DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering:

Lektor i realfag 8.-13.

Vårsemesteret, 2022

Åpen

Forfatter: Olav Hansen Hjellup

.....
(signatur forfatter)

Fagansvarlig: Alex Bentley Nielsen

Veileder(e): David Ploog

Tittel på masteroppgaven: RSA-kryptering med diskusjon om bruk av det i skolen

Engelsk tittel: RSA-encryption with discussion about applying it in school

Studiepoeng: 30 STP

Emneord:

Kryptografi, Kryptering, RSA kryptering, Angrep på RSA kryptering, Modulær aritmetikk, Kryptering i skolen

Sidetall: 36

+ vedlegg/annet: 2

Stavanger, 23.05 /2022

RSA-kryptering med diskusjon om bruk av det i skolen

Olav Hansen Hjellup

En oppgave presentert for graden
Lektor i realfag 8.-13.

Det Teknisk-Naturvitenskapelige Fakultet
Universitetet i Stavanger
Norge
23. mai 2022

Sammendrag

Masteroppgaven gir en grunnleggende introduksjon av RSA kryptering, og angrep på RSA kryptering. Vi presenterer, modulær aritmetikk, tallteorien som er nødvendig for å forstå RSA kryptering, og en generell innføring i hva kryptografi er, og introduserer ulike prinsipper for kryptering. Vi ser på de matematiske prinsippene bak RSA-systemet, og viser hvorfor det er 'sikkert'. Vi ser på måter for å oppnå suksessfulle 'real life' angrep på RSA-systemet. Tilslutt sees det på hvordan dette passer inn i undervisningen i skolen.

Innhold

Innledning

I det digitale universet er vi omgitt av kryptering. Majoriteten av befolkningen har liten kjennskap til hverken hva kryptografi er, omfanget av det eller hvilken betydning det har for vår digitale aktivitet. I studiet av kryptografi er det mange ulike retninger man kan fordype seg i. Jeg har valgt å fordype meg i RSA kryptering. Vi vil gå gjennom det grunnleggende i RSA-kryptosystemet, og se på enkelte vellykkede angrep på RSA. RSA kryptering baserer seg på tallteori og modulær aritmetikk, og som en forberedende del vil vi gå gjennom relevant teori i kapittel 1. RSA kryptering er en offentlig-nøkkel kryptering og for å opparbeide en forståelse av hva RSA kryptering er, presenterer vi i kapittel 2 en generell innføring i hva kryptografi er, og en kort introduksjon i ulike prinsipper/metoder for kryptering. Deretter går vi gjennom de matematiske hovedprinsippene til RSA-krypering, og viser hvorfor det er 'sikkert'. I masteroppgaven tar kapittel 1, modulær aritmetikk og kapittel 3, RSA - Kryptosystem utgangspunkt i artikkelen *Introduction to Public Key Cryptography and Modular Aritmetic* av Milson [13]. Etter introduseringen av RSA-kryptosystemet vil vi i kapittel 4, Angrep på RSA - kryptosystem se på noen av angrepene beskrevet av Boneh [3] i artikkelen *Twenty Years of Attacks on the RSA Cryptosystem*. Angrepene vi ser på er de elementære angrepene 'felles modulus' og 'blending' sammen med 'Wieners angrep' på lav privat eksponent og 'Franklin-Reiter angrep på relaterte meldinger' med lav offentlig eksponent. Til slutt i oppgaven vil vi vurdere om dette kan være relevant i matematikkundervisningen for 8.-13. trinn, gjennom å se på hvilke erfaringer andre lærere har gjort seg, samt se på emnet i lys av læreplanen Kunnskapsløftet 2020 [15].

1 Modulær aritmetikk

Dette kapitlet tar utgangspunkt i artikkelen *Introduction to Public Key Cryptography and Modular Aritmetic* av Milson [13]. Store norske leksikon [1] sier at 'Aritmetikk er læren om tallenes egenskaper og metoder til å regne med tall.'. Aritmetikk omfatter de fire regneartene. Også brøkregning, rotutdragning, logaritmer og potensregning er en del av aritmetikken.

Modulær aritmetikk er aritmetikk for heltall, og går under den matematiske grenen tallteori. I tillegg kalles denne type aritmetikk for klokkearitmetikk ettersom tallrekken er begrenset. Ett av de aller mest brukte eksemplene i modulær aritmetikk er klokken. Hvis vi teller hele timer på klokken, teller vi $1, 2, 3, ..., 10, 11, 12, 1, 2, 3, ...$ Vi kaller antall siffer som opptrer i en slik tallrekke for modulusen til tallrekken. På den analoge tolvtimersklokken er derfor modulus 12. Det vil si at 18:00 om ettermiddagen tilsvarer 06:00 om morgenen, vi vil altså si at 18 er ekvivalent med 6 modulus

12. Størrelsen på modulus n kan være ethvert heltall større enn 1. Et eksempel på modulus fra hverdagen er ukedagene som har modulus 7.

1.1 Kongruensregning

Tall som er ekvivalente i en modulus kalles kongruente tall. Kongruens modulus n er en ekvivalensrelasjon og klasseinndeling av heltallene. Alle heltall hvor ulikheten mellom dem er et multippel av n, vil være ekvivalente i modulus n. Eksempelvis er -11,1, 13, 25 ekvivalente i modulus 12:

$$\begin{aligned} -11 &= -1 \cdot 12 + 1 \\ 1 &= 0 \cdot 12 + 1 \\ 13 &= 1 \cdot 12 + 1 \\ 25 &= 2 \cdot 12 + 1 \end{aligned}$$

At to tall, a og b, er kongruente modulus n betyr at de gir samme rest ved divisjon med n, vi skriver $a \equiv b\,(mod\ n)$. Kongruensrelasjonen er refleksiv, symmetrisk og transitiv. $\mathbb{Z}/n\mathbb{Z}$ betegner mengden av kongruens (=rest) klassene.

Merk 1.1. To tall, a og b, er kongruente modulus n, $a \equiv b\,(mod\ n)$, hvis $a - b = kn$, hvor $a, b, k, n\ \in \mathbb{Z}$.

Ethvert heltall kan reduseres til et kongruent tall som ligger mellom 0 og $n-1$. Hvert av disse tallene representerer en kommutativ ring av restklasser for hele tall modulus n. Et annet ord som brukes for restklasse er *redusert form*. I modulus 4 er $0, 1, 2, 3$ hver sin restklasse, og alle andre heltall vil være kongruente til en av disse restklassene.
For å finne ut hvilken restklasse et heltall tilhører kan vi dividere tallet med modulusen, hvor 'resten' gir oss rest-tallet. La oss finne den reduserte formen til 4080 i modulus 378:

$$4080 \div 378 = 10\frac{300}{378} \longrightarrow 4080 = 10 \cdot 378 + 300$$

Vi ser at $4080 \equiv 300\,(mod\ 378)$, og derfor tilhører 4080 restklassen 300 i modulus 378.

Addisjon og subtraksjon i modulær aritmetikk er nokså enkel. Vi illustrerer dette gjennom eksempler i modulus 12. I addisjonen 4 + 5, starter en med fire og øker med fem og får dermed ni. I addisjonen 9 + 4, starter en med ni og øker med fire og får dermed tretten. Ettersom $13 \equiv 1$ i $\mathbb{Z}/12\mathbb{Z}$, er 13 ikke er et tall på den reduserte formen i modulus 12. Vi uttrykker svaret som den ekvivalente til 13 i modulus 12: $9 + 4 = 13 \equiv 1\,(mod\ 12)$. Subtraksjon følger den samme logikken: $9 - 4 = 5$ og

$1 - 4 = -3 \equiv 9 \, (mod \ 12)$.

I aritmetikk er multiplikasjon repetert addisjon, og slik fungerer multiplikasjon også i modulær aritmetikk. Hvis vi ønsker å kalkulere $3 \cdot 4$, altså $3 + 3 + 3 + 3$ i modulus 10, så er svaret 12. 12 er kongruent med 2 i modulus 10, og vi kan skrive det slik: $3 \cdot 4 \equiv 2 \, (mod \ 10)$.

Distribusjonslovene for multiplikasjon fungerer også ved kongruensregning. La oss vise det ved et eksempel i modulus 10.

$$(7 + 4) \cdot 3 = 11 \cdot 3 \equiv 1 \cdot 3 \equiv 3 \, (mod \ 10)$$
$$7 \cdot 3 + 4 \cdot 3 = 21 + 12 \equiv 1 + 2 \equiv 3 \, (mod \ 10)$$

Hva med $2 \cdot 13$ og $2 \cdot 3$ i modulus 10, er disse kongruente? Siden 13 og 3 er kongruente modulus 10, så vil produktene også være kongruente. Produktene er ekvivalente ettersom ulikheten mellom 13 og 3 er lik en multippel av 10. $13 = 10 + 3$ og vi får:

$$2 \cdot 13 = 2 \cdot (10 + 3) = 2 \cdot 10 + 2 \cdot 3$$

$2 \cdot 10$ er en multippel av 10 og er derfor kongruent med $0 \, (mod \ 10)$:

$$2 \cdot 13 = 2 \cdot 10 + 2 \cdot 3 \equiv 2 \cdot 3 + 0 \, (mod \ 10)$$

Modulær multiplikativ invers til et heltall a modulus n er et heltall b slik at $a^{-1} \equiv \ b$ i $\mathbb{Z}/n\mathbb{Z}$.

Vi kan vise dette med den multiplikativt inverse til 3 modulus 5:

$$3 \cdot 2 \equiv 6 \equiv 1 \, (mod \ 5) \rightarrow 3^{-1} = 2 \text{ i modulus } 5$$

Divisjon i modulær aritmetikk er, ikke alltid, tillatt, fordi enhver divisjonsoperasjon kan sees på som multiplikasjon mellom telleren og den multiplikative invers til nevneren.

$$\frac{a}{b} = ab^{-1}$$

Her er et eksempel på divisjon i kongruensregning:

$$\frac{2}{3} \equiv 2 \cdot 3^{-1} \equiv 2 \cdot 2 \equiv 4 \, (mod \ 5)$$

Vi kan sjekke at dette stemmer ved å multiplisere med 3 på begge sider av uttrykket:

$$2 \equiv 4 \cdot 3 \, (mod \ 5)$$

Siden 12 er kongruent med 2 modulus 5 vet vi at dette stemmer.

Spørsmål 1.2. *Hvordan finner vi multiplikativ invers til a i modulus n?*

Vi kan finne multiplikativ invers ved bruk av Euklids utvidede algoritme som beskrives i kapittel 1.2.2.

Spørsmål 1.3. *Hvordan dividere med en nevner som ikke har en multiplikativ invers?*

Ettersom divisjon i modulær aritmetikk er definert som multiplikasjon mellom teller og multiplikativ invers til nevner, så er det ikke mulig å dividere med en nevner som ikke har en multiplikativ invers i $\mathbb{Z}/n\mathbb{Z}$.

Definisjon 1.4. Et tall a er en **divisor av null**, hvis det finnes et tall $b \neq 0$ slik at $a \cdot b \equiv 0\,(mod\ n)$, hvor $a,\ b \in \mathbb{N}$

I restklasser av hele tall modulus 4, $\mathbb{Z}/4\mathbb{Z}$, så er restklassen 2 en divisor av null siden $2 \cdot 2 \equiv 4 \equiv 0\,(mod\ 4)$.

Definisjon 1.5. Et tall a kalles et **inverterbart tall** i modulus n hvis og bare hvis det har en multiplikativ invers i modulus n.

Setning 1.6. *$a \in \mathbb{Z}$ inverterbart tall $\mathbb{Z}/n\mathbb{Z} \iff a$ er ikke en divisor av null i $\mathbb{Z}/n\mathbb{Z}$*

Bevis. I modulær aritmetikk kan et inverterbart tall aldri være en divisor av null, og en divisor av null kan aldri være et inverterbart tall. Dette kommer av at hvis et tall a er et inverterbart tall så vil det per Definisjon 1.5 eksistere et tall b slik at $a \cdot b \equiv 1$. Hvis $c \neq 0$, kan ikke $a \cdot c$ bli null, fordi $b \cdot a \cdot c \equiv 1 \cdot c \equiv c$.
Hvis a er en divisor av null, så er $a \cdot b \equiv 0$ med $b \neq 0$. Det vil ikke eksistere en c slik at $c \cdot a \equiv 1$ fordi $c \cdot a \cdot b$ vil være ekvivalent med $1 \cdot b \equiv b$. Men vi vet at $c \cdot a \cdot b \equiv 0$, så vi vet at et tall aldri kan være både et inverterbart tall og en divisor av null. □

1.2 Største felles nevner

Den største felles nevneren til a og b er det største mulige tallet som gir heltallssvar når a og b divideres på tallet. Hvis vi for eksempel ser på tallene 12 og 18, er det tydelig at 2 er en felles nevner. Både 3 og 6 er også felles nevnere, hvor det er 6 som er den største felles nevneren. Vi skriver den største felles nevneren mellom to tall som GCD(12,18)=6. GCD er en forkortelse og står for 'Greatest Common Divisor'.

Merk 1.7. 1 er en felles nevner for ethvert par av tall. Hvis 1 er den eneste felles nevneren til tallene a og b, sier vi at a og b er **relativt primiske**.

Siden GCD(9,16)=1 er 9 og 16 relativt primiske.
Felles nevner er et viktig konsept i modulær aritmetikk, fordi det er et nøkkelkonsept for å forstå divisorer av null.

Fakta 1.8. Hvis tall a og n har en annen felles divisor enn 1, så er a en divisor av null i modulus n.

Siden 3 er en felles nevner til tallene 6 og 9, er 6 en divisor av null i modulus 9, $6 \cdot 3 = 18 \equiv 0\,(mod\ 9)$. For å vise hvorfor Fakta 1.8 er korrekt, la $d > 1$ være en felles nevner av a og n. Både $a \div d$ og $n \div d$ er heltall og derfor:

$$a \cdot (n \div d) = (a \div d) \cdot n \equiv 0\,(mod\ n)$$

Det finnes en systematisk tilnærming for å kalkulere hva som er største felles nevner til to heltall, denne prosedyren kalles Euklids algoritme.

1.2.1 Euklids algoritme

Euklids algoritme er basert på følgende ide. Anta at målet er å finne GCD av to heltall x og y. Hvis det eksisterer et tall d som er fellesnevner til x og y, så vil d også kunne dele følgende tall: $x - y, x - 2y, x - 3y$ og så videre. Faktisk så vil d kunne dele ethvert tall på formen $x - p \cdot y$, hvor p er et heltall. I motsatt tilfelle, hvis d kan dele y og $x - p \cdot y$, så vil d også kunne dele x. Konklusjonen er at for hvert hele tall p, så er GCD av x og y lik GCD av y og $x - p \cdot y$.

Dersom vi ønsker å finne GCD av 168 og 91, kan vi like gjerne kikke på GCD av 91 og $168 - 1 \cdot 91 = 77$. Vi repeter tankegangen som beskrevet i avsnittet ovenfor, og ser på GCD av 77 og $91 - 1 \cdot 77 = 14$. Deretter ser vi på GCD av 14 og $77 - 5 \cdot 14 = 7$. Det er klart at $14 = 2 \cdot 7$, som betyr at 7 er den største felles nevneren til 168 og 91.

Metode 1.9. *Mål: Finne GCD av heltallene $x > y$.*

- *Hvis $x \div y$ gir et heltall, så er GCD(x,y)=y.*
- *Hvis ikke, regner vi ut heltallssvaret, p, av $x \div y$.*
- *Deretter finner vi rest-tallet, r, gitt ved $r = x - p \cdot y$.*
- *Siden GCD(x,y)=GCD(y,r), går vi tilbake til start og gjør kalkuleringene på nytt, med den nye x lik den gamle y og den nye y lik den gamle r.*

Siden $y < x$ og $r < y$, vil tallene som settes inn i algoritmen bli mindre med hver gjennomgang. Derfor vil Euklids algoritme slutte etter et endelig antall repetisjoner.

Her bruker vi Euklids algoritme til å finne GCD(1113,504):

$$\begin{aligned} 1113 &= 2 \cdot 504 + 105 \\ 504 &= 4 \cdot 105 + 84 \\ 105 &= 1 \cdot 84 + 21 \\ 84 &\div 21 = 4 \\ GCD(1113, 504) &= 21 \end{aligned}$$

Euklids algoritme har blitt utvidet, noe som har gjort det mulig å finne multiplikativ invers til tall.

1.2.2 Euklids utvidede algoritme

I tallteori ønsker vi ofte å granske likninger på formen

$$a \cdot x + b \cdot y = k, \text{hvor } a, b, k \in \mathbb{Z}$$

Slike likninger kalles diofantiske likninger, og opererer kun med heltallsløsninger. Euklids utvidede algoritme er designet for å finne de kritiske tallene a og b i diofantiske likninger.

Merk 1.10. Gitt heltallene x og y, kan vi finne heltall a og b slik at $a \cdot x + b \cdot y = GCD(x, y)$.

Merk 1.11. Hvis heltallene x og y er relativt primiske kan vi finne heltall a og b slik at $a \cdot x + b \cdot y = 1$.

Den utvidede algoritmen bygger på Euklids algoritme, hvor GCD(x,y) uttrykkes på formen $a \cdot x + b \cdot y$. Vi finner de kritiske tallene a og b ved å reversere Euklids algoritme med utgangspunkt i GCD(x,y).

Nedenfor er Euklids utvidede algoritme eksemplifisert med utgangspunkt i GCD(1113,504):

$$\begin{aligned} 1113 &= 2 \cdot 504 + 105 \\ 504 &= 4 \cdot 105 + 84 \\ 105 &= 1 \cdot 84 + 21 \\ \\ 21 &= 105 - (1 \cdot 84) \\ &= 105 - (1 \cdot (504 - 4 \cdot 105)) = 5 \cdot 105 - 1 \cdot 504 \\ &= 5 \cdot (1113 - 2 \cdot 504) - 1 \cdot 504 = 5 \cdot 1113 - 11 \cdot 504 \\ 21 &= 5 \cdot 1113 - 11 \cdot 504 \end{aligned}$$

Vi kan bruke Euklids utvidede algoritme til å finne multiplikativ invers til x modulus n. Husk at x er et inverterbart tall modulus n hvis og bare hvis x og n er relativt primiske. Om dette er tilfellet, kan vi finne a og b slik at $a \cdot x + b \cdot n = 1$, noe som betyr at $a \cdot x \equiv 1 \, (mod \; n)$. Dette er ekvivalent med at $a \equiv x^{-1} \, (mod \; n)$.

La oss se på et eksempel, for å gjøre det enklere å forstå.

Eksempel 1.12. Hva er $59 \div 237 \, (mod \; 1466)$?
For å løse problemet trenger vi først å finne 237^{-1} i modulus 1466, fordi $\frac{59}{237} = 59 \cdot 237^{-1} \, (mod \; 1466)$
GCD(1466,237):

$$\begin{aligned} 1466 &= 6 \cdot 237 + 44 \\ 237 &= 5 \cdot 44 + 17 \\ 44 &= 2 \cdot 17 + 10 \\ 17 &= 1 \cdot 10 + 7 \\ 10 &= 1 \cdot 7 + 3 \\ 7 &= 2 \cdot 3 + 1 \\ GCD(1466, 237) &= 1 \longrightarrow 1466 \text{ og } 237 \text{ er relativt primiske.} \end{aligned}$$

$$\begin{aligned} 1 &= 7 - 2 \cdot 3 \\ &= 7 - 2 \cdot (10 - 1 \cdot 7) = 3 \cdot 7 - 2 \cdot 10 \\ &= 3 \cdot (17 - 1 \cdot 10) - 2 \cdot 10 = 3 \cdot 17 - 5 \cdot 10 \\ &= 3 \cdot 17 - 5 \cdot (44 - 2 \cdot 17) = 13 \cdot 17 - 5 \cdot 44 \\ &= 13 \cdot (237 - 5 \cdot 44) - 5 \cdot 44 = 13 \cdot 237 - 70 \cdot 44 \\ &= 13 \cdot 237 - 70 \cdot (1466 - 237 \cdot 6) = 433 \cdot 237 - 70 \cdot 1466 \\ 1 &= 433 \cdot 237 - 70 \cdot 1466 \end{aligned}$$

Vi har at $237^{-1} \equiv 433 \, (mod \; 1466)$, og kan nå regne ut $59 \div 237 \, (mod \; 1466)$.

$$\frac{59}{237} = 59 \cdot 237^{-1} = 59 \cdot 433 = 25547 \equiv 625 \, (mod \; 1466)$$

Sjekker svaret:

$$625 \cdot 237 = 148125 = 101 \cdot 1466 + 59 \equiv 59 \, (mod \; 1466)$$

1.3 Eulers teorem

For å presentere Eulers teorem, trenger vi å introdusere flere begreper og funksjoner i forkant. Primtall er et sentralt begrep i tallteori. Primtall har kun 1 og seg selv som

divisorer. Er to tall relativt primiske betyr det at de har ingen felles divisorer utenom 1. Eulers phi funksjon gir antallet heltall mindre enn n som er relativt primiske med n.

Definisjon 1.13. Eulers phi funksjon, $\varphi(n)$, for et positivt heltall n større enn 1 er definert til å være antall positive heltall mindre enn n som er relativt primiske til n. Hvor $\varphi : \mathbb{N} \longrightarrow \mathbb{N}$ er definert med $n \longrightarrow \varphi(n)$. Hvor $\varphi(1)$ er definert til å være 1.

Vi kan finne $\varphi(8)$ ved å sjekke hvilke tall som er relativt primiske til 8:

$$\begin{aligned} GCD(1,8) &= 1 \\ GCD(2,8) &= 2 \\ GCD(3,8) &= 1 \\ GCD(4,8) &= 4 \\ GCD(5,8) &= 1 \\ GCD(6,8) &= 2 \\ GCD(7,8) &= 1 \end{aligned}$$

$\varphi(8) = 4$ ettersom det er 4 heltall, mindre enn 8, som ikke har felles nevner med 8.

Fra Definisjon 1.13 følger det at $\varphi(p) = p-1$, for hvert primtall p. Eulers phi funksjon kan anvendes for å teste om et tall er et primtall, samt for å finne antall inverterbare tall i en modulus.

Fakta 1.14. Hvis m og n er relativt primiske, da har vi $\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$

Vi finner $\varphi(56)$ ved bruk av Fakta 1.14.
Vi velger å skrive 56 som et produkt av to relativt primiske faktorer, $56 = 7 \cdot 8$.

$$\varphi(56) = \varphi(7) \cdot \varphi(8)$$

Siden $\varphi(7) = 6$ og $\varphi(8) = 4$ så får vi:

$$\varphi(56) = 6 \cdot 4 = 24$$

Fakta 1.15. Hvis $m = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot p_3^{\alpha_3} \cdots p_r^{\alpha_r}$ er primtallsfaktorsieringen av det naturlige tallet m, da er

$$\varphi(m) = m \prod_{i=1}^{r} (1 - \frac{1}{p_i}) \tag{1}$$

eller

$$\varphi(m) = \prod_{i=1}^{r} (p_i - 1) \cdot p_i^{\alpha_i - 1} \tag{2}$$

Vi kan eksempelvis finne antall inverterbare tall i modulus 720 ved bruk av Fakta 1.15.
Primtallsfaktoriseringen gir $720 = 2^4 \cdot 3^2 \cdot 5$.
Ligning (1) gir oss:

$$\begin{aligned}\varphi(720) &= \varphi(2)^4 \cdot \varphi(3)^2 \cdot \varphi(5) \\ &= 720 \cdot (1 - \frac{1}{2}) \cdot (1 - \frac{1}{3}) \cdot (1 - \frac{1}{4}) \\ &= 720 \cdot \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{4}{5} \\ &= 192\end{aligned}$$

Ligning (2) gir oss:

$$\begin{aligned}\varphi(720) &= \varphi(2)^4 \cdot \varphi(3)^2 \cdot \varphi(5) \\ &= (2-1) \cdot 2^{4-1} \cdot (3-1) \cdot 3^{2-1} \cdot (5-1) \cdot 5^{1-1} \\ &= 1 \cdot 2^3 \cdot 2 \cdot 3^1 \cdot 4 \cdot 5^0 \\ &= 8 \cdot 2 \cdot 3 \cdot 4 \\ &= 192\end{aligned}$$

Fermats lille teorem sier at et tall opphøyd i et primtall er kongruent med tallet modulus primtallet.
Fermats lille teorem: Her er a et heltall og p et primtall. Hvor $a \not\equiv 0 \, (mod \; p)$.

$$a^{p-1} \equiv 1 \, (mod \; p)$$

Som følge av Fermats lille teorem kan vi multiplisere uttrykket med a på begge sider, og få $a^p \equiv a \, (mod \; p)$, hvor det er tillatt at $a \equiv 0 \, (mod \; p)$.

Hva er $169 \, (mod \; 3)$?
Vi kan bruke Fermats teorem til å svare på spørsmålet:

$$169 = 13^2, \text{ så } 169 = 13^{3-1} \equiv 1 \, (mod \; 3)$$

Euler har generalisert Fermats lille teorem.
Eulers teorem: Hvis n og a er relativt primiske positive heltall, og $\varphi(n)$ er Eulers phi funksjon, da er a opphøyd i $\varphi(n)$ kongruent til 1 modulus n.

$$a^{\varphi(n)} \equiv 1 \, (mod \; n)$$

Som følge av Eulers teorem vil a opphøyd i ethvert multiplum av $\varphi(n)$ være kongruent 1. Vi skriver $a^{k\cdot\varphi(n)} \equiv 1\,(mod\ n)$, hvor $k \in \mathbb{Z}$. Vi kan multiplisere uttrykket med a på begge sider, og få uttrykket $a^{k\cdot\varphi(n)+1} \equiv a\,(mod\ n)$, hvor $k \in \mathbb{Z}$.

Dette teoremet er enda mer anvendelig enn Fermats lille teorem. La oss bruke Eulers teorem til å finne det siste sifferet i 13^{2010}.
13 og 10 er relativt primiske, og $\varphi(10) = 4$. Eulers teorem sier at $13^4 \equiv 1\,(mod\ 10)$ og vi får da:

$$13^{2010} = 13^{4\cdot 502+2} = (13^4)^{502} \cdot 13^2 \equiv 1^{502} \cdot 13^2 \equiv 169 \equiv 9\,(mod\ 10)$$

Det siste tallet i 13^{2010} er 9.

Eulers teorem er en essensiell komponent i RSA kryptering- og dekrypteringsprossessen. Applikasjonen av kryptografi vil bli forklart nærmere i de neste kapitlene.

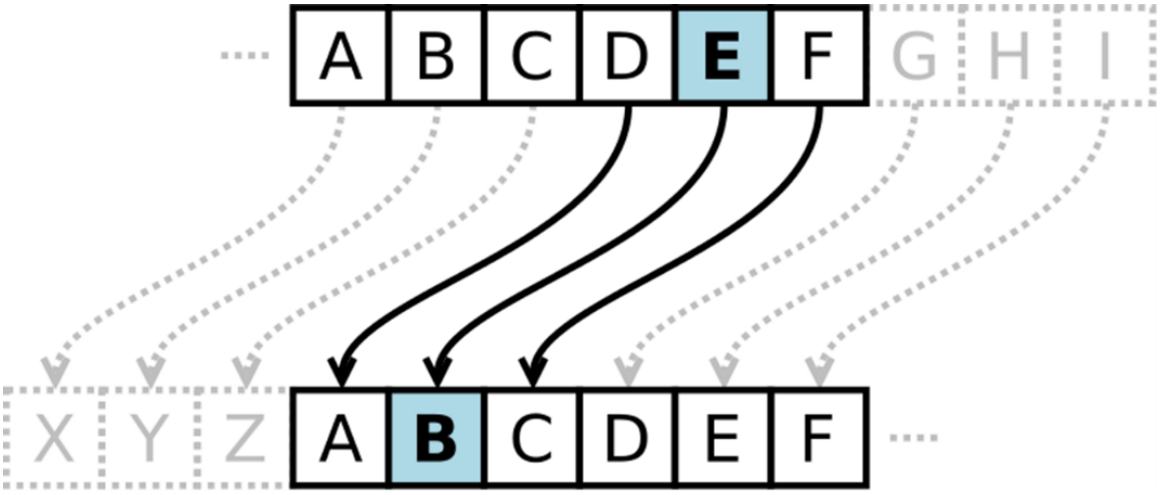
2 Kryptografi

Ordet kryptografi kommer fra de to greske ordene kryptos og graphein. Krypto- er en forstavelse som betyr skjult og graphein betyr skrive, altså 'skjult skrift'. Hvordan forstår vi begrepet kryptografi i dag? Store Norske Leksikon [10] sier at 'kryptografi er vitenskapen om prinsipper og teknikker for å skjule informasjon slik at bare en som er autorisert har mulighet til å avsløre innholdet.'. Datatilsynet [7] bruker følgende definisjon for kryptografi: 'Metode for å gjøre data (for eksempel tekst) uleselig for andre ved hjelp av en matematisk funksjon (krypteringsteknikk/-algoritme) og en forhåndsbestemt nøkkel.'. Kryptering handler om å endre en beskjed på en slik måte at en må vite en hemmelighet for å kunne dekryptere og lese beskjeden. Denne hemmeligheten kalles for en nøkkel.

Hva er hensikten med kryptografi?
Behovet for kryptografi oppstod da det var et behov for å sende hemmelige meldinger til andre mottakere, og var tidligere stort sett et militært anliggende. Med utviklingen av internett og økt bruk av datamaskiner er kryptografi en viktig del av all digital kommunikasjon både for sivile bedrifter og privatpersoner, for eksempel ved netthandel og banktjenester. Kryptering brukes for at parter skal kunne autentisere seg og beskytte transaksjoner mot uautorisert endring og innsyn.

For å forstå kryptografi så trenger vi å vite at den deles inn i to hovedgrupper: utbyttingskryptering og transposisjonskryptering.

Utbyttingskryptering går ut på at en bytter ut hver bokstav med en annen bokstav eller symbol for å skape chiffertekst. Problemet med ren utbyttingskryptering er at de statistiske skjevhetene som finnes i naturlige språk ikke blir skjult. Krypteringen er derfor enkel for utenforstående å knekke. 'Cæsar Chiffer' er et klassisk eksempel på utbyttingskryptering. Cæsar byttet rett og slett ut en bokstav fra alfabetet med en annen, ved å forskyve et visst antall plasser mot venstre. Hans vanligste metode var å forskyve 3 hakk til venstre.



Figur 1: Utbytting med Cæsar-kode

Transposisjonskryptering går ut på at man endrer rekkefølgen på bokstavene i beskjeden for å skape chiffertekst, ut i fra en avtalt metode, uten å gjøre utbyttinger. En klassisk transposisjonskryptering som ble brukt av spartanerne er 'skytale', der tykkelsen på pinnen er nøkkelen. Man tar en pinne av en valgt tykkelse, deretter surrer man en remse, med plass til en bokstav i bredden, i spiral rundt pinnen. Deretter skriver man over pinnen på langs. Når en tar bort pinnen er det ikke lengre mulig å lese beskjeden, fordi man bare har en papirremse med masse bokstaver. Når mottaker skal lese denne krypterte teksten trenger man en like tykk pinne som nøkkelen, ellers vil beskjeden være uleselig.

Figur 2: Transposisjonering med 'Skytale'

I all kryptering trengs det nøkler for å kryptere/dekryptere. Det er to hovedretninger for anvendelse av nøkler innen kryptografi: Symmetrisk og offentlig-nøkkel kryptografi.

2.1 Symmetrisk kryptering

I symmetrisk kryptering benyttes samme nøkkel til å låse ned og låse opp informasjon. Denne nøkkelen må utveksles mellom avsender og mottaker på en sikker måte. Både 'Cæsar Chiffer' og 'Skytale' er enkle eksempler på symmetrisk kryptering.

En utfordring som kommer med denne krypteringsmåten er at nøkkelen må være avtalt på forhånd, for deretter å bli holdt hemmelig. Hvis Kari og Ola bruker en krypteringsordning med symmetrisk nøkkel, må de først utveksle den hemmelige nøkkelen. Til utvekslingen må de bruke en sikker kommunikasjonskanal. Krypteringsmetoder med offentlig-nøkkel, som vi studerer i kapittel 2.2, brukes ofte til dette formålet. Krypteringsordninger med offentlig-nøkkel er mindre effektive og passer derfor ikke for store mengder data. Dermed utfyller symmetrisk nøkkelkryptering og kryptering av offentlig-nøkkel hverandre for å gi praktiske kryptosystemer.

Krypteringsalgoritmer som har symmetrisk-nøkkel har de raskeste implementeringene innen maskinvare og programvare. Derfor er de veldig godt egnet til kryptering av store mengder data.

2.1.1 Flytchiffer

Flytchiffer går inn under symmetrisk kryptering, der avsender og mottaker av dataene bruker den samme nøkkelen til å kryptere og dekryptere dataene. I flytchiffer krypteres en strøm $m := m_1m_2m_3...$ av ren tekst elementer til en strøm av chiffertekst $c := c_1c_2c_3...$ ved bruk av en nøkkelstrøm $k := k_1k_2k_3....$ En nøkkelstrøm er en tilfeldig 8-bits utgang som genereres ved å levere en hemmelig nøkkel til en pseudovilkårlig nummergenerator. Ren tekst strømmen m bli kryptert element for element. Nøkkelstrømmen brukes i kryptering og dekryptering av data i en gitt flytchifferalgoritme.

Eksklusiv disjunksjon, skrives også XOR, er operasjonen av addisjon av n-tupler over $\mathbb{Z}/2\mathbb{Z}$. De fleste flytchiffer benytter XOR operasjonen.

XOR	0	1
0	0	1
1	1	0

XOR av to bits a og b betyr å addere dem i modulus 2: a XOR $b = a + b\,(mod\ 2)$.

Kryptering krever ren tekst og nøkkelstrøm. Den rene teksten og nøkkelstrømmen produserer chifferkodetekst ved hjelp av XOR-operasjon. Ren tekst er XORet med nøkkelstrømmen bit for bit for å produsere chiffertekst. Eksempel:

Ren tekst: 10011001
Nøkkelstrøm: 11001100
Chiffertekst: 01010101

Dekryptering krever chiffertekst og samme nøkkelstrøm som ble brukt til kryptering. Chifferteksten og nøkkelstrømmen produserer ren tekst ved hjelp av XOR-operasjon. Chifferteksten er XORet med nøkkelstrømmen bit for bit for å produsere ren tekst. Eksempel:

Chiffertekst: 01010101
Nøkkelstrøm: 11001100
Ren tekst: 10011001

Flytchiffer er rask, lineær i tid, og konstant i rommet. Det er usannsynlig å overføre feil, ettersom feil oversettelse i en byte ikke vil påvirke neste byte. Imidlertid er det liten diffusjon da ett ren tekst-symbol er direkte oversatt til ett chiffertekst-symbol. Chifferteksten er også utsatt for innsettinger eller modifikasjoner. Hvis en angriper kan bryte algoritmen, kan det settes inn tekst som ser autentisk ut.

Vi bruker vanligvis flytchiffer når mengden ren tekst er ukjent, for eksempel ved lyd- eller videostrømming, eller når ekstrem ytelse er viktig, som med tilkoblinger med svært høy hastighet, samt for enheter som må være svært effektive og kompakte, for eksempel smartkort.

2.1.2 Blokkchiffer

Blokkchiffer konverterer ren tekst til chiffertekst, blokk etter blokk. Når det krypteres kan et blokkchiffer eksempelvis ta en 128-bit blokk av klartekst som inndata, og produsere en tilsvarende 128-bit blokk med chiffertekst. Nøyaktig hvilken omforming som benyttes er kontrollert av den hemmelige krypteringsnøkkelen. Dekryptering er tilsvarende: dekrypteringsalgoritmen tar chifferteksten og krypteringsnøkkelen som inndata, og utdata er den opprinnelige 128-bit klarteksten. For å øke sikkerheten til en kryptering kan man gjenta krypteringsprosessen flere ganger.

Wikipedia [20] sier at blokkchiffer består av to algoritmer, en for kryptering, E, og en for dekryptering, E^{-1}. Begge algoritmene trenger to inndata: inngående blokktekst på n-bit, og en krypteringsnøkkel på k bits. Begge algoritmene gir utdata som n-bit blokk. For enhver fast nøkkel, så er dekryptering den inverse funksjonen av E_K, slik at $E_K^{-1}(E_K(M)) = M$ for enhver blokk M og krypteringsnøkkel K. E_K er en permutasjon over settet av inndata blokker. Hver nøkkel velger en permutasjon av $2^n!$ mulige.

Blokkchiffer-algoritmer bruker en serie med utbyttinger og transposisjoner i lenkede matematiske operasjoner. Blokkchiffer har høy spredning, som betyr at informasjon fra en ren tekst er delt inn i flere chiffertekstsymboler. Det er vanskelig for en angriper å sette inn symboler uten gjenkjenning midt i en blokk. Blokkchiffer er tregere enn en flytchiffer, ettersom en hel blokk må overføres før kryptering/dekryptering kan skje. Hvis det oppstår en feil, kan den forplante seg gjennom hele blokken og ødelegge hele delen. Blokkchiffer er et bra valg når vi vet overføringsstørrelsen - for eksempel i filoverføring.

Det finnes ingen symmetriske krypteringmetoder som har en sikker løsning på utveksling av krypteringsnøkkel. En populær metode for å utveksle symmetriske krypteringsnøkler er å benytte offentlig-nøkkel kryptering.

2.2 Offentlig-nøkkel

Prinsippet bak offentlig-nøkkel, også kjent som asymmetrisk kryptografi, kan forstås gjennom følgende eksempel: Ola Nordmann har en ulåst hengelås som han sender til Kari Nordmann, men han beholder nøkkelen selv. Kari kan skrive en hemmelig beskjed, låse den med hengelåsen og sende den tilbake til Ola. Det er bare Ola som kan låse opp beskjeden, siden Ola er den eneste med riktig nøkkel.

Offentlig-nøkkelkryptografi er basert på en veldig enkel ide. Ideen er at alle bør ha to krypteringsnøkler, en offentlig og en privat. Der den offentlige nøkkelen er tilgjengelig for alle som ønsker å kommunisere med Ola, mens den private nøkkelen er hemmelig og det er kun Ola som kan bruke den. Krypteringsmetoden må fungere slik at beskjeder som er kryptert med Ola sin offentlige nøkkel, kun kan dekrypteres med Ola sin private nøkkel. På samme måte hvis en beskjed er kryptert med Ola sin private nøkkel, så er det kun Ola sin offentlige nøkkel som kan dekryptere beskjeden.

Systemet med offentlig-nøkkel kan enkelt bli brukt til å bekrefte identitet, ettersom ingen, utenom Ola, kan bruke Ola sin private nøkkel. La oss si at Per møter opp i banken og påstår at han er Ola. Hvis han faktisk er Ola, burde han være i stand til å kryptere en signaturmelding, noe som ligner på 'hei, mitt navn er Ola', ved bruk av Ola sin private nøkkel. Alle har, inkludert banken, tilgang til Ola sin offentlige nøkkel, og burde kunne dekryptere denne signaturmeldingen og gjennom det bekrefte, eller avkrefte, at det er Ola sin private nøkkel som ble brukt. Poenget er at Ola, og kun Ola, kan ha lagd en kryptert melding som er de-krypterbar ved bruk av Ola sin offentlige nøkkel. Systemet med offentlige og private nøkler kan derfor bli brukt til å fungere som et sikkert autentiseringssystem, og er blant annet veldig anvendelig for bank og kreditkort transaksjoner. Digital signering av elektroniske dokumenter er blitt mulig grunnet offentlig-nøkkelkryptering.

Det finnes mange matematiske teknikker for å lage offentlig-nøkkel systemer. Et av de mest brukte og mest kjente offentlig-nøkkel kryptosystemene er RSA, navner kommer fra initialene til skaperne av systemet, Ron Rivest, Adi Shamir og Leonard Adleman, og er fundamentet av moderne offentlig-nøkkel kryptografi.

3 RSA - Kryptografi

Hele dette kapitlet tar utgangspunk i Milson [13] sin artikkel *Introduction to Public Cryptography and Modular Aritmetic*.

Hvordan kan vi anvende offentlig-nøkkel ideen slik at den kan brukes? Først kan det være nyttig å huske potensregelen $(a^b)^c = a^{bc}$. I modulær aritmetikk er det enkelte eksponenter som ikke endrer grunntallet. La oss se på eksponenten 21 i modulus 22. Merk at $21 = 3 \cdot 7$, og vi kan derfor skrive det slik

$$6^{21} = (6^3)^7 = 216^7 \equiv 18^7 = 612220032 \equiv 6\,(mod\ 22)$$

Når vi jobber i modulus 22 kan vi først opphøye et tall med 3, for så å opphøye resultatet med 7, deretter vil sluttproduktet være kongruent med tallet vi startet med. I modulus 22 kan 3 være privat nøkkel og 7 offentlig nøkkel. La oss sende beskjeden '2,6,9' kryptert, ved å bruke den private nøkkelen.
Kryptering:
Hvert tall i beskjeden opphøyes med 3, siden 3 er den private nøkkelen:

$$\begin{aligned} 2^3 &\equiv 8\,(mod\ 22)\,, \\ 6^3 = 216 &\equiv 18\,(mod\ 22)\,, \\ 9^3 = 729 &\equiv 3\,(mod\ 22) \end{aligned}$$

Vår krypterte beskjed er derfor '8,18,3'.
Dekryptering:
Hvert tall opphøyes med 7, siden 7 er den offentlige nøkkelen:

$$\begin{aligned} 8^7 = 2097152 &\equiv 2\,(mod\ 22)\,, \\ 18^7 = 612220032 &\equiv 6\,(mod\ 22)\,, \\ 3^7 = 2187 &\equiv 9\,(mod\ 22) \end{aligned}$$

Vi er nå tilbake til den opprinnelige beskjeden '2,6,9'. Dette eksempelet viser prinsippet og essensen til RSA - kryptering.

Hvorfor er det slik at enkelte eksponenter ikke endrer grunntallet i modulær aritmetikk? Det kommer av Eulers teorem, $a^{k\cdot\varphi(n)+1} \equiv a\,(mod\ n)$, hvor $k \in \mathbb{Z}$. La oss sjekke dette opp mot eksempelet ovenfor. Først trenger vi å finne $\varphi(22)$, vi bruker Fakta 1.
Primtallsfaktoriseringen gir $22 = 2^1 \cdot 11^1$
Formelen fra Fakta 1 gir oss:

$$\begin{aligned} \varphi(22) &= (2-1)\cdot 2^{1-1}\cdot(11-1)\cdot 11^{1-1} \\ &= 1\cdot 2^0\cdot 10\cdot 11^0 \\ &= 10 \end{aligned}$$

Hvis vi velger $k = 2$, så får vi at $a^{2\cdot\varphi(22)+1} = a^{21} \equiv a\,(mod\ 22)$, som samsvarer med eksempelet ovenfor.

Spørsmål 3.1. *Hvordan vet vi at RSA offentlig-nøkkel krypteringsmetoden er sikker?*

Realiteten er at metoden ikke er en sikker kryptering hvis man ikke velger riktig modulus n. Se igjen på eksempelet med $n = 22$ og offentlig nøkkel 3, $e = 3$, med ukjent privat nøkkel, f. Når en vet $n = 22$, så kan alle finne ut at $\varphi(22) = 10$, og uansett hva f er så må $e \cdot f = k \cdot \varphi(n) + 1$, hvor $k \in \mathbb{Z}$. Altså må $e \cdot f$ i dette tilfellet være lik et av følgende tall: $1, 11, 21, 31, 41, ...$ Med andre ord så må $e \cdot f \equiv k \cdot \varphi(n) + 1 \, (mod \ \varphi(n))$. Sett inn $e = 3$, $k = 0$ og $n = 22$ i uttrykket, og vi får $3 \cdot f \equiv 1 \, (mod \ 10)$. En kjapp sjekk av tallene fra 1 til 9 vil avsløre at $f = 7$ er en løsning til likningen:

$$
\begin{aligned}
3 \cdot 1 &= 3 \equiv 3 \, (mod \ 10) \\
3 \cdot 2 &= 6 \equiv 6 \, (mod \ 10) \\
3 \cdot 3 &= 9 \equiv 9 \, (mod \ 10) \\
3 \cdot 4 &= 12 \equiv 2 \, (mod \ 10) \\
3 \cdot 5 &= 15 \equiv 5 \, (mod \ 10) \\
3 \cdot 6 &= 18 \equiv 8 \, (mod \ 10) \\
3 \cdot 7 &= 21 \equiv 1 \, (mod \ 10) \\
3 \cdot 8 &= 24 \equiv 4 \, (mod \ 10) \\
3 \cdot 9 &= 27 \equiv 7 \, (mod \ 10)
\end{aligned}
$$

Nå er den antatte hemmelige, private nøkkelen avslørt og hele metoden for kryptering er ubrukelig.

Poenget med eksempelet ovenfor er å påpeke viktigheten av å velge riktig modulus n. Det er en spesiell prosedyre for å velge n, som gjør påfølgende prosess av kryptering/dekryptering sikker. Velg to primtall, p og q, og sett modulus n lik $p \cdot q$. Store primtall er ofte på opptil hundre siffer. Produktet av to primtall på hundre siffer hver blir da på to hundre siffer. Å faktorisere en n som er et produkt av to store primtall er foreløpig ikke mulig for verdens raskeste computere. Hvorfor ønsker vi å lage en modulus n som ikke er faktoriserbar? Jo, det kommer av at vi ønsker å velge en n slik at den korresponderende $\varphi(n)$ er enormt vanskelig å finne.

En annen fordel med å velge $n = p \cdot q$ er at Eulers phi funksjon 1.13 gir oss, $\varphi(p) = p - 1$ hvor p er et primtall. Vi har nå en enkel måte å kalkulere den korresponderende $\varphi(n)$. Nemlig $\varphi = (p - 1)(q - 1)$.

Det positive er at med store p og q, så kan vi publisere n, og p og q vil forbli trygge fra å bli funnet gjennom faktorisering. Ettersom p og q er hemmelig, så er det ingen utenforstående som skal kalkulere $\varphi(n)$, og derfor ikke klare å gjette den private nøkkelen f. Kryptosystemet kan bli sett på som sikkert.

3.1 Å generere nøkler

- Velg to primtall p og q, hvor $p \neq q$, og sett $n = p \cdot q$ og $\varphi = (p-1)(q-1)$.
- Velg et heltall e slik at $e < n$, hvor e og φ er relativt primiske. (n, e) blir den offentlige nøkkelen.
- Sett f lik den multiplikativt inverse til $e \; mod \; \varphi(n)$, altså $f = e^{-1} \, (mod \; \varphi(n))$. Da er (n, f) den private nøkkelen.

Jo større verdier på p og q, desto mer sikker er krypteringen. La oss bruke $p = 13$ og $q = 17$ som eksempel, selv om de ikke er store. Vi får da $n = 221$ og $\varphi = 192$. Nå trenger vi ikke p og q lenger, så disse må elimineres slik at ingen vet hva vi har brukt. Et triks er å velge e som et primtall mindre enn φ, slik er vi garantert at e og φ er innbyrdes primiske og e er et inverterbart element i modulus φ. Vi velger $e = 43$, ettersom 43 er et primtall mindre enn φ.
Vi regner ut multiplikativ invers til e modulus φ ved bruk av Euklids utvidede algoritme:

$$
\begin{aligned}
192 &= 4 \cdot 43 + 20 \\
43 &= 2 \cdot 20 + 3 \\
20 &= 6 \cdot 3 + 2 \\
3 &= 1 \cdot 2 + 1 \\
GCD(192, 43) &= 1 \longrightarrow 192 \text{ og } 43 \text{ er relativt primiske.} \\
1 &= 3 - 1 \cdot 2 \\
&= 3 - 1 \cdot (20 - 6 \cdot 3) = 7 \cdot 3 - 1 \cdot 20 \\
&= 7 \cdot (43 - 2 \cdot 20) - 1 \cdot 20 = 7 \cdot 43 - 15 \cdot 20 \\
&= 7 \cdot 43 - 15 \cdot (192 - 4 \cdot 43) = -15 \cdot 192 + 67 \cdot 44 \\
1 &= -15 \cdot 192 + 67 \cdot 43 \\
&\text{Vi får at f=67}
\end{aligned}
$$

Vi offentliggjør vår offentlige nøkkel, som består av modulus n og eksponent e, (221,43). Eksponenten f forblir hemmelig og er den private nøkkelen, (221,67).

Hvis noen ønsker å sende en beskjed til oss, antatt at beskjeden kommer i en liste av tall modulus n, så opphøyes hvert tall i beskjeden med e, og den reduserte formen i modulus 221 er den kodete beskjeden.
La Ola Normann sende beskjeden 'Hallo' på tallformen 'H=8, A=1, L=12, L=12,

O=19' ved bruk av den offentlige nøkkelen (221,43) til oss:

$$8^{43} \equiv 70\,(mod\ 221)$$
$$1^{43} \equiv 1\,(mod\ 221)$$
$$12^{43} \equiv 142\,(mod\ 221)$$
$$12^{43} \equiv 142\,(mod\ 221)$$
$$19^{43} \equiv 59\,(mod\ 221)$$

Kryptert beskjed blir da '70,1,142,142,59'.
For å dekryptere beskjeden fra Ola opphøyer vi hvert tall i beskjeden med $f = 67$, og den reduserte formen i modulus 221 gir tilbake den opprinnelige meldingen.

$$70^{67} \equiv 8\,(mod\ 221)$$
$$1^{67} \equiv 1\,(mod\ 221)$$
$$142^{67} \equiv 12\,(mod\ 221)$$
$$142^{67} \equiv 12\,(mod\ 221)$$
$$59^{67} \equiv 19\,(mod\ 221)$$

Original beskjed blir da '8,1,12,12,19' som er 'Hallo'.

For å bekrefte ens identitet, vil vi lage en signaturbeskjed og opphøye tallene i beskjeden med f og finne den reduserte formen.
La oss sende signaturbeskjeden 'Hei' på tallformen 'H=8, E=5, I=9' ved bruk av den private nøkkelen (221,67):

$$8^{67} \equiv 70\,(mod\ 221)$$
$$5^{67} \equiv 125\,(mod\ 221)$$
$$9^{67} \equiv 100\,(mod\ 221)$$

Kryptert beskjed blir da '70,125,100'
Mottaker vil da utføre verifisering ved å opphøye tallene i den krypterte beskjeden med eksponenten $e = 43$, og finne den reduserte formen. På den måten dekrypterer mottaker tilbake til den opprinnelige signaturbeskjeden.

$$70^{43} \equiv 8\,(mod\ 221)$$
$$125^{43} \equiv 5\,(mod\ 221)$$
$$100^{43} \equiv 9\,(mod\ 221)$$

Original beskjed blir da '8,5,9' som er 'Hei'.
De som ønsker å dekryptere beskjeden vet nå med sikkerhet at det er vi som har sendt meldingen.

4 Angrep på RSA - Kryptosystem

Dette kapitlet tar utgangspunkt i Boneh [3] sin artikkel *Twenty Years of Attacks on the RSA Cryptosystem.*
Å forsere ('knekke') et kryptosystem vil si å dekryptere kryptoteksten uten å kjenne nøkkelen; vitenskapen om forsering av kryptosystem kalles kryptoanalyse. Et perfekt kryptosystem er umulig å forsere uansett hvor store ressurser en angriper rår over. Helt siden RSA-systemet ble publisert i 1977 har det blitt analysert for sårbarheter. Selv om det har vært flere fascinerende angrep, har ingen av dem gjort stor skade. Som oftest illustrerer angrepene faren ved feilbruk av RSA. Gjennom kapitlet vil vi bruke navnene Ola og Kari til å betegne generiske parter som ønsker å kommunisere med hverandre. Vi bruker Markus til å betegne en ondsinnet angriper som ønsker å tyvlytte eller tukle med kommunikasjonen mellom Ola og Kari.

I dette kapitlet vil vi hovedsaklig se på RSA-funksjonen framfor RSA-kryptosystemet. RSA-funksjonen baserer seg på vanskeligheten med å invertere funksjonen på tilfeldige inndata hvor en angriper er gitt (n, e, C). Derimot må et kryptosystem motstå mer subtile angrep. Hvis (n, e, C) er gitt, skal kryptosystemet være ugjennomtrengelig for å gjenopprette enhver informasjon om beskjedene som sendes.

Det første angrepet på en RSA offentlig-nøkkel (n, e) vi bør vurdere er å faktorisere modulus n. Gitt faktoriseringen $n = pq$, kan en angriper enkelt konstruere $\varphi(n)$, hvor dekrypteringseksponenten $f = e^{-1} \, (mod \; \varphi(n))$ kan bli funnet. Vi vil referere til faktorisering av modulusen som et **brute force-angrep** på RSA, og angrep på RSA som tar lengre tid enn dette er ikke interessante. Selv om faktoriseringsalgoritmer utbedres kontinuerlig, så er den nåværende situasjonen fortsatt langt ifra å utgjøre en trussel mot sikkerheten av RSA, gitt at RSA brukes riktig. Vi sikrer oss mot brute force-angrep ved å velge en tilstrekkelig stor nøkkel.

Nøkkelstørrelse refererer til antall bits i en nøkkel brukt av en kryptografisk algoritme. Nøkkellengen definerer upper-bound på en algoritmes sikkerhet, som er en logaritmisk måling av det raskeste kjente angrepet på algoritmen. En av de foreløpig raskeste faktoriseringsalgoritmene er 'General Number Field Sieve'. Algoritmens kjøretid på n-bit heltall har eksponential vekst

$$\exp((c + o(1))n^{\frac{1}{3}} \log^{\frac{2}{3}} n), \text{ for en } c < 2.$$

Siden RSA ble offentliggjort i 1977 og fram til idag, så har kravene for sikkerhet knyttet til nøkkelstørrelsen endret seg, fordi datamaskinene har utviklet seg og fått høyere ytelse. 512 bits (avrundet ned fra 664 bits eller 200 sifre i patentet) ble anbefalt som nøkkelstørrelse fra unnfangelsen i 1974 og gjennom hele 1980-tallet. Faktisk ble

463 bits ansett som tilstrekkelig på midten av 1990-tallet for RSA-140-utfordringen. Hvorvidt nøkkelstyrker så lave som 100 sifre (330 bits) noen gang ble brukt tidlig på 1980-tallet er uklart. RSAs anbefalte nøkkelstørrelse økte til 768 (bruker) eller 1024 (bedrift) på slutten av 1990-tallet på grunn av akademiske suksesser med å bryte bits-styrker som nærmet seg 512 bits.

I praksis samsvarer ikke disse nøkkelstyrkene jevnt med Moores lov, da hver nye nøkkelstørrelse krever overhaling av innebygde systemer og løsing av interoperabilitetskrav. Så trenden har vært å velge en nøkkelstørrelse mye større enn nødvendig, og beholde den til risikoen for brudd oppstår, og deretter velge en annen nøkkelstørrelse mye større enn nødvendig. En grov trinnvis økning styrt av infrastrukturkostnader og kollektiv offentlig oppfatning av nøkkelstyrke. Gjeldende anbefalinger (SP 800-572) [2] er nå 2048 eller 3072 bits, avhengig av interoperabilitetskrav. Det største kjente faktoriserte RSA tallet hadde 829 bits, sifferet har 250 desimaler, og ble gjennomført i 2020.

Vi ønsker å se på angrep som dekrypterer meldinger uten å direkte faktorisere modulus $n = pq$, men det er verdt å nevne at hvis $p - 1$ er et produkt av primtall mindre enn B, kan n bli faktorisert på mindre tid enn $O(B^3)$.

Å eksponere den private nøkkelen f og faktorisere n er ekvivalent. Altså er det ingen poeng i å gjemme faktoriseringen av n for parter som kjenner f.

Fakta 4.1. La (n, e) være en RSA offentlig-nøkkel. Gitt den private nøkkelen f, kan vi effektivt faktorisere modulus $n = p \cdot q$. Omvendt, gitt faktoriseringen av n, kan vi effektivt finne f.

4.1 Elementære angrep

Vi begynner med å beskrive noen gamle elementære angrep. Disse angrepene illustrerer åpenbar feilbruk av RSA.

4.1.1 Felles modulus

For å slippe å generere en ulik modulus $n = p \cdot q$ for hver bruker, vil det være enklere å sette en fast n. Den samme n vil da bli brukt av alle brukere. Da vil en tillitsverdig sentral autoritet kunne distribuere bruker i med et unikt par e_i, f_i som for hver bruker i danner en offentlig nøkkel (n, e_i) og en privat nøkkel (n, f_i).

Med første øyekast kan felles modulus se ut til å fungere: en chiffertekst $C = M^{e_k} \ (mod\ n)$ tiltenkt Kari, kan ikke bli dekryptert av Ola siden Ola ikke har tilgang til f_k. Ved felles modulus kan Ola dekryptere C tiltenkt Kari, og det resulterende systemet er

usikkert. Fakta 4.1 sier at Ola kan bruke sine egne eksponenter e_o og f_o til å faktorisere modulus n. Når n er faktorisert kan Ola få tak i Kari sin private nøkkel f_k, ut i fra hennes offentlige nøkkel e_k. Denne observasjonen viser at i et RSA-system bør en modulus aldri bli brukt av mer enn en enkelt bruker.

4.1.2 Blending

I kryptografi er blending en teknikk hvor en agent kan tilby en tjeneste for en klient i kryptert form uten at agenten får vite om hverken det ekte inndata eller det ekte utdata. Dette er en teknikk som brukes for å implementere anonym digital handel. Digitale penger kan bli brukt til å kjøpe gjenstander, uten å avsløre identiteten til kjøperen.

Mer konkret, Kari har en inndata x og Ola har en funksjon f. Kari vil at Ola skal beregne $y = f(x)$ for henne uten å avsløre verken x eller y for ham. Årsaken til at hun ønsker dette kan være at hun ikke kjenner funksjonen f eller at hun ikke har ressurser til å beregne den. Kari 'blender' meldingen ved å kryptere den inn i en annen inndata $E(x)$; krypteringen E må være en bijeksjon på inndatasområdet til f, ideelt sett en tilfeldig permutasjon. Ola gir henne $f(E(x))$, som hun bruker en dekryptering D for å få $D(f(E(x))) = y$.

RSA-blending innebærer for eksempel beregning av blendingsoperasjonen $E(x) = xr^e \,(mod\ n)$, der r er et tilfeldig heltall mellom 1 og n, x er chifferteksten, og e og n har den konvensjonelle betydningen fra RSA. Som vanlig brukes dekrypteringsfunksjonen $f(x) = E(x)^f \,(mod\ n)$, og til slutt er den ublendet med $D(x) = x/r \,(mod\ n)$.

Blending kan også brukes som et angrep. La (n, f) være Ola sin private nøkkel og (n, e) den korresponderende offentlige nøkkelen. Anta at motstander Markus ønsker Ola sin signatur på en melding $M \in (\mathbb{Z}/n\mathbb{Z})^*$. Ola ønsker derimot ikke å signere M. Da kan Markus prøve følgende blendingsteknikk for å få tak i en validert signatur på meldingen M:
Markus velger en tilfeldig $r \in (\mathbb{Z}/n\mathbb{Z})^*$ og setter $M' = r^e M \,(mod\ n)$. Deretter spør han Ola om å signere denne tilfeldige meldingen M'. Ola kan være villig til å signere S' på denne uskyldige meldingen M'. Husk at $S' = (M')^f \,(mod\ n)$. Markus trenger nå bare å kalkulere $S = S'/r \,(mod\ n)$ og får dermed tar i Ola sin signatur S. Nå har Markus fått Ola sin signatur S på den originale meldingen M:

$$S^e = (S')^e/r^e = (M')^{ef}/r^e \equiv M'/r^e = M \,(mod\ n)$$

4.2 Lav privat eksponent

For å redusere krypteringstid, eller signatur-genererings-tid, kan vi fortrinnsvis velge å bruke en lav verdi av f fremfor en tilfeldig f, siden modulær eksponering tar tid lineært i $\log_2 f$. En liten f kan forbedre ytelsen med minst en faktor av 10, for en 1024 bit modulus, men et lurt angrep funnet av Wiener [19] viser at en liten f resulterer i en total kollaps av kryptosystemet.

Wieners angrep på RSA bruker kjedebrøker til å finne den private eksponenten f.

4.2.1 Kjedebrøk

Historien om kjedebrøker er lang, og den begynner faktisk i skjult form med tilnærming av kvadratiske irrasjonelle tall, som $\sqrt{2}$. Et annet sted kjedebrøk dukker opp er at den er knyttet til en av de mest kjente algoritmene, Euklids algoritme. Kjedebrøker er også benyttet til tilnærming av tall, og det er dette Wieners angrep baserer seg på. Bosma og Kraaikamp [5] sier at en kjedebrøk er en måte å representere en brøk som en sum av et heltall og en brøk.

$$\frac{p}{q} = a_0 + \cfrac{b_0}{a_1 + \cfrac{b_1}{a_2 + \cfrac{b_2}{\ddots \cfrac{b_n}{a_n}}}}$$

hvor a_i, b_i er naturlige tall, og a_0 angir heltallsverdien. Kjedebrøker hvor $b_i = 1 \forall i$ er standard form for kjedebrøk. Hvis en standard kjedebrøk har et endelig antall a_i, kalles den endelig. For å representere et tall som standard kjedebrøk må tallet skrives som et heltall pluss den inverse av resten av tallet. Dette inverse tallet skrives så på samme måte, som et heltall pluss den inverse av resten av tallet. Denne prosessen fortsetter til man oppnår ønsket resultat. La oss gjøre dette for å representere $\frac{17}{11}$ som standard kjedebrøk:

$$\frac{17}{11} = 1 + \frac{1}{\frac{11}{6}} \qquad \frac{11}{6} = 1 + \frac{1}{\frac{6}{5}}$$

$$\frac{6}{5} = 1 + \frac{1}{\frac{5}{1}} \qquad \frac{5}{1} = 5 + 0$$

Kjedebrøken kan skrives som:

$$\frac{17}{11} = 1 + \frac{1}{\frac{11}{6}} = 1 + \frac{1}{1 + \frac{1}{\frac{6}{5}}} = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{\frac{5}{1}}}} = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{5}}}$$

En standard kjedebrøk kan skrives som en liste av koeffisienter (a_i), hvor skillet mellom heltallet og brøken betegnes med et semikolon:

$$x = [a_0;\ a_1,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6,\ ...]$$

For eksempelet ovenfor:

$$\frac{17}{11} = [1;\ 1,\ 1,\ 5]$$

Den $k$te partielle kjedebrøken er den numeriske verdien eller tilnærmingen som er beregnet ved å bruke de $k-1$ første koeffisientene til kjedebrøken. De første k partielle kjedebrøkene er:

$$\frac{a_0}{1},\ a_0 + \frac{1}{a_1},\ a_0 + \frac{1}{a_1 + \frac{1}{a_2}},\ \ ...\ , a_0 + \frac{1}{a_1 + \frac{\ddots}{a_{k-2} + \frac{1}{a_{k-1}}}}$$

De partielle kjedebrøkene til $\frac{17}{11} = 1, \overline{54}$ er:

$$\frac{1}{1} = 1,\ 1 + \frac{1}{1} = 2 \text{ og } 1 + \frac{1}{1 + \frac{1}{1}} = 1,5$$

For å gå fra en kjedebrøk til en brøk, starter en i høyre side, og forenkler hver sum i kjedebrøken etter tur. Her gjør vi det med kjedebrøken til $\frac{17}{11}$:

$$1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{5}}} = 1 + \frac{1}{1 + \frac{1}{\frac{6}{5}}} = 1 + \frac{1}{1 + \frac{5}{6}} = 1 + \frac{1}{\frac{11}{6}} = 1 + \frac{6}{11} = \frac{17}{11}$$

4.2.2 Wieners teorem om angrep på RSA

Wieners angrep på RSA bruker kjedebrøker til å finne den private eksponenten f når den er mindre enn $\frac{1}{3}\sqrt[4]{n}$.

Teorem 4.2. *[19] La $n = p \cdot q$ med $q < p < 2q$. La $f < \frac{1}{3}n^{\frac{1}{4}}$. Gitt (n, e) med $ef = 1\,(mod\ \varphi(n))$, kan Markus finne f i lineær tid.*

For å forstå Wieners angrep, kan vi se på følgende egenskaper til RSA. Først kan vi merke oss at kongruensen $ef \equiv 1\,(mod\ \varphi(n))$ kan skrives som likningen $ef = k\varphi(n)+1$ for en k. Vi kan også merke oss at $\varphi(n) = (p-1)(q-1) = pq-p-q+1$. Siden både p og q er mye kortere enn $n = pq$, så kan vi si at $\varphi(n) \approx n$. Å dividere likningen ovenfor

med $f\varphi(n)$ gir oss $\frac{e}{\varphi(n)} = \frac{k+1}{f}$, og ved å bruke sistenevnte tilnærming, kan vi skrive $\frac{e}{n} \approx \frac{k}{f}$. Legg merke til hvordan venstre side av likningen kun består av informasjon tilgjengelig for offentligheten, som er en viktig del av angrepet.
Polynomet $(x-q)(x-p)$ vil ha røttene p og q. Utvidelsen gir oss $x^2-(p+q)x+pq$, og bytter vi ut enkelte uttrykk med kjente variabler kan vi skrive dette som $x^2-(n-\varphi(n)+1)x+n$. ABC-formelen gir oss

$$p \wedge q = \frac{-b \pm \sqrt{b^2-4ac}}{2a}, \text{ hvor } a=1,\ b=-(n-\varphi(n)+1), \text{ og } c=n.$$

Wieners angrep består av å representere $\frac{e}{n}$ som en kjedebrøk og iterere gjennom de partielle kjedebrøkene for å sjekke ulike tilnærminger av $\frac{k}{f}$. For å gjøre prosessen med sjekking mer effektiv kan vi gjøre noen observasjoner:

- Siden $\varphi(n)$ er partall, og e og f er begge per definisjon relativt primiske til $\varphi(n)$, så vet vi at f er oddetall.
- Gitt likningene ovenfor og verdiene til e, n, f og k, så kan vi løse for $\varphi(n)$ med likningen $\varphi(n) = \frac{ef-1}{k}$, og vi vet at $ef-1$ må være delelig på k.
- Hvis vår $\varphi(n)$ er korrekt, vil polynomet $x^2-(n-\varphi(n)+1)x+n$ ha røttene p og q, som vi kan sjekke ved at $n=pq$.

4.2.3 Wieners angrep

Anta vi har den offentlige nøkkelen (n,e), så vil følgende angrep bestemme f:

1. Konverter brøken $\frac{e}{n}$ over til en kjedebrøk. $[a_0; a_1, a_2, ..., a_{k-2}, a_{k-1}, a_k]$
2. Iterer de partielle kjedebrøkene:

$$\frac{a_0}{1},\ a_0+\frac{1}{a_1},\ a_0+\frac{1}{a_1+\frac{1}{a_2}},\ \ldots,\ a_0+\frac{1}{a_1+\frac{\ddots}{a_{k-2}+\frac{1}{a_{k-1}}}}$$

3. Sjekk om leddet er $\frac{k}{f}$ ved å gjøre følgende:
   - Sett telleren til å være k og nevneren til å være f.
   - Sjekk om f er odde, hvis ikke, hopp til neste partielle kjedebrøk.
   - Sjekk om $ef \equiv 1\,(mod\ k)$, hvis ikke, hopp til neste partielle kjedebrøk.
   - Sett $\varphi(n) = \frac{ef-1}{k}$ og finn løsningene av polynomet $x^2-(n-\varphi(n)+1)x+n$

- Hvis løsningene av polynomet er heltall, så har vi funnet f. Hvis ikke, hopp til neste partielle kjedebrøk.

4. Hvis alle de partielle kjedebrøkene har blitt testet og ingen av dem ga en f, så er de gitte RSA-parametrene motstandsdyktige mot Wieners angrep.

La oss bruke Wieners angrep mot den offentlige nøkkelen $(5141, 1997)$, i håp om at det er benyttet en lav privat eksponent, $f < \frac{1}{3}\sqrt[4]{n}$, slik at vi kan forsere krypteringen:

1. $\frac{e}{n} = \frac{1997}{5141}$ skrives som kjedebrøken $[0; 2, 1, 1, 2, 1, 6, 4, 10]$.
2. Vi itererer de partielle kjedebrøkene: $\frac{0}{1}, \frac{1}{2}, \frac{1}{3}, \frac{2}{5}, \frac{5}{13}, \frac{7}{18}, \frac{47}{121}, \frac{195}{502}, \frac{1997}{5141}$.
3. Partiell kjedebrøk nummer 1, $\frac{k}{f} = \frac{0}{1}$, har nevner som oddetall, men har 0 som teller. Vi hopper derfor videre til neste partielle kjedebrøk.

   Partiell kjedebrøk nummer 2, $\frac{k}{f} = \frac{1}{2}$, har nevner som partall. Hopper videre til neste partielle kjedebrøk.

   Partiell kjedebrøk nummer 3, $\frac{k}{f} = \frac{1}{3}$, har nevner som oddetall.
   Sjekker om $ef \equiv 1\,(mod\ k)$: $1997 \cdot 3 = 5991 \equiv 1\,(mod\ 1)$
   Setter $\varphi(n) = \frac{1997 \cdot 3 - 1}{1} = 5990$ og finner løsningene av polynomet $x^2 - (5141 - 5990 + 1)x + 5141$:
   $p \wedge q = \frac{-848 \pm \sqrt{(848)^2 - 4 \cdot 5141}}{2} \approx -6,1 \wedge -841,9$, det er ikke heltallsrøtter. Hopper videre til neste partielle kjedebrøk.

   Partiell kjedebrøk nummer 4, $\frac{k}{f} = \frac{2}{5}$, har nevner som oddetall.
   Sjekker om $ef \equiv 1\,(mod\ k)$: $1997 \cdot 5 = 9985 \equiv 1\,(mod\ 2)$
   Setter $\varphi(n) = \frac{1997 \cdot 5 - 1}{2} = 4992$ og finner løsningene av polynomet $x^2 - (5141 - 4992 + 1)x + 5141$:
   $p \wedge q = \frac{150 \pm \sqrt{(-150)^2 - 4 \cdot 5141}}{2} = 97 \wedge 53$, hvor $53 \cdot 97 = 5141$. Vi har funnet den private eksponenten, $f = 5$, sammen med $\varphi(n) = 4992$, $p = 97$ og $q = 53$. Hele krypteringen er nå eksponert.

4.2.4 Tiltak mot Wieners angrep

Hvis n er på 1024 bits, følger det at f må være minst 256 bits lang for å avverge Wiener sitt angrep. Dette er beklagelig for enheter med lav ytekraft, slik som 'smart-kort', hvor en liten f vil resultere i stor tidsbesparelse. Alt er derimot ikke håpløst, Wiener beskriver et flust av teknikker som legger til rette for kjapp dekryptering og som ikke er mottakelige for hans angrep. Her beskriver vi to av dem:

Stor e: Anta at vi istedenfor å redusere e modulus $\varphi(n)$, bruker (n, e') som offentlig nøkkel, hvor $e' = e + t \cdot \varphi(n)$ for en stor t. Det er klart at e' kan benyttes istedenfor e for å kryptere beskjeder. Når en stor verdi av e blir benyttet, vil k i Wieners bevis ikke være liten lengre. En enkel kalkulering viser at hvis $e' > n^{1,5}$ kan f være av enhver størrelse, og angrepet til Wiener vil ikke fungere. Uheldigvis resulterer store verdier av e til økt krypteringstid.

Chinese Reminder Theorem: Anta at vi velger f slik at både $f_p = f \, (mod \; p-1)$ og $f_q = f \, (mod \; q-1)$ er små, la oss si 128 bits hver. Da kan dekryptering av chiffertekst C skje hurtig ved følgende metode: beregn først $M_p = C^{f_p} \, (mod \; p)$ og $M_q = C^{f_q} \, (mod \; q)$. Bruk deretter CRT (Chinese Reminder Theorem) til å beregne den unike verdien $M \in \mathbb{Z}_n$ som tilfredsstiller $M = M_p \, (mod \; p)$ og $M = M_q \, (mod \; q)$. Den resulterende M vil også etterkomme $M = C^f \, (mod \; n)$. Poenget er at selv om f_p og f_q er små, så kan verdien av $f \, (mod \; \varphi(n))$ være stor. Som et resultat vil angrepet til Wiener ikke fungere. Vi merker at hvis (n, e) er gitt så eksisterer det et angrep som gir en motstander mulighet til å faktorisere n på tiden $O(\min(\sqrt{f_p}, \sqrt{f_q}))$, og derfor kan ikke f_p og f_q være for små.

Vi vet ikke om noen av disse metodene er sikre, men vi vet at Wieners angrep er ineffektivt mot dem. Teorem 4.2 har blitt utbedret av Boneh og Durfee [4] som viser at så lenge $f < n^{0.292}$, kan en motstander effektivt få tak i f fra (n, e).

4.2.5 Bevis av Wieners teorem

Beviset er basert på tilnærming ved kjedebrøker. Eksponenten f er beregnet i likningen $e \cdot f \equiv 1 \, (mod \; \varphi(n))$, hvor denne relasjonen er nødvendig for at den offentlige eksponenten og den private eksponenten skal være inverser av hverandre. Det følger at det eksisterer et heltall k slik at $e \cdot f - k \cdot \varphi(n) = 1$. Derfor har vi at:

$$\left| \frac{e}{\varphi(n)} - \frac{k}{f} \right| = \frac{1}{f \cdot \varphi(n)}$$

Siden $n = p \cdot q > q^2$, har vi at $q < \sqrt{n}$ som fører til at vi har:

$$0 < n - \varphi(n) = p + q - 1 < 3 \cdot q - 1 < 3 \cdot q = 3 \cdot \sqrt{q^2} < 3 \cdot \sqrt{n}$$

Siden $\varphi(n) = n - p - q + 1$ og $p + q - 1 < 3 \cdot \sqrt{n}$, har vi at $|n - \varphi(n)| < 3 \cdot \sqrt{n}$. Med $\varphi(n)$ ukjent, så brukes n som en tilnærming. Med denne tilnærmingen ser vi at

$$\left|\frac{e}{n} - \frac{k}{f}\right| = \left|\frac{e \cdot f - k \cdot n}{f \cdot n}\right| = \left|\frac{e \cdot f - k \cdot \varphi(n) - k \cdot n + k \cdot \varphi(n)}{f \cdot n}\right| =$$
$$\left|\frac{1 - k \cdot (n - \varphi(n))}{f \cdot n}\right| \leqslant \left|\frac{3 \cdot k \cdot \sqrt{n}}{f \cdot n}\right| = \frac{3 \cdot k}{f \cdot \sqrt{n}}$$

Vi har at $k \cdot \varphi(n) = e \cdot f - 1 < e \cdot f$, og siden $e < \varphi(n)$, får vi $k < f < \frac{1}{3} n^{\frac{1}{4}}$, som fører til:

$$\left|\frac{e}{n} - \frac{k}{f}\right| < \frac{1}{f \cdot n^{\frac{1}{4}}}$$

Ettersom $3 \cdot f < n^{\frac{1}{4}}$ har vi at

$$\left|\frac{e}{n} - \frac{k}{f}\right| < \frac{1}{3 \cdot f^2} < \frac{1}{2 \cdot f^2}$$

Antallet brøker $\frac{k}{f}$ med $f < n$ som tilnærmer seg $\frac{e}{n}$ så tett er avgrenset av $\log_2 n$. Alle slike tilnærmede brøker er innehentet som partielle kjedebrøker av $\frac{e}{n}$ [8, Teorem 177]. En trenger å beregne de log n partielle kjedebrøkene av $\frac{e}{n}$, og en av disse vil være lik $\frac{k}{f}$. Ettersom $e \cdot f - k \cdot \varphi(n) = 1$, har vi $\mathrm{GCD}(k, d) = 1$, som betyr at $\frac{k}{f}$ er en brøk på forkortet form. Dette er en algoritme for å rekonstruere den private nøkkelen f i lineær-tid.

4.3 Lav offentlig eksponent

For å redusere kryptering eller signaturverifiseringstid, er det vanlig å bruke en liten offentlig eksponent e. Den minst mulige verdien eksponent e kan være er 3, men for å forhindre enkelte angrep er $e > 2^{16} + 1 = 65537$ anbefalt. Når verdien $e = 2^{16} + 1$ brukes krever signaturverifiseringen 17 multipliseringer, i motsetning til omtrent 1000 ved bruk av en tilfeldig $e \leqslant \varphi(n)$. Angrep som gjelder en liten e gir ikke en total kollaps av kryptosystemet, i stor kontrast til angrep beskrevet i forrige delkapittel.

4.3.1 Franklin-Reiter angrep på relaterte meldinger

Franklin og Reiter fant et angrep mot RSA når Ola sender flere relaterte krypterte meldinger til Kari ved bruk av samme modulus. Hvis to meldinger avviker fra hverandre med en kjent, fast forskjell og er RSA-kryptert under samme modulus n, så er

det mulig å forsere krypteringen, og rekonstruere meldingene.

La (n, e) være Kari sin offentlige nøkkel. Anta $M_1, M_2 \in \mathbb{Z}_n^*$ er to ulike meldinger som tilfredsstiller $M_1 = f(M_2) \, (mod \; n)$ for et offentlig kjent polynom $f \in \mathbb{Z}_n^*[x]$. For å sende M_1 og M_2 til Kari, så er kanskje Ola naiv og krypterer meldingene og sender chiffertekstene C_1 og C_2. Vi skal vise at gitt C_1 og C_2 kan Markus enkelt få tak i M_1 og M_2.

Denne situasjonen oppstår ganske ofte, som for eksempel bokstaver sendt til forskjellige adresser, tekster hvor datoen er det eneste som skiller dem fra hverandre for kompilering, eller en ny sending av en melding med et nytt ID-nummer på grunn av en feil.

Angrepet var originalt beskrevet med offentlig eksponent $e = 3$, men fungerer for enhver liten e.

Teorem 4.3. *La (n, e) være en RSA offentlig nøkkel. La $M_1 \neq M_2 \in \mathbb{Z}_n^*$ tilfredsstille $M_1 \equiv f(M_2) \, (mod \; n)$ for en lineær $f = ax + b \in \mathbb{Z}_n^*[x]$ med $b \neq 0$. Da vil Markus med (n, e, C_1, C_2, f) gitt, kunne gjenopprette M_1 og M_2 i kvadratisk tid i log n.*

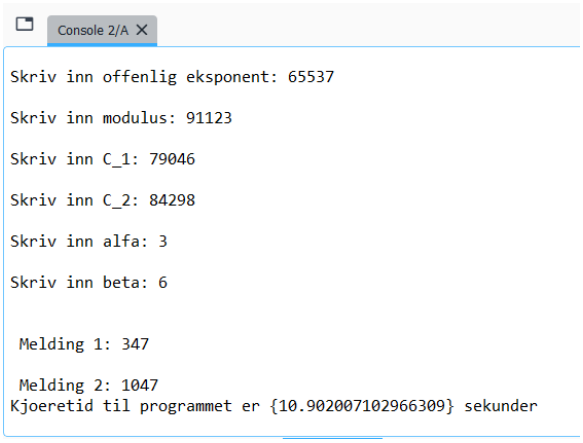
Bevis. Siden $C_1 = M_1^e \, (mod \; n)$, vet vi at M_2 er en løsning av polynomet $g_1(x) = f(x)^e - C_1 \in \mathbb{Z}_n[x]$. Likeså er M_2 en løsning av $g_2(x) = x^e - C_2 \in \mathbb{Z}_n[x]$. Den lineære faktoren $x - M_2$ deler begge polynomene. Således kan Markus bruke Euklids algoritme til å beregne største felles nevner av g_1 og g_2. Hvis $GCD(g_1, g_2)$ er lineær, så er M_2 funnet. $GCD(g_1, g_2)$ kan bli beregnet i kvadratisk tid i e og log n.

Polynomet $x^e - C_2$ faktoriserer både modulus p og q til lineære faktorer og kvadratiske faktorer ute av stand til å reduseres mer. Husk at $GCD(e, \varphi(n)) = 1$ og derfor har $x^e - C_2$ bare én løsning i $\mathbb{Z}_n$. Siden g_2 ikke kan dele g_1, må GCD være lineær. Imidlertid, for noen sjeldne M_1, M_2, og f, er det mulig å få en ikke-lineær GCD, hvis det er tilfellet vil ikke angrepet fungere. □

For å se et eksempel av Franklin-Reiter angrepet setter vi opp følgende scenario: Den offentlige krypteringseksponenten er $e = 2^{16} + 1 = 65537$ og modulus er 91123. La $f = ax + b$, være polynomet $3x + 6$, med $a = 3$ og $b = 6$. Anta at Markus har fanget opp to krypterte meldinger $C_1 = 79046$ og $C_2 = 84298$ fra Ola til Kari, hvor vi vet at $C_1 \equiv M_1^e \, (mod \; n)$ og $C_2 \equiv (a \cdot M_1 + b)^e \, (mod \; n)$.

For at Markus skal finne den opprinnelige meldingen kan han beregne $GCD(z^{65537} - 79046, (3z + 6)^{65537} - 84298) \in \mathbb{Z}/91123[z]$. Å bruke Euklids algoritme på polynomene

for hånd vil være særdeles tungvindt og ekstremt tidkrevende. Coppersmith, Franklin, Patarin og Reiter [6] fant i 1996 en polynomrelasjon mellom g_1 og g_2, nemlig $g_1 = z^e - c_1 = 0\,(mod\ n)$ og $g_2 = (a \cdot z + b)^e - c_2 = 0\,(mod\ n)$. Istedenfor å regne $GCD(g_1, g_2)$ for hånd bruker Markus pythonprogram 1, som benytter seg av polynomrelasjonen og finner originalbeskjedene $M_1 = 347$ og $M_2 = 1047$ på omtrent 11 sekunder:



Figur 3: Franklin-Reiter angrep

5 Modulær aritmetikk og kryptografi i skolen

Hvorfor undervise i kryptografi i skolen? Dr. Katharina Klembalski [9] mener at kryptografi bør være en del av undervisningen i skolen. Klembalski påpeker at i elektronisk datatrafikk er kryptografi av praktisk relevans i hverdagen, eksempelvis nettbank, e-post, elektroniske helsekort, elektroniske pass og beskyttelse av personlige data. Disse applikasjonene sikrer kryptografi hemmelighold av data som utveksles, og gir pålitelige midler for autentisering av kommunikasjonsdeltakere og for verifisering av integriteten til data. Klembalski sier at undervisning i kryptografi vil øke bevisstheten i forhold til datasikkerhet, og at informasjon utvekslet på internett i prinsippet kan overvåkes og dermed er usikkert. Videre mener hun at undervisningen kan gi innsikt i nødvendigheten av kryptering og evnen til å utføre og verifisere kryptering, samt at det dannes en forståelse av hvordan identiteten til kommunikasjonsdeltakere kan verifiseres. En bedre forståelse av kryptografi kan redusere potensiell angst og frykt for overvåking av informasjon og digital aktivitet.

Klembalski har flere gode argumenter for at kryptografi bør inn i skolen. Videre i dette kapittelet skal vi se hva flere forskere tenker om hvordan modulær aritmetikk

og kryptografi brukes i skolen og i matematikkfaget spesielt.

I Tyskland finnes undervisningspraksisen av kryptografi hovedsakelig i databehandligsfag, fremfor i matematikkfaget, sier Klembalski [9]. Ifølge Klembalski påvirker dette arten og omfanget av det matematiske fundamentet som presenteres. Det matematiske fundamentet blir ofte mindre prioritert til fordel for implementering av individuelle algoritmer. Lignende tilnærminger for å undervise i spesifikt matematisk innhold ved hjelp av kryptografi som går utover de rene kryptografiske algoritmene eller protokollene, er sjeldne og er ofte bare implisitte. Denne måten å undervise på fører til at det matematiske potensialet forblir ubrukt, mener Klembalski. Også på Universitetet i Stavanger finnes undervisningspraksisen av kryptografi i databehandligsfag, og ikke i matematikkfag. Kryptografi er ikke et emne som instituttet for matematikk og fysikk tilbyr, men kryptografi er en del av emnet 'Sikkerhet og sårbarhet i nettverk DAT510' som tilbys av instituttet for data- og elektroteknologi.

Det er ingen tvil om at korrelasjonen mellom kryptografi og datafag er stor. Både skoler i Tyskland og ihvertfall ett universitet i Norge har tatt en vurdering om at kryptografi skal undervises i sammenheng med datasikkerhet fremfor matematikk. I fra et matematisk synspunkt er dette et beklagelig resultat. Undervisning i kryptografi fra et datasikkerhets synspunkt fører til at den underliggende matematikken er mindre interessant. Dette kan føre til at elevene ikke tilegner seg matematiske kunnskaper for å validere resultater.

Maynard-Hahnfeld [12] underviste i kryptografi på High School allerede i 2001, og kom til konklusjonen at kryptografi bør være et emne som får elever til å identifisere og diskutere kryptografiske prosedyrer som angår oss i hverdagen. Han foreslår samtidig at kryptografi ikke bør være et eget fag, men et tema i både matematikk og datafag. Det er verdt å nevne at denne undervisningen gikk under faget Computer Science og ikke matematikk.

I 1997 framstilte Koblitz [11] tre grunner til å undervise i kryptografi i matematikk. For det første setter kryptografi matematikk i en dramatisk setting. Barn og ungdom er fascinert av intriger og eventyr. Det kan stå mer på spill for eleven enn en karakter: Hvis du gjør en feil, blir agenten din bli forrådt. For det andre gir kryptografi en naturlig måte å få elevene til å oppdage enkelte viktige matematiske begreper og teknikker på egen hånd. Altfor ofte serverer mattelærere alt på et sølvfat, ifølge Koblitz, og frarøver dermed elevene gleden av oppdagelsen og læringserfaringen. Derimot, hvis ungdommene etter grublig og hardt arbeid endelig utvikler en metode for å bryte et kryptosystem, vil de mest sannsynlig sette pris på kraften og skjønnheten i matematikken de har avdekket. Noen av de matematiske teknikkene elevene kan

gjenoppdage fra klassisk matematikk er euklidsk algoritme og gaussisk eliminering. For det tredje sier Koblitz at et sentralt tema i kryptografi er det vi ikke vet eller ikke kan gjøre. Sikkerheten til et kryptosystem hviler ofte på vår manglende evne til effektivt å løse et problem i algebra, tallteori eller kombinatorikk. Dermed er kryptografi en måte å motvirke inntrykket til elevene om at med riktig formel og en god datamaskin kan ethvert matteproblem løses raskt.

Maynard-Hahnfeld [12] mener at kryptografi bør undervises på en interaktiv og utforskende måte. Dette gjør det mulig for elevene å oppdage chiffer av seg selv. Elevene kan enkelt kryptere eller dekryptere ved hjelp av nøkler etter eget valg. Den interaktive opplæringen legger opp til at elevene kan jobbe individuelt eller i grupper om nødvendig. Maynard-Hahnfeld [12] underviste med interaktiv opplæring og mente at dette skapte en høy grad av samhandling mellom elevene, noe som ifølge han er ganske sjeldent i en tradisjonell klasserom setting, som gjør at elevene kan få kunnskap på en rask måte. I tillegg har elevene mulighet til å lære i sitt eget tempo. Det må imidlertid tas hensyn til at undervisning av kryptografi ikke forveksles med et programmeringsfag, og Maynard-Hahnfeld presiserer at papir- og blyantarbeid og debatt om tankevekkende spørsmål er like viktig som programmering. Gitt tilstrekkelig programmeringskunnskap, bør elevene lage sine egne krypteringer og være i stand til å diskutere dens sikkerhet.

Maynard-Hahnfeld [12] støtte på to utfordringer i sin undervisning av kryptografi på High school. Den første utfordringen var at elevene var motvillige til å gjøre ikketrivielle modulære beregninger for hånd. Eksempelvis fant elevene det unødvendig å beregne modulær multiplikativ invers for hånd ettersom det kunne gjøres enkelt via programmering. Maynard-Hahnfeld peker på viktigheten av at elevene forstår hvordan et svar er beregnet, slik at de er i stand til å verifisere svaret, og feil svar kan oppdages og bli tatt vekk. Den andre utfordringen han støtte på var at selv om alle elevene har muligheten til å lære den underliggende matematikken til RSA kryptering var ikke alle elevene villige til å gå inn i underliggende matematikk.

Det er viktig å gi elevene et autentisk bilde av matematikken som vitenskap, og da er det nødvendig å vise nåværende utvikling i vitenskapelig matematikk, sier både Klembalski [9] og Maynard-Hahnfeld [12]. Til dette formålet sier Klembalski at kryptologi er et godt alternativ, ettersom mange av dens moderne teknikker og algoritmer kan forklares fullt ut og krever liten matematisk bakgrunn for å bli forstått. De fleste av de åpne vitenskapelige spørsmålene innenfor matematikk er vanskelige å beskrive på skolenivå og er dermed knapt tilgjengelig for elevene. Noen åpne spørsmål innen grener som diskret matematikk, tallteori eller tall er på et matematisk nivå som gjør dem forståelige for elever. Undervisning i kryptografi gir en gyllen mulig-

het til å la elevene møte åpne spørsmål, ifølge Klembalski. Klembalski [9] påpeker også at kryptografi spesielt berører de matematiske konseptene tall og algoritmer. Kunnskapen om naturlige tall blir større. Årsaken til dette, sier hun, er spørsmål om nøkkelkonstruksjon for RSA-algoritmen og dens sikkerhet. Disse spørsmålene fører til primtall, deres distribusjon, primtalltesting og problemet med faktoriseringen. I tillegg er kjente delelighets-tester bevist med hjelp av modulær aritmetikk og legitimeres dermed retrospektivt. Kryptografi er egnet til å utfordre elevene i uløste spørsmål innenfor matematisk vitenskap i en autentisk kontekst. Samtidig er kryptografi et eksempel som motvirker den vanlige elevoppfatningen om at matematisk kunnskap er en fast struktur som bare utvides. Ved å jobbe med kryptografi fremstår matematikk som en vitenskap hvor det genereres nye grener og engasjerer til en konstant utveksling av ideer med praktiske applikasjoner. Spørsmålet om nøkkelgenerering fører til den euklidske algoritmen, som elevene kan oppdage av seg selv. Dette illustrerer fordelene ved algoritmisk problemløsning; spørsmål om effektiviteten av algoritmer vil oppstå nesten av seg selv. Viktigheten av datamaskinbruk for anvendelse av disse og andre algoritmer kan knyttes til kryptografiens historie. Dette står i kontrast til normal bruk av algoritmer i videregående skole (senior secondary level), som vanligvis er begrenset til å løse ligningssystemer eller til behandling av kalkulus problemer (kurve skissering), sier Klembalski.

Både Klembalski, Maynard-Hahnfeld og Koblitz mener at modulær aritmetikk og kryptografi bør inn i faget matematikk, men hva sier læreplanen Kunnskapsløftet 2020 (LK20) om modulær aritmetikk og kryptografi i matematikkfaget? I neste del av oppgaven skal vi se nærmere på dette spørsmålet. Vi skal også se om modulær aritmetikk eller kryptografi omtales i andre læreplaner. Oppgaven begrenses til 8.-13. trinn.

5.1 Hva sier læreplanen?

I læreplanverket Kunnskapsløftet (LK20) [15] er det ingen kompetansemål for ungdomstrinnet som er direkte knyttet til modulær aritmetikk eller kryptografi. I videregående opplæring er det kun i faget matematikk X, som bygger på matematikk 1T og er spesielt beregnet for elever på Vg2 som velger matematikk R1, hvor modulær aritmetikk og kryptering er knyttet til kompetansemålene. I matematikk X [16] er det fire kompetansemål som er relevante for modulær aritmetikk fra læreplanen, og det fjerde er også relevant for RSA kryptering:

- Utforske og gjøre rede for kongruenser av hele tall, og beskrive oppdagede regneregler og sammenhenger med et formelt symbolspråk
- Bruke kongruensregning til å analysere delelighet og løse lineære kongruenslig-

ninger

- Utforske og løse diofantiske ligninger og argumentere for eventuelle løsninger
- Gjøre rede for og presentere anvendelser av matematisk teori innenfor tallteori og geometri

Før fagfornyelsen av læreplanen var det også kun matematikk X som hadde relevante kompetansemål knyttet til modulær aritmetikk og kryptering [14]. Da var kompetansemålene tydeligere på kryptering sammenlignet med de nye kompetansemålene:

- Bruke kongruensregning til å analysere delelighet, løse lineære kongruenslikninger og avgjøre om enkle diofantiske likninger har løsninger
- Gjøre rede for praktiske anvendelser av kongruensregning i kryptering og feilrettingskoder
- Planlegge, utføre og presentere et selvstendig utforskende arbeid i et emne tilknyttet hovedområdet 'Tallteori'

Etter fagfornyelsen har kompetansemålene blitt mer åpne og må tolkes. Modulær aritmetikk er godt ivaretatt i læreplanen til matematikk X også etter fagfornyelsen, ettersom kongruenser, kongruensregning, kongruensligninger og diofantiske ligninger går under modulær aritmetikk. Vi ser derimot at kryptografi er utelatt etter fagfornyelsen og det er opp til faglærer å definere setningen 'gjøre rede for og presentere anvendelser av matematisk teori innenfor tallteori' og ut ifra dette vurdere å eventuelt undervise i temaet kryptering. Ettersom ordet kryptering er utelatt i matematikk X etter fagfornyelsen, og heller ikke finnes noe annet sted i læreplanen kan det tolkes som at kryptering ikke er et viktig tema for fremtidens elever. Dette er en undrende utvikling siden kryptering er en essensiell del av vår digitale hverdag.

Ettersom modulær aritmetikk og kryptering ikke finnes i kompetansemål, bortsett fra i matematikk X, vil vi se på hva læreplanen sier om faget matematikk. Under fagets relevans og sentrale verdier kan vi trekke fram nøkkelord som kan relateres til modulær aritmetikk og kryptering: mønster, sammenhenger, anvendelse, kritisk tenkning, kommunikasjon, utforsking, problemløsing, resonnere matematisk, reflektere, relevans, kreativitet og skapertrang. Disse nøkkelordene henger sammen med flere kjerneelementer, som også kan knyttes til modulær aritmetikk og kryptering, særlig kjerneelementet 'utforsking og problemløsing'. I kjerneelementene finner vi flere elementer som kan kobles til modulær aritmetikk eller kryptering, det står blant annet at elevene skal få innsikt i hvordan matematikk anvendes utenfor faget, de skal utforme egne resonnementer for å løse og forstå problemer, samt lete etter mønster og finne sammenhenger for å løse et problem som ikke er kjent. Det skal også legges det opp

til algoritmisk tenkning.

I tillegg til matematikk X kan vi finne spor av kryptografi i vg2 faget informasjonsteknologi [17] og vg3 faget IT-drift [18]. I begge fagene omtales datasikkerhet under det tverrfaglige temaet 'demokrati og medborgerskap'. Kryptering er en viktig del av datasikkerheten for individet, samfunnet og arbeidslivet.

Dersom vi ønsker å undervise i modulær aritmetikk og/eller kryptografi i matematikk, kan det se ut til at vi må koble det opp mot fagets relevans og sentrale verdier, kjerneelementer eller tverrfaglighet, siden ingen av kompetansemålene til matematikkfagene, utenom matematikk X, omhandler modulær aritmetikk eller kryptografi.

Referanser

[1] Johan F Aarnes. Aritmetikk. https://snl.no/aritmetikk. [På internett; besøkt 26-april-2022].

[2] Elaine Barker and Quynh Dang. Recommendation for key management - nist. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf, Januar 2015.

[3] Dan Boneh. Twenty years of attacks on the rsa cryptosystem. https://www.researchgate.net/publication/2538368_Twenty_Years_of_Attacks_on_the_RSA_Cryptosystem, Februar 2002.

[4] Dan Boneh and Glenn Durfee. Cryptanalysis of rsa with private key d less than 0.292. https://staff.emu.edu.tr/alexanderchefranov/Documents/CMSE491/Fall2019/BonehIEEETIT2000%20Cryptanalysis%20of%20RSA.pdf.

[5] Wieb Bosma and Cor Kraaikamp. Continued fractions. https://www.math.ru.nl/~bosma/Students/CF.pdf?msclkid=88fbf57ac16911ec9efa2f740a52cf9b, 2013.

[6] Don Coppersmith, Matthew Franklin, Jacques Patarin, and Michael Reiter. Low-exponent rsa with related messages. https://rd.springer.com/chapter/10.1007/3-540-68339-9_1, Mai 1996.

[7] Datatilsynet. Kryptering. https://www.datatilsynet.no/rettigheter-og-plikter/virksomhetenes-plikter/informasjonssikkerhet-internkontroll/kryptering/. [På internett; besøkt 26-april-2022].

[8] G. H. Hardy, E. M. Wright, J. H. Silverman, and D. R. Heath-Brown. An introduction to the theory of numbers. https://books.google.no/books?id=P6uTBqOa3T4C&printsec=frontcover&hl=no&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false, 2008.

[9] Katharina Klembalski. Cryptography and number theory in the classroom – contribution of cryptography to mathemtics teaching. http://math.unipa.it/~grim/21_project/klembalski323-327.pdf, 2009.

[10] Svein Johan Knapskog and Øyvind Eilertsen. Kryptografi. https://snl.no/kryptografi. [På internett; besøkt 26-april-2022].

[11] Neal Koblitz. Cryptography as a teaching tool. https://sites.math.washington.edu/~koblitz/crlogia.html#:~:text=Cryptography%20As%

20a%20Teaching%20Tool%20Cryptography%20has%20a,a%20grade%20on%20a%20test%3A%20if%20you%20make, 1997.

[12] Shawn E. Maynard-Hahnfeld. Teaching cryptography in high school. https://www.ti89.com/cryptotut/text/Ch5.doc, Mars 2001.

[13] Robert Milson. Introduction to the rsa algorithm and modular arithmetic. https://www.researchgate.net/publication/1845076_Introduction_to_the_RSA_algorithm_and_modular_arithmetic.

[14] Utdanningsforbundet. Læreplan i matematikk x - programfag i utdanningsprogram for studiespesialisering (mat2-01). https://www.udir.no/kl06/MAT2-01/Hele/Kompetansemaal/matematikk-x. [På internett; besøkt 26-april-2022].

[15] Utdanningsforbundet. Læreplanverket. https://www.udir.no/laring-og-trivsel/lareplanverket. [På internett; besøkt 26-april-2022].

[16] Utdanningsforbundet. Matematikk x (mat02-02) kompetansemål og vurdering. https://www.udir.no/lk20/mat02-02/kompetansemaal-og-vurdering/kv465. [På internett; besøkt 26-april-2022].

[17] Utdanningsforbundet. Vg2 informasjonsteknologi (itk02-01) - tverrfaglige temaer. https://www.udir.no/lk20/itk02-01/om-faget/tverrfaglige-temaer. [På internett; besøkt 26-april-2022].

[18] Utdanningsforbundet. Vg3 it-driftsfaget (itd03-01) - tverrfaglige temaer. https://www.udir.no/lk20/itd03-01/om-faget/tverrfaglige-temaer. [På internett; besøkt 26-april-2022].

[19] Michael J Wiener. Cryptanalysis of short rsa secret exponents. https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/krypto2ss08/shortsecretexponents.pdf, August 1989.

[20] Wikipedia. Blokkchiffer. https://no.wikipedia.org/w/index.php?title=Blokkchiffer&oldid=20904041, 2020. [På internett; besøkt 26-april-2022].

Vedlegg

Listing 1: **Franklin-Reiter angrep, pythonprogram**

```
import time

###################################
### Gitt informasjon
###################################

e=int(input("Skriv inn offenlig eksponent: "))
n=int(input("Skriv inn modulus: "))
c_1=int(input("Skriv inn C_1: "))
c_2=int(input("Skriv inn C_2: "))
alfa=int(input("Skriv inn alfa: "))
beta=int(input("Skriv inn beta: "))

# starter timer
start = time.time()
###################################
### Utregninger.
###################################

# Omformer uttrykket z^e-c til z^e+c'
c_1_hat=n-c_1
c_2_hat=n-c_2
# Setter initalverdi z=0
z=0
# Definerer polynomene
def g_1(z):
    return((z**e+c_1_hat))
def g_2(z):
    return((alfa*z+beta)**e+c_2_hat)
# Regner ut initial rest-verdi til polynomene
# dividert paa modulusen
g_1_hat=g_1(z)%n
g_2_hat=g_2(z)%n
# Faar z til aa tilfredsstille de polynomiale
# relasjonene z^e+c1'=0(mod n) og (az+b)^e+c2'=0(mod n)
if g_1_hat==0 and g_2_hat==0:
```

```
    print('\n\nMelding 1:',z,'\n'
    ,'Melding 2:',alfa*z+beta)
else:
    while g_1_hat!=0 and g_2_hat!=0:
        z+=1
        g_1_hat=g_1(z)%n
        g_2_hat=g_2(z)%n
    print('\n\n','Melding 1:',z,'\n\n'
    ,'Melding 2:',alfa*z+beta)

# stopp timer
stopp = time.time()

# total tid brukt
print('Kjoeretid til programmet er',
    {stopp - start},'sekunder')
```