Offentlig kryptering

Johan Håstad

KTH

1. Inledning. Denna uppgift går ut på att studera ett offentligt kryptosystem. Med detta menas ett kryptosystem där det är offentligt hur man krypterar, men trots detta kan bara någon som besitter en hemlighet dekryptera. Metoden vi kommer att beskriva kallas RSA–systemet efter Ron Rivest, Adi Shamir och Len Adleman som föreslog systemet. Huvuddelen av uppgiften går ut på att implementera systemet.

RSA–systemet bygger på talteori och för att beskriva och sedermera förstå det behövs litet bakgrund.

2. Litet elementär talteori. Låt m vara ett heltal. Vi kommer att räkna med talen $0, 1, 2 \ldots m-1$ modulo m. Detta innebär att vi har de vanliga räknesätten $+, -$ och $\cdot$, men vi är bara intresserade av vilken rest svaret ger vid divison med m. Vi skriver $+, -$ och $\cdot$ som vanligt medan vi använder likhetstecken med 3 streck samt lägger till $\pmod m$ för att markera att vi bara är intresserade av vilken rest talen ger vid division med m. Vi bör här observera att för att veta vilken rest svaret ger vid division med m är det tillräckligt att veta vilken rest operanderna ger vid division med m och inte behöver deras exakta värde.

Visa detta.

Till exempel har vi

$$4 \cdot 6 \equiv 6 \pmod 9$$
$$13 + 1 \equiv 14 \pmod{18}$$
$$2 - 8 \equiv 11 \pmod{17}$$

Vi kommer också att behöva division med tal modulo m och här får man tänka till litet eftersom kvoten av två heltal inte vanligtvis är ett heltal. Vi diskuterar inte detta problem närmare utan definierar för tillfället $a/b \pmod m$ som det tal $c \pmod m$, så att $b \cdot c \equiv a \pmod m$. De problem som kan uppstå (inget sådant c (t.ex. $7/4 \pmod{14}$), flera sådana c (t.ex. $10/4 \pmod{14}$)) kommer inte att spela någon större roll för oss, men du kan fundera på vad som bör göras. Hur man effektivt beräknar c beskrivs i slutet under *Bra att veta*.

En av hörnstenarna i RSA–systemet är följande klassiska sats.

FERMATS LILLA SATS. *Om p är ett primtal och $1 \leq a < p$ så är $a^{p-1} \equiv 1 \pmod p$.*

EXEMPEL. Låt $p = 17$ och $a = 3$.

$$3^{16} \equiv 9^8 \equiv 81^4 \equiv 13^4 \equiv 169^2 \equiv (-1)^2 \equiv 1 \pmod{17}.$$

Som vi gjorde här är det ofta bekvämt att använda t.ex. (-1) i stället för 16 vid handräkning. Detta ger inget problem ty (-1) och 16 ger samma rest vid division med 17.

Om du vill kan du försöka visa Fermats lilla sats. Enklast är kanske att visa $a^p \equiv a \pmod p$ med induktion över a. Använd binomialsatsen och att $\binom{p}{i}$ är delbart med p då $1 \leq i \leq p-1$.

Låt oss fortsätta med litet fler bakgrundsfakta. Låt N vara produkten av två olika primtal, $N = p \cdot q$.

LEMMA 1. *Anta att vi vet vilken rest ett tal ger vid division med p och vid division med q. Då vet vi också vilken rest det ger vid division med $N = p \cdot q$ förutsatt att $p \neq q$ och p och q är primtal.*

För att verifiera detta behöver vi bara visa att om a och b ger samma rest vid division med p och vid division med q, så ger de

samma rest vid division med N. Men detta är ganska uppenbart då $a - b$ är delbart med både p och q och således med $p \cdot q$.

Skriv ut beviset med alla detaljer.

Lemmat är ett specialfall av en sats som kallas den kinesiska restsatsen. Du kan hitta mer information om elementär talteori i boken av Hardy och Wright som nämns i litteraturlistan.

3. RSA–systemet. Låt N vara produkten av två primtal $N = p \cdot q$. Det är viktigt att det bara är mottagaren som vet p och q, det enda som kommer att offentliggöras är N. Låt M vara minsta gemensamma multipel av $p - 1$ och $q - 1$. Eftersom båda talen är jämna vet vi t.ex. att $M \leq \frac{(p-1)(q-1)}{2}$. Låt nu k och k' vara två heltal så att $k \cdot k' \equiv 1 \pmod{M}$. Talen M och k' kommer att hållas hemliga medan k publiceras. Innan vi fortsätter, låt oss ge ett exempel.

Låt $N = 323 = 17 \cdot 19,\ M = 144,\ k = 5,\ k' = 29$. Anta att 210 symboliserar det hemliga meddelandet. Det krypteras som

$$201^k = 201^5 \equiv 201 \cdot (201^2)^2 \equiv 201 \cdot 26^2 \equiv 201 \cdot 30 \equiv 216 \pmod{323}.$$

Observera att detta kan göras av någon som känner N och k.

Detta dekrypteras genom

$$\begin{aligned} 216^{k'} &\equiv 216^{29} \equiv 216^{16} \cdot 216^8 \cdot 216^4 \cdot 216 \\ &\equiv 273 \cdot 220 \cdot 64 \cdot 216 \equiv 305 \cdot 258 \equiv 201 \pmod{323}, \end{aligned}$$

och vi återfår meddelandet. Observera att mottagaren bara behöver k' och således kan glömma p, q och M.

Låt oss nu beskriva systemet formellt och förklara varför man återfår ursprungsvärdet.

Låt m vara det hemliga meddelandet som kan skrivas som ett tal (a blir 01, b blir 02 o.s.v.). Anta att $1 \leq m < N$. (Om meddelandet

är långt blir talet större, men då tänker vi oss det som flera tal som alla är mindre än N.)

OFFENTLIG KRYPTERINGS ALGORITM. I offentlig kryptering beräknas c (chiffertexten) genom $c \equiv m^k \pmod{N}$. N och k är offentliga som tidigare nämnts.

HEMLIG DEKRYPTERINGS ALGORITM. Meddelandet återfås genom $m \equiv c^{k'} \pmod{N}$.

Låt oss visa att dekrypteringen är korrekt. Vi har att $c^{k'} \equiv m^{k \cdot k'} \pmod{N}$ och därmed behöver vi bara följande.

LEMMA 2. *För alla* m, $0 \leq m < N$ *är det sant att* $m^{k \cdot k'} \equiv m \pmod{N}$.

Genom att använda Lemma 1 behöver vi bara visa att $m^{k \cdot k'} \equiv m \pmod{p}$ och $m^{k \cdot k'} \equiv m \pmod{q}$. Bevisen är identiska och därmed visar vi bara den första likheten. Genom valet av M, k och k' vet vi att

$$k \cdot k' = a \cdot M + 1 = a \cdot b \cdot (p-1) + 1$$

för heltal a och b. Således är om $m \not\equiv 0 \pmod{p}$

$$m^{k \cdot k'} \equiv (m^{p-1})^{a \cdot b} \cdot m \equiv 1^{a \cdot b} \cdot m \equiv m \pmod{p},$$

där vi har använt Fermats lilla sats. Om $m \equiv 0 \pmod{p}$ är identiteten självklar. Därmed har vi visat Lemma 2 och dekrypteringsalgoritmen är således korrekt.

4. Diskussion. Varför fungerar RSA som ett kryptosystem? Skälet är att det verkar som om det är svårt att finna k' givet N och k. Den bästa kända algoritmen för att göra detta är att faktorisera N och räkna fram M och sedan k'. Faktorisering tar dock lång

tid och även de bästa algoritmer på specialbyggda maskiner klarar av högst ungefär 80–siffriga tal på en månad. Det är möjligt att det finns bättre algoritmer för faktorisering eller att det finns ett sätt att dekryptera RSA utan att faktorisera N. Det pågår mycket forskning inom detta område, men ännu finns inga sådana resultat.

Om det nu är svårt att räkna ut k', hur går det då till att konstruera systemet? Skälet är förstås att konstruktören känner till p och q och kan räkna ut M och sedan k och k'. Enda svårigheten är att konstruera stora primtal p och q. Detta görs genom att välja ett slumpvis stort tal och testa om det är primtal. Mycket forskning har ägnats åt att testa om stora primtal, men vi kommer här bara att välja ett simpelt test som fungerar för det mesta.

Idé. Om $a^{p-1} \equiv 1 \pmod{p}$ för ett slumpvist valt a, $1 \leq a < p$ så är p antagligen primtal.

Denna idé kan göras mer precis, men för det mesta räcker den. Således blir det vårt primtalstest *Prova om* $a^{p-1} \equiv 1 \pmod{p}$ *för ett antal slumpvist valda* a.

För en utförligare diskussion om primtalstest kan du läsa artiklarna av Pomerance och Wagon som nämns i litteraturförteckningen.

5. Förslag till uppgifter. Nu har du all information som krävs för att implementera RSA–systemet. Gör det med så stora p och q (och därmed N) du kan. Det kan vara viktigt att komma ihåg att datorers heltal är av begränsad storlek. Några tips finns i avdelningen *Bra att veta*.

RSA kan användas till annat än bara enkel kryptering. Till exempel kan man signera meddelanden. Signaturen av ett meddelande m är $m^{k'} \pmod{N}$.

Kalla signaturen s. Det är lätt att verifiera signaturen då $s^k \equiv m \pmod{N}$ och k och N ju är offentliga. Att prestera en korrekt sig-

natur av ett meddelande är lika svårt som att dekryptera ett meddelande.

Försök göra något praktiskt med hjälp av RSA–systemet. I dagens hemlighetsfulla värld finns det mycket digital information som skall skyddas eller verifieras som autentisk. Om du lyckas riktigt bra och börjar tjäna pengar på någon produkt, så bör du veta att RSA är patenterat i USA.

6. Slutord. RSA–system ställer många frågor som är obesvarade.

Hur mycket tid måste en algoritm som faktoriserar heltal ta?

Varför är det lättare att visa att tal är primtal än att faktorisera dem?

Finns det andra bra system för offentlig kryptering?

Forskning pågår för att svara på dessa frågor (se referenslistan). Över huvud taget finns det många obesvarade frågor om existens av effektiva algoritmer. Detta område av matematiken brukar kallas komplexitetsteori.

Bra att veta. För att beräkna $a^k \pmod N$ för stora tal a, k och N kan det vara bra att organisera räkningarna med litet eftertanke. Vi tar exemplet $N = 1013$, $a = 514$ och $k = 411$. Börja med att skriva k binärt, d.v.s. som en summa av två–potenser

$$k = 256 + 128 + 16 + 8 + 2 + 1.$$

Beräkna sedan $a^{2^i} \pmod N$ för $i = 0, 1 \ldots 8$. Detta görs lätt genom

att kvadrera föregående tal.

$$\begin{aligned}
a &\equiv 514 \\
a^2 &\equiv 514^2 \equiv 816 \pmod{1013} \\
a^4 &\equiv 816^2 \equiv 315 \pmod{1013} \\
a^8 &\equiv 315^2 \equiv 964 \pmod{1013} \\
a^{16} &\equiv 964^2 \equiv 375 \pmod{1013} \\
a^{32} &\equiv 375^2 \equiv 831 \pmod{1013} \\
a^{64} &\equiv 831^2 \equiv 708 \pmod{1013} \\
a^{128} &\equiv 708^2 \equiv 842 \pmod{1013} \\
a^{256} &\equiv 842^2 \equiv 877 \pmod{1013}.
\end{aligned}$$

Det följer att

$$\begin{aligned}
514^{411} &\equiv 514^{256} \cdot 514^{128} \cdot 514^{16} \cdot 514^8 \cdot 514^2 \cdot 514 \\
&\equiv 877 \cdot 842 \cdot 375 \cdot 964 \cdot 816 \cdot 514 \\
&\equiv 970 \cdot 872 \cdot 42 \\
&\equiv 220 \cdot 872 \\
&\equiv 383 \pmod{1013}.
\end{aligned}$$

Man kan använda liknande trick men huvudidén är den samma. Det är också bra att kunna beräkna största gemensamma delare av två tal och att givet y_1, y_2 och m beräkna $y_1/y_2 \pmod{N}$. Båda görs med Euklides' algoritm.

Låt oss börja med största gemensamma delare. Största gemensamma delare av två tal a och b betecknas med (a, b) och är det största tal som delar både a och b. Idén bakom algoritmen är att om ett tal delar a och b så delar det också $a - k \cdot b$ för alla heltal k. Algoritmen beskrivs nu kanske enklast med ett exempel.

Låt oss ta talen 534 och 114. Anta att d delar dessa två tal, då delar det också

$$534 - 4 \cdot 114 = 78$$

och också

$$114 - 78 = 36$$

och också

$$78 - 2 \cdot 36 = 6$$

$$36 - 6 \cdot 6 = 0.$$

Nu slutar algoritmen eftersom vi fick talet 0. Det sista talet 6 kontrolleras lätt vara svaret.

Låt oss beskriva hur man beräknar $y_1/y_2 \pmod N$. Först måste vi definiera vad detta betyder. Låt oss säga att är det tal c så att $c \cdot y_2 \equiv y_1 \pmod N$. Om det finns flera c, så välj ett godtyckligt, finns det inget sådant tal är y_1/y_2 odefinierat.

Beräkna nu y_1/y_2 på följande sätt. Beräkna först (y_1, y_2). Om detta är d, använd att $y_1/y_2 = (y_1/d)/(y_2/d)$. Vi kan således anta att $(y_1, y_2) = 1$. Om nu $(y_2, N) > 1$ är y_1/y_2 odefinierat. (*Visa detta.*) Om $(y_2, N) = 1$ beräkna $e \equiv 1/y_2 \pmod N$ med Euklides algoritm. Sedan är svaret $y_1 \cdot e \pmod m$. Vi ger ett exempel på hur man beräknar $1/53 \pmod{91}$.

Utför Euklides algoritm på 53 och 91.

$$91 - 53 = 38$$

$$53 - 38 = 15$$

$$38 - 2 \cdot 15 = 8$$

$$15 - 8 = 7$$

$$8 - 7 = 1.$$

Låt oss nu använda ekvationerna baklänges.

$$\begin{aligned} 1 &= 8 - 7 = 8 - (15 - 8) = 2 \cdot 8 - 15 \\ &= 2 \cdot (38 - 2 \cdot 15) - 15 = 2 \cdot 38 - 5 \cdot 15 \\ &= 2 \cdot 38 - 5 \cdot (53 - 38) = 7 \cdot 38 - 5 \cdot 53 \\ &= 7 \cdot 91 - 12 \cdot 53. \end{aligned}$$

Ur detta följer att

$$12 \cdot 53 \equiv -1 \pmod{91}$$

och

$$(-12) \cdot 53 \equiv 1 \pmod{91}$$

och således

$$79 \cdot 53 \equiv 1 \pmod{91}$$

d.v.s.

$$\frac{1}{53} \equiv 79 \pmod{91}.$$

Litteratur

En referens för elementär talteori är
Hardy, G.H., & Wright, E.M., *An introduction to the theory of numbers.* Fifth edition, Oxford Univ. Press, Oxford 1979.

Där finns bl.a. Fermats lilla och kinesiska restsatsen.

RSA–systemet presenterades först i
Rivest, R., Shamir, A., Adleman, L., A method for obtaining digital signatures and public–key cryptosystems. *Communications of ACM* 21 (1979), s 120–126.

Om du vill läsa mer om primtalstest finns följande artiklar
Wagon, S., Primality testing. *Mathematical intelligenser*, 8:3 (1986), s 58–61.
Pommerance, C., Recent developments in primality testing. *Mathematical intelligenser*, 3 (1981), s 97–105.

För andra förslag på offentliga krypteringssystem, se
Merkle, R., Hellman, M., Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, IT 24 (1978), s 525–553.
McEliece, R.J., A public–key cryptosystem based on algebraic coding theory. *DSN Progress Report* 42–44, Jan. and Feb. 1978.

De första av dessa har forcerats, se t.ex.
Shamir, A., A polynomial time algorithm for breaking the basic Merkle–Hellman cryptosystem. *Proceedings of 23rd IEEE Symposium on Foundations of Computer Sience*, 1982, s 145–152.