Juliusz Brzezinski

Om felkorrigerande koder

Matematik i säkerhetens tjänst

Denna första artikel av två om matematik i säkerhetens tjänst behandlar felrättande koder – nästa kryptering. Både kodning och kryptering har en dominerande roll då det gäller säker överföring av information. Den bakomliggande matematiska teoribildningen är intressant i samband med gymnasieskolans nya kurs i diskret matematik. Artiklarna bygger på ett föredrag som hölls på Vetenskapsfestivalen i Göteborg den 7 maj 2001.

I tekniska sammanhang associeras säkerhet mycket ofta med säkra hus, säkra bilar, säkra broar, säkra telefonförbindelser och överhuvudtaget med nästan all teknik som vi använder i vardagliga situationer. Det är faktiskt matematik – ibland enkel, ibland mycket komplicerad – som oftast ger sitt bidrag till vår säkerhet i alla dessa sammanhang. Artikelserien tar upp två aktuella områden där matematiska teorier spelar en dominerande roll när det gäller mycket viktiga säkerhetsfrågor. Man kan säga att matematiken själv skapar och garanterar säkerhet. Dessa två områden är kodning och kryptering, och båda bygger på och utvecklas tack vare ett mycket gammalt, klassiskt och mycket vitalt område inom matematiken som kallas talteori. Kodningsteorin tillämpas på felkorrigering vid dataöverföring, medan krypteringsteorin används i samband med olika säkerhetssystem vid datakommunikation. Både kodning och kryptering är mycket intressanta i samband med den nya kursen i diskret matematik som ingår i vissa gymnasieutbildningar.

Juliusz Brzezinski
är professor i matematik vid
Göteborgs universitet
vid
Matematik och Datavetenskap
Chalmers tekniska högskola/
Göteborgs universitet

I många kommunikationssystem kodar man information till följder av nollor och ettor. Antag att man vill sända två meddelanden A och B. Det enklaste sättet är att översätta:

$$A \to 0$$
$$B \to 1$$

Överföringen sker med hjälp av tex ledningar eller radiovågor eller på något annat sätt. Resultatet kan bli att, beroende på störningar i kommunikationskanalen, nollan förvandlas till en etta eller tvärtom. Finns det

någon möjlighet att skydda sig mot en sådan störning? En möjlig lösning är att upprepa A och B till exempel två gånger dvs

$$A \to 0\,0$$
$$B \to 1\,1$$

Om den mottagna sekvensen nu är 01 eller 10 så kan man konstatera att det har inträffat ett fel. Med andra ord kan man upptäcka ett fel. Låt oss gå vidare och upprepa A och B tre gånger dvs

$$A \to 0\,0\,0$$
$$B \to 1\,1\,1$$

Situationen har förbättrats avsevärt. Om det inträffar högst ett fel i A eller B så får man följande sekvenser av signaler:

$$A \to 000,\ 100,\ 010,\ 001$$
$$B \to 111,\ 011,\ 101,\ 110$$

Nu kan man inte bara upptäcka högst ett fel utan också korrigera det. Om man nämligen har högst ett fel i A så får man en sekvens ur övre raden, däremot ger högst ett fel i B alltid en sekvens ur nedre raden. Detta betyder att högst ett fel i A aldrig kan leda till en sekvens som är ett resultat av högst ett fel i B. Om man får en sekvens ur övre raden och man antar att det har inträffat högst ett fel så kan man korrekt avläsa meddelandet som A. På samma sätt kan man sluta sig till B om man får en sekvens ur nedre raden.

Detta är det enklaste exemplet på en felkorrigerande kod. Rent allmänt kan man beskriva situationen på följande sätt: Man har en mängd av meddelanden X och en metod att översätta dessa meddelanden till sekvenser av 0 och 1 som kallas kodord. Alla dessa kodord som vi kan beteckna med $\mathbf{C}$ kallas för en kod. Varje sekvens av 0 och 1 (inte nödvändigtvis ett kodord) kallar man för en vektor. I vårt exempel består X av två meddelanden, koden $\mathbf{C}$ är 000 och 111. Det finns dessutom 6 vektorer av längden 3 som inte är kodord: 100, 010, 001, 110, 101, 011.

Vad är det som gör att koden i vårt exempel kan korrigera 1 fel? Svaret är att högst ett fel i ett av kodorden inte kan sammanblanda den resulterande vektorn med de vektorer som man får då högst ett fel inträffar i ett annat kodord. Hur kan man uttrycka den egenskapen i matematiska termer? Man kan säga att två olika kodord måste skilja sig på minst tre olika ställen. Detta är just den förutsättning som garanterar att ett fel i det ena kodordet inte kan ge upphov till en vektor som är ett resultat av ett fel i ett annat kodord. Vi summerar:

En kod korrigerar 1 fel om varje par av kodord skiljer sig på minst tre olika ställen.

Innan vi konstruerar mera effektiva koder låt oss bara tänka en stund på möjligheten att kunna korrigera två fel då man sänder två meddelanden. Om som tidigare dessa meddelanden är A och B så kan vi översätta:

$$A \to 0\ 0\ 0\ 0\ 0$$
$$B \to 1\ 1\ 1\ 1\ 1$$

Högst två fel i första kodordet 00000 ger högst två ettor, högst två fel i andra kodordet 11111 ger minst tre ettor. Alltså tolereras två fel av denna kod – meddelandet kan rekonstrueras även om två fel inträffar. Om vi däremot gjorde så att

$$A \to 0\ 0\ 0\ 0$$
$$B \to 1\ 1\ 1\ 1$$

så kunde två fel i A ge samma resultat som två fel i B (t ex 0011). Detta visar dock att en tillräcklig repetition av meddelanden i princip kan korrigera ett godtyckligt antal fel. I praktiska sammanhang testar man vanligen olika kanaler och vet vilket antal fel som bör korrigeras för att kanalen skall vara tillförlitlig. Vi skall begränsa oss till koder som korrigerar 1 fel. Tex har våra vanliga miniräknare en inbyggd kod som korrigerar ett fel – den berömda Hammingkoden som vi diskuterar längre fram. Metoden att skicka bara två korta meddelanden – ”ja” och ”nej” och upprepa varje tre eller fem gånger ter sig inte speciellt effektiv. I praktiska sammanhang måste man skicka många och ofta långa meddelanden. Även om tekniken kan

förbättras avsevärt är huvudidén densamma: om man har flera meddelanden måste dessa översättas till kodord (dvs sekvenser av nollor och ettor) på ett sådant sätt att olika kodord måste ligga ”tillräckligt långt” ifrån varandra. Om man t ex vill kunna rätta (högst) ett fel så måste kodorden skilja sig på minst tre olika ställen. Om detta villkor gäller så kan inte ett fel i ett av kodorden ge samma sekvens som ett fel i ett annat kodord.

Kan man konstruera koder på ett mera ”intelligent” sätt? Som exempel låt oss betrakta fyra meddelanden *A*, *B*, *C*, *D*. Vi skall också numrera våra meddelanden med nollor och ettor. Två meddelanden *A* och *B* kunde vi beteckna med 0 och 1, och vår ”repetitionskod” som korrigerar 1 fel kan vi då skriva som:

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

Om vi har fyra meddelanden kan vi anteckna dessa med 00, 01, 10 och 11. Vi vill konstruera en kod som korrigerar ett fel. Därför vill vi välja de fyra kodorden så att två godtyckliga kodord skiljer sig på minst tre ställen. Att repetera varje meddelande tre gånger skulle ge sekvenser av längden 6 dvs

$$A = 0\,0 \quad \to \quad 0\,0\,0\,0\,0\,0$$
$$B = 0\,1 \quad \to \quad 0\,1\,0\,1\,0\,1$$
$$C = 1\,0 \quad \to \quad 1\,0\,1\,0\,1\,0$$
$$D = 1\,1 \quad \to \quad 1\,1\,1\,1\,1\,1$$

Kan man göra samma sak bättre? Kan vi konstruera fyra kortare kodord som också ligger tillräckligt långt ifrån varandra? Låt oss försöka med sekvenser av längd 5 i stället för 6:

$$\begin{array}{rcl} & a\ b & \\ A = 0\,0 & \to & 0\,0\,0\,0\,0 \\ B = 0\,1 & \to & 0\,1\,0\,1\,1 \\ C = 1\,0 & \to & 1\,0\,1\,0\,1 \\ D = 1\,1 & \to & 1\,1\,1\,1\,0 \end{array} \qquad \begin{array}{rcl} & & a\ b\ c\ d\ e \end{array}$$

Vi väljer *c*, *d*, *e* så att $c = a$, $d = b$ och $e = a + b$. Vi summerar ettor och nollor enligt den binära aritmetikens lagar dvs enligt additionstabellen:

+	0	1
0	0	1
1	1	0

Nu kan vi jämföra alla de fyra sekvenserna som svarar mot *A*, *B*, *C*, *D* och konstaterar lätt att två godtyckliga av dessa fyra sekvenser skiljer sig på minst tre olika ställen. På det sättet har vi fått en 1-fel korrigerande kod. Låt oss observera att de första två symbolerna i varje kodord betecknar vårt meddelande. De övriga tre räknas ut i den binära aritmetiken i enlighet med vissa bestämda regler. Dessa ”kontrollsymboler” påminner om kontrollsiffran i våra personnummer (den sista siffran som kan beräknas med hjälp av de föregående kan användas till att upptäcka att personnumret inte är korrekt). Kodorden ovan är uppbyggda med hjälp av tre ”kontrollsiffror” – *c*, *d*, *e* som beräknas med hjälp av de föregående på ett bestämt sätt. Vi kan inte beskriva principer för den konstruktionen i denna artikel, men det är just algebraiska kunskaper, och i synnerhet den binära aritmetikens egenskaper som ligger bakom den konstruktionen.

Innan vi går vidare till Hammingkoden som är ännu mera effektiv låt oss uppskatta den vinst som vi har gjort i förhållande till repetitionskoden med kodorden av längden 6. Om vi sänder fyra meddelanden så behöver vi skicka 20 signaler (0 eller 1) då vi använder den sista koden. Repetitionskoden kräver i stället 24 signaler. Det är inte så stor vinst, men om man tänker sig en lång sekvens av t ex på 10 000 meddelanden som delas i 2500 paket med fyra meddelanden per paket så blir vinsten 10 000 signaler.

Den troligen mest berömda av alla 1-fel korrigerande koder är Hammingkoden som föreslagits år 1949 av den amerikanske matematikern Hamming och som startade utvecklingen av den algebraiska kodningsteorin. Idag är teorin mycket omfattande och ett antal matematiker runt om i världen arbetar med olika konstruktioner av effektiva koder som har stora tillämpningar i samband med säker dataöverföring. En av de

första tillämpningarna var användning av algebraiska koder vid dataöverföring mellan avlägsna satelliter och markstationer på Jorden t ex då Mariner 9 år 1972 skickade bilder av planeten Mars yta. Då använde man en algebraisk kod som heter Reed-Muller-koden.

En av de enklaste Hammingkoderna korrigerar ett fel och kan se ut så här:

a b c d		*a b c d e f g*
0 0 0 0	→	0 0 0 0 0 0 0
0 0 0 1	→	0 0 0 1 1 1 1
0 0 1 0	→	0 0 1 0 0 1 1
0 0 1 1	→	0 0 1 1 1 0 0
0 1 0 0	→	0 1 0 0 1 0 1
0 1 0 1	→	0 1 0 1 0 1 0
0 1 1 0	→	0 1 1 0 1 1 0
0 1 1 1	→	0 1 1 1 0 0 1
1 0 0 0	→	1 0 0 0 1 1 0
1 0 0 1	→	1 0 0 1 0 0 1
1 0 1 0	→	1 0 1 0 1 0 1
1 0 1 1	→	1 0 1 1 0 1 0
1 1 0 0	→	1 1 0 0 0 1 1
1 1 0 1	→	1 1 0 1 1 0 0
1 1 1 0	→	1 1 1 0 0 0 0
1 1 1 1	→	1 1 1 1 1 1 1

Här är *a, b, c, d* våra informationssymboler (dvs *a, b, c, d* betecknar olika meddelanden, sammanlagt 16 stycken) och *e, f, g* är ”kontrollsiffrorna” som räknas ut på följande sätt $e = a + b + d$, $f = a + c + d$, $g = b + c + d$. ”Hammingreceptet” är mycket enkelt att förklara, men utrymmet tillåter inte en vidare diskussion i denna artikel.

Hammingkoden är verkligen effektiv. Om vi skulle repetera varje meddelande tre gånger så hade vi använt $16 \cdot 12 = 192$ signaler (nollor och ettor). Samma effekt får man genom att använda $16 \cdot 7 = 112$ signaler. Om man har 10 000 meddelanden, uppdelade i 625 block om 16 meddelanden per block, blir vinsten $70 \cdot 625 = 43\,750$ signaler i förhållande till repetitionskoden!

Richard Hamming publicerade sin konstruktion år 1950. Hans koder har använts sen dess i många datorer och miniräknare. Två år tidigare publicerade en annan matematiker, Claude Shanon, ett arbete som spelade en mycket viktig roll i utvecklingen av kodningsteorin. I detta arbete bevisade Shanon att det med all säkerhet finns bra koder trots att han inte kunde ge en konkret konstruktion. Med en bra kod menar man en kod som rättar många fel och samtidigt är snabb. Med hastigheten av koden menas förhållandet mellan antalet informationssymboler och längden av koden (Hammingkoden har 4 informationssymboler och har längden 7 dvs dess hastighet är 4/7). Efter Shanons arbete konstruerades många koder med mycket imponerande prestanda – koder som rättar många fel och är mycket snabba. Som exempel kan vi nämna så kallade BCH-koder med hastigheten 92/127 som rättar 5 fel, hastigheten 231/255 som rättar 3 fel eller hastigheten 139/255 som rättar 15 fel. Kodningsteorin utvecklas hela tiden och då och då konstruerar matematiker nya och användbara algebraiska koder.

Litteraturen om algebraiska koder är mycket omfattande, men det finns inte så många populära presentationer av ämnet. Det finns dock delar av läroböcker till universitetskurser i algebra där kodningsteori utnyttjas som ett utmärkt exempel på direkta tillämpningar. En sådan bok används vid Göteborgs universitet: J. Brzezinski och J. Stevens, *Tillämpade diskreta strukturer*, där kapitel 11, 12 och 13 handlar just om algebraiska koder. En populär presentation av ”Kodningsteori och tipsproblemet” kan man hitta i *Nordisk Matematisk Tidskrift*, *Normat*, nummer 39 från 1991. I den artikeln ges en intressant och enkel tillämpning av algebraiska koder på ett annat praktiskt problem – en (lite problematisk) möjlighet att ”bli rik på matematik”. Låt mig avsluta med samma tanke som avslutar artikeln i Normat: Kodningsteorin visar att när det gäller matematik finns det inte någon tydlig gräns mellan teori och tillämpning och nyckeln till värdefulla tillämpningar ligger i goda teoretiska kunskaper i ämnet!