Institutionen för matematik, KTH

Kryptografi och primalitet

Anders Björner

1. Kryptografi

Hemliga koder har använts av diplomater och militärer sedan urminnes tider. På senare år har koder fått ökad aktualitet och helt nya tillämpningar inom datakommunikationsområdet. En bra kod bör vara snabb att chiffrera och dechiffrera, samtidigt som den bör vara svår att forcera för obehöriga. Innebörden i orden "snabb" och "svår" är här beroende av kapaciteten hos de datorer och algoritmer som för tillfället är kända.

Med ett *kryptosystem*

$$\mathcal{M} \;\underset{D}{\overset{E}{\rightleftarrows}}\; \mathcal{C}$$

menar vi två ändliga mängder $\mathcal{M}$ (meddelanden) och $\mathcal{C}$ (chiffer), och två funktioner E (chiffrering) och D (dechiffrering) sådana att

$$D\left(E(m)\right) = m, \quad \text{för alla } m \in \mathcal{M}. \tag{1}$$

Vanligtvis kan vi anta: $\mathcal{M} = \mathcal{C} = \mathbb{N}_n = \{0, \ldots, n-1\}$, t.ex. genom omkodning från andra teckensystem såsom alfabet (ex.vis ASCII-koden: $A \to 065, B \to 066, \ldots$).

Funktionen E bestäms av vissa parametrar, som kallas *krypteringsnyckel*, och D av parametrar som kallas *dekrypteringsnyckel*.

I klassisk kryptografi är kunskap om krypteringsnyckeln ekvivalent med kunskap om dekrypteringsnyckeln, i meningen att den ena lätt kan bestämmas ur den andra. Man måste alltså hålla dessa nycklar hemliga.

Exempel. (Caesar-chiffer) (användes av Julius Caesar, för 2000 år sedan). Välj $b \in \mathbb{Z}$, låt $E_b : \mathbb{N}_n \to \mathbb{N}_n$ definieras av

$$E_b(x) = x + b \pmod{n}.$$

Caesar använde $n = 25$ (alfabetets storlek) och $b = 3$, ex.vis: HEJ $\to$ KHM.
övning: Bestäm $D_b(x)$.

Exempel. (Vigenère-chiffer) (1586, populärt under flera århundraden). Detta är ett periodiskt Caesar-chiffer med olika nyckel för varje position modulo k. T.ex., med $k = 3$ och nyckeln $= 214$.

$$\text{HEJSAN} \mapsto \text{JFNUBR},$$

eftersom $E_2(H) = J$, $E_1(E) = F$, $E_4(J) = N$, $E_2(S) = U$ osv.

De flesta klassiska kryptosystem är lätta att forcera (t.ex. med frekvensanalys), och de har samtliga nackdelen att nyckeln under alla omständigheter måste hållas strängt hemlig.

1976 föreslog W. Diffie och M. Hellman principerna för en radikalt annorlunda typ av kryptosystem, kallad *kryptosystem med offentlig nyckel*. De tänkte sig ett nätverk av användare (t.ex. firmor, regeringar, bankkontor, militära baser), vi kallar dem $A, B, C, \dots$, som samtliga vill kunna kommunicera parvis med varandra under sekretess från övriga deltagare och ickebehöriga. Varje användare A väljer en krypteringsfunktion E_A med dekrypteringsfunktion D_A, sedan offentliggörs E_A medan D_A hålls hemlig. Alla krypteringsnycklar E_A publiceras i en "telefonkatalog", och den som vill skicka ett kodat meddelande m till A skickar $E_A(m)$, som sedan A kan dekryptera: $D_A(E_A(m)) = m$.

Med ett klassiskt kryptosystem skulle för ett nätverk med n användare krävas $\binom{n}{2}$ nycklar, en för varje par, som måste hållas hemliga. I ett system med offentlig nyckel krävs endast n nycklar, en för varje användare, och var och en behöver bara hålla sin egen dekrypteringsnyckel hemlig.

För att ett system med offentlig nyckel skall fungera krävs naturligtvis följande:

kunskap om E_A leder inte till kunskap om D_A.

Krypteringsfunktionen E_A skall vara en så kallad *envägsfunktion*, den skall vara lätt att beräkna för givna argument $m \in \mathcal{M}$, men dess invers $E_A^{-1} = D_A$ skall vara svår att beräkna. Vi går inte in på den exakta betydelsen av dessa ord, vilket kräver komplexitetsteori.

2. RSA-kryptografi

Den populäraste och hittills kanske bästa implementeringen av Diffie & Hellmans idé om kryptografi med offentlig nyckel föreslogs 1978 av R. Rivest, A. Shamir och L. Adleman. Den bygger på elementär modulär aritmetik och har kommit att kallas RSA-systemet efter upphovsmännens initialer.

Varje användare A som deltar i ett RSA-system väljer två hemliga primtal p och q av viss i förväg bestämd storleksordning (100-siffriga är i praktiken en lämplig storlek). Låt

$$n = pq \quad , \quad m = (p-1)(q-1).$$

A väljer också två tal e och d så att

$$1 < e, d < m \quad , \quad ed \equiv 1 \pmod{m}.$$

Detta kan göras genom att först välja e som är relativt primt till m, och sedan beräkna dess multiplikativa invers $e^{-1} = d$ modulo m med Euklides algoritm.

Krypterings- och dekrypteringsfunktionerna för A är nu:

$$E_A(x) = x^e \pmod{n} \quad , \quad D_A(y) = y^d \pmod{n} \tag{2}$$

för $x, y \in \mathbb{N}_n$. Således är meningen att A skall offentliggöra krypteringsnyckeln (n, e) men hemlighålla dekrypteringsnyckeln d. Då måste även talen p, q och m hemlighållas, annars kunde d lätt beräknas, se övning 4.

Vi bevisar nu att E_A och D_A uppfyller villkoret (1). De är i själva verket inversa permutationer av mängden $\mathcal{M} = \mathcal{C} = \mathbb{N}_n$.

Sats 1. *$D_A(E_A(x)) = x$ och $E_A(D_A(y)) = y$, för alla $x, y \in \mathbb{N}_n$.*

Bevis: De två ekvationerna säger samma sak, nämligen

$$x^{ed} \equiv x \pmod n \quad , \quad \text{för alla } x \in \mathbb{N}_n. \tag{3}$$

Vi visar först att $x^{ed} \equiv x \pmod p$. Om primtalet p delar x så är vänsterled och högerled kongruenta med noll modulo p, och således lika. Om p inte delar x så säger Fermats lilla sats (se Sats 2 nedan) att $x^{p-1} \equiv 1 \pmod p$. Eftersom $ed = 1 + k(p-1)(q-1)$ för något heltal k (detta är innebörden i $ed \equiv_m 1$), så har vi

$$x^{ed} = x^{1+k(p-1)(q-1)} = x \cdot \left(x^{p-1}\right)^{k(q-1)} \equiv_p x \cdot 1^{k(q-1)} = x.$$

På samma sätt inses att $x^{ed} \equiv x \pmod q$. Således är $x^{ed} - x$ delbart med de båda primtalen p och q, och därför även med deras produkt $n = pq$, vilket ger (3). $\square$

För att RSA-systemet skall fungera på önskat sätt krävs följande:

(a) Stora primtal (100-siffriga) kan snabbt och någorlunda slumpmässigt hittas.

(b) Det är svårt att inom rimlig tid dela upp ett stort sammansatt tal (200-siffrigt) i primtalsfaktorer.

(c) Stora potenser $x^e \pmod n$, där samtliga ingående tal har 100–200 siffror, kan snabbt beräknas.

Punkterna (a) och (b) kommer att diskuteras i avsnitt 4. Punkt (c) är lätt att motivera med hjälp av *metoden med upprepad kvadrering*:

Låt $e = e_r e_{r-1} \dots e_0$ vara den binära representationen av talet e, dvs. $e_i \in \{0, 1\}$ och

$$e = e_r \cdot 2^r + e_{r-1} \cdot 2^{r-1} + \dots + e_0.$$

Då har vi

$$x^e = x^{2^r \cdot e_r} \cdot \ldots \cdot x^{e_0} = \prod_{i: e_i = 1} x^{2^i}. \tag{4}$$

Talen x^{2^i} kan beräknas genom upprepad kvadrering:

$$x, x^2, x^4, x^8, \dots \tag{5}$$

Vi ser att allt som allt behövs högst $2r \approx 2\log_2 e$ multiplikationer för att beräkna x^e, först r stycken för att beräkna följden (5) av kvadrater och sedan högst r stycken för att beräkna produkten (4). Alla räkningar görs modulo n, så vi behöver aldrig röra oss med tal större än n^2. Med denna metod kan de potenser $x^e \pmod n$ som blir aktuella i RSA-kryptografi beräknas på ca. en sekund på medelstora datorer.

Exempel. Antag givet ett RSA-system med krypteringsnyckel $n = 91$ och $e = 5$.

(a) Bestäm dekrypteringstalet d.

(b) Kryptera meddelandet $x = 5$.

(c) Dekryptera sedan resultatet.

Vi har $n = 91 = 7 \cdot 13$, och därför $m = 6 \cdot 12 = 72$.

(a) Vi löser kongruensen $5d \equiv 1 \pmod{72}$ med Euklides algoritm:

$$\begin{aligned} 72 &= 5 \cdot 14 + 2 \\ 5 &= 2 \cdot 2 + 1. \end{aligned}$$

Således

$$\begin{aligned} 1 &= 5 - 2 \cdot 2 \\ &= 5 - 2 \cdot (72 - 5 \cdot 14) \\ &= -2 \cdot 72 + 29 \cdot 5, \end{aligned}$$

$\Rightarrow 29 \cdot 5 \equiv_{72} 1 \Rightarrow d = 29.$

(b) $E(6) = 6^5 \pmod{91}$

$$\begin{aligned} 6^2 &= 36 \\ 6^4 &= 36^2 = 1296 \equiv_{91} 22 \\ 6^5 &= 6^4 \cdot 6 \equiv_{91} 22 \cdot 6 = 132 \equiv_{91} 41 \end{aligned}$$

$\Rightarrow E(6) = 41$

(c) $D(41) = 41^{29} \pmod{91})$

$$\begin{aligned} 41^2 &= 1681 \equiv_{91} 43 \\ 41^4 &\equiv_{91} 43^2 = 1849 \equiv_{91} 29 \\ 41^8 &\equiv_{91} 29^2 = 841 \equiv_{91} 22 \\ 41^{16} &\equiv_{91} 22^2 = 484 \equiv_{91} 29 \end{aligned}$$

$41^{29} = 41^{16} \cdot 41^8 \cdot 41^4 \cdot 41 \equiv_{91} 29^2 \cdot 22 \cdot 41 \equiv_{91} 29 \cdot 41 \equiv_{91} 6.$

$\Rightarrow D(41) = 6.$

RSA-systemet (och liknande kryptosystem med offentlig nyckel) kan användas inom datorkommunikation för identitetskontroll. Vi skall här kortfattat omnämna två viktiga exempel på detta.

I många situationer krävs att en användare skall kunna identifiera sig för en dator via ett "lösenord". Men en systemkunnig "superuser" (eller kanske en "hacker") kan ju titta i filen med lösenord och därefter logga in med falsk identitet. För att skydda mot detta kan man använda ett RSA-system och i lösenords-filen lagra $E_A(x)$ i stället för lösenordet x. När någon loggar in beräknar ett program P snabbt talet $E_A(x)$ som jämförs med det tal som lagrats i lösenords-filen. Om någon tar sig in i systemet och avläser $E_A(x)$ och programmet P kan han ju ändå inte (inom rimlig tid) sluta sig till lösenordet x, och därmed inte logga in som den person som har detta lösenord.

En annan tillämpning gäller s.k. "elektronisk signatur". Säg att användare B skickar ett meddelande till användare A (t.ex. ett uppdrag till ett bankkontor att överföra en miljard kronor till ett visst konto). Hur skall A veta att meddelandet verkligen har skickats av B, och inte av någon bedragare? Med ett RSA-system kan man i princip göra så här (vi bortser i beskrivningen från att $n_A \neq n_B$).

Meddelandet x krypteras av B till $y = E_A(D_B(x))$. Sedan skickas y till A som dekrypterar

$$E_B(D_A(y)) = E_B \underbrace{D_A E_A}_{Id} D_B(x) = E_B D_B(x) = x,$$

se Sats 1. Märk att endast B kan kryptera på detta sätt (endast B känner funktionen D_B) och endast A kan dekryptera. Så om ett begripligt dekrypterat meddelande $E_B D_A(y) = x$ når A, så vet A att det måste ha sänts av B.

3. Fermats lilla sats

RSA-kryptografi bygger som vi har sett på följande sats. Den härleds i Biggs (sid. 121) som ett specialfall av en sats av Euler. Vi ger här ett direkt bevis.

Sats 2. (Fermat, ca. 1640) *Om p är ett primtal så gäller*

$$a^p \equiv a \pmod{p} \quad , \quad \textit{för alla } a \in \mathbb{Z}.$$

Korollarium. *Om dessutom $a \not\equiv 0 \pmod{p}$ så gäller*

(i) $a^{p-1} \equiv 1 \pmod{p}$
(ii) $a^{p-2} \equiv a^{-1} \pmod{p}$

Detta får man genom att multiplicera $a^p \equiv_p a$ med det inversa elementet a^{-1} i $\mathbb{Z}_p$. Märk att (ii) ger en metod att beräkna a^{-1} i detta fall. Sats 2 kommer att bevisas som en följd av tre lemman (hjälpsatser).

Lemma 1. $\binom{p}{i} \equiv 0 \pmod{p}$, *om p är primtal och $0 < i < p$.*

Bevis: Skriv $\binom{p}{i} = k$, så $p! = k \cdot i! \cdot (p-i)!$. Primtalet p delar vänsterledet, och alla faktorer i högerledet utom k är mindre än p. Alltså måste p dela k. $\square$

Lemma 2. $(a+b)^p \equiv a^p + b^p \pmod{p}$, *om p är primtal.*

Bevis: Enligt binomialsatsen och Lemma 1:

$$(a+b)^p = \sum_{i=0}^{p} \binom{p}{i} a^{p-i}\, b^i \equiv_p a^p + b^p.$$

$\square$

Lemma 3. $(a_1 + a_2 + \ldots + a_k)^p \equiv a_1^p + a_2^p + \ldots + a_k^p$.

Bevis: Följer ur Lemma 2 med induktion. $\square$

Bevis för Sats 2:

$$a^p = (1 + 1 + \ldots + 1)^p \equiv_p 1^p + 1^p + \ldots + 1^p = a.$$

$\square$

Exempel. Beräkna $7^{100} \pmod{17}$.

$$7^{100} = \left(7^{16}\right)^6 \cdot 7^4 \equiv_{17} 1^6 \cdot 49^2 \equiv_{17} (-2)^2 = 4.$$

4. Primtalstester

RSA-systemet är beroende av att man snabbt skall kunna hitta 100-siffriga primtal. Vi vet från en välkänd sats av Euklides att primtalen är oändligt många, men kanske är de så tunt fördelade (luckorna mellan dem så stora) att slumpmässigt valda tal inte har en chans att ligga nära ett primtal. En djup och svårbevisad matematisk sats, den så kallade *primtalssatsen* (förmodad av Gauss ca. 1790, bevisad 100 år senare av Hadamard och de la Vallée Poussin), visar att så inte är fallet. Enligt primtalssatsen kan man vänta sig att i ett intervall av längd $\ln(n)$ runt ett stort heltal n finna ett primtal. (Detta skall förstås i statistisk mening, att om experimentet upprepas många gånger för olika tal n så hittar vi i medeltal ett primtal i intervallen av längd $\ln(n)$.) Till exempel, i ett intervall av längd $\ln 10^{100} \approx 230$ runt ett tal av storleksordningen 10^{100} finns i medeltal ett primtal. För att slumpmässigt välja ett 100-siffrigt primtal kan vi alltså börja med ett slumpmässigt valt 100-siffrigt udda tal N. Om N visar sig vara sammansatt fortsätter vi med $N+2$, sedan med $N+4$, osv. När vi kommit till $N+230$ bör vi (statistiskt sett) ha hittat ett primtal.

Antag nu att vi har ett stort tal N och vill avgöra om detta är ett primtal. Säg att N *klarar Fermat-testet med bas* b, för $1 < b < N$, om

$$b^{N-1} \equiv 1 \pmod{N}). \tag{6}$$

Vi vet från Sats 2 att om N är ett primtal så klarar det Fermat-testet. Men även vissa sammansatta tal kan uppfylla (6), t.ex. $3^{90} \equiv 1 \pmod{91}$). Det sägs att Leibniz, och även kinesiska matematiker ca. 400 år f.Kr., trodde att inga sammansatta tal klarar Fermat-testet med bas 2, dvs. att (6) med $b = 2$ ger ett deterministiskt primalitetstest, men detta är tyvärr felaktigt.

Ett enkelt probabilistiskt primalitetstest kan göras så här. Välj slumpmässigt k stycken tal $b_1, \ldots, b_k$ mellan 1 och N. Utför Fermat-testet (6) med bas b_i för $i = 1, \ldots, k$. Om N inte klarar ett av dessa tester vet vi att N är sammansatt. Om N klarar samtliga tester vet vi bara att N *kan* vara ett primtal, och sannolikheten ökar med antalet tester k. Tyvärr finns det tal som klarar alla Fermat-tester (6) med b och N relativt prima, men ändå är sammansatta, de så kallade *Carmichael-talen*. De är kvadratfria (dvs det finns inget primtal p så att p^2 delar talet) sammansatta tal N så att om primtalet p delar N så delar $p-1$ $N-1$. Det minsta Carmichael-talet är $561 = 3 \cdot 11 \cdot 17$. Om de ingående primfaktorerna i ett sådant tal är mycket stora, är sannolikheten att vi slumpmässigt skall finna ett b så att talet inte klarar Fermat-testet mycket liten. Därför måste det primalitetstest vi beskrivit förbättras.

Vi går inte in på detaljerna, men vill nämna att M. Rabin 1980 lanserade ett probabilistiskt primalitetstest av i princip det slag vi har beskrivit, som antingen med säkerhet avgör att N är sammansatt eller också med mycket stor sannolikhet att N är primt. Chansen att ett icke-primtal skall passera Rabins test är så liten att denna metod anses tillförlitlig att använda i samband med RSA-kryptografi. På dagens generation av större datorer tar det med dessa metoder ca. 40 sekunder att primalitetstesta ett 100-siffrigt tal, och ca. 10 minuter för ett 200-siffrigt tal.

RSA-systemets säkerhet är beroende av att det skall vara omöjligt att snabbt faktorisera ett stort sammansatt tal. Mer precist, man vill inte genom att känna till det 200-siffriga talet $n = p \cdot q$ kunna sluta sig till de två primtalen p och q, och därigenom kunna beräkna dekrypteringsnyckeln d. Det pågår intensiv forskning runt faktoriseringsalgoritmer, men i dagsläget är de bästa algoritmerna helt obrukbara för att systematiskt kunna knäcka RSA-systemet. Följande uppgifter om deras kapacitet är hämtade ur Mackiw (1985):

Antal siffror i n	*Antal steg i algoritmen*	*Tidsåtgång*
50	$1,4 \times 10^{10}$	4 timmar
100	$2,3 \times 10^{15}$	74 år
200	$1,2 \times 10^{23}$	$3,8 \times 10^{9}$ år

Avslutningsvis bör det påpekas att ingen har lyckats teoretiskt bevisa att snabba (eller åtminstone väsentligt snabbare) faktoriseringsalgoritmer inte existerar. Därför måste användare av RSA-kryptografi leva med en viss oro att framsteg inom detta område kan göra deras system osäkra. T.ex. krävde amerikanska militärledningen (Pentagon) ett tag att få rätt att förhandsgranska och hemligstämpla forskningsrapporter som berör dessa delar av talteorin. Detta genomfördes aldrig, men belyser den gamla erfarenheten att delar av den "rena" matematiken som anses helt oanvändbara för praktiskt bruk plötsligt kan få viktiga tillämpningar.

Nyhet: Forskare i Indien (Agrawal, Kayal, Saxena) har lyckats utforma en determinsitisk algoritm och bevisat att den avgör om ett heltal är ett primtal eller inte med säkerhet i polynomiell tid (i antalet siffror). Detta är ett mycket stort genombrott i komplexitetsteori och blev mycket uppmärksammat när det presenterades i augusti 2002. Men det påverkar inte säkerheten hos RSA-algoritmen. Fortfarande har ingen visat om man kan faktorisera ett tal snabbt eller inte. Den intresserade kan läsa mer på t.ex.
http://fatphil.org/maths/AKS/

5. Referenser

1 Neal Koblitz. *A course in number theory and cryptography.* Springer-Verlag, New York, second edition, 1994.

2 George Mackiw. *Applications of abstract algebra.* John Wiley & Sons Inc., New York, 1985.

3 Paulo Ribenboim. *The little book of big primes.* Springer-Verlag, New York, 1991.

4 Hans Riesel. *Prime numbers and computer methods for factorization.* Birkhäuser Boston Inc., Boston, MA, second edition, 1994.

5 Manindra Agrawal, Neeraj Kayal, Nitin Saxena *PRIMES is in P* Ann. of Math. (2) 160 (2004), no. 2, 781–793.

6. Övningar

(1) Antag att du som deltagare i ett RSA-system har offentlig krypteringsnyckel $n = 35$ och $e = 7$.
   (a) Kryptera meddelandena 6, 10 och 17.
   (b) Dekryptera de chiffrerade meddelandena 3 och 8.

(2) Antag att A och B har offentliga nycklar $n_A = 91$, $e_A = 5$ och $n_B = 35$, $e_B = 7$.
   (a) A vill sända meddelandet 41 till B med elektronisk signatur. Bestäm det chiffrerade meddelandet.
   (b) Samma fråga om B vill sända meddelandet 8 till A med elektronisk signatur.

(3) Finn primtalen p och q, om $n = pq = 4386607$ och $m = (p-1)(q-1) = 4382136$.

(4) Visa att om för ett RSA-system ett av talen p, q eller m är känt (utöver n och e), så kan talet d lätt beräknas.

(5) (a) Bestäm resten vid division av 3^{201} med 11.
   (b) Bestäm $2^{-1} \pmod{11}$).

(6) Låt p och q vara skilda primtal. Visa att

$$p^{q-1} + q^{p-1} \equiv 1 \pmod{pq}.$$

(7) (a) Klarar talet 341 Fermat-testet med bas 2?
   (b) Är 341 ett primtal?

(8) (a) Visa att antalet icke-enfärgade cirkulära halsband med p numrerade pärlor i a färger (alla färger behöver inte användas), är $a^p - a$.
   (b) Härled Fermats lilla sats ur detta. (*Ledning*: Om p är primtal och numren tas bort kan en rotation av halsbandet inte överföra detta i sig självt. Räkna nu antalet icke-numrerade halsband upp till rotations-symmetri.)

(9) (a) Låt $n = p_1 p_2 \dots p_k$, där p_i är distinkta primtal. Sätt $m = (p_1 - 1)(p_2 - 1) \dots (p_k - 1)$, och välj e och d så att $ed \equiv 1 \pmod{m}$. Visa att

$$x^{ed} \equiv x \pmod{n}$$

för alla heltal x.
   (b) Är samma sak sann även om vi släpper på kravet att $p_i \neq p_j$ för $i \neq j$? Ge bevis eller motexempel.

SVAR:

1. (a) $E(6) = 6,\quad E(10) = 10,\quad E(17) = 3$ (b) $D(3) = 17,\quad D(8) = 22$.
3. $p = 1453,\quad q = 3019$ (eller tvärtom).
5. (a) 3, (b) 6.
7. (a) Ja, (b) Nej $(341 = 11 \cdot 31)$.