

OFFENTLIG-NØKKEL-KRYPTOGRAFI

S. O. SMALØ

ABSTRACT. I dette notatet, som skal inngå som pensum i etter- og viderutdanningskurs i datasikkerhet, vil vi gi en kort innføring i offentlig-nøkkel-kryptografi med illustrasjoner fra det såkalte RSA-skjemaet introdusert av Ronald Rivest, Adi Shamir og Leonard Adleman i 1977. Dette er basert på vanskeligheten av faktorisering av naturlige tall i deres primfaktorer, mens en del andre tallteoretiske operasjoner kan utføres raskt og effektivt.

1. INTRODUKSJON

La oss se på hvordan tre personer, kall dem A , B og E , kan kommunisere sikkert med hverandre over en åpen kanal slik at hver av dem kan lese det som er tiltenkt ham, uten at den tredje part kan få tak i innholdet av beskjeden. Vi vil også lage en måte å gjøre dette på, slik at den som beskjeden er tiltenkt også med sikkerhet kan avgjøre hvem som har sendt meldingen.

Dette virker kanskje ikke så enkelt til å begynne med. Problemstillingen er faktisk ikke så gammel heller, selv om kryptering i prinsippet går tilbake til oldtiden i militær sammenheng.

For å komme i gang, bør vi kanskje først som sist omforme våre "beskjeder" som er skrevet i klartekst på (for eksempel) norsk til tall. Vi kan for eksempel bryte hver tekstbeskjed ned i blokker der hver blokk er en rekke av 50 symboler der et symbol er en stor eller liten bokstav, mellomrom, tallene fra 0 til 9, og tegnene . , ? ! ; : ' ' ' osv. Vi skal i det etterfølgende komme tilbake for vårt valg av blokker. Totalt

er det mindre enn 100 tegn. Vi ordner så disse i rekkefølge og på den måten tilordner vi til hvert tegn et tosifret tall der vi starter med 01, så 02 osv til 99. En beskjed med 50 eller færre tegn blir da til et 100-sifret tall der vi fyller inn 00-er dersom beskjeden har færre en 50 tegn. En slik rekkefølge kan være $a\ b \cdots \text{å}\ 0\ 1\ 2 \cdots 9$ mellomrom $A\ B \cdots \text{Å}\ .\ ,\ ?\ !\ : \ ;\ \text{“}\ \text{”}\ -\ -\ * \ \% \ / \ (\) \ \{ \} \ [\] \ < \ > \ @ \ \text{£} \ \$$. Tegnet a blir da tilordnet det tosifrede tallet 01, b blir tilordnet tallet 02, mens mellomrom tilordnes det tosifrede tallet 40, og A tilordnes tallet 41. For eksempel vil beskjeden: Drep ham, ikke vent til jeg kommer. – svare til tallet en får ved å sette sammen tallene for D som er 44, for r som er 18, for e som er 05, for p som er 16, for “mellomrom” som er 40, osv til det 100-sifrede tallet 441805164008011361400911110540220514204020091240100507401115131305186000.

Etter at dette er gjort, kan en nå tenke seg at en opererer på tall med 100 siffer og disse tallene blir fra nå av kalt beskjeder.

Hver av personene A , B og E velger seg nå en hemmelighet som vi vil kalle henholdsvis H_A , H_B og H_E . (Vi skal komme tilbake til hvordan dette gjøres i RSA-systemet.) Fra disse hemmelighetene kan hver av personene A , B og E danne seg et par av respektive funksjoner (som vi også kaller nøkler) a_H, a_P, b_H, b_P og e_H, e_P . Funksjonene eller nøklene med indeks P blir publisert og dermed gjort tilgjengelige for alle, og tar et tall med 100 siffer til et tall som har litt flere sifre. Hvilket intervall denne funksjonen bringer tallet til skal også oppgis. Funksjonene a_H, b_H og e_H blir kalt de hemmelige funksjonene eller nøklene og de holdes strengt hemmelige av de respektive eierne. Egenskapen til disse funksjonene a_H, b_H og e_H er da at $a_H(a_P(m)) = m$, $b_H(b_P(m)) = m$ og $e_H(e_P(m)) = m$ for alle beskjeder m . Kjennskap til funksjonen a_P, b_P og e_P skal ikke gi informasjon om hvordan a_H, b_H og e_H kan beregnes.

Dersom B nå vil sende beskjeden m til A så tar B ganske enkelt sin beskjed m , anvender a_P på denne og sender $a_P(m)$ til A . A kan da bruke sin hemmelige funksjon eller nøkkel a_H på resultatet og får da beskjeden m fram da $a_H(a_P(m)) = m$, mens E som også har tilgang til $a_P(m)$, men ikke kjenner funksjonen a_H , ikke klarer å få fram m .

Vi skal senere innføre det som kalles signatur, men la oss først gi eksemplet på hvordan dette skjemaet for RSA-offentlig-nøkkel-kryptografi uten signatur virker.

2. RSA-OFFENTLIG-NØKKEL-KRYPTOGRAFISYSTEMET

RSA-systemet er basert på to fakta: Ett erfaringsmessig faktum som sier at det ikke er enkelt å finne primtallsfaktoriseringen av store tall når primfaktorene er store. Det andre faktum er et resultat som går tilbake til Euler (1707-1783).

Vi skal bruke en algoritme av Euklid (ca 300 f.k.) som også er rask i bruk. Denne kan uttrykkes på følgende måte: for et positivt heltall N og helt tall M finnes entydige heltall q og r slik at $M = q \cdot N + r$ med $0 \leq r \leq N - 1$. Tallet r blir kalt resten av divisjon av M med N og vi vil betegne r med M_N i dette notatet. Merk at M_N alltid blir et tall i intervallet 0 til $N - 1$.

La oss si at deltager A plukker ut to primtall Q_A og R_A av størrelsesorden 10^{51} som vil være hemmeligheten H_A til deltaker A . Han ganger dem sammen og får tallet $N_A = Q_A \cdot R_A$ som er et tall med ca 102 sifre. Enhver besked som er et tall med 100 sifre ligger da i intervallet fra 0 til $N_A - 1$. A kjenner da tallene Q_A , R_A og kan derfor også enkelt beregne tallet $\phi(N_A) = (Q_A - 1)(R_A - 1) = N_A - Q_A - R_A + 1$. De som bare kjenner tallet N_A er ikke i stand til innenfor rimelig tid,

å finne dette tallet $\phi(N_A)$. Å finne tallet $\phi(N_A)$ er ekvivalent med å finne primtallsfaktoriseringen av N_A .

Før vi går videre må vi innom litt modulær aritmetikk. For et positivt heltall N betrakter vi heltallene i intervallet fra 0 til og med $N - 1$. Blant disse tallene introduserer vi først en operasjon $+_N$ ved at $x +_N y = (x + y)_N$ er resten en får når $x + y$ er dividert på N . Med andre ord $x +_N y = x + y$ dersom $x + y \leq N - 1$ og lik $x + y - N$ dersom $x + y \geq N$. Likeledes defineres operasjonen \cdot_N ved at $x \cdot_N y = (x \cdot y)_N$ er resten en får ved divisjon av $x \cdot y$ med N . Med andre ord $x \cdot_N y$ er det entydige bestemte heltallet i intervallet fra 0 til $N - 1$ som er slik at $x \cdot y = z \cdot N + x \cdot_N y$ for et heltall z .

For eksempel dersom $N = 11$ så er $5 +_{11} 7 = 1$ siden $5 + 7 = 12 = 1 \cdot 11 + 1$ og $5 \cdot_{11} 7 = 2$ siden $5 \cdot 7 = 35 = 3 \cdot 11 + 2$.

Vi innfører også eksponering med hensyn til N for et positivt heltall y og med x i intervallet 0 til $N - 1$ ved at vi lar $(x^y)_N = (\cdots ((x \cdot_N x) \cdot_N x) \cdot_N \cdots) \cdot_N x$, der faktoren x er gjentatt y ganger. Her gjelder nå de vanlige regneregler som gir at parentesene i uttrykket over kan sløyfes og vi får $((x^y)_N)^z = (x^{y \cdot z})_N = (x^{z \cdot y})_N$ og $(x^{y+z})_N = (x^y)_N \cdot_N (x^z)_N = (x^z)_N \cdot_N (x^y)_N$ for positive heltall y og z og med x i intervallet fra 0 til $N - 1$.

To hele tall x og y blir kalt relativt primiske dersom største felles divisor er 1.

Eulers resultat sier nå at dersom N er et positivt heltall og a er et positivt heltall som er relativt primisk til N , så er tallet $(a^{\phi(N)})_N = 1$ der $\phi(N)$ er antall tall mellom 1 og N som er relativt primisk til N .

For vår situasjon over med $N_A = Q_A \cdot R_A$ er det bare tallene $Q_A, 2 \cdot Q_A, \dots, (R_A - 1) \cdot Q_A, R_A, 2 \cdot R_A, \dots, (Q_A - 1) \cdot R_A$ i intervallet fra 1 til $N_A - 1$ som har felles faktor forskjellig med 1 med

N_A . Dette utgjør totalt $(R_A - 1) + (Q_A - 1)$ tall mellom 1 og $N_A - 1$ som har felles divisor ekte større enn 1 med N_A . Av dette får vi da

$$\phi(N_A) = N_A - 1 - (Q_A - 1) - (R_A - 1) = N_A - Q_A - R_A + 1,$$

som er et kjent tall for A , men ikke for de andre deltagerne. A kan da også med letthet finne et tall $t_A \geq 3$ som er relativt primisk med $\phi(N_A)$ og kan ved Euklidsk algoritme finne et positivt heltall s_A slik at $s_A \cdot t_A = z \cdot \phi(N_A) + 1$ med z et positivt heltall. Deltaker A publiserer nå N_A og t_A , men holder primtallene Q_A og R_A samt tallet s_A hemmelig. Funksjonen a_P er nå definert ved $a_P(x) = (x^{t_A})_{N_A}$ og den hemmelige funksjonen eller nøkkelen a_H vil være gitt ved $a_H(x) = (x^{s_A})_{N_A}$ som vi nå skal se. Vi får da at for alle x i intervallet 0 til $N_A - 1$ som er relativt primisk til N_A at

$$\begin{aligned} a_H(a_P(x)) &= (((x^{t_A})_{N_A})^{s_A})_{N_A} = (x^{t_A \cdot s_A})_{N_A} = (x^{z \cdot \phi(N_A) + 1})_{N_A} \\ &= (((x^{\phi(N_A)})_{N_A})^z)_{N_A} \cdot x_{N_A} = ((1^z)_{N_A} \cdot x)_{N_A} = x \end{aligned}$$

der en bruker Eulers resultat i den nest siste identiteten. En ser også fra denne regningen at $a_P(a_H(x)) = x$ når x er relativt primisk med N_A og ligger i intervallet fra 1 til $N_A - 1$.

A ber så dem som vil sende beskjedene m til ham om å beregne $a_P(m) = (m^{t_A})_{N_A}$, og sende denne beskjedene over det åpne nettet til ham. Merk at m nå ligger i intervallet 1 til $N_A - 1$, og at det finnes raske algoritmer for å beregne tallet $(m^{t_A})_{N_A}$. Uten kjennskap til primtallfaktoreringen kjenner ingen andre tallet $\phi(N_A)$ og derfor har de heller ikke noen enkel måte å finne tallet s_A på, som medfører at de ikke kjenner funksjonen a_H .

Når A får beskjedene $a_P(m)$ beregner han $a_H(a_P(m))$ og får ut m . (Her har vi brukt at m og N_A er relativt primiske. Vi kan anta dette da

sansynligheten for at m ikke er relativt primisk med N_A er så liten at den muligheten ser vi bort fra. Sannsynligheten for å treffe på en divisor i N_A ved et tilfeldig valgt tall i intervallet 1 til $N_A - 1$ er $(Q_A + R_A - 2)/(N_A - 1) \leq 10^{-50}$ for størrelsesorden av de primtall vi bruker her. Det er imidlertid også et faktum at siden N_A ikke inneholder samme primtall mer enn en gang i sin primtallsfaktorisering at $a_H(a_P(m)) = m$ for alle m i intervallet fra 0 til $N_A - 1$)

De som ikke kjenner tallet s_A og dermed ikke funksjonen a_H finner ikke ut av dette, og for dem vil beskjeden ikke komme fram som en meningsfylt beskjed.

3. RSA-OFFENTLIG-NØKKELE-KRYPTOGRAFI MED SIGNATUR

Dette RSA-skjemaet for hemmelig sending av beskjeder som vi tok for oss i forrige seksjon kan også utvides slik at mottaker A ikke bare kan lese beskjeden som er tiltenkt ham, men blir også sikker på at det er B som har sendt beskjeden.

Nå har altså A valgt sine hemmelighet H_A som består av primtallene Q_A og R_A , beregnet N_A , $\phi(N_A)$, valgt t_A og beregnet s_A og publisert N_A og t_A og dermed funksjonen $a_P(x) = (x^{t_A})_{N_A}$. Deltaker B gjør nå det samme: Velger sin hemmelighet B_H bestående av to primtall Q_B og R_B , beregner $N_B = Q_B \cdot R_B$ og $\phi(N_B)$, velger t_B slik at t_B og $\phi(N_B)$ er relativt primisk, og beregner s_B slik at $t_B \cdot s_B = z \cdot \phi(N_B) + 1$ for heltall z , og gjør N_B og t_B offentlig tilgjengelig og dermed også funksjonen $b_P(x) = (x^{t_B})_{N_B}$ offentlig tilgjengelig. B vil nå sende beskjeden m til A på en slik måte at ikke bare er A istand til å lese beskjeden, men vil også bli overbevist om at det er virkelig B som har sendt beskjeden, "senderen B signerer beskjeden."

Dette kan gjøres på følgende måte. Senderen B tar sin beskjed m , og samtidig ordner de to offentlig tilgjengelige tallene N_A og N_B etter størrelse. La oss si at $N_A \leq N_B$. B beregner da først $a_P(m) = (m^{t_A})_{N_A}$ og deretter $b_H(a_P(m)) = (((m^{t_A})_{N_A})^{s_B})_{N_B}$ som B sender til A . A vet også at $N_A \leq N_B$ så han beregner nå først $b_P(b_H(a_P(m))) = a_P(m)$ ved å bruke den offentlig tilgjengelig informasjon N_B og t_B fra B som skal til for å kjenne funksjonen b_P . Dette gir da $a_P(m)$ og deretter bruker A sin hemmelige funksjon a_H til å beregne $a_H(a_P(m))$ og får ut m som resultat. For å få m ut til slutt, måtte den første beregningen ha gitt $a_P(m)$. Men denne første beregningen gir dette resultatet bare dersom den opprinnelige beskjeden etter at den var kryptert med den offentlige nøkkel til A , var kryptert med den hemmelige nøkkelen til B , som jo bare B kjenner. Altså kan A lese beskjeden og han er også sikker på at bare den som kjenner den hemmelige nøkkelen til B kan ha sendt beskjeden.

Her tok vi utgangspunkt i at $N_A \leq N_B$. Dersom ulikheten går den andre veien, altså $N_B \leq N_A$ så beregner B først $b_H(m)$ og deretter $n = a_P(b_H(m))$ som han sender til A . A som også vet at $N_B \leq N_A$ dekrypterer ved først å beregne $a_H(n) = a_H(a_P(b_H(m))) = b_H(m)$. Deretter beregner han $b_P(b_H(m))$ som blir m .

4. PROBLEMER MED SIGNATUREN I SISTE AVSNITT

Er det sikkert at A kan være sikker på at den opprinnelige meldingen kom fra B og ikke fra E , eller er det mulig for E å "lure" A til å tro at beskjeden kom fra E ?

Dersom E også ser at $N_B \geq N_A$ vil E kunne bruke B s offentlige nøkkel b_P på beskjeden $b_H(a_P(m))$ og få fram $a_P(m)$. Dersom nå også E s offentlige tall N_E er større enn N_A kan E lage trøbbel. E kjenner

ikke innholdet, men vil sende denne beskjedden til A og innbilde A at den kommer fra E . Dette gjør E ved å kryptere med sin hemmelige nøkkel og sende $e_H(a_P(m))$. A må da dekryptere med E s offentlige nøkkel samt sin hemmelige nøkkel før beskeden kommer fram meningsfull for ham. A får da inntrykk av at beskjedden kommer fra E og ikke fra B . En kan illustrer dette ved at en bruk av offentlig nøkkel setter på et segl (eller klistre igjen en konvolutt) og så bruke sin hemmelige nøkkel ved signering blir som å signere utenpå seglet eller konvoluttet. Hvem som helst kan da skifte signaturen, men kan ikke endre innholdet uten å bryte seglet. Det er to måter å unngå dette på slik at en alltid signerer først og deretter krypterer med en offentlig nøkkel. Hver av deltagerne velger to sett av primtall. For eksempel B velger som før Q_B, R_B og i tillegg Q_B^S og R_B^S slik at $Q_B^S R_B^S = N_B^S \leq N_X$ for $X = A, B$ og E . B publiserer nå også N_B^S og tilhørende offentlig signeringsnøkkelen b_P^S . B vil nå sende en hemmelig beskjed til A og samtidig overbevise A at det virkelig er B som sender beskjedden. B tar da sin beskjed m krypterer den med sin hemmelige signaturnøkkel og får $b_H^S(m)$. Deretter tar han så den offentlige nøkkelen til A og bruker på dette resultatet og får fram $a_P(b_H^S(m))$ som han sender til A . Nå kan ikke E skifte signatur for å lure A til å tro at beskjedden kommer fra E .

En annen måte å gjøre dette på er at B bare beholder sitt ene nøkkel-par. B vil sende beskjedden m til A med signatur. B krypterer da beskjedden først med sin hemmelige nøkkel og får $b_H(m)$. Dersom $b_H(m) \leq N_A$ fortsettes som i seksjon 2. Dersom $b_H(m) \geq N_A$ bryter B denne i to $m_1 m_2$ og bruker A s hemmelige nøkkel på hver av delene. A bruker nå sin hemmelige nøkkel på hver av biten og får fram tallet m_1 og m_2 som han setter sammen til $m_1 m_2$ før han bruker B s offentlige nøkkel

på resultatet, og får fram m . denne siste metoden er kanskje enklest i bruk da det blir færre nøkler å holde orden på.

5. KVITTERING

Kanskje avsender B i forrige seksjon også vil ha en bekreftelse på at A har mottatt beskjednen. Et oppsett for dette kan da være følgende: B sender meldingen m til A ved å sende den signerte og krypterte meldingen $a_P(b_H(m))$ til A . A dekryptere meldingen fra B og verifiderte at den kom fra B . Nå kan A for eksempel skrive meldingen speilvent, kryptere den med sin hemmelige nøkkel og deretter med B offentlige nøkkel og sende resultatet til B . B dekryptere, og ser at den dekrypterte meldingen passer med den speilvente. B verifiserer således at den meldingen som er sendt tilbake til ham kan bare være sendt av en som kan ha lest meldingen m , og det er det bare den som kjenner den hemmelige nøkkelen til A som kan ha gjort. Det er nemlig vanskelig å produsere den krypterte av den speilvendte meldingen uten først å finne meldingen m . Deltagerne må på forhånd ha avtalt som en felles signeringsmåte at den speilvente skal sendes tilbake som kvittering. Dette er selvsagt bare et eksempel på hvordan dette kan gjøres.

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY, DEPARTMENT OF
MATHEMATICAL SCIENCES, N-7491 TRONDHEIM, NORWAY.

E-mail address: sverresm@math.ntnu.no