RSA Kryptosystemet

Kryptologi ved Datalogisk Institut, Aarhus Universitet

1 Kryptering med RSA

Her følger først en kort opridsning af RSA kryptosystemet, som vi senere skal bruge til at lave digitale signaturer.

Generering af nøgler. Et sæt RSA nøgler laves på følgende måde:

1. Vælg to primtal p og q (de er typisk meget store, over 150 cifre)
2. Beregn $N = p \cdot q$ og $\phi(N) = (p-1) \cdot (q-1)$
3. Vælg et heltal e så $0 < e < \phi(N)$ og sådan at e og $\phi(N)$ er indbyrdes primiske[1]
4. Beregn d så $e \cdot d \equiv 1 \mod \phi(N)$

Den *offentlige nøgle* er (N, e) og den *hemmelige nøgle* er (N, d).

Vi kan se den offentlige nøgle som hængelåsen alle kan bruge til at kryptere ("låse") beskeder med. Den hemmelige nøgler kan omvendt dekryptere ("åbne") beskeder.

Kryptering. Hvis Alice vil sende en krypteret besked til Bob, og hun kender Bobs offentlige nøgle (N, e), kan hun gøre følgende.

Lad beskeden m være et heltal mindre end N. Krypteringen af en besked m skrives som $Enc(m)$ og er givet ved

$$c = Enc(m) = m^e \mod N$$

hvor resultatet, benævnt med c, kaldes en *ciffertekst*.

[1] Indbyrdes primiske betyder at kun 1 går op i begge tal, altså at største fælles divisor mellem de to tal er 1

Dekryptering. Når Bob, som kender den tilsvarende hemmelige nøgle (N, d), vil dekryptere en ciffertekst c kan han beregne

$$m = Dec(c) = c^d \mod N$$

for at får den oprindelige besked m tilbage.

2 Digitale Signaturer

Indtil nu har vi beskæftiget os med kryptering, det vil sige det at kunne sende en besked således at indholdet forbliver hemmeligt for alle andre end modtageren. Dette kaldes også for *konfidentialitet* - at holde noget hemmeligt. En anden lige så vigtig del af kryptologi er *autentifikation* - at bevise over for andre at man er den man påstår man er og at bevise at en besked kommer uændret frem.

Lad os kort overveje hvordan ganske almindelige papir-underskrifter (signaturer) virker, eller i hvert fald burde virke. Min underskrift på et dokument betyder at:

1. Det kan afgøres, at det er mig personligt, der har skrevet under.
2. Jeg er juridisk bundet af indholdet.
3. Modtageren af dokumentet kan vise det til en tredjepart, som kan verificere ovenstående.

Vi kan forsøge at implementere den digitale version af underskrifter på følgende måde, der ligner papir-udgaven: når Alice ønsker at underskrive et dokument vedhæfter hun en indscanning af sin underskrift til dokumentet, og sender det til Bob.

Opgave 1: Hvorfor er ovenstående løsningsforslag oplagt en meget dårlig ide?

Som opgaven antyder er det meget vigtigt at underskriften på en eller anden måde både hænger sammen med indholdet af dokumentet og vedkommende der skriver under. Her kan RSA hjælpe os.

Opgave 2: Argumenter for at der for RSA gælder:

$$m = Dec(Enc(m)) = Enc(Dec(m))$$

Brug nu dette til at vise hvordan Alice kan sende en (kærligheds-)besked m til Bob, således at Bob kan verificere, at den virkelig kommer fra Alice, samt kan bevise dette over for en tredje person.

Opgave 3: Alices værste fjende Eva er vild med Bob og ville utrolig gerne erstatte meddelelsen m med en anden meddelelse m' hvor Alice siger farvel for altid til Bob. Argumenter for hvorfor det ikke er muligt for Eva.

Opgave 4: Lad den offentlige RSA nøgle være ($N = 33$, $e = 3$) og den hemmelige nøgle ($N = 33, d = 7$). Hvad er underskriften på meddelelsen $m = 2$?

Ovenfor har vi stiltiende antaget at Bob allerede har den offentlige nøgle han skal bruge for at checke Alices underskrift. I praksis er den nøgle naturligvis nødt til at komme et eller andet sted fra, f.eks. kan det være Bob på sin maskine har en database hvor han kan slå offentlige nøgler op for dem han kommunikerer med. Stort set det samme som telefonnummerlisten i en mobiltelefon.

Opgave 5: Antag at Bob opbevarer offentlige nøgler i en database som beskrevet ovenfor på sin PC. Alice og Bob er stadig forelskede, og Eva er præcis lige så jaloux som før. Bob har offentlige nøgler for både Alice og Eva på sin liste. En nat bryder Eva ind på Bobs PC, og får adgang til at se og evt. manipulere med Bobs liste med offentlige nøgler uden at Bob opdager noget. Beskriv hvad hun kan gøre for at ødelægge forholdet mellem Alice og Bob. Drag en generel konklusion omkring hvordan offentlige signatur-nøgler skal behandles.

3 Deling af den hemmelige nøgle

Tallene for RSA nøglerne i foregående opgave var meget små. I praksis er tallene der skal bruges dog så store at de på ingen måde kan huskes i hovedet. Man er derfor nødt til at opbevare den hemmelige nøgle et eller andet sted, og der er derfor en vis risiko for at den kan blive stjålet. Vi skal nu se nærmere på en teknik til at dele den hemmelige nøgle op i to for derved at reducere risikoen ved tyveri.

Lad (N, d) være en hemmelig nøgle. Vi kan nu dele d ved at vælge et tal d_1 helt tilfældigt, og bagefter beregne $d_2 = d - d_1$. Dette betyder at vi nu har to tal d_1, d_2 hvorom det gælder at

$$d_1 + d_2 = d$$

og som kan opbevares to forskellige steder, f.eks. på to forskellige computere. I eksemplet fra opgave 1, hvor $d = 7$, kunne vi f.eks. have $d_1 = 17$ og $d_2 = -10$.

Opgave 6: Argumenter for at hvis nogen får fat i enten d_1 eller d_2, men ikke dem begge, så har vedkommende stadig ingen anelse om hvad d er.

For er lave en underskrift på beskeden m er ideen nu at den computer der har d_1 beregner $m^{d_1} \mod N$ mens computeren med d_2 tilsvarende beregner $m^{d_2} \mod N$. Dette giver to "halve signaturer" der kan samles:

Opgave 7: Vis at man kan beregne signaturen $m^d \mod N$ ud fra de to bidrag $m^{d_1} \mod N$ og $m^{d_2} \mod N$ på følgende måde:

$$m^d \mod N = (m^{d_1} \mod N) \cdot (m^{d_2} \mod N) \mod N$$

(**Hint**: $(a \mod N) \cdot (b \mod N) \mod N = (a \cdot b) \mod N$.)

4 Beregning af store potenser

Når man skal regne på de meget store tal der i virkeligheden bruges til RSA, er det vigtigt at bruge de mest effektive metoder. Selvom computere er hurtige, går det alligevel alt for langsomt hvis vi ikke tænker os om når vi programmerer dem. Forestil jer for eksampel at I skal beregne $23^{17} \mod 33$ med blyant og papir eller med en lommeregner der kun kan de grundlæggende regningsarter. Hvis man bare går i gang uden at tænke, ville man måske gøre som antydet her:

$$23^{17} \mod 33 =$$
$$23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \cdot 23 \mod 33$$

altså bare gange 23 med sig selv 17 gange, for derefter at reducere modulo 33. Det virker som en temmelig stor opgave, men er heldigvis helt unødvendigt

Opgave 8: Argumenter for at

$$\begin{aligned}
23^{17} \mod 33 &= (23^{16} \mod 33) \cdot 23 \mod 33; \\
23^{16} \mod 33 &= (23^{8} \mod 33) \cdot (23^{8} \mod 33) \mod 33; \\
23^{8} \mod 33 &= (23^{4} \mod 33) \cdot (23^{4} \mod 33) \mod 33; \\
&\cdots \text{ (fortsæt selv her)}
\end{aligned}$$

og brug dette til at forklare hvordan $23^{17} \mod 33$ kan beregnes med kun 5 multiplikationer og divisioner.

Opgave 8 er et eksempel på en generel teknik der hedder "square and multiply" (opløft til anden, og gang sammen), som bruges i alle computerprogrammer der anvender RSA. Den er en helt nødvendig forudsætning for at RSA kan bruges til noget i praksis.

For de skarpe knive i skuffen kommer her en opgave der går tættere på denne metode. I kan eventuelt springe denne opgave over i første omgang og vende

tilbage hvis der er tid. I opgaven kræves det at man ved at ethvert positivt tal kan skrives som en sum af 2-potenser (1, 2, 4, 8, osv.):

$$
\begin{aligned}
&1\\
&2\\
&3 = 2 + 1\\
&4\\
&5 = 4 + 1\\
&6 = 4 + 2\\
&7 = 4 + 2 + 1\\
&8\\
&9 = 8 + 1\\
&\vdots
\end{aligned}
$$

Dem der kender til binære tal kan måske se hvorfor, men det behøver man ikke at vide noget om i denne opgave:

Opgave 9: *(Spring evt. over i første omgang.)* Brug ovenstående fact, og ideen i Opgave 8, til at designe en metode der beregner $a^e \mod N$ for vilkårlige positive tal a, e, N. Hvor mange multiplikationer og divisioner skal man bruge med jeres metode hvis e = 1026? Bemærk at den naive metode bruger 1025 multiplikationer.

5 Usikker brug af et sikkert kryptosystem

Alice har hørt at det er meget svært at bryde RSA kryptosystemet, hvorfor hun vælger at bruge RSA til at sende en besked til Bob. De gør det på følgende måde:

- Bob genererer et nøglesæt og sender den offentlige nøgle (N, e) til Alice
- Alfabetets bogstaver nummereres så A = 0, B = 1, ... , Å = 28.
- Hvert enkelt bogstav i beskeden krypteres hvert for sig med Bobs offentlige nøgle, og de krypterede bogstaver sendes til Bob.
- Bob kan med sin hemmelige nøgle dekryptere de modtagne bogstaver og læse beskeden.

Opgave 10: Antag at nøglen sendt fra Bob til Alice er ($N = 836201, e = 768641$). Forestil dig at du er Eva og at du opsnapper beskeden:

399644, 407526, 424845, 407526,
391559, 598468, 138351, 407526,
625324, 727185, 530887, 424845.

Forsøg nu at dekryptere beskeden. Hvis I har husket jeres TI lommeregner kan I nok let faktorisere N, som stadig er alt for lille, men prøv at finde ud af hvordan man let kan bryde krypteringen, også selvom man ikke kan faktorisere N.

(**Hint**: Hvordan vil beskeden "E" komme til at se ud i krypteret tilstand?)

En teknik til at modvirke dette problem er, at sætte hemmeligheden sammen med noget tilfældighed før kryptering: man holder hemmeligheden i de mindste par cifre, og "fylder op" med tilfældige tal i de andre cifre indtil man har et tal med lige så mange cifre som N. Hvis man kan dekryptere, er de tilfældige tal nemme at fjerne igen. F.eks. kunne E = 4 laves om til 437**04** før det krypteres, så kun de to sidste cifre indeholder beskeden. Når man gør dette krypteres den samme besked (det samme bogstav) ikke altid til det samme tal, og systemet kan derfor ikke brydes så let.

6 Andre egenskaber ved RSA

Vi har nu set to anvendelser af RSA kryptosystemet, kryptering og digital signatur, og har argumenteret for dets sikkerhed. Som vi bemærkede i afsnit 5 skal man dog stadig tænke sig om inden man bruger RSA.

Lad os antage at en bank tilbyder deres kunder at udstede checks til hinanden vha. digitale signaturer. Mere præcist, lad os antage at Alice har en offentlige nøgle (N, e) der er kendt af alle, samt en hemmelig nøgle der kun er kendt af hende. Banken tillader Alice at udstede en check til Bob på m kroner ved, at hun simpelthen sender ham en signatur $s = Dec(m)$. Når Bob dukker op i banken med s, udbetaler de m kroner fra Alices konto hvis s er en korrekt signatur i forhold til (N, e). Banken kontrollerer naturligvis også at de ikke har set s før; ellers kunne Bob blot dukke op med samme s dagen efter og få udbetalt m kroner igen.

Siden Alice er den eneste der kender hendes hemmelige nøgle, må det betyde at hun har godkendt udbetalingen. Desværre viser dette sig ikke at være tilfældet:

Opgave 11: Vis hvordan en korrupt Bob kan få udbetalt meget mere af Alices formue end hun havde tiltænkt. (**Hint**: Se hintet til opgave 7.)

Et tilsvarende problem opstår, hvis vi naivt bruger RSA til at afvikle en elektronisk auktion. Lad os antage at Alice har sat en dyr antik vase til salg til højstbydende og at både Bob og Eva er interesserede i at købe den. Hverken Bob eller Eva er interesserede i at den anden ved, hvor meget de har tænkt sig at byde, så de bliver alle enige om at sende deres bud krypteret til Alice. Hvis Eva gerne vil være sikker på at overbyde Bob, kunne hun naturligvis blot byde meget højt; det viser sig dog ikke at være nødvendigt:

Opgave 12: Antag at Eva har opsnappet Bobs ciffertekst $c_B = Enc(x_B)$. Vis da hvordan Eva kan indsende et bud til Alice der er nøjagtigt det dobbelte af hvad Bob har budt.

Opgave 13: *(Spring evt. over i første omgang.)* Forbedr opgave 12 og vis hvordan Eva kan indsende et bud til Alice der er nøjagtigt 1% højere end Bobs. (**Hint**: Antag at Bobs bud altid er et multiplum af 100.)

Problemet illustreret i opgave 12 og 13 kan løses ved at kræve at Eva faktisk kender den besked hun krypterer. Én måde at gøre dette på er vha. *zero-knowledge protokoller*, der tillader en part at vise sandhed af et udsagn uden at røbe andet end at det er sandt. I auktionen ovenfor kan Alice kræve at parterne beviser at de kender deres bud; rent intuitivt vil dette forhindre Eva i at snyde fordi hun ikke kan dekryptere Bobs ciffertekst og derfor ikke ved hvad hun selv byder.

I afsnit 3 så vi en fordel ved at der for RSA gælder at

$$Dec(m_1) \cdot Dec(m_2) = Dec(m_1 \cdot m_2)$$

og vi har nu også set nogle ulemper ved at der for RSA også gælder at

$$Enc(m_1) \cdot Enc(m_2) = Enc(m_1 \cdot m_2).$$

Et kryptosystem hvori den sidstnævnte lighed holder kaldes for *homomorfisk*. Vi skal nu se at der også kan være fordele ved et sådanne kryptosystem. F.eks. kan man bruge det til at lave *sikre beregninger*, dvs. beregninger på krypteret data.

Eksempelet vi skal kigge på er elektronisk afstemning, hvor en stemme enten er Ja eller Nej. For at finde resultatet af en afstemning kan man "lægge stemmer samme" vha. OG-operatoren der har følgende regneregler

$$\begin{aligned} \text{Nej OG Nej} &= \text{Nej} \\ \text{Nej OG Ja} &= \text{Nej} \\ \text{Ja OG Nej} &= \text{Nej} \\ \text{Ja OG Ja} &= \text{Ja} \end{aligned}$$

og som intuitivt siger at resultatet af en afstemning med to stemmer er Ja udelukkende hvis begge stemmer er Ja; hvis en (eller begge) stemmer Nej er resultatet Nej.

En alternativ måde at finde resultatet på, er ved at tolke stemmer som tal

$$\begin{aligned} \text{Nej} &= 0 \\ \text{Ja} &= 1 \end{aligned}$$

og efterfølgende regne på disse:

Opgave 14: Givet to stemmer $x_1, x_2 \in \{\mathsf{Nej}, \mathsf{Ja}\}$ hvordan kan vi da finde resultatet af afstemning x_1 OG x_2 ved at regne på tal i stedet? (**Hint**: Hvad svarer OG-operatoren til?)

Vi kan bruge observationen i opgave 14 til at bygge en sikker afstemningsprotokol. Antag at Alice, Bob, og Caroline ønsker at stemme om hvorvidt de skal tage i biografen. De bliver enige om kun at tage afsted hvis de alle tre gerne vil. Samtidigt ønsker de at ingen skal føle et gruppepres, og vil derfor gerne kunne stemme på en sikker måde, hvor ingen ved hvad de hver især har stemt. Det vil sige, at hvis de bliver enige om ikke at tage afsted, så ved f.eks. Alice, der gerne ville i biografen, ikke om det var Bob eller Caroline der hellere ville noget andet.

Opgave 15: Lad $x_A, x_B, x_C \in \{\mathsf{Nej}, \mathsf{Ja}\}$ være hhv. Alices, Bobs, og Carolines stemme for eller imod at tage i biografen. Beskriv hvordan et homomorfisk kryptosystem kan hjælpe dem med på en sikker måde at afgøre om de skal tage i biografen eller ej, dvs. finde resultatet af afstemning $(x_A$ OG $x_B)$ OG x_C. (**Hint**: Hvis vi tolker de tre stemmer som tallene $b_A, b_B, b_C \in \{0, 1\}$ skal vi blot udregne $(b_A \cdot b_B) \cdot b_C$ sikkert.)

Opgave 16: Som vi så først i dette afsnit er RSA et homomorfisk kryptosystem. Dog så vi også i afsnit 5 at der skal tilføjes noget tilfældighed før RSA er sikkert at bruge. Hvilke problemer opstår der så ved at bruge RSA som kryptosystem i afstemningsprotokollen fra opgave 15?

Vi har nu set hvordan vi sikkert kan regne med OG-operatoren. Det er måske ikke så overraskende, at der er grænser for hvad vi kan gøre, ved kun at bruge denne operator. Det viser sig dog, at hvis vi også kan finde ud af at regne sikkert med ELLER-operatoren:

$$\begin{aligned} \mathsf{Nej}\ \text{ELLER}\ \mathsf{Nej} &= \mathsf{Nej} \\ \mathsf{Nej}\ \text{ELLER}\ \mathsf{Ja} &= \mathsf{Ja} \\ \mathsf{Ja}\ \text{ELLER}\ \mathsf{Nej} &= \mathsf{Ja} \\ \mathsf{Ja}\ \text{ELLER}\ \mathsf{Ja} &= \mathsf{Ja}, \end{aligned}$$

så kan vi faktisk sikkert udregne alt hvad en computer kan! Antag at vi har et homomorfisk kryptosystem der både kan multiplicere og addere ciffertekster (sådanne kryptosystemer findes). Antag også at vi har et program P der tager $x_1, \ldots, x_n$ som input og returnerer $y = f(x_1, \ldots, x_n)$ som output. Nu kan vi så lave et program P' der i stedet tager $Enc(x_1), \ldots, Enc(x_n)$ som input og returnerer $Enc(y)$ som output, og som ikke på noget tidspunkt kommer til at kende $x_1, \ldots, x_n$ eller y.