Bezpečnostní kódy, díl 3.

V dnešním díle se nejprve v krátkosti dotkneme tématu lineárních prostorů, na které navážeme výkladem o kódech lineárního typu. Budeme se zabývat zejména obecnými vlastnostmi těchto kódů, které si budeme demonstrovat na jednoduchých příkladech.

V klidu a bezpečí (3)

Obecný výklad teorie lineárních prostorů si dále dovolím maximálně zestručnit. Ukážeme si pouze ty nejzákladnější vlastnosti, které budeme potřebovat pro správné pochopení dalšího výkladu. Zájemce o hlubší studium této problematiky si tímto dovoluji odkázat na [ADAM89]. Jako elementární úvod do lineární algebry si v případě potřeby dovoluji doporučit [DEPO99].

Lineární prostory

Stručně řečeno, za lineární prostor považujeme každou množinu vektorů (označíme si ji třeba P), která je uzavřená vzhledem k operaci vektorového součtu a skalárního násobení (pro každé x, y $\in$ P a $\alpha \in$ R platí, že $\alpha(x + y) = (\alpha x + \alpha y) \in P$). V případě zmíněných operací dále platí obvyklé asociativní, komutativní a distributivní zákony. Každý lineární prostor obsahuje též nulový prvek (nulový vektor).

Ačkoliv říkáme, že lineární prostor se skládá z vektorů, nemusí se vždy jednat o aritmetické vektory, tedy o vektory typu $x = (x_1, x_2, \ldots, x_n)$. Daný prostor může být stejně dobře tvořen například množinou všech funkcí reálné proměnné se společným definičním oborem. V tomto případě budeme pod pojmem vektor chápat nějakou konkrétní reálnou funkci. Toto "přetížení" jména vektor může být někdy matoucí, a proto na něj raději upozorňuji. Naštěstí pro nás budeme prakticky vždy pracovat s prostory, které jsou tvořeny aritmetickými vektory (tedy těmi "pravými" vektory), takže zde nedorozumění nehrozí.

Předpokládejme, že máme dán lineární prostor P (tedy množinu vektorů P s danými vlastnostmi). Vezměme si nyní nějakou podmnožinu $M \subset P$. Nyní nás zajímá, zda i tato množina M tvoří lineární prostor. Snadno nahlédneme, že to nemůžeme obecně určit, neboť dané M můžeme zajisté vybrat tak "nešikovně", že pro nějaké x, y $\in$ M bude x + y ležet mimo původní M. Co s tím? Pro tento účel se zavádí pojem lineární obal množiny, který v našem případě značíme jako <M> a který je definován jako nejmenší lineární prostor v P, který obsahuje M ($M \subseteq$ <M>). Uvedené "obalení" množiny M nedělá v podstatě nic jiného, než že nám k této množině přidá všechny "chybějící" vektory tak, abychom docílili uzavřenosti výše uvedených operací. Poznamenejme, že lineární obal nějaké množiny M vytvoříme snadno jako množinu všech lineárních kombinací vektorů z M. Pokud tedy například máme M = { a, b }, potom <M> = { $(\alpha a + \beta b)$: a, b $\in$ M, $\alpha, \beta \in$ R }.

Máme-li lineární prostor P a nějakou jeho podmnožinu $M \subset P$, která je už sama prostorem (M = <M>), potom M označujeme jako podprostor prostoru P – *definice D3.1*.

Představme si nyní, že jsme z množiny P vybrali takovou její podmnožinu B $\subset$ P, jejíž lineární obal <B> tvoří celou množinu P, tedy <B> = P. Neprázdné množině B, obsahující lineárně nezávislé vektory s uvedenou vlastností, říkáme báze prostoru P – *definice D3.2*. Poznamenejme, že obecný lineární prostor nemusí mít konečnou bázi. Nás však budou zajímat pouze ty prostory, které konečnou bázi mají, a ty budeme nazývat vektorovými prostory dimenze dim(P), kde dim(P) udává právě počet prvků v bázi B – *definice D3.3*.

Uvedená vlastnost vektorových prostorů nám dává velmi užitečnou možnost popsat tyto jinak třeba velmi rozsáhlé množiny pomocí lineárních kombinací vektorů báze B, jejíž velikost odpovídá už "jen" dimenzi daného prostoru. Konkrétní báze přitom jednoznačně určuje vektorový prostor, který je jejím obalem. Prakticky si takový popis ukážeme za okamžik.

Poslední poznámka bude patřit algebraickým strukturám, se kterými budeme pracovat. Až na výjimky se budeme zabývat hlavně binárními kódy, což znamená, že základním stavebním kamenem našich teorií bude těleso Z_2. Pomocí prvků tohoto tělesa budeme tvořit aritmetické vektory $x = (x_1, x_2, \ldots, x_n)$, $x_i \in Z_2$, ze kterých budeme nakonec stavět vektorové prostory dle definice 3.3. Volbou tělesa Z_2 je dáno i chování operací vektorového součtu a skalárního násobení, které jsou všem čtenářům paralelně běžících seriálů o kryptografii jistě velmi dobře známy (bližší úvod do obecné algebry viz [ADAM89]). Pro jistotu připomínám, že hlavní zajímavostí tělesa Z_2 je fakt, že $1 \equiv -1 \pmod 2$, což například znamená, že operace přičtení a odečtení jedničky nám dávají stejný výsledek. Tato vlastnost někdy vede až k tomu, že se v obecných vztazích pro binární kódy zcela ignoruje znaménko minus a všude se píše jen plus. Pro zachování obecnosti se však pokusíme těmto "vylepšením" vyhnout, neboť při přechodu ke kódům o jiném základu (třeba ke Golayovu ternárnímu kódu) bychom pak mohli mít problémy.

Lineární kódy

Základem každého q-árního lineárního kódu typu (n,k) je vektorový prostor složený z q-árních aritmetických vektorů délky n, který označíme jako V(n,q). Platí, že dimenze dim(V(n,q)) = n a tento prostor obsahuje všechna slova tohoto kódu (tedy C = V(n,q)). Na tomto prostoru se dále definuje jeho podprostor $L \subseteq V(n,q)$ o dimenzi dim(L) = k, do kterého se zobrazují kódovaná slova (tedy $C_k = L$).

Pro účely kódování chápeme kódovaná slova též jako aritmetické vektory o délce k. Vlastní operace kódování pak tomuto (souřadnicovému) vektoru přiřazuje odpovídající vektor v prostoru L (pokud dále nebudeme chtít zdůrazňovat, že L je podprostorem V(n,q), budeme L označovat jako prostor). Jak jsme si už řekli, každý prvek prostoru je možné jednoznačně určit pomocí lineární kombinace vektorů jeho báze, což odpovídá násobení matice báze daným souřadnicovým vektorem. Odtud již přímo dostáváme návod pro kódovací předpis ve tvaru $\varphi(x) = xG$, kde G je matice, jejíž řádky tvoří vektory báze prostoru L. V teorii kódů se tato matice označuje jako generující matice daného kódu – *definice D3.4*. O matici G víme, že pro kód typu (n,k) je typu [k,n], neboli že má k řádků (odpovídá dimenzi dim(L)) a n sloupců (odpovídá délce vektorů daného kódů).

Vidíme, že pro úspěšné kódování vstupních znaků nám postačuje znát matici báze daného prostoru a umět ji vynásobit kódovaným vektorem. Jako praktický příklad si uveďme třeba náš známý kód sudé parity typu (4,3), který má generující matici G = ((1,0,0,1), (0,1,0,1), (0,0,1,1)). Předpokládejme, že chceme zakódovat třeba slovo x = (1,0,1). Podle předpisu tedy provedeme operaci $c = \varphi(x) = xG = (1,0,1,0)$. Stejně tak i slovo x = (1,0,0) snadno zakódujeme jako c = xG = (1,0,0,1). Snadno ověříme, že se opravdu jedná o kód sudé parity.

Systematičnost

V minulém příkladu, kde jsme si ukázali generující matici pro kód sudé parity, jste si mohli povšimnout toho, že kódová slova měla nejen sudou paritu (což bychom očekávali především), ale že vykazovala i vlastnost souvislé systematičnosti dle D2.2.

Celkem snadno nahlédneme, že daný lineární kód je souvisle systematický, když jeho matice G má tvar G = (E_k | B). Přitom E_k je jednotková matice řádu k a B je matice typu [k,n-k] – *tvrzení T3.1*. Poznamenejme, že v tomto případě je permutace informačních znaků volena tak, aby začátky jednotlivých kódových slov přímo odpovídaly kódované informaci. Platí totiž, že $\varphi(x) = x_1, x_2, ..., x_k, y_1, y_2, ..., y_{n-k}$, kde y = xB.

Nyní, když známe vztah mezi maticí G a vlastností kódu být souvisle systematickým, zbývá již jen dořešit otázku, co dělat v případě, kdy daný kód souvisle systematický není. Potom se nám bude hodit tvrzení, které říká, že jakýkoliv lineární kód je možné pomocí základních úprav neměnících hodnost matice G (viz [DEPO99]) převést na ekvivalentní souvisle systematický kód s generující maticí G = (E_k | B) – *tvrzení T3.2*.

Pro další výklad tedy budeme předpokládat, že matici G máme dánu ve tvaru G = (E_k | B).

Váha slova

Před dalším výkladem bude vhodné si poněkud rozšířit zavedené pojmy o výraz váha slova. Váhu slova x značíme jako w(x) a definujeme ji jako počet nenulových znaků ve slově x – *definice D3.5*. Například w(1001) = 2, w(1101) = 3, apod.

Pojem váha slova jsme si zavedli proto, abychom si vytvořili alternativní možnost výpočtu Hammingovy vzdálenosti. Platí totiž, že d(x,y) = w(x-y) – *tvrzení T3.3*. Důkaz tohoto tvrzení je snadný, neboť hodnota d(x,y) nám udává počet pozic, na kterých se slova x a y liší, což je právě počet nenulových pozic v jejich rozdílu.

Vlastnosti

Díky tomu, že lineární kódy jsou vystavěny nad vektorovým prostorem, získávají jejich kódová slova určité vlastnosti, které je vhodné mít na zřeteli. Nejdůležitější z těchto charakteristik si shrneme v následujících bodech, které označíme jako *tvrzení T3.4*:

- libovolná lineární kombinace dvou kódových slov je kódové slovo,
- nulový vektor je kódové slovo,
- minimální kódová vzdálenost odpovídá minimu váhy přes všechna nenulová kódová slova, tedy $d_{min}(\varphi) = \min \{w(x): x \in C_k \setminus \{0\}\}$.

Vlastnosti 1 a 2 plynou přímo z toho, že kód je vektorovým prostorem. Bod tři snadno dokážeme pomocí bodu jedna, neboť platí (dle T3.3), že $d_{min}(\varphi) = \min \{w(x-y): x,y \in C_k, x \neq y\}$. Neboli hledáme minimum váhy rozdílu dvou libovolných různých kódových slov. Z bodu jedna ovšem víme, že rozdíl dvou kódových slov je též kódové slovo, takže vlastně stačí najít minimum váhy přes všechna nenulová ($x \neq y$) kódová slova – *důkaz P3.1*.

Pro praktický návrh je důležitý zejména bod 3, díky němuž můžeme celkem snadno určit minimální kódovou vzdálenost, aniž bychom museli zkoumat všechny dvojice kódových slov.

Detekce chyb

Jak správně zakódovat vysílané slovo a jaké vlastnosti toto zakódování bude mít, už víme. Teď nám zbývá ještě rozebrat druhou část problému, a to určit, jak budeme s přijatou informací zacházet na straně dekodéru. Už minule jsme si naznačili, že podstatou lineárních kódů je fakt, že množina kódových slov tvoří nějaký lineární podprostor, díky čemuž můžeme odlišit slova kódová od slov nekódových na základě jejich příslušnosti (nebo naopak nepříslušnosti) k danému podprostoru.

Dobrá, buďme tedy konkrétnější. Řekněme, že jsme právě přijali slovo odpovídající vektoru w a že chceme zjistit, zdali je toto slovo kódové. Už víme, že slovo w bude kódové, pokud se nám podaří prokázat jeho příslušnost k podprostoru L, který je generován příslušnou maticí G, jež jednoznačně popisuje daný kód. Nyní si stačí uvědomit druhý možný způsob popisu našeho podprostoru L, který říká že tento podprostor je též množinou všech řešení homogenní rovnice ve tvaru $Hx^T = 0$. Matici H typu [n-k,n] a hodnosti hod(H) = n-k budeme nazývat kontrolní maticí kódu φ – *definice D3.6*.

To, co jsme si právě řekli, znamená, že pro každý lineární kód φ, který je generován maticí G, můžeme najít kontrolní matici H takovou, že platí $Hx^T = 0$ právě tehdy, když x je kódové slovo – *tvrzení T3.5*.

Zbývá ještě dořešit otázku, jak danou matici H najít. Řekli jsme si, že je to matice soustavy n-k homogenních rovnic, jejichž řešení tvoří prostor kódových slov. Známe-li dobře algebraickou strukturu použitého kódu, můžeme matici H vytvořit jednoduše tak, že najdeme všechny zmíněné rovnice. Jako příklad si uveďme opět kód sudé parity. O něm víme, že je typu (n,n-1), a tudíž hledáme pouze jednu homogenní rovnici, jejíž koeficienty budou tvořit kontrolní matici H. V případě sudé parity to bude známá rovnice $x_1+x_2+...+x_n = 0$. Pro konkrétní kód typu (4,3) dostaneme H = (1,1,1,1). Zvolme nyní namátkou třeba jedno kódové slovo c = (1,0,1,0) a jedno slovo nekódové y = (1,1,1,0). Vidíme, že $Hc^T = 0$, zatímco $Hy^T = 1 \neq 0$, což je plně v souladu s našimi předpoklady (nedůvěřivci si mnohou projet všech 2^4 slov).

Pro sudou paritu nám uvedený způsob hledání matice H postačoval, avšak jistě bychom rádi našli nějaké obecnější pravidlo, nejlépe pak takové, které umožní snadno přecházet od matice G k matici H a obráceně. Takový vztah skutečně existuje a vypadá následovně: máme-li generující matici G typu [k, n] kódu (n,k), kde $G = (E_k \mid B)$, potom kontrolní matici H určíme jako $H = (-B^T \mid E_{n-k})$. Matice H je typu [n-k,n] – *tvrzení T3.6.*

Prakticky si využití uvedeného tvrzení ukážeme na příkladu koktavého kódu typu (6,2), který má generující matici G = ((1,1,1,0,0,0), (0,0,0,1,1,1)). Nejdříve tuto matici prohozením druhého a čtvrtého sloupce upravíme na matici pro souvisle systematický kód G' = ((1,0,1,1,0,0), (0,1,0,0,1,1)). Nyní si všimneme, jak vypadá matice G' z pohledu předpisu $G' = (E_k \mid B)$. Vidíme, že matice E_k je jednotková matice druhého řádu a že je následována maticí B = ((1,1,0,0), (0,0,1,1)). Odtud již snadno vytvoříme

matici H = ((1,0,1,0,0,0), (1,0,0,1,0,0), (0,1,0,0,1,0), (0,1,0,0,0,1)). Jako malé cvičení si nyní můžete vyzkoušet, že pomocí této matice jste schopni odlišit slova kódová od nekódových.

Oprava chyb

Předpokládejme, že bylo vysláno kódové slovo c, které bylo v průběhu cesty zatíženo chybovým vektorem e. Přijali jsme tedy slovo w = c + e. Díky tomu, že pracujeme nad tělesem Z_2, můžeme změnu bitu přenášeného slova jednoduše popsat jako přičtení vektoru e, který má jedničky právě na těch pozicích, kde došlo ke změnám.

Nyní provedeme výše popsaný způsob dekódování, takže vypočteme hodnotu $s = Hw^T = Hc^T + He^T = 0 + He^T = He^T$. Hodnotu s budeme nazývat syndromem slova w. Vidíme, že pokud nedošlo během přenosu k chybám (platí e = 0), potom je syndrom přijatého slova nulový. To je plně v souladu s tím, co jsme si řekli v minulé části.

Dalším případem, kdy obdržíme nulový syndrom, je situace, kdy chybový vektor odpovídá nějakému kódovému slovu. Takové chyby nelze ani detekovat, natož pak opravit. To je ovšem opět plně v souladu s vlastnostmi lineárních kódů.

V ostatních případech se můžeme pokusit na základě nenulového syndromu s provést opravu přijatého slova w. Způsob, který si tu dnes v krátkosti ukážeme, je modifikací takzvané standardní metody dekódování (viz [ADAM89]). Využijeme zde předpokladu, že konkrétnímu chybovému slovu odpovídá konkrétní hodnota syndromu s. Na základě této úvahy potom sestavíme převodní tabulku T (může být realizována například pamětí typu ROM), jejíž pomocí určíme předpokládaný chybový vektor a provedeme opravu slova w jako $w' = w – T[s] = w – T[Hw^T]$.

Uvedený způsob ovšem předpokládá, že chybový vektor je syndromem s určen jednoznačně. To však platí pouze v případě, kdy nastalo maximálně tolik chyb, kolik je daný kód schopen opravit podle své minimální kódové vzdálenosti (viz. T1.2). Pokud bychom se pokoušeli opravit více chyb, zjistíme, že více různých chybových slov odpovídá stejnému syndromu, takže nebudeme schopni správně zkonstruovat tabulku T. Prakticky si to můžete vyzkoušet na výše uvedeném příkladu koktavého kódu. V případě opravy jednonásobných chyb (uvažujeme pouze vektory e o váze jedna) tabulku T zkonstruujeme snadno. Pokusíme-li se však o totéž pro dvojnásobné chyby, zjistíme, že v některých případech nejsme schopni rozhodnout, která chyba vlastně nastala.

Závěr

V tomto díle jsme si ukázali základní vlastnosti vektorových prostorů a lineárních kódů, které se nad nimi budují. Na první pohled možná působí probraná látka složitým dojmem, avšak z praktického hlediska se jedná pouze o několik pravidel, která je třeba si dobře osvojit, a hlavně pochopit vzájemnou provázanost jednotlivých tvrzení.

Příští pokračování bude již kompletně zasvěceno výkladu Hammingových kódů, na které nám zde už dnes nezbylo dost místa. S výhodou přitom zúročíme všechny dnes nabyté vědomosti.

Tomáš Rosa (tomas.rosa@decros.cz)

Literatura:

[ADAM89] Adámek, J.: Kódování. Praha, SNTL 1989.

[DEPO99] Demlová, M. – Pondělíček, B.: Úvod do algebry. Skripta ČVUT FEL, 1999.