*Universitetet i Oslo - IN2120 Informasjonssikkerhet – 2022 – midtsemestereksamen*

# Historien og Utviklingen av Kryptografi – fra Steintavler til Kvantekryptering

## T397

## Kryptografi


Kandidatnummer:

15193

15169

15327


Antall ord: 6522

## Innhold

# 1 Innledning

Så lenge vi har hatt evnen til å dele informasjon, særlig språklig, har vi hatt nødvendigheten til å skjule slik informasjon. Det er her oppfinnelsen av kryptografi kommer inn. Ordet *kryptografi* kommer opprinnelig fra Hellas og betyr *hemmelig melding*. Siden før tiden av historiske opptak har skjuling av informasjon vært en del av menneskeliv. Før de første kjente krypteringsalgoritmene ble oppfunnet var det andre måter å skjule viktig informasjon på. Grekerne kalte dette steganografi. Dette gikk ut på å fremstille hemmelig informasjon gjemt blant en annen melding eller fysisk objekt. Dette viser oss hvor tidløs idéen om kryptering er.

Motivasjonen for denne oppgaven ligger i at kryptografi er et av de viktigste elementene for informasjonssikkerhet. Evnen til å kryptere og dekryptere meldinger der ingen andre kan se dem bortsett fra deg og noen andre er liv og død for noen. Kryptografi og kryptoanalyse kan finnes overalt i vårt moderne liv fra mobiler, datamaskiner, banken og til og med en nettbutikk. Men vi skal se fra bruksområdene til krypteringsalgoritmene vi skal undersøke i denne oppgaven at det vil være en spesielt fremtredende konsument av kryptografiske løsninger, nemlig militæret. Dette vil vedvare som hovedmålgruppen for kryptografi frem til det forekommer et stort skifte ved introduksjonen av datamaskinen og vi beveger oss inn i et nytt paradigme for kryptografi i skiftet til den moderne verden.

Kombinasjonen av disse faktorene danner et svært interessant fagområde innenfor datavitenskap. Derfor skal vi i denne oppgaven undersøke historien av kryptografi mens vi ser på utviklingene gjort sammenlignet med de tidligere algoritmene underveis. Selv om noen av de senere algoritmene vi skal se på kanskje virker såpass fjernt relatert til de originale, skal vi se at det er mulig å trekke en kjede med utvikling der algoritmen sakte inkrementeres til å passe nye krav mens den gjøres mer robust og avansert. Denne langstrakte utviklingen gjennom historien er veldig interessant og ved å gå gjennom historien av kryptografi i kronologisk rekkefølge kan man få et nytt innblikk i hvordan vi har endt opp med de algoritmene vi har i dag.

Som vi skal se var de tidligste eksemplene på kryptografi gjerne basert på mye mer trivielle algoritmer sammenlignet med det vi har i dag. Disse var gjerne transposisjons- og substitusjonschiffere. Derfor er dette det første vi skal se på. Nemlig implementasjonene av disse chifferne, hva de ble brukt til og til slutt hvordan man best kan kombinere og utvide på dem for å lage ganske robuste krypteringsalgoritmer. Algoritmene vi skal gå gjennom i denne delen utgjør det første hovedkapitlet om historisk kryptografi.

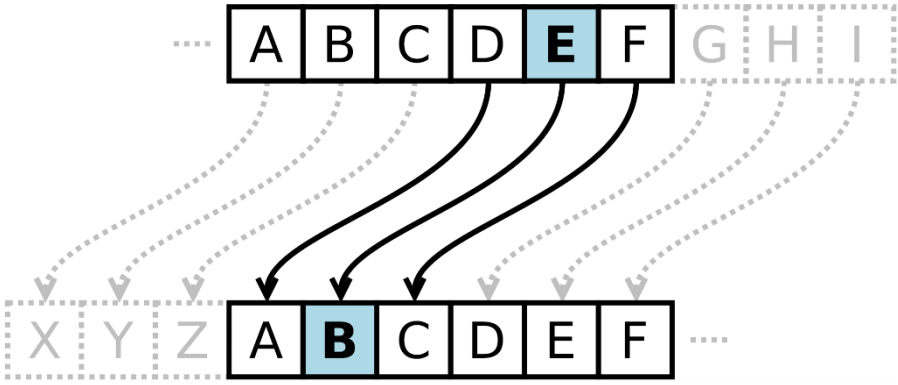
Selv de mest avanserte algoritmene vi skal se på i den første delen, er ikke sammenlignbare med krypteringsmetodene vi har i moderne kryptografi. Kryptografien begynner å ta form etter 1. verdenskrig. Rett før slutten av 1. verdenskrig ble rotormaskinen oppfunnet hvor en nøkkel er innebygd i en roterende skive. Dette ville være forløperen til det som ender opp som Enigma-maskinen. Dette vil bli brukt av det tyske militæret på grunn av dets effektivitet for kryptering. Men selv enigma er en foreldet krypteringsalgoritme i begynnelsen av informasjonstiden hvor vi har datamaskiner raskere enn man noen gang kunne drømme om sammen med komplekse algoritmer for å dekryptere all informasjon kryptert med gamle metoder. Med utviklingen innen maskinvare har vi også gjort fremskritt innen kryptografi. Denne utviklingen skal vi se med Diffie-Hellman nøkkelutveksling og RSA, før vi avslutter med det nyeste innen verden av kryptografi – kvantekryptering. Disse algoritmene vil utgjøre det andre hovedkapitlet om moderne kryptografi.

# 2 Historisk Kryptografi

## 2.1 Substitusjonschiffer

Substitusjonschiffere virker ved å bytte ut hvert enkelt tegn i en melding med en fast erstatning. Hvordan man definerer hva man skal erstatte hvert tegn med kan gjøres på et par forskjellige måter. For eksempel kan du ha en forhåndsbestemt tabell med verdier der du setter en bestemt verdi til hver bokstav. Verdiene i chifferteksten her trenger ikke nødvendigvis å være bokstaver, du kan for eksempel bruke tall, tegnsetting, mellomrom, figurer/tegninger for å gjøre algoritmen mer robust. Men antageligvis den mest kjente metoden er å bruke et forskjøvet alfabet slik som i Cæsar-chiffer. I denne algoritmen tar man den originale meldingen og forskyver hver bokstav X antall plasser rundt i alfabetet. For eksempel hvis man velger +2 plasser blir hver forekomst av A til C, B til D, C til E, osv.

I forhold til historisk bruk er opprinnelsen til Cæsar-chiffer kreditert til dens navn navnebror Julius Cæsar. Algoritmen ble brukt for å beskytte militære hemmeligheter og skal historisk ha blitt brukt med et skift på 3 bokstaver, dvs. at A blir til D ved kryptering og D til A ved dekryptering. Figur 1 viser hvordan alfabetet hadde blitt dekryptert ved Cæsar tradisjonelle bruk av chifferet. Selv om andre former for substitusjonschiffere skal ha vært i bruk før Cæsar-chiffer, er dette den første historiske registrerte, hvilket gjør den spesiell for historien av kryptografi. I virkeligheten er mye av detaljene rundt algoritmen tapt, så mesteparten av det vi vet nå kommer fra senere historikeres beskrivelser, spesielt Suetonius . (RomeAndArt, 2022)




*Figur 1*
*(Cepheus, 2006)*

Figur 2 viser en chiffer disk brukt til å både kryptere og dekryptere slike meldinger der chiffer disken er instilt til å forskyve alfabetet 13 plasser. Disken ble utviklet i 1470 av Leon Battista Alberti, mannen som også lagde det første polyalfabetiske chifferet, Alberti chiffer (Wikipedia, 2022). Dette ble senere utvidet av Blaise de Vigenere til Vigenere chiffer. Med denne chiffer disken kan du rotere den indre ringen etter håndtaket fram til du har riktig skift, så kan du sammenligne indre og ytre ring for å enten oversette klartekst til chiffertekst (kryptering), eller chiffertekst til klartekst (dekryptering). Dette gjøres bokstav for bokstav i en melding og kan dermed ta ganske lang tid, i tillegg til at det kan forekomme feil ettersom det ikke er en automatisk prosess.

*Figur 2*
*(Berberich, 2013)*

Chiffer diskens rotasjon på 13 plasser er i likhet med ROT13 som er en type substitusjonschiffer der man velger å forskyve alfabetet med 13 plasser i motsetning til Cæsar-chiffers tradisjonelle 3 plasser. Siden det er 26 bokstaver i det latinske alfabetet, vil du ved å velge en rotasjon på 13 bokstaver ha egenskapen av å kunne bruke samme algoritme både for kryptering og dekryptering. Det vil si at hvis

du for eksempel krypterer «A» og får «N» vil du kunne bruke samme algortime for å dekryptere «N» tilbake til «A». Dette er fordi når du skifter en bokstav 13 plasser 2 ganger ender man opp med den samme bokstaven igjen. ROT13 algoritmen for både kryptering og dekryptering kan fremstilles med følgende formel:

$$(x + 13)\ mod\ 26$$

Her er x indeksen til bokstaven i alfabetet (A = 1, B = 2, C = 3, osv.) som vi plusser med 13 for å skifte den 13 til høyre. Til slutt tar vi modulo 26 av dette for å rulle tilbake til starten av alfabetet dersom addisjonen tar oss utenfor alfabetet (x + 13 > 26). Vi kan på en måte se på addisjonen som krypteringen og modulo operasjonen som dekrypteringen. Med det sagt var ROT13 chifferet utviklet av programmerere på 1980-tallet så det er antageligvis bare en tilfeldighet at chiffer disken fra 1400-tallet er innstilt til en rotasjon på 13 plasser ettersom brukere av ROT13 antageligvis ikke hadde bruk for slik primitiv teknologi.

## 2.2 Transposisjonschiffer

Transposisjonschiffere er konseptuelt veldig simpel i likhet med substitusjonschiffer. Disse lages ved å endre på rekkefølgen av karakterer i en melding, noe som også kan gjøres på forskjellige måter. For eksempel kan du manuelt bytte bokstavene i tilfeldig rekkefølge sånn at chifferteksten «atj gikel rme» fremstille meldingen «jeg liker mat». En mer strukturell type transposisjonschiffer er *kolonnemessig transposisjon,* en operasjon brukt i ADFGVX-chiffer, mer om dette om litt. Men siden hverken transposisjons- eller substitusjonschiffrene er særlig robuste algoritmer alene, skal vi nå, gjennom ADFGVX-chiffer, se hvordan du kombinere flere simple operasjoner til å lage en mer avansert algoritme.

ADFGVX-chiffer er et fraksjonerende transposisjonschiffer utviklet av Fritz Nebel for bruk i den tykse hæren i første verdenskrig (Wikipedia, 2022). Algoritmen utvider på ideen om transposisjon med å introdusere fraksjonering og ved å kombinere transposisjon med substitusjon. Dette går ut på at man legger ut alfabetet i et rutenett, et såkalt polybius kvadrat. I ADFGX, en tidligere implementasjon av algoritmen, var størrelsen på rutenettet 5x5. Dette ga oss totalt 25 mulige krypterbare tegn, der man kunne slå sammen 2 bokstaver, i og j i original implementasjon, i en rute for å dekke hele det latinske alfabetet. Men i denne algoritmen inneholdte chifferteksten alltid alle tall ukryptert, noe som var en åpenbar svakhet. Derfor ble V'en i ADFGVX lagt til, noe som gir oss et 6x6 rutenett. Dette gir oss i stedet 36 mulige krypterbare tegn, nok til hele alfabetet (26) og alle 10 siffer. Dette rutenettet skal vi så fylle opp med hele alfabetet og sifferene våre, men rekkefølgen på dette alfabetet skal holdes hemmelig. Det er i lagingen av dette hemmelige blandede alfabetet vi gjør substitusjonsdelen av algoritmen. Så la oss lage dette polybius kvadratet.

| | **A** | **D** | **F** | **G** | **V** | **X** |
|---|---|---|---|---|---|---|
| **A** | a | 3 | p | x | 9 | m |
| **D** | 7 | e | w | g | d | 5 |
| **F** | f | 6 | u | 4 | v | l |
| **G** | 8 | i | h | c | 2 | y |
| **V** | o | t | 0 | j | b | n |
| **X** | q | r | s | k | 1 | z |

Nå som vi har dette kvadratet kan vi kryptere meldingen vi skal sende. Da vil for eksempel meldingen «jeg liker mat» oversettes til koordinatene til hver bokstav i tabellen vår. Her vil den første bokstaven være raden og den andre bokstaven være kolonnen i polybius kvadratet vårt. Da ender vi opp med chifferteksten «VG DD DG FX GD XG DD XD AX AA VD»

| J | E | G | L | I | K | E | R | M | A | T |
|---|---|---|---|---|---|---|---|---|---|---|
| VG | DD | DG | FX | GD | XG | DD | XD | AX | AA | VD |

Så skal vi innføre en såkalt *kolonnemessig transposisjon.* Det vil si at vi skriver denne chiffersekvensen inn i et nytt rutenett med et nøkkelord eller frase som overskrift, la oss bruke «hallo». Overskriften har en lengde på 5 bokstaver så rutenettet får en lengde på 5 som vi fyller ut med koordinatene i våres originale polybius kvadrat, vi wrapper over til neste rad for hver femte koordinat. Vi skriver opp dette nye rutenettet**.**

| **H** | **A** | **L** | **L** | **O** |
|---|---|---|---|---|
| V | G | D | D | D |
| G | F | X | G | D |
| X | G | D | D | X |
| D | A | X | A | A |
| V | D | | | |

Neste steg er å alfabetisk sortere overskriften og dens tilhørende kolonne. Vi holder denne operasjonen stabil. Det vil si at dersom det er to kolonner med lik bokstav i overskriften beholder de deres forholdsvise rekkefølge før denne operasjonen. Altså i vårt tilfelle forblir L kolonnen med tilhørende DG og XD verdiene før L kolonnen med FX og AX verdiene. I denne prosessen er det også vanlig å fylle inn tomme verdier på nederste rad med tilfeldige verdier, sånn at man ikke skal kunne se hvilken rekkefølge kolonnene har blitt transposisjonert til. La oss legge inn dette også.

| **A** | **H** | **L** | **L** | **O** |
|---|---|---|---|---|
| G | V | D | D | D |
| F | G | X | G | D |
| G | X | D | D | X |
| A | D | X | A | A |
| D | V | F | X | D |

Vi er nå ferdig med den kolonnemessige transposisjonen og har det vi trenger for å sende meldingen vår kryptert. For å sende meldingen vår trekker vi verdiene ut av rutenettet og skriver det på en linje. Da ender vi til slutt opp med chiffersekvensen «GFGAD VGXDV DXDXF DGDAX DDXAD».

For å dekryptere denne meldingen kan du bruke nøkkelordet (hallo) for å ganske enkelt rekonstruere rutenettet. For å gjøre dette legger du chiffersekvensen inn i et rutenett der hver kolonne separeres av mellomrommene i chiffersekvensen. Så kan vi sortere nøkkelordet vårt og legge det til som overskrift. Da ender vi opp med det siste rutenettet tegnet ovenfor. Derfra kan vi ta kolonnene tilbake til sin egentlige plass, det vil si der de lå før den kolonnemessige transposisjonen. Da får vi det første rutenettet tegnet ovenfor. Til slutt kan vi lese av koordinatene fra det hemmelige polybius kvadratet for å endelig dekryptere meldingen.

Hver for seg er transposisjons- og substitusjonschiffer ganske enkelt å dekryptere, men som vi ser gjennom ADFGVX-chiffer kan du få noe ganske komplisert hvis du kombinerer disse metodene. Vi

kan nå se at vi begynner å nærme oss kryptografiske løsninger som er svært vanskelige å dekryptere for utenforstående med mindre de har tilgang til de(n) hemmelige nøkkelen(e). I motsetning til klassisk substitusjon- og transposisjonschiffer, der hvem som helst kan dekryptere chifferteksten så lenge de vet hvordan den er kryptert, ser vi at det nå begynner å danne seg en ny flaskehals rundt hemmelige nøkler, den nye store tingen innen kryptografi, i det vi nærmer oss moderne kryptografi.

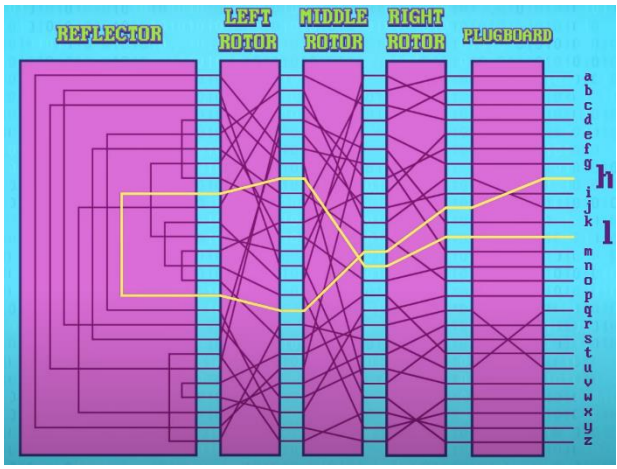
## 2.3 Enigma

Antageligvis den mest velkjente krypteringsalgoritmen som finnes, var enigma et gigantisk steg fremover i historien av kryptografi. Den viste oss hvor avansert krypteringsalgoritmer kunne være og hvor vanskelig de kunne bli å knekke. I tillegg, gjennom Alan Turing og hans medarbeideres arbeid i å knekke enigma, bidro den til utviklingen av datamaskinen og alt som kom med den, antageligvis en av de viktigste utviklingene i historien. I dette delkapittelet skal vi undersøke hvordan enigma maskinen virket, hvordan den til slutt ble knekket, og til slutt hvilke bidrag denne hadde for fremtiden.

*Figur 3*
*(Nassiri, 2012)*

Utviklingen av enigma maskinen begynner på 1910-tallet av Arthur Scherbius, men den vil ikke fremstilles offentlig før 1923 og ville undergå flere endringer. Til tross for dens notoriske bruk av Nazi Tyskland inkluderte produktets originale markedsmålgruppe både ikke-militære og utenlandske organisasjoner, som vi kan se på variantene av enigma deriblant «Swedisch» og «Swiss K» (Crypto Museum, 2022) .

Ettersom enigma er den første krypteringsalgoritmen som avhenger av fysiske komponenter for kryptering trenger den noe teknologi for å virke. Derfor kan vi si at historien til maskinen virkelig begynner ved oppfinnelsen av rotoren. Kreditert til de to nederlandske marine offiserene Hengel og Spengler (Kowalczyk, 2020), var oppfinnelsen av rotoren kjernen til enigma. Ideen bak rotor maskinen er å utføre et substitusjonschiffer. En rotor består av 26 mulige input og 26 mulige output, en for hver bokstav der inputet hentes fra skrivemaskinen. Rotoren tar dette inputet og sender det videre som en annen bokstav, dette er substitusjonschifferet. I enigma maskinen gjøres dette 3 ganger før det går inn i *reflektoren.* Reflektoren tar input fra den siste rotoren og sender det tilbake inn den samme rotoren som en annen bokstav. Men for å utvide enigmaen til noe mer enn et enkelt substitusjonschiffer roterer rotorene med et steg hver gang en bokstav ble skrevet inn på skrivemaskinen. Dette gjør algoritmen vesentlig mer robust ettersom samme bokstav kan fremstå som ulike bokstaver i chifferteksten. Siden det var så mange kombinasjoner av innstillinger for rotorene (26^4 = 456 975, 26^4 fordi det var 3 rotorer og en reflektor), i tillegg til at disse kombinasjonene ble endret til en ny kombinasjon på starten av hver dag, var det veldig vanskelig å knekke enigma. Denne kombinasjonen av innstillinger som ble endret hver dag utgjorde den «hemmelige nøkkelen» for enigma.




*Figur 4*
*(Crash Course, 2017)*

For dekryptering av en melding sendt med enigma var det viktig at både mottakeren og avsenderen hadde riktige kombinasjon av innstillinger på rotorene når de skulle kryptere og dekryptere

meldingen. Dette var også det eneste man trengte for å dekryptere enigma ettersom så snart man hadde denne og en enigma maskin kunne man dekryptere alle meldinger sendt med samme innstillinger. Som du kan se på figur 4 går elektrisiteten som krypterer meldingen i en krets som kan gå begge veier. Dette betyr at hvis l krypterer til h krypterer h til l, altså har vi samme funksjon for kryptering og dekryptering i likhet med det vi så i ROT13 algoritmen. På den andre siden betyr dette at en bokstav aldri vil kunne krypteres til seg selv, en svakhet ettersom dette kan ses når man analyserer chifferteksten.

Som den første elektrisk drevne kryptografiske algoritmen ser vi gjennom enigma maskinen overgangen fra de primitive historiske krypteringsalgoritmene til et nytt paradigme innen kryptologi, nemlig den hemmelige nøkkelen, snart også utvidet til offentlige nøkler. Dette i kombinasjon med datamaskinen som snart vil i full sving ta inn historien av kryptografi for å kombinere alt vi har sett av krypteringsalgoritmer for å tilfredsstille nye krav som nå også vil utvide markedet for kryptografi utenfor det militære. Snart vil enhver person initiere kryptografiske prosesser hundrevis av ganger daglig uten at de en gang vet det. Det er disse algoritmene vi nå skal se på i det vi går inn i den moderne alderen av kryptografi.

# 3 Moderne kryptografi

## 3.1 Diffie-Hellman nøkkelutveksling

Diffie-Hellman nøkkelutveksling er et av de tidligste eksemplene på offentlig nøkkelutveksling. Den ble laget gjennom et felles samarbeid mellom Ralph Merkle mens navnet kommer fra Whitfield Diffie og Martin Hellman. Denne metoden for digital kryptering lar to parter som ikke har noen forkunnskaper om hverandre, lage en dekrypteringsnøkkel sammen. For å oppnå dette må imidlertid to partier, la oss kalle dem Alice og Bob, utveksle informasjon gjennom en usikker kommunikasjonskanal for å kunne utføre DH-nøkkelutvekslingen.

Begge partier må bli enige om et veldig stort primtall, $N$. De må også bli enige om et andretall, $G$ der $G$ er mindre enn $N$ og må være primitivt i forhold til $N$. I tallteori er et tall $G$ en primitiv rot modulo $N$ hvis hvert tall er et relativt primtall til $N$ er kongruent med potensen $G$. Et eksempel vil være tallet 3 er en primitiv rot modulo 7 siden:

$$
\begin{aligned}
3^1 &= 3^0 \times 3 \equiv 1 \times 3 = 3 \equiv 3 \\
3^2 &= 3^1 \times 3 \equiv 3 \times 3 = 9 \equiv 2 \\
3^3 &= 3^2 \times 3 \equiv 2 \times 3 = 6 \equiv 6 \\
3^4 &= 3^3 \times 3 \equiv 6 \times 3 = 18 \equiv 4 \\
3^5 &= 3^4 \times 3 \equiv 4 \times 3 = 12 \equiv 5 \\
3^6 &= 3^5 \times 3 \equiv 5 \times 3 = 15 \equiv 1 \\
3^7 &= 3^6 \times 3 \equiv 1 \times 3 = 3 \equiv 3
\end{aligned}
$$

(Wikipedia, 2022)

Alle verdier i den siste kolonnen er alle rester som ikke er null modulo 7, noe som betyr at 3 er en primitiv rot modulo 7. Dette kan observeres fra det faktum at sekvensen $g^k$ alltid gjentar modulo $n$ etter en verdi av $k$ , siden modulo $n$ produserer bare et begrenset antall verdier. Hvis $g$ er en primatrot modulo $n$, og $n$ er primtall, begynner repetisjonen ved $n - 1$. (Wikipedia, 2022)

Med introduksjonen av primitiv rot modulo ute av veien, velger partiene, Alice og Bob, $N$ og $G$ å utveksle disse verdiene til hverandre. De vil deretter følge denne algoritmen for å lykkes med å lage sin egen private og offentlige nøkkel. Vi skal se at denne algoritmens måte å velge private nøkler på er en stor forbedring fra de foreldede metodene til de sene førmoderne algoritmene som *ADFGVX* og *enigma*. Siden de «private nøklene» for disse algoritmene helst måtte deles ut fysisk utgjorde dette en stor trussel for sikkerheten til disse algoritmene i tillegg til at det la til fysisk arbeid, 2 ting vi nå slipper med DH nøkkelutveksling. Måten vi slipper dette på er med bruk av offentlige nøkler. La oss nå gå gjennom prosessen som muliggjør alt dette.

Alice

1. Alice skal velge et enormt tall, $a < N$. Dette skal lage Alice sin private nøkkel.

2. Alice skal da beregne $A = G^a \bmod N$. Dette skal lage Alice sin offentlige nøkkel.

3. Alice skal da utveksle sin offentlige nøkkel til Bob.

4. Og da skal beregne $K_A = B^a \bmod N$.

(Kessler, 2022)

Bob

1. Bob skal velge et enormt tall, $b < N$. Dette skal lage Bob sin private nøkkel.

2. Bob skal da beregne $B = G^b \bmod N$. Dette skal lage Bob sin offentlige nøkkel.

3. Bob skal da utveksle sin offentlige nøkkel til Alice.

4. Og da skal beregne $K_B = A^b \bmod N$.

(Kessler, 2022)

De offentlige nøklene, $A$ og $B$, er åpent til alle sammen mens den private nøkkelen, $a$ og $b$, ikke er offentlig bortsett fra henholdsvis til Alice og Bob. La oss ta et eksempel for å vise prosessen med algoritmen. Normalt er $N$ og $G$ veldig store tall, men i dette eksemplet skal vi ha $G$ til å være 3 og $N$ til å være 7 for å gjøre det enklere med våre forkunnskaper fra tabellen.

Alice

1. Alice skal velge,

$a = 4$. Dette skal lage Alice sin private nøkkel.

2. Alice skal beregne $A = 3^4 \bmod 7 = 4$. Dette skal lage Alice sin offentlige nøkkel.

3. Alice skal da utveksle sin offentlige nøkkel med Bob.

4. Og skal da beregne $K_A = 5^4 \bmod 7 = 2$.

Bob

1. Bob skal velge,

$b = 5$. Dette skal lage Bob sin private nøkkel.

2. Bob skal beregne $B = 3^5 \bmod 7 = 5$. Dette skal lage Bob sin offentlige nøkkel.

3. Bob skal da utveksle sin offentlige nøkkel til Alice.

4. Og da skal da beregne $K_B = 4^5 \bmod 7 = 2$.

De eneste nøklene som holdes hemmelige er de private nøklene. Alle de andre verdiene utveksles gjennom en usikker kommunikasjonskanal. Styrken til Diffie-Hellman nøkkelutveksling er avhengig av at $B^a \bmod N = A^b \bmod N$ tar ekstremt lang tid å beregne med dagens moderne maskiner. Til tross for at du har kunnskap om alle verdier bortsett fra de private nøklene, er det fortsatt ekstremt vanskelig å beregne selv ved hjelp av komplekse algoritmer.

Diffie-Hellman-nøkkelutvekslingen er veldig sikker mot avlytting hvis begge parter velger riktig verdi for $G$. Men Diffie-Hellman-utvekslingen kan fortsatt bli utsatt for et man-in-the-middle angrep som kunne vært unngått hvis Alice og Bob hadde autentikasjon. En angriper kan etablere to distinkte nøkkelutvekslinger, en med Alice og den andre med Bob, og etterligne begge partier som lar angriperen dekryptere og kryptere informasjonen som utveksles gjennom en usikker kommunikasjonskanal. Derfor er Diffie-Hellman implementert med en algoritme som RSA for å autentisere en eller begge partiene som anses som den sikreste implementasjonen blant mange varianter av Diffie-Hellman nøkkelutvekslingen. I neste delkapittel skal vi se hvordan RSA blir implimentert for å gjøre Diffie-Hellman sikker.

## 3.2 RSA

Rivest–Shamir–Adleman, eller ofte kjent som RSA, implementeres ofte sammen med Diffie-Hellman nøkkelutveksling for å gi autentisering mellom Alice og Bob og for å forhindre et man-in-the-middle angrep. I motsetning til Diffie-Hellman, kan RSA ikke bare brukes til å utveksle digitale signaturer, men kan også kryptere informasjon. Kryptering av meldinger med RSA brukes imidlertid ikke ofte da det er for ineffektiv og brukes først og fremst til å kryptere øktnøkkelen for meldingsintegritet eller digital signatur. Styrken til RSA er identisk med Diffie-Hellman i den forstand at matematiske operasjoner er raske å beregne, men svært vanskelige å reversere. Begge protokollene muliggjør kryptering av informasjon, men det er vanskelig å reversere informasjonen.

Implementeringen av RSA er sterkt avhengig av modulær aritmetikk, Eulers teorem og totientfunksjon. For å opprette en RSA offentlig eller privat nøkkel, må denne algoritmen følges.

> 1. Velge to prim tall, $p$ and $q$, og beregne $n = pq$. Dette vil være den første halvdelen av offentlige nøkkelen.
> 2. Beregne deretter $\phi(pq) = (p-1)(q-1)$, og velg et tall $e$ som ikke deler noen andre faktor enn 1 med $\phi(pq)$. Dette vil være den andre halvdelen av den offentlige nøkkelen.
> 3. Beregne deretter $d$, den modulære multiplikative inverse av $e$ modulo $\phi(n)$, $de \equiv 1 \bmod \phi(n)$. $d$ vil være den private nøkkelen.
>
> (Brilliant.org, 2022)

Den offentlige nøkkelen er tallparet av $n$ og $e$. Til tross for at begge tallene utveksles på en usikker kommunikasjonskanal, vil det fortsatt ta ekstremt lang tid å bestemme $n$ med moderne datamaskiner selv ved hjelp av komplekse algoritmer.

Med kunnskapen vi har om å lage offentlige og private nøkler med RSA, skal vi nå gå gjennom hvordan RSA krypterer. For å kryptere en melding $M$ med den offentlige nøkkelen for å lage chifferteksten $C$ må denne algoritmen følges.

1. Avsenderen må beregne $C = M^e \bmod n$ for å kryptere meldingen sin.
2. Mottakeren må deretter dekryptere meldingen med den private nøkkelen ved å beregne $M = C^d \bmod n$.

(Brilliant.org, 2022)

Som med Diffie-Hellman tidligere, skal vi demonstrere et eksempel med små tall. Tallene $p$ og $q$ er generelt svært store tall for å ytterligere styrke sikkerheten til protokollen på grunn av at dens styrke er avhengig av datamaskiner tar kort tid å beregne $n$ i forhold til å finne de to primtallene som er faktor til $n$ som kan ta år eller noen ganger så lenge som levetiden til vår egen planet.

1. Vi skal velge $p = 3$ og $q = 5$ , deretter beregne $n = pq = 4 \cdot 7 = 28$.
2. Vi skal da velge et tall, $e$, som er relativt primisk til $\phi(pq) = (p-1)(q-1) = (4-1)(7-1) = (3)(6) = 18$. Vi skal velge $e = 5$.
3. Vi skal da beregne $d = 11$ gjennom den modulære multiplikative inverse av $e$ modulo $\phi(n)$.

Vi har nå vår offentlige nøkkel $(28, 5)$ og vår private nøkkel $11$. Hvis vi skulle utveksle en melding, $M$, bruker vi en vanlig konverteringsprosess med ASCII-alfabetet. Men hvis vi skulle lage en melding som er større enn $n$, ville meldingen bli delt opp i biter. La oss ta ordet **MATTE** som et eksempel, siden alle alfabeter har sin egen unike verdi, må vi dele verdiene og bytte dem individuelt. Siden alle alfabeter er større enn n må vi først dele M til 7 og 7, A til 6 og 5 og så videre.

| **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
| **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** | **Y** | **Z** |
| 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |

1. Avsender må først beregne $C = M^e \bmod n = 7^5 \bmod 28 = 7$ for å kryptere meldingen sin.
2. Mottakeren må deretter dekryptere meldingen med den private nøkkelen ved å beregne $M = C^d \bmod n = 7^{11} \bmod 28 = 7$.
3. Og gjenta deretter fra trinn 1 for å motta alle ASCII-verdiene som sendes for å få den endelige meldingen 7765848469 som oversettes fra tabellen til **MATTE.**

Med RSA nå implementert, er autentisering nå mulig. Begge partier, Alice og Bob, vil signere en melding ved å bruke sin private RSA-nøkkel og utveksle den til hverandre. Med dette kan de bekrefte identiteten til mottakeren ved å sjekke de signerte meldingene med deres offentlige nøkkel og sammenligne med den andres digitale sertifikat. Med begge partier autentisert er det teknisk mulig å fortsette å kryptere meldinger og trygt utveksle dem på en usikret kommunikasjonskanal med RSA, men det vil være for ineffektiv. For å unngå denne ineffektiviteten brukes RSA ofte sammen med andre sikkerhetsprotokoller som Diffie-Hellman nøkkelutveksling for å etablere en delt symmetrisk nøkkel. Dette brukes deretter i en symmetrisk nøkkelalgoritme for å kryptere informasjonen som Alice og Bob alltid hadde hensikt om å utveksle sikkert seg imellom.

På grunn av robustheten til RSA, brukes den overalt hvor kryptering er nødvendig. Det brukes også for å sikre at nettsteder er legitime for å filtrere bort bedrageren siden bare det virkelige nettstedet har den private nøkkelen. Det eliminerer derfor effektivt et man-in-the-middle angrep der informasjonen din ville bli fanget opp og etterlignet en eller begge partier. Mens RSA anses som sikker, er styrken avhengig av nøkkelstørrelsen, som er antall bits $n = pq$. Med moderne algoritmer og datamaskiner kan 512-bits RSA lett bli brutt forcet der den private nøkkelen trekkes ut på bare et par timer. I dag skapes det 4090-bit RSA ettersom det er standarden, da hvor som helst lavere anses som risikabelt. Hvis en sårbarhet skulle bli funnet i RSA ville det ha en katastrofal konsekvens ikke bare i kryptografi, men overalt på grunn av at kryptering er så viktig i bank, netthandel og så mange andre. En mann ved navn Peter Shor utviklet en algoritme for kvantedatamaskiner som er i stand til å faktorisere tall raskere enn noen maskin vi har i dag. Kvantedatamaskiner har imidlertid fortsatt en lang vei å gå.

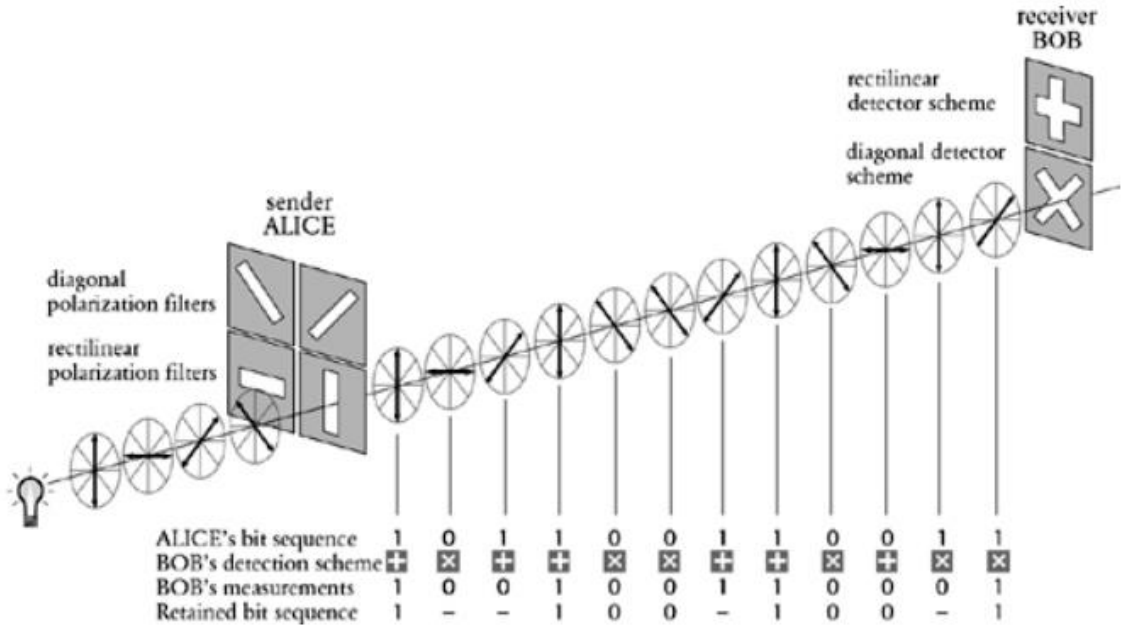
## 3.3 Kvantekryptering

Utviklingen av datamaskiner igjennom årene har ført til at datamaskiner blir raskere og smartere hele tiden. Dette gjør det mye enklere å dekryptere meldinger, noe som fører oss til trusselen potensielle kvantedatamaskiner kan utgjøre. I motsetning til dagens moderne datamaskiner som bruker enere og nuller, kalt bits, er kvantedatamaskiner basert på kvantemekanikk og bruker kvantebits, kalt qubits. Vanlige bits kan ha to forskjellige tilstander, 1 og 0 som representerer av og på, imens qubits derimot har i tillegg en tredje tilstand som heter *superposisjon*. Superposisjon refererer til et fenomen innenfor kvantemekanikk hvor et kvantesystem kan være i flere tilstander eller steder samtidig, som innenfor kvantedatamaskiner baserer seg på partiklene i superposisjon (Amyx, 2021). Disse partiklene representerer qubits, som kan holde verdien 0, 1 eller begge samtidig. Det som gjør kvantedatamaskiner så utrolig mektig er at vi kan benytte qubits i superposisjon for å representere et eksponentielt antall tilstander, som ved hjelp av kvantealgoritmer vil kunne beregne mange mulige svar for ett gitt input. Dermed vil kvantedatamaskiner klare å knekke koder på noen minutter i motsetning til i dag hvor det kan ta hundrevis og kanskje opptil millioner av år å knekke. Nå som vi forstår at kvantedatamaskiner kan utgjøre en stor trussel for informasjonssikkerhet, mer spesifikt kryptografi, over hvor mektige de kan bli, skal vi se på ulike måter vi kan muligens forsvare oss mot denne forestående trusselen.

Den mest utbredte og kjente metoden innenfor kvantekryptografi er det vi kaller *kvantenøkkelutveksling*, på engelsk heter det *Quantum Key Distribution*, og blir ofte forkortet til *QKD*. I stedet for å bruke en tilfeldig streng av bits som vår nøkkel som i tradisjonell kryptografi, er nøkkelen innenfor kvantekryptografi en sekvens av fotoner, eller lyspartikler. Disse fotonene vibrerer og har en viss rotasjon som er polariseringen til fotonet. En lyskilde, som for eksempel en lyspære eller en lysdiode, vil kunne produsere fotoner av alle slags polariseringer. For å gjøre ting enklere å oversette, så blir det antatt at fotoner kun har fire polariseringer, nemlig horisontalt —, vertikalt |, diagonalt venstre \ og diagonalt høyre / (Singh, 1999, s. 332). Innenfor kvantemekanikk så har vi et prinsipp som heter *Heisenberg Uncertainty Principle* som i bunn og grunn forteller oss at det er umulig å kunne måle, og dermed vite, en partikkel sin posisjon og fart samtidig (Caltech, 2022). Det er en del detaljer og iboende egenskaper ved kvantemekanikk som fører oss til dette prinsippet, men poenget er at desto vi forsøker å måle posisjonen til en partikkel med nøyaktighet, vil vi ikke klare å måle farten også, og vice versa. Det å måle partiklene i en kvantetilstand vil altså påvirke og endre tilstanden selv. På den lyse siden så kan vi benytte dette prinsippet om usikkerhet til vår fordel angående kvantekryptering. Når en sekvens av fotoner blir dannet av en lysdiode, kan vi endre polariseringen til fotonene ved hjelp av de polariserte filtrene som ble nevnt tidligere. Dersom et

foton er polarisert, så kan ikke fotonet bli målt på en nøyaktig måte utenom et polarisert filter (Clark, 2007). For eksempel hvis et foton med en diagonal polarisering blir målt med et rettlinjet filter, vil fotonet enten endre polarisering til horisontalt eller bli absorbert. I begge tilfellene vil hvilken som helst informasjon som var knyttet til fotonet sin polarisering, eller rotasjon, nå være tapt.

Hvordan kan to parter dele privat informasjon, som en melding eller en hemmelig nøkkel, sammen ved hjelp av kvantenøkkelutveksling? Det er her QKD kommer inn i praksis. Først og fremst må fotonene representere et stykke informasjon, ved hjelp av binærkode, for så å kunne oversette informasjonen til en nøkkel. La oss si fotoner med en vertikal polarisering | og fotoner med en diagonalt høyre polarisering / bety 1, og fotoner med en horisontal polarisering — og fotoner med en diagonalt venstre polarisering \ bety 0. La oss si vi har to parter, Alice og Bob, hvor Alice skal sende privat informasjon til Bob.




Bilde: (Singh, 1999, s.342)

I dette eksemplet vist på bilde vil Alice begynne å sende fotoner, som representerer enere og nuller, til Bob med et fullstendig tilfeldig utvalg av de fire polariserte filtrene i henhold til hvilke ener og null verdier polariseringene representerer. Bob har detektorer, en rettlinjet detektor + og en diagonal detektor X, for å kunne måle hvert foton. Han vil velge detektor tilfeldig for hvert foton som gir oss en 50% sjanse for at han får riktig hver gang. Den avgjørende egenskapen med hele denne sekvensen er at den må være tilfeldig, dette er fordi den er utledet fra Alice sin sekvens, som i seg selv er tilfeldig (Singh, 1999, s.343-344). Som sagt så er tilfellene hvor Bob bruker riktig filter også tilfeldig, som betyr at sekvensen som blir sendt vil derfor ikke utgjøre en melding, men den kan utgjøre og bli brukt som en tilfeldig nøkkel.

Hva skjer hvis det er en *man in the middle*, kalt Eve, som lytter etter sekvensen imellom Alice og Bob? Imens Alice sender tilfeldig polariserte fotoner til Bob, vil Eve, til å begynne med, være i samme båt som Bob. Det er her *Heisenberg Uncertainty Principle* som ble nevnt tidligere blir viktig. Siden prinsippet forteller oss at hvis man forsøker å måle partiklene i en kvantetilstand, vil dette altså påvirke selve partikkelen slik at tilstedeværelsen av en avlytter vil bli oppdaget og kjent til Alice og

Bob. Den eneste måten Eve kan måle fotonene sin polarisering på er å bruke et filter, i dette tilfelle en rettlinjet detektor + og en diagonal detektor X. For eksempel hvis Eve skal forsøke å måle et foton med en vertikal polarisering | ved å bruke en diagonal detektor X, vil Eve gjette feil og dermed endre polariseringen til fotonet som gjør at Eve sin tilstedeværelse er kjent. Basert på at Alice sender fotonene tilfeldig og alt Eve kan gjøre er å gjette hva som er riktig, vil hun gjennomsnittlig gjette feil halvparten av gangene. Så langt har vi et stort problem, Bob vet heller ikke hvilke filtre han skal bruke. Måten man fikser det på er at etter overføringen av fotonene, nemlig nøkkelen, vil Alice og Bob ha en ikke-kryptert samtale. Alice forteller ikke Bob hvilken polarisering, eller rotasjon, fotonene hadde, eller om det var en ener eller null, hun vil kun fortelle han hvilke som var rettlinjet og hvilke som var diagonalt (Clark, 2007). Hvis Bob brukte riktig filter, beholder begge to sifferet for det fotonet, ellers hvis Bob brukte feil filter, så blir begge to kvitt sifferet for det fotonet. Basert på sannsynlighet, vil Bob ende opp med å velge omtrent halvparten av filtrene riktig og halvparten feil. Dermed beholder vi den halvparten hvor han valgte riktig filter og denne sekvensen blir da nøkkelen. Selv om Eve lytter på denne ikke-krypterte samtalen, vil hun ikke klare å gjette om det er en ener eller null kun basert på hvilken detektor, rettlinjet + eller diagonal X, og dermed vil den eneste måten at hun får tak i nøkkelen være hvis hun gjetter alle filtrene til Alice riktig, noe som er ekstremt usannsynlig og praktisk talt umulig.

Kvantekryptografi baserer seg på den tilsynelatende tilfeldigheten til kvantemekanikk. Denne tilfeldigheten kan være ekte 100% tilfeldighet, men hva om det ikke er det? Hva skjer med kryptografien i det tilfellet? Kan dette være en potensiell svakhet for kvantekryptografi i fremtiden, noe vi burde vurdere på forhånd slik at vi er forberedt? Vanskelig å si, men ideen bak kvantekryptografien sin tilfeldige natur kan virke litt urimelig ved første øyekast. Dette er fordi fysiske objekter, innenfor vår kunnskap, må følge fysikkens lover, ellers gir det ikke mening og ting vil bare skje uten en grunn eller årsak. Dersom vi zoomer inn til kvanteverden, virker det som om prosessene er tilfeldige, men hvis ting er tilfeldige på mikro nivå, vil ikke de samme tingene være tilfeldige på makro nivå når vi zoomer ut igjen? Disse større objektene, som mennesker, biler, planeter, osv. er jo tross alt bygd opp av disse mindre tingene i kvanteverden. Kan hende kvanteverden ikke er like tilfeldig som vi tror, at vi bare ikke klarer å måle mønstre på en såpass liten skala at det gir mening og dermed virker det uforutsigbart, altså tilfeldig.

En grunn for at kvanteverden virker så kaotisk at det blir tilfeldig, kan skyldes at vår teknologi ikke er bra nok til å kunne nøyaktig måle og utforske kvanteverden riktig ennå. Dermed ser det ut til at det kan være litt for tidlig for oss å konkludere om kvanteverden faktisk er tilfeldig, pseudo-tilfeldig eller deterministisk. Dette ble litt filosofisk, men poenget med dette er at dersom vi konkluderer basert på uvitenhet om tilfeldigheten bak en slik kryptografisk algoritme, kan dette føre til en svakhet innenfor kvantekryptografi. Eller på en annen side kan det hende at det fremdeles forblir så utrolig kompleks uansett hvilken forutsigbarhet algoritmen har at en kvantedatamaskin fortsatt ikke kan knekke det. I så fall vil det ikke være en svakhet. En lignende problemstilling blir da at hvis algoritmen for krypteringen er tilfeldig, hvordan vil man da kunne dekryptere det med en nøkkel gitt at chifferteksten er fullstendig tilfeldig og har dermed ikke noe mønster som kan spores tilbake til den originale meldingen? Hvis chifferteksten faktisk ikke er fullstendig tilfeldig og har da et mønster, så må vi kunne anta at det er mulig å knekke denne antatte "uknekkelige" algoritmen med en mektig nok kvantedatamaskin.

# 4 Konklusjon

Alt i alt har vi nå kommet oss igjennom en svært varig og sofistikert historie av kryptografien helt fra eldgamle Hellas til dagens informasjonsalder hvor ting er i konstant endring og utvikling. Blant de tidligste formene for kryptografiske algoritmer var substitusjonschiffere og transposisjonschiffere, som gikk ut på å manipulere klarteksten selv for å skjule informasjon. Problemet med disse algoritmene var at hvis man visste algoritmen, så var det ikke altfor vanskelig å knekke koden. I løpet av en lang periode kommer vi til Enigma maskinen på starten av 1900-tallet. Denne maskinen var utrolig mye mer avansert enn tidligere algoritmer, og den ble blant annet brukt under andre verdenskrig av tyskerne for å sende krypterte meldinger til hverandre. Enigma ble senere knekket den også. Etter andre verdenskrig begynte vi å bevege oss inn i informasjonsalderen når de moderne datamaskinene begynte å finne sitt sted på bordene hjemme hos vanlige sivile mennesker. Nå begynte kravet om enda bedre kryptografiske algoritmer å stige drastisk basert på hvor mange folk som brukte disse maskinene og hvor raske disse maskinene faktisk var sammenlignet med tidligere. På dette tidspunktet har vi ufattelig sofistikerte algoritmer, som Diffie-Hellman nøkkelutveksling og RSA, som kom med konseptet av å benytte seg av offentlige og private nøkler for å øke sikkerheten. Disse algoritmene, blant mange andre, baserer seg på matematikk og store tall for å gjøre det ekstremt vanskelig å dekryptere. Videre har vi det som er potensielt den neste store tingen for hele verden, nemlig kvantedatamaskiner. Disse fremtidige maskinene vil være såpass raske og mektige at de vil kunne knekke algoritmene vi har i dag med letthet. Kvantekryptografi baserer seg på fysikk, nemlig kvantemekanikk, i stedet for matematikk for å gjøre beregninger. Måten disse algoritmene baserer seg på fysikken, vil føre til en ekstrem økning av ytelse og hastighet. Som vi nå har sett er kryptografi en av de viktigste elementene innenfor informasjonssikkerhet, det finnes tross alt overalt i det moderne samfunnet vi lever i nå. Dette betyr at hvis noen utnytter forskjellige feil i et system, eller at en myndighet, land, hva enn, har utviklet moderne teknologi som de holder skjult for resten av verden som klarer å knekke dagens algoritmer, kan dette ikke bare bety en trussel til dine personlige data, men hele verden sin globale sikkerhet vil være i faresonen.

# Referanser

Berberich, H. (2013). *CipherDisk2000.* Wikimedia Commons. Hentet Oktober 26, 2022

Brilliant.org. (2022). *RSA Encryption*. Hentet 10 31, 2022 fra https://brilliant.org/wiki/rsa-encryption/

Cepheus. (2006). *Caesar3.* Wikimedia Commons. Hentet Oktober 26, 2022

Crash Course. (2017, Oktober 26). *Cryptography: Crash Course Computer Science #33*. Hentet Oktober 28, 2022 fra https://www.youtube.com/watch?v=jhXCTbFnK8o

Crypto Museum. (2022, April 16). *History of the Enigma*. Hentet Oktober 28, 2022 fra https://www.cryptomuseum.com/crypto/enigma/hist.htm

Jøsang, A. (2021). *Informasjonssikkerhet - teori og praksis.* Universitetsforlaget.

Kessler, G. C. (2022, Oktober 01). *An Overview of Cryptography*. Hentet Oktober 23, 2022 fra https://www.garykessler.net/library/crypto.html#dhmath

Kowalczyk, C. (2020, Mars 9). *Cryptographic Rotor Machines*. Hentet Oktober 28, 2022 fra Crypto-IT: http://www.crypto-it.net/eng/simple/rotor-machines.html

Nassiri, A. (2012). *Enigma (crittografia) - Museo scienza e tecnologia Milano.* Wikimedia Commons.

RomeAndArt. (2022). *RomeAndArt*. Hentet Oktober 26, 2022 fra https://www.romeandart.eu/en/art-cipher-jfulius-caesar.html

Wikipedia. (2022). *ADFGVX cipher*. Hentet Oktober 27, 2022 fra https://en.wikipedia.org/wiki/ADFGVX_cipher

Wikipedia. (2022). *Cipher disk*. Hentet Oktober 27, 2022 fra https://en.wikipedia.org/wiki/Cipher_disk

Wikipedia. (2022). *Diffie-Hellman Key Exchange*. Hentet Oktober 23, 2022 fra https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

Wikipedia. (2022). *Primitive root modulo n*. Hentet Oktober 22, 2022 fra https://en.wikipedia.org/wiki/Primitive_root_modulo_n

Amyx, S. (2021, December 26). *Quantum computing series, Part 4: Superposition in Quantum Mechanics*. IoT Practitioner. https://iotpractitioner.com/quantum-computing-series-part-4-superposition-in-quantum-mechanics/

Singh. (1999). *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor Books.

Caltech. (2022). *What is the uncertainty principle and why is it important?* Caltech Science Exchange. https://scienceexchange.caltech.edu/topics/quantum-science-explained/uncertainty-principle

Clark. (2007, 23. Oktober). *How quantum cryptology works*. HowStuffWorks. https://science.howstuffworks.com/science-vs-myth/everyday-myths/quantum-cryptology.htm