FFI Forsvarets forskningsinstitutt

24/01624

FFI-RAPPORT

Sikre beregninger fra kryptografi

– hvordan samarbeide om hemmelige data uten å røpe hemmeligheter

Martin Strand
Jan Henrik Wiik
Frode Lillevold

Sikre beregninger fra kryptografi

– hvordan samarbeide om hemmelige data uten å røpe hemmeligheter

Martin Strand
Jan Henrik Wiik
Frode Lillevold

Forsvarets forskningsinstitutt (FFI) 23. oktober 2024

Emneord
Kryptografi
Kryptologi
Beregninger
Datasikkerhet
Outsourcing

FFI-rapport
24/01624

Prosjektnummer
1614

Engelsk tittel
Secure computations from cryptography

Elektronisk ISBN
978-82-464-3553-4

Godkjennere
Raymond Haakseth, *forskningsleder*

Dokumentet er elektronisk godkjent og har derfor ikke håndskreven signatur.

Sammendrag

Ved hjelp av moderne kryptografi kan vi utføre sikre beregninger, der ingen får innsyn i andre data enn de skal. Det gjør oss i stand til å redusere risikoen som følger av å dele data med andre organisasjoner eller personer, uavhengig av hvor gode de er til å holde informasjonen hemmelig.

Vi har studert tre metoder:

Sikre flerpartsberegninger (secure multiparty computations – MPC) Når flere parter har fortrolige data, kan man bruke MPC for å regne på disse dataene i fellesskap og gi resultatene til partene som skal ha dem. Ingen av partene vil se mer enn deres egne data og sluttresultatet. MPC er et forholdsvis modent område, og selv om det i liten grad finnes pakkeløsninger, kan det brukes i dag.

Homomorf kryptering (fully homomorphic encryption – FHE) Dersom én part har sensitive data, og en annen, ikke-tiltrodd part har beregningsressursene, kan den første kryptere dataene med et FHE-system og sende dem til den andre. Den andre kan da regne på dataene og sende dem tilbake, men uten å ha sett noe i klartekst. FHE er også nyttig dersom parten med beregningsressursene ønsker å holde algoritmene for seg selv. Forskerne forstår FHE godt, men på grunn av svært store chiffertekster er det mindre anvendbart i praksis.

Skreddersydd dekryptering (functional encryption – FE) Ideen forklares enklest med et eksempel: Eieren av en stor database kan kryptere hele databasen og distribuere den til flere interessenter med ulike behov og tillitsnivå. Disse kan få utdelt nøkler som tillater dem å bruke databasen på forskjellige måter. Noen kan for eksempel dekryptere alle dataene, noen kan bare dekryptere felter som samsvarer med deres tilgangsnivå, mens andre igjen kan beregne statistikk fra databasen, men ikke dekryptere noen enkeltdatapunkter.
FE er fortsatt umodent, også i grunnforskningen.

Disse metodene har noen iboende begrensninger som påvirker mulige anvendelser, men disse begrensningene finnes i stor grad også i ikke- kryptografiske løsninger som benytter seg av tiltrodde tredjeparter.

Vi har tidligere utarbeidet en katalog over mulige anvendelser av teknikkene over innen nasjonal sikkerhet. Nå har vi også gått dypere inn i flere av anvendelsene og undersøkt i hvilken grad teknikkene over kan brukes til disse. Vi ser tilfeller der disse fungerer godt, og tilfeller der de ikke fungerer så godt. Det ikke bare er et spørsmål om teknologi, men også om jus, organisasjon og ikke minst intensjonen til de som laget reglene uten å kjenne til mulighetene som kryptografien kan tilby.

Summary

Modern cryptography can be used to do secure computations. It enables us to reduce the risk involved in sharing data with other organisations or persons, regardless of their ability to keep the information secret.

We have studied three methods:

Secure Multiparty Computations (MPC) If multiple parties have sensitive data, MPC can be used to collectively do computations on the data, and to release the results to the intended parties. No party can access more than their own data and the end result. MPC is reasonably mature, and while it is not yet available off the shelf, it has already seen several real-life applications.

Fully Homomorphic Encryption (FHE) If one party has sensitive data and another, untrusted party has the resources to do computations on the data, the former can encrypt the data using FHE before sending the data to the latter. The other party can then compute and return the encrypted result without having access to any cleartext. FHE is particularly useful if the party doing the computations wants to keep their algorithms to themselves. FHE is well understood in the research community, but it is less applicable in practice due to very large ciphertexts.

Functional Encryption (FE) The core idea is best explained through an example: The owner of a large database can encrypt the whole database and distribute it to numerous parties with different needs and trusts. These can be given keys that allow them to calculate different properties of the database. Some might be allowed to decrypt all the data, some might only decrypt some of the data corresponding to their specific access privileges, while others yet might only be given access to certain statistical aggregate values of the data in the database without having access to any single data point.
FE is still primarily a topic for basic research.

These methods have some inherent limitations that could affect possible applications, but these limitations are in most cases also present in non-cryptographic solutions based on a trusted third-party.

In previous work, we have made a catalogue of possible applications of the techniques above in the area of national security. In this work we have expanded our exporation of a selection of the applications, and tried to examine whether the methods above are applicable. The results unveil both promise and challenges. We observe that they are dependent not solely on technology, but also on law, organisation, and not least the intention of those who made the rules without being aware of the possibilities offered by cryptography.

Innhold

1 Introduksjon

Den vanligste bruken av kryptografi er sikring av kommunikasjonslinjer. Forsvaret var naturlig nok tidlig ute med å bruke sterk kryptografi og har veletablerte rutiner knyttet til forvaltning av utstyr og nøkler. De siste årene har sterk kryptografi blitt en selvfølge i enhver digital tjeneste for privatpersoner, og den sivile kryptografiforskningen har gjort store konseptuelle nyoppdagelser de siste tiårene. De siste årene har sivile tjenester generelt vært raskere til å ta i bruk nyvinninger, mens det har vært en mer tilbakeholden holdning fra militær side, som i større grad har holdt seg til etablerte metoder.

De siste årene har vi utforsket i hvilken grad forsvars- og justissektoren kan dra nytte av moderne kryptografiske teknikker. Dette er en krevende øvelse. Normalt må man starte med problemet, og så finne ut hvilke verktøy som trengs for å lage en løsning, mens vi her i større grad starter med teknolgier for å se etter anvendelser. Med dette i tankene er innholdet i denne rapporten utforskende i sin natur: Vi har identifisert og fordypet oss i mulige anvendelser, men det må være opp til relevante beslutningstakere å gjøre de endelige vurderingene.

Teknikkene vi behandler her har til felles at de lar oss utføre beregninger på data mens de fortsatt er kryptert. Homomorf kryptografi gjør at datasenteret kan regne på våre data, men uten at det overhodet er mulig for datasenteret å lese dataene. Skreddersydd dekryptering kan gi noen tilgang til å beregne en forhåndsgitt funksjon av dataene, for eksempel en statistisk oppsummering. Sikre flerpartsberegninger gjør at vi – for anledningen uten tillit til hverandre – i fellesskap kan regne på svært sensitiv informasjon fra flere kilder, men uten at informasjonen lekker, og uten at vi må stole på noen tredjeparter.

Forutsetningen blant kryptologer er at man skal basere sikkerhet på matematikk, ikke tillit, og at det derfor må bygges matematisk sikkerhet inn i enhver interaksjon. Forutsetningen stammer ikke fra en dystopisk mistro mot menneskeheten, men ganske enkelt fra mangelen på mekanismer som kan godtgjøre hvorfor vi kan være sikre på at en fremmed på internett er til å stole på. Hvis man antar at tjenester og leverandører er pålitelige inntil det motsatte er bevist, mister kryptografien sin *raison d'être* – da trengs det ingen sikkerhet.

For Forsvaret og andre sentrale statstjenester er derimot forutsetningen ikke nødvendigvis sann. Gjennom sikkerhets- og leverandørklareringer kan man etablere en samling organisasjoner og personer man har tillit til, og så autorisere disse for visse typer informasjon. En ser derfor ikke nødvendigvis behovet for verken sikre flerpartsberegninger eller homomorf kryptografi, rett og slett fordi risikoen knyttet til å dele informasjon med den andre parten er kvantifisert og funnet akseptabel.

Til tross for at risikoen er forsvarlig, er den ikke null, og vi tar for gitt at beslutninger om å dele informasjon har vært fulgt av tanken «Hm, helst ikke, men her er resultatet for viktig til at vi kan la være». I andre tilfeller vil organisasjonens forståelse av sikkerhet ganske enkelt gjøre at man umiddelbart skyver fra seg muligheten for å sammenligne eller bruke data, for eksempel i et bilateralt forhold mellom to statsmakter.

Vi har derfor forsøkt oss på en letning etter tilfeller der noen enten tenker «Disse dataene må vi dele, selv om vi helst vil holde dem for oss selv» eller «Dette skulle vi gjerne gjort, men vi kan

ganske enkelt ikke dele disse dataene». Det er en vanskelig leting, for disse tankene blir i liten grad dokumentert, og vi er derfor avhengige av å nå fram til de riktige personene. De tiltakene som umiddelbart blir lagt bort fordi de virker umulige blir gjerne heller ikke dokumentert.

Oversikten vi har tatt fram de siste årene er derfor langt fra komplett. I kapittel 4 viser vi noen av eksemplene vi har funnet, og hvordan vi har arbeidet for å lage prototyper som løser problemene. Vi har tidligere samlet mulige eksempler i et notat [29].

Vi har forsøkt å gjøre rapporten lesbar for flest mulig. Vi ønsker å gjøre leseren i stand til å identifisere problemstillinger som kunne vært løst med sikre beregninger i sin egen tjeneste, men også ha et realistisk bilde på i hvilken grad en slik løsning er klar til å bli tatt i bruk innen rimelig tid.

For at flere skal kunne forstå eksemplene gir vi en innføring i teknikkene i kapittel 2. Vi har så forsøkt å gjøre rapporten så lite teknisk som mulig, men det er tilfeller der en interessert leser vil ha nytte av å se noe av matematikken som ligger bak. Dette skal være godt merket, og lesingen er ikke avhengig av forståelse av dette.

Vi vil fortsette arbeidet med å identifisere og teste mulige anvendelser av sikre beregninger, og oppfordrer alle lesere som kommer på ideer underveis til å ta kontakt.

2 Innføring i sikre beregninger

Før vi kan diskutere anvendelser, må vi ha en felles forståelse av hva vi mener med sikre beregninger. Vi fokuserer her på tre sentrale teknikker, presentert etter fallende teknisk modenhet.

2.1 Sikre flerpartsberegninger

Vi skal nå se på hvordan vi kan dele en beregning mellom flere parter. Den vanligste motivasjonen for å gjøre det er et ønske om å redusere belastningen på hver enkelt node, men det er ikke tilfellet her. I stedet er formålet å forhindre at noen av nodene får kjennskap til noe informasjon fra dataene.

En beregning vil ofte avhenge av sensitive data fra flere parter. Partene ønsker kanskje å lage statistikk basert på alles data, men ingen ønsker å gi fra seg sine egne data til noen av de andre. La oss derfor anta at det finnes en perfekt, lekkasjefri beregningsboks. På forhånd konfigurer vi boksen med hvor mange som skal sende data, hvilke data den skal få inn, og hvilke data som skal returneres (og, til hvem). Siden boksen er perfekt vil alle partene stole på den: at den får vite våre sensitive data, gjør ingenting. Når boksen er ferdig kommer statistikken ut igjen til partene, og den glemmer alt den har sett.

Boksen lar oss resonnere rundt sikkerheten. For det første er det klart at dersom noen sender inn gale tall, så vil svaret også være galt. Det problemet kan ikke løses av boksen. Videre er det slik at hver enkelt deltaker bare ser det som sendes inn i boksen og det som kommer ut tilbake. Dette åpner for at de kan lære noe, men boksen garanterer at de ikke lærer *mer* enn dette. Til sist kan vi observere at ingen av partene kan sende inndata som avhenger av noen av de andre sine data: ingen kan si «jeg byr én mer enn den forrige».

Eksempel 1. To kollegaer ønsker å finne ut hva gjennomsnittslønna deres er, men uten å røpe hva hver tjener. Begge sender inn sin egen lønn til boksen og får gjennomsnittet tilbake ut. Den andre er imidlertid litt smartere enn den første, så hen forstår at hen kan regne ut kollegaens lønn basert på sin egen og gjennomsnittet. Likevel har ikke boksen lekket informasjon: dette er en konsekvens av beregningen i seg selv.

Eksempel 2. Denne gangen er det ti kollegaer som ønsker å finne gjennomsnittslønna. Som før sender alle sin lønn inn i boksen og får til slutt gjennomsnittet tilbake. Den ene nysgjerrige kollegaen i eksempelet over kan i dette tilfellet beregne gjennomsnittslønna til de øvrige ni kollegaene, men verken prosessen eller boksen lekker ytterligere detaljer.

For det neste eksempelet antar vi at boksen har en avknapp, og at den sender ut svarene til én og én etter hverandre.

Eksempel 3. En gruppe investorer ønsker å bruke deres felles data til å generere investeringstips. Tre av medlemmene har observert at de er de første som får meldinger, og stiller seg derfor opp ved siden av avknappen. Så snart de får tilsendt tipsene skrur de av boksen, og utnytter dette til å gjøre en god investering de andre ikke får tilgang på. I dette tilfellet er ikke boksen *rettferdig* (*fairness*), og oppfyller dermed ikke alle egenskapene vi ønsker at vår perfekte boks skal ha.

En slik perfekt boks finnes ikke, iallfall ikke som en fysisk boks. I stedet står den som en modell for det vi kaller en *ideell funksjonalitet*. Imidlertid er det – under visse tekniske antagelser – mulig å bruke *sikre flerpartsberegninger* til å beregne enhver funksjon [16], og fortsatt oppnå sikkerhetsegenskapene vi allerede har skissert [23]:

Fortrolighet Ingen part skal lære mer enn utdataene den er ment å motta, og det som kan utledes fra disse.

Korrekthet Beregningen er utført korrekt, og ingen kan påvirke dette.

Uavhengige inndata Hver part må velge sine inndata uavhengig av andre parter. For eksempel skal ikke den uærlige kollegaen over kunne oppgi lønna si som «det dobbelte av den forrige».

Garantert levering Uærlige parter skal ikke kunne forhindre ærlige parter å motta utdata, for eksempel ved å kunne avbryte beregningen.

Rettferdighet Uærlige parter skal motta utdata hvis og bare hvis de ærlige partene også gjør det.

De to siste egenskapene ligner på hverandre, men det er bevist [9] at det er en meningsfylt forskjell på dem når det er mer enn to parter.

Den ideelle funksjonaliteten vi beskrev over oppfyller disse kravene, men det kan være vanskelig å bevise at man oppnår disse egenskapene i en ekte protokoll. Derfor går sikkerhetsresonnementet gjennom følgende prosess:

1. Først definerer vi den ideelle funksjonaliteten med inn- og utdata. Denne er per definisjon sikker.
2. Deretter lager vi en protokoll med meldinger fram og tilbake mellom partene, og som vi håper er sikker. Protokollen kan inneholde betingelser for å avbryte eller kaste ut parter som ikke oppfører seg.
3. Til sist lager vi en *simulator* som får sluttresultatet som input, men som ellers har akkurat de samme meldingsutvekslingene som protokollen over. Simulatoren genererer meldinger for de ærlige partene, men uten å kjenne hemmelighetene deres. De uærlige partene sender meldingene sine inn og ut av simulatoren. Vi kan garantere at de ikke kan påvirke eller lære noe utover sluttresultatet, siden dette er det eneste som er kjent for simulatoren.
4. Dersom de uærlige partene ikke kan se forskjell på om de kommuniserer med simulaturen eller de ærlige partene, så kan vi vise at protokollen faktisk realiserer den ideelle funksjonaliteten, og at de uærlige ikke kan ha lært eller påvirket noe.

Vi går videre ut fra noen forskjellige modeller for hvor kraftig motstanderen er. Konvensjonen er å anta at det finnes én motstander, men som kan kontrollere og koordinere alle de uærlige partene, og ellers overvåke nettverket.

Semi-ærlig En semi-ærlig motstander vil alltid opptre i henhold til protokollen, men der en helt ærlig part vil ignorere data den ikke skal se, vil denne motstandren samle det inn og sammenstille det med andre data. Motstanderen kalles også passiv.

Ondsinnet En ondsinnet motstander opptrer om mulig i strid med protokollen, og vil kunne forsøke å sende gale meldinger i et håp om at det vil framprovosere lekkasjer. Derfor kalles også denne motstanderen aktiv.

Fordekt Denne motstanderen opptrer som den forrige, men med den betingelsen at den unngår å gjøre noe som vil gjøre at den blir oppdaget.

Det kan også stilles betingelser til hvordan motstanderen kan velge ut de uærlige partene, og eventuelt hvor stor andel. Vi henviser til den gode innføringen av Lindell [23] for flere detaljer.

Sikre flerpartsberegninger er et stort teoretisk forskningsfelt, og ble første gang for alvor tatt i bruk i 2008 for å gjennomføre danske sukkerbeteauksjoner [6]. Danmark har mange sukkerbetebønder, og én dominerende grossist, Danisco. På grunn av endringer i markedet var det nødvendig å avholde en auksjon for å finne riktig pris for sukkerbetene. Bøndene ønsket ikke å røpe informasjon om egen stilling og produktivitet, mens Danisco naturlig nok ønsket lavest mulig pris. Ved å legge inn bud kunne partene i fellesskap beregne den prisen som gjorde at markedet ble ryddet: alle produksjonskvotene ble omsatt, og til den laveste prisen som gjorde at det ikke var ytterligere tilbud.

Bøndene, Danisco og danske kryptologer gikk derfor sammen om å utføre auksjonen, og i løpet av 30 minutter hadde partene funnet riktig pris, og kunne reallokere produksjonen av 25 000 tonn beter.

Siden MPC har vært et aktivt felt i flere tiår har det også blitt laget flere implementasjoner. Vi vil dra fram to av disse, og viser til Dragoș Rotarus oversikt over MPC-ressurser for flere detaljer[1].

Det første rammeverket er MP-SPDZ av Keller [20]. MP-SPDZ er et rammeverk for å teste og benchmarke MPC-protokoller. Det er primært laget for at kryptologer kan undersøke hvordan nye måter man kan utføre disse beregningene på («protokoller») måler seg mot andre. Imidlertid inkluderer MP-SPDZ også en kompilator som gjør at brukeren kan beskrive beregningene sine ved hjelp av Python-kode. Deretter velger man hvilke sikkerhetsegenskaper man ønsker, og dermed hvilken protokoll. MP-SPDZ støtter kjøring av programmet på separate maskiner i et nettverk, eller ved å sette opp flere små virtuelle instanser på samme maskin. Vi har tidligere laget en enkel innføring til MP-SPDZ[2]. Kildekoden distribueres med en fri lisens og er gratis å bruke, men er i likhet med de øvrige tilgjengelige implementasjonene ikke ment for produksjonsbruk.

Vi vil også nevne CipherCompute fra det franske selskapet Cosmian. Verktøyet lar brukeren skrive et program i Rust, og lager deretter et brukervenlig grensesnitt der brukere kan legge inn sine data og kjøre beregningen. Heller ikke denne programvaren er klar for produksjonsmiljøer, men det er et tydelig tegn på at også kommersielle selskaper ser potensialet i sikre flerpartsberegninger.

2.2 Homomorf kryptografi

Homomorf kryptografi handler om å gjøre beregninger på krypterte data uten å dekryptere dem først [15]. Det inngår to parter: en part som eier data og en part som gjør beregninger på dataene.

[1] `https://github.com/rdragos/awesome-mpc?tab=readme-ov-file#software`

[2] `https://github.com/martstr/mp-spdz-tutorial`

Den som eier dataene krypterer dataene sine, og sender deretter de krypterte dataene til den andre parten. Mottakeren kan ikke dekryptere dataene, og får dermed ingen kjennskap til hva de innholder. Likevel vil hen kunne gjøre beregninger på dataene, men resultatet av disse beregningene vil fortsatt være kryptert, og vil ikke gi mening før de sendes tilbake til den første parten, som kan dekryptere resultatet.

Eksempel 4. Microsoft har tatt i bruk homomorf kryptografi i praksis. «Password Monitor»-funksjonen i nettleseren Edge brukes for å sjekke om et passord er kjent og dermed ligger i lister som brukes for å knekke passord. Veldig forenklet vil en kandidat til passord bli oversendt fra brukeren til Microsoft i kryptert form. Microsoft gjør en homomorf beregning som bedømmer om dette passordet er kjent eller ikke, og sender resultatet tilbake til brukeren, fortsatt i kryptert form. På den måten vil ikke Microsoft lære noe om passordet, men de slipper samtidig å dele hele passorddatabasen med brukeren.

Hva kan vi oppnå med å dele opp beregningene på denne måten? Ofte beskrives en situasjon der den første parten har veldig liten beregningskapasitet, og derfor setter bort beregningene til en annen part, for eksempel som en del av en skytjeneste. De beregningene som gjøres på homomorft krypterte data har imidlertid fortsatt veldig stor overhead sammenlignet med å gjøre beregningene på de opprinnelige dataene, så et slikt scenario er foreløpig lite realistisk. En annen mulighet er at den parten som gjør beregningene sitter på algoritmer eller beregningsparametre som hen enten ikke ønsker å dele, eller som det er lite praktisk å dele av andre årsaker som f.eks. størrelse. I slike situasjoner ønsker vi at homomorf kryptografi skal kunne sikre både data og beregninger.

Sikkerhetsmodellen for homomorf kryptografi beskytter altså primært data mot at disse blir kjent for den som utfører beregningene. Samtidig stoler vi på at den som utfører beregningen gjør dette korrekt, siden det ikke er noen beskyttelse mot en motstander som med vilje ønsker å utføre en uriktig beregning. En annen sikkerhetsegenskap som relativt enkelt kan oppnås ved homomorf kryptografi er det som kalles *circuit privacy* [8]. Det betyr at de beregningene som gjøres, altså selve algoritmen, kan holdes fullstendig skjult for den som eier dataene. Modellen med homomorf kryptografi sikrer altså at parten som eier dataene ikke lærer mer om beregningene enn selve resultatet, og at den som gjør beregningene ikke lærer mer om selve dataene enn hvor mye data det er.

Det fins ulike varianter av homomorf kryptografi, som skiller seg fra hverandre ut fra hvilke beregningsoperasjoner som det er mulig å gjøre på dataene. Vi fokuserer her på *Fully Homomorphic Encryption* (FHE) som tillater at man kan gjøre vilkårlig komplekse operasjone på dataener. Selv om det har vært en stor utvikling på dette området de siste 15 årene, er det fortsatt slik at selv enkle beregninger tar lang tid. Praktisk bruk er derfor ofte begrenset til anvendelser som krever relativt lette beregninger.

Programvare for homomorf kryptografi er forholdsvis omfattende, og man vil nesten alltid benytte seg av rammeverk som har implementert de grunnleggende matematiske funksjonene. Det fins mange rammeverk å velge blant. Ett eksempel er den åpne kodebasen til selskapet Zama[3] som er

[3] `https://www.zama.ai/`

skrevet i Rust med overbygninger i Python. Andre eksempler er OpenFHE[4] og Microsoft Seal[5] som er skrevet i C++.

2.3 Skreddersydd dekryptering

Skreddersydd dekryptering (FE, *functional encryption*) lar oss dekryptere en beregning på de krypterte dataene, men ikke nødvendigvis dataene direkte.

En typisk krypteringsprosess innebærer å bruke to algoritmer: én for å kryptere klarteksten slik at den blir uleselig, og én for å gjenskape originalteksten fra den krypterte teksten. Disse algortimene skal fungere slik at dersom du først krypterer en melding, for så å dekryptere den, skal du ende opp med den opprinnelige meldingen. Motivasjonen bak skreddersydd dekryptering er at man ser utover disse begrensnignene og setter opp kryptering og «dekryptering» slik at man ikke får tilbake klarteksten, men noe annet basert på hva klarteksten var.

Dette kan bli veldig bredt. For eksempel kan det være at «dekrypteringen» kun avslører noen egenskaper av klarteksten, eller at man bare kan finne ut informasjon om en gruppe med meldinger. Definisjonen er såpass bred at det ikke er åpenbart hva det skal bety at noe er sikkert i skreddersydd dekryptering [7]. La oss ta derfor for oss et eksempel.

> **Eksempel 5.** En kommune ønsker seg sikre, krypterte valg. De implementerer det gjennom skreddersydd dekryptering, og genererer nøkler via en MPC-protokoll, slik at ingen sitter med den private nøkkelen. I stedet gir protokollen ut en nøkkel som kan brukes til å beregne og dekryptere valgresultatet.
>
> Stemmer blir kryptert og kan ikke leses direkte, men myndighetene kan ta inn alle stemmene og beregne en oversikt over hvor mange stemmer som gikk til hver mulighet. De kan ikke dekryptere stemmene eller beregne noe annet.

Eksempel 5 viser både styrken og svakheten med skreddersydd dekryptering. Styrken er at matematikken gir sikkerhetsgarantier for at man ikke kan lære noe mer enn hva man lærer av å beregne funksjonen man har tilgang til. Ulempen er at det ikke er åpenbart hvor mye informasjon som lekker gjennom å få resultatet av en gitt beregning. Dersom vi antar at myndighetene i eksempel 5 kan beregne funksjonen på større men fleksible mengder av krypterte stemmer kan de i praksis dekryptere enkeltstemmer indirekte gjennom å velge å beregne en oversikt fra alle stemmene, for så å unnlate én stemme og gjøre beregningen på nytt og observere forskjellen.

Skreddersydd dekryptering har to konseptuelle forgjengere. Den tidligste er identitetsbasert krypto (IBE) [27, 28], som gjør det mulig å bruke en identitet, for eksempel en e-postadresse, til å kryptere slik at bare riktig mottaker skal kunne dekryptere. Den neste heter attributtbasert kryptering (ABE) [18]. Det kan man bruke til å merke nøkler eller chiffertekster med attributter, og en mottaker kan bare dekryptere når det er samsvar mellom nøklene og merkingen av chifferteksten.

[4] `https://www.openfhe.org/`
[5] `https://github.com/Microsoft/SEAL`

Skreddersydd dekryptering generaliserer dette videre. Vi kan beskrive ABE som skreddersydd dekryptering via funksjonen «dekrypter alt hvis det er samsvar, ellers ingenting». Generell FE kan beregne vilkårlige funksjoner, men foreløpig er det bare kjent hvordan man kan lage effektiv FE for forholdsvis enkle funksjoner[6] [1, 22, 30]. Det finnes implementasjoner av ABE, men de er eksperimentelle.

Det er også teoretisk kjent hvordan vi kan lage skreddersydd dekryptering for vilkårlige funksjoner, men disse er det foreløpig ikke mulig å implementere og bruke effektivt [13, 17].

[6] Konkret betyr «forholdsvis enkel» her at funksjonen kan uttrykkes som et flervariabelt polynom av grad 2.

3 Iboende begrensninger

Konstruksjonene vi har beskrevet i kapittel 2 utfordrer i noen grad tradisjonelle måter å tenke på sikkerhet i formene av konfidensialitet, integritet og tilgjengelighet. For å ha en riktig forståelse av sikkerhetsegenskapene til algoritmene som inngår i sikre beregninger må man være presis i hvordan man definerer disse begrepene, og kanskje like viktig være klare på hva de *ikke* dekker.

Dette kapittelet beskriver en måte å tenke på om hva man beskytter i sikre beregninger, og hvilke begrensninger som ligger i det.

3.1 Ulike typer fortrolighet i beregninger

Vi skal nå se nærmere på *fortrolighet*, her forstått som egenskapen at mekanismen som vi bruker til å utføre beregninger ikke lekker privat data. Hvis vi bare behandlet persondata, så ville personvern vært et godt synonym. Bogdanov [4] deler fortrolighet inn i fire ulike kategorier basert på hva man er interessert i å beskytte og hvilket utgangspunkt man tar. De to første kategoriene tar utgangspunkt i dataene i seg selv, mens de to siste tar utgangspunkt i beregningsprosessen. Det er verdt å merke seg at studiet av fortrolighet og personvern går utover bare kryptografifeltet, og forskningsarbeid på området er ofte gjort i forbindelse med sikre databaser og dataanalyse [4, 11, 12, 21] som ligger utenfor det vi diskuterer i denne rapporten.

Den første kategorien er fortrolighet for enkeltdatapunkter, slik at beregningen ikke lekker noe informasjon om disse. Denne egenskapen er spesielt relevant når man gjør beregninger basert på skjermet eller sensitiv informasjon som helsedata eller personidentifiserende data. Det burde da ikke være mulig å lære noe om enkeltindivider basert på resultatene av beregningen. Dette betyr typisk at man må introdusere en form for tilfeldighet, og vil gå på bekostning av treffsikkerhet.

Den andre kategorien er fortrolighet for en samling med data fra samme kilde. Dette skiller seg fra den første kategorien ved at man ikke bare ser på identifiserende informasjon, men også egenskaper eller metadata som tilhører hele samlingen, som for eksempel hvor mye data det er, relative frekvenser eller avvikende datapunkter. Disse betrakningene er spesielt relevante for beregninger som inkluderer et samarbeid mellom separate aktører.

Den tredje kategorien av fortrolighet er fortrolighet for resultatene til en beregning, og innebærer å studere hvilken informasjon som lekker fra selve resultatet. Dette er spesielt relevant for beregninger der vi ønsker å beregne noe som blir allment tilgjengelig basert på informasjon vi ønsker å beskytte, som jammeryggsekkscenarioet vi beskriver i avsnitt 4.5.

Den fjerde og siste kategorien kommer litt på siden av de andre og kan kalles kryptografisk fortrolighet (*cryptographic privacy*). Dette innebærer at matematikken man gjør i beregningene ikke lekker noe informasjon utover selve resultatet.

Metodene for sikre beregninger som er beskrevet i denne rapporten tar utgangspunkt i at denne egenskapen er oppfylt, ettersom det ikke er funnet noen svakheter i metodene så langt. Når kryptografisk fortrolighet er oppfylt kan vi se på beregningene som om de er gjort i en svart boks, noe som gjør det lettere å undersøke de andre typene fortrolighet.

Dersom kryptografien beskytter systemet vårt perfekt vil beregningen i all praktisk forstand oppføre seg som en svart boks for en potensiell motstander. Det betyr ikke at det ikke er noen potensielle sårbarheter og lekkasjer – det kan fortsatt være utfordringer knyttet til fortrolighet i beregningene.

3.2 Begrensninger i sikre flerpartsberegninger

Anta at vi kan gjøre sikre flerpartsberegninger i en svart boks, slik at vi vet at vi kryptografisk beskytter alle datapunkter og det ikke er noen sårbarheter som avslører hvem som bidrar med hva til beregningene. Hva er det som fortsatt kan lekke av informasjon? Vi starter med et illustrerende eksempel.

Eksempel 6. Medlemmene i en komité skal stemme over en viktig beslutning, men stemmene er hemmelige. Hvert medlem kan velge å stemme JA eller NEI til forslaget, og det endelige JA/NEI-resultatet blir delt i etterkant. Beslutningen krever enstemmig JA for å vedtas. Dersom resultatet blir NEI, kan alle som observerer resultatet konkludere med at det finnes et land som stemte NEI – noe lekkasje av informasjon, men ikke mye. Dersom resultatet derimot blir JA vil alle sitte igjen med en perfekt oversikt over alle stemmer, ettersom alle stemmer må ha vært JA. Dette er fullstendig brudd på konfidensialitet av stemmene, kun basert på det åpne resultatet.

Som det kommer fram av eksempel 6 kan resultatet av en beregning føre til fullstendig lekkasje av dataene som ble brukt til beregningen. I noen tilfeller kan det være tilsiktet at en slik lekkasje oppstår, men i andre kan det være utilsiktet – det viktige er å være klar over informasjonslekkasjen som følger av resultatet. Som det også framgår av eksempelet kan informasjonslekkasjen også være avhengig av hvilket resultat som faktisk ender opp som svar, så en analyse av fortrolighetsegenskapene må omfatte en forståelse for mange mulige resulater. Slike analyser blir nødvendigvis spesifikke til hver beregningsalgoritme og hver implementasjon.

I noen beregninger vil det også være forskjell på hva som lekker til deltakere i beregningen og hva som lekker til aktører som observerer beregningen fra utsiden. Dette illustreres i eksempel 1, der deltakerne kan beregne den andres lønn. Et annet eksempel på det samme følger.

Eksempel 7. To venner ønsker å avgjøre hvilken film de skal se sammen på kino. De lager begge en liste over hvilke filmer de har lyst til å se fra programmet, og så gjør de en sikker flerpartsberegning for å komme fram til en liste over de filmene begge har lyst til å se. En observatør fra utsiden som kun ser listen over felles favoritter lærer hvilke filmer de begge ønsker å se, og at minst én av de to vennene ikke ønsket å se filmene som ikke endte på den ferdige listen – men de vet ikke hvem, eller om det var begge. På den andre siden, hver av vennene som var involvert

i beregningen vet nøyaktig hva den andre mente om de filmene som de selv hadde lyst til å se – dersom den andre ønsket å se filmen endte den på den ferdige listen, og dersom de ikke ønsket å se filmen endte den ikke på listen. Merk også at dersom en av de to involverte kun ønsker å innhente informasjon om den andres preferanser kan de oppgi at de har lyst til å se alt på programmet, for så å ende opp med en perfekt gjengivelse av hva den andre har lyst til å se. Brukere av dating-apper vil kunne gjenkjenne dette som *always swipe right*-strategien.

Informasjonslekkasje kan også oppstå som følge av at en aktør selv sitter på noe informasjon om beregningen. Vi illustrerer igjen med et eksempel.

Eksempel 8. En rekke land samarbeider om å koordinere radarsignaturer, og har satt opp et system som gjør en sikker flerpartsberegning som slår opp en observert signatur mot alle radardatabasene og beregner hvilken signatur som passer best fra det sammensatte datagrunnlaget. En motstander ønsker å finne ut av om deres radarsignatur finnes i en av databasene, og framprovoserer en beregning ved å fly forbi en lyttepost. Motstanderen observerer at resultatet av beregningen ikke gir riktig signatur, og konkluderer med at ingen av databasene inneholder deres radarsignatur.

På den ene siden er slike lekkasjer en trussel mot informasjonen som vi ønsker på beskytte. På den andre siden kan disse iboende lekkasjene også brukes for å dele hemmeligheter i løpet av beregningen som uansett vil røpes av det som blir kjent til slutt, som kan gi ytelsesgevinster i konkrete implementasjoner. Det er gjort noen slike analyser [2, 5, 19], men de er veldig spesifikke helt ned på implementasjonsnivå.

3.3 Begrensninger i skreddersydd dekrytpering

Anta at vi kan gjøre skreddersydd dekryptering som beskrevet i avsnitt 2.3 i en svart boks som garanterer kryptografisk fortrolighet. Da vet vi at ingenting annet enn funksjonen eller funksjonene som protokollen tillater kan lekke informasjon. Målet blir da å få et bilde av hva en trusselaktør kan få ut av en slik funksjon, gitt tilgang til den. I likhet med hva vi så for sikre flertallsberegninger blir en slik analyse fort veldig spesifikk og utfordrende å evaluere.

Som et første eksempel på begrensning, anta at vi bruker skreddersydd dekryptering for å gjøre oppslag i en kryptert database, for eksempel fordi databasen er sensitiv, men at den trengs ombord en ubevoktet enhet. En trusselaktør som lykkes med å plukke opp enheten kan da utsette den for ekstern påvirkning for å utløse tilsvarende databaseoppslag. Vi kan forenkle dette resonnementet til å anta at motstanderen i dette tilfelle vil kunne utføre vilkårlige databaseoppslag. Dersom denne funksjonen for eksempel gjør et oppslag mot kjente radarsignaturer slik vi beskriver i kapittel 4.1,

kan trusselaktøren evaluere funksjonen med sine egne radarsignaturer for å finne ut hvilke som er i databasen og hvilke som ikke er i databasen.

I overnevnte scenario bruker dekrypteringen kun én funksjon, men i det mer generelle tilfellet kan vi sette opp flere funksjoner med skreddersydd dekryptering. Da må man ikke bare ta med i betraktningen hva hver enkelt funksjon lekker, men også hva enhver sammenstilling lekker. Som eksempel på dette, si man har satt opp skreddersydd kryptering for en liste slik at man kan få vite hvilket element i listen som er minst, samtidig som man også har tilgang til en funksjon som gir mulighet til å slette et element. Hver for seg lekker funksjonene kun det minste elementet, men sammenstilt kan vi bruke de iterativt for å sortere hele listen. En slik sortert liste vil da lekke mer informasjon, som igjen kan sammenstilles med annen informasjon.

4 Eksempler

I de forrige kapitlene har vi gått gjennom hva som både er teoretisk mulig og ikke mulig med sikre beregninger fra kryptografi. Nå skal vi gå gjennom noen eksempler vi har arbeidet med, og se hvordan det vi har diskutert tidligere kommer til anvendelse her. Eksemplene er løst satt sammen etter hvilken teknikk fra kapittel 2 vi har brukt.

4.1 Radardeteksjon

En vellykket operasjon krever god situasjonsforståelse, samtidig som man ønsker å nekte motstanderen den samme informasjonen. Tidlig i andre verdenskrig hadde Storbritannia gjort store forbedringer i radarteknologi, noe som førte til store tap for motstanderens luftstyrker. Radaren er fortsatt en sentral sensor både i lufta og på vannet.

Radaren virker ved å sende en serie med signaler på bestemte bølgelengder, og så pause sendingen mens man venter på at signalene skal reflekteres tilbake. Ved å måle refleksene over tid kan man få informasjon om en fremmed gjenstand og hvordan den beveger seg. Forskjellige konfigurasjoner av radaren gir forskjellig effekt avhengig av hva man ønsker å oppdage og hvor langt unna dette er, og radaren kan derfor også være tilpasset det aktuelle fartøyet og bruksområdene.

Imidlertid kan også radaren brukes til å avsløre at et fartøy opererer i området. Dersom man monterer en antenne som kan oppdage radarsignalet, så kan man bruke målingene til å gjenkjenne visse karakteristika ved radaren. Ved å sammenligne dette med tidligere målinger av kjente fartøy får vi en indikasjon på om det er radaren til en fregatt eller en fiskebåt.

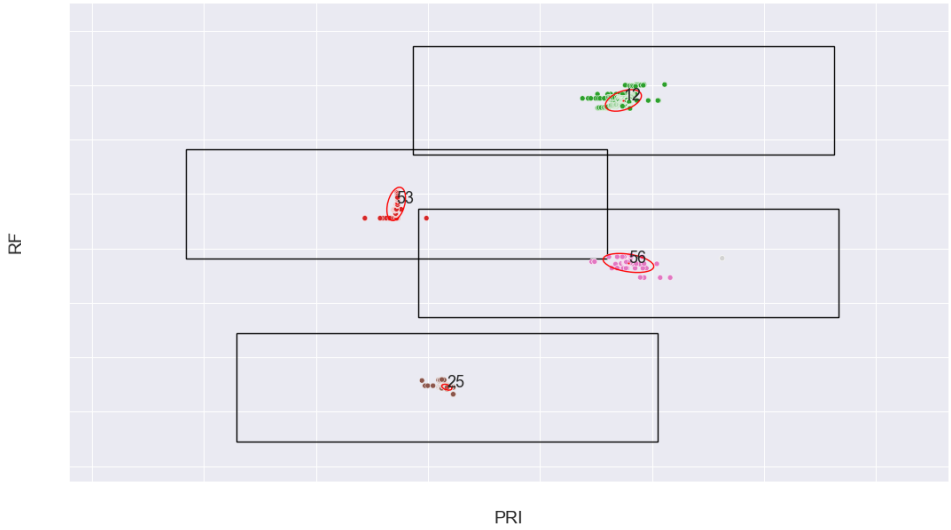
Databaser med signaturer av slike radarsignaler er derfor svært verdifulle for å forstå hvem som befinner seg i eget operasjonsområde, og uten at en selv trenger å røpe sin posisjon. På grunn av verdien kan slike databaser bli høyt gradert, og noe man ugjerne deler selv med koalisjonspartnerne sine.

Spørsmålet blir dermed hvordan man kan få full effekt fra den samlede kunnskapen i koalisjonen, men uten å måtte dele annen data enn «venn» eller «fiende», og kanskje hvilken type fartøy det er snakk om. Spesifikt ønsker vi heller ikke at det blir kjent hvilken database som inneholdt den riktige oppføringen, fordi det kan si noe om kapabilitetene til eieren av den databasen. Fordi det er flere parter som alle bidrar med data, anser vi sikre flerpartsberegninger som det beste verktøyet.

Denne utfordringen ble undersøkt av sommerstudentene Mathias Nordal og Benjamin Hansen Mortensen sommeren 2023, og resultatet er beskrevet i detalj i en vitenskapelig artikkel [26].

4.1.1 Teknisk beskrivelse av løsningen

Vi har brukt to sentrale karakteristika ved radarsignalet for å klassifisere: frekvensen som blir brukt i selve signalet (RF, *radio frequency*) og tiden mellom hver sending (PRI, *pulse repetition interval*). Som utgangspunkt fikk vi tilgang til et datasett som hadde målt skip i Vestfjorden i løpet av et drøyt døgn. Dette datasettet kunne vi bearbeide til å utarbeide signaturer for hver radar. Hver signatur består av tre elementer:



Figur 4.1 Klynger med overlappende bokser. Ellipsene beskriver kovariansmatrisen.

- gjennomsnittet av målingene for samme fartøy
- en boks om midtpunktet; ±3 MHz på RF-aksen og ±10 % på PRI-aksen
- en kovariansmatrise som beskriver spredningen av datapunktene

Alle elementene er illustrert i figur 4.1.

Anta nå at vi får tilgang på en ny måling av et ukjent fartøy. For å identifisere riktig kilde starter vi med å finne alle signaturene som har en boks som omfatter dette punktet. Vi sitter da igjen med en samling av plausible fartøy. For å avgjøre nøyaktig hvilket fartøy det mest sannsynlig er, bruker vi statistisk hypotesetesting ved hjelp av midtpunktene og spredningen. Det lar oss kvantifisere hvor rimelig det er å tro at den nye målingen faktisk kommer fra en gitt kilde, og vi velger da den som virker mest rimelig.

Vi har deretter implementert denne ideen i et MPC-rammeverk. Der har vi flere forskjellige varianter for å vise virkningen av å ha færre eller flere deltakere. Felles for alle er at én part bidrar med den nye målingen, mens den komplette databasen er delt mellom én til fire andre parter. Beregningen gir ut en liste over plausible fartøy, sammen med et tall som er koblet til sikkerheten for at det faktisk kan være dette fartøyet. Ikke overraskende øker kjøretiden når flere parter skal gå sammen om å utføre beregningene, og flere dermed må kommunisere med hverandre. Testene er utført på en enkel bærbar datamaskin. Se tabell 4.1 for de konkrete kjøretidene.

Vi henviser til artikkelen av Mortensen, Nordal og Strand [26] for de matematiske detaljene.

4.1.2 Demonstrasjon

I forbindelse med dette arbeidet utarbeidet vi også en demonstrator, der vi tillot oss å lage et hypotetisk scenario. Anta at vi har satt opp to antenner på kysten av Møre og Romsdal. Vi kan anta

m	2	3	4	5
Kjøretid (s.)	31	86	200	400

Tabell 4.1 Kjøretider med to signifikante sifre for signaturmatching, med $m = 2, 3, 4, 5$ parter. Målingen ble utført på en HP Elitebook 840 G6 med 16 GB RAM og Intel i5 8265U 1,6 GHz CPU.

at disse kan fange opp et signal fra et fartøy som er opptil 100 km unna, og fordi det er to antenner kan vi bruke informasjonen til å triangulere posisjonen til fartøyet.

Det kommer et ukjent fartøy inn i dekningsområdet til disse sensorene. Dersom skipet holder 13 knop, så vil en detekteringstid på 31 sekunder sørge for at skipet er identifisert innen det har seilt omtrent 200 meter inn i dekningsområdet.

La oss også se for oss et svært hurtig fartøy i 60 knop. Selv når vi bruker fire databaser, så får vi et resultat i løpet av 400 sekunder. På den samme tiden har fartøyet beveget seg 12,3 km inn i området.

4.1.3 Iboende lekkasjer

I arbeidet har vi antatt at observatøren ikke deler det konkrete signalet med noen av partene som sitter med databasene. Det gjør at ingen kan bruke bruke resultatet til å plassere et en ny signatur i sin egen database: man vet ganske enkelt ikke noe hvordan den signaturen skal se ut.

Det er flere utfordringer knyttet til denne antagelsen. For det første er antagelsen i seg selv tvilsom. Måleutstyret tilhører en av partnerne, og det er ingen grunn til å tro at den partneren vil la være å ta vare på målingen, og sammenligne denne med all tilgjengelig informasjon i etterkant.

For det andre så blir listen over mulige treff synlig for alle: Dersom vi selv har noen av disse mulige treffene, så vil vi også kunne gjøre rimelige antagelser om omtrent hvordan signaturen ser ut, om enn uten detaljer.

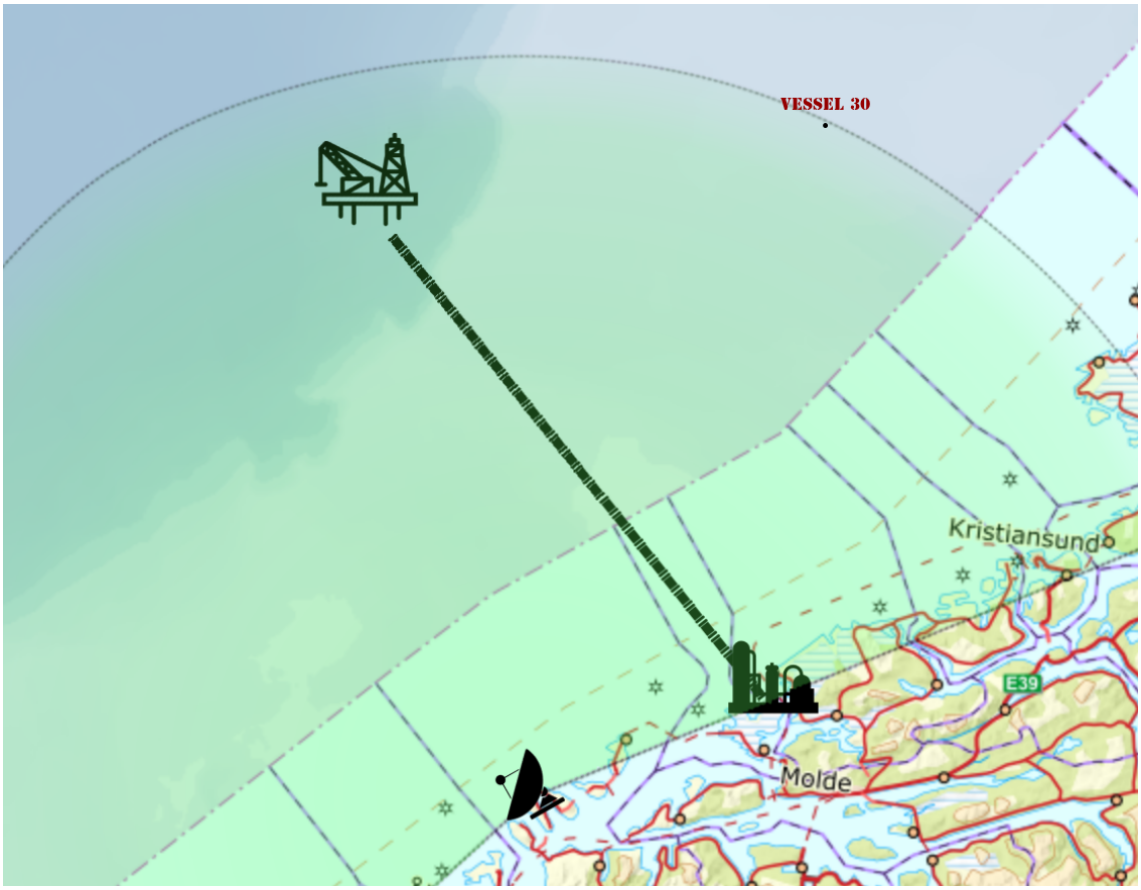
Legg merke til at dette er utfordringer som er direkte knyttet til funksjonaliteten i seg selv. Dersom man ønsker å kunne samarbeide om å identifisere fartøy, så vil disse lekkasjene uansett oppstå.

4.2 Passasjerlister

Politiregisterforskriften [3, kap. 60] hjemler innsamling av passasjerlister for fly inn og ut av Norge i PNR-registeret. Registeret er lokalisert hos Kripos, og etter forskriften er politiet, Politiets sikkerhetstjeneste (PST), påtalemyndigheten, Tolletaten og Etterretningstjenesten kompetente myndigheter for å be om søk i registeret.

Vi legger til grunn at PST og Etterretningstjenesten kan ha behov for å gjøre søk som i seg selv er gradert, og at Kripos ikke skal ha innsyn i disse søkene. Vi står derfor i en situasjon der én part har en sensitiv database, mens en annen part har et sensitivt søk.

I drøftinger av dette arbeidet har vi fått oppgitt at det grovt sett er tre typer søk som er aktuelle:



Figur 4.2 Skjermbilde fra demonstratoren. Kartet er © norgeskart.no (CC-BY 4.0).

1. historiske søk på flygninger og personer
2. faste søk på nye lister
3. regelbaserte søk som setter sammen flere kriterier og returnerer de mest relevante resultatene

Passasjerinformasjonsenheten (PIU) ved Kripos utfører menneskelige vurderinger av om hvorvidt et søk er lovlig, og om søketreffene er rimelig avgrenset til å ikke svare på vesentlig mer enn det søket skal få fram. Intensjonen for dette er personvern: Registeret skal utlevere minst mulig informasjon, og bare når det er tilstrekkelig begrunnet.

Vi har hatt kontakt med PIU og andre relevante aktører. I februar 2024 demonstrerte vi en løsning som kan utføre de to første søkene over. Arbeidet er gjort av masterstudent Benjamin Hansen Mortensen [25].

Vår vurdering er at flerpartsberegningene og passasjerlisteproblemet foreløpig ikke er modent for hverandre. Det skyldes først og fremst at det i dag er krav om en rekke menneskelige vurderingene, noe som står i direkte konflikt med å holde både søket og resultatene hemmelig for PIU. Det er også vanskelig å kunne gjennomføre nøyaktig de søkene som de kompetente myndighetene ønsker på en effektiv og sikker måte. Mer forskning trengs for å avgjøre både om en slik ambisjon er mulig og ønskelig gitt dagens lovverk. Ikke minst må den forskningen også være bredere enn bare teknologi.

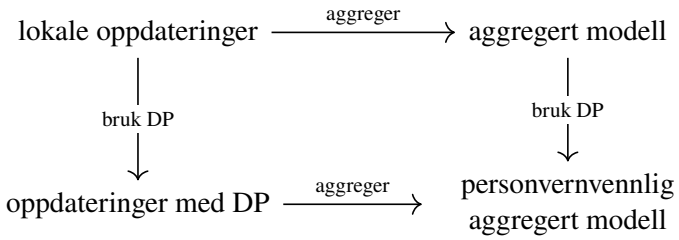
4.3 Sikker aggregering i maskinlæring

For å lage de store maskinlæringsmodellene som brukes i kunstig intelligens må vi prosessere store datamengder. For eksempel kan man trene opp en modell som kan kjenne igjen mistenkelig trafikk i et mobilnettverk. Dette er blant temaene som behandles i Horisont Europa-prosjektet PRIVATEER [10], der blant andre FFI deltar.

Anta videre at de ulike eierne av data ikke ønsker å dele data med andre, men likevel ønsker å nyte godt av en felles modell som lar dem oppdage trusler tidlig. Hvordan kan de bygge en felles modell uten å dele data?

Den første delen av svaret kalles føderert læring (FL, *federated learning*). Den sentrale noden distribuerer den gjeldende modellen til de øvrige nodene, og de beregner oppdateringer til denne basert på sine egne data. Disse oppdateringene blir så sendt inn til den sentrale noden, som setter sammen oppdateringene til en ny modell. Dette må vi gjøre over flere runder, før vi til slutt sitter igjen med en felles, oppdatert modell.

Føderert læring alene beskytter imidlertid ikke opplysningene mot innsyn; oppdateringen og modellen kan lekke data. Derfor trenger vi også differensielt personvern (DP, *differential privacy*). Denne egenskapen garanterer at den resulterende modellen ikke lekker noe informasjon om de enkelte datapunktene i kildematerialet.



Figur 4.3 To veier fra lokale data til personvernvennlig modell.

Betrakt figur 4.3. Vi ønsker å komme oss fra øverste venstre hjørne til nedre venstre hjørne, og vi kan da gå to veier: enten legge til DP først, og så aggregere, eller først aggregere og deretter legge til DP. Den første varianten vil gjøre sluttresultatet dårligere, fordi summen av kostnaden med å legge til DP vil gjøre modellen mindre presis. Den andre varianten vil kunne bli mer presis, men forutsetter at den sentrale tjenesten får tilgang til oppdateringene før vi har brukt DP, noe som kan være en personvernrisiko.

Løsningen på dette kan være å bruke sikre flerpartsberegninger til å aggregere og legge til DP. På den måten får vi en mest mulig presis modell, men uten at grunnlagsdataene lekker. Vi viser til Mansouri et al. [24] og Gehlhar et al. [14] for ytterligere detaljer.

4.4 Navigasjon i minelagt farvann

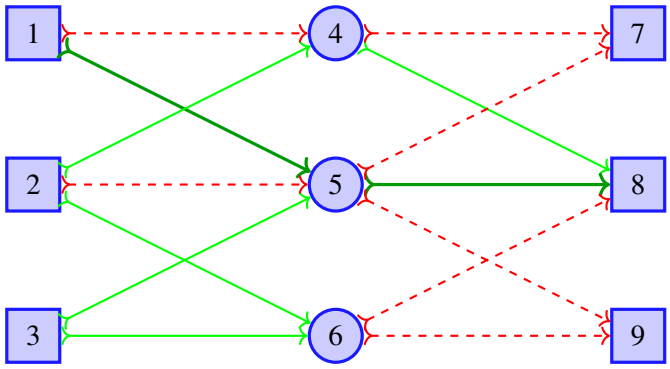
Som et eksempel på et scenario der vi kan bruke homomorf kryptografi har vi sett på et enkelt tilfelle av ruteplanlegging i et minelagt farvann. Scenarioet har to aktører: et skip og en ruteplanlegger.

Skipet vet hvor det er og hvor det ønsker å seile, mens ruteplanleggeren vet hvor alle minene i området er.

Hvis vi tenker oss at skipet ikke ønsker å oppgi sin posisjon og rute til ruteplanleggeren, og at ruteplanleggeren ikke vil røpe mer om hvor minene er plassert enn det som trengs for å gi skipet en trygg rute, har vi et problem som kan løses ved hjelp av homomorf kryptografi. Skipet krypterer først sin posisjon og ønsket destinasjon, og sender det til ruteplanleggeren. Ruteplanleggeren gjør beregninger på de krypterte dataene med en algoritme som ikke er kjent for skipet. Resultatet av beregningene er en kryptert versjon av en sikker rute. Denne sendes så tilbake til skipet, som kan dekryptere resultatet.

4.4.1 Implementasjon

For å vise at et slikt scenario kan implementeres har vi basert oss på programmvare fra selskapet Zama[7]. De utvikler et rammeverk for FHE som er åpen kildekode og lett tilgjengelig. Rammeverket er i hovedsak skrevet i Rust, men vi har her brukt et overordnet grensesnitt mot Python.



Figur 4.4 Et forenklet scenario for navigasjon i minelagt farvann.

Eksemplet som er valgt er gjort så enkelt som mulig, og er vist i figur 4.4. Det består av tre mulige startpunkter (1–3) til venstre, tre punkt underveis (4–6), og tre destinasjoner til høyre (7–9). Heltrukne grønne piler er trygge, mens de stiplete røde viser områder som er minelagt. Skipet krypterer endepunktene og får tilbake den første trygge ruten i en forhåndsbestemt liste av alle rutene. Hvis for eksempel skipet vil fra 1 til 8, kan det få ruten 1–5–8.

4.4.2 Begrensninger

FHE krever tunge beregninger, og kjøretider blir fort lange. Eksemplet er derfor ment som en illustrasjon og er med vilje gjort veldig enkelt. De algoritmene som er brukt vil derfor ikke direkte skalere opp til et nettverk av mer realistisk størrelse. MPC kan vise seg å være bedre egnet enn FHE for dette problemet, men forutsetter at partene beholder en aktiv forbindelse gjennom hele prosessen.

[7] `https://www.zama.ai/`

Når det gjelder sikkerhet, så vil naturligvis de returnerte rutene gi en del informasjon om hvor minene befinner seg. Garantien fra kryptografien er som alltid at man ikke lekker mer informasjon enn det man ville gjort hvis beregningene hadde gått gjennom en tiltrodd tredjepart. Om dette er godt nok må vurderes i hvert enkelt tilfelle.

4.5 Jamming fra ryggsekk

Overvåkning og mottiltak spiller en viktig rolle i moderne militære anvendelser, og ett element som inngår i dette er overvåkning av radarsignaturer slik vi beskrev i avsnitt 4.1. I tillegg til arbeidet vi presenterte i avsnitt 4.1 har vi også utforsket dette feltet som en mulig kandidat for bruk av skreddersydd dekryptering. Heller enn å se på flere aktører som ønsker å samarbeide om å koordinere flere databaser med radarsignaturer ser vi i dette scenarioet på et utplassert autonomt system («ryggsekken») som sitter og lytter etter radarsignaturer og tester det den finner opp mot en database den har tilgang til. Dersom den oppdager en signatur som finnes i databasen, starter den mottiltak, for eksempel ved å jamme. Ettersom radarsignaturene er sensitive og de i dette scenarioet er plassert på en autonom jammeryggsekk, ønsker vi å minimere hvor mye en motstander kan lære om signaturene gjennom å fange og inspisere ryggsekken.

Det er her vi bruker skreddersydd dekryptering. Jammeryggsekken har en kryptert database med signaturer som den i utgangspunktet ikke har tilgang til å kunne dekryptere. Derfor vil heller ingen som fanger opp ryggsekken kunne lese ut signaturene direkte. For å kunne gjøre oppgaven sin gir vi ryggsekken en skreddersydd dekrypteringsalgoritme som fungerer slik at den klarer å dekryptere den signaturen som er mest lik en gitt måling, men ikke noe mer.

Sammenligningsfunksjonen går gjennom listen av signaturer, og undersøker om målingen er innenfor en akseptabel feilmargin fra et senterpunkt. Da vil en motstander kun kunne observere resultatene av signaturberegningene, og ikke kunne hente ut signaturene direkte fra databasen. På den måten gjør vi det vanskeligere for motstanderen å lære hvilke signaturer jammeryggsekken ser etter.

4.5.1 Informasjonslekkasje fra jammeryggsekken

Dette delkapittelet er teknisk, og er tiltenkt særlig interesserte lesere.

Matematisk kan vi beskrive problemet med å slå opp en slik signatur på følgende måte: Radarparametre er $x \in \mathbb{R}_+^N$, databasesignatur $s \in \mathbb{R}_+^N$ og tillatt avvik $d \in \mathbb{R}_+^N$. Man kan da bruke avstandsmålet

$$f(x, s, d) = \begin{cases} 0 & \text{hvis } |x_j - s_j| < d_j \text{ for } j = 0, 1, \ldots, N-1 \\ 1 & \text{ellers} \end{cases}$$

Vi gjør skreddersydd dekryptering gjennom å gi jammeryggsekken tilgang til en funksjon som beregner den signaturen s som er slik at avstanden fra de observerte parameterne til s er minst mulig.

Selv om vi antar at krypteringen fungerer perfekt er det utfordringer med informasjonslekkasje i jammeryggsekkscenarioet. Informasjonslekkasjen vi beskriver her er også beskrevet i et tidligere arbeid [29], men vi gjengir det her fordi det illustrerer godt begrensningene som ligger i å gjøre sikre beregninger.

La oss betrakte en motstander $\mathcal{A}$ som kjenner sine egne radarsignaturer, og vil vite hvor god kjennskap vi har til dem. Anta at $\mathcal{A}$ vet om x slik at $f(x, s, d) = 0$, og ønsker å finne s.

La $x'_i = x_i + d_i$ for en valgt i, og $x'_j = x_j$ for $j \neq j$. Hvis $f(x', s, d) = 0$, så var $x_i < s_i$. Prøv i så fall på nytt med $x_i'' = x_i - d_i/2$. Dette er essensielt sett et binærsøk for hver komponent, og gjør at man kan få vilkårlig godt treff på s i logaritmisk tid.

Dette angrepet bruker kjennskapen til d, men er ikke avhengig av den. Man kan enkelt estimere d_i ved å variere x_i i begge retninger inntil hver av retningene gir $f(\tilde{x}, s, d) = 1$. Da kjenner vi ytterpunktene $s_i \pm d_i$, og kan enkelt utlede (s_i, d_i) fra disse.

En slik ryggsekk vil – selv under optimale forhold – altså ikke kunne ha særlig mye sikkerhet mot en motstander som leter etter bekreftelse og informasjon om dens egne radarsignaturer. Denne konklusjonen er uavhengig av om hvorvidt avvikene d holdes hemmelig eller ikke.

5 Konklusjon

Det er, etter vår vurdering, fortsatt for dårlig kjent at kryptografi kan brukes til mer enn å sikre kommunikasjonslinjer og lagrede data. Når sensitive data behandles på utrygge måter innebærer det nødvendigvis også en risiko, og vi er usikker på i hvilken grad dataeierne er bevisst på denne.

Både sikre flerpartsberegninger (MPC), homomorf kryptografi (FHE) og skreddersydd dekryptering (FE) gir muligheter til å utføre sikre beregninger, og vil kunne fungere som et risikoreduserende tiltak i behandlingen av sensitive data. Det er samtidig klart at disse teknikkene er på forskjellige modenhetsnivå: MPC er klart til bruk for mange anvendelser, selv om man må beregne mye tid på implementasjon og tilpassing. FHE er godt forstått, men begrenses av store chiffertekster og svært lange beregningstider. FE er foreløpig først og fremst et interessant tema for grunnforskning.

Vi må også anerkjenne at selv om kryptografien gir sterke muligheter her, så vil anvendelsene i seg selv kunne føre til at data lekker. Det ville imidlertid også vært det samme dersom man utførte oppgaven ved hjelp av en tiltrodd tredjepart. Det må altså alltid være en del av risikovurderingen når man vurderer hvordan sensitive data skal brukes. Det er utfordrende å gjøre analyse av slike datalekkasjer, og mer arbeid kreves på dette feltet.

Vi kan ikke løse alle problemer ved å blindt legge på sikre beregninger. Noen ganger finnes det nyanser som gjør beregningene mye vanskeligere, og noen ganger er det juridiske utfordringer. Disse bestemmelsene kan ha oppstått i vekselvirkning med at man har antatt at usikker behandling var den eneste muligheten, så det er mulig at vi ved hjelp av kryptografi kan oppnå formålet med behandlingen, men likevel være i strid med de vedtatte reglene. Av denne grunnen er det nødvendig med mer arbeid, både for å spre kunnskap og avdekke flere mulige anvendelser, men også av ikke-kryptografisk art for å klargjøre hvilke muligheter og begrensninger omkringliggende regulering gir.

Ordliste

angrep et generisk uttrykk for å få et system til å opptre på en hvilken som helst annen måte enn det protokollen eller definisjonen tilsier, eller at man kan knekke egenskaper på kortere tid enn det ville tatt å sjekke alle mulige nøkler.

homomorf kryptering (FHE, *fully homomorphic encryption*), en samling av kryptosystemer der dekrypteringsfunksjonen er *homomorf*, slik at f.eks. summen av to chiffertekster vil dekrypteres til summen av de krypterte verdiene; formålet er å utføre hele algoritmer på krypterte data.

homomorfi en matematisk funksjon mellom to algebraiske strukturer, og som bevarer strukturen, f.eks. potensreglene: $x^a \cdot x^b = x^{a+b}$; selv om multiplikasjon endres til addisjon beholdes den grunnleggende strukturen.

integritet en ønsket egenskap som sikrer at meldingen ikke har blitt endret siden den ble sendt; et kryptosystem bør *bevare* integriteten til meldingen, også mot en angriper.

klartekst se: *melding*.

konfidensialitet én av egenskapene et kryptosystem kan gi til meldingen, en garanti om at chifferteksten ikke lekker informasjon om det som er kryptert; det er flere forskjellige definisjoner, avhengig av hvor kraftige angrepstaktikker motstanderen har tilgang til.

melding en mengde informasjon som skal sendes fra Alice til Bob, før kryptering.

motstander den man skal forsvare systemet sitt mot; motstanderen er antatt å ha tilgang til all dokumentasjon og algoritmer, og vil som regel kunne kontrollere noen aktører og samordne disse.

MPC sikre flerpartsberegninger (MPC, *secure multiparty computations*), en protokoll der flere jevnbyrdige parter til sammen regner ut en funksjon, men uten at noen av partene lærer noe om verken inndata eller midlertidige verdier.

protokoll en samtale mellom to eller flere parter, og der alle partene må opptre nøyaktig i henhold til de reglene som er bestemt på forhånd; aktørene i protokollen skal normalt avbryte umiddelbart dersom de oppdager at andre ikke følger protokollen, og de som gjør det skal deretter regnes å være under motstanderens kontroll.

skreddersydd dekryptering (*functional encryption*), slik at en bruker kan få tilgang til et komplett datasett, men bare utstyres med nøkler som tillater dekryptering av en bestemt funksjon av datasettet.

Referanser

[1] Michel Abdalla, Florian Bourse, Angelo De Caro og David Pointcheval. Simple functional encryption schemes for inner products. I Jonathan Katz, redaktør, *PKC 2015*, bind 9020 av *LNCS*, side 733–751. Springer, Berlin, Heidelberg, mars/april 2015.

[2] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Hugo Pacheco, Vitor Pereira og Bernardo Portela. Enforcing ideal-world leakage bounds in real-world secret sharing MPC frameworks. I Steve Chong og Stephanie Delaune, redaktører, *CSF 2018 Computer Security Foundations Symposium*, side 132–146. IEEE Computer Society Press, 2018.

[3] Justis og beredskapsdepartementet. Forskrift om behandling av opplysninger i politiet og påtalemyndigheten (politiregisterforskriften), 2013. https://lovdata.no/dokument/SF/forskrift/2013-09-20-1097?q=politiregisterforskriften.

[4] Dan Bogdanov. *Sharemind: programmable secure computations with practical applications*. Doktorgradsoppgave, University of Tartu, 2013.

[5] Dan Bogdanov, Sven Laur og Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. I Sushil Jajodia og Javier López, redaktører, *ESORICS 2008*, bind 5283 av *LNCS*, side 192–206. Springer, Berlin, Heidelberg, oktober 2008.

[6] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach og Tomas Toft. Secure multiparty computation goes live. I Roger Dingledine og Philippe Golle, redaktører, *FC 2009*, bind 5628 av *LNCS*, side 325–343. Springer, Berlin, Heidelberg, februar 2009.

[7] Dan Boneh, Amit Sahai og Brent Waters. Functional encryption: Definitions and challenges. I Yuval Ishai, redaktør, *TCC 2011*, bind 6597 av *LNCS*, side 253–273. Springer, Berlin, Heidelberg, mars 2011.

[8] Florian Bourse, Rafaël del Pino, Michele Minelli og Hoeteck Wee. FHE circuit privacy almost for free. I Matthew Robshaw og Jonathan Katz, redaktører, *CRYPTO 2016, Part II*, bind 9815 av *LNCS*, side 62–89. Springer, Berlin, Heidelberg, august 2016.

[9] Ran Cohen og Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. I Palash Sarkar og Tetsu Iwata, redaktører, *ASIACRYPT 2014, Part II*, bind 8874 av *LNCS*, side 466–485. Springer, Berlin, Heidelberg, desember 2014.

[10] The PRIVATEER consortium. Privacy-first security enablers for 6G networks (PRIVATEER), 2023. https://www.privateer-project.eu.

[11] David Dobkin, Anita K. Jones og Richard J. Lipton. Secure databases: protection against user influence. *ACM Trans. Database Syst.*, 4(1):97–106, mars 1979.

[12] Cynthia Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, januar 2011.

[13] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai og Brent Waters. Attribute-based encryption for circuits from multilinear maps. I Ran Canetti og Juan A. Garay, redaktører, *CRYPTO 2013, Part II*, bind 8043 av *LNCS*, side 479–499. Springer, Berlin, Heidelberg, august 2013.

[14] Till Gehlhar, Felix Marx, Thomas Schneider, Ajith Suresh, Tobias Wehrle og Hossein Yalame. Safefl: Mpc-friendly framework for private and robust federated learning. I *2023 IEEE Security and Privacy Workshops (SPW)*, side 69–76. IEEE, 2023.

[15] Craig Gentry. Fully homomorphic encryption using ideal lattices. I Michael Mitzenmacher, redaktør, *41st ACM STOC*, side 169–178. ACM Press, mai/juni 2009.

[16] Oded Goldreich, Silvio Micali og Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. I Alfred Aho, redaktør, *19th ACM STOC*, side 218–229. ACM Press, mai 1987.

[17] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan og Nickolai Zeldovich. How to run Turing machines on encrypted data. I Ran Canetti og Juan A. Garay, redaktører, *CRYPTO 2013, Part II*, bind 8043 av *LNCS*, side 536–553. Springer, Berlin, Heidelberg, august 2013.

[18] Vipul Goyal, Omkant Pandey, Amit Sahai og Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. I Ari Juels, Rebecca N. Wright og Sabrina De Capitani di Vimercati, redaktører, *ACM CCS 2006*, side 89–98. ACM Press, oktober/november 2006. Available as Cryptology ePrint Archive Report 2006/309.

[19] Liina Kamm og Jan Willemson. Secure floating-point arithmetic and private satellite collision analysis. Cryptology ePrint Archive, Report 2013/850, 2013.

[20] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. I *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[21] Krishnaram Kenthapadi, Nina Mishra og Kobbi Nissim. Simulatable auditing. I *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, side 118–127, New York, NY, USA, 2005. Association for Computing Machinery.

[22] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima og Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. I Henri Gilbert, redaktør, *EUROCRYPT 2010*, bind 6110 av *LNCS*, side 62–91. Springer, Berlin, Heidelberg, mai/juni 2010.

[23] Yehuda Lindell. Secure multiparty computation (MPC). Cryptology ePrint Archive, Report 2020/300, 2020.

[24] Mohamad Mansouri, Melek Önen, Wafa Ben Jaballah og Mauro Conti. SoK: Secure aggregation based on cryptographic schemes for federated learning. *PoPETs*, 2023(1):140–157, januar 2023.

[25] Benjamin Hansen Mortensen. Private database search. Masteroppgave, Universitetet i Oslo, 2024.

[26] Benjamin Hansen Mortensen, Mathias Karsrud Nordal og Martin Strand. Matching radar signals and fingerprints with MPC. *IACR Communications in Cryptology*, 1(3), 2024.

[27] Amit Sahai og Brent R. Waters. Fuzzy identity-based encryption. I Ronald Cramer, redaktør, *EUROCRYPT 2005*, bind 3494 av *LNCS*, side 457–473. Springer, Berlin, Heidelberg, mai 2005.

[28] Adi Shamir. Identity-based cryptosystems and signature schemes. I G. R. Blakley og David Chaum, redaktører, *CRYPTO'84*, bind 196 av *LNCS*, side 47–53. Springer, Berlin, Heidelberg, august 1984.

[29] Martin Strand, Frode Johan Lillevold og Jan Henrik Wiik. Sikre beregninger ved hjelp av kryptografi. *Forsvarets forskningsinstitutt*, oktober 2022. FFI-eksternnotat 22/01492 (Unntatt offentlighet).

[30] Junichi Tomida. Unbounded quadratic functional encryption and more from pairings. I Carmit Hazay og Martijn Stam, redaktører, *EUROCRYPT 2023, Part III*, bind 14006 av *LNCS*, side 543–572. Springer, Cham, april 2023.

Om FFI

Forsvarets forskningsinstitutt ble etablert 11. april 1946. Instituttet er organisert som et forvaltningsorgan, med særskilte fullmakter underlagt Forsvarsdepartementet.

FFIs formål

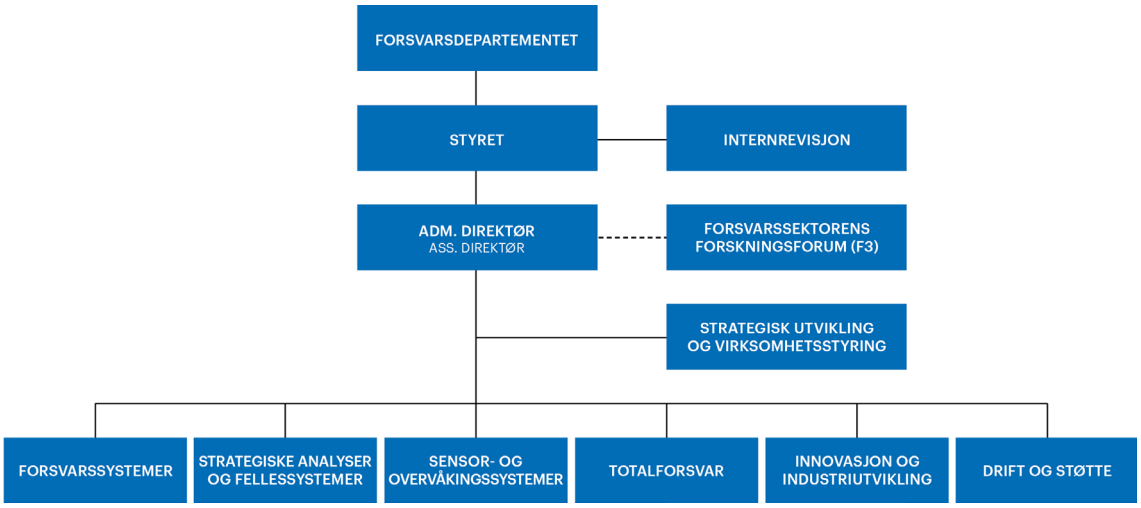
Forsvarets forskningsinstitutt er Forsvarets sentrale forskningsinstitusjon og har som formål å drive forskning og utvikling for Forsvarets behov. Videre er FFI rådgiver overfor Forsvarets strategiske ledelse. Spesielt skal instituttet følge opp trekk ved vitenskapelig og militærteknisk utvikling som kan påvirke forutsetningene for sikkerhetspolitikken eller forsvarsplanleggingen.

FFIs visjon

FFI gjør kunnskap og ideer til et effektivt forsvar.

FFIs verdier

Skapende, drivende, vidsynt og ansvarlig.



Forsvarets forskningsinstitutt (FFI)
Postboks 25
2027 Kjeller

Besøksadresse:
Kjeller: Instituttveien 20, Kjeller
Horten: Nedre vei 16, Karljohansvern, Horten

Telefon: 91 50 30 03
E-post: post@ffi.no
ffi.no

Norwegian Defence Research Establishment (FFI)
PO box 25
NO-2027 Kjeller
NORWAY

Visitor address:
Kjeller: Instituttveien 20, Kjeller
Horten: Nedre vei 16, Karljohansvern, Horten

Telephone: +47 91 50 30 03
E-mail: post@ffi.no
ffi.no/en