Bezpečnostní kódy, díl 11.

V klidu a bezpečí (11)

Z předchozího dílu víme, jak zkonstruovat generující matici cyklického kódu ve tvaru $G = [BE_k]$, která nám umožňuje nejen snadnou extrakci přenášené informace, ale i efektivní realizaci kódovacího algoritmu. Nyní na tuto zkušenost navážeme konstrukcí dekódovací procedury, která je stejně jako předchozí kódovací algoritmus vhodná zejména pro HW realizaci dekodéru.

Na úvod si ukážeme, jak pro generující matici ve tvaru $G = [BE_k]$ nalezneme odpovídající kontrolní matici. Vzhledem k tvrzení T3.6 je nasnadě očekávat, že toto hledání nebude příliš složité. Obdobně jako v případě tohoto tvrzení můžeme pro generující matici ve výše uvedeném tvaru najít kontrolní matici jako $H = [E_{n-k} -B^T]$. Vzhledem k tomu, že se jedná v podstatě o symetrickou úpravu T3.6, nebudeme si tento poznatek zavádět jako samostatné tvrzení. V případě potřeby jej nicméně lze formulovat jako důsledek v předchozím dílu uvedeného tvrzení T10.1 – využijeme vlastností duálního kódu a ukážeme, že matice H ve tvaru $H = [E_{n-k} -B^T]$ jej generuje díky platnosti výrazu $H*G^T = 0$ a dimenzi podprostoru generovaného maticí H.

Právě popsanou matici H bychom sice nyní mohli bez problémů použít k detekci a opravě chyb podle postupů uvedených ve třetím dílu tohoto seriálu (viz zde uvedená modifikace standardní metody dekódování pomocí syndromů), avšak v případě cyklických kódů se tohoto postupu příliš nevyužívá. Stejně jako v předchozím výkladu, kde jsme sice odvodili generující matici G, ale k vlastnímu kódování jsme využili postupy vycházející ze specifických vlastností cyklických kódů, i zde se dává přednost konstrukci dekódovací procedury pomocí operací na F[x]/f(x). Důvodem je zejména větší bohatost dostupného matematického aparátu a snadná realizace těchto operací pomocí posuvných registrů.

Vlastnosti syndromu

Pro využití polynomiální reprezentace přenášených slov k detekci a opravě chyb budeme potřebovat následující stěžejní tvrzení, jehož důkaz (založený na studiu chování operace $s = Hx^T$ pro výše odvozený tvar matice H) uvádí [VAOO89]. Mějme cyklický kód typu (n,k) s generujícím polynomem g(x). Nechť r(x) představuje polynom odpovídající přijatému slovu a s(x) je polynom odpovídající jeho syndromu. Potom je polynom s(x) zbytkem po dělení polynomu r(x) generujícím polynomem g(x), tj. $r(x) = q(x)g(x) + s(x)$, $\deg(s(x)) < \deg(g(x)) = n-k$ – tvrzení T11.1.

Uvedené tvrzení nám umožňuje určit syndrom přijatého slova s pomocí algoritmu dělení polynomů na F[x] (viz 8. díl tohoto seriálu), aniž bychom k tomu museli znát příslušnou kontrolní matici. Toto samo o sobě nám však nestačí, neboť nyní bychom stejně museli použít postup dle standardního dekódování. Naším cílem je však odvodit postup, který umožní tuto operaci provést efektivněji. K tomu účelu budeme muset nejprve zjistit, jakým způsobem se mění hodnota syndromu v závislosti na cyklickém posuvu dekódovaného slova.

Z předchozího výkladu víme, že cyklický posuv vektoru r vpravo odpovídá násobení odpovídajícího polynomu r(x) hodnotou x, přičemž tyto operace se provádějí na F[x]/f(x), kde $f(x) = x^n-1$. Nyní se zaměříme na způsob, jakým tato operace ovlivní původní syndrom polynomu r(x). Pro jednoduchost budeme nejprve uvažovat operace na okruhu F[x]. Podle T11.1 pro syndrom dekódovaného polynomu platí $r(x) = q(x)g(x) + s(x)$. Pro syndrom polynomu xr(x) proto platí $xr(x) = xq(x)g(x) + xs(x)$. Pokud dále platí, že $\deg(s(x)) < n-k-1$, potom je dle T11.1 polynom x(s(x)) rovněž syndromem polynomu xr(x). Toto však nemusí být vždy splněno, takže obecně je třeba počítat s tím, že bude třeba provést modulární redukci polynomu xs(x) polynomem g(x).

Cílem nyní bude požadovanou redukci mod g(x) provést co možná nejefektivněji. Vzhledem k tvaru s(x) a g(x) je možné takový způsob snadno najít. Naším účelem je vypočítat s'(x), které vyhovuje následující rovnici $xs(x) = q(x)g(x) + s'(x)$, $\deg(s'(x)) < \deg(g(x))$. Víme přitom, že $\deg(xs(x)) \leq n-k$ (podle T11.1), a proto také $\deg(q(x)g(x)) \leq n-k$. Dále víme, že $\deg(g(x)) = n-k$, a proto musí být q(x) nejvýše konstantní polynom (viz T8.3). Z uvedeného již s přihlédnutím k tomu, že g(x) je normovaný polynom, snadno odvodíme, že $q(x) = s_{n-k-1}$ pro $s(x) = s_0 + s_1x^1 + \ldots + s_{n-k-1}x^{n-k-1}$. Hledanou hodnotu s'(x) pak určíme jako $s'(x) = xs(x) – s_{n-k-1}g(x)$.

Na základě právě rozpracovaných úvah dostáváme jako jejich důsledek následující tvrzení: Buď φ cyklický kód typu (n,k) nad tělesem F s generujícím polynomem g(x). Nechť r(x) představuje polynom se syndromem $s(x) = \Sigma_{i=0}^{n-k-1} s_i x^i$. Potom syndromem s'(x) polynomu xr(x) je polynom $s'(x) = xs(x) – s_{n-k-1}g(x)$ – tvrzení T11.2.

Musíme ovšem připomenout, že právě uvedené tvrzení jsme dokázali s užitím operace na F[x] pro výpočet xr(x). Pro nás je však důležité vědět, jestli toto tvrzení platí i při použití operace na F[x]/f(x), která pro f(x) = x^n-1 odpovídá cyklické rotaci polynomu r(x) vpravo. Nyní se proto musíme vrátit k problému, jehož odložením jsme si před okamžikem poněkud ulehčili práci. Není však zas tak těžké ukázat, že T11.2 platí i při uvažování operace cyklického posuvu vpravo místo xr(x) (tj. posunu bez rotace). K tomuto účelu si zformulujeme a dokážeme následující tvrzení: Buď φ cyklický kód typu (n,k) nad tělesem F s generujícím polynomem g(x). Nechť a(x) představuje polynom se syndromem s(x) a nechť b(x) je polynom, pro který platí b(x) $\equiv$ a(x) (mod f(x)), f(x) = x^n-1. Potom je polynom s(x) syndromem polynomu b(x) – tvrzení T11.3.

K důkazu tohoto tvrzení si nejprve uvědomíme, že pro polynomy a(x) a s(x) platí a(x) $\equiv$ s(x) (mod g(x)). Dále víme, že polynom g(x) dělí polynom f(x), a proto z platnosti b(x) $\equiv$ a(x) (mod f(x)) plyne, že b(x) $\equiv$ a(x) (mod g(x)). Díky tranzitivnosti relace kongruence potom také b(x) $\equiv$ s(x) (mod g(x)). Vzhledem k tomu, že podle T11.1 pro stupeň polynomu s(x) platí deg(s(x)) < deg(g(x)), je rovněž s(x) zbytkem po dělení polynomu b(x) polynomem g(x) a podle T11.1 též syndromem tohoto polynomu.

Toto obecné tvrzení nyní snadno využijeme k vyřešení předchozího problému jednoduše tím, že položíme a(x) = xr(x) a b(x) = a(x) mod f(x). Polynom a(x) nyní odpovídá nějakému polynomu z F[x], pro který jsme prováděli úvahy při odvozování T11.2. Polynom b(x) zase koresponduje s cyklickým posuvem polynomu r(x), jehož syndrom chceme znát. Vzhledem k platnosti b(x) $\equiv$ a(x) (mod f(x)) můžeme nyní použít T11.3 k důkazu platnosti T11.2 i pro případ užití cyklického posuvu místo násobení na F[x].

Dekódovací procedura

V následující části se budeme věnovat odvození efektivní dekódovací techniky, která je v literatuře [VAOO89] uváděna pod názvem “zachytávání chyb” (Error Trapping). Výhodou této procedury je opět snadná obvodová realizace pomocí posuvných registrů se zpětnými vazbami. Musíme zde však podotknout, že metoda jako taková je schopná opravovat pouze chyby určitého typu (viz dále). Na druhou stranu je ale fakt, že většina typů kódů a chyb, které v těchto kódech přicházejí v úvahu jako opravitelné, dále stanovené podmínky splňuje. V příkladech si potom ukážeme možné rozšíření uvedeného algoritmu pro ty případy, kterým jeho standardní podoba nevyhovuje.

Nejdříve si zadefinujeme pojem cyklický běh: Cyklickým během délky m $\leq$ n nazveme posloupnost m cyklicky po sobě jdoucích znaků ve slově délky n znaků – definice D11.1. Jako příklad si vezmeme binární slovo e = (0100 0101), které obsahuje cyklický běh tří nul a jedné jedničky. Obdobně e = (1100 10111) obsahuje cyklický běh pěti jedniček a dvou nul.

Pokud nebude řečeno jinak, tak pro další úvahy předpokládáme, že máme dán cyklický kód typu (n,k) o minimální kódové vzdálenosti d_{min}. Generující polynom označme jako g(x). Dále zavedeme hodnotu t jako t = $\lfloor (d_{min}-1)/2 \rfloor$. Poznamenejme, že hodnota t udává maximální počet chyb, které je daný kód schopen v přijatých slovech opravit. Předpokládejme dále, že jsme přijali slovo r reprezentované polynomem r(x), které má syndrom s (v případě výpočtu jako s = Hr^T uvažujme tento syndrom po transpozici), respektive s(x). Na základě teorie vyvinuté pro standardní dekódovací metody je možné dokázat, že pokud platí w(s) $\leq$ t, pro w(s) představující váhu slova s (viz D3.5), potom je možné určit odpovídající chybový vektor e jako e = (s, 0) – tvrzení T11.4.

Právě uvedené tvrzení nám tedy umožňuje snadno nalézt chybové vektory způsobující chybu na prvních n-k pozicích kódových slov. Ačkoliv je toto jistě zajímavá vlastnost, nemůžeme ji ještě považovat za dostatečnou. Podíváme-li se na tvar chybového slova z T11.4 z jiného úhlu, vidíme, že obsahuje cyklický běh nul nejméně délky k. Vezmeme-li tuto vlastnost za podmínku (viz výše), kterou musí chybový vektor splňovat, abychom jej mohli pomocí T11.4 identifikovat, a spojíme-li ji s výše odvozeným tvrzením T11.2, které nám umožňuje sledovat změny syndromu v závislosti na rotacích dekódovaného slova, dostaneme dále popsaný algoritmus A11.1 založený na technice zachytávání chyb.

Algoritmus A11.1 předpokládá, že chybový vektor e přijatého slova r (r = c + e) obsahuje cyklický běh nul délky alespoň k. Postupným cyklickým posuvem vektoru r se snažíme získat slovo, u něhož chybový vektor odpovídá tvaru e = (s, 0). Tento stav je podle T11.4 možné identifikovat na základě platnosti podmínky w(s) $\leq$ t. Jednotlivé syndromy z i–tého průchodu algoritmem, které značíme s_i, přitom počítáme na základě s_0 (určen při přijetí slova r) a rekurzivní aplikace T11.2.

Předpokládejme, že v kroku i, 0 $\leq$ i < n, byl vypočten syndrom s_i, pro který platí w(s_i) $\leq$ t. Označíme-li e(x) chybový polynom přijatého slova, potom pro e(x) platí, že x^ie(x) mod f(x) odpovídá vektoru (s_i, 0). Chybovému vektoru e potom odpovídá cyklický posuv vektoru (s_i, 0) o n-i pozic doprava (záměrně zde dodržujeme směr rotace doprava kvůli jejímu vyjádření v podobě násobení

nezápornou mocninou x).

Příklady

Z předchozího výkladu víme, že nutnou podmínkou k tomu, aby algoritmus A11.1 správně identifikoval chybový vektor přijatého slova, je, že tento chybový vektor musí obsahovat cyklický běh nul v délce nejméně k znaků. Dále si ukážeme dva příklady – první, ve kterém tato podmínka splněna je, a druhý, ve kterém sice splněna není, avšak kde je možné provést úpravu A11.1 tak, aby tento fakt nečinil potíže.

Ještě než se pustíme do vlastních příkladů, učiníme drobnou poznámku ohledně užité symboliky. Na rozdíl od formálních definic a tvrzení je pro řadu příkladů vhodné poněkud "popustit uzdu fantazii" při značení operací a jejich výsledků. Tam, kde nebude hrozit nedorozumění, si proto dále dovolíme poněkud více směšovat vektorovou a polynomiální notaci (zejména ve výrazech) s jistou volností v rozlišování operací na F[x] a F[x]/f(x) (tato praxe je ostatně běžná i v řadě renomovaných publikací na toto téma).

Příklad první

Mějme cyklický kód typu (15,7) nad Z_2 (tj. binární kód), který je generován polynomem $g(x) = 1 + x^4 + x^6 + x^7 + x^8$. Předpokládejme, že o tomto kódu víme, že má minimální kódovou vzdálenost $d_{min} = 5$ (tj. opravuje všechny dvojnásobné chyby). Snadno zjistíme, že každý chybový vektor váhy nejvýše 2 (větší váhy z důvodu $d_{min} = 5$ neuvažujeme) musí obsahovat cyklický běh nul v délce nejméně 7. Podmínka pro funkci algoritmu A11.1 je zde proto triviálně splněna.

Předpokládejme, že jsme v prostředí tohoto kódu přijali slovo r = (1100 1110 1100 010). Převodem na odpovídající polynom r(x) a vydělením polynomem g(x) obdržíme syndrom $s(x) = 1 + x^2 + x^5 + x^7$. Vidíme, že váha tohoto syndromu zjevně není menší nebo rovna dvěma, a proto začneme postupně počítat jeho deriváty pro cyklické posuvy vektoru r. Konkrétní hodnotu syndromu v závislosti na rotaci slova r uvádí následující tabulka.

Posuv i	Syndrom s_i
0	1010 0101
1	1101 1001
2	1110 0111
3	1111 1000
4	0111 1100
5	0011 1110
6	0001 1111
7	1000 0100

Vidíme, že při rotaci o sedm pozic směrem doprava jsme obdrželi syndrom s_7, $w(s_7) \leq 2$. Podle T11.4 tomuto syndromu odpovídá chybový vektor $(s_7,0)$. Pro chybový polynom odpovídající slovu r potom platí: $x^7 e(x) \equiv s_7(x) \pmod{f(x)}$. Z této kongruence poté určíme chybový polynom e(x) jako $e(x) = x^{15-7} s_7(x) \bmod f(x)$. Zapsáno vektorově-polynomiální notací pak pro chybový vektor e platí: $e = x^8(1000\ 0100\ 0000\ 000) = (0000\ 0000\ 1000\ 010)$. Nakonec provedeme opravu na kódové slovo $c = r - e = (1100\ 1110\ 0100\ 000)$.

Příklad druhý

Pro účely tohoto příkladu budeme předpokládat binární cyklický kód typu (15,5) určený generujícím polynomem $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$, který má minimální kódovou vzdálenost $d_{min} = 7$. Podmínku pro správnou funkčnost A11.1 splňují všechny chybové vektory s váhou nejvýše tři s výjimkou vektoru e_q = (10000 10000 10000) a všech jeho cyklických posuvů. Tento vektor nebude možné pomocí A11.1 tak, jak byl popsán, identifikovat, neboť pro žádný z jeho cyklických posuvů nebude mít syndrom váhu menší nebo rovnu třem. Nicméně je zde možná následující modifikace.

Podívejme se nejprve na syndrom chybového vektoru e_q, pro který platí $s_q(x) = 1 + x^5 + a(x)$,

kde a(x) představuje zbytek po dělení polynomu x^{10} polynomem g(x). Záměrně zde ponecháváme syndrom v tomto tvaru, neboť z něho je dobře patrné, že po odečtení a(x) od $s_q(x)$ obdržíme syndrom o váze rovné dvěma.

Z výše uvedeného plyne následující návod na úpravu základní verze A11.1: v každém kroku i budeme kromě s_i počítat také hodnotu s_i-a. Pokud nastane situace, kdy platí $w(s_i\text{-}a) \leq 2$, potom víme, že chybovým vektorem přijatého slova je $e = x^{15-i}(s_i\text{-}a, (10000))$.

Pro příklad nyní předpokládejme, že jsme přijali slovo r = (11100 01111 00100). Postupný výpočet syndromů v jednotlivých krocích shrnuje následující tabulka.

Posuv i	Syndrom s_i	Hodnota s_i – a
0	00110 10001	11011 00011
1	11110 11010	00011 01000
2	01111 01101	10010 11111
3	11010 00100	00111 10110
4	01101 00010	10000 10000

Zde vidíme, že při rotaci o čtyři pozice vpravo jsme obdrželi syndrom, který má po odečtení vektoru a váhu rovnou dvěma. Chybový vektor slova r tak určíme jako $e = x^{11}$(10000 10000 10000) = (01000 01000 01000). Přijaté slovo tak dekódujeme na c = r – e = (10100 00111 01100).

Závěr

Popsaný dekódovací algoritmus pracující na principu zachytávání chyb je další z řady efektivních metod pro práci s cyklickými kódy, které byly navrženy se snahou o co nejsnazší realizaci s pomocí posuvných registrů se zpětnými vazbami. Ve své základní variantě (tj. při splnění podmínky na tvar chybových vektorů) je tento algoritmus skutečně velmi jednoduše realizovatelný pomocí základních logických obvodů. Nutnost ošetřování speciálních tvarů syndromů odpovídajících příslušným chybovým vektorům, které nesplňují stanovené podmínky, sice realizaci pomocí základních logických obvodů komplikuje, nicméně pro rozsah uvedený v příkladu číslo dvě je tento postup stále ještě únosný.

Tomáš Rosa, tomas.rosa@decros.cz