IN1020 - Nettverkskommuniksjon og sikkerhet: Løsningsforslag oppgaver gruppetime 4 (Uke 38)

1. Praktisk anvendelse av kryptografi

   (a) Sikre trygg lagring av data på utrygge lagringsenheter, sikre trygg overføring av data i f.eks åpne datanettverk.

   (b) Kryptografi kan som nevnt i a) bidra til å sikre konfidensialitet. I tillegg kan integritet til data sjekkes ved å benytte sjekksumalgoritmer (hash-funksjoner). Bruk av symmetrisk og asymmetrisk kryptering kan brukes for å sikre autentisitet på data/meldinger sendt og/eller autentisitet til mottaker (når du vet hvem som kjenner den hemmelige/private nøkkelen), mens bruk av asymmetrisk kryptering for digital signering også kan sikre uavviselighet.

   (c) Er disk på PC kryptert? Hva med mobiltelefon? Nettverkskommunikasjon, inkludert over wifi til trådløse aksesspunkter? Sender du kryptert epost, evt. krypterte dokumenter i epost? Kommunikasjon til nettsider? Mobilapper? Signerer du noe digitalt? Lagring av passord? Sjekker du eller datamaskinen din integritet til data/filer/nedlastet programvare/meldinger over nett?

2. Nøkkel og nøkkeldistribusjon

   (a) En kryptografisk nøkkel er rett og slett et stykke informasjon, som brukes som parameter til krypteringsalgoritmen (chifferen) man benytter, og som er med på å bestemme resultatet (altså cipherteksten) når klartekst krypteres med en kryptografisk algoritme. Algoritmene må antas å være kjent, mens kryptografiske nøkler (men unntak av offentlige nøkler) skal holdes hemmelige for å unngå at uvedkommende får tak i dem og klarer å dektyptere ciphertekst de ikke skal ha tilgang til.

   (b) *Typer av nøkler:* Symmetrisk hemmelig nøkkel, asymmetrisk privat nøkkel og asymmetrisk offentlig nøkkel.

   Symmetriske hemmelige nøkler og **asymmetriske private nøkler** må alltid holdes konfidensielle (dvs. beskyttet mot uautorisert eksponering) og må ha integritet/autentisitet (dvs. beskyttet mot forfalskning/uautorisert modifisering).

   Asymmetriske offentlige nøkler må ha integritet/autentisitet (dvs. beskyttet mot forfalskning/uautorisert modifisering), men trenger ikke konfidensialitet.

   (c) Formålet med *offentlig-nøkkel sertifikater* og *offentlig-nøkkel infrastruktur (PKI)* er å legge til rette for autentisering av entiteter på internett, samt kryptert overføring av informasjon mellom enheter. PKI skal sikre autentiske offentlige nøkler. PKI er et rammeverk for å legge til rette for å binde en offentlig nøkkel til en navngitt enhet, og denne bindingen kan bekreftes av en betrodd myndighet som utsteder et *sertifikat*. Et sertifikat forteller/bekrefter følgende: "Offentlig nøkkel K eies av X".

3. Symmetrisk kryptering

   (a) Alle kan lytte til radiosignaler så hvis nøkkelen blir lekket eller koden knekkes kan det være svært alvorlig.

   (b) Ikke alle har maskiner som kan spille av DVD-er eller kjøre CD-er. De må også transporteres på en måte. Posten er ikke 100 prosent sikkert og dersom du transporterer den selv er det jo nesten litt unødvendig å kryptere.

(c) USB-er er små og kan lett mistes. De kan også holde på mye data, som gjør det ekstra ille å miste dem.

(d) Ofte kan man ikke kryptere selve e-posten. Så mottaker kan se avsender, evt. andre mottakere, emnefelt, all tekst som er inne i selve e-posten og navnene på filer man sender med. Det er kun vedlegg (filer) som gjerne krypteres. Dersom du sender e-posten feil eller noen klarer å få tilgang til e-posten er det altså mye de kan lese ut av denne infoen.

Å overlevere kryptografiske nøkler på en trygg og sikker måte er heller ikke uproblematisk, men det bør alltid skje over en annen kanal enn den man bruker for å overlevere de krypterte dataene.

4. Asymmetrisk kryptering

   I asymmetrisk kryptering har man et nøkkelpar, der én nøkkel brukes til å kryptere og én brukes til å dekryptere. Som regel er en av disse hemmelige, mens den andre er offentlig. Hvem har den offentlige nøkkelen og hvem har den hemmelige nøkkelen i disse scenarioene?

   (a) Du bruker en offentlig nøkkel for å kryptere. Kollegaen bruker den tilhørende private nøkkelen for å dekryptere. Dette sikrer konfidensialitet for dokumentet mellom deg og kollegaen din, fordi ingen andre enn den som kjenner til den private nøkkelen kan dekryptere dokumentet.

   (b) Du bruker en privat nøkkel for å kryptere og kollegaen din bruker den tilhørende offentlige nøkkelen for å dekryptere (også kalt å verifisere avsender). Dette sikrer autentisitet og uavviselighet så lenge det kun er en aktør (du) som kjenner den private nøkkelen, og dermed har man en visshet om at det er du som har signert dokumentet.

   (c) Er besvart i løsningsforslag a) og b).

5. Hash-algoritmer (sjekksumalgoritmer)

   - Lett å beregne: Enkelt å regne ut sjekksum for en gitt melding
   - Enveis: Ikke mulig (vanskelig) å finne en melding for en gitt sjekksum
   - Kollisjonsresistent: Ikke mulig (vanskelig) å finne to ulike meldinger med samme sjekksum, og Ikke mulig (vanskelig) å endre en melding uten at sjekksummen også blir en annen.

6. Kommunikasjon i datanettverk

   (a) Med http som protokoll går all kommunikasjon mellom Bruker og Server i klartekst, og hvem som helst (som har teknisk innsikt) kan lytte til kommunikasjonen.

   (b) Med sikker forbindelse mellom Bruker og WiFI-ruter forhindres Lytter1 i å fange opp kommunikasjonen. Alle de andre aktørene kan fremdeles se data i klartekst.

   (c) Kun Bruker og Server ser data de sender seg imellom i klartekst.

   (d) Kun Bruker og Server ser data de sender seg imellom i klartekst.

   (e) Hvis man bruker et usikret trådløst nettverk, kan Lytter1 (og alle de andre) se headere i datapakkene som sendes, f.eks. at Bruker har en «dialog» med serveren devilry.ifi.uio.no. Men kryptert trådløsforbindelse krypteres også headerne mellom Bruker og WiFi-ruter, og Lytter1 forhindres i å fange opp noe som helst fornuftig.

7. Eksamensoppgave 3.5 og 3.6 fra H2023. For løsningsforslag se sensurveiledningen

8. Case-oppgave (Hash-kryptering/sjekksumalgoritme)

   (a) Dersom to brukere har samme passord, har de den samme hashen. Det finnes databaser som inneholder hasher til vanlige passord, så hvis brukerne deres benytter dårlige passord kan trusselaktører finne disses passord-hashene i slike databaser, og dermed finne en match. Teamet ditt burde kanskje ikke prøve å lagre brukeres passord selv? En mulighet er å benytte en eller annen tilbyder av autentiseringstjenester isteden (google, facebook, Feide, BankID?).

(b) Her ivaretas faktisk konfidensialitet ved at passord ikke lagres i klartekst men kun som en hash-verdi, og dermed ikke kan leses av uvedkommende.

(c) Hash algoritmer "fordøyer" data ned til en streng som har fast lengde (ofte 16, 32 eller 64 bit og i heksadesimal).

9. Forbindelsesstrategier

   Pull: En klient initierer en forbindelse til en tjener (innholdsleverandør) med forespørsel om en tjeneste. Eks. HTTP eller FTP.

   Push: En tjener "dytter"innhold til en klient når den har noe relevant å levere. Initiert av innholdsleverandøren. Eks "Pushmeldinger"på mobil, HTTP/2 Push.

   Publish-subscribe: Mange klienter abonnerer på en tjeneste. Tjeneren dytter innhold til abonnentene nå den har noe interessant å levere. Eks. Rich Site Summary (RSS)

10. Aksessmodeller

    Klient-tjener: Typisk pull-tjeneste. Tjeneren har en spesiell rolle der den leverer en tjeneste til mange potensielle brukere. Eks. HTTP eller IMAP mail.

    P2P: alle noder i nettverket er likeverdige og samarbeider om å levere en tjeneste. Eksempler: Bittorrent, The Onion Router (TOR), Bitcoin

11. Content Delivery Network - CDN

    (a) Innhold som trenger å leveres til svært mange brukere over hele verden kopieres til mange maskiner som ligger fysisk nær sluttbrukerne. Dette gjør at kildemaskinen ikke blir overbelastet og sparer kapasitet på nettverket.

    (b) Svar:

    - Å ha innholdet fysisk nærmere brukeren reduserer round trip time (RTT) og gjør det derfor raskere å aksessere,
    - Ved å cache innholdet nær brukeren sparer man trafikk over backbone-nettet,
    - Ved å distribuere innholdet, avlaster man tjeneren som leverer dette innholdet, noe som gir en mer skalerbar tjeneste.

12. Domain Name System - DNS

    (a) Et DNS-navn er 2 eller flere ord med punktum imellom som man bruker for å koble opp mot maskiner på internett i stedet for å huske på IP-adressen til maskinen. Vi benytter dette da IP-adresser ikke er særlig menneskevennlig (vanskelig å huske). Det er også flere tjenester som bedre lar seg løse ved å koble navn til IP-adressen, som f.eks. aliasing (fler DNS-navn til én IP) og round-robin DNS (samme DNS-navn går på rundgang mellom flere IP-adresser).

    (b) De holder oversikten over hvilke maskiner som har ansvaret for hvert TLD (Top Level Domain). TLDer er det høyeste nivået i DNS-hierarkiet, ordet etter det siste punktumet (f.eks .org, .net, .com, .no).

13. Video Streaming (DASH)

    En måte å strømme video ved hjelp av HTTP-tjenere. Videoen "klippes" opp i små uavhengige segmenter som kan kodes i flere kvalitetsnivåer. Dette gjør at en klient kan spørre etter høy kvalitet i perioder når den opplever gode nettverksforhold og redusere kvaliteten når forholdene er dårligere.

14. Hyper Text Transfer Protocol - HTTP

    (a) En TCP-forbindelse opprettes med et 3-veis håndtrykk, om SSL skal brukes, forhandles dette frem ved å først bruke asymmetrisk kryptering til å utveksle en felles nøkkel for videre

symmetrisk kryptering av forbindelsen. Klienten sender så en GETforespørsel på det aktuelle dokumentet til HTTP-tjeneren. Tjeneren leverer et "200 OKsvar tilbake sammen med det aktuelle dokumentet. Om hjemmesiden inneholder dokumenter fra flere kilder opprettes tilsvarende forbindelser for å hente resten av det aktuelle innholdet.

(b) HTTP bruker TCP og har port 80 som standard. *Bonusoppgave:* Hvilken port er standard for kryptert (HTTPS) oppkobling? HTTPS har port 443 som standard.

(c) En ikke-persistent forbindelse blir lukket så snart et svar er levert. Påfølgende forespørsler må åpne en ny forbindelse. En persistent forbindelse vil gjenbruke den samme forbindelsen til flere forespørsler.

15. Sjekksumalgoritme (hash-funksjon) – praktisk oppgave

(a) Nei, sjekksummen blir en annen når fila endres. Dette er en grunnleggende egenskap ved sjekksumalgoritmer. Men observer gjerne at sjekksummen blir den samme når du bruker programmet sha256sum på samme fil flere ganger.

(b) Man kan sammenligne sjekksum av en datafil opp mot en kjent sjekksum av denne fila. Er den lik? isåfall er fila ikke endret mellom første og andregangskjøring av sjekksum.

(c) Sjekksumalgoritmer brukes ofte ved deling av programvare (f.eks. nedlasting fra nett). Tilbyderen av programvare genererer en sjekksum av orignalen (ofte et filarkiv, f.eks. .zip-fil), som publiseres sammen med programvaren. Brukere som laster ned programvare kan enkelt selv kjøre en sjekksumalgoritme etter nedlasting, og kan med det finne ut om programvaren er det den utgir seg for å være, eller er den kanskje endret etter at sjekksummen ble generert? Vil også kunne avdekke hvorvidt en nedlasting har gått greit eller ikke (pakketap?). Helt konkret eksempel: https://www.postgresql.org/ftp/source/v17.6/, nedlasting av kildekode PostgreSQL databaseserver. Her finner du både programvaren og tilhørende sjekksummer for både MD5 og SHA256.

16. SSL – praktisk oppgave

Praktisk oppgave uten løsningsforslag.