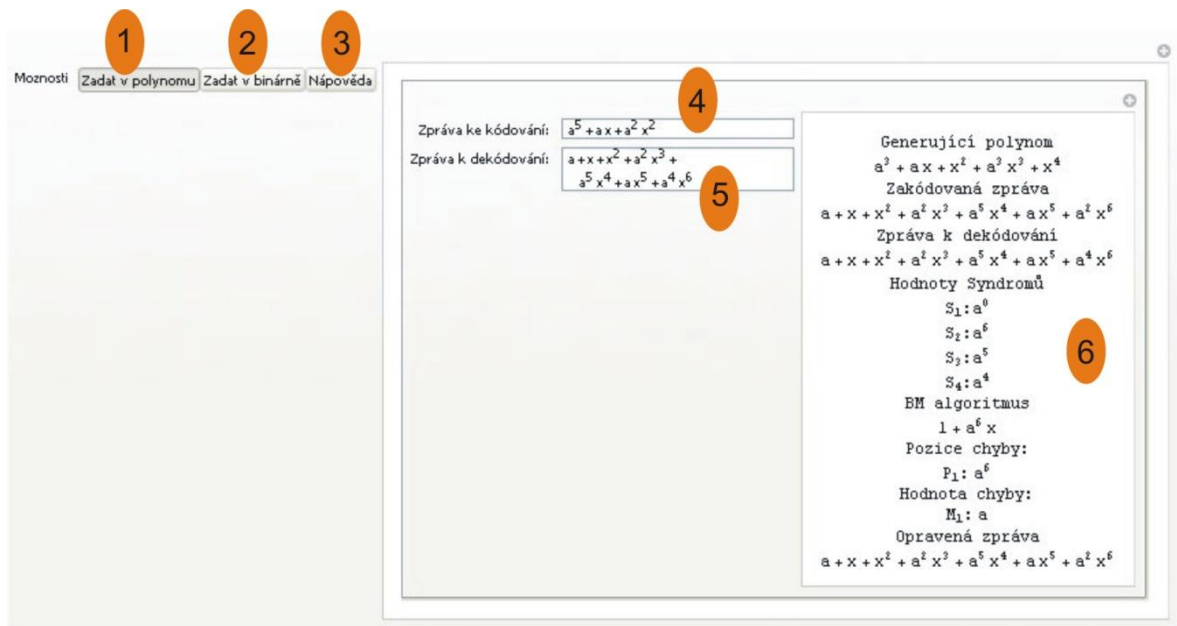
A poté se za pomoci knihovny převede polynomiální tvar na binární a použije se operace XOR. Na konci se převedou členy zpět na polynomiální a dostáváme generující mnohočlen ve tvaru (91).

$$a^3 + x + a^2x^2 + ax^3 \quad (91)$$

6.2 Popis programu

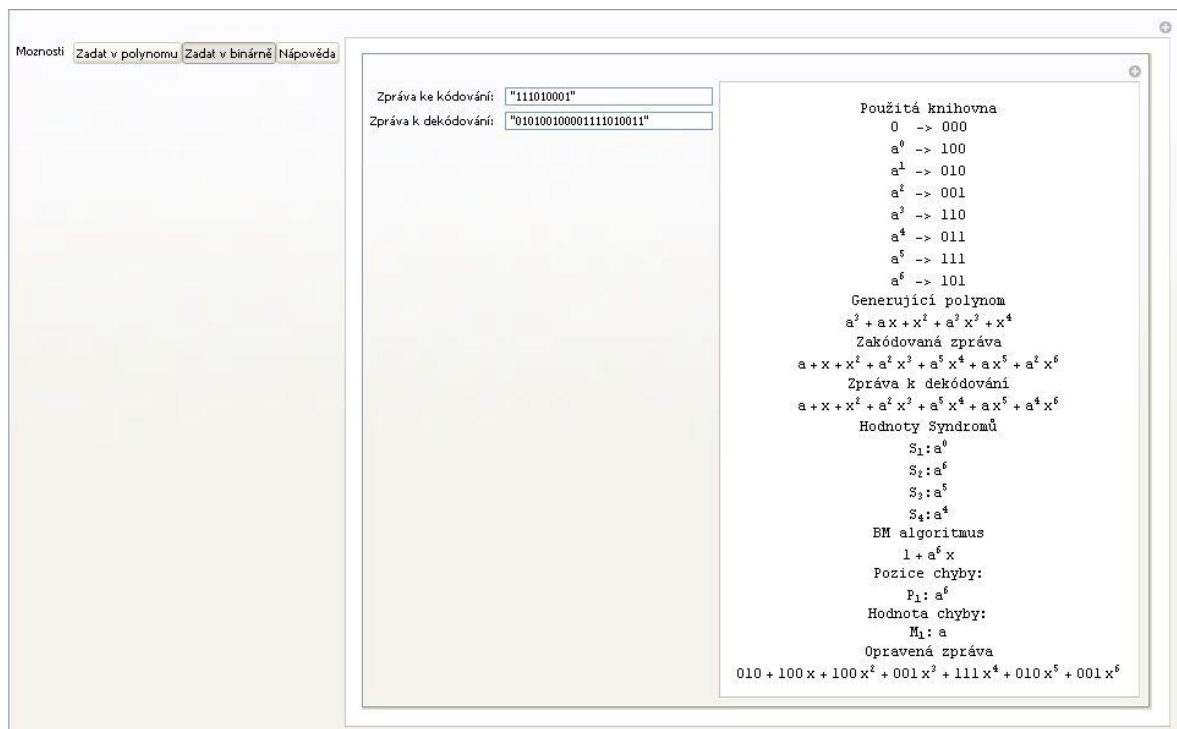
Na Obr. 6. je vyobrazen výstup programu s následujícími významy jednotlivých bodů:

1. Možnost zadávání v polynomiálním tvaru
2. Možnost zadávání v binárním tvaru
3. Nápověda
4. Textový panel pro zadání zprávy ke kódování
5. Textový panel pro zadání zprávy k dekódování
6. Zobrazení výstupu



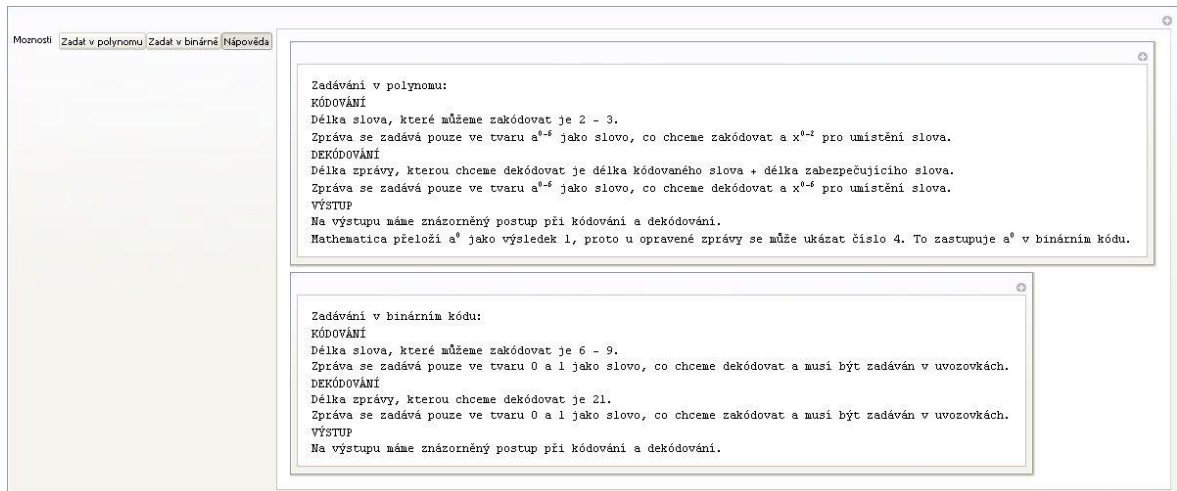
Obr. 6. Zadávání v polynomiálním tvaru

V bodech 4 a 5 se zadávají zprávy pro kódování a dekódování na Obr. 6. ve tvaru polynomiálním i se členy x a na Obr. 7. ve tvaru binárním do uvozovek (to znamená, že se zadá jako text). Mezivýsledky jsou zobrazeny v bodě 6.



Obr. 7. Zadávání v binárním tvaru

Na Obr. 8. je zobrazena podrobná nápověda, která lze vidět při otevření programu.



Možnosti Zadat v polynomu Zadat v binárně nápověda

Zadávání v polynomu:

KÓDOVÁNÍ
Délka slova, které můžeme zakódovat je 2 - 3.
Zpráva se zadává pouze ve tvaru a^{b-c} jako slovo, co chceme zakódovat a x^{b-c} pro umístění slova.

DEKÓDOVÁNÍ
Délka zprávy, kterou chceme dekodovat je délka kódovaného slova + délka zabezpečujícího slova.
Zpráva se zadává pouze ve tvaru a^{b-c} jako slovo, co chceme dekodovat a x^{b-c} pro umístění slova.

VÝSTUP
Na výstupu máme znázorněný postup při kódování a dekodování.
Mathematica přeloží a^b jako výsledek 1, proto u opravené zprávy se může ukázat číslo 4. To zastupuje a^b v binárním kódu.

Zadávání v binárním kódu:

KÓDOVÁNÍ
Délka slova, které můžeme zakódovat je 6 - 9.
Zpráva se zadává pouze ve tvaru 0 a 1 jako slovo, co chceme dekodovat a musí být zadán v uvozovkách.

DEKÓDOVÁNÍ
Délka zprávy, kterou chceme dekodovat je 21.
Zpráva se zadává pouze ve tvaru 0 a 1 jako slovo, co chceme zakódovat a musí být zadán v uvozovkách.

VÝSTUP
Na výstupu máme znázorněný postup při kódování a dekodování.

Obr. 8. Nápověda

ZÁVĚR

Hlavní cíl této práce bylo zpracovat literární rešerši na téma bezpečnostní kódy, zejména oblast RS kódů. Vytvořit simulaci k RS kódu v prostředí Mathematica s kompletními postupy kódování a dekódování.

V teoretické části byly popsány dřívější práce, které vznikly na téma RS kódy nebo BCH kódy. Dále tato část seznamuje s bezpečnostními kódy, jejich rozdělením a hlavní pozornost je věnována oblasti cyklických kódů, zejména pak kódování a dekódování RS kódů. Je zde také nastíněna problematika konečných těles a Galoisových polí a to zejména pole $GF(2^3)$, které se hojně využívá v praktické části. Poslední podkapitola této části popisuje prostředí, ve kterém je naprogramovaná simulace kódování a dekódování $RS(7, 3)$ kódu.

Praktická část obsahuje názorný příklad kódování a dekódování kódu $RS(7, 3)$. Pro účely detekce chyby bylo vytvořeno kódové slovo se dvěma chybami. Je zde podrobně popsána také vytvořená simulace algoritmu RS kódu v prostředí Mathematica.

Prostředí Mathematica, která je vhodná pro práci s polynomiálními tvary, bylo k vytvoření této simulace občas nevhodné. Kvůli automatickému sčítání a zarovnání se některé operace museli řešit pomocí mnoha podmínek vnořených do sebe. Při vytváření simulace pomocí binární kombinace problémy nemusely nastat.

Simulace je vytvořená pro kódování a dekódování kódu $RS(7, 3)$, kde na výstupu jsou vidět výsledky postupů, které jsou nutné pro zakódování zprávy nebo opravu chyb.

Teoretická část i simulace by mohli sloužit jako základ pro zpracování informací a pokračování v simulaci pro více algoritmů dekódování u RS kódů.

ZÁVĚR V ANGLIČTINĚ

The main object of this work was process retrieval on the theme safety codes, especially region RS codes. Create simulation to RS codes in environment Mathematica with complete procedures coding and decoding.

In theoretical part was described previously works, which created on theme RS codes or BCH codes. Next this part introduces with safety codes, their distribution and main attention is paid region cyclic codes, particularly then coding and decoding RS codes. Is here also outlined issues final bodies and Galois field and it particularly field $GF(2^3)$, which abundantly uses with practical part. The last subchapter this par described environment, in which is programmed simulation coding and decoding $RS(7, 3)$ code.

The practical part contains graphic example coding and decoding code $RS(7, 3)$. For purposes detection error was created code word with two errors. Is here detail described also created simulation algorithm RS code in environment Mathematica.

Environment Mathematica, which is suitable for work with polynomial form, was to create this simulation sometimes inappropriate. For automatic addition a justification some operation had handle with many conditions nested together. At creating simulation with binary combination problems did not happen.

Simulation is created for coding and decoding code $RS(7, 3)$. On output are see results procedures, which are necessary for encoding message or repair errors.

Theoretical part and simulation would by serve like base for treatment information and continue in simulation for more algorithms decoding at RS codes.

SEZNAM POUŽITÉ LITERATURY

- [1] ADAMEC, Milan. *Cyklické bezpečnostní kódy*. Zlín, 2011. Bakalářská práce. FAI Zlín. Vedoucí práce Bronislav Chramcov.
- [2] ADÁMEK, Jiří. *Kódování*. Praha: STNL, 1989. 191 s.
- [3] CLARK, C. K. P. *Reed – Solomon error correction* [online]. BBC, 2002 [cit. 2012-01-30]. Dostupné z: <http://downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP031.pdf>
- [4] ČÍKA, P., KOTON, J., KŘÍVÁNEK, V. *Samoopravné Reed-Solomonovy kódy 2006* [online]. [cit. 2012-04-14]. Dostupné z: <http://access.feld.cvut.cz/view.php?cisloclanku=2006080002>
- [5] HANZO, L, T LIEW a B YEAP. *Turbo coding, turbo equalisation and space-time coding: for transmission over fading channels*. Chichester: John Wiley, 2002, 748 s. ISBN 04-708-4726-3.
- [6] CHRAMCOV, Bronislav. *Základy práce v prostředí Mathematica*. Zlín: Univerzita Tomáše Bati, 2006, 122 s. ISBN 80-7318-510-5.
- [7] JANÍK, Petr. *Bezpečnostní kódy a algoritmy jejich zabezpečení (rešerše)*. Zlín, 2007. Bakalářská práce. FAI Zlín. Vedoucí práce Miloš Krčmář.
- [8] KETTNER, Jakub. *Kódování - cyklické kódy*. Zlín, 2008. Bakalářská práce. FAI Zlín. Vedoucí práce Miloš Krčmář.
- [9] MACWILLIAMS, Florence Jessie a Neil James Alexander SLOANE. *The theory of error-correcting codes*. Amsterdam: North-Holland, 1978, 762 s. ISBN 04-448-5193-3.
- [10] MAREŠ, Jan. *Teorie kódování*. Praha: ČVUT, 2008. ISBN 978-80-01-04203-8.
- [11] MOON, Todd K. *Error correction coding: mathematical methods and algorithms*. Hoboken: John Wiley, 2004, 756 s. ISBN 04-716-4800-0.
- [12] MORELOS-ZARAGOZA, Robert H. *The art of error correcting coding*. Chichester: John Wiley, 2002, 221 s. ISBN 04-714-9581-6.
- [13] NĚMEC, Karel. *Protichybové kódové zabezpečení s Bose-Chaudhury-Hocquenhemovými kódy*. [online]. 2005, [cit. 2012-04-14]. Dostupné z: <http://www.elektrorevue.cz/clanky/05015/index.htm>

- [14] SKLAR, B. *Reed-Solomon Codes* [online]. [cit. 2012-04-14]. Dostupné z: http://www.facweb.iitkgp.ernet.in/~pallab/mob_com/art_sklar7_reed-solomon.pdf
- [15] ŠICNER, Jiří. *Srovnání algoritmů dekódování Reed – Solomonova kódu*. Brno, 2011. Diplomová práce. VUT Brno. Vedoucí práce Pavel Šilhavý.
- [16] VEČERKA, Jiří. *Kódování - BCH kódy*. Zlín, 2009. Bakalářská práce. FAI Zlín. Vedoucí práce Miloš Krčmář.
- [17] VLČEK, Karel. *Kompresa a kódová zabezpečení v multimediálních komunikacích*. Praha: BEN – technická literatura, 2004, 258 s. ISBN 80-730-0134-9.
- [18] VLČEK, Karel. *Teorie informace a kódování*. Ostrava: VŠB, 1998. ISBN 80-7078-494-6.
- [19] VLČEK, Karel. *Teorie Informace, kódování a kryptografie*. Ostrava: VŠB, 1999. ISBN 80-70-78-614-0.
- [20] *Výuka na FAI UTB ve Zlíně: Kurz: Základy informatiky* [online]. [cit. 2012-04-14]. Dostupné z: <http://vyuka.fai.utb.cz>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

BCH Bose – Chaudhuri – Hocquenghem

CRC Cyclic Redundancy Check

GF Galois Field

RS Reed – Solomon

d_{min} minimální kódová vzdálenost

G generující matice

H kontrolní matice

k počet vstupních symbolů do kodéru

M_l hodnota chyby

n počet výstupních symbolů z kodéru

P_l pozice chyby

Rem zbytek po binárním dělení

S_j syndromy přijatých dat

t korekční schopnost kódu

Z_q konečné těleso

$A(x)$ pomocný chybový mnohočlen

$c(x)$ zabezpečená kódová zpráva

$e(x)$ chybový mnohočlen

$E(x)$ mnohočlen pro určení hodnoty chyb

$g(x)$ generující mnohočlen

$i(x)$ nezabezpečená zpráva

$p(x)$ zbytkový mnohočlen

$r(x)$ přijatý mnohočlen

$S(x)$ syndromový mnohočlen

$T(x)$ dočasný chybový mnohočlen

$L(x)$ lokalizační mnohočlen

SEZNAM OBRÁZKŮ

Obr. 1. Rozdělení bezpečnostních kódů	12
Obr. 2. Návod v programu Mathematica.....	22
Obr. 3. Popis funkce Length	31
Obr. 4. Zobrazení výsledků přes Manipulate.....	32
Obr. 5. Funkce pro generující mnohočlen	33
Obr. 6. Zadávání v polynomiálním tvaru.....	34
Obr. 7. Zadávání v binárním tvaru.....	34
Obr. 8. Návod	35

SEZNAM TABULEK

Tab. 1. Tabulka pro XOR a AND v konečném tělese Z_2	20
Tab. 2. Galoisovo pole generované polynomem $q(x) = 1 + x + x^3$	21

SEZNAM PŘÍLOH

P I Př. kódování

P II Př. dekodování

PŘÍLOHA P I: PŘ. KÓDOVÁNÍ

Použijeme prvky, které jsou v Tab. 2.

$$g(x) = a^3 + ax + x^2 + a^3x^3 + x^4 \quad (92)$$

Zpráva ke kódování (93).

$$i(x) = a^4 + a^6x + a^3x^2 \quad (93)$$

Vypočítá se zbytek po dělení $p(x)$ pomocí binární operace XOR (94).

$$\begin{array}{r} a^3x^6 + a^6x^5 + a^4x^4 : x^4 + a^3x^3 + x^2 + ax + a^3 = a^3x^2 + a^6 \\ -(a^3x^6 + a^6x^5 + a^3x^4 + a^4x^3 + a^6x^2) \\ \hline 0 + 0 + a^6x^4 + a^4x^3 + a^6x^2 \\ -(a^6x^4 + a^2x^3 + a^6x^2 + a^0x + a^2) \\ \hline 0 + ax^3 + 0 + a^0x + a^2 \end{array} \quad (94)$$

$$p(x) = ax^3 + a^0x + a^2$$

Zabezpečená kódová zpráva $c(x)$ (95).

$$c(x) = a^3x^6 + a^6x^5 + a^4x^4 + ax^3 + a^0x + a^2 \quad (95)$$

Zpráva převedená do binárního tvaru (96).

$$c = (110\ 101\ 011\ 010\ 000\ 100\ 001) \quad (96)$$

PŘÍLOHA P II: PŘ. DEKÓDOVÁNÍ

Použijeme kódovou zprávu z přílohy I.

$$c(x) = a^3x^6 + a^6x^5 + a^4x^4 + ax^3 + a^0x + a^2 \quad (97)$$

Kódová zpráva bez chyby:

$$e(x) = 0 \quad (98)$$

Zpráva k dekodování $r(x)$ (99).

$$r(x) = c(x) + e(x) = a^3x^6 + a^6x^5 + a^4x^4 + ax^3 + a^0x + a^2 \quad (99)$$

$$r(x) = (110\ 101\ 011\ 010\ 000\ 100\ 001) \quad (100)$$

Výpočet syndromů:

$$\begin{aligned} S_1(a) &= r(a) = a^2 + a^4 + a + a^4 + a + a^2 = 0 \\ S_2(a^2) &= r(a^2) = a + a^2 + a^5 + a^0 + a^2 + a^2 = 0 \\ S_3(a^3) &= r(a^3) = a^0 + a^0 + a^2 + a^3 + a^3 + a^2 = 0 \\ S_4(a^4) &= r(a^4) = a^6 + a^5 + a^6 + a^6 + a^4 + a^2 = 0 \end{aligned} \quad (101)$$

Všechny syndromy se rovnají 0, proto je zpráva k dekodování bez chyby.

Kódová zpráva s 1 chybou:

$$e(x) = a^3x^2 \quad (102)$$

Zpráva k dekodování $r(x)$ (103).

$$r(x) = c(x) + e(x) = a^3x^6 + a^6x^5 + a^4x^4 + ax^3 + a^3x^2 + a^0x + a^2 \quad (103)$$

Výpočet syndromů:

$$\begin{aligned} S_1(a) &= r(a) = a^2 + a^4 + a + a^4 + a^5 + a + a^2 = a^5 \\ S_2(a^2) &= r(a^2) = a + a^2 + a^5 + a^0 + a^0 + a^2 + a^2 = a^0 \\ S_3(a^3) &= r(a^3) = a^0 + a^0 + a^2 + a^3 + a^2 + a^3 + a^2 = a^2 \\ S_4(a^4) &= r(a^4) = a^6 + a^5 + a^6 + a^6 + a^4 + a^4 + a^2 = a^4 \end{aligned} \quad (104)$$

Všechny syndromy se nerovnají 0, takže existují chyby.

Lokalizační mnohočlen:

$$i = 0; l = 0; A(x) = 1; L(x) = L_0 = 1 \quad (105)$$

$i \neq 4$

$$d = \sum_{n=0}^l L_n \cdot S_{i+1-n} = L_0 \cdot S_1 = 1 \cdot a^5 = a^5 \quad (106)$$

$d \neq 0$

$$T(x) = L(x) - d \cdot x \cdot A(x) = 1 - a^5 \cdot x \cdot 1 = 1 + a^5 x \quad (107)$$

$2l \leq i \Rightarrow$ platí

$$A(x) = \frac{L(x)}{d} = \frac{1}{a^5} = a^2 \quad (108)$$

$$L(x) = T(x) = 1 + a^5 x \quad (109)$$

$l = i + 1 - l = 1$

$i = i + 1 = 1$

$$i = 1; l = 1; A(x) = a^2; L(x) = 1 + a^5 x \quad (110)$$

$i \neq 4$

$$d = \sum_{n=0}^l L_n \cdot S_{i+1-n} = L_0 \cdot S_2 + L_1 \cdot S_1 = 1 \cdot a^0 + a^5 \cdot a^5 = a \quad (111)$$

$d \neq 0$

$$T(x) = L(x) - d \cdot x \cdot A(x) = 1 + a^5 x - a \cdot x \cdot a^2 = 1 + a^2 x \quad (112)$$

$2l \leq i \Rightarrow$ neplatí

$$L(x) = T(x) = 1 + a^2 x \quad (113)$$

$$A(x) = A(x) \cdot x = a^2 x \quad (114)$$

$i = i + 1 = 2$

$$i = 2; l = 1; A(x) = a^2 x; L(x) = 1 + a^2 x \quad (115)$$

$i \neq 4$

$$d = \sum_{n=0}^l L_n \cdot S_{i+1-n} = L_0 \cdot S_3 + L_1 \cdot S_2 = 1 \cdot a^2 + a^2 \cdot a^0 = 0 \quad (116)$$

$d = 0$

$$A(x) = A(x) \cdot x = a^2 x^2 \quad (117)$$

$i = i + l = 3$

$$i = 3; l = 1; A(x) = a^2 x^2; L(x) = 1 + a^2 x \quad (118)$$

$i \neq 4$

$$d = \sum_{n=0}^l L_n \cdot S_{i+1-n} = L_0 \cdot S_4 + L_1 \cdot S_3 = 1 \cdot a^4 + a^2 \cdot a^2 = 0 \quad (119)$$

$d = 0$

$$A(x) = A(x) \cdot x = a^2 x^3 \quad (120)$$

$i = i + l = 4$

$$i = 4; l = 1; A(x) = a^2 x^3; L(x) = 1 + a^2 x \quad (121)$$

$i = 4 \Rightarrow$ konec

$$L(x) = 1 + a^2 x \quad (122)$$

Chienovo vyhledávání:

$$\begin{aligned} L(a^0) &= 1 + a^2 = a^6 \\ L(a^1) &= 1 + a^3 = a \\ L(a^2) &= 1 + a^4 = a^5 \\ L(a^3) &= 1 + a^5 = a^4 \\ L(a^4) &= 1 + a^6 = a^2 \\ L(a^5) &= 1 + a^0 = 0 \Leftarrow \\ L(a^6) &= 1 + a = a^3 \end{aligned} \quad (123)$$

$$P_1 = (a^5)^{-1} = \frac{a^0}{a^5} = a^2 \quad (124)$$

Forneyův algoritmus

$$\begin{aligned}E(x) &= (1 + a^2x) \cdot (a^5x + a^0x^2 + a^2x^3 + a^4x^4)(\text{mod}x^4) \\E(x) &= (a^5x + a^0x^2 + a^0x^2 + a^2x^3 + a^2x^3 + a^4x^4 + a^4x^4 + a^6x^5)(\text{mod}x^4) \\E(x) &= a^5x\end{aligned}\tag{125}$$

$$M_1 = \frac{E(P_1^{-1})}{(1 - P_1)} = \frac{a^3}{a^0} = a^3\tag{126}$$

$$e(x) = M_1P_1 = a^3x^2\tag{127}$$

Oprava chyb

$$c(x) = r(x) + e(x) = a^3x^6 + a^6x^5 + a^4x^4 + ax^3 + a^3x^2 + a^3x^2 + a^0x + a^2\tag{128}$$

Opravený polynom (pomocí operace XOR) $c(x)$ (129).

$$c(x) = a^3x^6 + a^6x^5 + a^4x^4 + ax^3 + a^3x^2 + a^0x + a^2\tag{129}$$

$$c(x) = (110\ 101\ 011\ 010\ 110\ 100\ 001)\tag{130}$$