Bezpečnostní kódy, díl 10.

V klidu a bezpečí (10)

V tomto díle si dále prohloubíme znalosti z oblasti teorie cyklických kódů. Nejprve konstrukcí kontrolní matice H navážeme na předchozí výklad. Poté popíšeme způsob vytváření systematického kódu, pro který si ukážeme kódovací proceduru vhodnou pro hardwarovou realizaci kodéru.

V předchozím díle jsme se seznámili se způsobem konstrukce cyklického kódu. Podíváme-li se na generující matici v podobě, v jaké jsme ji odvodili pomocí T9.8, vidíme, že není ve tvaru $G = [E_k B]$, který by zaručoval vlastnost souvislé systematičnosti. Není přitom vhodné pokoušet se takový kód sestavit pomocí základních úprav generující matice, které jsme používali v případě obecných lineárních kódů (tj. u těch schémat, kde nám nezáleželo na vlastnosti cykličnosti). Mohlo by se nám totiž stát, že výsledný kód bude sice souvisle systematický, ale nebude již cyklický. Nicméně konstruovat cyklické kódy umožňující snadnou identifikaci informačních znaků samozřejmě umíme, avšak používáme k tomu poněkud odlišný postup, který si zde za okamžik ukážeme.

Než se ale budeme zabývat konstrukcí takového kódu, dokončíme ještě výklad z minulého dílu tím, že ukážeme obecný způsob konstrukce matice H odpovídající matici G, která byla vytvořena pomocí T9.8. Připomeňme, že se jedná o téma, které je důležité zejména pro obecný přehled (proto jej také uvozujeme výkladem duálního kódu). V praxi se většinou využívá jednodušší a účelnější postup založený na dále popsané konstrukci kódu s generující maticí ve tvaru $G = [BE_k]$.

Duální kód

Duální kód se v případě lineárních kódů využívá k rozhodování, zda je přijaté slovo kódové, či nikoliv. Na první pohled by se možná dalo říci, že se jedná o alternativní způsob detekce chyb k tomu, který jsme si dříve uvedli, avšak prakticky vzato to není pravda. V konečném důsledku totiž i při použití duálního kódu nakonec skončíme u kontrolní matice H, kterou odvodíme z matice G a použijeme nám již dobře známým způsobem (viz T3.5). Duální kód z tohoto úhlu pohledu proto není alternativním nástrojem pro detekci chyb, nýbrž alternativní teorií pro popis a tvorbu matice H. To je také důvod, proč jsme se až dosud bez tohoto pojmu dokázali obejít. Pro získání širšího přehledu však bude vhodné se s ním alespoň rámcově seznámit.

Mějme lineární kód φ typu (n,k) nad tělesem F. Duální kód (někdy též ortogonální doplněk) k tomuto kódu značíme jako $\varphi\perp$ a jeho množinu kódových slov definujeme jako $C_{k\perp} = \{x \in V_n(F)$: $x.y = 0$, pro všechna $y \in C_k\}$– definice D10.1.

Operace x.y použitá v definici přitom značí skalární součin uvedených vektorů. Budeme-li chtít užít geometrické interpretace duálního kódu, potom jej můžeme považovat za množinu vektorů, které jsou kolmé na každé kódové slovo.

Následující tvrzení ukazuje, že duální kód je sám o sobě podprostorem (tedy je to “opravdu kód”): Mějme lineární kód φ typu (n,k) nad tělesem F. Potom $\varphi\perp$ je lineární kód typu (n,n-k) nad tělesem F – tvrzení T10.1. Poznamenejme, že platí $(\varphi\perp)\perp = \varphi$.

Z právě popsaných vlastností duálního kódu se rýsuje velmi snadný, i když zároveň dosti nepohodlný postup pro identifikaci kódových slov: postupně vypočteme skalární součin přijatého slova s každým vektorem duálního kódu. Přijaté slovo potom označíme za kódové právě tehdy, když budou všechny takto získané výsledky nulové. Zde je však na první pohled vidět možné vylepšení, které říká, že není nutné procházet celý podprostor duálního kódu – místo toho stačí použít jeho generující matici.

Pomocí generující matice pak dostáváme následující prakticky již využitelné tvrzení: Buď φ lineární kód typu (n,k) nad tělesem F a nechť H je generující matice duálního kódu $\varphi\perp$. Potom je vektor $x \in V_n(F)$ kódovým slovem právě tehdy, když $Hx^T = 0$ – tvrzení T10.2.

Jak už jsme si řekli, představuje duální kód alternativní teorii vedoucí nakonec k sestavení kontrolní matice H. Výsledek tohoto postupu obvykle dostáváme v podobě následující definice: Buď φ lineární kód typu (n,k). Generující matici H duálního kódu $\varphi\perp$ nazýváme kontrolní maticí kódu φ – definice D10.2. Spolu s D3.6 tak nyní již umíme popsat matici H dvěma různými způsoby.

Konstrukce matice H

S využitím duálního kódu se můžeme pokusit najít kontrolní matici cyklického kódu pomocí generujícího polynomu, který bude vytvářet podprostor, jehož vektory budou ortogonální k vektorům všech kódových slov. Pro tuto konstrukci si vezměme například polynom $h(x) = f(x)/g(x)$, kde $f(x) = x^n-1$ a $g(x)$ je generující polynom ($\deg(g(x)) = n-k$) daného cyklického kódu typu (n,k).

Na první pohled se může zdát postačující využít $h(x)$ jako generující polynom hledaného duálního kódu. Podle T9.5 totiž $h(x)$ opravdu generuje cyklický podprostor, který má dle T9.8 dimenzi n-k. Vezmeme-li navíc libovolné polynomy $a(x)$, $c(x)$ takové, že $a(x) \equiv b(x)h(x) \pmod{f(x)}$ a $c(x) \equiv d(x)g(x) \pmod{f(x)}$, potom platí, že $a(x)c(x) \equiv 0 \pmod{f(x)}$ (připomeňme, že $f(x) = h(x)g(x)$). Odtud se dá již tušit, že jsme při hledání duálního kódu na správné cestě. Avšak pozor! Je třeba si uvědomit, že i analogie mezi ideály a cyklickými podprostory má své meze. Zde se konkrétně jedná o to, že nulový součin dvou polynomů na daném ideálu ještě neznamená nulový skalární součin jim příslušejících vektorů. Duální kód (poznamenejme, že nepožadujeme, aby byl cyklický) je přitom definován pomocí operace na vektorovém podprostoru, nikoliv na ideálu.

Po počátečním nadšení, kdy všechno šlo hladce, se tak nyní dostáváme do poněkud obtížné situace. Naštěstí její vážnost není tak tragická – ukazuje se totiž, že námi vytvořený cyklický kód má k hledanému duálnímu kódu velmi blízko – jsou to ekvivalentní kódy. Detailní popis přechodu od kódu generovaného výše zavedeným polynomem $h(x)$ k duálnímu kódu uvádí [VAOO89]. My si zde uvedeme pouze výsledek tohoto pochodu. Ještě předtím však pro úplnost zopakujeme následující definici reciprokého polynomu: Buď $h(x) = \sum_{i=0}^{k} a_i x^i$ polynom stupně k. Potom reciprokým polynomem $h_R(x)$ polynomu $h(x)$ nazýváme polynom: $h_R(x) = \sum_{i=0}^{k} a_{k-i} x^i$ – definice D10.3. Poznamenejme, že platí $h_R(x) = x^k h(1/x)$.

S využitím reciprokého polynomu je potom možné odvodit následující konstrukci duálního kódu: Mějme normovaný polynom $g(x)$ stupně $\deg(g(x)) = n-k$, který na $F[x]$ dělí $f(x) = x^n-1$, a tudíž je generujícím polynomem cyklického kódu φ typu (n,k). Nechť $h(x) = f(x)/g(x)$. Potom reciproký polynom $h_R(x)$ polynomu $h(x)$ generuje duální kód $\varphi\perp$ – tvrzení T10.3.

Vlastní kontrolní matici H (neboli generující matici duálního kódu) potom již vytvoříme stejným postupem jako v případě generující matice z T9.8. Konkrétní příklad této matice pro kód (7,4) generovaný polynomem $g(x) = 1+x+x^3$ (viz obrázek 2 předchozího dílu) je na obrázku 1. Jako matici $G^{'}$ jsme zde přitom označili generující matici kódu generovaného přímo polynomem $h(x)$. Takto můžeme snadno ukázat, že kódy $\varphi^{'}$ a $\varphi\perp$ jsou ekvivalentní, neboť vidíme, že jejich generující matice se liší permutací sloupců (pokud se nebudeme striktně držet postupu z T9.8, můžeme matici H vytvořit pouhým obrácením pořadí sloupců matice $G^{'}$).

Systematický kód

Struktura cyklických kódů, kterými jsme se až doposud zabývali, nebyla pro řadu praktických aplikací optimální v tom smyslu, že v přenášených slovech nebylo možné snadno identifikovat znaky nesoucí informaci. Proto nyní budeme hledat cestu, jak sestrojit generující matici cyklického kódu, která bude mít tvar $G = [BE_k]$. Všimněme si zde, že na rozdíl od obecného lineárního kódu zde užíváme tvar mající jednotkovou matici na konci místo na začátku matice G (tj. informační bity se vyskytují na konci kódových slov). To je dáno vlastnostmi cyklických kódů, konkrétně zvoleným pořadím zápisu koeficientů. Na vlastnosti daného kódu ani na způsob zacházení s ním tento fakt však nemá prakticky žádný vliv. Poznamenejme dále, že hovořit zde o konci a začátku přenášených slov ve vztahu k jejich přenosu může být ošidné, neboť většina algoritmů pro rychlé HW zpracování operací s cyklickými kódy přenáší data počínaje nejvyšším bitem – v tomto případě posledním bitem v "papírovém" zápisu. Z tohoto pohledu jsou potom informační znaky přenášeny jako první.

Předpokládejme nyní, že chceme vstupní zprávu $m = (a_0, a_1,...,a_k)$ převést na odpovídající kódové slovo, které bude mít tvar $c = (s_0, s_1, ...,s_{n-k-1}, a_0, a_1, ...,a_k)$, kde podřetězec $s_0, s_1, ...,s_{n-k-1}$ většinou označujeme jako paritní bity. Možný způsob konstrukce takových kódových slov nám ukazuje následující rovnice (na $F[x]$): $x^{n-k}a(x) = q(x)g(x) + t(x)$, kde $g(x)$ je generující polynom daného (n,k) kódu. Podle T8.6 má tato rovnice pro $q(x)$ a $t(x)$ jediné řešení splňující podmínku $\deg(t(x)) < \deg(g(x))$. Toto řešení přitom umíme najít nám známým algoritmem pro dělení polynomů.

Jednoduchou úpravou této rovnice dostaneme: $q(x)g(x) = -t(x) + x^{n-k}a(x)$. Odtud vidíme, že pravá strana rovnice jednak představuje kódové slovo (neboť se jedná o násobek generujícího polynomu), jednak má námi požadovaný tvar. Tímto jsme odvodili jednoduchý, avšak velmi užitečný způsob

kódování přenášených slov, který umožňuje snadno extrahovat přenášenou informaci.

Generující matici odpovídající uvedenému schématu sestrojíme následujícím postupem: postupně pomocí algoritmu pro dělení polynomů vypočítáme polynomy $q_i(x)$ a $t_i(x)$ vyhovující rovnici: $x^{n-k+i} = q_i(x)g(x) + t_i(x)$, $\deg(t_i(x)) < \deg(g(x))$, pro $0 \leq i \leq k-1$. Řádky matice G pak položíme rovny vektorům odpovídajícím polynomům $x^{n-k+i} - t_i(x)$. Praktický postup konstrukce generující matice pro nám dobře známý kód typu (7,4) je uveden na obrázku 2.

Kódovací algoritmus

Na základě uvedeného způsobu kódování přenášených zpráv je možné sestrojit efektivní kódovací algoritmus, který je vhodný zejména pro hardwarovou realizaci kodéru. Ačkoliv se realizací operací s cyklickými kódy budeme zabývat v samostatném dílu, je pro lepší pochopení užitečnosti předkládaných algoritmů vhodné poznamenat, že tyto operace jsou (byly) povětšinou realizovány pomocí posuvných registrů se zpětnými vazbami. Tyto jednoduché automaty totiž bylo poměrně snadné konstruovat už v dobách, kdy se teprve čekalo na první jednočipové mikropočítače.

Slíbený algoritmus (A10.1), který pro binární kód (n,k) očekává na vstupu zprávu $m = (a_0a_1...a_k)$ a na výstupu vytváří kódové slovo ve tvaru $c = (s_0s_1...s_{n-k-1}a_0a_1...a_k)$, je uveden na obrázku 3.

Ačkoliv to není z vlastního zápisu algoritmu na první pohled patrné, ve své podstatě uvedený sled instrukcí provádí výpočet zbytkového polynomu t(x), který vyhovuje rovnici $x^{n-k}a(x) = q(x)g(x) + t(x)$ a podmínkám T9.8. Jako výsledek operace kódování potom vystupuje polynom $c(x) = -t(x) + x^{n-k}a(x)$, což, jak jsme si ukázali výše, je kódové slovo v námi požadovaném tvaru.

Výkonný cyklus algoritmu je navržen tak, že na vstupu očekává sériový vstup kódované zprávy počínaje souřadnicí nejvyššího indexu (a_{k-1}). Výstup je rovněž sériový počínaje koeficientem nejvyšší mocniny polynomu c(x). Toto, pro ECC typické zpracování dat má výhodné vlastnosti pro hardwarovou realizaci řady algoritmů. Zde konkrétně využíváme tento fakt k tomu, abychom si nemuseli v registrech tvořících vnitřní proměnné algoritmu udržovat celou kódovanou zprávu. Místo toho pouze v registrech s_0 až s_{n-k-1} udržujeme zbytek po dělení vstupního polynomu polynomem g(x).

Zaměřme se nyní na klíčovou myšlenku tohoto algoritmu, která nám dovoluje počítat koeficienty polynomu t(x), aniž bychom k tomu potřebovali mít k dispozici celou kódovanou zprávu. Začněme jednoduchým odvozením následujícího zápisu: $x^{n-k}a(x) = a_0x^{n-k} + a_1x^{n-k+1} + ... + a_{k-1}x^{n-1} = (... (((a_{k-1}x^{n-k})x + a_{k-2}x^{n-k})x + a_{k-3}x^{n-k})x + ...)x + a_0x^{n-k}$.

Budeme-li realizovat výpočet polynomu $t(x) = x^{n-k}a(x) \bmod q(x)$ po jednotlivých “závorkách” výše uvedeného zápisu, zjistíme, že pro správný výpočet potřebujeme znát v i-tém kroku algoritmu pouze hodnotu a_{k-i} a výsledek předchozí iterace, který máme uložen v registrech s_0 až s_{n-k-1}.

Tělo cyklu v bodě (5) tedy nedělá nic jiného, než že výsledek předchozí iterace posunuje o jeden znak doprava (odpovídá násobení polynomem x), k této hodnotě přičte aktuální vstupní znak a_{k-i} a výsledek modulo g(x) uloží zpět do vnitřních registrů. I zde se samozřejmě využívá několik řádně vypilovaných fines. Hlavní z nich spočívá v samotné realizaci výpočtu operace modulo g(x). V případě, že se zjistí, že posuvem výsledku předchozí iterace vznikne polynom stupně x^{n-k} (vyššího stupně vzniknout nemůže), je provedena jeho redukce tím způsobem, že koeficient této mocniny je z polynomu vynechán (odečten) a místo něho je přičten polynom $\hat{g}(x) = g(x) - x^{n-k}$.

Důkaz správnosti takového postupu se opírá o operaci sčítání definovanou na okruhu F[x]/g(x). Zde platí, že $x^{n-k} \equiv \hat{g}(x) \pmod{g(x)}$ pro $\hat{g}(x) = g(x) - x^{n-k}$. Oba polynomy jsou reprezentanty téže třídy ekvivalence, takže pro výpočet zbytku po dělení polynomem g(x) můžeme “místo” x^{n-k} použít $\hat{g}(x)$, který má nižší stupeň než g(x), a tudíž nám jeho použití realizuje požadovanou operaci redukce modulo g(x). Z pohledu operací na F[x]/g(x) vlastně redukce modulo g(x) znamená náhradu redukovaného polynomu t(x) polynomem $t'(x)$ ze stejné třídy ekvivalence ($t'(x) \in [t(x)]$), avšak se stupněm $\deg(t'(x)) < \deg(g(x))$. Rozhodnutí o tom, zda se použije pouze operace posuvu, nebo posuv spolu s redukcí (přičtením $\hat{g}(x)$), je možné provést na základě znalosti hodnot a_{k-i} a s_{n-k-1}. Samotný posuv (bez redukce) se provádí právě tehdy, když jsou obě hodnoty jedničky nebo nuly (tj. jsou si rovny). V ostatních případech se provádí posuv spolu s redukcí. Toto pravidlo je možné jednoduše odvodit na základě sčítání na Z_2, které odpovídá známé logické operaci exclusive-or neboli xor.

Ohledně volené abecedy poznamenejme, že tento algoritmus byl navržen s důrazem na snadnou obvodovou realizaci kodéru, a tudíž předpokládá použití binárního cyklického kódu. Z čistě teoretického hlediska je možné celý algoritmus včetně předvedených fines upravit pro libovolnou abecedu o počtu znaků odpovídajícím mocnině nějakého prvočísla (potřebujeme, aby na abecedě bylo možné definovat těleso). Vzhledem k těsné vazbě na předpokládané použití diskrétních logických obvodů však takové

rozšíření patrně nemá pro praktické nasazení valný smysl, i když i zde samozřejmě může výjimka potvrzovat pravidlo.

Závěr

Výkladem duálního kódu jsme si rozšířili obzor obecného přehledu o teorii bezpečnostních kódů. Dále vyložená konstrukce kontrolní matice H nabízí univerzální způsob sestavení této matice na základě daného generujícího polynomu. V mnoha aplikacích ECC je vhodné, aby v kódových slovech bylo možné snadno identifikovat znaky nesoucí přenášenou zprávu. Pro tento účel jsme si ukázali způsob konstrukce generující matice ve tvaru $G = [BE_k]$, která takový kód vytváří. Ukázali jsme si rovněž, že na základě vlastností takto popsaného kódu je možné sestrojit efektivní kódovací algoritmus pro snadnou HW realizaci kodéru.

V příštím díle se budeme zabývat obdobně pojatými úpravami kontrolní matice H (přesněji maticí $H = [E_{n-k} – B^T]$), která nám zase umožňuje sestavit algoritmus vhodný pro HW realizaci dekodéru.

Tomáš Rosa, tomas.rosa@decros.cz