Security Protocols, Applications and Standards

Hva jeg må kunne etter denne forelesningen:

- Have knowledge of what Key exchange algorithms are used for.
- Describe the Diffie–Hellman key exchange algorithm.
- Have knowledge of Public Key Infrastructure (PKI) and why it's needed.
- Understand and explain the difference between Public Key Cryptography and Public Key Infrastructure (PKI).
- Understand the concept of Trusted Third Parties and Certificate Authorities.
- Have knowledge of the Certificate Life Cycle and the different parties involved.
- Describe the process of obtaining an X.509 certificate.
- Understand and explain the process of revoking a certificate.
- Understand and explain Pretty Good Privacy (PGP) and how it's different from Certificate based PKI.
- Explain PGP Web of Trust Model.

KRYPTOGRAFISKE MEKANISMER

De kryptografiske mekanismene vi har lært så langt (symmetrisk og asymmetrisk kryptering, digital signatur og melding autentiserings koder (MACs), kan være relativt enkle å oppnå for å oppnå de grunnleggende sikkerhets målene:

- Konfidensialitet (med krypteringsalgoritmer)
- Integritet (med MAC eller digitale signaturer)
- Autentisitet (med MAC eller digitale signaturer)
- Non-repudiation (med digitale signaturer)

Disse kryptografiske mekanismene tar utgangspunkt i at nøklene er distribuert på et trygt vis mellom to parter involvert som Alice og Bob.

NØKKEL UTVEKSLING OG ETABLERING

Hvis nøkler skal utveksles trygt i for eksempel et symmetrisk kryptosystem, så trenger vi en mekanisme som hjelper oss gjøre det. I mange sikkerhetssystemer er det ønsket å bruke kryptografiske nøkler som kun er gyldige for en begrenset tidsperiode for eksempel en session i internettet. Nøkkel utveksling og etablering muliggjør to systemer til å generere en hemmelig symmetrisk nøkkel som kan da brukes til å kryptere data for en begrenset tidsperiode.

SESSION KEYS

Motivasjonen for å bruke session nøkler:

- For å begrense tilgjengelig krypteringstekst for kryptoanalyse
- For å begrense eksponeringen forårsaket av kompromittering av en økt nøkkel.
- For å unngå langvarig lagring av et stort antall hemmelige nøkler.
- Nøkler opprettes etter behov eller når de faktisk kreves.
- For å skape uavhengighet mellom kommunikasjonssesjoner eller applikasjoner.

NØKKEL ETABLERING (OPPRETTELSE)

Dette refererer til prosessen med å opprette, utveksle eller etablere kryptografiske nøkler mellom to eller flere parter for å sikre sikker kommunikasjon eller dataoverføring.

Metoder for å oppnå nøkkel etablering kan deles i to grupper:

1. nøkkel transport: en nøkkel transport protokoll er en teknikk som på et vis trygt overfører en hemmelig verdi til noen andre.

2. nøkkel enighet (agreement): I en nøkkelavtaleprotokoll utveksler to (eller flere) parter verdier som er nødvendige for å generere den delte hemmelige nøkkelen, og hvor alle parter bidrar til hemmeligheten.

FLERE EGENSKAPER FOR NØKKELETABLERINGSPROTOKOLL

Nøkkelkontroll:

- I noen protokoller kontrollerer en av partene verdien til nøkkelen (nøkkeltransport), mens i andre kan ingen av partene kontrollere (forutsi) verdien (nøkkelavtale).

Nøkkelfriskhet:

- En part blir forsikret om at nøkkelen er ny (aldri brukt tidligere).

Nøkkelbekreftelse:

- Én part blir forsikret om at en annen part faktisk har økt nøkkelen.

Krav til tredjepart:

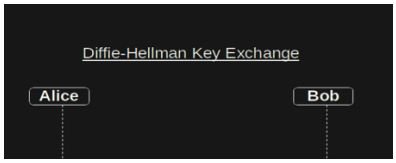
- Om det er et krav om å bruke en betrodd tredjepart eller ikke.

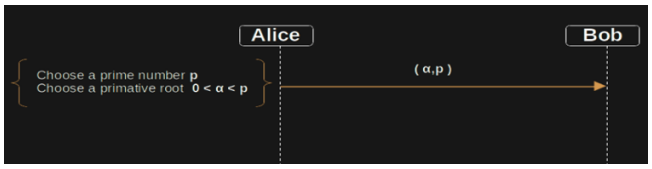
Full fremoversecrecy (Perfect Forward Secrecy - PFS):

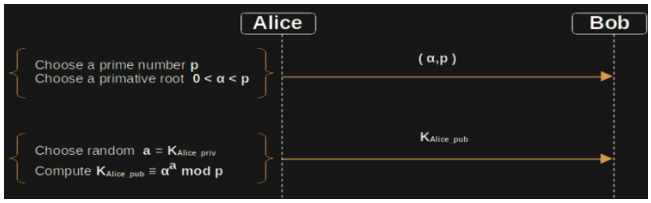
- En protokoll sies å ha egenskaper for full fremoversecrecy hvis en kompromittering av de langsiktige nøklene ikke kompromitterer tidligere øktnøkler.

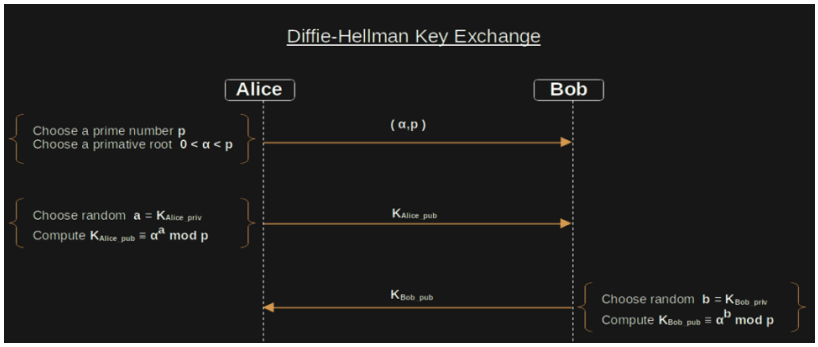
En av de tidligste og mest bredt brukte nøkkelutvekslingsalgoritmene er Diffie-Hellman.

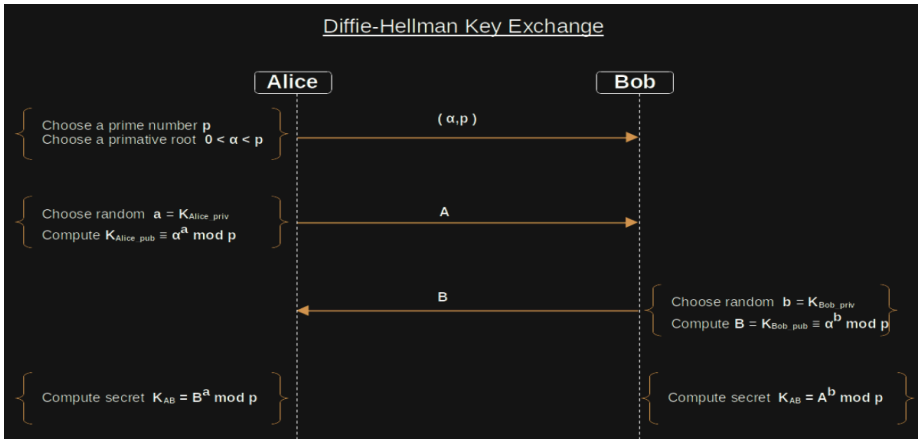
- Diffie-Hellman er en offentlig nøkkelprotokoll utviklet av Whitfield Diffie og Martin Hellman i 1976.
- Diffie-Hellman er basert på det diskrete logaritmeproblemet, og sikkerheten avhenger av vanskeligheten med å beregne den diskrete logaritmen.
- Protokollen lar to brukere utveksle en hemmelig nøkkel på en sikker måte, som senere vil bli brukt av symmetriske krypteringsalgoritmer.
- Den opprinnelige Diffie-Hellman-algoritmen er sårbar for et "mann-i-midten"-angrep!

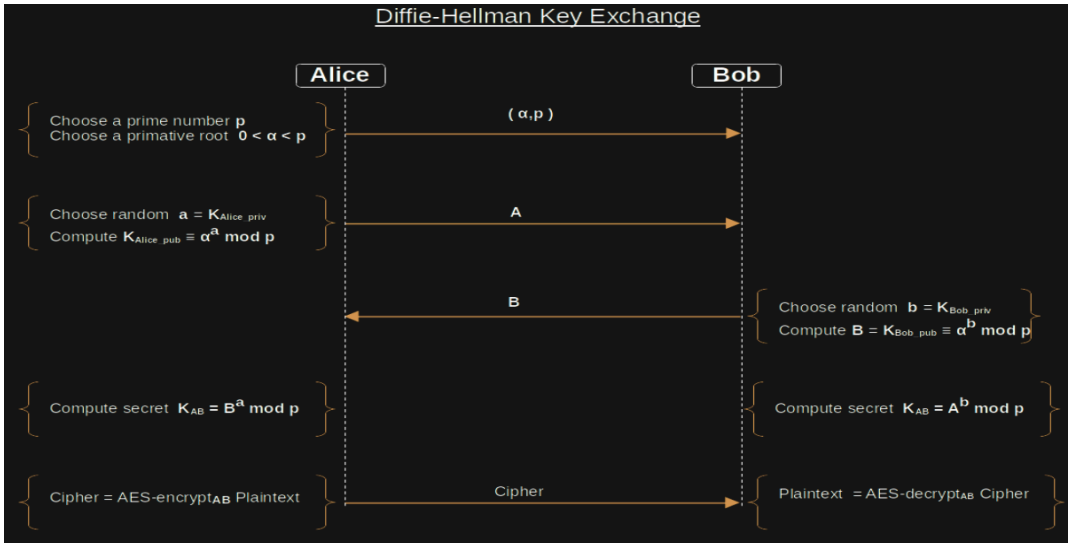

Diffie-Hellman Key Exchange
Alice
Bob

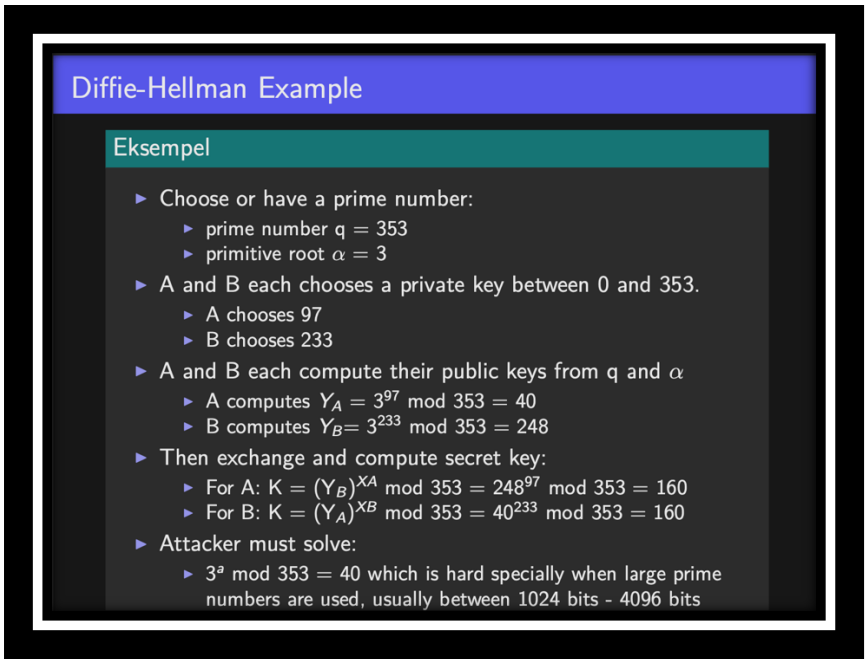

Alice
Bob
Choose a prime number p
Choose a primative root 0 < α < p
(α,p)


Alice
Bob
Choose a prime number p
Choose a primative root 0 < α < p
(α,p)
Choose random a = KAlice_priv
Compute KAlice_pub ≡ α^a mod p
KAlice_pub


Diffie-Hellman Key Exchange
Alice
Bob
Choose a prime number p
Choose a primative root 0 < α < p
(α,p)
Choose random a = KAlice_priv
Compute KAlice_pub ≡ α^a mod p
KAlice_pub
KBob_pub
Choose random b = KBob_priv
Compute KBob_pub ≡ α^b mod p

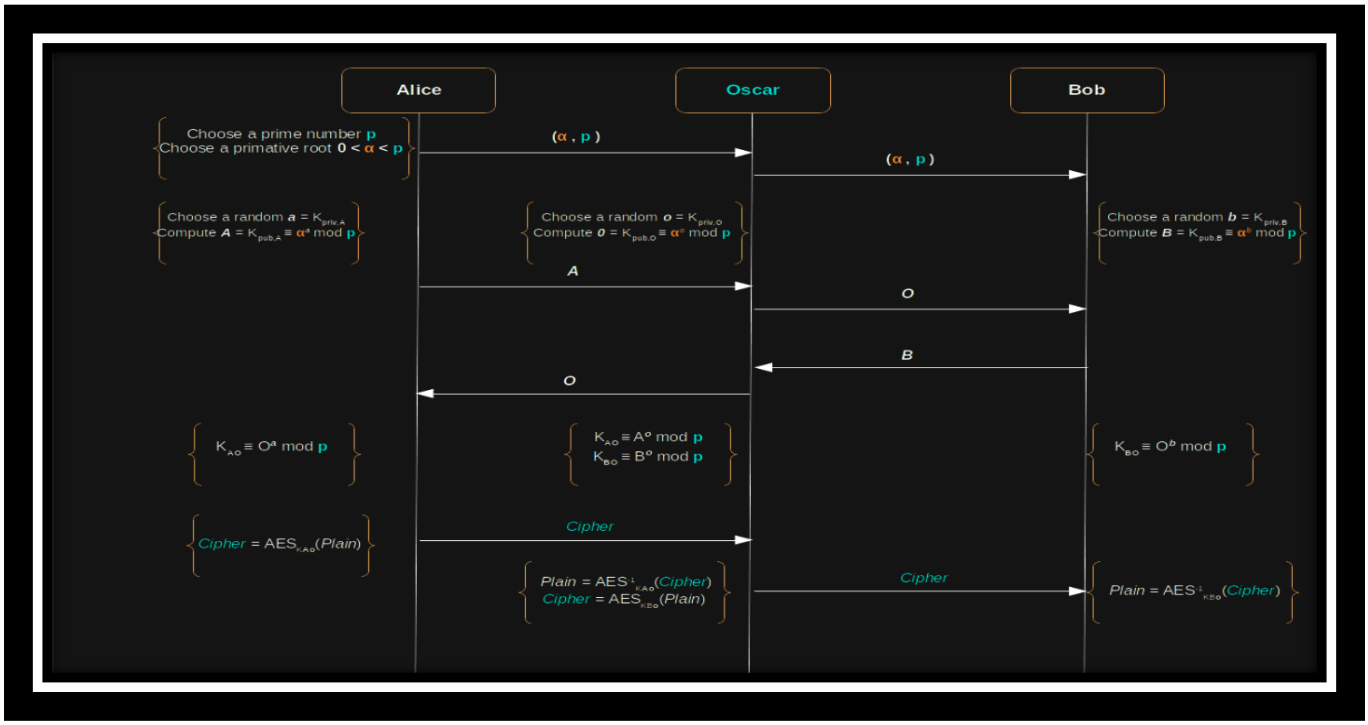

Diffie-Hellman Key Exchange
Alice
Bob
Choose a prime number p
Choose a primative root 0 < α < p
(α,p)
Choose random a = KAlice_priv
Compute KAlice_pub ≡ α^a mod p
A
B
Choose random b = KBob_priv
Compute B = KBob_pub ≡ α^b mod p
Compute secret KAB = B^a mod p
Compute secret KAB = A^b mod p


Diffie-Hellman Key Exchange
Alice
Bob
Choose a prime number p
Choose a primative root 0 < α < p
(α,p)
Choose random a = KAlice_priv
Compute KAlice_pub ≡ α^a mod p
A
B
Choose random b = KBob_priv
Compute B = KBob_pub ≡ α^b mod p
Compute secret KAB = B^a mod p
Compute secret KAB = A^b mod p
Cipher = AES-encryptAB Plaintext
Cipher
Plaintext = AES-decryptAB Cipher



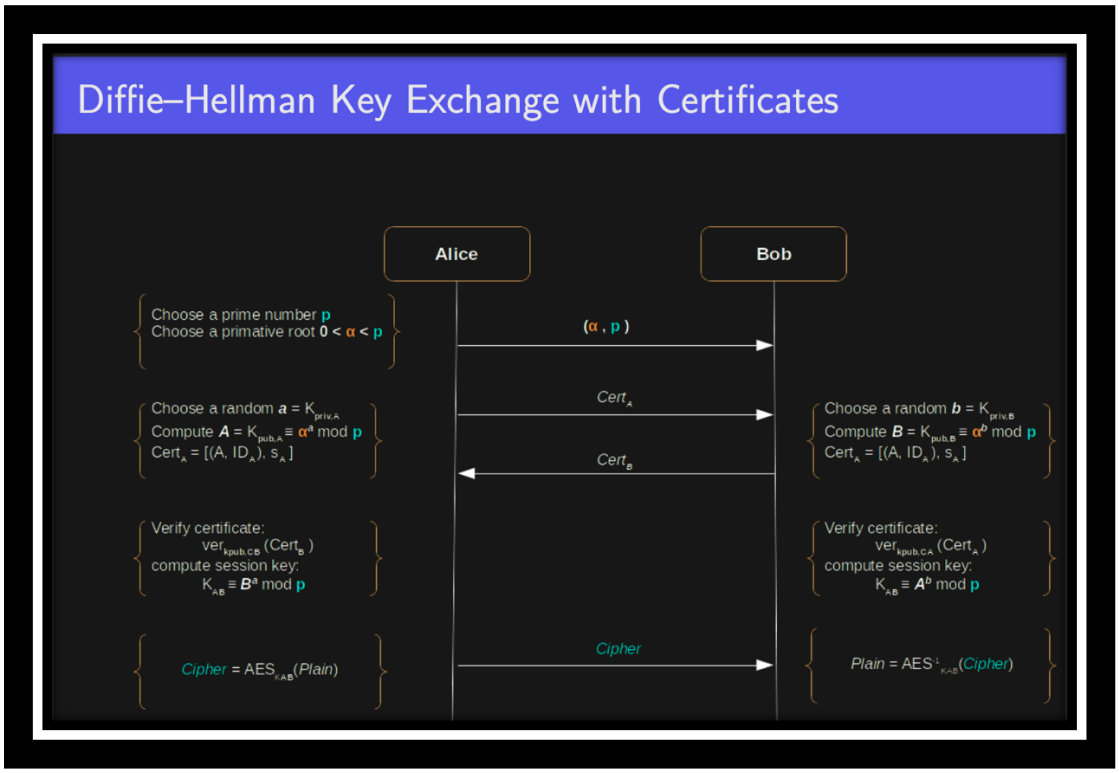
MAN-IN-THE-MIDDLE ATTACK

Seriøst angrep på utbyttet av hemmeligheter. Den grunnleggende ideen bak angrepet er at angriperen ofte blir kalt Oscar, Eve eller Mallory, erstatter for eksempel de offentlige nøklene til Alice og Bob med sine egne nøkler.



PUBLIC KEY INFRASTRUKTUR (PKI) SOM LØSNING

Det underliggende problemet med Man-in-the-middle angrepet er at de offentlige nøklene ikke er autentisert. Det er flere måter å løse problemet om nøkkel autentiseringen på. Hoved mekanismen som brukes av kryptografiske protokoller er bruken av offentlige nøkkel infrastrukturer (PKI): TTP (trusted third party AKA certificateAuthorities (CA)) og PGP (Pretty good privacy).



PUBLIC KEY INFRASTRUCTURE

Takket være OFFENTLIG NØKKEL KRYPTOGRAFI har vi muligheten til å privat kommunisere med folk som vi har trygt delt en offentlig nøkkel med, men det er en rekke andre utfordringer som er uløste. Hvordan kan vi sikkert kommunisere med folk vi aldri har møtt før? Hvordan lagrer vi nøklene? Hvordan tilbakekaller vi nøklene dersom det trengs? Hvordan gjør vi det på verdensbasis, med millioner av servere og milliarder av mennesker og enheter?

DETTE ER HVA PKI BLE UTVIKLET FOR Å LØSE

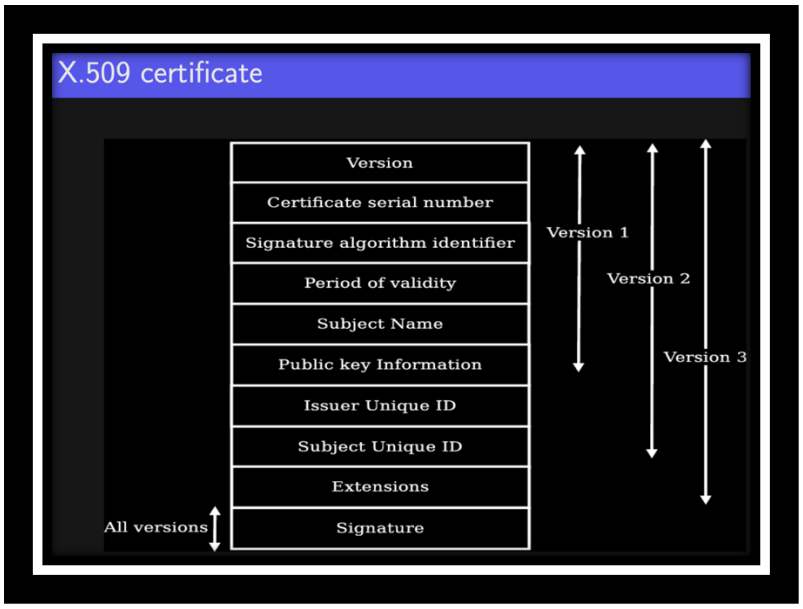
HVA ER FORSKJELLEN PÅ PKC OG PKI?

Forskjellen mellom Public Key Cryptography (PKC) og Public Key Infrastructure (PKI) er at PKC er selve kryptografiske teknikkene, mens PKI er en infrastruktur som omfatter tjenester og prosedyrer for å administrere offentlige nøkler og digitale sertifikater. PKC er en del av PKI og brukes til å oppnå sikkerhet i en PKI-basert infrastruktur.

Målet til PKI er å forsikre kommunikasjon mellom parter som aldri har møttes før. Modellen som PKI bruker i dag er avhengig av noe som kalles TTP også kjent som certificate authorities (CA). CA utdeler sertifikater til de vi kan stole på. Et sertifikat er et digitalt dokument som inneholder offentlig nøkkel, litt informasjon om eieren som er assosiert med den offentlige nøkkelen og en digital signatur fra den som utdeler sertifikatet. Slike sertifikater gir oss muligheten til å utveksle, lagre og bruke offentlige nøkler. Sertifikater er de grunnleggende bygge blokkene til PKI.

PKI STANDARDER

PKI har sine røtter i X.509, en internasjonal standard for Public Key Infrastructure som opprinnelig ble utviklet for å støtte X.500, en standard for elektroniske katalogtjenester. PKIX-arbeidsgruppen ble etablert høsten 1995 av IETF med mål om å utvikle internettstandarder for å støtte X.509-basert Public Key Infrastructure (PKI). RFC 5280 er det viktigste dokumentet som ble produsert av PKIX-arbeidsgruppen. Denne dokumentasjonen beskriver sertifikatformatet og tillitsstien, samt formatet for Certificate Revocation List (CRL).



SERTIFIKAT FELTER

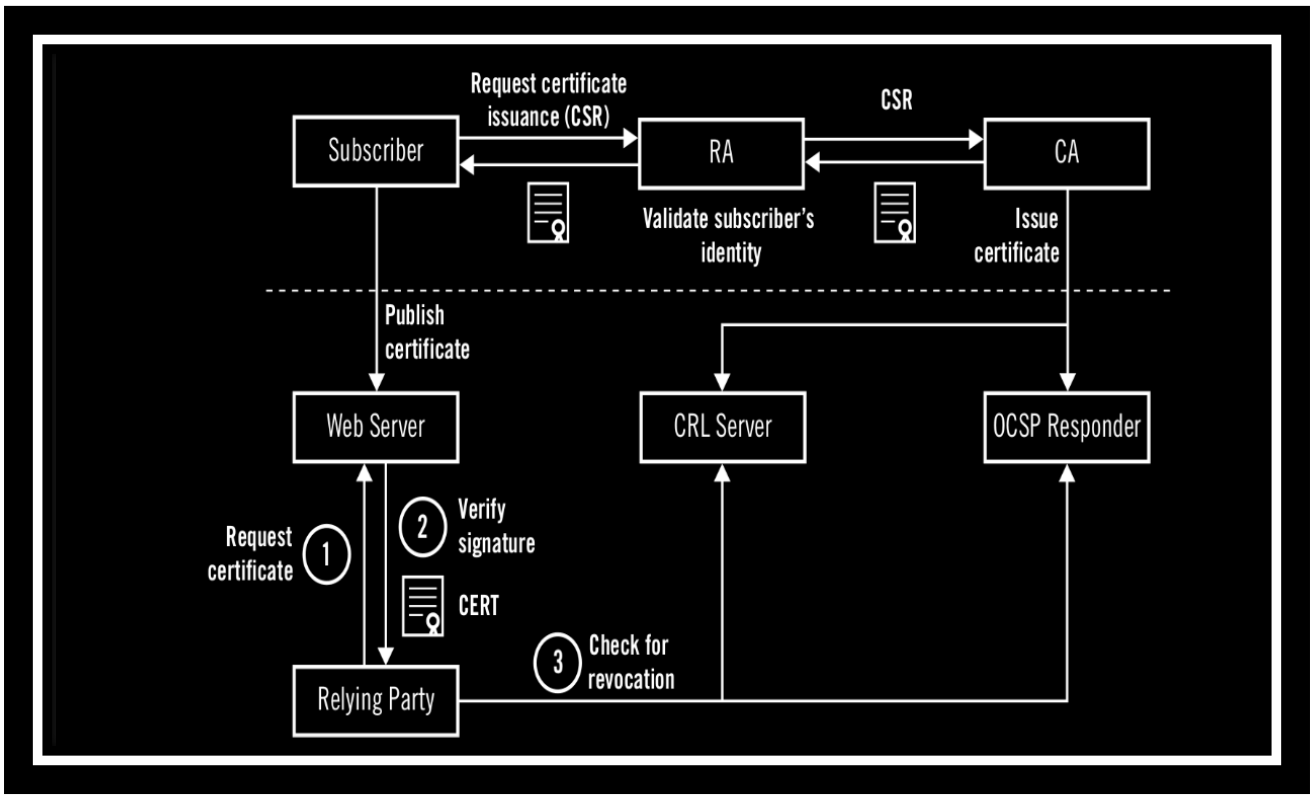
- Versjon
  - Det finnes versjon 1, 2 og 3
- Serienummer
  - Et unikt nummer som identifiserer et sertifikat utstedt av en gitt CA
- Signatur algoritme
  - Spesifiserer algoritmen brukt for sertifikat signaturen
- Angir algoritme
  - Inneholder det unike navnet (distinguished name (DN)) av sertifikat issuer
- Gyldighet
  - Tidsintervallet i hvilket sertifikatet er gyldig med to verdier: start og slutt
- Emne
  - Er det unike navnet (distinguished name (DN)) til hvilke det er lenket en offentlig nøkkel for når sertifikatet var utstedt.
- Offentlig nøkkel
  - Inneholder informasjon om den offentlige nøkkelen

SERTIFIKATUTVIDELSER (X.509 v3)

I. Alternativt navn for emne (Subject Alternative Name)

II. Navnbegrensninger (Name Constraints)
III. Bruksområde for nøkkel (Key Usage)
IV. Utvidet bruksområde for nøkkel (Extended Key Usage)
V. Sertifikatpolicyer (Certificate Policies)
VI. Distribusjonspunkter for Certificate Revocation List (CRL Distribution Points)
VII. Tilgang til autoritetsinformasjon (Authority Information Access)
VIII. Emneidentifikator for nøkkel (Subject Key Identifier)
IX. Autoritetsidentifikator for nøkkel (Authority Key Identifier)

Bildet under symboliserer syklusen til et sertifikat:



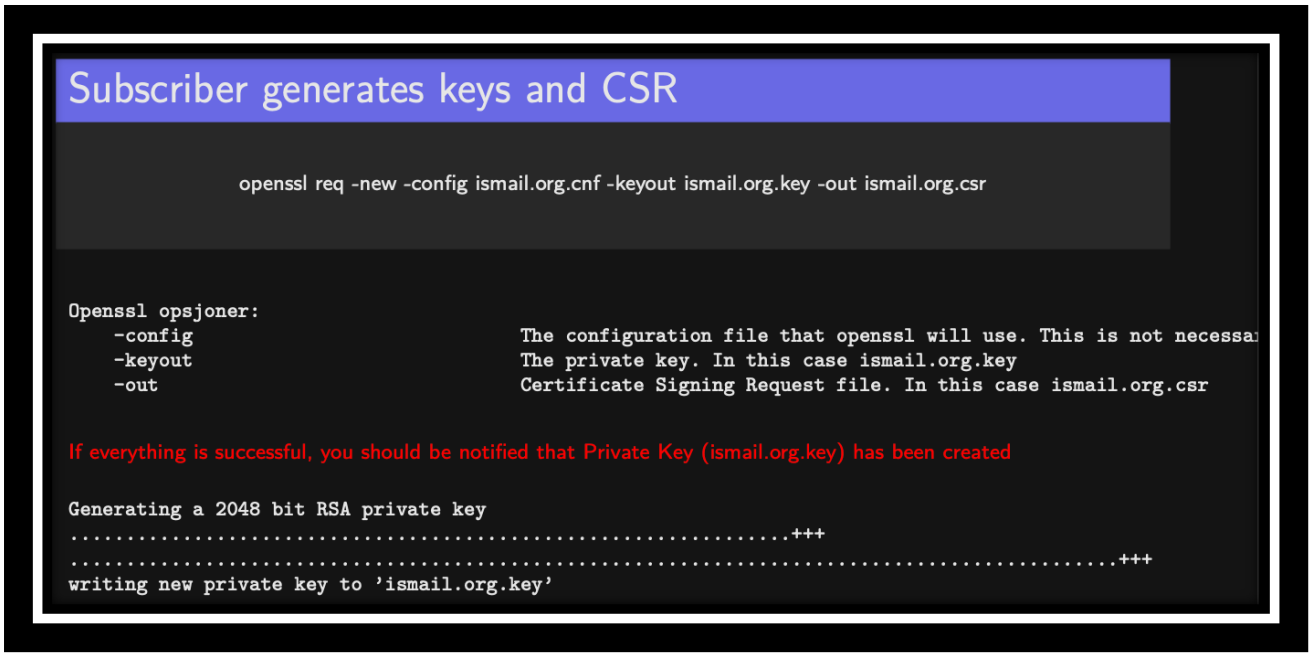
PARTER SOM INVOLVERT I LIVSSYKLUSEN TIL SERTIFIKATER

1) Subscriber eller end entity: er den som holder offentlige nøkkelen og ønsker et sertifikat som lenker nøkkelen til dems entitet.
2) Certifiction authority (CA): er en betrodd tredje parti som utdeler sertifikat til subscriber/EE

3) Registration authority (RA): en RA er de som identifiserer brukeren før et sertifikat kan forespørres av CA.
4) Relying parties (RP): en RP er forbruker av sertifikatet. Teknisk sett er disse nettlesere, andre programmer og operativsystemer som utfører sertifikat valideringer.

PROSEDYREN FOR Å SKAFFE ET SERTIFIKAT

Subscriber (end entity) genererer en offentlig/privat nøkkelpar og et certificate signing request (CSR). Dette kan gjøres med for eksempel med openssl:



CERTIFICATE SIGNING REQUEST (CSR)

Når vi har privat og offentlig nøkkel altså (RSA) kan vi forespørre en CSR. Men for å kunne gjøre det er det nødvendig at vi har den private nøkkelen for å generere CSR. Se demo.

Det finnes tre typer sertifikat:

1) Domene validering (DV)

   a. Her er bare domene validert av sider og identifisert av browser. Dette er tilstrekkelig for å etablere en kryptert kommunikasjon, men gir ingen annen verifisering for brukeren angående om domenet eies av organisasjonen som nettsiden ser ut til å representere.
2) Organisasjons validering (OV)
   a. Begge domene og organisasjon identifiseres til browser. Hengelås i adresselinjen
3) Utvidet validering (EX)
   a. Tilbyr høyeste nivå med troverdighet og krever streng validering av begge domene og organisasjon. Kravene for utstedelse av en EV-sertifikat (Extended Validation Certificate) er definert av CA/Browser Forum Baseline Requirements.

Når en end entity eller en subscriber har bestemt hvilken type sertifikat de vil ha skal se sende en CSR til en CA. Dette gjøres ofte gjennom en nettleser hvor CSR filen skal lastes opp eller limes inn i et felt.

RA sjekker identifiseringen fra subscriber eller EE. Avhengig av type sertifikat kan RA forespørre identifisering, organisasjons nummer (NO domain). Når identity er bekreftet kan RA sende CSR til CA. Ofte er RA og CA den samme entiteten. CA lager sertifikatet med subscriber eller EE med den offentlige nøkkelen og informasjonen fra RA. Sertifikatet baserer seg på X.509 standard vil også inneholde informasjonen vi gikk gjennom før.

En kan også lage Self-Signed-Certificate. Ulempen med dette er at det ikke er utdelt av en TTP. Noe som kan lede klienter og andre til å tro at sertifikatet er ugyldig.

CERTIFICATE VALIDATION PROCESS

Hvis sertifikatet er gyldig kan nettleseren vise en liten lås som man kan klikke på for å få mer informasjon om sertifikatet i adresselinjen. Dersom det er problemer med siden, vil nettleseren fortelle dette. RP er forbruker av sertifikatet, nettlesere eller operativsystemer. OS, nettlesere og andre applikasjoner må ha sertifikater som beviser at de er autentiserte, som også

CA gir. I de fleste tilfeller må de ha en samling (Trust store) av roten CA sertifikater som de stoler på.

REVOCATION

- Sertifikater blir tilbakekalt når tilhørende private nøkler er kompromittert eller ikke lenger er nødvendige
- I begge tilfellene er det en risiko for misbruk
- Tilbakekallingsprotokoller og prosedyrer er utformet for å sikre at kommuniserende parter kan fastslå om sertifikatet de bruker er gyldig eller tilbakekalt
- Det finnes to standarder for tilbakekalling av sertifikater:
    - Sertifikatets tilbakekallingsliste (Certificate Revocation List, CRL)
    - Protokoll for statussjekk av sertifikater på nett (Online Certificate Status Protocol, OCSP)

SERTIFIKATETS TILBAKEKALLINGSLISTE (CERTIFICATE REVOCATION LIST, CRL)

CRL er en liste av alle serie nr som tilhører tilbaketrukket sertifikater som ikke har gått ut ennå

- CA holder styr på en eller flere lister
- Hvert sertifikat må inneholde informasjon om hvor CRL listen kan hentes fra

Hoved utfordringen med CRL er at de bruker å være lange som får sanntid letinger til å ta lang tid.

ONLINE CERTIFICATE STATUS PROTOCOL (OCSP)

- OCSP kan gi tilbakekallingsstatus for enkeltsertifikater.
- CA (Certificate Authority) vedlikeholder én eller flere slike lister.

- Sertifikatet må inneholde informasjon om hvilken OCSP-server man kan hente tilbakekallingsstatus fra.
- OCSP muliggjør sanntidsoppslag og løser hovedutfordringen med CRL (Certificate Revocation List).
- OCSP medfører visse kostnadsutfordringer for CA. Protokollen krever at de svarer i sanntid til hver klient for et gitt sertifikat.
- OCSP-stapling er utviklet som et alternativ til OCSP for å løse problemet.

OCSP gir sanntidsstatus om sertifikater og kan derfor være raskere og mer effektiv for klienter.

PRETTY GOOD PRIVACY (PGP)

PGP VS PKI

I PKI verden stoler vi på en rot entitet som Root CA (certificate authority). Vi stoler på at CA har en prosess og prosedyre i plass for å forsikre oss at de aldri skal utdele et sertifikat til Amna som inneholder en privat nøkkel som ikke er Amnas.

PGP er litt annerledes. Der baseres det på «web-of-trust» modellen. I stedet for å stole på en allmektig CA bestemmer vi om vi kan stole på om en PGP nøkkel tilhører en spesifikk bruker eller ikke. For eksempel har jeg en venn miski som jeg stoler på. Hvis miski sier «her en amna sin PGP nøkkel» (hun sier dette ved å signere en kopi av amna sin nøkkel) så kan du velge å stole på om det er sant eller ikke.

PGP KEY MANAGMENT (HÅNDTERING):

PGP bruker begrepet "Nøkkelring" for å administrere nøkler. Nøkler lagres på en eller flere nøkkelringer. Hver bruker vil ha minst to nøkkelringer, en for private nøkler og en for offentlige nøkler. En offentlig nøkkel i nøkkelringen er gyldig hvis:

- Du har generert eller signert den selv.

- Noen du stoler på har signert den.
- Antallet personer du delvis stoler på har signert den, der N er et justerbart tall. En

Person kan stole på en nøkkel hvis:

- Du har gitt den offentlige nøkkelen full tillit eller delvis tillit.
- Tillit kan bare tildeles eksplisitt!

Så når det kommer til PGP nøkler er de validerte dersom de oppnås gjennom web of trust eller med en digital signatur fra CA.

Hvorfor velge PGP fremfor TTP?

1. Enkel oppsett: Lett å konfigurere.
2. Krever ikke en komplett infrastruktur: Krever ikke en hel infrastruktur.
3. Best egnet for uformelle grupper, venner, kolleger og bekjentskaper.
4. Enhver kan bli en "CA" ved å bruke tillitssignaturer (trust signatures).
5. Mangfoldig tillitsmodell (varied trust model).
6. Mer kontroll over egne nøkler. Tilbakekalling (revocation) er enkelt.

GPG OG PGP

GnuPG er et program som er forhånds installert på alle Linux distribusjoner. PGP applikasjoner er også tilgjengelige på andre plattformer som Windows og apple. Man må generere asymmetriske nøkler (privat og offentlig), så må man ta en del i nøkkelsignerings fester, så må man publisere offentlige nøkler på en nøkkel server (PGP server). Nøklene kan da brukes av andre som vil sende deg krypterte meldinger. GPG er operativsystemet som ismail brukte i demoen.

PROBLEMER MED NØKKELSERVERE ETTER GDPR (general data protection regulations)

Problemet med PGP (Pretty Good Privacy) nøkkelservere etter GDPR (General Data Protection Regulation) handler primært om personvernbekymringer og overholdelse av GDPR-regler. PGP nøkkelservere kan potensielt bryte GDPR-prinsipper ved å:

- gjøre personlig identifiserbar informasjon (PII) lett tilgjengelig uten eksplisitt samtykke
- ikke sikre dataminimering og gjøre det utfordrende å fjerne personlige data.

GDPRs strenge krav til databeskyttelse og individuelle rettigheter er i konflikt med den tradisjonelle funksjonen til PGP nøkkelservere, noe som skaper utfordringer med overholdelse og personvernrisiko.