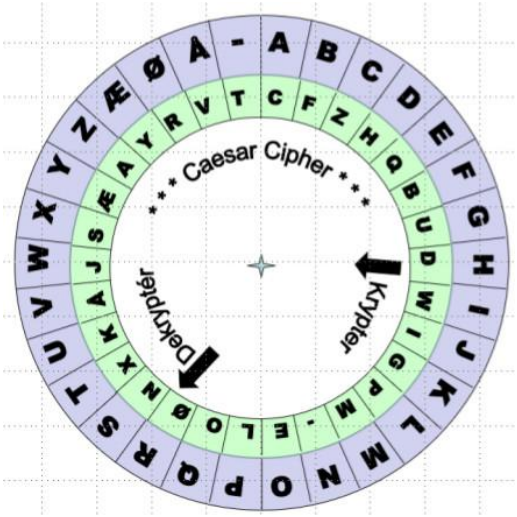
Universitetet i Oslo
IN2120 Informasjonssikkerhet
Høst 2025
Teoretiske oppgaver med løsningsforslag

Teori 2 (Del 2): Kryptografi



Oppgave 1: Cæsar-chiffer

Anta en substitusjons-chifferalgoritme som ligner Caesar-algorithmen, men der bokstavene i den innerste sirkelen er omstokket. Bokstavene kan ha en vilkårlig rekkefølge, der figuren til høyre viser et eksempel. Rekkefølgen av tegnene i det omstokkede alfabetet på den inder sirkelen utgjør nøkkelen. Det er til sammen 30 tegn, som består av 29 bokstaver og bindestrek '-'.

a. Gi et enkelt matematisk uttrykk for hvor mange forskjellige nøkler denne chifferalgoritmen har, og finn tallsvaret med en kalkulator, f.eks. online: https://www.calculatorsoup.com/calculators/discretemathematics/factorials.php
b. Hva er nøkkelstørrelsen for 30 tegn uttrykt i antall bits? Vurder om nøkkelstørrelsen er tilstrekkelig for å motstå uttømmende søk i hele nøkkelrommet.
c. Hva hadde nøkkelstørrelsen vært om alfabetet hadde hatt 34 tegn? Er nøkkelstørrelsen tilstrekkelig for å motstå uttømmende søk gjennom hele nøkkelrommet? Forklar svaret.
d. Kan algorithmen motstå statistisk kryptanalyse? Forklar svaret.
e. Forklar kort hvordan man kan kryptanalysere en chiffertekst som er kryptert med denne chifferalgorithmen.

Løsningsforslag

a. Antall nøkler beregnes med funksjonen «fakultet» (eng. factorial) som tar heltall som argument. Funksjonen fakultet(30) skrives som 30! som uttrykkes som:

$$30! = \prod_{n=1}^{30}(n) = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot 30 = 2.65 \cdot 10^{32}$$

b. Nøkkelstørrelsen S i bits beregnes med logaritme base 2 av antall nøkler.
$S = \log_2(2.65 \cdot 10^{32}) = \log_2(2.65) + (\log_2(10) \times 32) = 1.4 + (3.32 \times 32) \approx 108$ bit.
Nøkkelstørrelsen er bare 20 bit mindre enn 128, som er minste mulige nøkkelstørrelse i AES, dvs. at nøkkelrommet er ca. 1/1.000.000 av nøkkelrommet til AES 128. Denne varianten av Ceasr chifferalgoritmen har altså en rimelig god nøkkelstørrelse og vil være resistent mot uttømmende nøkkelsøk i de fleste situasjoner.

c. Med 34 tegn er antall nøkler 34!, dvs. fakultet(34) som uttrykkes som:

$$34! = \prod_{n=1}^{34}(n) = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot 34 = 2.95 \cdot 10^{38}$$

Nøkkelstørrelsen er dermed S = $\log_2(2.95 \cdot 10^{38}) = 1.5 + (3.32 \times 38) \approx 128$ bit. Dette nøkkelrommet er altså det samme som i AES 128, som gir sterk beskyttelse mot uttømmende søk, selv med den kraftigste computer.

d. Algoritmen har ikke egenskapen at statistiske mønster i klarteksten blir skjult i chifferteksten. Det vil derfor være trivielt å finne nøkkelalfabetet med enkel kryptanalyse av chifferteksten.
e. Analysen går utpå å telle hvor mange det er av hvert tegn i chifferteksten. Gitt at det er engelsk tekst, vil tegnet som forekommer oftest antageligvis tilsvare boktaven 'E' i klarteksten. Det betyr at bokstaven som oftest forekommer i chifferteksten sannsynligvis representerer 'E'. Deretter finner man bokstaven som forekommer nest oftest osv. Man prøver seg fram til man får på plass rekkefølgen av alle tegnene i nøkkelalfabetet. Deretter kan chifferteksten dekrypteres.

Oppgave 2: Digital signatur

Alice ønsker å sende melding M med digital signatur $S(M)$ til Bob. De har hverandres offentlige nøkler, og har blitt enige om en kryptografisk hash-funksjon Hash og en signatur-algoritme som opererer i signeringsmodus Sig (tilsvarende dekrypteringsmodus D) eller i verifiseringsmodus Ver (tilsvarer krypteringsmodus E).

a. Beskriv trinnene Alice må følge for å sende M.
b. Beskriv trinnene som mottaker Bob må følge for å verifisere autentisiteten av M.
c. Forklar hvordan den digitale signaturen beviser for Bob at den mottatte meldingen er autentisk, og forklar hvordan også enhver tredjepart, ikke bare Bob, kan bli overbevist om at meldingen er autentisk. Hva kalles egenskapen?
d. Hvilke plausible grunner kan avsender Alice gi for å nekte å ha sendt den signert meldingen selv om den har hennes signatur? Hva sier det om begrepet 'ubenektelighet'?
e. Diskuter den semantiske tolkningen av "digitalt signert melding", som kan bety: I) at jeg Alice er enig i innholdet av meldingen, eller II) Alice sendte meldingen uten nødvendigvis å være enige i innholdet?

Løsningsforslag

a. Generering av digital signatur utført av Alice:
   i. Alice klargjør melding M.
   ii. Alice bruker hash-funksjon Hash å produsere hash-verdi $H(M)$.
   iii. Alice bruker sin private nøkkel $K_{Priv}(A)$ med den asymmetriske algoritmen i signaturmodus Sig for å produsere signatur $S(M) = \text{Sig}(H(M), K_{Priv}(A))$.
   iv. Alice sender melding M og signatur $S(M)$ til Bob, sammen med sin identitet og spesifisering av algoritmene hun brukte.

b. Verifisering av digital signatur utført av Bob:
   i. Bob mottar melding M' (merket som M' og ikke som M, fordi fra Bobs synspunkt er meldingens avsender fortsatt uviss), samt signatur $S(M)$.
   ii. Bob bruker hashfunksjon Hash på M' for å produsere hashverdi $H(M')$.
   iii. Bob regenererer hash $H(M)$ fra signatur $S(M)$ ved hjelp av Alices offentlige nøkkel $K_{Pub}(A)$ ved å bruke den asymmetrisk algoritmen i verifiseringsmodus Ver for å produsere $H(M) = \text{Ver}(S(M), K_{Pub}(A))$.
   iv. Bob sjekker $H(M) =? H(M')$. Hvis det stemmer er signaturen $S(M)$ gyldig, som betyr at M' er autentisk, dvs. at $M = M'$ og Bob er overbevist om at Alice sendte M. Hvis det ikke stemmer er $S(M)$ ugyldig, slik at $M \neq M'$. Bob vet

derfor ikke hvem som sendte den mottatte meldingen *M'*. Han kan da bestemme seg for å avvise meldingen, eller bruke den selv om dens opprinnelsen er uviss.

c. Grunnen til at Bob, og faktisk enhver som har Alice sin offentlige nøkkel $K_{Pub}(A)$, kan være overbevist om at Alice sendte *M* er at bare Alice kunne ha generert meldingens digital signatur $S(M)$ fordi det antas at bare hun kjenner og er istand til å bruke den private nøkkelen $K_{Priv}(A)$. Denne form for autentisering kalles ubenektelig (eng. non-repudiable), fordi Alice i teorien ikke kan benekte å ha signert og sendt meldingen. I praksis fins det likevel grunner for at Alice kan benekte å ha sendt meldingen, se oppgave d) nedenfor.
d. Avsenderen kan hevde at den private signaturnøkkelen ble stjålet. Alice kan hevde at hun ble tvunget med makt til å signere og sende meldingen. Begrepet 'ubenektelig' er derfor ikke absolutt, men relativt. Begrepet 'ubenektelig' kan derfor være misvisende. Der er iallefall temmelig pretensiøst.
e. Den semantiske tolkningen av en digital signatur er i utgangspunktet tvetydig. Det kan f. eks bety at
   i. Alice er enig i innholdet av meldingen.
   ii. Alice sendte meldingen, men ikke nødvendigvis at hun er enig i innholdet

Når det brukes papir og penn er tolkning (a) den vanligste. I digital kommunikasjon er det ofte vanlig å anta tolking (b), der meldingen er et resultat av en systemtransaksjon der det ikke nødvendigvis antas noen spesifikk mening som avsender er enig i.

Oppgave 3: Strømchiffer

Anta at en binær additiv strømchiffer-algoritme (som bruker en pseudotilfeldig nøkkelstreng eller en engangsnøkkel (One-Time-Pad)) har blitt brukt til å kryptere en elektronisk pengetransaksjon. Anta at det ikke brukes andre kryptografiske mekanismer. Forklar hvordan en angriper kan endre overføringsbeløpet uten å vite noe om nøkkelen som brukes (du kan anta at angriperen vet formatet på klartekstmelding som brukes ved overføring av pengene.)

Løsningsforslag

Den delen av meldingen der beløpet oppgis har samme posisjon i chifferteksten som i klarteksten. Angriperen kan derfor endre noen bit i posisjonen der beløpet står slik at det endrer beløpet som overføres. Generelt er det ikke mulig for angriperen å vite om endringen vil øke eller redusere beløpet. I praksis kan angriperen vite at beløpet sannsynligvis vil være lite og derfor bare endrer sifrene i de høye binærposisjonene. Dette illustrerer at en binær strømchiffer ikke gir meldingsintegritet selv om det gir perfekt konfidensialitet hvis en ekte tilfeldig engangsnøkkel brukes.

Oppgave 4: Fremoverhemmelighold

Kryptering av data med asymmetriske algoritmer er upraktisk fordi det gir for stor belastning med beregning. I praksis brukes en kombinasjon av både symmetrisk og asymmetrisk kryptering, der en symmetrisk nøkkel utveksles mellom avsender og mottager ved hjelp av en asymmetrisk metode, og den symmetriske nøkkelen benyttes for selve kryteringen av data.

a. Hva er fremoverhemmelighold (Forward Secrecy) (noen gang kalt perfekt fremoverhemmelighold) for sikkerhetsprotokoller?

b. Anta at Alice og Bob etablerer økter der de krypterer data med en hemmelig symmetrisk øktnøkkel som Alice hver gang sender til Bob kryptert med Bobs offentlige nøkkel. Gir dette fremoverhemmelighold? Forklar hvorfor/hvorfor ikke.
c. Hva er en typisk metode for å oppnå fremoverhemmelighold? Nevn en prominent sikkerhetsprotokoll som støtter fremoverhemmelighold.

Løsningsforslag

a. Fremoverhemmelighold betyr at tidligere økter ikke skal kunne dekrypteres av en angriper som klarer å få tak i en langtids privat nøkkel en gang i fremtiden.
b. Denne protokollen gir ikke fremoverhemmelighold, fordi hvis en angriper får tak i Bobs private nøkkel (som er en langtidsnøkkel) vil angriperen kunne dekryptere alle tidligere økter ved først å dekryptere de hemmelige øktnøklene, og deretter benytte øktnøklene til å dekryptere data.
c. En typisk metode for å oppnå fremoverhemmelighold er å benytte Diffie-Hellman (DH) nøkkelutveksling av symmetriske hemmelige nøkler, der de hemmelige nøklene signeres digitalt av en eller begge partene (Alice og/eller Bob). Hvis de private signeringsnøklene stjeles av angripere en gang i fremtiden vil de likevel ikke kunne dekryptere tidligere økter kryptert med symmetriske nøkler. Det er fordi symmetriske nøklene etablert med DH ikke blir utvekslet i kryptert form. Dette prinsippet benyttes i TLS 1.3, som derfor gir fremoverhemmelighold.

Oppgave 5

Diffie-Hellman-algoritmen for nøkkelgenerering og utveksling lar to parter opprette en felles hemmelig nøkkel over en usikker kanal.

a. Forklar med formell notasjon trinnene i Diffie-Key-algoritmen.
b. Forklar hvorfor Diffie-Hellman-algoritmen i seg selv ikke gir gjensidig autentisering av partene, og hva som kan gjøres i tillegg for å oppnå autentisitet.

Løsningsforslag

a. Partene deler en base g og en modulo m. La a og b være de hemmelige nøklene til partene A og B hhv. Deretter:
   • A → B: g^a (modulo m)
   • B → b: g^b (modulo m)
   • *A* beregner $(g^b)^a = g^{ab}$ (modulo m)
   • *B* beregner $(g^a)^b = g^{ab}$ (modulo m)
   A og *B* har nå den felles symmetriske nøkkelen g^{ab}

b. Diffie-Hellman algorithmen gir ingen autentisering av meldingene som utveksles fordi informasjon om identitet ikke inngår i protokollen. Derfor har Alice ingen visshet for at hun kjører protokollen med Bob og Bob har ingen visshet om at han kjører protokollen med Alice. Autentisering må gjøres separat, f.eks. med digital signatur på den utvekslede hemmelige nøkkelen.

Oppgave 6: Symmetrisk kryptering

Alice ønsker å sende melding *M* til Bob, uten at Eve observere den. Alice og Bob har blitt enige om å bruke en symmetrisk kryptoalgoritme som kan brukes til kryptering med funskjonen E eller dekryptering med funskjonen D. Nøkkelutveksling er allerede gjort, slik at de har felles nøkkel *K* for den spesifikke krypteringsalgoritmen. De er også enige om å bruke en sikker krypteringsmodus for bruk av algoritmen (f.eks. CTR-modus), men detaljer om krypteringsmodus er uvesentlig for denne oppgaven.

a. Beskriv trinnene Alice må følge for å kryptere *M* og sende chifferteksten til Bob.
b. Beskriv trinnene som Bob må følge for å dekryptere den mottatte chiffertekst *C*.

Løsningsforslag

a. Kryptering:

i) Alice klargjør melding *M*.
II) Alice krypterer meldingen ved hjelp av den symmetriske krypto-algoritmen med krypteringsfunksjon E og nøkkel *K*, for å produsere chiffertekst *C*, der $C = E(M, K)$.
III) Alice sender chiffertekst *C* til Bob.

b. Dekryptering:

i) Bob mottar chiffertekst *C*.
II) Bob dekrypterer chifferteksten ved hjelp av den symmetriske krypto-algoritmen med dekrypteringsfunksjon D og nøkkel *K*, for å gjenopprette *M*, der $M = D(C, K)$.
III) Bob leser/tolker melding *M*.

Oppgave 7: Hashfunksjoner

Hashfunksjoner brukes ofte til å verifisere meldingsintegritet.

a. Oppgi fire fundamentale krav til kryptografiske hash-funksjoner.
b. Hva er forskjellen mellom de to variantene av kollisjonsresistens?
c. Bruk Internett til å finne en SHA-2 demo webside. Du finner en interaktiv webside laget av Geraint Luff som kan finnes på: http://geraintluff.GitHub.io/sha256/
   Undersøk hash-funksjons egenskaper ved å beregne SHA-2 hash for følgende:
   (i) ta ut $100 fra min konto
   (II) ta ut $1000 fra min konto
   (III) ta ut $100 fra din konto
   (IV) (du kan prøve å lage hashe for både lengre og kortere meldinger)

Løsningsforslag

a. De fire fundamentale egenskaper hashfunksjoner må ha er:
   - 1: lett å beregne $H(M)$.
   - 2: hash-verdien har alltid samme lengde uansett størrelse på inndata
   - 3: enveisfunksjon, som betyr at selv om $H(M)$ er gitt, så er det beregningsmessig umulig å finne melding *M*
   - 4: kollisjonsresistent, som betyr at det beregningsmessig er umulig å finne M_1 og M_2 slik at $H(M_1) = H(M_2)$

b. Svak og sterk kollisjonsresistens:
   i. Svak: Gitt én bestemt M_1 skal det være beregningsmessig umulig å finne M_2 slik at $H(M_1) = H(M_2)$.
   ii. Sterk: Det være beregningsmessig umulig å finne et vilkårlig par med datasett M_1 og M_2 slik at $H(M_1) = H(M_2)$.

c. Du ser at selv om to inndata-meldinger er nesten like, genereres hash-verdier som er svært ulike. Merk at uansett størrelse på inndata generer SHA-2 (256) alltid hash-verdier med 256 bits, som er det samme som 64 hexadesimale sifre.

Oppgave 8: Meldingsautentisering

Alice ønsker å sende en melding *M* med en meldingsautentiseringskode *MAC*(*M*) til Bob. Alice og Bob deler en hemmelig nøkkel *K*, og har blitt enige om å bruke en bestemt nøkkelstyrt hashfunksjon Hash(*M*, *K*) som tar inndataparametere *M* og *K* for å produsere *MAC*(*M*).

a. Beskriv trinnene Alice må følge for å sende *M* med meldingsautentisering.
b. Beskriv trinnene som mottaker Bob må følge for å verifisere autentisiteten til *M*.
c. Forklar hvordan en *MAC* beviser for Bob at den mottatt melding er autentisk, og hvorfor Bob **ikke er istand** til å bevise overfor en tredjepart at meldingen er autentisk.

Løsningsforslag

a. MAC-generering utført av Alice:
   i. Alice klargjør melding *M*.
   ii. Alice bruker en nøkkelstyrt hashfunksjon med inndataparametere *M* og *K* for å produsere *MAC*(*M*) = Hash(*M*, *K*).
   iii. Alice sender melding *M* og *MAC*(*M*) til Bob, sammen med sin identitet og spesifisering av hvilken MAC-funksjon (nøkkelstyrt hashfunksjon) hun brukte.

b. MAC-verifisering utført av Bob:
   i. Bob mottar melding *M'* (merket som *M'*, og ikke som *M*, fordi fra Bobs synspunkt er meldingens avsender fortsatt uviss), så vel som *MAC*(*M*).
   ii. Bob anvender den nøkkelstyrte hashfunksjonen på *M'* for å produsere *MAC*(*M'*) = Hash(*M'*,*k*).
   iii. Bob sjekker om *MAC*(*M*) =? *MAC*(*M'*). Hvis det stemmer er *MAC*(*M*) gyldig, som beviser at *M'* er autentisk, altså at *M* = *M'*. Bob er derfor overbevist om at Alice virkelig er avsender av meldingen *M*. Hvis det ikke stemmer er meldingsautentiserings-koden *MAC*(*M*) ugyldig, noe som betyr at mottatte melding *M'* ikke er autentisk. Bob vet derfor ikke hvem som sendte melding *M'*. Han kan da bestemme seg for å avvise meldingen, eller alternativt kan han bruke den selv om dens opprinnelse er uviss.

c. Bob er overbevist om at Alice sendte melding *M* når bare han og Alice kjenner den hemmelige nøkkelen *K*, og fordi han vet at han ikke sende meldingen selv. Men Bob er ikke i stand til å overbevise noen andre om at Alice sendte meldingen, fordi fra perspektivet til en tredjepart er det mulig at Bob kunne ha sendt meldingen selv.

Oppgave 9: Bruk av kryptografi

a. For hvilke datatilstander (lagring, overføring, prosessering) kan kryptografi brukes til å beskytte informasjon? Hvordan kan kryptografi beskytte data under prosessering?
b. Hvilke sikkerhetsmål/tjenester kan støttes av kryptografi?

Løsningsforslag

a. Kryptografi kan brukes til å beskytte informasjon når den overføres via ubeskyttede nettverk eller lagres på steder med utilstrekkelig fysisk eller logisk tilgangskontroll. Såkalt homomorf kryptografi kan brukes til å beskytte data under prosessering i en mikroprosessor, dvs. at data holdes kryptert hele tiden. Problemet med homomorf kryptorafi er at det bruker svært mye prosesseringskraft.
b. Kryptografi kan gi konfidensialitet, integritet og autentisitet (inkludert uavviselighet) av informasjon.

Oppgave 10: Postkvantekrypto

Kvantecomputing har potensiale til å knekke de fleste standardiserte asymmetriske kryptografiske algoritmer. Kvanteresistente kryptoalgoritmer kalles PQC (Post-Quantum Crypto) fordi de skal være sikre selv etter at store kvantecomputere er tilgjengelige på markedet.

a. Hvilke tre PQC-algoritmer ble publisert som utkast til NIST-standarder i august 2024?
b. Hvilken type nye PQC-algoritmer ønsker NIST fremdeles å velge?

Løsningsforslag

a. De tre utkastene til NIST-standarder for PQC-algoritmene er:
   - *Draft FIPS 203 (CRYSTALS-Kyber)* for asymmetrisk kryptering (lattice-basert)
   - *Draft FIPS 204 (CRYSTALS-Dilithium)* for DigSig (lattice-basert)
   - *Draft FIPS 205 (SPHINCS+)* for DigSig (hash-basert)
b. NIST ønsker forslag til nye DigSig-algorithmer som **ikke** er basert på lattice eller hash. Dette for å ha variasjon i det matematiske grunnlaget i tilfellet matematisk gjennombrudd som gjør at alle PQC-algoritmer med samme matematikk, knekkes.