FEIDE: FElles Elektronisk ID for UoH-sektoren

Alf Hansen
(UNINETT FAS)
Anund Lie
Jon Ølnes
(NR)

Norsk Regnesentral
ANVENDT DATAFORSKNING

Rapport/Report

Tittel/Title:
FEIDE: FElles elektronisk ID for UoH-sektoren

Dato/Date: 29. september
År/Year: 2000
ISBN: 82-539-0467-3
Publikasjonsnr.: 963
Publication no.:

Forfatter/Author:
Alf Hansen (UNINETT FAS), Anund Lie, Jon Ølnes (NR)

Sammendrag/Abstract:

Formålet med FEIDE er å etablere en felles infrastruktur (PKI) for sikker identifikasjon (autentisering) av studenter og ansatte i universitets- og høgskolesektoren. FEIDE skal brukes som grunnlag for sikker datakommunikasjon, utveksling av data og dokumenter på tvers av lærestedene, adgangskontroll til Internett-baserte tjenester, og spesielt for adgangskontroll til Felles Administrative Systemer (FAS). Det er også aktuelt å bruke FEIDE for andre formål som krever identifikasjon, f.eks. nøkkelkort, innlogging på arbeidsstasjoner etc. Den eneste løsningen som vurderes til å gi god nok sikkerhet er offentlig nøkkel-kryptografi med bruk av smartkort som medium for private nøkler.

Digital signatur er en sentral funksjon i FEIDE. Det legges opp til relativt høye sikkerhetskrav i FEIDE (jfr. bruk av smartkort i stedet for programvareløsninger for kryptografi), men likevel ikke til å tilfredsstille kravene til kvalifisert signatur etter den foreslåtte loven om elektronisk signatur. Grunnen til dette er at en kvalifisert signatur er et svært sterkt krav, som ikke uten videre er forenlig med andre bruksområder for FEIDE.

En ID som er utstedt av et av lærestedene skal uten videre kunne aksepteres av de andre lærestedene. En FEIDE ID er ikke knyttet til et bestemt lærested (heller ikke det lærestedet der den først ble utstedt). Sertifikattjenesten for FEIDE kan realiseres på flere måter. De eksisterende eller planlagte kommersielle sertifikattjenestene (Postens Elektroniske ID, Telenor Zebrasign, BankID) kan muligens benyttes mer eller mindre direkte.

FAS og andre systemer som bruker FEIDE for brukerautentisering nås i første rekke gjennom web-grensesnitt, dvs. vanlige web-lesere med drivere for smartkortbasert kryptografi og SSL/TLS med klientautentisering som transportsikkerhetsløsning. De samme tekniske løsningene gir samtidig epost-sikkerhet, med digital signatur og konfidensialitet. Det er også mulig at FEIDE kan brukes for andre løsninger, som f.eks. aksess-VPN, men her er neppe tiden (og produktene) modne for felles spesifikasjoner på tvers av lærestedene.

Emneord/Keywords: offentlig nøkkel infrastruktur (PKI), smartkort, digital signatur

Tilgjengelighet/Availability: åpen

Prosjektnr./Project no.: 28 001

Satsningsfelt/Research field:

Norsk Regnesentral / Norwegian Computing Center
Gaustadalléen 23, Postboks 114 Blindern, 0314 Oslo, Norway
Telefon 22 85 25 00, telefax 22 69 76 60

Antall sider/No. of pages: 85 sider

Innholdsfortegnelse

Sammendrag

Formålet med FEIDE er å etablere en felles infrastruktur (PKI) for sikker identifikasjon (autentisering) av studenter og ansatte i universitets- og høgskolesektoren. FEIDE skal brukes som grunnlag for sikker datakommunikasjon, utveksling av data og dokumenter på tvers av lærestedene, adgangskontroll til Internett-baserte tjenester, og spesielt for adgangskontroll til Felles Administrative Systemer (FAS). Det er også aktuelt å bruke FEIDE for andre formål som krever identifikasjon, f.eks. nøkkelkort, innlogging på arbeidsstasjoner etc.

Den eneste løsningen som vurderes til å gi god nok sikkerhet er offentlig nøkkel-kryptografi med bruk av smartkort som medium for private nøkler. Rene programvareløsninger beskytter ikke private nøkler godt nok, spesielt ikke på lab-maskiner og andre maskiner som er forholdsvis åpent tilgjengelige og deles av mange brukere. Dette krever at smartkortlesere og drivere er installert på alle klientmaskiner, men kostnaden for dette er relativt liten, og samme utstyr kan dessuten benyttes for andre smartkortanvendelser. Identifikasjonskort for ansatte og studenter utstedes som smartkort med elektronisk ID-applikasjon ferdig installert med nøkler og sertifikater. Dersom et multi-applikasjons-smartkort blir valgt, kan samme kort også brukes til andre formål, som elektronisk småpengebetaling (kantine, kopiering, utskrift m.m.), nøkkelkort (hvis dette ikke dekkes av elektronisk ID), bankkort o.l. Utlevering av kort krever personlig fremmøte og legitimasjon, men ellers kan rutinene rundt utstedelse og initialisering legges opp forholdsvis fleksibelt etter lærestedenes behov og forutsetninger.

Digital signatur er en sentral funksjon i FEIDE. Det legges opp til relativt høye sikkerhetskrav i FEIDE (jfr. bruk av smartkort i stedet for programvareløsninger for kryptografi), men likevel ikke til å tilfredsstille kravene til kvalifisert signatur etter den foreslåtte loven om elektronisk signatur. Grunnen til dette er at en kvalifisert signatur er et svært sterkt krav, som ikke uten videre er forenlig med andre bruksområder for FEIDE. Blant annet antar vi at nøkler som er godkjent for kvalifisert signatur ikke vil bli tillatt å bruke på vanlig PC-type utstyr (ikke minst pga. risikoen for misbruk dersom maskinen er angrepet av virus eller trojanske hester.) Dette betyr ikke at signaturen ikke er sterk nok for de formålene som er aktuelle i FEIDE, men forskrifter og reglementer må eksplisitt anerkjenne det sikkerhetsnivået som er gitt ved FEIDEs sertifikatpolicy, i stedet for bare å stille opp et generelt krav om signatur. Der det er spesielt høye krav (f.eks. utbetaling av studielån) kan en kjoskløsning med spesielt sikrede og overvåkede maskiner være aktuelt.

En ID som er utstedt av et av lærestedene skal uten videre kunne aksepteres av de andre ærestedene. Den skal gi grunnlag for sikker **autentisering**, mens **autorisering** og **adgangskontroll** i utgangspunktet er opp til hvert enkelt lærested, på grunnlag av denne autentiseringen. FEIDE legger foreløpig ikke opp til bruk av eller noen felles infrastruktur for rettighets-/attributtsertifikater. En FEIDE ID er ikke knyttet til et bestemt lærested (heller ikke det lærestedet der den først ble utstedt). Bytter personen lærestedstilknytning, beholdes dermed samme ID. Av personvernhensyn benyttes ikke personnummer i FEIDE-ID. Institusjoner som har legitime behov for å koble FEIDE-ID til personnummer får i stedet adgang til (utvalgte deler av) tabellen der koblingen mellom FEIDE-ID og personnummer er registrert, men adgang til denne tabellen er strengt regulert.

Alle FEIDE-sertifikater, tilbakekallingslister etc. er registrert i en sentral katalog (LDAP). Denne katalogen (eller en speiling av den) er tilgjengelig for FAS. Et uttrekk av den er også allment tilgjengelig på Internett (men inneholder muligens bare informasjon om de personene som aktivt har gitt samtykke til det). Lærestedene (eller systemene i FAS) vil typisk også ha lokale LDAP-kataloger som bl.a. kan inneholde autorisasjonsinformasjon, og som ellers videresender oppslag til de sentrale katalogene. Alle med en gyldig FEIDE-ID vil være autorisert for noen grunnleggende funksjoner, f.eks. adgang til informasjon om seg

selv, registrering eller endring. Andre autorisasjoner tildeles trolig best gjennom registrering i de lokale LDAP-katalogene eller tilsvarende.

Sertifikattjenesten for FEIDE kan realiseres på flere måter. De eksisterende eller planlagte kommersielle sertifikattjenestene (Postens Elektroniske ID, Telenor Zebrasign, BankID) kan muligens benyttes mer eller mindre direkte. Tjenester, utstyr og programvare kan kjøpes inn over Forvaltningsnettsamarbeidets rammeavtaler (med noen små justeringer av spesifikasjonene). Hvis ingen av disse viser seg å passe, kan det hende det er nødvendig å spesifisere en egen tjeneste for FEIDE. Det er neppe aktuelt at UNINETT eller lærestedene selv driver sertifikattjenester. Derimot kan det være aktuelt å bruke to eller flere leverandører av sertifikattjenester og la valget være opp til lærestedet eller den enkelte bruker.

FAS og andre systemer som bruker FEIDE for brukerautentisering nås i første rekke gjennom web-grensesnitt. Den primære tekniske løsningen for klientene er vanlige web-lesere, som Opera, Communicator, Internet Explorer. Vi forutsetter nyere versjoner som er tilgjengelig med «sterk» kryptografi internasjonalt. Det er kurant å utstyre disse pakkene med drivere for smartkort-basert kryptografi. Løsningen blir ikke FEIDE-spesifikk, men kan også brukes i andre sammenhenger der smartkort-basert kryptografi er aktuelt. De samme tekniske løsningene gir samtidig mulighet for sikker epost, med digital signatur og konfidensialitet.

Kommunikasjonen mellom klient og applikasjonstjener sikres med SSL/TLS, som web-leserne allerede har innebygget støtte for. Brukeren autentiseres overfor applikasjonen (som f.eks. er en web-frontend for FAS) ved SSL klientautentisering, som støttes rimelig godt av mange av de aktuelle plattformene for web-applikasjoner. Konkret betyr det at brukeren setter sitt FEIDE-smartkort i kortleseren knyttet til klientmaskinen og aktiverer smartkortet ved å taste inn PIN-kode i det sesjonen mot web-applikasjonen starter. Web-applikasjonen på sin side vil ha adgang til resultatet av autentiseringen (brukerens sertiserte ID, gjennom sertifikatet), og dette kan kobles mot applikasjonsplattformens aksesskontrollmekanismer på forskjellige måter.

FEIDE er først og fremst aktuelt for sikkerhetstjenester på applikasjons- evt. transportnivå, i praksis transportlagssikkerhet med SSL/TLS eller meldingssikkerhet med PKCS#7 og S/MIME. Vi utelukker likevel ikke helt at FEIDE eller FEIDE-PKIen blir brukt i andre sammenhenger, f.eks. til aksess-VPN, men her tror vi ikke tiden (og produktene) er modne for felles spesifikasjoner på tvers av lærestedene.

1. Bakgrunn: Felles Administrative Systemer og FEIDE

1.1 Felles Administrative Systemers rolle i den elektroniske høgskolen -- en visjon

De «Felles Administrative Systemer», FAS, som KUF og høgskolene anskaffer og driver koordinert gjennom UNINETT FAS, er en sentral byggesten i «den elektroniske høgskolen». Denne visjonen gjelder for alle høgskoler som deltar i TROFAST-prosjektet ved at de tiltrer rammeavtalen mellom KUF og UNINETT FAS om teknisk drift av TROFAST-tjenerne.

Hovedmålsettingen med administrative systemer er å støtte/utføre de oppgavene som skal gjøres, som f.eks. å utbetale lønn, betale regninger, foreta studentopptak, gjennomføre eksamensavvikling, produsere vitnemål, lette saksbehandlingen osv. Forholdene skal legges til rette for å få fram sammenlignbare data mellom høgskolene samtidig som veksten i utgiftene til administrative datasystemene skal begrenses. Det er nødvendig med et felles rammeverk for lover, regler, definisjoner og en koordinering av et begrepsapparat.

Visjonen innebærer at man skal oppnå følgende:

1. Alle administrativt ansatte hjemme og på kontoret kommuniserer i ett elektronisk nettverk, Internett.
2. Brukergrensesnittet er felles, med tilgang til samme standard applikasjoner.
3. Elektronisk informasjon registreres bare en gang og blir straks tilgjengelig for alle som skal ha tilgang til den, avhengig av hvilke brukerrettigheter den enkelte får tildelt.
4. Skriftlig informasjonsutveksling og saksbehandling ved høgskolene skjer i hovedsak elektronisk, og ikke på papir.

For å oppnå dette vil man arbeide for at de felles administrative systemene (systemer for økonomi, studieadministrasjon, lønn- og personal, saksbehandling/arkiv) skal oppfylle følgende forventninger:

- De bør være web-basert, bl.a. fordi man forventer å gå over til et nettverk med tynne klienter p.g.a. brukervennlighet, effektiv drift av klienter og integrasjon mellom systemene til ett system sett fra brukernes side.
- De må kunne håndtere elektronisk signerte dokumenter på en tilfredsstillende måte. Felles Elektronisk IDEntifisering (FEIDE) er et viktig element i videre utvikling av «Den elektroniske høgskolen». Det å etablere en tjenesteinfrastruktur der man kan verifisere personers identitet over et åpent Internett synes å være en stor utfordring og gi et betydelig effektiviseringspotensiale.
- De må kunne integreres så sømløst som mulig seg i mellom og med andre systemer som er i bruk eller vil bli innført. Særlig viktig vil det være at de felles administrative systemene kan levere underlag for adressekatalogen (organisatoriske enheter og enkeltmedarbeidere/studenter, med organisatoriske forkortelser, initialer, offentlige nøkler, e-postadresser, etc.) til andre systemer, jfr. punkt 3 i visjonen. Standard utvekslingsformater (f.eks. XML) skal benyttes. De systemene som først og fremst er aktuelle for integrasjon er:
    - Økonomisystemet Agresso (felles for alle høgskoler).
    - Studieadministrasjon: FS er utviklet som et internt system for høgskoler og universiteter. MSTAS benyttes av de fleste høgskolene.
    - Lønn/personalsystem: Produkt ikke valgt enda, felles system utredes.
    - Kataloger basert på LDAP
    - System for administrasjon av brukerkonti og tilgangs- og adgangskontroll (e-post, PC, etc.).
- De må kunne levere nødvendige, ensartede, godkjente, elektroniske rapporter til DBH, KUF, Finansdepartementet, etc.

- Systemene må kunne generere og presentere helhetlig og oppdatert statistikk og styringsinformasjon som kan tilpasses ledernes informasjonsbehov.
- De må kunne levere underlag til systemer som benyttes for produksjon av diverse trykksaker.
- Systemene må være brukervennlige, driftssikre og trygge for innbrudd.
- Det må være mulig å få tilgang til systemene fra hvor som helst i verden der det er Internett-forbindelse, også fra mobile enheter.

UNINETT FAS er KUFs verktøy for å gjennomføre utvikling og tilpasning, koordinering, avtaler, innføring, initiell teknisk drift, avtaler om drift og brukerstøtte med underleverandører og annen sentral aktivitet som er hensiktsmessig for å oppnå besparelser samlet sett.

Dagens TROFAST-systemer realiserer tjenestene i startfasen i en distribuert arkitektur. Arkitekturen må tilpasses etter hvert som nye systemer innføres, og en må være åpen for kombinasjoner av sentrale/distribuerte arkitekturer. Driftsopplegget må være fleksibelt slik at en mest mulig fritt har mulighet til å velge hvem som skal drive de enkelte systemene teknisk.

1.2 FEIDE i den elektroniske høgskole -- en visjon

I forrige avsnitt presenteres en visjon for FAS, der FEIDE inngår som en komponent. Denne visjonen for FEIDE gjelder for alle høgskoler.

Det overordnede mål er å spare ressurser for universitets- og høgskolesektoren. Administrasjon av brukere av systemer skal effektiviseres ved å etablere en tjenesteinfrastruktur der man kan verifisere personers identitet over et åpent Internett. Når identiteten er bestemt, kan dette brukes til å bestemme tilgang til Felles Administrative Systemer (FAS) eller andre systemer som drives på høgskolene. Dette aktualiseres sterkt etter hvert som saksbehandling og undervisning går over på elektronisk form der brukerne ikke nødvendigvis er fysisk tilstede på høgskolen.

Visjonen innebærer at man vil oppnå følgende:

1. En student eller en ansatt skal kunne identifiseres sikkert over et åpent Internett. Derved vil man kunne gi aksess til lukkede systemer selv om man ikke er fysisk tilstede.
2. Brukergrensesnittet er felles for alle høgskolene der brukerutstyret er tilpasset det standard utstyr som er nødvendig for å benytte tjenestene.
3. Utstedelse av en elektronisk identitet i form av et sertifikat og en hemmelig nøkkel på et smartkort er en integrert del av opptaks- og ansettelsesprosessene ved en høgskole.
4. Administrasjon av aksess til informasjon og systemer kan utføres på en mest mulig effektiv måte.
5. Høgskolene skal bli en tidlig bruker av elektroniske sertifikattjenester som er under utvikling i det norske samfunn. Det skal IKKE bygges opp en særskilt sertifikattjeneste spesielt for vår sektor. Dette gir god utnyttelse og samordning med felles tiltak innen dette området, og forbereder sektoren på bl.a. e-handel.

For å oppnå dette vil FEIDE arbeide for at høgskolene koordineres for å oppnå nødvendige fordeler, slik at høgskolene ser at følgende forventninger blir oppfylt:

- Valg av felles smartkortteknologi (operativsystem, leverandør(?)).
- Et standardisert opplegg for praktisk logistikk og system for administrasjon av tildeling av sertifikater og nøkler på smartkort, samt spredning av offentlige nøkler f.eks. i kataloger.
- En koordinert prosjektaktivitet for å tilpasse de systemer som skal benytte elektronisk identifisering for å kontrollere aksess til systemene.
- En samordnet avtale med nasjonal leverandør av sertifikattjenester.
- En felles plan for hvordan brukerutstyr for studenter og ansatte kan utbygges med nødvendige tilleggsenheter (smartkortlesere og tilhørende programvare).

- Oversikt over hvilken eksisterende programvare (e-postklienter, nettlesere, etc.) som kan benyttes direkte eller som må tilpasses.
- En felles plan for å samordne koblingen mot katalog (LDAP) og brukerkontoadministrasjon.
- Avklaring av juridiske aspekter og personvern.
- Man skal kunne benytte resultatene fra FEIDE til å kontrollere aksess til systemer for
    - Studieadministrasjon: FS er utviklet som et internt system for høgskoler og universiteter. Utviklingsplaner for M-STAS er ikke kjent.
    - Økonomisystemet Agresso (felles for alle høgskoler).
    - Lønn/personalsystem: Produkt ikke valgt enda, felles system utredes.
    - Kataloger basert på LDAP.
    - System for administrasjon av brukerkonti og tilgangs- og adgangskontroll (e-post, PC, etc.).

FEIDE-teknologien skal også være forberedt for bruk innen områder som

- Fysisk adgangskontroll.
- Kontroll av studenters adgang til undervisning på nettet.
- Diverse administrativ kontroll av personers forbruk av lokale tjenester som f.eks. kopiering.

FEIDE vil ikke legge vekt på proprietære, lukkede løsninger, men vil etablere åpne standardløsninger for sektoren som helhet, med sikte på en samlet effektiviseringsgevinst for sektoren.

FEIDE er avhengig av egeninnsats fra høgskolene og av teknisk kompetanse fra hele UNINETT-miljøet.

1.3 Om FEIDE-prosjektet

UNINETT FAS initierte FEIDE prosjektet våren 2000 gjennom et seminar med deltakere fra store deler av UoH-sektoren. Deretter ble det etablert et prosjekt med deltakere fra UNINETT, NR (Norsk Regnesentral) og USIT (Universitetes Sentrale IT Enhet ved Universitetet i Oslo). USIT fikk ansvaret for å se på (krav til) samordning for administrative systemer og andre anvendelser, mens NR skulle se på selve den elektroniske identiteten, og på et rammeverk for hvordan web-baserte applikasjoner skal kunne utnytte elektronisk ID til å dekke forskjellige sikkerhetskrav.

Prosjektgruppa fikk frist til 1. oktober 2000 med å legge fram sin rapport. Prosjektet er å betrakte som et forprosjekt, og resultatene vil måtte bearbeides litt videre før det er aktuelt å trekke noen helt klare konklusjoner. Det ligger likevel noen klare anbefalinger i rapporten.

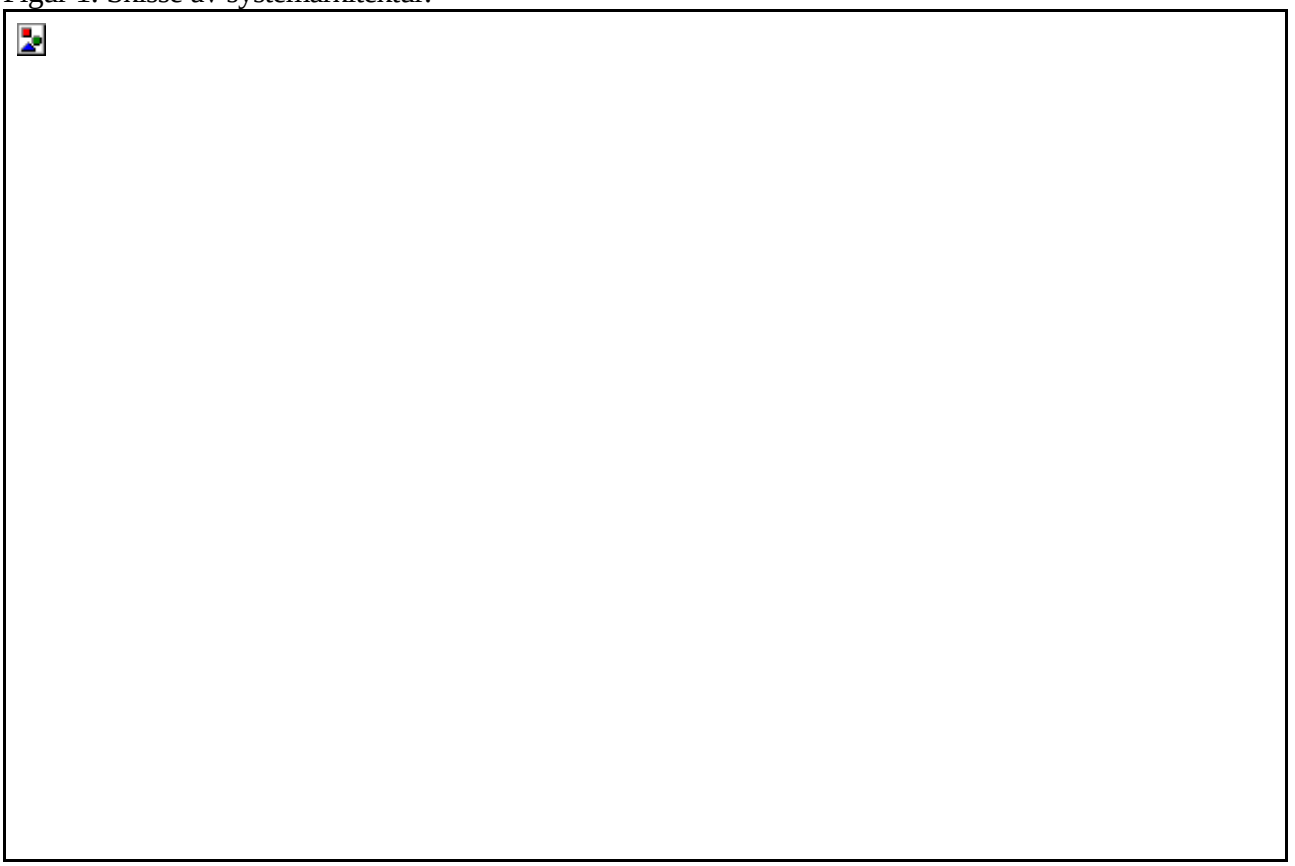
Det er viktig at rapporten legges på web og annonseres bredt både innen UoH-sektoren og innen fagmiljøer som arbeider med tilsvarende problemstillinger, f. eks. leverandører av sertifikattjenester, offentlige myndigheter, og andre, sammenlignbare prosjekter. Rapporten skal kun inneholde helt åpen informasjon.

2. Systembeskrivelse -- skisse

2.1 Skisse av systemarkitektur

Figur 1 viser overordnet hvordan FEIDE skal gi adgang til et felles datagrunnlag for fellesadministrative systemer. Utvalgt informasjon (en god del informasjon vil være rent lokal) fra FS og andre kilder, f. eks. Agresso, tilgjengeliggjøres enten ved at informasjonen lagres eller kopieres til sentrale databaser, eller ved at det gis tilgang til de lokale systemene. Svært mye av denne informasjonen vil ha adgangsbegrensinger, enten ved at bare visse grupper har adgang (f. eks. administrativt personale) eller ved at det er personopplysninger f. eks. om studenter. Brukerne skal ha tilgang til systemene gjennom et web-grensesnitt, i praksis vil det si bruk av standard nettlesere (evt. med nødvendig tilleggsprogramvare og utstyr, f. eks. for bruk av smartkort) mot tjenester som er tilgjengelige over Internett.

Figur 1: Skisse av systemarkitektur.



Det er tre sentrale sikkerhetsfunksjoner som må implementeres i et slikt system:

- Identifisering og autentisering (av brukere og tjenester),
- Autorisering, dvs. tildeling av rettigheter,
- Aksesskontroll, dvs. mekanismer som sikrer at all aksess er i henhold til autorisasjoner.

Det er viktig å merke seg at dette er tre separate funksjoner, men med klare avhengigheter. Tildeling av rettigheter gjøres vanligvis til enkeltpersoner eller definerte grupper. Aksesskontrollen må kunne sjekke operasjoner mot autorisasjonene, og vil da enten være avhengig av autentisering av brukeren, eller av en autentisering som sikrer bevis for gruppetilhørighet. (Dvs. mekanismer basert på aksesskontrollister (ACL). Det finnes alternativer, f. eks. kapabiliteter (capability based access control), men dette brukes i liten grad i praksis.) Autorisasjoner og aksesskontroll kan gjøres avhengig av en viss kvalitet på

autentiseringen, og det kan brukes tilleggsinformasjon som lovlige tidspunkter, lovlige maskiner / systemer for brukeren osv.

For *autentisering* er det verdt å merke seg at alle brukere må ha en unik identifikasjon mot systemet, dvs. alle studenter og ansatte i UoH-sektoren pluss potensielt «andre brukere», og at alle disse brukerne må kunne autentiseres ved tilgang til forskjellige tjenester fra i prinsippet vilkårlige klientmaskiner. Vi ser at det ikke er noen enkelt inngang (f. eks. en «front-end» tjener) til systemene, men at disse kan nås over forskjellige tjenester på forskjellige maskiner. Alle disse må kunne autentisere.

Autorisasjoner kan tildeles gjennom administrative systemer med web-grensesnitt, ved bruk av den samme arkitekturen som for tilgang til tjenester. Det er selvfølgelig ekstra viktig med god sikkerhet her.

Den viktigste egenskapen ved *aksesskontroll* er at den må være effektiv, dvs. ingen aksess må gå utenom mekanismene. Her er kvalitetssikring av systemene som helhet meget viktig, for å sikre at feil ikke åpner bakdører inn til sensitiv informasjon og operasjoner.

I tillegg til basisfunksjonene autentisering, autorisering og aksesskontroll er det i mange tilfelle også andre sikkerhetskrav, spesielt knyttet til beskyttelse (integritet og konfidensialitet) av informasjon som overføres over nettverk, spesielt over Internett. Det er også klare krav til logging og sporbarhet i systemene, og til funksjoner for oppretting dersom noe går galt (sikkerhetskopiering mm.). Det er krav om elektroniske signaturer for enkelte tjenester. Det er også en helt klar forutsetning at systemene håndterer personinformasjon på en tilfredsstillende måte. Som ett eksempel bruker alle administrative systemer innen UoH-sektoren fødselsnummer som nøkkel til informasjon.

Som en siste kommentar kan vi nevne at felles administrative systemer for sikkerhet er en inter-domene problemstilling. Hver UoH-enhet er autonom, og kan ha sin egen sikkerhetspolicy. Denne policyen kan i prinsippet være i konflikt med policyer i andre UoH-enheter. For de felles systemene er det klart at det må lages felles retningslinjer, felles policy, i stor grad. De enkelte enhetene kan da enten tilpasse seg for grenseflaten mot de felles systemene, eller intern policy kan endres til å samsvare med felles policy. Men dersom policy for felles systemer avviker sterkt fra det som er praksis innen de enkelte UoH-enhetene, har en helt klart skaffet seg et problem.

2.2 Alternativer for autentisering

Autentisering betyr å bevise at en oppgitt identitet er korrekt. Det finnes en rekke forskjellige mekanismer for autentisering, med forskjellige egenskaper, og forskjellig styrke på bevisene. FEIDE anbefaler bruk av offentlig-nøkkel systemer og sertifikater utstedt innen en PKI (Public Key Infrastructure). De viktigste argumentene for dette er skaleringsegenskapene til en PKI-basert løsning, tilgjengeligheten av produkter og tjenester, potensielt god sikkerhet, og det faktum at dette i dag er den eneste metoden for å få til åpne systemer for elektronisk signatur. (Terminologien er som følger: *Elektronisk signatur* er et teknologinøytralt begrep som skal dekke alle mulige mekanismer for signaturer på elektroniske dokumenter. *Digital signatur* er en elektronisk signatur produsert ved offentlig-nøkkel kryptografi. I dag er dette i realiteten eneste mulighet for elektroniske signaturer, slik at de to begrepene i praksis vil bety det samme.)

Vi går her kort gjennom alternative løsninger for autentisering, og grunnene til at disse er forkastet. Ett felles argument mot de fleste av alternativene er at de ikke gir signeringsmuligheter.

- Brukernavn / passord gir forholdsvis dårlig sikkerhet, men vil nok være fullt ut akseptabelt for svært mange anvendelser i en FEIDE-sammenheng. Men metoden skalerer ikke spesielt godt. Administrasjon av passord for UoH-sektoren som helhet vil være meget vanskelig, spesielt dersom passord må kopieres fordi det ikke er en enkelt inngang til systemene.

- Internett-bankene bruker forskjellige systemer for engangspassord, enten med tabeller for valg, eller med en «kalkulator» (DigiPass) som ved hjelp av kryptografiske metoder beregner passordet. Disse autentiseringsmetodene -- spesielt DigiPass -- er meget sikre. Problemene er først og fremst at det er liten grad av standardisering av løsningene, og at i hvert fall en DigiPass-løsning vil bli svært dyr (kalkulatorer til alle personer innen UoH-sektoren). Alle løsningene er basert på en sentral autentiseringstjeneste for en bedrift / tjeneste (en bank), og det kan være problematisk å bruke dem i en inter-domene situasjon.
- Kerberos [Steiner, Neuman og Schiller 1988] er en spesifikasjon av en autentiseringstjeneste basert på bare symmetrisk kryptografi og passord. Denne har ganske stor popularitet, og er bl.a. valgt av Microsoft for Windows 2000. Igjen er problemet at slike systemer virker godt innen en organisasjon, mens samvirke mellom (Kerberos-tjenester i) forskjellige organisasjoner er et mye svakere punkt. Det finnes spesifikasjoner som kombinerer bruk av Kerberos internt med PKI-baserte løsninger for autentisering mellom organisasjoner, men produktstøtte og erfaringer med slike løsninger er for begrenset til at vi tør å anbefale dette. Men Kerberos er et reelt alternativ der det ikke er behov for signaturer.
- Pretty Good Privacy (PGP) er et program basert på offentlig-nøkkel kryptografi uten bruk av noen PKI. Tillitsmodellen er «web of trust», der brukerne utsteder sertifikater for hverandre. Problemet med PGP i en FEIDE-sammenheng er først og fremst at PGP neppe kan skalere med tilstrekkelig tillit for et så stort antall brukere. En kan riktignok presse PGP inn i en slags infrastruktur ved å forlange bruk av noen tiltrodde sertifikatutstedere, men da har en egentlig laget en PKI, og systemer som er designet for PKI, vil antagelig være bedre. PGP har som et grunnleggende problem at tilbakekalling av sertifikater ikke støttes. Dersom en privat nøkkel er på avveie, kan det være vanskelig for eieren å beskytte seg mot misbruk Vår konklusjon er at PGP er et godt alternativ, men at en løsning med bruk av en (PKI) infrastruktur vil skalere bedre og gi bedre sikkerhet.
- SESAME (https://www.cosic.esat.kuleuven.ac.be/sesame/) tilbyr flere varianter for autentisering, både Kerberos-lignende mekanismer og mekanismer basert på offentlig nøkkel-kryptografi. (SESAME som sådan er et rammeverk for autentisering, aksesskontroll og relaterte tjenester, mer enn en spesifikk autentiseringsmekanisme, og i den forstand er det ikke uaktuelt i sammenheng med FEIDE.)
- Systemer basert på bruk av mobiltelefon og GSM-autentisering har vært nevnt. Vi skal ikke se bort fra at en her kan finne brukbare løsninger, men det finnes neppe noen ferdige spesifikasjoner eller produkter å ta utgangspunkt i, slik at arbeidet må starte nærmest fra bunnen av. En kan heller ikke forlange at alle personer skal ha mobiltelefon. Det arbeides med systemer der private (signerings-) nøkler lagres på en tiltrodd server, slik at en fra mobilt utstyr, over en sikker og autentisert kanal, kan initiere f. eks. en digital signatur. Slike systemer har noen uavklarte problemer knyttet til tillit til og sikkerhet av serveren, og spesielt det faktum at eieren ikke har full kontroll over nøklene. Dessuten er de ikke spesifisert i slik grad at de kan tas i bruk enda, og produkter mangler.
- Biometriske systemer fungerer først og fremst lokalt, hvor en kan sikre at det virkelig er en levende person til stede. Over et nett har biometri samme svakhet som passord. Biometri er imidlertid, på litt sikt, svært aktuelt som et alternativ til PIN-koder for å sikre at et smartkort ikke kan misbrukes av andre enn kortholderen selv. (M.a.o., biometri er ikke et alternativ til, men komplementerer smartkort og offentlig nøkkel-kryptografi.)

Et sterkt argument for PKI-baserte løsninger er at det begynner å bli relativt god støtte for dette i «hyllevare» programvare, både klient-programvare og vanlige tjener-plattformer. For web-baserte anvendelser er transportsikkerhetsprotokollene SSL (Secure Sockets Layer) og TLS (Transport Layer Security) nesten selvskrevet.

2.3 Noen krav og konklusjoner

Ut fra beskrivelsen så langt kan vi allerede sette opp en del konklusjoner angående valg av teknologi og løsninger. Disse er førende for arbeidet med resten av rapporten.

- Offentlig-nøkkel kryptografi og PKI infrastruktur brukes p.g.a. skaleringsegenskapene, behovet for digitale signaturer, og tilgjengelighet av produkter og tjenester.
- Smartkort brukes som bærer av private nøkler mm. Spesielt studenter er mobile, og alternativer for lagring av private nøkler er knyttet til lagring på spesielle maskiner. Alle studenter skal ha et fysisk studiekort, som f. eks. ved UiO allerede har smartkortfunksjonalitet. Denne kombinasjonen søkes opprettholdt.
- Smartkort og sertifikater brukes kun til autentisering, ikke til å bære autorisasjoner. Autorisasjoner og aksesskontroll ligger i systemene. Bruk av attributtsertifikater / privilegiesertifikater kan være en framtidig utvidelse, men dette tas ikke med nå.
- Viktigste anvendelser er autentisering og beskyttelse av informasjon under overføring. For web-aksess gjøres dette vanligvis ved SSL / TLS, og dette må støttes.
- FEIDE skal kunne tilby digitale signaturer -- dette er et krav i noen sammenhenger.
- Smartkortet skal i prinsippet kunne brukes i vilkårlige maskiner tilkoplet Internett. Dette reduserer sikkerheten, men gir nødvendig fleksibilitet.
- Bruk av fødselsnummer til identifikasjon er ikke akseptabelt selv om de administrative systemene er avhengige av tilgang til fødselsnummer.
- Bruk av FEIDE for adgangskontroll til utstyr, nettverk og bygninger kan være aktuelt, og løsningene bør i hvert fall ikke være til hinder for slik bruk.

3. Omgivelsene -- lover, regelverk, produkter og tjenester

3.1 Lover og regelverk

Det meste av eksisterende og pågående arbeid med lover og regelverk innen digitale signaturer og sertifikattjenester har liten betydning for FEIDE, og vi legger derfor ikke så stor vekt på området her. Men siden det er en del misforståelser om omfanget av det reguleringsarbeidet som finner sted, tar vi med litt forklaring.

Det er imidlertid meget viktig å få fram at FEIDE må ha et meget bevisst forhold til Datatilsynets retningslinjer. Det er godt mulig at vi burde ha hatt en dialog med Datatilsynet allerede i dette forprosjektet, men det har det ikke blitt tid til. Datatilsynet bør få tilsendt eller gjøres oppmerksom på denne rapporten, og videre arbeid med FEIDE bør foregå med dialog med Datatilsynet.

Fokus på arbeid med lover og regelverk både i Norge og EU er lagt på elektroniske signaturer. Norge er langt framme i denne prosessen. EU har vedtatt et direktiv om elektroniske signaturer [1999/93/EC]. Her defineres et kvalifisert sertifikat som et sertifikat som understøtter en elektronisk signatur som kan betraktes som ekvivalent til en håndskrevet signatur (kvalifisert elektronisk signatur) innen et nærmere angitt lovområde. Det stilles særskilte krav til innholdet i sertifikatene, til tilbydere av sertifikattjenester, og til utstyret/programvaren som har produsert signaturen. EU-direktivet skal implementeres i medlemslandene, inkludert EØS-området. Deler av det norske arbeidet, spesielt på lovsiden, er derfor «obligatorisk».

Merk at vi i Norge (og de fleste andre europeiske land) har fri avtaleslutning og fri bevisførsel ved tvister. Det er derfor ikke snakk om at kvalifiserte sertifikater / signaturer skal være eneste løsning, eller at andre former for elektronisk korrespondanse med eller uten signaturer ikke vil være «gyldig». Men det kan tenkes anvendelser der elektroniske dokumenter bare blir akseptert dersom de har en kvalifisert signatur. Hovedpoenget er at en kvalifisert signatur skal være sterk nok til at den kan brukes alle steder hvor lover, forskrifter eller avtaler krever signatur (uten at det er spesielle formkrav eller lignende).

EU-direktivet og den foreslåtte lov om elektroniske signaturer i Norge gjelder bare for kvalifiserte signaturer og utstedelse av kvalifiserte sertifikater. Kvalifisert signatur vil stille krav som neppe er forenlige med andre krav i FEIDE. Spesielt er det vanskelig å tenke seg at en skal kunne produsere slike signaturer på vanlig (PC-)utstyr. Vi diskuterer dette litt nærmere i vedlegg A og C. Vi kan derfor trygt konkludere med at FEIDE ikke vil tilby kvalifiserte signaturer og sertifikater, og vi vil derfor ikke bli underlagt lov om elektroniske signaturer og tilhørende bestemmelser.

Dersom en anvendelse i FEIDE trenger signaturer av høy kvalitet (se vedlegg A for en diskusjon), kan dette løses ved «kioskløsninger» der egne «fiklesikre» maskiner brukes på klientsiden. Anvendelsen vil bare være tilgjengelig fra disse spesielle maskinene (maskinene autentiseres mot tjenesten), som må være fullstendig tiltrodde når det gjelder hvilken programvare de kjører. Kombinert med smartkort for brukerne vil dette gi en løsning der både anvendelsen og brukerne kan være sikre på at det som signeres faktisk er det korrekte datagrunnlaget.

Arbeids- og administrasjonsdepartementet har nedsatt et utvalg som skal se på det offentliges behov for TTP-tjenester og sertifikater. Her kan det komme konklusjoner som kan berøre FEIDE, siden det meste av UoH-sektoren er offentlig, og siden det offentlige trenger å definere hva som gjelder på grenseflaten mot privat sektor. Det er usannsynlig at det vil komme konklusjoner fra utvalget som vil legge noen som helst hindringer i veien for FEIDE.

3.2 Produkter og tjenester

3.2.1 Sertifikattjenester

Dersom en skal ha en viss kvalitet på en sertifikattjeneste, og det bør FEIDE ha, stilles det meget strenge sikkerhetskrav. Det er derfor helt usannsynlig at UNINETT eller andre innen UoH-sektoren vil fysisk etablere og drive en slik tjeneste. Tjenesten må leies fra en profesjonell leverandør. Eventuelt kan en ha en løsning med flere leverandører, der hver enkelt UoH-enhet kan velge den de ønsker, men erfaringer med samtrafikk mellom forskjellige tjenester og produkter tilsier at en i hvert fall i en startfase bør velge bare en leverandør.

Det er i dag fire leverandører av sertifikattjenester i Norge. Vi oppgir også i parentes leverandør av utstyr og programvare til tjenesten, fordi dette faktisk kan ha betydning for valg av brukerprogramvare og utstyr: Telenor Bedrift (Entrust, Canada), Posten SDS (ID2, Sverige), BBS (Baltimore, Irland) og Fellesdata (SSE (eies av Siemens), Irland). Fellesdatas tjeneste har en litt usikker skjebne p.g.a. fusjonen med EDB-gruppen som kontrolleres av Telenor.

Vi antar at det neppe er aktuelt for FEIDE å gå til en utenlandsk leverandør, men en bør jo være oppmerksom på innkjøpsprosessene i forhold til EØS-regelverket og regler for når det skal være internasjonale anbud. Tjenester som Verisign og Thawte er også uaktuelle så lenge en legger opp til bruk av smartkort.

Som vi har antydet: Forskjellig utstyr kan ha forskjellige «særegenheter» som gjør at sertifikater som er produsert av en type utstyr, ikke nødvendigvis går sammen med brukerprogrammer mm. som er tilpasset annet utstyr for sertifikatproduksjon. Det er viktig å gjøre praktisk utprøving før en bestemmer seg endelig for valg av tjeneste.

En må skille mellom sertifikat*utsteder* og sertifikat*produsent*. Leverandørene kan (og ønsker gjerne) levere sertifikater der de selv står som utsteder. Men det er fullt mulig å ha tjenester der «FEIDE» (eller UNINETT, eller den enkelte UoH-enhet) står som utsteder, mens sertifikatene faktisk er produsert av en (mer eller mindre usynlig) tjenesteleverandør. Det er produksjonen som er sikkerhetskritisk.

3.2.2 Brukerprogramvare mm.

Leverandører av utstyr for sertifikatproduksjon tilbyr også utvalg av klientprogramvare for forskjellige formål. Her finnes det selvfølgelig også mye fra andre leverandører, og også en del gratisprogrammer. Men det er viktig å teste ut at det en ønsker å velge, faktisk virker mot den sertifikattjenesten og de smartkortene som en ønsker å bruke.

Entrust er PKI-bransjens svar på Microsoft. Entrust lager sine egne spesifikasjoner og har fått ganske mange tredjepartsleverandører til å følge disse, mot å utstede et «Entrust ready» stempel. Dette gir jo en garanti for at produktene virker sammen, men til gjengjeld er det mindre sjanser for at dette spiller sammen med produkter som ikke er Entrust ready, men som ellers forholder seg til de samme standardene. Entrust-spesifikasjonene «spikres», og med Entrust (ready) løsninger har en lite spillerom og fleksibilitet. Til gjengjeld har en altså noe som ganske garantert virker, og tilbudet av programvare er temmelig stort.

Andre leverandører er mer fleksible med hensyn til å tilpasse løsninger til de ønskede spesifikasjonene, men dette fører jo også til mer arbeid for å få ting til å spille sammen, og en vil også gjerne få problemer i forhold til den delen av verden som kjører etter Entrusts spesifikasjoner.

Det finnes en rekke spesifikasjoner og standarder som programvaren må eller bør støtte.

3.2.3 Smartkort

Samtlige leverandører av sertifikattjenester, med et mulig unntak for Fellesdata, har tjenester som forutsetter bruk av smartkort, og kan inkludere smartkort (fra underleverandører) i en leveranse / kontrakt. I tillegg finnes det en rekke andre leverandører av kort og utstyr (kortlesere, utstyr for fysisk trykking eller preging av kort, utstyr for å legge informasjon ned på kortene, osv.), som Bull, Strålfors, Unikey, Norsik, Giesecke & Devrient, Telenor Conax og sikkert flere, og disse igjen representerer bare et utvalg av mulige kortprodusenter på verdensbasis (så vidt vi vet er det bare Conax som er en rent norsk leverandør).

Dersom vi forutsetter at sertifikatprodusenten, i hvert fall for noen UoH-enheter, vil få ansvaret for å legge informasjon på smartkortene, må FEIDE sikre seg at sertifikatprodusenten faktisk kan håndtere de valgte kortene (eller velge kort fra den mengden alternativer som sertifikatprodusenten kan takle).

Som vi skal se mer på seinere, bør smartkortene som velges, ha rom for flere anvendelser. Dette tilsier et multifunksjon smartkort. Det er to standard operativsystemer for slike kort, i tillegg til en rekke proprietære løsninger: Multos og Java Card. Ved valg av smartkort bør en først og fremst se på disse to alternativene.

Det er flere standarder og spesifikasjoner for grensesnitt mot kortene mm. som er aktuelle.

3.3 Andre prosjekter

3.3.1 Forvaltningsnettsamarbeidet

Forvaltningsnettsamarbeidet er et sett av felles innkjøpsavtaler for hele norsk offentlig sektor. Det finnes en rekke avtaler for alt fra PCer og maskinvare, via forskjellige typer programvare, til nettverks-komponenter og nett- og teletjenester. Det inngås rammeavtaler med utvalgte leverandører, og kunder innen offentlig sektor kan handle over disse avtalene til nærmest ferdigforhandlede (og gunstige) betingelser.

En av rammeavtalene er innen området digital signatur, meldingskryptering og TTP-tjenester, og denne omfatter smartkort, kortlesere, programvare for digital signatur og meldingskryptering, og tjenester for utstedelse av sertifikater. Det er sju leverandører som har rammeavtale, der tre stykker er totalleverandører, dvs. inkludert sertifikattjeneste. Dette er Telenor Bedrift, Posten SDS og Strålfors / Merkantildata (Merkantildata bruker Posten SDS' sertifikattjeneste, og er derfor ikke med i lista over leverandører i 3.2.1). Giesecke & Devrient leverer alt unntatt sertifikattjenester, Unikey og Norsik leverer kun smartkort og lesere, mens Bull bare leverer kortlesere. De tre totalleverandørene har inngått en kryssertifiseringsavtale slik at sertifikater utstedt av en av dem skal kunne godkjennnes hvor som helst. Det er utarbeidet en egen sertifikatpolicy (felles for alle leverandører) som tilfredsstiller meget strenge krav til kvalitet.

NR har vært meget sentral i arbeidet med disse rammeavtalene som innleid konsulent. Dette omfattet kravspesifikasjon, forhandlinger med leverandørene, sertifikatpolicy og diverse notater om navngivning, veiledninger osv.

I forhold til FEIDE har vi fått avklart med Statens Forvaltningstjeneste / Statskjøp, som forvalter avtalene, at UoH-sektoren kan handle over rammeavtalene. Dette skal også gjelde for private høgskoler. Dette er ett av flere alternativer for FEIDE, men FEIDE må muligens gjøre noen endringer i forhold til kravspesifikasjoner for det som tilbys i Forvaltningsnettsamarbeidet. Det må avklares hvor store endringer som er tillatt, men noe kan ordnes gjennom tilleggsspesifikasjoner. Spesifikt har vi:

- Navngivningen i sertifikatene knytter en person til en jobb i en offentlig virksomhet, og det er ikke brukt noen unik identifikator i navnet. Knyttingen mot jobb (som kan være studiested) er kanskje ikke ønskelig.

- FEIDE har større behov for autentisering ved SSL / TLS o.l. enn for digitale signaturer -- selv om dette også trengs. Her trengs det tilleggsspesifikasjoner for å dekke opp SSL / TLS autentisering.
- Dersom en baserer seg på bruk av studiekortene, trenger en trykking / preging av kortene. Dette inngår ikke i avtalene, men kan leveres av alle kortleverandører.
- Multifunksjon smartkort er ønskelig -- dette kan leveres, men ekstra funksjoner krever tillegg.

3.3.2 BankID

Bankene i Norge (dvs. de som er medlemmer i Finansnæringens Hovedorganisasjon eller Sparebankforeningen) har inngått et samarbeide om en felles løsning for elektronisk identifikasjon og sertifikater, kalt BankID. Dette innebærer et samarbeide på samme nivå som for bankkort mm. i dag, dvs. at en BankID for en kunde i en bank vil bli gjenkjent og akseptert av andre banker. Endelig tilslutning fra de enkelte bankene til opplegget mangler, men er sannsynlig. Spesifikasjonene for tjenesten er på det nærmeste ferdig. BBS er sannsynlig leverandør av sertifikattjenestene. BankID vil bli markedsført som et varemerke, på linje med f. eks. BankAxept. NR har vært inne i dette prosjektet i en kvalitetssikringsrolle for de tekniske spesifikasjonene.

I første omgang er BankID en programvareløsning, dvs. at nøkler og sertifikater installeres på brukernes maskiner og ikke på smartkort. Men bruk av smartkort er et klart strategisk mål på litt sikt.

Anvendelsesområdet og problemstillingene for BankID er svært like det bildet vi ser for FEIDE: Aksess til felles tjenester fra et stort antall personer. Det er derfor naturlig at FEIDE ser på BankID som alternativ løsning, enten direkte, eller ved gjenbruk av store deler av spesifikasjonene.

BankID har en navngivning som identifiserer enkeltpersoner med navn og en unik identifikator. De har med «Organisasjon=BankID» i navnet for å identifisere varemerket, og utsteder av sertifikatet vil være brukerens bank. Hver bank vil ha en egen (logisk sett) sertifikattjeneste, som for de aller fleste vil bli drevet av en sertifikatprodusent (BBS?). Disse tjenestene vil operere etter en felles sertifikatpolicy, og vil ligge under en felles rot, slik at sertifikater fra alle banker vil bli godkjent overalt i systemet.

Forholdet til utenlandske banker i Norge og til utenlandske grener av norske banker er ikke helt avklart. Per i dag er det tenkt at bare kunder i norske banker skal kunne få BankID.

3.3.3 Postens Elektroniske ID

Posten SDS har utviklet «Postens Elektroniske ID» som er et personlig sertifikat med smartkort som bærer av privat nøkkel for privatpersoner. Posten SDS legger vekt på at flere funksjoner skal kunne legges inn i smartkortene (Multos plattform), og har bl.a. demonstrert kombinasjonen av elektronisk ID og betaling med Mondex elektroniske kontanter fra samme kort. NR har vært leid inn av Posten SDS til diverse arbeid med utviklingen av tjenesten, spesielt knyttet til bruk av smartkort.

Navngivningen er navn og unik identifikator for person, og utsteder er Posten. Tjenesten er kryssertifisert (eller vil bli det) med tilsvarende tjenester i Sverige, Finland og Irland. Basert på dette arbeider den internasjonale postunionen med et verdensomspennende system med en «postverk sertifikattjeneste» i hvert medlemsland (ca. 180 stykker) under en felles rot.

Postens Elektroniske ID er et klart alternativ for FEIDE.

3.3.4 Telenor Zebrasign

Vi har ikke undersøkt nærmere hvilke tilbud Telenor Bedrift har fra sin tjeneste, men det er svært sannsynlig at også denne leverandøren kan ha en tjeneste som kan passe for FEIDE, eller at de kan tilpasse en tjeneste.

3.3.5 Universitetet i Oslo studiekort

Studentsamskipnaden i Oslo (SiO) har allerede et pilotprosjekt med studiekort med smartkortfunksjonalitet. I dag inneholder smartkortdelen kun et lukket, midlertidig småpengesystem. Kortet ser ellers ut som et vanlig studiekort, men er utstyrt med to magnetstriper -- en for adgangskontroll til bygninger og en for bankkort for de studentene som ønsker dette (kun Kreditkassen) -- og en strekkode for biblioteket, i tillegg til smartkortdelen.

Det er også andre UoH-steder som har sett på og eksperimentert med smartkortteknologi, men initiativet i Oslo er så vidt vi vet det som er kommet lengst, og det som kommer til å få størst innflytelse, også på grunn av UiOs størrelse.

SiO (og UiO) ser et stort potensiale i smartkortteknologien, og ønsker å utvide funksjonaliteten. De mest aktuelle funksjonene er en elektronisk ID og et åpent (dvs. koplet mot banker og virkelige penger) småpengesystem. De ønsker helt klart et multifunksjon smartkort basert på åpen teknologi, dvs. Multos eller Java Card. SiO er også svært interessert i å fortsette samarbeidet med Kreditkassen, og evt. andre banker, med bankkortfunksjonalitet i studiekortene. SiO uttaler at de er åpne for å bytte ut alle kort som er utstedt, med kort etter endrede spesifikasjoner og med mer funksjonalitet, men de er lite interessert i å ta denne kostnaden mer enn en gang. Elektronisk ID er interessant av samme grunner som FEIDE oppgir, men de ser også anvendelser som adgang til deler av en studentportal som ganske snart vil bli lansert på Internett, og betaling av semesteravgift fra småpengesystemet kombinert med autentisering.

SiO ser ut til å lande på Proton som småpengesystem, siden det er dette systemet som er valgt av bankene i Norge. Proton kan bare gis til personer som er kunde av bankene. Bankene har ikke konkrete planer om kombinasjon av BankID og Proton, men det må antas at slike planer vil komme ganske raskt når BankID kommer på smartkort. Til sammenlikning har Posten SDS allerede demonstrert kombinasjonen av Mondex betalingssystem og Postens Elektroniske ID, men her er problemet at Mondex har en høyst usikker framtid som betalingssystem i Norge.

Det er helt klart at den elektroniske IDen som havner i SiOs (og UiOs) smartkort, må være den samme som FEIDE anbefaler. I motsatt fall har en kjørt seg opp i en blindgate. Det har derfor vært svært mye kontakt mellom NR og SiO i prosjektfasen, og NR har sammen med SiO også hatt møter med både Kreditkassen og BBS for å ta opp forholdet til BankID og andre initiativer.

Ett viktig spørsmål som må avklares mellom UNINETT og SiO / UiO, er om en skal standardisere en smartkortløsning for UoH-sektoren i Norge, og hvem som i tilfelle skal lage disse spesifikasjonene.

4. Autentiseringssertifikater: elektronisk legitimasjon

4.1 Sertifikater og sertifikatautoriteter

En sertifikatautoritet (SA) er en type TTP-tjeneste. Et sertifikat er en «melding» signert av en SA. Sertifikatet godkjennes ved hjelp av SAens offentlige nøkkel, som en da må få tak i på en måte som gjør at en kan stole på den. Typisk ligger denne i smartkortene, eller den kan distribueres på en annen måt som gjør at en kan stole ubetinget på den.

Alle aktuelle sertifikattjenester i dag bruker sertifikatformatet X.509v3 [X.509]. X.509 er en del av standardene for katalogsystemer, utarbeidet i fellesskap av Den Internasjonale Standardiseringsorganisasjonen (ISO) og den Internasjonale Telekommunikasjonsunionen (ITU). X.509 er et rammeverk for autentisering, inkludert spesifikasjon av noen alternativer for autentiseringsprotokoller. Den biten av standarden som har fått gjennomslag, er likevel sertifikatformatet, der siste utgave er versjon 3 (X.509v3).

Et autentiseringssertifikat inneholder som minimum et navn, en offentlig krypteringsnøkkel, identiteten til SAen, et tidsintervall som angir gyldighetsperiode, versjonsnummer (X.509v3), algoritmeidentifikasjon (for SAs signatur og for sertifikatets offentlige nøkkel) og serienummer for utstedelse (se Figur 2). I tillegg finnes det en rekke former for tilleggsinformasjon som kan tas med i sertifikatet, og en SA kan lage egne utvidelser til formatet (private extensions). Et sertifikat bør (skal) kun inneholde åpen informasjon, men endringer av informasjonen vil bli oppdaget siden det er signert av SA. Et sertifikat kan derfor utveksles og oppbevares på vilkårlige måter. (Navnet i sertifikatet er ofte virkelig identifikasjon av en person, men kan også være f.eks. bedrift, organisasjon, gruppe, rolle, pseudonym, datamaskin, program, osv. -- i prinsippet alt som kan gis et navn.)

Figur 2: Struktur på X.509-sertifikat

- Sertifikat-innhold
  - Identitet (DN)
  - Offentlig nøkkel og algoritme ID
  - Gyldig fra
  - Gyldig til
  - SAs identitet (DN)
  - Serienummer
- Signert av SA

En sertifikattjeneste defineres gjennom elementene:

- *Tillitsmodell* definerer hvordan SA forholder seg til andre SAer.
- *Sertifikatpolicy* er en sikkerhetspolicy som angir nivå for tjenesten gjennom beskrivelse av de overordnede kravene som må være oppfylt.
- *Sertifikatpraksis* er konkrete spesifikasjoner for hvordan sertifikatpolicy er implementert for en gitt tjeneste.
- *Sertifikatformat*, inkludert navngivning.
- Opplysningstjenester / *katalog* for distribusjon og lagring av sertifikater mm.

Alle disse elementene beskrives nedenfor.

4.2 Gyldighet og tilbakekalling

Et sertifikat utstedes med en gyldighetsperiode. Den som mottar et sertifikat, plikter å sjekke at det ikke er utløpt. Men sertifikater kan også trekkes tilbake før utløpstida. Årsaker til dette kan være f. eks. at smartkort med private nøkler er mistet eller ødelagt, at en person har sluttet i et arbeidsforhold, eller i mer dramatiske tilfeller at det er mistanke om at private nøkler eller smartkort er kompromittert eller misbrukt.

En SA plikter derfor med jevne mellomrom å utstede tilbakekallingslister (CRL, engelsk Certificate Revocation List, også definert i X.509 standarden). Ei tilbakekallingsliste inneholder serienummer på sertifikater som er tilbakekalt, med tidspunkt for tilbakekalling og eventuelt også årsak. Lista er signert av SA, og må publiseres på et velkjent sted, typisk i en LDAP-katalog. Det brukes vanligvis komplette tilbakekallingslister, dvs. at ei liste tar med alle sertifikater som var i forrige liste, pluss de sertifikatene som er tilbakekalt siden sist. Sertifikater kan fjernes fra listene når tidspunktet har passert utløpstida -- da vil de uansett bli forkastet av en bruker. (Det framgår av dette at bruk av sertifikater krever rimelig korrekte klokker i systemene. Det er imidlertid ikke noen strenge krav, gitt levetid for sertifikater, frekvens for tilbakekallingslister osv.)

Tilbakekallingslister kan utstedes med frekvens fra en time [FSP-1] til en måned [UNISA-policy]. En SA kan utstede tilbakekallingslister før oppgitt tidspunkt. Med månedlige lister vil en fort bli nødt til det.

En mottaker av et sertifikat plikter å hente siste utgave av tilbakekallingslista for den SAen som er utsteder av sertifikatet. I motsatt fall må mottakeren selv bære risikoen. Det går an å lagre tilbakekallingslister lokalt, og hente bare når det blir utstedt en ny. Ei tilbakekallingsliste har alltid med en tidsangivelse for når neste liste er planlagt. Men ved lokal lagring vil en ikke få med seg lister som utstedes før tida, siden tjenester for automatisk distribusjon av tilbakekallingslister ikke er i bruk i dag.

Effektiv tilbakekalling av sertifikater er et stort problem i sertifikatsystemer, og kan vanskelig gjøres uten en skikkelig infrastruktur. Kanskje den største svakheten med PGP er at tilbakekalling ikke støttes.

Det er nå definert et alternativ til tilbakekallingslister gjennom OCSP-protokollen (Online Certificate Status Protocol) [RFC 2560]. Dette er en tjeneste der en kan spørre om status (gyldig, tilbakekalt mm.) for et gitt sertifikat. OCSP er i liten grad tatt i bruk enda, men kan på sikt komme til å erstatte tilbakekallingslister.

Et annet (framtidig) alternativ kan være tiltrodde tjenester for godkjenning av signaturer. En kan sende kopi av meldingen til tjenesten, og få tilbake et ja / nei svar på om signaturen er i orden. Mottakeren må selv dekryptere meldingen, siden tjenesten bare kan dekryptere hvis den selv står som mottaker. En slik tjeneste kan potensielt være mer avansert med tanke på de sjekkene den kan gjøre i forhold til policyer osv. Det mest aktuelle er å ha en tjeneste internt i en virksomhet, men en kan også tenke seg eksterne tjenester.

4.3 Sertifikatpolicy

Sertifikatpolicy er egentlig en sikkerhetspolicy som beskriver overordnede og av og til mer detaljerte krav til sertifikattjenesten. Sikkerheten som tjenesten er underlagt, gir grad av tiltro brukere kan ha til sertifikater utstedt av tjenesten. Policy kan defineres av den enkelte SA. Typisk kan hver SA ha flere forskjellige klasser av (kvalitet på) sertifikater, eller sertifikater for forskjellige formål utstedt under forskjellige policyer. Men policy kan også være pålagt av en eller annen myndighet. Ett eksempel er Forvaltningsnettsamarbeidet i Norge, der tre SAer har kontrakt, og er pålagt å følge den samme sertifikatpolicyen [FSP-1], spesifisert av Arbeids- og administrasjonsdepartementet.

Sertifikattjenester kan ha vidt forskjellige sikkerhetsnivåer. Tjenester som skal understøtte formelle digitale signaturer (kvalifiserte sertifikater), må være meget sikre. Men andre tjenester kan ha helt andre nivåer, som f. eks. UNINETTs UNISA-tjeneste for pilotprosjekter innen forskning og utdanning.

Andre eksempler er sertifikattjenester på Internett som Verisign og Thawte. Felles for disse er at de er basert på kun elektronisk kontakt mellom sertifikattjeneste og bruker og nøkkellagring på disk. (En del leverandører, f. eks. Verisign, tilbyr forskjellige klasser av sertifikater, der klasser for høy tillit krever prosedyrer som gir en forholdsvis solid autentisering av brukeren. I praksis brukes ikke disse klassene for privatpersoner.) Nøkler og sertifikater brukes primært sammen med nettlesere, for klientautentisering i SSL (Secure Sockets Layer) og signering / kryptering av epost. Sertifikattjenestene har betalt en bra sum for å få sine offentlige nøkler lagt inn i distribusjonsutgavene for Netscapes og Microsofts web-lesere (se 4.8 for mer om dette).

Det følger av dette at det vil eksistere svært mange forskjellige sertifikatpolicyer, som delvis vil avvike sterkt fra hverandre, delvis være tilnærmet overlappende. En sertifikatpolicy er et skriftlig dokument, og ikke egnet til behandling av programvare. En policy skal imidlertid tilordnes et navn, i form av en objektidentifikator, OID. (Standard for tilordning er definert av ISO, og i Norge er Post- og teletilsynet myndighet for tildeling.) Denne OIDen vil vanligvis være med i sertifikatet i en definert attributt (certificate policies extension), og sertifikatet kan også ha en peker til policy f. eks. gjennom en URL.

En skal derfor kunne identifisere hvilken policy som gjelder for et gitt sertifikat, og kunne gjøre seg opp en mening om tilliten til sertifikatet. Dette er egentlig kun teori, siden en sertifikatpolicy er et potensielt svært omfattende og teknisk dokument som ikke vil være forståelig for en vanlig bruker. Det kan dessuten være skrevet på f. eks. russisk, og inneholde referanser til lover og regelverk som er totalt ukjente for brukeren. Det er mulig å sette opp brukerens system slik at det har identifisert OIDer for de policyene som kan godkjennes, men dette blir fort meget komplisert for brukerne. Et bedre alternativ er å konfigurere godkjenning av policyer inn for en OCSP-tjeneste, som kan være felles for alle brukere innen en organisasjon. Hierarkier og kryssertifisering med policy-mapping (se 4.8) bidrar også til å bestemme nivå på en policy. Men det er nødvendig med systemer for å kategorisere sertifikatpolicyer. Kvalifiserte sertifikater, som er et definert nivå for å understøtte formelle digitale signaturer, er ett eksempel på dette. Viktigste opplysning i en policy vil da være at den støtter kvalifiserte sertifikater -- detaljene er uvesentlige.

Det er definert et anbefalt dokumentformat for sertifikatpolicyer [RFC 2527]. En policy behøver ikke å stille krav på alle områder, og den kan stille ytterligere krav. De viktigste elementene som kan være med, er:

- Formål og anvendelser,
- Forpliktelser, ansvar, jurisdiksjon, personvern, eventuelt også økonomiske forhold,
- Internkontroll eller ekstern revisjon for å verifisere at tjenesten drives ihht. policy og praksis,
- (Krav til) prosedyrer for identifisering og autentisering i forbindelse med sertifikatutstedelse (se også 5), navngivning og prosedyrer for tilbakekalling,
- Operasjonelle krav til hvordan sertifikatutstedelse, tilbakekalling, loggføring, arkivering, bytte av SAs egen nøkkel mm. skal foregå,
- Krav til publisering (kataloger mm.) av sertifikater, tilbakekallingslister og annen informasjon,
- Katastrofeplaner for SA og regler for terminering av tjenesten eller overføring av tjenesten til en annen operatør,
- Fysisk, logisk og administrativ sikkerhet knyttet til tjenesten, inkludert krav til personell og utstyr,
- Tekniske sikkerhetstiltak knyttet til behandling (livssyklus fra generering til ødeleggelse) av private og offentlige nøkler -- kan f. eks. inkludere krav om at brukerne skal ha smartkort,
- (Deler av) profiler for sertifikater og tilbakekallingslister,
- Ansvar og administrasjon av selve policyen, inkludert prosedyrer for endringer.

4.4 Sertifikatpraksis

Sertifikatpraksis spesifiserer hvordan en sertifikatpolicy er implementert for en gitt SA. Sertifikatpraksis er anbefalt å følge samme dokumentstruktur som policy. Den kan utfylle policyen der policy ikke har spesifisert krav, men vil ellers være en ytterligere presisering som ikke kan være i konflikt med policy. Ett eksempel kan være at policy setter et generelt krav til personlig frammøte og legitimering. Sertifikatpraksis kan da si at frammøte skal være på et postkontor eller til et landpostbud (realistisk scenario for Postens Elektroniske ID), og at dette skal være når en søker om/bestiller sertifikat og smartkort (alternativt at bestillingen kan være over telefon, epost eller annet, men at en må møte personlig fram for å hente smartkortet).

Vi går ikke nærmere inn på sertifikatpraksis her. Merk at mange SAer ikke har skilt policy og praksis i separate dokumenter, men kaller alt sammen sertifikatpolicy (eller sertifikatpraksis).

4.5 Sertifikatprofil

X.509v3 sertifikatformat er en omfattende spesifikasjon med mange valgmuligheter. Det er derfor nødvendig å lage sertifikatprofiler for å velge ut hvilken informasjon som skal inngå i sertifikatene, og hvordan informasjonen skal kodes. Det er ikke anbefalt å ta med for mye informasjon i sertifikater, og spesielt er det ikke anbefalt å ta med annet enn informasjon med lang levetid. Sertifikatprofilen kan helt eller delvis bestemmes av sertifikatpolicyen. Den vanskeligste delen av profileringen er navngivningen, som diskuteres separat nedenfor.

Det er ikke gitt at brukerne bare skal ha ett sertifikat hver. Eksempelvis krever Forvaltningsnettsamarbeidets policy [FSP-1] at hver bruker skal ha tre forskjellige sertifikater med forskjellige formål og nøkler:

- Autentisering for pålogging til lokalt system, eller til tjenester som nås f.eks. over Internett,
- Digital signatur for signering av meldinger og dokumenter,
- Utveksling av symmetriske nøkler for kryptering.

Skillet med separat nøkkelpar for hver funksjon har sikkerhetsmessige årsaker. Som eksempel: En autentiseringsprosess foregår gjerne ved at brukeren får en utfordring som hun signerer over. En angriper kan sørge for at utfordringen er en hashverdi for en melding, som brukeren dermed faktisk har signert. Dersom sertifikatet som skal brukes for å godkjenne signaturen, sier at sertifikatet bare er gyldig for autentisering, men ikke signering, vil signaturen være ugyldig og derfor unyttig for angriperen. (De fleste aktuelle autentiseringsprotokoller er heldigvis ikke så naive at de åpner for denne typen angrep.)

For kvalifiserte signaturer vil det derfor bli krevet at nøkkel og sertifikat ikke skal kunne brukes til noe annet enn digital signatur, men spesifikasjoner for kvalifisert signatur vil ikke si noe om at en trenger mer enn to nøkkelpar -- ett for signatur og et annet for andre formål.

Andre spesifikasjoner, spesielt Entrust, sier to sertifikater, med et eget for nøkkelutveksling/kryptering og et annet for alle andre formål. Dette skyldes at en kan ha behov for (eller det kan være lovpålagt i enkelte land -- ikke i Norge) å oppbevare en sikkerhetskopi av privat nøkkel for krypteringsformål. Dette er unødvendig og uaktuelt for autentiserings- og signeringsnøkler.

Vi ser at det er konflikt mellom Entrust-spesifikasjonene og kravene til kvalifiserte signaturer, og Forvaltningsnettsamarbeidet (og noen andre) har søkt å løse dette ved å kreve tre sertifikater. Dette er den ideelle løsningen, men har dessverre begrenset produktstøtte.

Lovlig bruk av den offentlige nøkkelen i et sertifikat kodes i sertifikatet. (Key usage extension som består av et antall flagg som kan settes. Eksempler på andre flagg er: Tillatelse til å utstede sertifikater for brukere, tillatelse til å utstede sertifikat for underordnede SAer.) Alle nøkler og sertifikater utstedes typisk under ett,

og legges på samme smartkort. En kan selvfølgelig ikke garantere at programvare som behandler sertifikatene, holder seg til korrekt bruk, men programvaren bør jo gjøre det.

I likhet med fysiske legitimasjonsbevis utstedes sertifikater med en begrenset levetid, typisk to år. Ved utløp av denne perioden må brukeren skaffe seg nye sertifikater. Nøkler kan gjerne ha lenger levetid enn sertifikatene, fem år er vanlig. Nye sertifikater kan utstedes for samme offentlige nøkler, og brukeren trenger da ikke nytt smartkort. Dersom en skifter nøkler, enten fordi disse også er utløpt eller av andre grunner, må en vanligvis få nytt smartkort. (Noen kort tillater at nye nøkler legges inn i kortet.)

SA kan sende et varsel til en bruker, f. eks. som epost, når brukerens sertifikater nærmer seg dato for utløp. En del programvare vil gi brukeren samme type varsling på skjermen når programvaren oppdager at det er kort tid til utløp av et sertifikat.

Et annet spørsmål er adgangen til å bruke samme offentlige nøkkel i flere sertifikater i ulike sammenhenger, slik at brukere kan greie seg med en privat nøkkel og ett smartkort. En sertifikatpolicy kan f. eks. si at dersom en bruker signerer en sertifikatforespørsel med en «kvalifisert signatur» (eller et liknende krav), så kan nytt sertifikat utstedes uten personlig frammøte for en offentlig nøkkel som finnes fra et sertifikat som brukeren selv legger ved. Denne prosedyren må sikre at det ikke oppstår konflikter med anvendelsesområde for sertifikatene -- en nøkkel for digital signatur skal ikke brukes til noe annet.

Eksempelvis kan en utstede, som en offentlig tjeneste, borgerkort til privatpersoner. Dette må være en tjeneste som gir sertifikater av høy kvalitet. Ved hjelp av dette kan personer så skaffe seg de sertifikater de trenger i forbindelse med andre roller, f. eks. i jobbsammenheng, og gjenbruke nøklene.

Slik bruk av offentlige nøkler har fordeler og ulemper. Fordelen er knyttet spesielt til bruk av smartkort og færre private nøkler. Ulempene er at det kan bli en omfattende jobb å tilbakekalle alle sertifikater dersom en enkelt nøkkel er kompromittert, og at det kan tenkes betenkelige situasjoner med hensyn på personvern ved at en kan søke etter bestemte nøkler og få fram sammenhenger. Dessuten må det være samsvar mellom sertifikatpolicies for sertifikatene, spesielt når det gjelder restriksjoner på bruksområder og krav til beskyttelse av den (felles) private nøkkelen.

Et nøkkelpar som skal brukes for kvalifisert signatur kan heller ikke brukes for signaturgenerering med utstyr og programvare som ikke oppfyller de temmelig sterke kravene som vil bli stilt til lovforslaget kaller et «godkjent signaturgenereringsprodukt». I så fall vil angriperen klare seg med å knekke det svakeste signaturgenereringsutstyret for å produsere en falsk signatur, f.eks. med å installere en trojansk kryptomodul som signerer over noe annet enn det brukeren tror blir signert. Dermed kan det faktisk være behov for to uavhengige sertifikater og nøkkelpar for digital signatur, ett som er reservert for kvalifisert signatur og ett som kan benyttes friere, dersom kravene ikke er like sterke. Dette er det imidlertid knapt noen produkter eller profiler som støtter.

4.6 Katalog, sertifikatdistribusjon

SAer vil normalt legge alle sertifikater og tilbakekallingslister inn i en søkbar katalog. Aksess til kataloger er som regel ved hjelp av LDAP, og kataloger er ofte basert på X.500 standarden. Katalogene kan være dedikerte for sertifikater, eller de kan være generelle kataloger som også lagrer sertifikater. SA vil normalt drive en katalog selv, men kataloger kan også drives av andre.

Det er mulig å laste ned sertifikater til andre kataloger, f. eks. lokalt i en virksomhet eller lokalt for en bruker (ala PGPs nøkkelring). Det anbefales å publisere sine sertifikater på web-sider o.l.

Katalogen brukes i hovedsak i to tilfeller:

- Hente sertifikat med korrekt offentlig nøkkel når en skal kryptere en melding til en mottaker;
- Hente tilbakekallingsliste for å sjekke om et mottatt sertifikat er gyldig

Sertifikater som trengs under en (f. eks. SSL) autentiseringsprotokoll, vil vanligvis sendes med som en del av protokollen. Tilsvarende vil sertifikater som er nødvendige for å godkjenne en signert melding, vanligvis være vedlagt meldingen. Dersom de ikke er det må de hentes fra katalogen. (I PKCS#7/CMS er det valgfritt å legge ved sertifikater, men det gjøres vanligvis. Men det kan hende at en bare legger ved brukerens sertifikat, og at sertifikat for brukerens SA, osv. i hierarkier, må hentes fra katalog.)

Sertifikatkataloger har en begrenset verdi foreløpig. Kataloger for forskjellige SAer er ikke koplet sammen, slik at en må en vite hvilken SA som har utstedt sertifikat for en mottaker, før en kan søke etter det. Forvaltningsnettsamarbeidet har krevd at SAer må kople sine kataloger, men dette er så vidt vi vet det eneste eksempelet på slik kopling. Et alternativ er å opprette en felles katalog for alle SAer i stedet for flere separate. Kopling av kataloger krever i praksis bruk av X.500-systemer. Et alternativ er LDAP redirect, men vi er usikre på hvor stor støtte dette i dag har i produkter.

Automatisering av søk i katalogene er vanskelig med mindre programmene som søker, har nok informasjon til å kunne gå rett inn og hente riktig sertifikat. Dersom et søk returnerer flere treff, er det nødvendig med et menneske for å gjøre riktige valg. Dette er spesielt et problem når en skal kryptere meldinger til en bestemt mottaker. I andre tilfeller, dvs. når en trenger et sertifikat for å godkjenne en signert melding, inkludert SA-sertifikat for å validere et annet sertifikat, vil en normalt ha en unik referanse til sertifikatet: navn på SA og serienummer. (Det forekommer imidlertid også at en signatur eller autentiseringsprotokoll identifiserer den offentlige nøkkelen, «subject public key identifier», i stedet for ett bestemt sertifikat. I så fall er det opp til mottakeren å finne et sertifikat som både inneholder denne nøkkelen og aksepteres av mottakeren.)

Det er lettere å søke etter tilbakekallingslister, fordi en kjenner sertifikatet en skal sjekke gyldigheten for, inkludert utsteder. Sertifikater kan også inneholde en direkte peker (f. eks. URL med LDAP-referanse) til tilbakekallingsliste (CRL Distribution Point Extension). Det er foreløpig ikke definert hvordan sertifikater eventuelt skal kunne holde en peker til en OCSP-tjeneste for SAen.

I tillegg til katalogen bør en SA arkivere alle sertifikater og tilbakekallingslister som utstedes, i «ubegrenset tid». I arkivet vil en finne sertifikater som er utløpt eller tilbakekalt, selv om disse normalt er fjernet fra katalogen. Sertifikatarkivet trenger ikke å være integrert med katalogen eller på noen annen måte direkte tilgjengelig for brukere, men informasjon må kunne framskaffes på forespørsel, f. eks. for å kunne godkjenne en gammel digital signatur.

SAer bør også arkivere opplysninger som samles inn i prosessen med å utstede sertifikater, og loggføre alt som skjer i forbindelse med forespørsler om tilbakekalling.

4.7 Navngivning

4.7.1 Navn for SA / utsteder

Det er tre modeller for drift av SAer:

- Intern drift, der virksomheten driver sin egen SA,
- Outsourcing, der en ekstern bedrift med spesialkompetanse driver SA på vegne av virksomheten;
- Tjeneste, der en kjøper sertifikater som er utstedt av en SA-tjenesteleverandør.

De to første modellene er logisk sett like, og brukes primært for intranett og ekstranett. Her er navngivningen slik at utsteder er virksomheten selv. Dette kan generelt føre til et stort antall SAer, men er absolutt brukbart der mengde aktuelle virksomheter er mer begrenset. Dette gjelder alt i dag innen UoH-sektoren, der UNISA-tjenesten fungerer slik at hver virksomhet (universitet, høgskole eller andre) har sin egen logiske SA. Med unntak av én drives alle UNISA-SAer av Norsk Regnesentral (outsourcing). En slik løsning krever at SAene organiseres hierarkisk for at de skal kunne virke ut over eget domene, slik bl.a.

UNISA-tjenesten gjør det. Vi ser også at BankID vil bli organisert etter denne modellen, antagelig med BBS som operatør av en logisk SA for hver bank.

Der virksomheten står som utsteder, representerer sertifikatet en form for binding mellom utsteder og bruker. Policy vil da vanligvis si at SA bare kan utstede sertifikater til egne ansatte / studenter, eller f. eks. bare til egne kunder hvis utstederen er en bank. Privatpersoner som ikke ønsker en slik binding, er nødt til å få sine sertifikater fra en «nøytral» leverandør, etter den tredje modellen.

Denne modellen kan også brukes av virksomheter som ikke ønsker å etablere en SA i eget navn, men heller velge sertifikater fra en velkjent utsteder. Sjansene for at disse sertifikatene kan bli gjenkjent og anerkjent er da potensielt større. Eventuell binding mellom bruker og virksomhet må da gjøres gjennom navnereglene for brukerne. Denne modellen ser ut til å få et visst gjennomslag. Den gjør det enklere å få på plass (nasjonale) krav til sertifikatutstedere. Det vil være et begrenset antall utstedere, og en kan sjekke at en aktuell utsteder er godkjent for å utstede en bestemt type sertifikat (f. eks. kvalifisert sertifikat) under den oppgitte policyen. Både Postens Elektroniske ID og Forvaltningsnettsamarbeidet har valgt denne modellen.

4.7.2 Navnetyper

Et X.509v3 sertifikat inneholder et «hovednavn» (Subject Name) som er et DN (Distinguished Name) etter X.500 standarden [X.521], og eventuelt ett eller flere alternative navn (i AltNames Extension). Det er lov, men ikke anbefalt, å ha sertifikater uten hovednavn, forutsatt at det finnes minst ett alternativt navn. Det mest vanlige eksemplet på alternativt navn er epostadresse. Det finnes en masse forskjellige attributter som kan inngå i et DN. En del av definisjon av sertifikatprofil er å spesifisere hvilke attributter som skal brukes, og på hvilken måte. Ett eksempel på et enkelt DN er:

CN=Jon Ølnes, O=Norsk Regnesentral, OU=Avdeling GEM, C=no

Det opprinnelige formålet til et DN er å angi lokasjon i et globalt katalogtre. For dette formålet må et DN være globalt unikt. Men merk at DN som brukes i sertifikater, ikke trenger å være likt personens DN for oppslag i en X.500 eller LDAP-katalog. (Bortsett fra hvis en forholder seg til Entrust-spesifikasjonene, som faktisk har dette kravet. Dette er grunnet i at en skal være sikker på at en får returnert korrekt sertifikat ut fra at en har søkt seg fram til korrekt DN for det en er ute etter.) DN i sertifikater trenger derfor heller ikke nødvendigvis å være globalt unike.

DN er en navneform som ikke brukes ut over kataloger og sertifikater. Dette skaper problemer når det gjelder å etablere samsvar / ekvivalens mellom forskjellige navn for en aktør. Typisk vil en ikke kjenne DN for en person, men derimot fornavn og etternavn, epostadresse osv. En kan ikke være garantert at det er noen opplagt sammenheng mellom f. eks. DN i det sertifikatet en får returnert fra en katalog, og den epost-adressen som en søkte på, og i hvert fall ikke at det er noe samsvar som kan oppdages enkelt av programvare uten menneskelig assistanse.

Dette kan åpne for angrep av typen mannen i midten, der en angriper kan returnere sitt eget sertifikat i stedet for mottakerens. Avsender kan da bli lurt til å kryptere meldingen ved hjelp av den offentlige nøkkelen til angriperen, som kan dekryptere, lese innholdet, og så kryptere på nytt og videresende til tiltenkt mottaker. Ett forsvar mot dette er å ta med alternative navn i sertifikatet, for dermed å binde den offentlige nøkkelen ikke bare til DN, men også til de alternative navnene.

Denne situasjonen med oversettelse mellom navneformer vil en alltid ha, med mindre det dukker opp en form for navn som kan bli brukt universelt. Men en kan diskutere om ikke epost-adresse ville være et vel så godt valg som hovednavn og garantert unikt navn i sertifikater, med andre navneformer som alternative navn.

Et annet spørsmål når det gjelder valg av navn er hva eller hvem mottakeren er? En menneskelig mottaker trenger et sertifikatformat, i hvert fall gjelder det navnedelen av sertifikatet, som egner seg for en lesbar framvisning på en skjerm. Men dersom mottaker er en applikasjon / tjeneste, må sertifikatet og navnet være egnet til automatisk behandling i programvare. En applikasjon vil være tjent med å finne en eller annen form for identifikator, kodet på et velkjent sted, som kan brukes som «navn». Fødselsnummer er det opplagte eksempelet for FEIDE, siden administrative systemer bruker dette som nøkkel, men siden fødselsnummer regnes som sensitivt, kan en ikke bruke dette direkte (se 4.7.4).

Dersom DN kobles med oppslag i LDAP, kan brukernavn eller andre identifikatorer som applikasjonen trenger hentes fra katalogen. Dette ser etter hvert ut til å være ganske gjennomført for tjener-plattformer med sertifikat-basert autentisering (f.eks. SSL), hvis ikke rett og slett DN i seg selv (i tekstlig form) benyttes som brukernavn.

Nasjonale tegn i DN kan være et problem. Alle som skal bruke et sertifikat må nødvendigvis håndtere den tegnkodingen som ble valgt da sertifikatet ble utstedt, endres kodingen kan en ikke lenger verifisere signaturen på sertifikatet. Den generelle og «korrekte» måten å representere nasjonale tegn på (bruke tegnkodingene UNICODE/UCS-2 evt. UTF-8) støttes ikke i mange av de aktuelle produktene. I praksis brukes gjerne ISO 8859/1 på en måte som er en tilsnikelse i forhold til standardene. Vi anbefaler at man unngår bruk av nasjonale tegn i navn eller deler av navn der det er spesielt viktig å kunne gjøre sammenligninger på en entydig måte, f.eks. navn på sertifikatautoriteter. (Hva med samiske navn? ISO 8859/4 og ISO 8859/10 er uaktuelle i DN, men UNICODE eller UTF-8 vil håndtere dem. Det er rimelig å bruke samme konvensjoner som i Folkeregisteret/DSP -- hva nå det er.)

4.7.3 Navneautoritet

For X.500 DN og andre navnerom der en må være sikret unike navn må det finnes navneautoriteter som er ansvarlige for tildeling av navn for hver sine deler av navnetreet. Dette er definert i en del tilfeller, som f. eks NORID for tildeling av domenenavn under .no på Internett. En navneautoritet vil operere under et sett navneregler, som til dels kan være pålagt av andre autoriteter.

Behovet for å definere navneautoriteter for DN i sertifikater vil variere. Dersom en skal ha globalt unike DN for enkeltpersoner basert på personenes virkelige navn, kan en gjerne bruke Folkeregisteret som navneautoritet for å definere hvordan dette skal gjøres, inkludert bruk av fødselsnummer eller annen unik identifikator (se neste avsnitt). Tilsvarende kan Enhetsregisteret (Brønnøysundregisterne) fungere som navneautoritet for virksomheter. Som minimum må DN være unike innen domenet av en SA.

Hvis en virksomhet har fått tildelt et unikt navn, kan virksomheten selv i sin tur være navneautoritet for egne ansatte, og sørge for at alle ansatte får unike DN med virksomhetens DN som ett element.

Dersom en ikke krever at DN i sertifikater skal være unike annet enn innen hver SAs domene, kan SAene selv være navneautoriteter, eventuelt bare ved å godkjenne navn foreslått av brukerne. Både Norsk Regnesentral og Norges Rederiforbund kan be om å få utstedt sertifikat på «O=NR, C=NO». Dersom en krever globalt unike navn, kan en skille ved å ta med organisasjonsnummeret fra Enhetsregisteret i tillegg. Dersom dette ikke er et krav, kan begge få utstedt sertifikater med samme navn, men ikke fra samme SA. Dette vil typisk være en «først til mølla» situasjon, slik det er for domenenavn på Internett (Norsk Regnesentral har domenet nr.no, mens nr.com er noe helt annet). Det er ikke gitt at dette er noen fornuftig løsning.

Vi ser at en del SAer sørger for egen navneautoritet over brukernavnene og globalt unike navn ved å ta med et navneelement SAen selv kontrollerer, i alle brukernavn, f. eks. sette «O=navn på SA» for alle brukere. Dette er litt uheldig, siden brukernavnene får et element som ikke egentlig er et attributt for brukerne. SA-navn bør kun ligge i utstedernavnet.

4.7.4 Navn for brukere

Et sertifikat binder en offentlig nøkkel til et navn. Første relevante spørsmål er om dette navnet trenger å være unikt? Det finnes mange Per Hansen i Norge. I mange sammenhenger er vi vant til at en likevel bare bruker navn. Vi må ha ytterligere informasjon for å vite hvilken Per Hansen det er som har undertegnet et brev. Andre ganger er det nødvendig med unik identifikasjon, og da brukes en unik identifikator, typisk fødselsnummer (eller kundenummer osv.). En gitt SA må internt ha unike navn for sine brukere for å kunne skille dem fra hverandre. Kombinasjonen brukernavn og utstedernavn må derfor alltid være unik. Behovet for globalt unike brukernavn er ikke opplagt, men trenden er at en prøver å få til dette, siden det er et krav i enkelte tilfeller. Unntak kan være sertifikater som bare skal brukes til lukkede løsninger som intranett og ekstranett.

Konstruksjon av et unikt navn er avhengig av hvilken informasjon som skal være med i navnet. Dersom brukernavnet også skal ha med f. eks navn på arbeidsgiver, kan dette være ganske enkelt. Unike navn for virksomheter kan en få til f. eks. ved bruk av organisasjonsnummer. Sjansen for at flere personer innen samme virksomhet skal ha samme navn er begrenset, og der dette forekommer kan virksomheten selv definere lokale regler for å skille mellom personene. Men for sertifikater for privatpersoner må en bruke nasjonalt unike identifikatorer for å sikre globalt unike navn.

De fleste land har definert en eller annen form for unik identifikator for sine borgere. I Norge har vi fødselsnummer. Det er derfor nærliggende å inkludere fødselsnummer i sertifikater for å sikre unike navn, men dette er problematisk av personvernhensyn. Bruk av unik identifikator er ikke avklart i Norge. Finland og Danmark har valgt å definere en ny type unik identifikator spesifikt for sertifikater. Via tabelloppslag kan de som er autorisert til det, oversette den unike identifikatoren til fødselsnummer.

De fleste land har også en unik identifikator for registrerte virksomheter. I Norge er dette organisasjonsnummer fra Enhetsregisteret. Til forskjell fra fødselsnummer er organisasjonsnummeret ikke på noen måte sensitiv informasjon, og det er derfor ukontroversielt å bruke dette for å sikre globalt unike navn for virksomheter i sertifikater.

Hvilke opplysninger som ellers skal inngå i brukernavnet i sertifikatene, er avhengig av formålet -- hvilken rolle og tilhørende navn en ønsker å autentisere gjennom sertifikatene. Også i dag kan en person ha en rekke legitimasjoner for ulike forhold, og det finnes regler for hvilke legitimasjoner som godtas i forskjellige sammenhenger (jeg kan ikke gå i banken med id-kortet mitt fra jobben). Eksempler på elektronisk legitimasjon kan være:

- Elektronisk bankkort / kredittkort,
- Elektronisk identitetskort fra jobben,
- «Borgerkort» for kommunikasjon mot det offentlige,
- Elektronisk pass,
- Eget epost-sertifikat med epost-adresse som viktigste navn, evt. i stedet for DN.

Det er langt fra trivielt å finne rolle / type sertifikat ut fra navn eller andre opplysninger i sertifikatet selv. I noen tilfeller kan utstedernavnet gi en indikasjon, f. eks. dersom dette er arbeidsgiver eller en bank, Men generelt er en henvist til å trekke slutninger fra bare brukernavnet, som kan inneholde elementer som organisasjonsnavn, organisasjonsenhet, tittel osv. BankID er ett eksempel på et «varemerke» for sertifikater, i form av et navneelement som alltid vil være med, og som kan gjenkjennes.

Merk at et annet aspekt er om navnet skal være egnet til å behandles automatisk av programvare hos mottakeren, eller om det primært skal være egnet for å presenteres for menneskelige brukere.

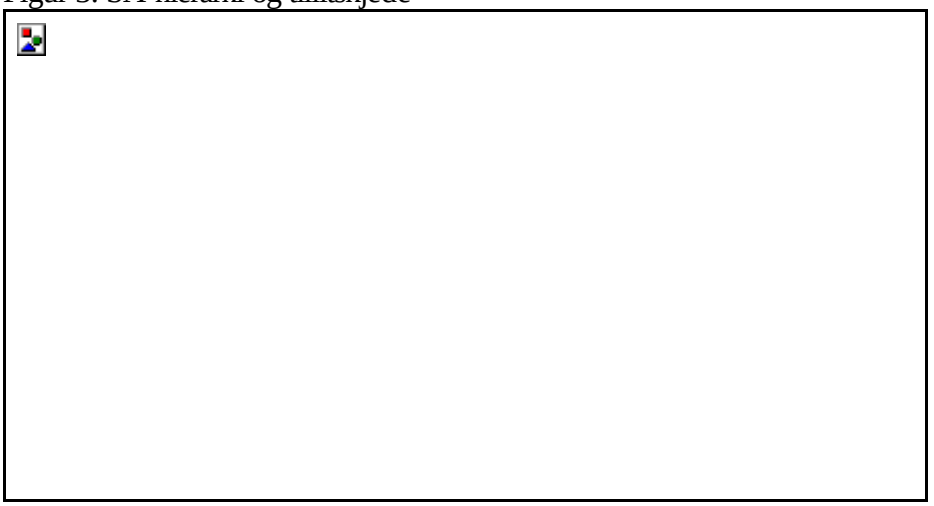
4.8 Tillitsmodeller: strukturering av sertifikattjenester

Godkjenning av SAens signatur på et sertifikat gjøres med SAs offentlige nøkkel. Det betyr at en trenger å knytte denne nøkkelen sikkert til SAens identitet, på samme måte som brukernes offentlige nøkler må knyttes til deres identitet gjennom sertifikater. Dersom den SAen som har utstedt sertifikatet, er den samme som en selv bruker, skal dette være enkelt. SAens offentlige nøkkel ligger da vanligvis på smartkortet, eller er tilgjengelig på en annen betryggende måte. En SA vil vanligvis legge sin offentlige nøkkel i et egensignert sertifikat. SA er både «bruker» og utsteder for sertifikatet. Dette gir ikke noen egentlig autentisering, siden det er lett å forfalske et slikt sertifikat, men det gir integritetsbeskyttelse for nøkkelen når sertifikatet hentes fra et tiltrodd sted.

Men det finnes jo flere SAer, og hva gjør en dersom utsteder for motpartens sertifikat er en annen SA? Det finnes tre modeller for samtrafikk mellom SAer:

- Monolittisk, dvs. helt separate tjenester: Alle sertifikater må godkjennes i forhold til utsteder, og den som skal godkjenne sertifikatet, må selv skaffe korrekt offentlig SA-nøkkel og avgjøre tiltro til sertifikatet. Dette er dagens situasjon på Internett, der offentlige nøkler for de SAene som har betalt nok for det, ligger i standardutgavene av de vanligste web-leserne. Brukere kan selv legge inn flere SA-nøkler i web-leserne hvis de ønsker det.
- Hierarki, der en SAs offentlige nøkkel har et (vanlig X.509) sertifikat utstedt av en høyere nivå SA, osv. tilbake til en rot-SA (se figur 3). Rot-SAs offentlige nøkkel må gjøres kjent på en betryggende måte. En bruker vil kunne godkjenne alle sertifikater utstedt under hierarkiet.
- Kryssertifisering, der SAer parvis utsteder sertifikater for hverandres offentlige nøkler. En bruker hos en SA vil kunne godkjenne et sertifikat utstedt av den andre SAen gjennom å bruke den offentlige nøkkelen fra kryssertifikatet fra sin tiltrodde SA. Kryssertifisering vil vanligvis være gjensidig, men kan også være enveis. Det kan være lovlig, og til og med anbefalt, å kryssertifisere, «lage snarveier», mellom SAer i samme hierarki. (Egentlig er kryssertifisering samme tillitsmodell som PGPs «web of trust», med den forskjellen at det ikke er vanlige brukere som utsteder sertifikater for hverandre, men SAer.)

Figur 3: SA-hierarki og tillitskjede



En monolittisk struktur har sine klare begrensninger når det gjelder skalering. Det vil etterhvert bli svært mange SA-tjenester i et internasjonalt perspektiv, og det blir umulig for en bruker å forholde seg til alle disse, spesielt når det gjelder vurdering av nivået på tjenesten. En viss strukturering vil tvinge seg fram.

Hierarki eller kryssertifisering gir ikke nødvendigvis noen hjelp når det gjelder nivået på tjenesten. Strengt tatt innebærer en slik strukturering en gjenkjennelse av andre SAer i systemet, ikke en godkjennelse. SAer innen samme hierarki, eller kryssertifiserte SAer, kan drives etter forskjellige policyer, og med forskjellig

kvalitet. Men hierarki eller kryssertifisering innebærer i hvert fall en forsikring: At SAer faktisk drives i henhold til sin policy og praksis. En SA pålagt å sjekke dette hos motparten før den utsteder et kryssertifikat eller et sertifikat for en underordnet SA.

Dersom en skal ha en forsikring om nivået på SAene i systemet i forhold til sin egen SA, må en pålegge ytterligere regler, som at alle SAer i et hierarki må ha (nivåmessig) samme policy. For kryssertifisering kalles dette for «policy mapping», og innebærer at en anerkjenner den kryssertifiserte SAen som å være på samme nivå som en selv. I Forvaltningsnettsamarbeidet endte en opp med en felles policy, og kryssertifisering mellom SAene med en enkel policy mapping basert på denne. Policy mapping kan ellers være en meget vanskelig øvelse. Policy mapping angis ved et flagg i kryssertifikatene.

Strukturering av SAer er basert på at tillit regnes som en transitiv egenskap. Når en SA jeg stoler på, går god for en annen, så stoler jeg også på den andre SAen. Hvis vi bruker en hierarkisk struktur som eksempel, har vi at den enkelte bruker har et forhold til, og stoler på, den SAen som har sertifisert henne, og må kjenne til og stole på alle SAer opp til rot-SA når det gjelder sine egne sertifikater. I praktisk bruk må en også være forberedt på å stole på alle andre SAer i systemet. Det etableres en tillitskjede mellom to brukere, gjennom alle mellomliggende SAer (se figur 3). Denne kjeden trenger ikke å gå helt til roten, bare til en SA som har begge brukerne i sitt subtre.

Den første modellen for et sertifikatsystem for Internett (fra PEM spesifikasjonene [RFC 1424]) gikk ut på å samle alt under en felles rot-SA, kalt IPRA (Internet Policy Registration Authority). Under IPRA skulle det ligge et sett av PCAer (Policy Certification Authority) med forskjellige policyer. En SA skulle da velge den PCAen som passet med sitt nivå, og få utstedt et sertifikat fra denne. Denne strukturen la opp til et dypt hierarki under dette, med flere lag med underordnede SAer med tilsvarende policyer.

Denne strukturen fikk ikke gjennomslag. Ett av problemene er at tillitskjedene i systemet kan bli lange, og dette svekker tilliten siden det aldri er 0 sannsynlighet for svikt i ett gitt ledd i kjeden. Et annet problem er at stor dybde gir tunge beregninger, siden det kan være en lang kjede av sertifikater som skal godkjennes.

Trenden nå ser ut til å være i retning av «grunne» hierarkier. Disse vil bestå av en rot-SA, som typisk vil ha rollen som en PCA og bestemme (nivået for) policy i hierarkiet, og ett nivå av SAer under dette.

Ett eksempel ser vi i Tyskland, der alle SAer som ønsker å utstede sertifikater i henhold til den tyske digital signatur-loven, må få et sertifikat fra en offentlig rot-SA. Ett annet eksempel er arbeid innen Den Internasjonale Postunionen, der en ønsker å opprette en rot-SA med en underordnet SA i hvert medlemsland (ca. 180 land) drevet av postverkene. Dette arbeidet er basert på erfaringer fra et nordisk prosjekt (Norge, Sverige og Finland, og også Irland). Det legges opp til utstrakt kryssertifisering mellom post-SAer i forskjellige land.

Hierarkiene kan være separate, eller de kan koples sammen ved at rot-SAene kryssertifiserer hverandre, eller ved at det opprettes ett nivå til. F. eks. kan det tyske systemet koples sammen med tilsvarende systemer i andre land ved at det opprettes en europeisk (eller internasjonal) rot over de nasjonale.

4.9 Akkreditering av sertifikattjenester, juridisk ansvar

Skal hvem som helst ha rett til å drive sertifikattjenester, eller skal slik virksomhet underlegges regulering, dvs. at kun akkrediterte (tildelt lisens) operatører kan drive tjenester? Det kan skilles mellom to typer akkreditering:

- Retten til å drive sertifikattjenester i eget navn (retten til å utstede legitimasjon),
- Retten til fysisk å drive sertifikattjenester, for egen del eller på vegne av andre.

I Norge er dette utredet av Torvund-utvalget [NHD 2000], i tråd med retningslinjene i EU-direktivet for elektroniske signaturer. Utvalget anbefaler ikke regulering av SA-tjenester generelt, men anbefaler å

opprette en registreringsordning for tjenester som utsteder kvalifiserte sertifikater, med Post- og teletilsynet som offentlig tilsynsmyndighet. Siden FEIDE ikke skal forholde seg til kvalifiserte sertifikater, har ikke dette noen betydning for FEIDE.

Det pekes likevel på ordninger som skal sikre at en SA skal kunne opparbeide tillit til sine sertifikater i markedet. Det er rimelig klart at det ikke vil være lett for små firmaer å bevege seg inn på dette markedet. Det understrekes ved at det er store, tunge aktører som er i gang med slike tjenester i Norge i dag. Flere av disse spiller på at de har lang tradisjon for drift av tiltrodde tjenester, f. eks. banksystemer og systemer for offentlige registre.

Felles for alle disse aktørene er også at de ser sertifikatutstedelse som en del av et totalt tjenestetilbud. Dersom prisen på smartkort og sertifikater skulle dekke opp kostnadene ved utstedelse, ville antagelig prisen (også i framtida, med høye volumer) bli altfor høy. Men dersom en ser på forenklinger og samordning av aksess til tjenester, og økt sikkerhet, tilgjengelighet og funksjonalitet (f. eks. signaturer) for totalen av de tjenestene en aktør tilbyr, kan dette fort forsvare store investeringer i SA-funksjonen.

En annen årsak til at SAene trenger tyngde er de juridiske forpliktelser en SA potensielt påtar seg ved å utstede et sertifikat. Ved avanserte tjenester, som kvalifiserte sertifikater, som skal kunne brukes til transaksjoner av høy verdi, kan erstatningsbeløpet for feil, og spesielt uaktsomhet, bli meget stort. En slik SA stiller også meget strenge krav til sikker drift, fysisk sikre lokaler, kvalifisert og klarert personale, døgnkontinuerlig operasjon osv. De nevnte aktørene og noen til i Norge har dette på plass allerede, mens det vil være meget dyrt for en nykommer å få alt på plass.

Selv om det offentlige ikke kommer til å gå inn og gjennomregulere dette markedet, er det klart at det offentlige selv vil formulere krav til hva de selv kan godta, f. eks. for å akseptere en digitalt signert søknad. Det er nedsatt et utvalg under Arbeids- og administrasjonsdepartementet som skal utrede dette fram til slutten av 2000. I praksis vil dette arbeidet ha en sterk regulerende effekt, siden SAene vil være svært opptatt av å oppfylle disse kravene i markedet. Merk at dette utvalget skal se på mer enn bare kvalifiserte sertifikater.

Det ligger ingen direkte initiativer eller reguleringer her når det gjelder internasjonal samordning, f. eks. hvordan et norsk sertifikat skal kunne godkjennes / brukes i utlandet. Men norske myndigheter (og i hvert fall en del av de norske SAene) har et utstrakt samarbeid med aktører i andre land i Europa, spesielt i Norden.

Det er ingenting i norske lover og regelverk som kan hindre en norsk borger (eller norsk bedrift) i å benytte tjenester fra en SA i utlandet, eller hindre en norsk SA å tilby sine tjenester internasjonalt. Men i hvert fall for kvalifiserte sertifikater må de juridiske forholdene rundt ansvar mm. være tilfredsstillende avklart før norske myndigheter kan akseptere noe annet enn en norsk SA for kommunikasjon mot offentlig sektor. I praksis vil det si at SAen skal forholde seg til norsk lovgivning, selv om SA-installasjonen fysisk ligger i utlandet. I andre sammenhenger bør en utenlandsk SA kunne fungere like godt som en norsk.

Fysisk drift av TTP-tjenester kan knyttes til et «stempel» på at en organisasjons tjenester drives tilfredsstillende, dvs. en sikkerhetsmessig evaluering. Torvund-utvalget legger opp til at dette skal være markedsdrevet, slik at det eventuelt vil bli et krav fra kundene.

Det arbeides nå med å få på plass en internasjonal standard, ISO/IEC IS17799, som kan danne grunnlag for en sertifiseringsordning av en organisasjons sikkerhetsmessige kompetanse og sikkerhetsadministrasjon. Et sertifikat utstedt på grunnlag av en slik standard kan igjen danne gunnlaget for en akkreditering av organisasjonen, som i sin tur kan bidra til å gi organisasjonen den nødvendige legitimitet og tillit i markedet til å kunne drive en sertifiseringsordning for offentlige nøkler. Et norsk akkrediteringsorgan som kan gjennomføre prosessen forventes å være operativt innen rimelig tid.

5. Roller og prosedyrer for utstedelse av sertifikater

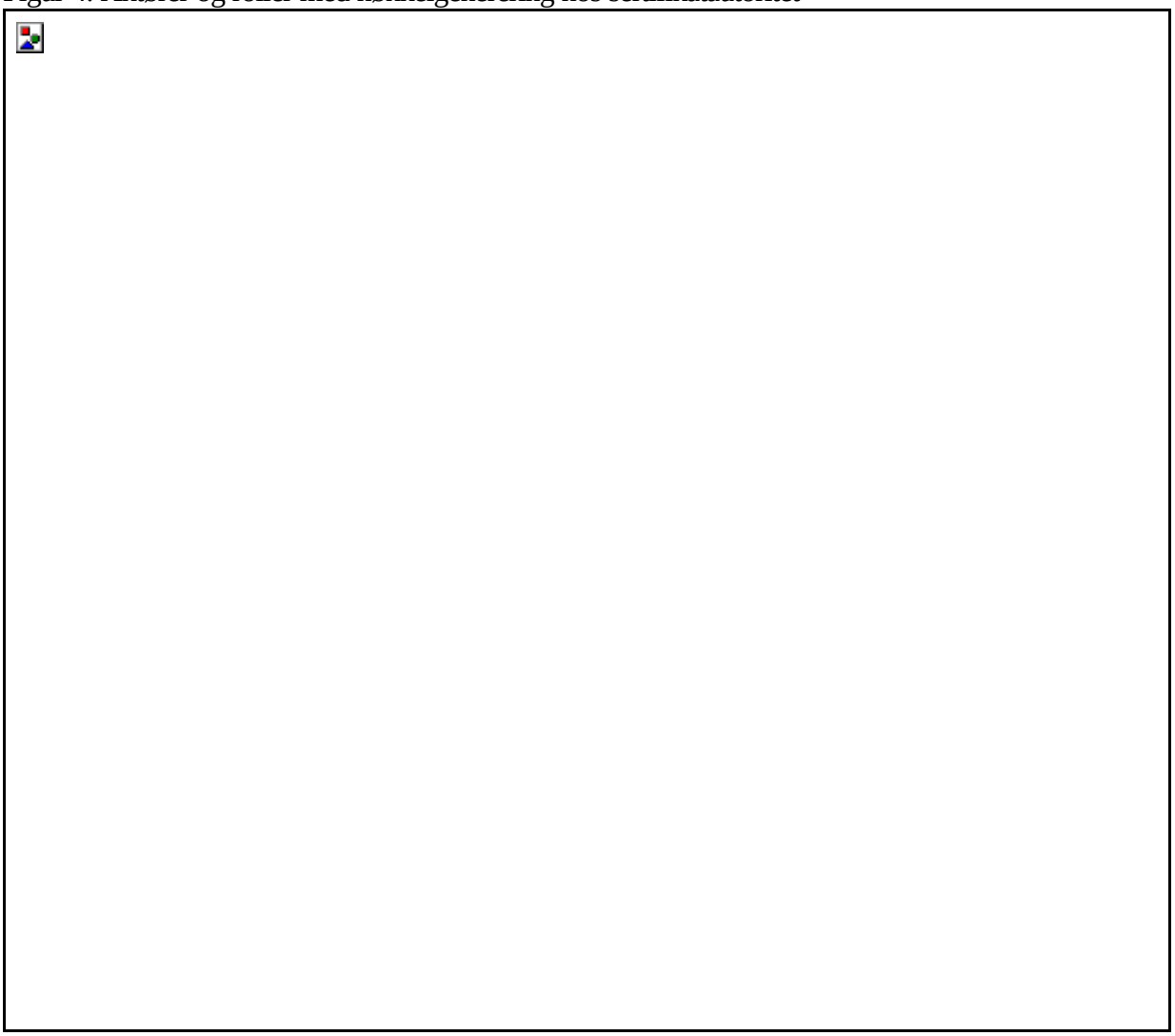
5.1 Aktører og roller

Det er en rekke roller knyttet til utstedelse og administrasjon av sertifikater. Disse rollene og prosedyrene rundt dem er definert i sertifikatpolicy, og kan til en viss grad fylles av ulike aktører. Hver rolle trenger utstyr og programvare for å utføre sin funksjon. Rollene er:

- Sertifikatautoritet (SA) -- dette er sertifikattjenesten, som behandler forespørsler og utsteder sertifikater,
- Registreringsautoritet (RA) står for kontakten med brukerne, mottar og godkjenner forespørsler om sertifikater, og sender disse videre til SA,
- Bruker som må henvende seg til sin lokale RA (personlig frammøte dersom det skal være et noenlunde høyt sikkerhetsnivå) for å få utstedt sertifikater og smartkort,
- Navneautoritet (trenger ikke å være direkte involvert i sertifikatutstedelsen) fastsetter regler for navngivning og hvem som skal kunne be om sertifikater for gitte navn -- dette er diskutert tidligere,
- Smartkortleverandør / -produsent leverer kortene, blanke eller med noe informasjon lagt inn,
- Nøkkelgenerator produserer par av offentlig / privat nøkkel av godkjent kvalitet,
- Nøkkeldeponi for sikkerhetskopiering av private nøkler kan være ønskelig for å sikre at informasjon ikke går tapt dersom dekrypteringsnøkkelen går tapt,
- Katalog for lagring av sertifikater og tilbakekallingslister er nødvendig for bruk av systemene.

Figur 4 viser roller og funksjoner i prosessen for utstedelse av sertifikater og smartkort i et scenario der SA har ansvaret for å produsere nøkkelpar for brukerne og for å ferdigstille smartkortene. Dette vil være den normale situasjonen ved bruk av smartkort. Noen av rollene beskrives nærmere nedenfor.

Figur 4: Aktører og roller med nøkkelgenerering hos sertifikatautoritet



5.2 Registreringsautoritet (RA)

En RA må være fysisk nær brukerne for å kunne godkjenne forespørsler når det kreves fysisk frammøte. For virksomheter vil typisk denne rollen fylles av angitte, autoriserte, interne personer f. eks. fra personalavdeling eller IT-drift. For tjenester mot privatpersoner må SA operere med et kontaktnettverk, som godt kan være postkontor, bankfilialer eller annet.

En gitt RA vil normalt ha et avgrenset ansvarsområde, f. eks. en eller flere spesifikke virksomheter. RA vil bare ha lov til å formidle forespørsler for brukere som naturlig hører under ansvarsområdet, og da kun til en spesifikk SA (normalt, men det kan tenkes RAer med avtale med flere SAer). SA må verifisere at RA holder seg innen sitt ansvarsområde. I tillegg til sikkerhetsmessige årsaker, er dette nødvendig for å garantere unike navn i sertifikater, dvs. at to ulike personer ikke kan få sertifikater med identiske navn -- i hvert fall ikke fra samme SA.

En RA kan operere en «enkel» tjeneste, som bare sender enkle forespørsler til SA. SA tar seg da av all personalisering av smartkort mm., og smartkort og annen informasjon sendes fra SA til brukeren (eventuelt via RA). Alternativt kan en RA være mer «avansert» med tanke på personalisering av smartkort, og til og med utføre nøkkelgenerering og innlegging på kortet.

5.3 Sertifikatautoritet (SA)

SA mottar forespørsler fra brukere via RA, godkjenner forespørslene, og utsteder sertifikater (normalt også smartkort) for brukerne. Sertifikatene leveres til brukeren, i smartkortet eller på annen måte, og legges i katalogen. SAs egen offentlige nøkkel legges normalt i et egetsignert sertifikat, i katalogen, på brukernes smartkort, og distribueres ellers på andre måter om ønskelig.

En SA kan utføre flere av rollene i systemet. En SA vil normalt ha et overordnet ansvar for RA-funksjonen når det gjelder godkjenning av utstyr, programvare, kommunikasjon, rutiner, støttefunksjoner og opplæring. SA kan tilby initialisering, personalisering og distribusjon av smartkort, nøkkelgenerering og nøkkeldeponi. SA vil også normalt ha ansvaret for katalogen.

5.4 Nøkkelgenerering

Produksjon av nøkkelpar for smartkort kan gjøres av kortprodusenten, SA, RA eller internt på smartkortene under kontroll av brukeren selv. Det siste alternativet -- internt på smartkort -- kan ikke velges dersom en bruker nøkkeldeponi (se nedenfor), siden det ikke er mulig å hente private nøkler ut fra et smartkort. For å sikre god kvalitet på nøklene vil disse normalt bli produsert ved hjelp av spesialutstyr, og deretter overført til smartkortet.

5.5 Nøkkeldeponi

Det kan være ønskelig å ha en sikkerhetskopi av private nøkler som brukes til dekryptering (dvs. til utveksling av hemmelige, symmetriske nøkler for kryptering). Et slikt nøkkeldeponi sikrer tilgjengelighet av informasjon som er lagret i kryptert form, f. eks. i tilfelle ødelagt eller mistet smartkort. Merk at private nøkler som brukes til digital signatur eller autentisering, aldri skal være kopiert til et nøkkeldeponi, og dette er en viktig grunn til å bruke forskjellige nøkler til forskjellige formål.

I et arbeidsforhold kan arbeidsgiver kreve bruk av nøkkeldeponi, siden den informasjonen som utveksles, ansees for å være arbeidsgivers eiendom. Dette er et viktig argument for å skille mellom privat bruk av digital signatur og kryptering og bruk i et arbeidsforhold. Adgang til nøkkeldeponiet gis kun til brukeren selv, eller i enkelte tilfeller til andre personer i brukerens virksomhet.

Det er viktig å presisere at denne bruken av nøkkeldeponi er et tiltak for tilgjengelighet av informasjon for brukeren selv og eventuelt arbeidsgiver. Nøkkeldeponering bør definitivt ikke være obligatorisk. Det finnes ingen krav til nøkkeldeponering til fordel for politi og andre autoriserte myndigheter i Norge, og i hvert fall foreløpig ingen initiativer i den retningen. Men en må jo anta at dersom det finnes en deponert kopi av en nøkkel, så vil myndigheter kunne få adgang til den. Med nøkkeldeponi blir de systemene som er tiltrodd å håndtere hemmelige nøkler vesentlig større og mer kompliserte, så en kan i hvert fall være sikker på at risikoen for misbruk av nøklene ikke blir mindre.

5.6 Smartkort

Smartkort utstedes i tre operasjoner:

- *Produksjon*, der «tomme» smartkort leveres fra produsenten -- evt. kan kort leveres med et visst innhold.
- *Initialisering*, der applikasjoner og en del generell informasjon legges inn på kortet -- dette vil typisk gjøres av SA, men kan også gjøres under produksjon eller personalisering,
- *Personalisering*, der informasjon om brukeren legges inn -- dette kan utføres av SA eller RA, eller i enkelte tilfeller av brukeren selv.

Innlegging av nøkler gjøres normalt under initialisering, men kan også gjøres under personalisering eller produksjon.

Personalisering av smartkort gjøres ved at det registreres hvilken bruker som har fått overlevert et spesifikt smartkort, ved at det utstedes sertifikater med brukerens navn for de offentlige nøklene som tilsvarer de private nøklene i smartkortet, og ved at all nødvendig personlig informasjon legges inn i smartkort og katalog. Smartkort og tilhørende PIN-kode overleveres brukeren på en betryggende måte, direkte eller via RA.

Informasjonen som legges inn ved initialisering og personalisering av kortene vil ofte være umulig å endre siden. (Dette avhenger av type kort, kort-operativsystem og utformingen av applikasjonen.) For noen anvendelser er dette en viktig sikkerhetsegenskap, og f.eks. kan det utnyttes til å sikre at bilde og tekst som er preget på kortet stemmer med navn og sertifikater lagret i kortet. Fysisk trykking eller preging av smartkortet skjer i forbindelse med (eller før) personaliseringen.

I dag er det ikke nok lagringskapasitet på smartkortene til å lagre mange sertifikater. Lagring av flere sertifikater på smartkortene kan være et langsiktig alternativ, men foreløpig må en basere seg på kataloger, lagring på disk på brukernes maskiner, og lagring av sertifikatinformasjonen «i kortversjon» på smartkortene. Merk at det først og fremst er de private nøklene som *må* lagres på smartkortet. Det er liten grunn til å ha sertifikatene på smartkortet dersom applikasjonsprogrammet kan få tak i de aktuelle sertifikatene annetsteds fra, og det er vanligvis tilfelle.

6. Bruksområder for digital ID

6.1 Sikker tilgang til nettverkstjenester

For å gi sikker tilgang til forskjellige typer nettverkstjenester, kan digital ID brukes i forbindelse med transportlagssikkerhet. Det mest aktuelle er å bruke SSL med klientautentisering [Freier, Karlton og Kocher 1996] , [RFC 2246]. Dette gjelder spesielt for web-baserte tjenester, siden alle de vanlige web-klientene støtter HTTP over SSL [RFC 2818] på en brukbar måte. (Oppgradering til TLS etter oppkobling mot en regulær HTTP-tjeneste [RFC 2817] er derimot ikke særlig utbredt.)

Tjenester basert på andre protokoller kan også sikres med SSL:

- For CORBA-baserte tjenester er IIOP over SSL standardisert [COSS, kap. 15.14], og støttes etter hvert av de fleste ORB-produktene. (Men det er uklart hvor god interoperabilitet en får på tvers av produkter.)
- For en del protokoller, bl.a. LDAP [RFC 2830] , SMTP [RFC 2487] , POP3 og IMAP [RFC 2595] , finnes standarder for hvordan disse skal brukes over TLS.
- Protokoller med autentisering basert på SASL [RFC 2222] (f.eks. LDAP) kan bruke autentiseringstype EXTERNAL for å referere til autentisering i transportlaget.

Teknisk sett kan SSL brukes sammen med de fleste TCP-baserte protokoller, men det er begrenset hvilke ferdige løsninger som finnes. I verste fall kan en falle tilbake på generelle SSL-proxyer. (Utfordringen er i tilfelle hvordan autentiseringen kommuniseres sikkert mellom proxy og den virkelige tjenesten. Dette vil være protokoll- og implementasjonsavhengig. På klientsiden vil alltid være litt problematisk å installere programvare utover ordinær web-klient og annen vanlig klient-programvare.)

SSL gir konfidensialitet, men i sammenheng med FEIDE er det klientautentiseringen som er den viktigste tjenestesten. (Konfidensialitet og autentisering på transportnivå er imidlertid temmelig tett forbundet: Alternativet til klientautentisering er passord-basert autentisering, og i så fall er konfidensialitet i transportkanalen vesentlig. Konfidensialitet er på sin side nokså meningsløst uten autentisering av mottakeren.) SSL klientautentisering, i likhet med alle andre autentiseringsmekanismer basert på X.509-sertifikater, betyr i praksis at nettverkstjenesten kjenner klienten gjennom et X.509 distinguished name (DN). Dette blir et nøkkelpunkt i vår arkitektur for sikker tilgang til nettverkstjenester:

- Autentiseringsmekanismen identifiserer brukere gjennom DN (uavhengig av hvilken mekanisme som faktisk brukes)..
- Autorisering av brukere skjer ved å koble rettigheter til DN.
- Aksesskontroll skjer på basis av autentisert DN og håndhever autorisasjonene knyttet til dette.

I den grad andre autentiseringsmekanismer brukes, må de altså avbilde eventuelle lokale eller mekanisme-spesifikke brukeridentiteter til DN.

Når SSL brukes, baseres konfidensialiteten på tjenerens sertifikat og nøkler. Det er altså ikke noen forutsetning at klienten har noen digital ID dersom formålet bare er konfidensialitet.

Et alternativ til transportlagssikkerhet med SSL er nettverkslagsikkerhet, dvs. VPN-løsninger. Slike løsninger finnes i to hovedvarianter:

- Aksess-VPN, for sikker tilknytning av eksterne/mobile klienter. (Protokoller: L2TP + IPsec, PPTP.)
- Intra/ekstranett-VPN, for sikker sammenkobling av nett/organisasjoner. (Protokoller: IPsec.)

FEIDE kan være aktuelt for aksess-VPN, dvs. tilknytning av en enkelt, enbruker klientmaskin. (For intra/ekstranett-VPN er FEIDE neppe relevant. I disse tilfellene er entitetene som autentiseres nett og maskiner, ikke personer.) En del av produktene for aksess-VPN har muligheter for kryptografisk

autentisering, som muligens kan konfigureres til å bruke FEIDEs smartkort. Dette betyr igjen at FEIDE DN vil opptre som bruker-ID, eller må kunne avbildes til bruker-ID gjennom katalog. Fundamentalt innebærer brukerautentiseringen i aksess-VPN bare autentisering overfor aksesstjeneren. (L2TP network server, LNS, i L2TP-terminologi; Microsoft RAS for PPTPs vedkommende.) Hvor langt denne brukerautentiseringen rekker avhenger av produktet. I en del tilfeller, f.eks. Windows 2000 og i noe mindre grad NT, integreres denne autentiseringen i operativsystemets egne innloggings-/brukerautentiseringsmekanismer. (I så fall er det et spørsmål om FEIDEs DN-format passer med operativsystemets, f.eks. de konvensjonene som gjelder for Active Directory for Windows' vedkommende.)

Mest vanlig er det nok at autentiseringen bare brukes internt i aksesstjeneren, til å implementere en aksesskontroll som i prinsippet bare er en filtrering av IP-trafikken fra klienten på IP-adresse og port. De tjenestene som klienten dermed får adgang til, kjenner imidlertid ikke brukerens identitet, annet enn indirekte gjennom IP-adressen. (Aksesstjeneren kan tilby et grensesnitt for å finne brukerens virkelige identitet, men så vidt vi vet finnes ingen åpne løsninger for dette.) For de aller fleste tjenestene som er aktuelle i forbindelse med FEIDE er dette for svakt.

Flere operativsystemer, bl.a. Windows 2000, Linux, Solaris 8, har støtte for smartkortbasert autentisering ved innlogging, og med en utvidbar arkitektur der selve autentiseringsmekanismen håndteres av en pluggbar autentiseringsmodul (PAM). Her bør det være gode muligheter for å bruke FEIDE-kortet for innlogging, enten det skjer lokalt på arbeidsstasjonen eller med SSL, VPN etc.

6.2 E-post-sikkerhet

Meldingssikkerhet for elektronisk post støttes etter hvert brukbart av hyllevare e-post-klientprogramvare, og det finnes også separate løsninger basert på «plug-ins» eller helt frittstående programvare. Den utbredte standarden er S/MIME [RFC 2633]. Den viktigste tjenesten er her antakelig også autentisering av meldinger, dvs. digital signatur. Konfidensialitet er også vesentlig, og kombineres med signering i S/MIME-formatet.

En digital signatur på elektronisk post gir en viss grad av ikke-benekting. Det er neppe aktuelt å integrere løsninger for **kvalifisert** digital signatur i vanlige e-post-systemer på vanlige kontor-, lab- eller hjemme-PC-er. Men det er ikke nødvendig å utføre signaturen på samme maskin som e-post-klienten kjører på. Dersom det er bruk for sterk ikke-benekting eller kvalifisert signatur kan innholdet signeres på en egen, spesielt sterkt sikret, maskin og transporteres til brukerens vanlige e-post-system for å bli sendt som et hvilket som helst annet dokument/vedlegg.

Meldingsautentisering har altså ikke noen spesielle behov for integrasjon mot e-post-systemet: Signeringsnøkkel og -sertifikat som skal brukes er vanligvis gitt. Brukeren har bare en aktuell nøkkel av dem som er installert på maskinen eller finnes i smartkortet som sitter i leseren. Og selv om signeringsprogrammet må be brukeren velge mellom flere aktuelle sertifikater er det sjelden noe behov for samsvar mellom sertifikatet og avsenderadressen. Når meldinger krypteres for konfidensialitet er det derimot praktisk med tettere kobling. Grunnen er at meldingen (egentlig: innholdsnøkkelen) må krypteres til hver enkelt mottaker som en ønsker skal kunne dekryptere og lese meldingen. Skjer krypteringen i kontekst av e-post-systemet er listen av adressater tilgjengelig, og dersom adressatenes sertifikater er tilgjengelig f.eks. i en LDAP-katalog, kan dette automatiseres. Men en slik automatikk er et tveegget sverd: Risikoen for å bli forledet til å ta med en mottaker for mye er avgjort til stede, og det er heller ikke gitt at det er en-til-en-sammenheng mellom e-post-adressater og mottakere av sikkerhetstjenesten. Bl.a. er det helt legitimt å sende en sensitiv, kryptert melding via en tredjepart, sekretær o.l. som selv ikke selv har nøklene som trengs for dekryptering, og i det tilfellet ville det være helt feil å inkludere denne mottakeren som «sikkerhetsmottaker».

For mottak av signert eller kryptert e-post er det heller egentlig ikke noe behov for noen annen integrering enn det som er vanlig i e-post-lesere, at de generelt kan håndtere innhold og vedlegg av forskjellige typer. Formatene for signatur og kryptering (sikkerhetssyntaksen) er helt selvstendige og uavhengige av e-post-systemet, og spesielt identifiserer sikkerhetssyntaksen selv hvilke nøkler/sertifikater som skal brukes hhv. for signaturverifisering og dekryptering.

En digital signatur kan behandles løsrevet fra det innholdet den signerer. Dette er spesielt aktuelt i forbindelse med e-post: Dersom en melding skal sendes en mottaker som en ikke er sikker på om er i stand til å håndtere signaturen, eller flere mottakere hvor bare noen kan håndtere eller er interessert i signaturen, kan signaturen legges i et separat vedlegg, ved siden av det signerte innholdet, slik at signaturen kan ignoreres og innholdet likevel kan leses på vanlig måte. Denne formen for signatur blir nødvendigvis noe sterkere knyttet opp til e-post-systemet, siden den både baserer seg på konvensjoner på det nivået for hvordan innhold kobles til signatur, og dessuten på at innholdet behandles slik at det kan rekonstrueres nøyaktig slik det var, bit for bit, da det ble signert. (Men selve ideen om **klartekst-signatur** er ikke avhengig av denne bestemte måten å realisere det på. De fleste aktuelle formater for signerte data er av type «digital signature with appendix», dvs. at innholdet kan leses uten nødvendigvis å verifisere signaturen.)

6.3 Andre anvendelser av digital signatur

Digital signatur har mange andre anvendelser enn i direkte tilknytning til e-post. Det kan dreie seg om signering av forskjellige typer dokumenter (rapporter, notater, oppgaver), registreringsskjemaer, søknader m.m. I mange av disse tilfellene er ikke-benekting svært vesentlig, og det kan være bruk for såvidt sterk sikkerhet i forbindelse med signeringen (f.eks. kvalifisert signatur) at signeringen må utføres på eget utstyr. I så fall må også brukerne utstyres med flere signeringsnøkler med tilhørende sertifikater, som er utstedt under forskjellige sertifikatpolicies. Det er sertifikatpolicy som bestemmer styrken i signaturen, og sertifikatpolicy vil i tilfelle stille krav til brukeren om at nøkkelen aldri skal brukes noe annet sted enn på en spesielt sikret maskin. (Nøkkelen kan ligge på samme smartkort som andre nøkler, bare den er sikret med en egen PIN-kode eller tilsvarende, men for å gjøre feil bruk mindre sannsynlig kan det nok være gunstig å bruke et eget kort for nøkler med spesielle bruksområder.)

De tekniske løsningene avhenger av i hvilken form innholdet som skal signeres er tilgjengelig. En stor del av tilfellene dekkes sannsynligvis av frittstående signerings- og verifikasjonsprogrammer som arbeider på filnivå. Slike er ikke spesielt kompliserte å sette opp, men løsningene må oppfylle kravene som stilles for ikke-benekting som vi diskuterte ovenfor.

Signering av skjema er et viktig område. Vanlige HTML-skjema i Web egner seg imidlertid ikke for signering, i hvert fall ikke når det er et element av ikke-benekting inne i bildet. Løsninger som skal gi tilfredsstillende ikke-benekting krever at skjema fylles inn, presenteres og signeres lokalt.

6.4 Lokal identifikasjon og autentisering

Smartkortet kan også brukes til rent lokal identifikasjon eller autentisering, f.eks. som nøkkelkort til bygninger, bibliotekskort, for tilgang til lab-maskiner, skrivere, kopimaskiner og tilsvarende. Selve den kryptografiske smartkortapplikasjonen kan være den samme, og et SSL klientautentiseringssertifikat (f.eks.) vil sannsynligvis også være helt greit å bruke for autentisering i en annen sammenheng (både ut fra teknisk kompatibilitet og kravene som sertifikatpolicy stiller). Men det vil også være forhold som tilsier at disse anvendelsene skilles, både med forskjellige nøkler og sertifikater, og med forskjellige smartkortanvendelser. For det første kan det finnes ferdige løsninger med egne grensesnitt mot smartkortet som kan være aktuelle å ta i bruk. Dessuten kan sikkerhetskravene være forskjellige: For SSL klientautentisering, og autentisering over nettet i det hele tatt, bør nok hver enkelt autentisering bekreftes

med PIN-kode e.l. mot smartkortet. For en del av de lokale anvendelsene kan bruk av PIN-kode være både overflødig og upraktisk. Når kortet samtidig er et ID-kort med foto er det lite behov for å verifisere at kortet ikke er i feil hender.

Rent teknisk kunne en nok ha en løsning der autentiseringsnøkkelen normalt beskyttes med PIN, men at PIN er unødvendig når kortet brukes i en tiltrodd og overvåket terminal som autentiserer seg overfor kortet. Så lenge det ikke er noen helt spesiell grunn til at det bare skal være én nøkkel, har dette lite for seg. (Bl.a. er dette knapt støttet av standarder og profiler for digital ID-applikasjoner. Disse har stort sett ikke rom for flere alternative måter å aktivere en nøkkel på.)

6.5 Andre smartkortanvendelser

Med multiapplikasjons-smartkort lar det seg gjøre å legge inn flere applikasjoner på samme kort, og legge til og fjerne applikasjoner dynamisk. (Til en viss grad er dette mulig med andre kort også: Selv fast programmerte kort kan ha flere applikasjoner lagt inn fra begynnelsen av. Det finnes også en rekke kort hvor selve kommandosettet er fast, men det er mulig å legge inn nye applikasjoner, filer, nøkler og andre objekter på kortene.) Inntil videre er imidlertid plassen på kortet en sterkt begrensende faktor.

Alle slike applikasjoner er i utgangspunktet uavhengige av digital ID-anvendelsen. (Det kan riktignok være indirekte koblinger på grunn av krav til plattform, programvare, type smartkort, leverandører osv.)

Ved siden av digital ID er det nok forskjellige typer betalingsanvendelser som er mest aktuelt:

- Bankkort eller kredittkort. (Smartkort-baserte debetkort og kredittkort er i prinsippet bare en annen type digital ID som bekrefter at kortholderen er knyttet til en viss konto.)
- Generelle småpengesystemer (Proton http://www.protonworld.com, MONDEX http://www.mondex.com, CEPS.)
- Lokale småpengesystemer (kantine, utskrift, kopimaskiner, salgsautomater o.l.)

For bankkort og småpengekort generelt er det ikke noen spesiell grunn til å legge dem på samme kort som digital ID, annet enn å spare inn på utstedelseskostnader og oppnå større utbredelse for sine smartkortløsninger. Poenget med å legge inn et lokalt småpengesystem vil være å beholde kontroll over systemet og de tekniske løsningene. For eksempel vil det kunne brukes til å håndtere kvoter på gratis bruk av ressurser som skrivere og kopimaskiner (ved at et visst beløp lastes inn i kortet uten at kortholderen betaler for det) eller for automatisk å gi rabatter (kortet inneholder en egen «valuta» som bare er gangbar lokalt, og som kjøpes til en verdi under pålydende). (Generelle småpengesystemer har også gjerne mekanismer som gjør at de kan brukes på samme måte, f.eks. muligheten for å ha en egen «pengepung» med begrensninger på hvor pengene kan brukes.)

Det finnes en rekke løsninger, stort sett leverandørspesifikke, for smartkortbasert innlogging (til operativsystem, tjenester i nett, VPN o.l.) I prinsippet dreier dette seg om autentisering av brukere, og FEIDEs elektroniske ID-applikasjon er i utgangspunktet velegnet, som vi har diskutert ovenfor. Mange av de løsningene som finnes, f.eks. i Windows 2000 og Suns SunRay, er imidlertid basert på vanlig innlogging med brukernavn og passord (eller CHAP, RADIUS eller lignende mekanismer), bare at passordet lagres i smartkortet. Med multiapplikasjons-smartkort kan det være mulig å legge inn smartkortapplikasjoner for disse anvendelsene på samme kort som FEIDE, men uavhengig av FEIDE-applikasjonen. Vi anbefaler imidlertid at man prioriterer å integrere FEIDE dersom det teknisk lar seg gjøre.

Hva som skal være tilgjengelig av andre applikasjoner på kortene kan antakelig være opp til de enkelte lærestedene. Men det betyr at når studenter eller ansatte beveger seg mellom lærestedene vil de av og til trenge å legge inn nye applikasjoner på kortene. En skal være oppmerksom på at det kan være noen konfliktsituasjoner, avhengig av hvilken (eller hvilke) multiapplikasjons-plattformer en velger og hvordan utstedelsen organiseres.

- Kortenes utsteder har ofte muligheter til å sperre kortene for installasjon av annet enn de applikasjonene de selv har godkjent.
- Flere av lærestedene kan komme til å bruke samme produkt (f.eks. småpengesystem, innlogging: SunRay el.l.) men med logisk distinkte installasjoner (f.eks. forskjellige sett av brukernavn og passord), og smartkortet kan ofte ikke holde to applikasjoner av samme type, eller terminalene klarer ikke å velge den rette av dem.
- Mange smartkort-anvendelser gjengjenner smartkort av rett type gjennom ATR-strengen (Answer To Reset), som kortet sender umiddelbart etter at kortleseren har resatt det. For multiapplikasjons-smartkort er det gjerne smartkort-operativsystem og utsteder (eller første applikasjon som legges inn) som bestemmer ATR, slik at applikasjoner som legges inn senere ikke vil reflekteres i ATR. Uansett vil det være vanskelig eller umulig å emulere smartkort der anvendelsen er laget for enkelt-applikasjons-kort og forutsetter at kortet gjenkjennes gjennom en bestemt ATR.

7. Krav til klient: Programvare og utstyr

En del av tjenestene bør være mulig å bruke uten noe annet enn en vanlig web- eller e-post-klient, uten noen spesiell tilleggsprogramvare eller annet ustyr. Problemet med dette er at det krever at de innebygde programvareløsningene for kryptografi i disse klientene brukes. Dette er ikke akseptabelt for nøkler og sertifikater som trenger sterk beskyttelse, og i særdeleshet ikke for nøkler som gir kvalifisert signatur eller en høy grad av ikke-benekting i det hele tatt. Dermed er det fare for at dette vil gjøre det nødvendig å ha flere sett med nøkler og sertifikater, med ulik sertifikatpolicy. (Verken i Netscapes eller Microsofts kryptomoduler har sikkerheten vært spesielt overbevisende. Ingen av dem gir noen åpenbar og grei mulighet for å plassere nøkkel- og sertifikatdatabase på en diskett -- som kan fjernes fysisk fra maskinen, og selv om basen er kryptert med en passfrase ligger den svært utsatt på lokal disk.)

Et alternativ er å kreve bruk av smartkort for all kryptografisk autentisering, men tillate innlogging med bruker-ID og passord (over SSL) eller andre autentiseringsmekanismer for enkelte mindre kritiske tjenester. Dette kan bygge på den samme infrastrukturen med unike bruker-ID og LDAP-kataloger som det sertifikatbaserte systemet.

På noe sikt tror vi ikke smartkort er et for sterkt krav. Som nevnt ovenfor skal en smartkortleser, når den først er installert, kunne brukes for de aller fleste aktuelle smartkortanvendelser, med kort av forskjellige fabrikat og fra forskjellige utstedere. Programvare-infrastrukturen, i form av drivere for kortlesere og kort og rammeverket som applikasjonsprogrammene benytter for å nå smartkort-tjenester, er også i ferd med å stabilisere seg. Prisene for kortlesere er ikke spesielt høy (det dreier seg om noen få hundre kroner), og vil i hvert fall ikke øke. Lisenskostnaden for tjenestemodulen som kreves for å tilpasse en bestemt type kort kan være av samme størrelsesorden, men det bør være mulig å forhandle seg frem til en gunstig pris i dette tilfellet.

Selve installasjonen er ikke spesielt komplisert: Typiske lesere på PC-plattform kobles til en COM-port, eller mellom tastatur og maskin, og drivere installeres på vanlig måte for Windows. I tillegg til installasjon av drivere, må også brukeren lokalt installere sertifikater, spesielt selvsignerte sertifikater for tiltrodde sertifikatautoriteter (CA-sertifikater). Det er neppe aktuelt å basere seg bare på de sertifikatautoritetene som er ferdig installert i web-leserne, selv i de tilfellene hvor en bare bruker de innebygde programvare-kryptoløsningene.

En stor del av de systemene/tjenestene som er aktuelt å nå har trolig allerede web-frontends, eller kan utstyres med det på en enkel måte, når bare brukerautentisering og dermed adgangskontroll kan løses på en akseptabel måte. Web-baserte brukergrensesnitt er etter hvert vanlig for databasesystemer og applikasjonstjenere, og nyere versjoner av mange av dem har SSL-basert brukerautentisering innebygd. (Der det ikke finnes innebygd støtte for SSL kan dette løses gjennom en frontend eller proxy.) Dermed regner vi med at web-leser med SSL klientautentisering dekker en ganske stor del av klientside-behovene.

Smartkort-baserte krypto-løsninger som dekker web + SSL for Netscape Navigator og Internet Explorer vil også dekke e-post-sikkerhet i hhv. Netscape Messenger og Outlook/Outlook Express. (Sannsynligvis vil de også kunne brukes av andre e-post-pakker på samme plattform.) Forutsetningen er selvfølgelig at det blir utstedt sertifikater som er gyldig for disse bruksområdene i henhold til keyUsage og andre attributter i sertifikatene.

Skal det brukes sikrede versjoner av andre protokoller og tjenester enn dette kreves det antakelig tilrettelegging fra UNINETTs eller driftsavdelingenes side, ved at egen eller tilpasset klientprogramvare gjøres tilgjengelig for nedlasting, proxy-løsninger blir satt opp eller lignende. Så langt har vi imidlertid ikke identifisert konkrete behov for noe slikt.

Behov for tilpasninger på klienten kan også dekkes ved forskjellige typer web-leser-utvidelser, klientside-scripting (Java, JavaScript) eller automatisk nedlastbare plugins. Men slike mekanismer er ikke uproblematiske, og spesielt ikke når det dreier seg om sikkerhetskritiske forhold. Spesielt er det verdt å merke seg at smartkort og -leser, og lokale kryptografitjenester i det hele tatt, i utgangspunktet ikke kan nås fra script og annen upriviligert kode; noe annet ville også være et grovt sikkerhetsbrudd. Se vedlegg B for en nærmere diskusjon. Digital signatur med sterk ikke-benekting krever i praksis at både presentasjon av dokumentet som skal signeres og selve signeringen er tett koblet i lokal, tiltrodd programvare på klienten, og det det finnes ikke noen enkle, generelt brukbare løsninger for dette i web- (eller WAP-)løsninger, se vedlegg C.

8. Transportlagssikkerhet: SSL og TLS

For sikkerhet på transportlaget er det i praksis SSL (Secure Socket Layer) [Freier, Karlton og Kocher 1996] , eller TLS (Transport Layer Security) [RFC 2246] som er aktuelt. Forskjellen på TLS versjon 1 og SSL versjon 3 er minimal i praksis, men det er TLS som vil bli Internett-standard, mens SSL er en proprietær spesifikasjon. Praktisk talt alle produkter implementerer SSL v3. TLS er noe mindre utbredt ennå, men det har liten praktisk betydning at mange produkter bare støtter SSL v3, siden TLS og SSL v3 er kompatible, både TLS-klient mot SSL-tjener og vice versa. (SSL v2 har en del sikkerhetsproblemer og bør fases helt ut.) Ellers i denne rapporten bruker vi derfor SSL som fellesbetegnelse for begge protokollene.

Det er en del forvirring i forbindelse med hva SSL er og hvor sikker protokollen er. Ikke minst når FEIDE skal presenteres utad kan det være verdt å presisere:

- SSL er ikke en protokoll for sikker betaling i forbindelse med elektronisk handel, selv om den ofte sammenlignes med f.eks. SET. (Dette er en nokså meningsløs sammenligning.)
- SSL er ikke inherent usikker. *Protokollen* i SSL versjon 2 har svakheter, men er utdatert og uaktuell. Enkelte av *krypto-algoritmene* som tilbys i SSL er usikre, og *implementasjoner* av SSL som bare tilbyr «export grade»-kryptografi kan ikke gi god sikkerhet.
- Slik SSL vanligvis blir brukt, er det uten klientautentisering.

Sikkerhetsprotokollen WTLS (Wireless Transport Layer Security) [WAP-163] i WAP er avledet fra TLS, og er nærmest bare en reformulering av TLS for en datagramorientert transportprotokoll. Meldingsformater og kryptografiske operasjoner er langt på vei identiske. (Dessverre ser det likevel ut til at WTLS har noen sikkerhetsproblemer som ikke finnes i SSL og TLS [Saarinen 1999]). Men mens SSL normalt brukes ende-til-ende, kan WTLS bare brukes mellom WAP-klient og WAP-portner. (WAP-portneren er en applikasjonslags-portner, typisk: WSP til HTTP.) I WAP må altså WAP-portner og applikasjons- eller innholdstjener bygges sammen for å oppnå ende-til-ende-sikkerhet, og dette er en stor begrensning.

SSL tilbyr tjenerautentisering og konfidensialitet (kryptering av sesjonen), og klientautentisering er valgfritt. (Tjener og klient i SSL henspeiler på hvem som er «initiator» og setter opp forbindelsen. Applikasjonsprotokollen som transporteres over SSL kan godt være «peer to peer».) SSL har dessuten en sesjonsmekanisme, som tillater sikre sesjoner å vare over flere separate TCP-oppkoblinger mellom samme klient og tjener.

Sikkerhetsnivået i en SSL-sesjon avhenger av *chiffer-suiten* som benyttes: En chiffer-suite definerer hvilke kombinasjoner av kryptografiske algoritmer og nøkkellengder som benyttes for hhv. kryptering og autentisering.

Når en SSL-klient kobler seg opp mot en SSL-tjener, gjennomgår de en forhandlingsfase som bestemmer sikkerhetsparametrene i sesjonen og setter opp nøklene som brukes for å kryptere og autentisere innholdet:

- Klienten foreslår en serie chiffer-suiter som den vil akseptere, og tjeneren gjør det endelige valget.
- Tjeneren tildeler en sesjons-ID.
- Tjeneren sender sitt sertifikat (som må stemme med valgt chiffer-suite).
- Dersom klientautentisering skal brukes, er det tjeneren som må kreve det. Tjeneren sender en liste av sertifikatautoriteter som den vil akseptere, og det er opp til klienten å svare med et akseptabelt klientsertifikat.

Både klienten og tjeneren kan avbryte forhandlingen (og dermed, sesjonen) dersom noe går feil, for eksempel dersom den andre parten velger for svake chiffer-suiter, ugyldige sertifikater eller sertifikater fra en sertifikatautoritet som ikke blir akseptert.

Både klienten og tjeneren kan ta initiativ til å reforhandle sikkerhetsparametrene for en aktiv sesjon. Dette kan brukes for å forsterke sikkerheten dersom applikasjonsprotokollen tilsier det. (Forhandlingen skjer første gang før noe data fra applikasjonsprotokollen er overført, slike at den f.eks. ikke kan avhenge av hvilken URL en HTTPS-klient ber om. Dersom klienten viste seg å be om en spesielt sensitiv URL kan tjeneren initiere reforhandling.)

Bruk av sikkerhetstjenestene, tilgjengelighet og valg av chiffer-suiter og reforhandling av sikkerhetparametrene er mekanismer som ligger i protokollen, men med stor grad av valgfrihet. Sikkerhetsegenskapene til en gitt implementasjon av protokollen avhenger av om og hvordan disse mekanismene er implementert og hvordan de er integrert med med en bestemt applikasjon og applikasjonsprotokoll over SSL.

8.1 Klientautentisering med sertifikater

SSL med klientautentisering støttes av alle de nyere versjonene av de aktuelle web-leserne, inkludert de versjonene som gir «sterk» kryptografi. For å kunne bruke klientautentisering, må brukeren bruke en sertifikatautoritet som utsteder slike sertifikater. Brukeren må selv laste inn sitt autentiseringssertifikat i web-leseren, og indikere at dette sertifikatet skal brukes for klientautentisering. (Brukes den innebygde kryptografimodulen vil normalt brukeren generere et nøkkelpar lokalt. Denne prosessen genererer også en sertifiseringsforespørsel som sendes til valgt sertifikatautoritet, og når sertifikatautoriteten returnerer sertifikatet kan det installeres automatisk. Brukes i stedet en smartkort-basert kryptografimodul, vil brukeren antakelig få smartkortet med nøkler og sertifikater ferdig installert og tilgjengelig for web-leseren uten videre.)

Klientautentisering aktiveres for en enkelt SSL-sesjon dersom SSL-tjeneren krever det. Brukeren må gjøre følgende:

- Velge hvilket sertifikat som skal sendes til tjeneren. (Hvis brukeren har flere sertifikater, og ikke ett av dem allerede er valgt som standard autentiseringssertifikat.) Dette bestemmer hvilken identitet tjeneren kjenner denne brukeren under.
- Taste PIN-kode for å aktivere den private nøkkelen som hører til sertifikatet. (Både for Netscape og Internet Explorer gjelder: Det er mulig å velge om PIN-koden skal tastes hver gang nøkkelen brukes, eller bare første gang etter at web-leseren er startet opp. For den innebygde nøkkeldatabasen er PIN-koden knyttet til nøkkeldatabasen, ikke til hver enkelt nøkkel.)

SSL-tjeneren er konfigurert med hvilke sertifikatautoriteter den aksepterer for klientautentisering, og setter først opp sesjonen dersom klientsertifikatet den mottar er gyldig og autentiseringen ellers er korrekt. Når SSL-sesjonen er etablert, brukes den normalt for flere påfølgende HTTP-oppslag fra samme klient (IP-adresse), slik at det ikke er nødvendig å gjenta autentiseringen.

Når web-tjeneren kommer så langt at HTTP-oppslaget behandles, f.eks. CGI-programmet kjører, er altså autentiseringen fullført og godkjent. Klientsertifikatet, eller i det minste et utvalg av informasjonen fra det, er tilgjengelig. Nøyaktig hvordan SSL-tjeneren konfigureres og hvordan autentiseringsinformasjonen er tilgjengelig varierer fra produkt til produkt.

Merk at klientautentisering i prinsippet er autentisering av klientmaskinen, ikke nødvendigvis brukeren som sitter ved den. Men slik SSL-støtten i web-klientene er utformet, fungerer klientautentisering i praksis også som brukerautentisering: Nøkkeldatabasen er beskyttet med et personlig passord/PIN-kode, og er evt. fysisk plassert på et personlig smartkort. Kun én bruker har adgang til maskinen om gangen. Klienten bør nok konfigureres slik at den ber om passord for nøkkeldatabasen for hver autentisering, ikke bare første gang.

SSLs persistente sesjoner kan være et lite problem på maskiner som flere brukere har adgang til, siden det ikke finnes noen eksplisitt mekanisme for å avslutte sesjoner (slette sesjonsnøklene). Det er et spørsmål om sesjonsnøklene kan bli liggende igjen i minne noe sted slik at de er tilgjengelig for andre programmer på samme maskin, muligens også etter at klientprogrammet som åpnet sesjonen har blitt avsluttet. (Brukes WTLS med WIM ligger sesjonstilstanden i smartkortet. Både med PKCS#11 og CryptoAPI er det i prinsippet også mulig å la sesjonsnøklene bli liggende i smartkortet, men om Netscape hhv. Internet Explorer faktisk er i stand til å utnytte det er ikke klart. Ytelsesmessig vil det imidlertid neppe være aktuelt å kjøre all kryptering og beregning av autentiseringskoder i smartkortet.)

SSL og andre sertifikat-baserte autentiseringsmekanismer identifiserer brukeren gjennom et X.509 DN. Det er derfor naturlig å basere brukerkataloger på LDAP, og støtte for det ser etter hvert ut til å være ganske utbredt. (Katalogen brukes da for å slå opp DN etter autentiseringen, og finne ytterligere opplysninger, f.eks. autorisasjonsinformasjon, om brukeren. Dessuten kan katalogen brukes for å finne sertifikater for sertifikatvalidering som en del av autentiseringen, men dette logisk sett en annen funksjon.) Legg merke til at den underliggende autentiseringsprotokollen i prinsippet er uvesentlig, så lenge resultatet av autentiseringen er at brukeren er identifisert gjennom DN (eller X.509-sertifikat), så disse konvensjonene vil ikke binde FEIDE til SSL som autentiseringsprotokoll.

8.2 Tjenerside-løsninger for klientautentisering

Den grunnleggende støtten for SSL og klientautentisering på tjenersiden er rett og slett adgang til informasjon om den innkommende SSL-sesjonen:

- Klientens sertifikat, eller utvalgt informasjon fra det, i første rekke subjektnavnet.
- Protokollversjon (SSL v2, v3, TLS)
- Valgt chiffer-suite
- Sesjons-ID

(Det er viktig å kunne kontrollere protokollversjon og chiffer-suite, henholdsvis fordi SSL v2 har sikkerhetsproblemer, og fordi enkelte av chiffer-suitene som kan velges er svake, f.eks. null-chiffere og eksport-grad chiffere med 40-biters symmetriske nøkler.)

Web-tjeneren Apache kan f.eks. kombineres med SSL-implementasjonen OpenSSL. (Det finnes 2 varianter av denne integrasjonen: Apache-SSL og mod_ssl.) Disse tilbyr:

- Klientsertifikat, subjektnavn (DN), utsteders sertifikat, utsteders navn og andre parametre blir gjort tilgjengelig for CGI-program gjennom CGI-variable på vanlig måte. (Dvs. som environment-variable, på samme måte som «&»-parametre gitt gjennom URL-strengen.)
- Klientautentisering kan konfigureres slik at den integreres med basis HTTP-autentisering. Klientens navn fra sertifikatet kontrolleres mot den samme brukerdatabasen, og aksesskontroll og CGI-programmer utføres på samme måte som om brukeren hadde brukt basis-autentisering og oppgitt brukernavn og passord.
- Konfigurerbare sikkerhetskrav (pr. katalog i URL-treet): protokollversjon, chiffer-suite, bruk av klient-autentisering, m.m. samt generelle boolske kombinasjoner av CGI-variable.

Flere nyere applikasjons-tjener-produkter har SSL-integrasjon som også støtter klientautentisering, f.eks. Oracle Application Server og Oracle 8i (Advanced Security), IBM WebSphere, Sybase Jaguar CTS, BEA WebLogic. SSL støttes gjerne for både HTTP, IIOP og andre (proprietære) protokoller (Oracle Net8, BEA/Tuxedo T3) på like linje.

Mange av dem har integrasjon med LDAP-katalog. I enkleste tilfelle brukes LDAP-katalogen bare for å verifisere at subjekt-DN fra sertifikatet virkelig svarer til en gyldig bruker. Katalogen kan også brukes for å avbilde DN til brukernavn (principals) eller finne autorisasjoner for en gitt DN, slik at de innebygde

aksesskontrollmekanismene i applikasjonstjeneren kan utnyttes. Alternativt må applikasjonskoden (CGI-programmer, Java servlets, Enterprise Java Beans, CORBA-objekter, PL/SQL-prosedyrer osv.) selv hente informasjon om klientens sertifikat og implementere sin egen aksesskontroll. Detaljene i løsningene varierer, men inntrykket er at stort sett støttes **autentisering** med identitetssertifikater rimelig bra, uten at det er noen spesielt problematiske bindinger til SA-strukturen, form og struktur på DN og lignende. Forskjellene ligger først og fremst på **autorisasjon** og **aksesskontroll**, dels fordi dette knyttes opp til mekanismer som er plattformspesifikke.

8.3 SSL-proxyer

For TCP-baserte protokoller kan klienter og tjenere uten SSL-støtte ofte sikres ved å bruke generelle SSL-proxyer, f.eks. SSLway. Dette kan brukes for TELNET, POP3, HTTP, LDAP m.fl. Kjernen av proxyen er en enkel SSL-klient eller tjener som tar mot en vilkårlig TCP-forbindelse på den ene siden og sender den videre som en SSL-sesjon på den andre siden, eller vice versa.

Denne teknikken er enklest å anvende på tjenersiden. Utfordringen her blir å transportere autentiseringsdata fra proxy til tjener. På klientsiden avhenger det av hvordan applikasjonsprotokollen og klienten støtter proxyer: Klienten må koble seg opp mot proxy i stedet for den virkelige tjeneren, og proxy må styres til å koble seg opp mot tjeneren. (For HTTP er dette relativt uproblematisk på begge sider: Protokollen støtter eksplisitt proxyer, og autentiseringsinformasjonen kan bringes videre i HTTP-hodet, f.eks. ved å simulere HTTP basis-autentisering. Men siden SSL-støtte er så utbredt i HTTP-tjenere og -klienter allerede er antakelig HTTP den minst aktuelle protokollen å bruke SSL-proxyer for.)

8.4 Tjenerautentisering

SSL brukes alltid med tjenerautentisering. (Det finnes riktignok en mulighet for å velge chiffer-suiter som ikke gir autentisering av tjeneren, men det er neppe aktuelt å bruke, i hvert fall i FEIDE-sammenheng.) I de vanlige tilfellene brukes tjenerautentisering bare som en ekstra kontroll på at klienten virkelig har nådd den tjeneren den hadde til hensikt å kontakte. Klientside-støtte for tjenerautentisering er ganske forskjellig fra tjenerside-støtte for klientautentisering, som ble diskutert ovenfor. På det tekniske nivået er den i og for seg ganske enkel, å kontrollere at det er samsvar mellom tjener-adressen (f.eks. navnet på web-tjeneren fra URL-en) og sertifikatet som ble mottatt. Utfordringen er mer på brukergrensesnitt-nivå: Brukeren forholder seg sjelden direkte til tjener-adresser og SSL-sesjoner, men må likevel ha kontroll over hvor data stammer fra og sendes til. Interaktive web-klienter løser dette typisk ved en kombinasjon av:

- Automatisk kontroll av at navnet i sertifikatet (konkret: *common name*, CN i DN) stemmer overens med nettverksadressen som ble brukt for å sette opp sesjonen (dvs. tjenernavn fra URL).
- Gi brukeren en eksplisitt mulighet til å undersøke tjener-sertifikater.
- Dialogbokser som ber om eksplisitt bekreftelse i en del nærmere angitte situasjoner (der man antar at brukergrensesnittet ikke ellers gir tydelig nok indikasjon på hvilken tjener som er involvert), og med noe valgfrihet for brukeren til å velge nivå, avhengig av situasjon og hvilken tjener som er inne i bildet.

Slike generiske løsninger er ikke helt tilfredsstillende, og svakheten ligger nettopp i de to siste punktene. (Det gjør ikke saken noe bedre at det har forekommet feil i implementasjonen av mekanismene for det siste punktet, slik at operasjoner som burde vært bekreftet har blitt gjennomført uten. Dette understreker egentlig bare at disse mekanismene er temmelig ad-hoc.) Konklusjonen er at når en bruker utfører kritiske operasjoner, spesielt dersom en sender passord eller andre sensitive opplysninger til tjeneren, er det viktig å kontrollere tjenersertifikatet eksplisitt og ikke bare stole på web-klientens advarsler. Dette har ikke noen direkte betydning for FEIDE, men en bør være oppmerksom på det når en utformer web-applikasjonene.

Alle web-tjenere som skal bruke SSL må imidlertid ha tjenerautentiserings-sertifikater som fungerer sammen med web-klientene. Det finnes Microsoft- og Netscape-spesifikke sertifikatutvidelser og andre konvensjoner, men såvidt vi vet går det rimelig greit å styre unna disse og sette opp tjenestene med sertifikater som fungerer like bra med begge disse klientene (og andre i tillegg). De viktigste kravene til sertifikater og nøkler er:

- Sikkerhetsnivået for sesjonen er ikke sterkere enn tjenerens private nøkkel, uansett hva klienten har, og tjeneren må altså ha minst 768-1024 biters RSA-nøkkel eller tilsvarende dersom kravet er sterk kryptografi.
- For at den automatiske kontrollen av sertifikat mot tjener-navn skal fungere, må CN i sertifikatets navn inneholde tjenerens fullt kvalifiserte domenenavn.

De vanlige web-klientene leveres med rot- og SA-sertifikater for en del (høyt profilerte) sertifikattjenester ferdig installert. Hvis en tjener-operatør ikke velger å bruke en av disse sertifikattjenestene, må brukerne selv installere sertifikater i sine klienter. Dette er ikke spesielt komplisert rent teknisk, men det er en sikkerhetsmessig kritisk operasjon som en ikke skal ta for lett på. Det beste er sannsynligvis å unngå sertifikatinstallasjon over nettet, og i stedet gjøre dette som en del av installasjonen av smartkort-støtteprogramvare, fra diskett eller CD-ROM.

Netscape og Microsoft har mekanismer kalt hhv. Global Server og Server Gated Cryptography, der spesielle tjener-sertifikater (som såvidt vi vet bare kan utstedes av bestemte logiske SA-er hos Verisign) åpner for sterk kryptografi også i eksport-versjonen av web-klientene. Disse mekanismene er lite aktuelle når klienter som i utgangspunktet har sterk kryptografi er tilgjengelige, og dessuten det finnes sertifikattjenester som tilbyr sterke tjener-sertifikater.

9. Virtuelle private nett

9.1 VPN-typer

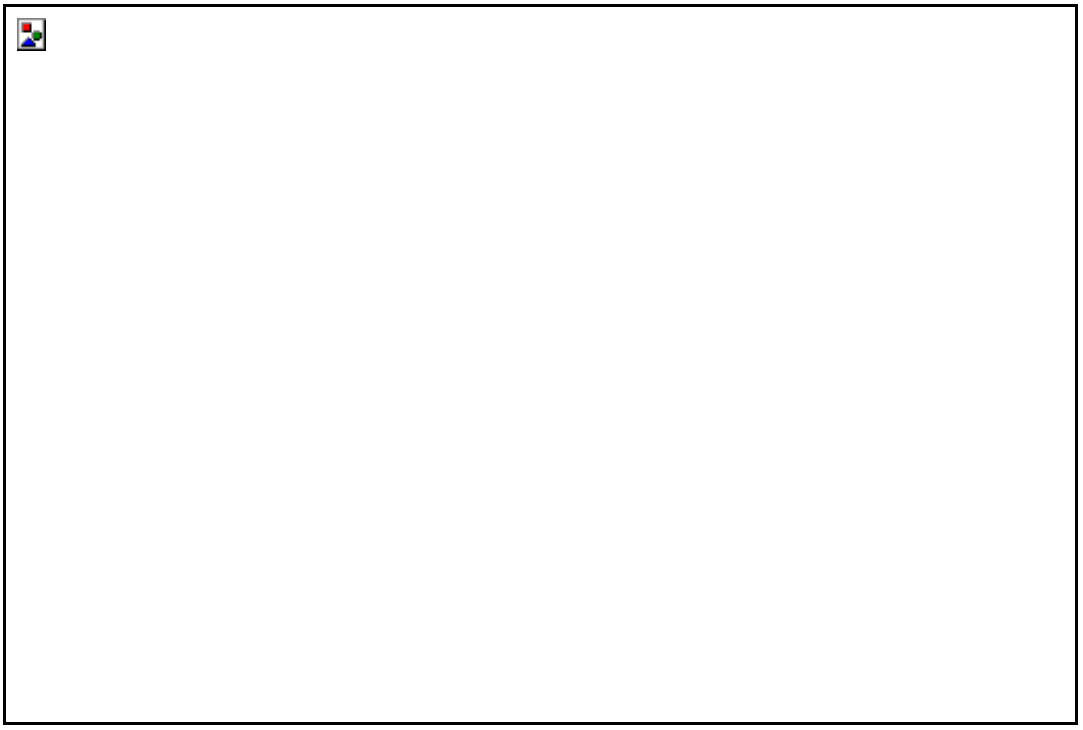
RFC 2764 [RFC 2764] klassifiserer IP-baserte VPN som følger:

- *Virtuelle leide linjer* (VLL), hvor en IP-tunnel erstatter en punkt-til-punkt leid linje.
- *Virtuelle private rutede nett* (VPRN), hvor IP-trafikk *rutes* gjennom IP-tunneller.
- *Virtuelle private oppringte* (dial) *nett* (VPDN), eller *aksess-VPN*, hvor en bruker kobler seg opp til en aksesstjener (PPP) gjennom et IP-nett, logisk sett på samme måte som via PSTN (telefon) eller ISDN.
- *Virtuelle private lokalnett-segmenter* (VPLS), som tilsvarer VPRN, bortsett fra at IP-tunnellene brukes på linklaget (av broer) i stedet for på nettverkslaget (av rutere).

En IP-tunnel er en virtuell forbindelse (assosiasjon) mellom to rutere/maskiner. Trafikken sendes fysisk som vanlige IP-pakker over Internettet eller et annet åpent nett, men innholdet består av IP- eller linklags-pakker (f.eks. PPP) som er kapslet inn, kryptert og/eller autentisert kryptografisk slik at IP-tunnellen sikkerhetsmessig blir som en dedikert linje. (VPN har andre aspekter enn sikkerhet, bl.a. tjenestekvalitet, men det er ikke relevant i denne diskusjonen.)

For VLL, VPRN og VPLS er endepunktene for IP-tunnellene maskiner, rutere, broer, ikke personer. For disse typene nett anvendes primært IPSec [RFC 2401][RFC 2411] alene. For nøkkeladministrasjon brukes gjerne en PKI, men kravene til PKI-løsningen avviker en del fra løsninger for personlige sertifikater (f.eks. PKIX [RFC 2459]). (Det foreligger et utkast [draft-ietf-ipsec-pki-req] til PKI-profil for nøkkeladministrasjonsprotokollene i IPSec -- ISAKMP [RFC 2408] , IKE [RFC 2409]). Både det at FEIDE-sertifikater er knyttet til personer, og den mer tekniske utformingen av sertifikatprofiler og PKI-løsningen gjør at FEIDE neppe er egnet til denne typen løsninger. Sannsynligvis er det behov for slike løsninger i forbindelse med FAS, f.eks. VPRN-løsninger for å knytte sammen nett med administrative systemer generelt, eller VPLS-løsninger for spesifikke systemer/tjenester som er tettere sammenknyttet på tvers av de enkelte lærestedene. Disse trenger sin egen PKI, som det trolig er best å behandle som en helt separat tjeneste i forhold til FEIDE.

Figur 5: Aksesstjener



Aksess-VPN, VPDN, brukes for tilknytning av eksterne brukere eller maskiner. Aksess-VPN er en slags generalisering av den vanlige modellen for Internett-tilknytning via oppringte samband (telefonlinje, ISDN): Brukeren ringer opp en *aksesstjener* (network access server, NAS) som er knyttet til et IP-nett. Trafikken over det oppringte sambandet er innkapslet i PPP-protokollen, og rollen til aksesstjeneren er primært å rute denne trafikken inn i og ut av IP-nettet. Aksesstjeneren står også for brukerautentisering og aksesskontroll på IP/port-nivå der det er behov for det. Dersom aksesstjeneren direkte er tilknyttet et internt nett, gir dette forsåvidt en så sikker tilknytning til dette nettet som det oppringte sambandet tillater.

En annen grunn til at en ønsker å ha aksesstjeneren nær nettet som den er inngangsport til, er at aksesstjeneren ofte brukes for brukerautentisering og aksesskontroll (på IP-adresse/protokoll/port-nivå). Aksesstjeneren trenger en sikker tilknytning til autentiserings- og autorisasjonsdatabasen (f.eks. via RADIUS [RFC 2865]), og dette er vanskeligere å få til dersom aksesstjeneren og databasen ikke er i samme (beskyttede) nett.

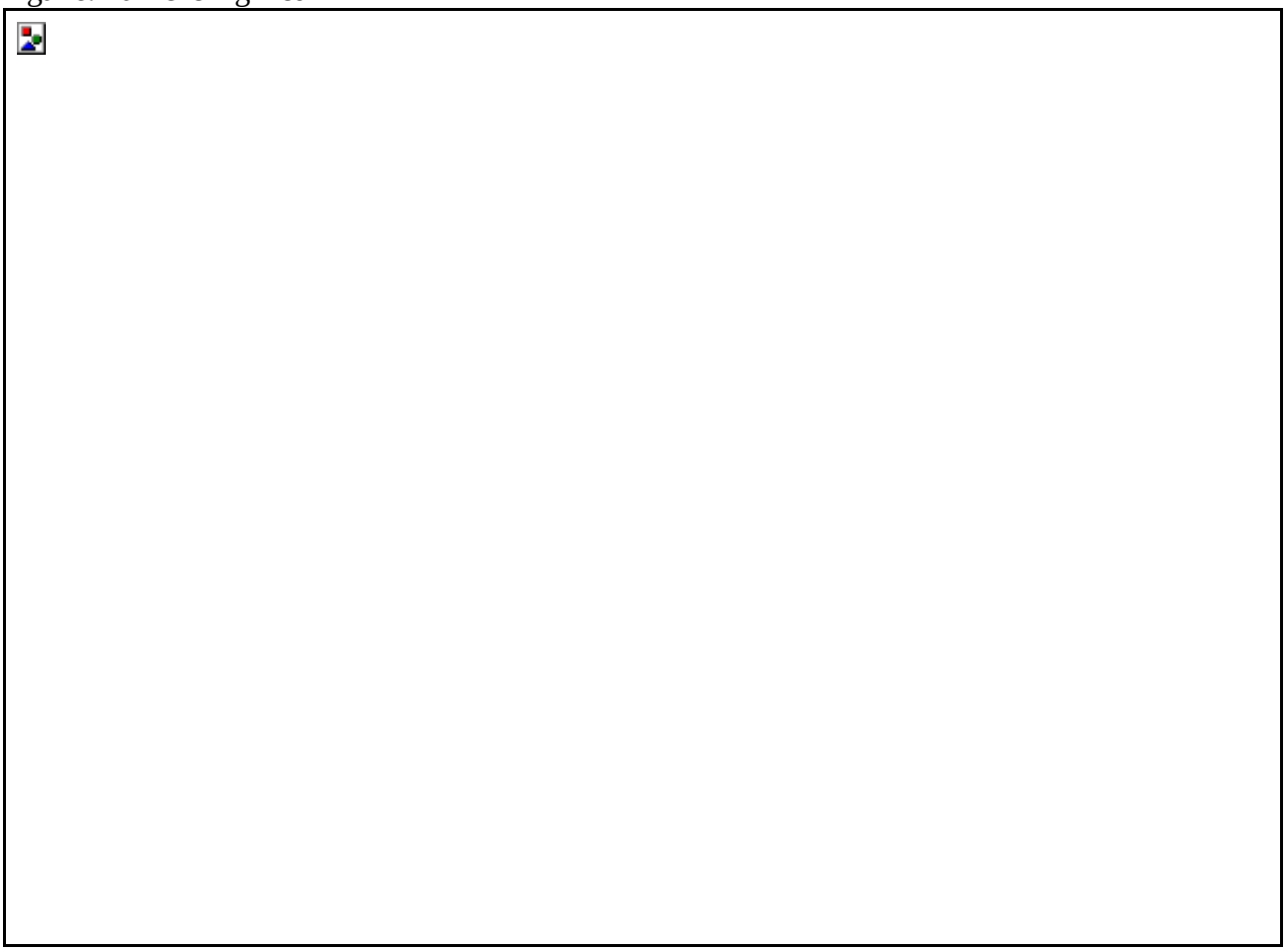
Men med dette kommer disse problemstillingene:

- Det er billigere å bruke oppringte samband lokalt, mens langdistansetrafikk transporteres over et åpent internett. Det er ikke fornuftig å ringe opp en aksesstjener i det nettet en til en hver tid skal ha trafikk mot. I stedet ønsker en å bruke en mest mulig lokal aksesstjener, som ruter trafikken videre over internettet. En slik aksesstjener betegnes *aksesskonsentrator* (L2TP access concentrator, LAC, hhv. PPTP access concentrator, PAC) når den sender PPP-trafikken videre i en IP-tunnel til en PPP-tjener.
- Sikkerheten i nettet som PPP-trafikken transporteres over er ikke alltid tilstrekkelig, og spesielt ikke når PPP tunnelleres gjennom et åpent internett.

Protokollene PPTP (Point-to-Point Tunneling Protocol) [RFC 2637] og L2TP (Layer 2 Tunneling Protocol) [RFC 2661] løser det første problemet: Med disse protokollene kan en sette opp en IP-tunnel for PPP-trafikk, f.eks. mellom en oppringt aksesskonsentrator og aksesstjeneren, men i prinsippet mellom

vilkårlige Internett-tilknyttede maskiner. PPTP er definert av Microsoft. Den finnes som en del av Microsofts RAS (Remote Access Server) og dessuten i en del uavhengige implementasjoner (se f.eks. http://www.moretonbay.com/vpn/pptp.html), og er dermed mest utbredt. L2TP, som er avledet av PPTP og Ciscos L2F (Layer 2 Forwarding), er på vei til å bli Internett-standard, og den som på noe sikt helst bør brukes.

Figur 6: Tunnellering med L2TP



9.2 PPP-nivå sikkerhetsprotokoller

PPTP og L2TP i seg selv inneholder ikke noen sikkerhetsmekanismer. Man kan enten sikre trafikken ved å bruke sikkerhetsmekanismer på PPP-nivå, dvs. inne i IP-tunnellen, eller sikre tunnellen selv ved å bruke mekanismer på IP-nivå (dvs. IPSec). Det har vært mest vanlig å bruke PPP-nivå sikkerhetsmekanismer, og f.eks. Microsofts PPTP-implementasjon er egentlig en pakke med PPTP + et utvalg autentiserings- og krypteringsprotokoller. (Av denne grunnen omtales PPTP og L2TP ofte som om de er sikkerhetsprotokoller, noe de egentlig ikke er, og det kan være litt forvirrende.)

PPP er utvidbar med hensyn på autentiseringsprotokoller. Den mest brukte er antakelig CHAP (Challenge Handshake Authentication Protocol) [RFC 1994]. Denne protokollen baserer seg på en felles hemmelighet (dvs. et passord) mellom brukeren og aksesstjeneren, og er ikke spesielt anvendbar i forbindelse med FEIDE. EAP (Extensible Authentication Protocol) [RFC 2284] kan brukes sammen med en rekke ulike autentiseringstyper, også token-kort, engangspassord o.l., men det finnes ikke noen standardisert type som direkte egner seg for autentisering ved offentlig nøkkel-kryptografi og X.509-sertifikater. (En eksperimentell RFC [RFC 2716] definerer en autentiseringstype avledet fra TLS. Det

finnes et utkast [draft-aboba-pppext-eapgss] til en autentiseringstype basert på GSS. Disse er ikke stabile, offisielle standarder. Dessuten er det registrert EAP-typer for bl.a. RSA offentlig nøkkel-autentisering og DSS hos IANA, se http://www.isi.edu/in-notes/iana/assignments/ppp-numbers, men etter alt å dømme for proprietære løsninger uten tilgjengelige spesifikasjoner.)

Link-nivå kryptering i PPP styres av ECP (Encryption Control Protocol) [RFC 1968]. På samme måte som for EAP kan forskjellige kryptografiske algoritmer og operasjonsmoder plugges inn i denne, og det finnes spesifikasjoner for DES [RFC 2419] og 3DES [RFC 2420]. (Det finnes ikke standardiserte nøkkelutvekslingsmekanismer for disse protokollene.)

Microsofts implementasjon av PPTP bruker Microsoft-spesifikke varianter av autentiserings- og krypteringsprotokoller for PPP, MS-CHAP [RFC 2759] og MPPE (Microsoft Point-to-Point Encryption). (Krypteringsnøklene avledes her som en sideeffekt ved autentiseringen.) I den første versjonen hadde disse en del svært alvorlige svakheter. Disse er til en viss grad rettet i andre versjon, men kritiseres likevel av blant andre Schneier [Schneier og Mudge 1999].

Konklusjonen er nok at sikkerhetsprotokoller på PPP-nivå i praksis er vanskelig å kombinere med en PKI. Det mangler standarder, og vi kjenner heller ikke til produkter som tilbyr slike kombinasjoner. Men i prinsippet passer denne typen løsninger når det en ønsker er sikker tilknytning for en *person* (i motsetning til en maskin), altså at autentisering og nøkkelutveksling for kryptering skjer ved hjelp av et personlig smartkort.

9.3 L2TP og IPSec

Alternativet til PPP-nivå sikkerhetsmekanismer er å kombinere L2TP og IPSec [RFC 2888]. Her er autentiseringsprotokollene i utgangspunktet utformet for kryptografisk autentisering, og inkluderer varianter som passer sammen med offentlig nøkkel-kryptografi. Dessuten er protokoller for nøkkelutveksling på plass. Dette regnes vanligvis som en anbefalt løsning for aksess-VPN, men den er ikke uten videre egnet når en ønsker å autentisere *personen* som sitter ved klienten som skal knyttes til nettet. Problemet er at IP-tunnellen, og dermed sikkerhetstjenestene, strekker seg fra LAC til LNS. Sambandet mellom klient og LAC dekkes ikke av sikkerhetstjenestene. Konkret: Hvis brukeren skal autentiseres med sitt FEIDE-kort, er det en applikasjon på LAC som må benytte smartkortet, og det er bare mulig dersom LAC og klient er samme fysiske maskin. En bruker som allerede bruker et oppringt samband for å knytte seg til nettet kan altså ikke autentisere seg med smartkortet.

9.4 VPN -- autentisering og aksesskontroll

Som tidligere nevnt: Siden autentisering og aksesskontroll i VPN foregår på nettverks- eller link-laget, er også de entitetene som autentiseres og de som er gjenstand for aksesskontroll de typene av entiteter som er kjent på dette nivået. Det er maskiner, ikke personer som autentiseres. Aksess kontrolleres til nett, maskiner, protokoller, porter, ikke til applikasjonsnivå entiteter som objekter, metoder, databaser, tabeller, tupler.

RADIUS [RFC 2865] er mye brukt i forbindelse med autentisering og aksesskontroll i oppringte samband. Formålet med RADIUS er å samle autentiserings/autorisasjonsdatabasen på en RADIUS-tjener, som deles mellom flere, kanskje geografisk spredte aksesstjenere. RADIUS er altså ikke en autentiseringsprotokoll i riktig samme betydning som f.eks. CHAP i PPP eller «handshake» i SSL. RADIUS svarer snarere til LDAP og OCSP, riktignok mer spesialisert til sitt formål. Når en klient skal autentisere seg overfor en aksesstjener, videresender aksesstjeneren klientens aksessforespørsel, brukernavn og passord eller andre autentikatorer til RADIUS-tjeneren. Det er RADIUS-tjeneren som verifiserer at autentiseringen er gyldig og at aksessen er autorisert. (M.a.o., denne forholdsvis sensitive

operasjonen -- og databasen som ligger til grunn for den -- er lokalisert til RADIUS-tjeneren, ikke eksponert på aksesstjenerne. Støtten for aksesskontroll i RADIUS er egentlig minimal, stort sett begrenset til å regulere om en gitt bruker har adgang til å bruke en gitt aksesstjener.)

RADIUS støtter både enkel passord-basert og utfordring/respons-type autentisering. Standarden støtter ikke eksplisitt offfentlig nøkkel-krypto, f.eks. RSA og DSA. (Dette kan håndteres ved å legge en spesiell tolkning på innholdet i utfordring- og respons-meldingene. Vi har ikke undersøkt om dette blir støttet av noen aktuelle produkter, men må i alle tilfeller anta at det ikke vil være interoperabelt mellom forskjellige produkter.) Passord sendes ikke i klartekst, og responser fra RADIUS-tjeneren er autentiserte, så RADIUS kan om nødvendig kjøres over et usikkert nett.

Når RADIUS (eller andre autentiseringsmekanismer) brukes sammen med en PPP-aksesstjener, er det i utgangspunktet bare adgangen til å bruke tjeneren som reguleres. Aksesstjeneren kan kombineres med brannvegg-løsninger på forskjellige måter. Aksesstjeneren kan også være integrert i et tjener-operativsystem på en slik måte at autentiseringen overfor aksesstjeneren samtidig tjener som innlogging til operativsystemet. (Det er f.eks. tilfelle i Microsofts RAS, og kan sannsynligvis også integreres nokså greit for tredjepartsprodukter mot operativsystemer som Linux og Solaris 8 med pluggbare autentiseringsmoduler.) Men slike løsninger er operativsystem og leverandørspesifikke.

Det kan være fornuftig å samkjøre bruker-ID i RADIUS og FEIDE. Hvert lærested kjører i utgangspunktet en frittstående og autonom RADIUS-tjener som de har full kontroll over, og som gir lokalt registrerte brukere adgang til lærestedets egne systemer. Det blir lærestedets eget valg om FEIDE-ID brukes som lokale bruker-ID og eventuelt om FEIDE-kortet med privat nøkkel brukes som autentiseringsmekanisme. For aksesser med FEIDE-ID som ikke er registrert lokalt kan RADIUS-tjeneren enten slå opp i FEIDE-katalogen vida LDAP, videresende til en felles FEIDE RADIUS-tjener eller evt. til en av de andre lærestedenes RADIUS-tjener. Standardene legger ikke noen spesielle føringer på hvordan dette kan gjøres, og hva som er praktisk og hvor mye interoperabilitet som er mulig å oppnå avhenger sannsynligvis ganske mye av utformingen av de aktuelle produktene. Blant annet må det etableres felles konvensjoner for hvordan aksessbegrensninger for ikke-lokale brukere konkret skal uttrykkes i RADIUS-protokollen og evt. for hvordan bruk av FEIDE-autentiseringsnøkler og sertifikater kan kapsles inn i RADIUS. Siden det ikke er noe uttrykt behov for denne typen autentisering og aksesskontroll på tvers av lærestedene vil vi i denne omgangen ikke gå videre inn på dette.

FEIDE-smartkortet kan benyttes til lagring av passord (CHAP/RADIUS). En del eksisterende produkter for «single sign-on» er rett og slett basert på at klartekst passord lagres i smartkort. (Gevinsten med dette er først og fremst at en kan bruke lange passord som brukeren ellers ville ha vanskelig for å huske, og at den forholdsvis enkle PIN-koden som gjør det mulig å bruke passordet bare brukes lokalt mot smartkortet. Med CHAP og/eller RADIUS behøver ikke passordet en gang å forlate smartkortet, men vi har ikke undersøkt om det finnes produkter som faktisk utnytter dette.) I alle tilfeller vil dette være en helt separat, uavhengig smartkort-applikasjon i FEIDE-kortet. En infrastruktur for passord-håndtering på tvers av lærestedene blir parallell til FEIDE-PKIen -- og sannsynligvis nesten like komplisert som den.

10. Smartkort- og kryptografiløsninger

Smartkort- og kryptografiløsninger, sertifikattjenester og applikasjonsprogramvare stammer gjerne fra forskjellige leverandører, og interoperabilitet mellom dem er ikke alltid gitt. Applikasjonsprogramvaren støtter typisk ett eller to ulike grensesnitt mot kryptografiske funksjoner, ofte ett proprietært og ett standardisert. Et annet produkt som realiserer ett av disse grensesnittene må velges, og dette må kunne håndtere alle sertifikater og nøkler som skal brukes. Formatene på sertifikater, smartkort som sertifikatautoriteten kan håndtere osv. setter ytterligere begrensninger her. Dessuten er det store variasjoner mellom produktene mhp. hvordan de tolker og implementerer standardene.

Interoperabilitet i smartkortløsninger krever tilpasning på to nivåer.

1. Smartkortleser: Leserne tilkobles på ulike måter, og har ganske forskjellige kommandosett, selv om protokollene mot kortene er standardiserte.
2. De applikasjonsrettede tjenestene som smartkortet tilbyr, for eksempel kryptografitjenestene, må gjøres tilgjengelig for applikasjonsprogrammer på en standardisert måte. (Det finnes ikke tilstrekkelig detaljerte standarder for hvordan disse tjenestene er realisert i smartkortene selv.)

I praksis betyr dette gjerne at det må installeres 2 sett med drivere. En driver for hver enkelt (type) smartkortleser, på samme måte som for andre ytre enheter knyttet til maskinen. Dessuten finnes ett eller flere sett med drivere (tjenestemoduler) for hver enkelt type smartkort, kanskje til og med for hver enkelt sertifikatautoritet som tilbyr sertifikater og nøkler på en bestemt korttype. I tillegg finnes et rammeverk som registrerer hvilke kort som befinner seg i leserne, mottar forespørsler om spesifikke tjenester fra applikasjonene og på bakgrunn av dette aktiverer de rette driverne.

Grunnleggende elektriske og mekaniske egenskaper og link-nivå protokoller mellom leser og kort er stort sett tilstrekkelig standardisert til at kurante kort og lesere av ulike typer fungerer sammen. (Det er likevel nok variasjonsmuligheter på protokollnivå til at det ikke alltid er tilfelle...)

Det er spesielt to industristandarder som er aktuelle:

- **PKCS#11**, som gjør det mulig å «plugge inn» tredjeparts kryptoløsninger i Netscape (og nylig også med sterk krypto).
- **PC/SC**, som gir en generell smartkortstøtte til applikasjonsprogrammer på Windows, og **CryptoAPI**, som kan utstyres med drivere for å bruke kryptografiske funksjoner fra tredjeparts kryptoløsninger. Disse blir tilgjengelig for alle applikasjoner under Windows som bruker kryptografiske sikkerhetsfunksjoner, inkludert Internet Explorer.

Med andre ord, for begge de to mest aktuelle web-leserne: Netscape og Internet Explorer, finnes det kurante smartkortløsninger. I prinsippet kunne nok universitets- og høyskolesektoren nøye seg med å henvise til tredjeparts utstyr og drivere som vil fungere sammen med løsningene for FEIDE, men en felles innkjøpsordning via Forvaltningsnettsamarbeidet er bedre.

En skal være oppmerksom på at det er store variasjonsmuligheter i hvordan disse standardene implementeres. Selv om applikasjonen forventer PKCS#11 (hhv. CryptoAPI) og krypto-modul tilbyr det samme grensesnittet, er de ikke nødvendigvis interoperable. Basis-funksjonene som hhv. Netscape og Internet Explorer bruker for signering og signaturverifikasjon støttes sannsynligvis av de fleste.

Vi har ikke gjort noe fullstendig produktsøk og -evaluering, men noen gjennomgående inntrykk er at:

- For praktisk talt alle kortlesere finnes PC/SC-drivere for Windows.
- De aller fleste leverandører av kryptografi-smartkort tilbyr CryptoAPI-støtte for disse. PKCS#11-støtte er også utbredt, men ikke i like stor grad som CryptoAPI.

- Kryptografiske applikasjoner, f.eks. tredjeparts epost/fil-krypto, sertifikatutstedelse og andre sertifikatautoritet-verktøy bruker først og fremst PKCS#11. Det er stort sett bare Microsofts egen programvare som bruker CryptoAPI.

For andre web-lesere er det ikke kjent hva slags smartkortstøtte som finnes. Opera Software og Telenor Conax har annonsert at Opera skal støtte Conax' smartkortbaserte PostSec-løsning. Det er ikke kjent om denne baserer seg på noen form for standardisert grensesnitt mellom web-leser og smartkortløsningen. I så fall ville dette rent praktisk bety at Opera kan brukes mot andre smartkortløsninger også.

En oppsummering av leverandørrollene i forbindelse med smartkortløsninger for kryptografi:

- **Leverandør av applikasjonsprogramvare** (f.eks. web-klienter med SSL, epost-systemer med signering og kryptering), forholder seg normalt bare til et standard grensesnitt som f.eks. PKCS#11, CryptoAPI, OCF eller lignende.
- **Leverandør av smartkortleser**, leverer fysisk maskinvare og drivere for denne med et standard grensesnitt som f.eks. PC/SC.
- **Sertifikatautoritet**, utsteder sertifikater og evt. nøkler.
- **Registreringsautoritet**, håndterer direkte brukerkontakt (kontroll av legitimasjon, utlevering av kort m.m.) på vegne av sertifikatautoritet og kortutsteder.
- **Leverandør av smartkort**, leverer ikke-personaliserte kort.
- **Leverandør av smartkortapplikasjon**, leverer applikasjonen som skal ligge i smartkortet samt tjenestemodulen (for PKCS#11, CryptoAPI, OCF) som driver applikasjonen.
- **Utsteder av smartkort**, personaliserer (installerer personlige data, nøkler, sertifikater m.m. i) kortene.

I prinsippet kan alle disse rollene være utført av forskjellige organisasjoner. Praktiske hensyn, f.eks. manglende interoperabilitet, gir imidlertid en del bindinger. Leverandøren av smartkort og smartkortapplikasjon er ofte den samme, men med multiapplikasjonskort eller andre programmerbare kort er det nettopp et poeng at dette er to distinkte leverandører. Vanligvis er disse underleverandører til kortutstederen, og brukerne forholder seg ikke til dem i det hele tatt. Kortutstederen er gjerne samtidig sertifikatautoritet, men i hvert fall vil sertifikatautoriteten måtte godkjenne kortutsteder og kortet og kort-applikasjonen som kortutsteder har valgt. Kortutsteder vil ofte også tilby smartkortleser, og det er praktisk første gang en tar i bruk smartkortløsninger å få en pakkeløsning som består av kort, leser, driver for leser og tjenestemoduler for kort og kortapplikasjon.

10.1 PKCS#11

Netscape støtter pluggbare kryptografimoduler gjennom grensesnittet PKCS#11 [PKCS#11], både på PC og UNIX-plattform. Nøkkelkarakteristikker ved PKCS#11:

- PKCS#11 definerer et API, som for Netscapes vedkommende er realisert gjennom et dynamisk lenket/delt bibliotek (DLL på Windows, .so på UNIX). API-et er formulert i C/C++, (men en Java-språkbinding er visstnok også under utarbeidelse.)
- Støtter bare kryptografiske tjenester, dvs. lagring av sertifikater og nøkler, (de)kryptering, hash-beregning og signering/verifikasjon, ikke andre smartkortbaserte tjenester.

En PKCS#11-modul som ikke er avhengig av smartkort, men implementerer alle funksjoner i programvare, er preinstallert i Netscape. For å støtte en ny kombinasjon av smartkort og leser på en gitt plattform, må leverandøren produsere sin egen PKCS#11-modul tilpasset denne kombinasjonen. (PKCS#11-modulen er ikke nødvendigvis monolittisk. Den kan godt være basert på standardiserte grensesnitt mot kortleser eller lignende, for den saks skyld PC/SC Resource Manager. Men Netscape forholder seg ikke til dette.)

Etter å ha koblet til leseren, installerer brukeren PKCS#11-modulen i Netscape. (Dette gjøres ved å fortelle Netscape filnavnet for PKCS#11-biblioteket gjennom «Create New Security Module»-dialogen. Evt. gjøres det helt automatisk av et installasjonsprogram: InstallShield eller lignende.)

PKCS#11 definerer et standardisert grensesnitt mot et bredt utvalg av kryptografiske funksjoner. En praktisk implementasjon av PKCS#11 vil bare tilby en delmengde av disse og noen kombinasjoner av parametre og variasjoner. (I noen tilfeller er det aktuelt å tilby nøyaktig de funksjonene som støttes i smartkortet. I andre tilfeller kan det være aktuelt å tilby smartkortets funksjoner og komplettere med funksjoner som er realisert i programvare internt i PKCS#11-modulen.)

Netscape er, for visse kombinasjoner av operasjoner, i stand til å kombinere funksjoner fra forskjellige moduler, men stiller noen krav om hvilke funksjoner som må finnes og hvordan disse er realisert for å kunne bruke en PKCS#11-modul til et bestemt formål (som f.eks. vår signaturanvendelse).

PKCS#11 er utviklet av RSA Data Security, og er uavhengig av Netscape. Andre leverandører som bruker PKCS#11:

- Baltimore Technologies tilbyr både kryptografimoduler med PKCS#11-grensesnitt (selv, eller i samarbeid med flere smartkort-leverandører), og applikasjoner som bygger på PKCS#11: MailSecure, som gir S/MIME-støtte i flere populære epost-klienter. FormSecure for digital signering av web-skjema. Utviklerverktøy for utvikling av kryptografi- og PKI-løsninger, bl.a. X/Secure for digital signering i XML-dokumenter.
- Entrust tilbyr også en omfattende serie med PKI-applikasjoner med PKCS#11 mot kryptomodulene: SA-verktøy, epost- og web-sikkerhet, utviklingsverktøy. (Entrust leverer ikke selv kryptomoduler.)
- Algorithmic Research leverer smartkort, -lesere, utviklingsverktøy m.m. med PKCS#11-grensesnitt.
- ID2 leverer PKCS#11 og CryptoAPI CSP-er for en rekke smartkort, samt ulike PKI-applikasjoner med PKCS#11-grensesnitt.
- Mange av leverandørene av smartkort leverer PKCS#11-drivere, blant andre: ActivCard, Bull, GemPlus, Schlumberger, Siemens

Det finnes to versjoner av PKCS#11. Den aktuelle versjonen er 2.1 eller høyere.

10.2 CryptoAPI

Dette API-et brukes av alle krypto-anvendelser i Windows, f.eks. SSL-sesjoner i Internet Explorer, kodesignering, kryptering og signering av e-post i MS Outlook. Smartkort og lesere som kan integreres i CryptoAPI blir dermed tilgjengelige fra et stort utvalg av applikasjoner. CryptoAPI er en del av Microsofts «Platform SDK» [Platform SDK], dvs. basisfunksjonalitet som er tilgjengelig for alle applikasjoner på Windows-plattformen. PKCS#11 og CryptoAPI er noenlunde sammenlignbare, men CryptoAPI (spesielt versjon 2.0) tilbyr en del høyere-nivå funksjoner som ikke finnes i PKCS#11, bl.a.:

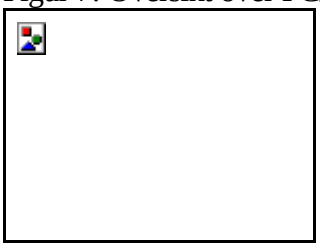
- Sertifikat- og CRL-håndtering. (PKCS#11 tilbyr bare lagring og attributt-aksess for sertifikater, applikasjonen må f.eks. håndtere sertifikatvalidering selv.)
- Meldingsorienterte funksjoner (generering og dekoding av PKCS#7-formaterte meldinger).

CryptoAPI i seg selv er uavhengig av smartkort. Selve implementasjonen av de kryptografiske algoritmene ligger i såkalte CSP-er (cryptographic service providers). Windows installeres med et sett ferdige CSP-er med ren programvare-implementasjon, med forskjellige algoritmer og nøkkellengder. «128-biter krypto»-oppgraderingen av Internet Explorer er i hovedsak installasjon av en ny CSP.

Smartkort integreres mot CryptoAPI ved å tilby en CSP som implementerer kryptografiske algoritmer på kortet. En slik CSP er samtidig en SSP (smartcard service provider) i PC/SC, slik at mekanismene i PC/SC brukes for å håndtere kortene. Smartkort-baserte CSP-er finnes ikke i Windows i utgangspunktet. Den som ønsker å gjøre krypto-tjenester i en bestemt type smartkort tilgjengelig for bruk gjennom CryptoAPI må altså lage en CSP for dette kortet.

10.3 PC/SC

Figur 7: Oversikt over PC/SC



PC/SC er en mer overordnet arkitektur [PCSC-1] enn PKCS#11. PC/SC er noe mer spesifikt rettet mot smartkort, men i motsetning til PKCS#11 dekker PC/SC alle aktuelle smartkortanvendelser, ikke bare kryptografi. Kryptografi er bare en kategori av smartkort-relaterte tjenester i PC/SC.

PC/SC definerer flere nivåer med grensesnitt. En «smart card service provider» (SSP) [PCSC-6] er en pluggbar programvaremodul som tilbyr en bestemt tjeneste (et bestemt grensesnitt) mot visse typer smartkort. En SSP for kryptotjenester kalles CSP («cryptographic service provider»), og kryptografi-tjenestene i CryptoAPI baserer seg på slike CSP-er. Alle SSP-er er COM-objekter, og installeres på samme måte som andre COM-klasser, i tillegg til at de må registreres i PC/SC slik at PC/SC gjenkjenner korttypen. (Dette gjøres av automatiske installasjonsprogrammer, og skulle være enkelt å installere for sluttbrukeren.)

Det lavere nivået i PC/SC kalles «resource manager» (RM) [PCSC-5] . RM tilbyr forskjellige funksjoner slik at applikasjonsprogrammene (f.eks. via CryptoAPI) kan finne aktive smartkort (eller be brukeren sette inn/velge kort), sette opp sesjoner mot og sende kommandoer til et valgt kort og laste og bruke en SSP som kan betjene kortet. RM styrer også reservasjon og deling av kort/kortleser mellom flere applikasjoner. Alt dette kan gjøres uavhengig av type kortleser og, til en viss grad, tekniske detaljer ved kortet. SSP-ene kommuniserer med kortet gjennom RM, slik at de ikke er avhengig av kortleseren.

RM tilpasses i sin tur til de fysiske kortleserne som finnes gjennom enhetsdriveren for kortleseren. Disse driverne installeres og håndteres på vanlig måte for enhetsdrivere under Windows. (Selve den fysiske tilkoblingen av kortleseren kan også skje på flere måter, bl.a. gjennom en vanlig serieport, mellom tastaturport og tastaturet eller integrert i tastaturet.)

PC/SC på Windows er gjennomgående basert på COM, og kan trolig brukes fra alle språk/omgivelser som har en noenlunde brukbar COM-integrasjon: Dette omfatter i hvert fall Visual C++, Visual Basic (inkludert Visual Basic-makroer i andre applikasjoner som f.eks. Office) og Microsoft Java.

Det arbeides også med å tilby PC/SC på Linux: http://www.linuxnet.com/smartcard/index.html.

De aller fleste kortleserne som er tilgjengelig har PC/SC-drivere, og mange av leverandørene av smartkort/smartkortapplikasjoner eller annen krypto-maskinvare tilbyr CSP-er for sine kort:

- ActivCard
- Bull

- GemPlus
- ID2
- Intel
- Schlumberger

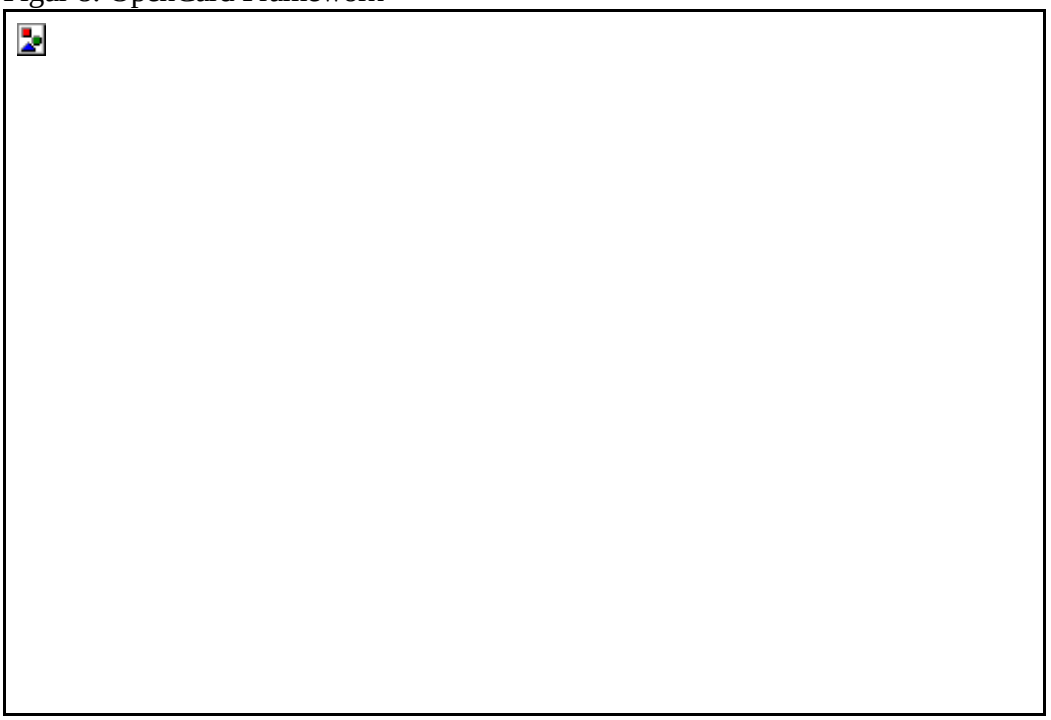
Mange av leverandørene tilbyr både PKCS#11 og CryptoAPI/CSP-grensesnitt mot sine kort. PKCS#11-implementasjoner på Windows bruker antakelig i stor grad PC/SC RM for aksess mot kortene.

Merk: Hvis flere applikasjoner skal dele kort eller kortlesere kreves at alle går gjennom den samme infrastrukturen, dvs. PC/SC RM på Windows. Applikasjonene må vanligvis reservere kortet for en viss periode, f.eks. etter at en PIN-kode er validert, mens de PIN-beskyttede kommandoene utføres, eller etter at en kryptografisk nøkkel er valgt, inntil alle data som skal krypteres er behandlet o.l. Denne reservasjonen er en av de viktige funksjonene i RM. Lavnivå låsing, f.eks. av serielinjen kortleseren er knyttet til, er ikke godt nok. (Når PC/SC RM kjører, tar den dessuten kontroll over alle tilgjengelige kortlesere slik at evt. drivere utenfor PC/SC-rammeverket ikke får tilgang overhodet.)

10.4 OpenCard Framework

OpenCard Framework (OCF) er et rammeverk for smartkort-anvendelser i Java. OCF er «ferskere» enn de andre industristandardene ovenfor, og støttes foreløpig ikke av noen standard-applikasjoner. OCF er heller ikke inkludert i Java-implementasjonene i hhv. Netscape og Internet Explorer, slik at OCF-bibliotekene må installeres separat. OCF 1.2 kjører under både Suns (1.1, 1.2) og Microsofts (4.x) Java-implementasjon.

Figur 8: OpenCard Framework



OCF har en arkitektur som i grove trekk svarer til PC/SC, men den er mer Java-spesifikk. På samme måte som PC/SC støtter den ulike smartkortbaserte tjenester, ikke bare krypto. En del generelle tjenester er predefinert i OCF (f.eks. verifikasjon av PIN-kode, aksess til «filsystemet» som de fleste smartkort inneholder, digital signatur), men det er også mulig å definere sine egne CardServices. Mekanismen for å velge kort og søke etter kort som støtter spesifikke tjenester er en del mer fleksibel enn PC/SC. Hver tjeneste *(CardService)* er definert ved et Java-grensesnitt *(interface)*, og for å støtte tjenesten på en bestemt type kort må en registrere tjenestemoduler i form av Java-klasser som implementerer de aktuelle grensesnittene.

Mens OCF gir generell adgang til smartkort, parallelt til PC/SC, gir API-et Java Cryptography Extension (JCE) adgang til kryptografiske tjenester fra Java, og svarer dermed til CryptoAPI. Nye kryptotjenester installeres i form av tjenestemoduler (i dette tilfellet, Java-klasser på samme måte som OCF, såkalte *Providers*). Den naturlige arkitekturen for smartkort-støttet kryptografi i Java blir da å bygge JCE *Providers* oppå OCF, dvs. på basis av OCF *CardServices*. (For øyeblikket finnes ikke noen slike, men det skal være utvikling i gang.)

En av standard-driverne for kortlesere i OCF går direkte mot PC/SC resource manager. Det betyr at alle kortlesere som støttes av PC/SC på en gitt Windows-installasjon også er tilgjengelige for OCF. (Andre drivere aksesserer typisk kortleseren direkte gjennom en vanlig serieport, basert på `javax.comm`-pakken.)

(En annen måte å integrere Java-kryptografi mot Windows på ville være å bygge JCE Providers basert på CryptoAPI. Såvidt vi kjenner til finnes ikke noen slike i dag.)

Solaris 8 støtter smartkort gjennom OCF. [Solaris-smartkort]

10.5 CDSA

The Open Group har spesifisert arkitekturen CDSA [CDSA] , med et API mot kryptografi-moduler som minner litt om PKCS#11 (men ikke er direkte kompatibelt). CDSA dekker sikkerhetstjenester i bredere forstand enn både PC/SC og PKCS#11, og ikke smartkort spesielt, eller andre smartkortanvendelser.

Støtte for CDSA ser ikke ut til å være særlig utbredt, og i hvert fall ikke i vanlige sluttbrukerverktøy. En referanse-implementasjon av CDSA er tilgjengelig fra Intel: http://www.intel.com/ial/security/

10.6 Standarder for kortene

PKCS#11, CryptoAPI og OCF definerer alle API-er mot tjenestemoduler og rammeverk på vertsmaskinsiden (klient-maskinen), og forutsetter at detaljene i grensesnittet mot selve smartkortet skjules inne i disse komponentene. For de fleste anvendelser er det fornuftig å programmere mot ett av disse API-ene, men ulempen er at for å kunne bruke en bestemt type smartkort eller smartkortapplikasjon må tjenestemodulene som håndterer kortet eller applikasjonen være installert på forhånd.

Det finnes også en del standarder som direkte definerer grensesnittet mot smartkortet, altså protokollene på ulike nivåer mellom kort og terminal. For det første har vi ISO 7816 del 1-3 [ISO 7816-1][ISO 7816-2][ISO 7816-3] som definerer fysiske karakteristikker ved kortene, elektriske egenskaper samt link-nivå protokoller. Dette er grunnlaget for at kort og lesere stort sett er interoperable. (EMV og GSM SIM-standardene har egne spesifikasjoner på disse nivåene. De er kompatible med ISO 7816, og så vidt vi vet er disse i praksis bare profiler som innskrenker valgfriheten en del i forhold til ISO 7816.)

ISO 7816 del 4-6 [ISO 7816-4][ISO 7816-5][ISO 7816-6] definerer en applikasjons-nivå protokoll for å sende kommandoer og motta respons fra kortet, samt en form for filsystem med kommandoer for å velge, lese og skrive filer, verifisere PIN-koder, autentisere av kort og terminal m.m. Alle standardene nedenfor bruker denne protokollen, og de fleste adopterer også filsystemet og mange av de generelle kommandoene fra ISO 7816 (men i noen tilfeller med mindre variasjoner.) Disse standardene er også basis for multiapplikasjonskort: Hver applikasjon i kortet er en «katalog» (ISO 7816-4: *dedicated file*) under rotkatalogen på kortet (ISO 7816-4: *master file*), og det å velge ut og kommunisere med en bestemt applikasjon skjer ved å skifte «stående katalog» (ISO 7816-4: *select file*-kommandoen). Alle applikasjoner som bruker applikasjonsprotokollen fra ISO 7816-4 og følger mekanismene for applikasjonsvalg derfra kan sameksistere i samme multiapplikasjonskort, selv om kommandoene til applikasjonene ellers ikke nødvendigvis følger ISO 7816. (Det ISO 7816 derimot ikke definerer like klart er hvordan terminalen skal kunne gjenkjenne eller finne en passende applikasjon i kortet: ISO 7816-4 definerer en fil med fast fil-ID under master file, kalt *directory file*, som skal inneholde ett dataobjekt, *application template*, for hver applikasjon. Innholdet i disse dataobjektene er bare et stykke på vei standardisert: ISO 7816-6 definerer en del primitive dataobjekter, som «kortholders navn», «kontonummer» o.l., men lite konkret semantikk. Dessuten åpner ISO 7816-4 for flere alternative måter å gjøre applikasjonsvalg på.)

ISO 7816 del 8 [ISO 7816-8] og 9 [ISO 7816-9] definerer flere kommandoer utover de som finnes i del 4, hhv. for å utføre kryptografiske operasjoner og for å administrere, opprette og slette applikasjoner og filer på kortet. Disse standardene er ferske og lite utbredt.

Det finnes også en rekke bransjestandarder for applikasjoner. De fleste av disse består av ett eller flere av disse elementene:

- En profil på ISO 7816 del 1-3.
- Kommandosett og filsystem basert på ISO 7816 del 4-6, men med noen tilpasninger og utvidelser.
- Applikasjonsvalg-mekanismer som er kompatible med ISO 7816 del 4 og 5, men med nærmere spesifikasjon av metadata og katalogstrukturer, og av prosessen som terminalen skal bruke for å identifisere og velge applikasjonen.
- Filformater og strukturer på innholdet i applikasjons-«katalogen».
- En del applikasjonsspesifikke kommandoer.

Vi oppsummerer noen av dem raskt:

- EMV [EMV-CARD][EMV-TERM][EMV-APPL] er en bransjestandard for debet/kredittkort-applikasjoner. (Litt spesielt med denne er at samme kort kan inneholde flere instanser av EMV-kompatible applikasjoner, og standarden går i adskillig detalj på hvordan terminalen skal velge mellom dem.)
- GSM SIM [TS 100 977] og SIM toolkit [TS 101 267] . (Disse definerer først og fremst filer og filformater for å inneholde informasjonen i SIM-kortet, og et kommandosett som stort sett er det samme som ISO 7816-4, men med noen små variasjoner. Dessuten finnes egne kryptografiske operasjoner, uavhengig av ISO 7816-8, for autentisering mot GSM-operatøren og kryptering av samtalen. SIM toolkit definerer mekanismer slik at håndsettet kan motta kommandoer fra SIM-kortet.)
- De svenske standardene utviklet av SEIS [SS 61 43 30][SS 61 43 31][SS 61 43 32] , som også Forvaltningsnett-samarbeidet bruker. Disse standardene definerer filformater og strukturer, adgangskontroll-regler m.m. for å representere sertifikater og nøkler for en digital ID-applikasjon på kortet, men uten å definere selve de kryptografiske kommandoene. (Strengt tatt defineres heller ikke filaksess-kommandoene, men det forutsettes at de dekkes av ISO 7816-4.)
- PKCS#15 [PKCS#15] er en parallell standard til PKCS#11. PKCS#15 er et noe mer generelt alternativ til SEIS, som tilsvarende SEIS definerer datastrukturene i kortet, men ikke (kryptografiske) kommandoer.
- WAP WIM [WAP-160] definerer en kryptografisk modul for WAP, spesielt for å understøtte WTLS og `Crypto.signText` i WMLScript. WIM er basert på PKCS#15, med kryptografiske operasjoner fra ISO 7816-8. (WIM er foreløpig bare på «proposed standard»-nivå, og det finnes neppe noen tilgjengelige implementasjoner av den.)

For FEIDE er de aktuelle kort-standardene SEIS eller evt. PKCS#15, men dette valget er egentlig litt underordnet. I alle tilfeller vil det være nødvendig å installere tjenestemoduler for den valgte kort-typen på klientmaskinene. Dersom FEIDE skal håndtere flere typer kort (f.eks. ulike leverandører) må en altså regne med å installere tjenestemoduler for alle aktuelle kort på klientmaskinene. Dermed er det mer vesentlig å standardisere på grensesnittet mellom tjenestemodul og applikasjonsprogram, dvs. PCKS#11/CryptoAPI/OCF-nivået, og på de logiske funksjonene som kortet tilbyr, enn akkurat på de konkrete kommandoene og filstrukturene kortet har.

Problemstillingen er imidlertid ikke bare hvilken standard FEIDE-applikasjonen selv skal følge, men også hvilke andre applikasjoner og standarder den skal kunne sameksistere med. Skal man f.eks. kunne ha en EMV-applikasjon på samme kort som FEIDE, må en bank-anvendelse (f.eks. en minibank) som følger EMVs spesifikasjoner for å gjenkjenne og velge EMV-applikasjonen ikke bli forvirret av FEIDE-applikasjonen, og vice versa. Og omvendt, hvis FEIDE-kortene skal kunne brukes for anvendelser som også godtar andre typer elektronisk ID, må man unngå at de kan forveksles med noen av de andre typene (med mindre de er **helt** kompatible, og f.eks. kan håndteres av samme tjenestemodul). ISO 7816-4 og -5 sikrer dette, men bare et stykke på vei, og dessuten risikerer man å komme bort i kort som ikke fullt ut oppfyller disse standardene.

10.7 Tjenerside-kryptografiløsninger

Kryptografiløsninger for tjenersiden er ikke programvaremessig veldig forskjellig fra klientside-løsninger. I prinsippet er det kanskje noe mindre behov for standardiserte grensesnitt mot pluggbare kryptografimoduler. (På tjenersiden er det et mindre problem enn på klientsiden om kryptografiløsningen må tilpasses applikasjonsprogramvaren.) I praksis er nok likevel grensesnitt som PKCS#11, CryptoAPI og JCE like aktuelle der som på klientsiden. Forskjellen er i første rekke hvordan disse grensesnittene implementeres:

- Smartkort kan også brukes på tjenersiden. Problemet med dette er i første rekke ytelsen på smartkortet.
- Rene programvareløsninger **kan** være mer akseptable på en beskyttet tjener.
- Høyt belastede tjenere kan trenge kryptografiske ko-prosessorer, dvs. spesiell maskinvare (i form av egne kort, eller til og med, separate tjenere) som har samme logiske funksjon som smartkort.

Selv om mekanismene er de samme, blir sikkerhetspolicy for klient- og tjenerside imidlertid helt vesensforskjellig. På klientsiden er normaltilfellet at hver autentisering (signaturgenerering, dekryptering) skjer under direkte kontroll av den ansvarlige brukeren, dvs. det er rimelig å oppfatte det som at det er personen -- ikke maskinen -- som blir autentisert. På tjenersiden er dette aldri tilfelle: Maskinen er knapt under kontinuerlig overvåkning i det hele tatt, og hvis den er det, er det gjerne av en driftsavdeling som ikke har noe direkte ansvar for de forretningsmessige funksjonene som tjenesten tilbyr. I alle fall utføres de kryptografiske operasjonene uten at noen personlig bekrefter hver enkelt. En annen forskjell, som går i samme retning, er at klient-sertifikater normalt er personlige, mens tjener-sertifikater identifiserer en organisasjon eller lignende.

(Vi vil presisere at det vi snakker om her er typiske tilfeller, og klient vs. tjener i mer overordnet forstand enn på protokollnivå. Det som i denne sammenhengen er å oppfatte som en tjener, kan godt operere som f.eks. en SSL-klient i forhold til en annen tjener, og i så fall er heller ikke «klientautentiseringen» personlig. Og selv om det ikke er vanlig, er det ikke noe i veien for at en personlig maskin, altså en «klient», kjører en SSL-tjener der brukeren må bekrefte hver ny sesjon med PIN-kode. Dette tilfellet er ikke så vidløftig som det kanskje kan høres ut, f.eks. kan det oppstå dersom et CORBA-objekt i en tjener har behov for å kalle tilbake til et objekt som befinner seg på klientmaskinen.)

Forskjellene i sikkerhetspolicy nedfeller seg i forskjeller i beskyttelsen av nøkler. På klientsiden er det helt rimelig å kreve PIN-kode hver gang en privat nøkkel brukes, eller i hvert fall for hver sesjon hvor smartkortet med nøkkelen blir brukt. På tjenersiden er dette åpenbart ikke mulig. Kravet om å bekrefte hver enkelt gang nøkkelen brukes kan egentlig føres tilbake til to delvis uavhengige behov:

- Verifisere at det er rettmessig eier som er i besittelse av og prøver å bruke nøkkelen.
- Gjøre det mulig for eieren å kontrollere at nøkkelen bare brukes til formål eieren har godkjent (f.eks.at den ikke brukes til å signere noe uten at eieren vet om det).

Det første er ikke noe særlig problem for tjenermaskiner. Vi forutsetter at maskinen er fysisk utilgjengelig for uvedkommende, og at den også er godt nok beskyttet mot innbrudd via nettet. Og hvis situasjonen faktisk skulle være at en uvedkommende har tatt kontroll over maskinen, er det ikke lenger bruk av den kryptografiske nøkkelen som er problemet: I så fall har angriperen uansett adgang til det nøkkelen skulle beskytte, f.eks. sensitive data som klientene kan finne på å sende til tjeneren, eller andre måter å misbruke klientenes tillit. **Men dette forutsetter at hver nøkkel er utstedt til en enkelt maskin eller tjeneste.** Dette bør være en grunnregel for tjenerautentisering og annen tjenerside kryptografi, på samme måte som at andre sertifikater og nøkler normalt skal være personlige, også der man egentlig bare har behov for å identifisere en person som medlem av en gruppe.

I tjenerside-sammenheng dreier det andre punktet seg om muligheten for misbruk av tjenesten utenfra. Det ligger i tjenestens natur at den kan brukes uten noe manuelt inngrep på tjenersiden, og utformingen av tjenesten, inkludert sikkerhetsmekanismer må selvfølgelig være i samsvar med det. Om det er mulig, og hvor mye som kan legges inn av beskyttelsesmekanismer som forhindrer misbruk i forkant avhenger helt av tjenesten. I stedet må kontrollen ligge i etterkant, ved logging av hver bruk av sensitive nøkler. Dersom dedikerte krypto-tjenere eller annen spesiell maskinvare benyttes, kan loggingen gjøres uavhengig av eventuelle angrep på tjenestemaskinen.

Det anbefales imidlertid at tjeneste og kryptoløsning settes opp slik at PIN-kode, passfrase eller lignende verifikasjon må brukes ved oppstart (eller omstart) av tjeneren. Det betyr for det første at tjeneren lett kan

settes i en tilstand hvor misbruk av nøklene pr. definisjon ikke er mulig, og hvor fysisk adgang til maskinen ikke er noen spesiell sikkerhetsrisiko. Dessuten kan beskyttelse på operativsystemnivå sikre at det bare er den prosessen eller brukeren som blir tiltrodd adgang til nøklene ved oppstart som er i stand til å bruke dem.

10.8 Kryptografiske løsninger og eksportrestriksjoner

Amerikanske eksportrestriksjoner har inntil nylig gjort det umulig å inkludere sterk kryptografi i amerikanske produkter for internasjonal bruk i et massemarked. Dette har rammet både Netscapes og Microsofts web-lesere og epost-klienter, og gjort det generelt vanskelig å inkludere sterk krypto som infrastruktur i vanlige operativsystem-plattformer. Også andre land har restriksjoner på bruk eller eksport av kryptografi.

Nå er de amerikanske eksportrestriksjonene vesentlig lettet. Netscape Communicator i siste versjon er tilgjengelig internasjonalt med sterk krypto. For Microsoft Internet Explorer og Outlook finnes også en 128-bits-oppgradering. (Denne oppgraderingen er først og fremst installasjon av en ny standard-CSP for CryptoAPI, slik at andre programmer som benytter kryptografi også får glede av den.) Med disse oppgraderingene tilbyr Communicator, Internet Explorer og Outlook de samme kryptomulighetene som i den amerikanske («domestic») versjonen, dvs. sterk innholdskryptering (128 biters RC4, opp til 168 biter Triple DES) og sterk digital signatur/autentisering (1024 biters RSA). Windows 2000 (og trolig Windows ME) inneholder denne CSPen og gir sterk kryptografi i utgangspunktet, uten noen oppgraderinger.

Utvikling av CSPer til CryptoAPI er underlagt restriksjoner på grunn av de amerikanske eksportreglene. Utviklingsomgivelsen (SDK) for CSP-utvikling, altså biblioteker, headerfiler osv. er ikke fritt tilgjengelig, og CSPer som blir utviklet av tredjeparter må signeres av Microsoft. CSPer som ikke oppfyller eksportkontrollbetingelsene blir ikke signert, og vil da ikke kunne brukes under CryptoAPI. Det er uklart om dette blir endret nå. (For Netscape var det ingen tilsvarende restriksjoner, siden begrensningene på krypto-styrken lå i Navigator selv, ikke PKCS#11-modulen.)

Eksportreglene har imidlertid tillatt eksport og signering av CSPer som tilbyr sterk kryptografi bare for digital signatur. Microsofts innebygde CSP i eksportversjonen av CryptoAPI gjør ikke det (antakelig fordi denne tilbyr både signering og kryptering med de samme nøklene), men det finnes smartkortbaserte CSP-er som tilbyr signering med 1024-biters RSA-nøkler. (F.eks. ID2. Microsofts egen fortegnelse over CSP-leverandører ser ikke ut til å være komplett.) En slik CSP vil være tilstrekkelig for mange formål, når det ikke er behov for sterk innholdskryptering. Vi antar det ganske raskt vil bli smarkort-CSPer tilgjengelig med generelt sterk krypto.

JCE 1.2 fra Sun har ikke vært tilgjengelig utenfor USA, men det finnes uavhengige leverandører i andre land. JCE 1.2.1 finnes i en eksporterbar versjon, men det er trolig fordi den ikke inkluderer tjenestemoduler som gir sterk krypto, ikke nødvendigvis på grunn av lettelsene i eksportrestriksjonene. Såvidt vi vet er det imidlertid ikke noen restriksjoner på utvikling av tjenestemoduler for JCE. Det finnes også en åpen kildekode-implementasjon av JCE fra Cryptix.

Eksportrestriksjoner er også grunnen til at SSL-støtten til Apache distribueres som et sett med modifikasjoner til standardutgaven. (Apache, med modifikasjonene for å kunne bruke OpenSSL, men selv uten at OpenSSL med selve krypto-funksjonene var inkludert i distribusjonen, ville ikke kunne eksporteres fra USA.)

11. Anbefalinger for FEIDE

Vi ser på krav til sertifikattjeneste etter de «definerende elementer» som ble gjennomgått i kapittel 4 : Sertifikatpolicy, sertifikatpraksis, sertifikatformat, navngivning, tillitsmodell, katalogtjenester -- og tar også med en diskusjon rundt RA (registreringsfunksjonen) og smartkort. Deretter ser vi på valg av sertifikattjeneste ut fra tilgjengelige alternativer.

11.1 Sertifikatpolicy og sertifikatpraksis

11.1.1 Generelt

Ved å velge smartkort som nøkkelbærer har en indikert et sikkerhetsnivå som er høyere enn det de enkleste tjenestene i markedet kan tilby. Det er derfor nærliggende å velge et forholdsvis høyt ambisjonsnivå for sikkerhet / kvalitet av sertifikattjenesten for FEIDE. Men vi anbefaler som før nevnt å holde seg unna begrepet «kvalifisert sertifikat».

Det er helt klart at en vil forlange personlig oppmøte for å få utstedt en ID -- det er praksis alt i dag for utstedelse av fysiske studiekort.

I forhold til eksisterende tjenester / spesifikasjoner må en først og fremst sjekke at policyens formål, anvendelser og paragrafer om utsteder (og andres) forpliktelser og ansvar er akseptable. Personvern mm. må også være ivaretatt, og det som eventuelt står om sertifikatprofil, må kunne brukes av FEIDE. Videre er RA-funksjonen viktig i en UoH-sammenheng, og de skisserte prosedyrene må være fleksible nok.

Merk at en 100% må følge en policy for å kunne påberope seg denne. Ved det minste avvik må en definere en ny policy, og dermed egentlig en ny tjeneste. Men i forholdet til sertifikatpraksis er det full fleksibilitet. En kan godt ha en egen (eller et sett av) egne sertifikatpraksiser for UoH-sektoren så lenge disse ikke er i konflikt med policy. En tjeneste som er lite spesifikk i policyen, og overlater mer til praksisen, er derfor mer fleksibel, men også (sikkerhetsmessig og ellers) mer «ullen».

Vi har ikke hatt ressurser til å vurdere forskjellige, tilgjengelige policyer opp mot FEIDEs krav, men dette er en opplagt oppgave i neste fase av arbeidet.

11.1.2 Registreringsfunksjonen og krav til smartkort

Registrering av nye brukere ved personlig frammøte er en litt spesiell problemstilling for UoH-sektoren, og diskuteres derfor separat. Det som er klart, er at hele logistikken rundt bruk av smartkort som skal kombineres med et fysisk studiekort, kanskje magnetstripe, og evt. andre funksjoner på kortet, er komplisert, og en kritisk faktor. Det vil også være forskjeller fra lærested til lærested.

UiO ønsker å styre prosessen selv, slik at studenter kan møte fram og legitimere seg, og så få utstedt ferdig studiekort mens en venter. Dette krever et lokalt produksjonsapparat som kan håndtere personalisering (kanskje også initialisering) av smartkort i tillegg til prosessen med trykking / preging av kortene. SA-tjenesten behøver ikke å forholde seg til smartkortene i det hele tatt, evt. kan denne ta ansvaret for kortleveranse med produksjon og initialisering. SA mottar ellers bare en sertifikatforespørsel fra f. eks. UiO, og utsteder de nødvendige sertifikatene.

Nøkler kan legges i smartkortene under produksjon eller initialisering. Ved registrering plukkes et kort fra lageret, en henter de offentlige nøklene som tilsvarer de private i det valgte kortet, og sender forespørsel til SA for sertifikater for disse med korrekt identitet. Sertifikatene returneres i løpet av maksimalt minutter, og kan legges inn i kort og kataloger (hvis ikke SA styrer katalogen, se nedenfor). (Sertifikatene kunne også være produsert på forhånd og legges inn i kortene sammen med privat nøkkel, slik at kortet, bortsett fra

preging, ligger klart til avhenting. Men dette ville kreve vesentlig strammere sikkerhetsopplegg rundt kortene på lager: Uten sertifikat er de egentlig ikke sensitive.)

Dersom studentene skal ha valgmuligheter med hensyn til hvilke funksjoner som skal ligge i kortet, må registreringsfunksjonen også gjøre initialisering av kortene. Det kan tenkes at både elektronisk ID, småpengekort, og andre, framtidige funksjoner skal kunne velges, eller velges bort, individuelt. Antagelig kan en ikke dytte på noen en elektronisk ID uten at personen selv ønsker det -- det innebærer tross alt en viss risiko for brukeren med hensyn på misbruk.

Mindre læresteder vil ikke ha ressurser til å kjøre denne prosessen lokalt. Her vil en i stedet foreta en «enkel» registrering av studentene og samle inn all nødvendig informasjon, inkludert bilde og signaturprøve. Dette overføres, gjerne over nettverk forutsatt god beskyttelse, til SA, og eventuelt til smartkortleverandør dersom dette er en annen enn SA. Kortene produseres der, og returneres i vanlig post enten til lærestedet med avhenting ved personlig oppmøte, eller direkte til studenten etter sikre prosedyrer (dette vil være både kortet selv og PIN-kode for adgang til elektronisk ID). En slik «enkel» RA-funksjon vil tilbys av alle aktuelle leverandører, og skulle ikke medføre noen problemer.

Men en er altså avhengig av at policy og leverandør er i stand til å tilby et fleksibelt opplegg for RA.

11.1.3 Sertifikatprofil

Bortsett fra navngivning, som diskuteres i neste avsnitt, har FEIDE få spesifikke krav til sertifikatprofil. Her kan en være ganske fleksibel mhp. alternativer.

Det skal være separate nøkkelpar til hvert formål, ikke minst av sikkerhetsmessige grunner. Privat nøkkel for digital signatur må beskyttes med en separat PIN-kode som må verifiseres på nytt for hver enkelt signatur som genereres, m.a.o. at brukeren alltid har eksplisitt bekreftet hver enkelt signatur. For autentiserings- og krypteringsnøkler er det tilstrekkelig med en PIN-kode som verifiseres én gang etter at kortet er satt i leseren, og brukerne vil antakelig oppfatte det som upraktisk om PIN-kode må tastes inn på nytt for hver enkelt autentiserings- eller dekrypteringsoperasjon som skjer. Bruk av signaturnøkkel bør forutsette at brukeren får anledning til å lese og kontrollere det som signeres, mens dette er upraktisk for autentisering og dekryptering.

En løsning med tre nøkkelpar / sertifikater er sikkerhetsmessig den beste, men vi har problemer med å anbefale dette p.g.a. manglende produktstøtte. En bedre anbefaling er antagelig to nøkkelpar, der digital signatur er skilt ut som en egen funksjon, mens autentisering og nøkkelutveksling (dvs. kryptering) holdes samlet for det andre nøkkelparet. Dette forutsetter at en (i policy eller praksis) slår fast at nøkkeldeponi eller sikkerhetskopiering av private nøkler for dekryptering aldri skal brukes -- kopier av private nøkler som også brukes for autentisering eller digital signatur, er ikke akseptabelt.

Merk at dette vil ekskludere Entrust-ready produkter, som har digital signatur og autentisering på samme nøkkelpar / sertifikat, og nøkkelutveksling / kryptering separat. Dersom en velger to nøkkelpar og sertifikater, og Entrusts spesifikasjoner, får en etter vårt skjønn svakere sikkerhet for de digitale signaturene: Bruken som autentiseringsnøkkel tilsier at brukeren ikke har like direkte og eksplisitt kontroll med bruken av nøkkelen som det er rimelig å kreve for en signeringsnøkkel.

11.1.4 Navngivning

Det er et krav i FEIDE at elektronisk ID skal kunne brukes mot FAS og andre tjenester, altså må navnene i sertifikatene egne seg for å bli tolket av applikasjoner / tjenester. Samtidig er det et sterkt ønske om at sertifikatene skal egne seg til kommunikasjon mellom mennesker, f. eks. sikker epost, og da bør også navnene ha elementer som egner seg for å presenteres for mennesker. Konklusjoner på dette er:

- Navnene skal ha en unik identifikator. Det har ikke festet seg noen standard for hvordan dette skal kodes i sertifikater, men «serialNumber» attributten i DN ser ut til å være mest aktuell. Diskusjoner i FEIDE prosjektgruppa har konkludert med at vi ønsker en identifikator som er unik innen hele UoH-sektoren, altså ikke bare innen hvert lærested. Identifikator kan ikke være fødselsnummer siden dette strider mot prinsippet om at sertifikater bare kan holde åpen informasjon. Det anbefales heller å vedlikeholde en tabell som gir en en-til-en mapping mellom unik identifikator og fødselsnummer. Tabellen (eller deler av den) må bare være tilgjengelig for autoriserte brukere og applikasjoner.
- Navnene skal primært inneholde brukerens vanlig brukte navn (fornavn og etternavn mm.) kodet i CN-attributten i DN. Her må en åpne for pseudonymer (kanskje også «anonyme» sertifikater) for brukere som ikke kan offentliggjøre navnet sitt -- f. eks. asylanter. Det er ikke noe prinsipielt i veien for å tillate bruk av pseudonymer, kallenavn o.l. også for andre brukere. Dette er noe det må tas en policy-avgjørelse på på ett eller annet tidspunkt.

Neste navnespørsmål er om navnene skal knytte brukeren opp mot et spesifikt lærested. Dette kan gjøres på to måter: Enten ved en O-attributt i brukerens DN (O=lærestedets navn) eller ved at navn på utsteder av sertifikatet settes til navn på lærested (i en eller annen form -- DN for utsteder må også spesifiseres). Det første alternativet tilsvarer det som er valgt i Forvaltningsnettsamarbeidet. Diskusjoner i FEIDE arbeidsgruppa har konkludert med at:

- Navn knyttes ikke opp mot lærested.

Dette er ikke noen «hard konklusjon» og den kan nok revurderes avhengig av hva en ender opp med av valg for sertifikattjenesten.

Et nytt spørsmål er om sertifikatene skal inneholde noe informasjon om roller eller funksjoner for brukerne. Her er en klar anbefaling at dette unngås unntatt der det er snakk om informasjon med svært lang levetid. En kan m.a.o. vurdere dette i hvert fall for visse grupper ansatte. Men sertifikater i FEIDE skal kun gi autentisering. Det er ikke meningen å bruke dem til å bevise rettigheter av noe slag.

Skal sertifikatene «merkes» (ala BankID varemerket) på en spesiell måte for å vise at de tilhører FEIDE? I utgangspunktet svarer prosjektgruppa «ja» på dette, men dette må jo fravikes dersom en velger en eksisterende tjeneste / spesifikasjon. Merkingen kan gjøres som for BankID, med en O-attributt (O=FEIDE ID). Dette forutsetter at en ikke trenger O-attributten til navn på lærested. En kan vurdere om det er andre navneattributter som egner seg bedre for formålet. En kan også merke ved å sette DN for utsteder til «FEIDE», og prosjektgruppa vil anbefale dette alternativet. Merk at dette (antagelig) impliserer en monolittisk tillitsmodell med én utsteder for hele UoH-sektoren.

11.1.5 Kataloger, sertifikatdistribusjon

Vi ser her kun på behovet for kataloger for sertifikater og CRLer. Dette kan selvfølgelig kombineres med generelle kataloger med mer informasjon. Vi forutsetter at alle kataloger nås ved LDAP, fortrinnsvis LDAPv3, og vi ønsker ikke å stille noen krav som binder en til å velge X.500-systemer.

Med dagens status for distribuerte katalogsystemer vil vi anbefale å opprette en hovedkatalog der alle utstedte sertifikater samles. Dette er «originalen», og oppdateringer (innlegging og fjerning av sertifikater) gjøres her. Av sikkerhetsmessige grunner bør en antagelig ikke la det være noen oppslag eller tilgang for vanlige brukere mot denne katalogen, kun administrativ aksess for oppdatering og kopiering til andre kataloger. Denne katalogen kan godt drives av den valgte sertifikatleverandøren -- i hvert fall dersom den bare skal inneholde sertifikater.

De felles administrative systemene, og eventuelt andre systemer, kan trenge adgang til alle sertifikater, og det trengs derfor en fullstendig FAS-katalog for dette formålet. Dette kan være en full kopi av

hovedkatalogen eller, dersom leverandøren har hovedkatalogen for en mer generell tjeneste, de delene som er relevante for UoH-sektoren. Kanskje er det beste å ha en hovedkatalog hos leverandøren med bare sertifikater, med en FAS-katalog som laster ned alle relevante sertifikater og kombinerer med annen informasjon? FAS-katalogen må ha begrenset aksess.

Fra FAS-katalogen må UoH-enheter kunne laste ned hele eller relevante deler av FAS-katalogen til interne kataloger. Igjen må disse ha begrenset aksess, f. eks. til interne administrative systemer.

Fra FAS-katalogen -- eller rett fra hovedkatalogen -- må en også kunne trekke ut informasjon til en helt åpen katalogtjeneste på Internett. Her er en avhengig av en siling av informasjonen. Det må være mulig å reservere seg mot at informasjon havner i en åpen katalog -- det er til og med mulig at en må ha aktivt samtykke fra alle brukere for å få lov til dette. Dette må avklares med Datatilsynet, og registreringsprosedyrene og avtalene med brukerne må tilpasses. (Siden brukerne i utgangspunktet må underskrive en avtale om bruk av sertifikattjenesten for å få utstedt sertifikat, kan dette samtykket innhentes samtidig.)

Tabellen som oversetter mellom unik identifikator og fødselsnummer, skal være tilgjengelig for de anvendelser og personer som har behov for det. Dersom en laster ned bare deler av FAS-katalogen, bør en bare få med de delene av tabellen som er relevante. Tabellen skal selvfølgelig aldri være åpent tilgjengelig, spesielt ikke på Internett.

CRLer skal alltid være åpent tilgjengelige, også på Internett.

11.1.6 Tillitsmodell

Tillitsmodell må tilpasses til reglene for navngivning, men i utgangspunktet er det to aktuelle modeller:

- En tjeneste for hele UoH-sektoren (monolittisk tillitsmodell),
- En (logisk) tjeneste for hver UoH-enhet under en felles rot (grunt hierarki).

Imidlertid bør FEIDE kunne være ganske fleksibel her. Dersom en velger en «ferdig» tjeneste, importerer en også tillitsmodellen fra denne tjenesten, f. eks.:

- BankIDs system der utsteder vil være brukerens bank, uavhengig av f. eks. studiested.
- Forvaltningsnettsamarbeidet, der en har tre likestilte leverandører og kryssertifisering.

Dette burde ikke være særlig problematisk.

11.2 Valg av sertifikattjeneste

11.2.1 Oversikt, alternativer

FEIDE har fem alternativer for tjeneste for elektronisk ID:

- Postens Elektroniske ID, som er en tilgjengelig tjeneste,
- BankID, som er en framtidig tjeneste, spesielt når vi snakker om smartkort,
- Tjeneste fra Telenor, som antagelig betyr at en binder seg til Entrust-ready produkter (merk at vi ikke har undersøkt konkret hvilke tjenester Telenor Bedrift tilbyr -- dette må gjøres før en bestemmer seg endelig),
- Handling over Forvaltningsnettsamarbeidets rammeavtaler,
- Egne spesifikasjoner, og avtale basert på tilbud fra en eller flere leverandører -- her bør en i tilfelle ta utgangspunkt i (spesifikasjonen for) en av de tilgjengelige tjenestene og legge seg så tett opp til denne som mulig.

Utgangspunktet for FEIDE var det siste alternativet, altså en egen tjeneste, levert fra en anerkjent leverandør, med en egen ID for UoH-sektoren. Diskusjoner under møter underveis i prosjektet har klarlagt

at det ikke er nødvendig at det er en separat ID. Bruk av f. eks. Postens Elektroniske ID eller BankID er helt greit dersom tjenestene fyller de krav FEIDE må stille.

11.2.2 Postens Elektroniske ID

Denne tjenesten har som sin største fordel at den faktisk eksisterer. Vi antar i utgangspunktet at den kan brukes for studenter og ansatte i UoH-sektoren, men vi har ikke sjekket policy eller vært i kontakt med leverandøren. Imidlertid er intensjonene at Postens Elektroniske ID skal kunne utstedes til alle personer i Norge. Postens Elektroniske ID har en navngivning som passer god med FEIDEs behov, inkludert en unik identifikator. FEIDE må få tilgang til relevante deler av tabellen som oversetter mellom unik identifikator og fødselsnummer. Posten er selv utsteder for sertifikatene. Tillitsmodellen er monolittisk, etter hvert med hierarki for internasjonale postverk.

Postens Elektroniske ID leveres i dag på Multos smartkort, og flere applikasjoner kan da legges på kortene. Posten SDS har demonstrert kombinasjonen elektronisk ID og Mondex betaling. Problemet er at Mondex ser ut til å være en blindgate for betalingssystemer i Norge. Det er ikke sannsynlig at Posten SDS vil ønske å kombinere elektronisk ID med Proton, som er valgt av bankene, men vi har ikke undersøkt dette. (Den tekniske muligheten er der: Proton-applikasjonen finnes på Multos plattform.)

Posten SDS har basert seg på programvare fra svenske ID2, og dette gir potensielt et begrenset utvalg av brukerprogramvare. Etter det vi kjenner til, leverer ID2 kryptografiske tjenestemoduler både for PKCS#11 (dvs. Netscape m.fl.) og Microsoft CryptoAPI (Internet Explorer, Microsoft Outlook og annet på Windows-plattform).

11.2.3 BankID

BankID er prosjekt under Bankenes Standardiseringskontor, støttet av både Sparebankforeningen og Finansnæringens Hovedorganisasjon, og dermed av så godt som alle banker som opererer i Norge. Det er utviklet et sett av spesifikasjoner, inkludert utkast til avtale mellom bankene, kundeavtale og sertifikatpolicy. Det mangler formelle vedtak fra bankene vedrørende lansering av tjenesten som et varemerke, men antagelig er dette bare en formalitet. Alternativet er at bankene velger individuelle løsninger, og det er klart dårligere sett med kundenes øyne, og også for i hvert fall de fleste bankene en dårligere løsning. I første omgang er BankID ikke basert på smartkort, men dette vil komme etter hvert. Grunnen er rett og slett at en ikke kan forvente at bankkundene skal skaffe seg smartkortlesere. Spesifikasjonene er laget med tanke på framtidig introduksjon av smartkort.

BankID vil i utgangspunktet egne seg godt for FEIDE. Navngivningen passer, og inkluderer også en unik identifikator. Kreditkassens person i prosjektet mente at det ikke burde være noe problem å få kopiere de deler av oversettelsestabellen til fødselnummer som FEIDE trenger.

Med BankID vil en kunne videreføre det samarbeidet SiO har med Kreditkassen om banktjenester knyttet til studiekortet, og en vil kunne utvide dette til å gjelde alle banker som støtter BankID (en forutsetning -- en kan ikke binde alle personer i UoH-sektoren til en bank). Kreditkassen og BBS mener at BankID på smartkort, med Proton betaling inkludert, kan være realistisk til høstsemesteret 2001. Det ligger ikke noen forpliktelse i denne uttalelsen, men potensialet i å utstede slike kort til etter hvert alle personer innen UoH-sektoren vil nok føre til at utvikling av en slik kortløsning vil være meget interessant.

Det er imidlertid også noen problemer knyttet til bruk av BankID:

- Usikkerhet om når tjenesten kommer og når den kan være klar på smartkort.
- Nødvendige policyavklaringer må gjøres i forhold til at bankene må akseptere denne bruken av BankID, og eventuelt hvilke forpliktelser bankene tar på seg i en slik sammenheng.

- BankID kan slik policy er i dag bare utstedes til kunder (dvs. disposisjonsrett til en konto er nok) av en BankID-bank. (Merk at Proton betalingssystem også forutsetter at brukerne er kunder av en «godkjent» bank.) Det vil finnes personer som ikke har et slikt kundeforhold, f. eks. enkelte utlendinger. Dette kan løses ved å opprette en «null-bank» som utsteder sertifikater for de som ikke er bankkunder, eller at bankene akseptere å utstede BankIDer for andre en kunder -- disse vil da ikke gi noen rettigheter mot banksystemene, men vil være en generell elektronisk ID.
- Det kan tenkes at flere personer vil reservere seg mot å måtte ha et bankkort på studiekortet, i forhold til valg av en mer «nøytral» ID.
- Det må avklares at det er OK å ha flere BankIDer for samme person utstedt av samme bank. De personene som ikke ønsker å ha studiekortet som eneste bankkort, må ha en BankID for hvert kort, i det alle smartkort har garantert forskjellige nøkler.

BankID er likevel et lovende alternativ for FEIDE. Baltimore-plattformen til BBS (dersom BBS blir valgt som leverandør) sikrer tilgang til et godt utvalg av brukerprogramvare.

11.2.4 Telenor Zebrasign

Vi antar uten videre at Telenor vil være i stand til å levere en tjeneste som passer for FEIDE, men vi har ikke undersøkt om de i dag tilbyr en tjeneste som kan brukes uten videre. Dette må undersøkes i starten av en videreføring -- i motsatt fall vil en avtale med Telenor måtte være basert på et eget spesifikasjonsarbeide (se nedenfor).

Det største potensielle problemet med Telenor er en svært sterk binding mot Entrust. Merk at det må undersøkes hvor sterk denne bindingen egentlig er. Entrust-spesifikasjonene gir adgang til en mengde programvare på brukersiden, og dette er en klar fordel i forhold til andre alternativer, der mengden brukerprogramvare kan være nokså begrenset.

Vi vil ikke anbefale FEIDE å binde seg til Entrust-spesifikasjoner, men vi vil sterkt anbefale at en undersøker med Telenor hvor sterk bindingen faktisk er før en avviser dette alternativet.

11.2.5 Forvaltningsnettsamarbeidets rammeavtaler

Disse avtalene gir adgang til å kjøpe det meste som trengs for elektronisk ID for FEIDE gjennom ferdigforhandlede avtaler, der en stort sett bare diskuterer pris. Her er en garantert at tjenester og produkter faktisk er tilgjengelige. Merk at det er fullt mulig å handle bare deler av det en trenger over avtalene, f. eks. kan en handle smartkort og lesere til gunstige priser, mens en velger å handle sertifikattjenestene på annen måte. Vi har ikke gjort noen evaluering av hvordan spesifikasjonene passer med FEIDEs behov, men noen avklaringer er nødvendig:

Navngivningen, som er nedfelt i policy, krever «arbeidssted» (tolkning til å omfatte studiested burde være uproblematisk) som en del av navnet. Navnet inneholder ikke noen unik identifikator. Disse valgene var svært bevisste fra Forvaltningsnettsamarbeidet da policy ble skrevet, og passer med de fleste situasjoner innen offentlig sektor. Men dersom FEIDE ikke kan bruke denne navngivningen, kan det tenkes at en må definere en ny, litt endret, policy, og dette kan være tvilsomt i forhold til om avtalene kan brukes.

Det må undersøkes om FEIDE / UNINETT kan gjøre ett avrop på rammeavtalene for hele UoH-sektoren, eller om hvert enkelt lærested må handle separat. I tilfelle kan forskjellige læresteder ende opp med å bruke forskjellige leverandører, uten at dette nødvendigvis er noe problem.

Enkelte deler av FEIDEs behov, som fysisk trykking / preging av kortene, er ikke dekket av avtalene, men leverandørene kan levere dette gjennom tilleggsavtaler.

Vi anbefaler at det gjøres en evaluering av hvordan rammeavtalene passer med FEIDEs behov, og at en deretter konkluderer med om dette er en farbar vei. Det store spørsmålet, etter vårt skjønn, er om kan få lov til å endre navngivningen eller tilpasse denne slik at FEIDEs behov blir fyllt.

11.2.6 Egne spesifikasjoner

Etter at en har gjort en evaluering av brukbarheten av de ferdige (eller «ferdig i nær framtid») alternativene, kan det hende at konklusjonen er at ikke noe av dette passer godt nok. Da ender en opp med å måtte spesifisere sertifikattjenesten selv, noe som kan være en stor jobb.

Imidlertid anbefaler vi i tilfelle å legge seg tettest mulig opp mot ett av de alternativene som er nevnt over. Det mest nærliggende er i tilfelle å «kopiere» BankID. Med det kan en oppnå at det er mulig å få til en gjensidig godkjennelse mellom de to IDene, slik at en etter hvert kan bruke FEIDE IDen mot bankene og BankID mot lærestedene. Dette forutsetter at tjenestene sikkerhetsmessig (dvs. policy) er på samme nivå, og at det opprettes en form for kryssertifiseringsavtale. I et slikt tilfelle vil en kunne opprettholde systemet med banktjenester tilgjengelig på studiekortene.

Merk at et teoretisk alternativ, særlig for multifunksjon-smartkort, er å legge to IDer på samme kort, f. eks. både FEIDEs ID og BankID. Dette kan vi ikke anbefale fordi kapasiteten på dagens kort kan være for liten, og ikke minst fordi dette ikke har vært prøvd i praksis. Det er sannsynlig at svært få applikasjoner vil kunne håndtere et slikt kort og være i stand til å velge mellom to like funksjoner på samme kort. (Dette kan gjøres på to ulike måter, enten ved å legge flere sett med nøkkelpar i samme smartkort-applikasjon, dvs. «katalog» på smartkortet, eller ved å legge inn separate og uavhengige smartkort-applikasjoner. De tekniske problemstillingene er forskjellige, men de praktiske problemene og konklusjonen er de samme.)

11.3 Programvare- og systemtekniske løsninger

11.3.1 Klientside

Vanlige web-klienter, dvs. Netscape Communicator, Microsoft Internet Explorer m.fl., antas å dekke en stor del av behovet for klientside programvare. Sikkerhetstjenestene for web-baserte anvendelser baseres på bruk av SSL, som uten videre støttes av disse web-klientene. Epost-sikkerhet ved S/MIME dekkes av de tilhørende e-post-klientene. Tilstrekkelig sikkerhet krever bruk av de variantene av pakkene som tilbyr sterk kryptografi, som først nylig har blitt tilgjengelig utenfor USA.

Sertifikattjenesten bør ikke legge noen føringer på valget av slik standard-programvare. Det vil si at sertifikatene utformes slik at de kan benyttes sammen med både Communicator, Internet Explorer og evt. andre aktuelle klienter. Kryptomoduler for både PKCS#11 og CryptoAPI bør være tilgjengelige for de aktuelle kortene. Det anbefales også at PKCS#11-moduler er basert på PC/SC på Windows-plattform.

Basiskonfigurasjonen for en vanlig (PC-basert) klient er med andre ord:

- «Standard» web-klient, variant med sterk krypto.
- Smartkort med elektronisk ID.
- Kortleser med drivere (PC/SC) for aktuelt operativsystem.
- Kryptomodul for smartkort og -applikasjon, PKCS#11 eller CryptoAPI CSP avhengig av web-klient.

ID-kortene utstedes alltid med elektronisk ID (med mindre brukeren reserverer seg). De andre komponentene kan distribueres f.eks. ved å tilby en pakke som inneholder kortleser samt en CD-ROM med drivere og kryptomoduler. Web-klient antas det at brukerne allerede har eller kan skaffe, men for å sikre tilgang til sterk krypto-varianten kan den også distribueres på samme CD-ROM. Ved installasjon

legges samtidig SA-sertifikatene for FEIDEs SA inn, og klientprogramvaren kan konfigureres til å bruke FEIDEs LDAP-tjenere. (Samme CD-ROM bør kunne dekke varianter for ulike preferanser av web-klient og forskjellige operativsystem/maskinvare-plattformer.)

Vi anbefaler at nøkler og sertifikater utstedt for FEIDE ikke uten videre brukes sammen med programvareløsningene for kryptografi i web-klientene. Programvareløsninger for nøkkellagring og kryptografi gir vesentlig svakere beskyttelse av nøklene. I tilfelle bør sertifikatene utstedes av en logisk separat SA med en mindre strikt sertifikatpolicy, eller sertifikatene på annen måte merkes slik at de kan spesialbehandles i sertifikatpolicy. Nøkler som er tillatt brukt med programvare nøkkellagring og kryptografi vil ikke være akseptable for mer sensitive formål. Denne komplikasjonen (to parallelle sett med nøkler og sertifikater) er neppe verd å ta hvis eneste formål er å slippe installasjon av smartkort-infrastrukturen.

Det finnes flere forskjellige kategorier av klientmaskiner:

1. Personlige (hjemme) maskiner, dvs. drift og utstyrsanskaffelser gjøres av eieren selv. Eier kan være både student og ansatt. Det er ikke mulig å gjøre noen spesielle forutsetninger om hvor godt beskyttet maskinen er.
2. Lab-maskiner med generell adgang, men drevet av lærestedet selv og med en viss beskyttelse (brannvegg, antivirus m.m.)
3. Personlige arbeidsplassmaskiner med et forholdsvis liberalt driftsopplegg. (F.eks., brukeren selv tillates programvareinstallasjon, noenlunde fri adgang til Internettet.)
4. Arbeidsplassmaskiner med kontrollert driftsopplegg. (F.eks., kun bestemte programpakker installert, liten adgang for brukeren selv til å rekonfigurere.)
5. Spesielt tiltrodde maskiner for bestemte formål, f.eks. kvalifiserte signaturer.

Både med hensyn på utstyrs- og programvarekonfigurasjoner og i sertifikatpolicy må den siste kategorien spesialbehandles. Ellers ser vi ikke noen grunn til å legge opp til vesensforskjellige konfigurasjoner i de forskjellige kategoriene. De to første kategoriene maskiner er nettopp der hvor det er mest problematisk å kreve smartkort-infrastruktur, men samtidig hvor programvare-kryptografi er mest uaktuelt av sikkerhetsmessige hensyn.

All annen klientside-programvare enn dette installeres bare etter behov. (Dette kan være: OCF, VPN-klienter, sikre ORB-klienter, m.m.) Vi har foreløpig ikke identifisert noen spesielle behov.

11.3.2 Tjenerside

Aksesskontroll-mekanismene i forskjellige produkter er svært forskjellige. Utgangspunktet for aksesskontroll med FEIDE er sikker autentisering, hvor brukeren er kjent gjennom et FEIDE DN. Derfra er det flere strategier for å implementere aksesskontroll:

- DN avbildes til bruker-ID («principals») eller gruppe-ID i plattformen som tjenerapplikasjonen kjører på (operativsystem, database, ORB, e.l.), og aksesskontrollen håndheves av plattformen.
- DN avbildes til et sett med privilegier (kapabiliteter) i forhold til applikasjonsplattformen, og aksesskontroll håndheves av plattformen, som ovenfor.
- Applikasjonsprogramvaren selv (f.eks. FAS) tolker selv DN og håndhever aksesskontroll.

Hva som egner seg avhenger av applikasjonsplattform og implementasjonsverktøy. Noen generelle anbefalinger kan vi likevel gi:

- Valget av aksesskontroll- og autorisasjonsmekanismer er lokale innenfor en organisasjon. Det samme gjelder konvensjoner og policies for hvordan bruker-ID, privilegier etc. kobles til FEIDE-DN. Ved kommunikasjon på tvers av organisasjoner brukes i utgangspunktet ikke plattform-spesifikke autentiserings- eller aksesskontroll-mekanismer.

- Avbildning av DN til bruker-ID, privilegier, autorisasjonsstrenger etc. skjer i størst mulig grad ved oppslag i LDAP-katalog. (Dette må ligge i en *lokal* LDAP-katalog, jfr. forrige punkt.)

Med andre ord, FEIDE etablerer felles konvensjoner og mekanismer for *autentisering*; mens *autorisasjon* og *aksesskontroll* håndteres lokalt for hvert lærested (evt. avdeling/system/tjeneste). Dels tror vi lærestedene stort sett vil ønske å beholde kontrollen over autorisasjon mot egne systemer. Dessuten vil format og innhold av autorisasjonsdatabaser være så produktspesifikt og avhengig av lokale oppsett at det sannsynligvis vil være vanskelig å etablere noen brukbare felles konvensjoner.

Aksesskontrollmekanismer i plattformen bør utnyttes så godt det er mulig, men har klare begrensninger i sammenheng med FEIDE: Det er selvfølgelig ikke mulig å ha separate brukere på operativsystem- eller databasenivå for hver enkelt FEIDE-ID som noen gang har blitt utstedt. En måte å gjøre det på, er å kategorisere alle ID-er (ved en kombinasjon av informasjon i sentral og lokal katalog), f.eks.:

- Lokale studenter, med underkategorier.
- Lokale ansatte, med underkategorier.
- Andre studenter.
- Andre ansatte.

For en gitt applikasjon, lab, lokalt nett osv. vil ID-er i noen få av kategoriene (typisk: lokale studenter og ansatte) avbildes en-til-en til bruker-ID eller lignende. For hver av de andre kategoriene avbildes alle ID-er til en felles bruker-ID, og hvis noe mer finkornet adgangskontroll skal håndheves, blir det opp til applikasjonsprogram.

Et typisk eksempel er et studentadministrativt system. Her kan adminstrativt personale ha personlige bruker-ID i databasen som tildeles aksessrettigheter etter behov. Studenter har også adgang til systemet, men gjennom en felles bruker-ID som bare gir adgang til persondata gjennom visse transaksjonsprogrammer (som mottar FEIDE-ID som en parameter og utelukkende gir adgang til data som gjelder denne personen.)

11.3.3 VPN-løsninger

PKIen som bygges opp for FEIDE vil sannsynligvis ikke være velegnet som PKI for VPN-løsninger, først og fremst fordi FEIDE identifiserer personer, ikke maskiner. Vi anbefaler altså at FEIDE og en eventuell PKI for VPN (spesielt IPSec) behandles hver for seg.

Det ville imidlertid være naturlig å benytte FEIDE som autentiseringsmekanisme i aksess-VPN, slik at brukere med FEIDE-ID kan få privilegier ut over det vilkårlige Internett-brukere har:

- Hvert lærested opererer sine egne, lokale, aksesstjenere (NAS eller LNS)
- Brukere (klienter) autentiserer seg mot aksesstjeneren med FEIDE-ID i stedet for lokal bruker-ID.
- Aksesstjeneren gir aksesskontroll mot interne nett og tjenester basert på en kombinasjon av informasjon fra lokal RADIUS-tjener og FEIDE-katalogen (LDAP), f.eks. alle FEIDE-ID som ikke er eksplisitt registrert med en tilsvarende lokal ID i RADIUS-databasen avbildes til en bestemt «gjestebruker».

Skal denne løsningen kunne anbefales, må det imidlertid finnes interoperable produkter som tilbyr offentlig nøkkel-autentisering basert på FEIDE-sertifikatet. Det tviler vi på, siden det ikke finnes standarder som dekker dette. Den mest kurante løsningen i dag er antakelig PPP + CHAP og RADIUS. Dette ville kreve samvirkende RADIUS-tjenere på tvers av lærestedene, med en ganske stor grad av lokale konvensjoner, som antakelig ville tøye interoperabiliteten mellom forskjellige leverandørers produkter til det ytterste. Dessuten ville det kreve en egen, sikker infrastruktur for håndtering av CHAP-passord, og bortsett fra en viss koordinering av bruker-ID ville denne løsningen være nokså uavhengig av FEIDE. Det er selvfølgelig

fritt frem for hvert av lærestedene å etablere egne aksess-VPN, men da uten å synkronisere passord eller nøkler, slik at brukerne må registrere seg overfor hvert enkelt av lærestedene der de skal ha tilgang.

11.3.4 Smartkort

Det er antakelig for strikt å standardisere på en bestemt type smartkort og/eller detaljspesifisere FEIDEs elektronisk ID-applikasjon slik at det blir full interoperabilitet på kommandonivå mellom kort og leser. En må regne med at det oppstår behovet for å ta inn kort av et nytt fabrikat eller godta en ny sertifikattjeneste (som legger begrensninger på kortene), og dette må uansett håndteres ved å tilby nye kryptografiske tjenestemoduler (cryptographic service providers). FEIDE må imidlertid definere en profil på API-et til tjenestemodulen, dvs. hvilke nøkler, sertifikater og andre data som skal finnes, hvordan disse skal lokaliseres (navn, aksessveier) osv., og dette legger føringer på hvordan innholdet i kortet er organisert.

Det er også behov for å etablere noen overordnede konvensjoner for hvordan kortene skal identifiseres av terminalen, spesielt når det gjelder ATR-strengen og innholdet i DIR-filen. Noen prinsipper vi tror er fornuftige:

- FEIDE vedlikeholder sentralt en overordnet fortegnelse over de aktuelle kort-typene (dvs. kombinasjon av fabrikat/operativsystem for kort, type og versjon av ID-applikasjon, sertifikatautoritet), med en tjeneste for å laste ned tjenestemoduler for disse kortene. Vi antar at det maksimalt dreier seg om 3-5 ulike typer. (Når det er behov for å innpasse nye versjoner, kan dette antakelig ofte gjøres ved å tilby en oppgradert tjenestemodul som håndterer både ny og gammel versjon, slik at det ikke blir nødvendig å ha et stort antall tjenestemoduler installert overalt.)
- FEIDE er den primære applikasjonen på kortet. Kortet skal normalt kunne gjenkjennes som en bestemt av de godkjente typene av FEIDE-kort bare på grunnlag av ATR-strengen. Anvendelser som legger spesielle føringer på ATR (i konflikt med FEIDE) for å fungere må vike. FEIDE bør ellers ikke forutsette noe om innholdet i ATR.
- Hver enkelt kort-type i FEIDE bør ha en distinkt ISO 7816-5 applikasjons-ID, med mindre applikasjonene er så like at de kan håndteres av samme tjenestemodul.
- Hvis det skal være mulig å gjenkjenne et kort som et FEIDE-kort (uten nødvendigvis å ha rett tjenestemodul installert), bør dette skje på grunnlag av DIR-filen.

SEIS, PKCS#15, EMV, MULTOS m.fl. legger føringer på disse konvensjonene, utover det som ISO 7816 definerer, så det er nødvendig å ta hensyn til disse (og evt. andre beslektede spesifikasjoner).

Multiapplikasjonskort (Multos, JavaCard eller tilsvarende) er ønskelig, men dette gir en del ekstra administrasjon rundt utvikling, sertifisering/signering av applikasjoner, utstedelse og initialisering av kort som FEIDE eller lærestedene antakelig ikke skal ta på seg. Vi har ikke sett noe behov for FEIDE-spesifikke applikasjoner eller egenutvikling i FEIDE-regi, og det er trolig bedre å basere seg på etablerte kortutstederes infrastruktur for disse formålene.

FEIDE skal i utgangspunktet bare dekke elektronisk ID, og å unngå å forsinke eller komplisere prosjektet bør det ikke stille absolutte krav om multiapplikasjonskort, eller om integrasjon av andre smartkortapplikasjoner. Ved å legge opp til en infrastruktur som teknisk sett er i stand til å håndtere kort av flere typer, og desentralisere valg av kortleverandør og utstedelsesprosedyrer til de enkelte lærestedene, kan multiapplikasjonskort anvendes der det er behov for det uten at det påvirker FEIDE selv på noen vesentlig måte.

Vedlegg

A. Sikkerhetskrav og sikkerhetstjenester

Elektronisk ID benyttes til å realisere en rekke ulike sikkerhetstjenester: *Autentisering*, *ikke-benekting* og *konfidensialitet*, anvendt både som *meldingssikkerhet* (på applikasjonsnivå) og som *kommunikasjonssikkerhet* (på transport- eller nettverksnivå). Forskjellige bruksområder stiller forskjellige funksjonelle og sikkerhetsmessige krav. Dessuten er sikkerhetstjenestene en funksjon av hele systemet, der selve den elektroniske ID-applikasjonen, PKI, smartkort og kryptografi, bare er en del av helheten. Som bakgrunn for diskusjonen av plattformer og tekniske løsninger vil vi derfor gå nærmere inn på hva disse sikkerhetstjenestene innebærer, og hvilke krav de stiller.

A.1 Sikkerhetstjenester

A.1.1 Meldingssikkerhet

De tekniske mekanismene for meldingssikkerhet er:

- Kryptering av innhold.
- Digital signatur.

En digital signatur, slik begrepet vanligvis forstås, realiserer disse to sikkerhetstjenestene:

Autentisering av opphav:

> At innholdet av dokumentet virkelig stammer fra den som angivelig har signert det, dvs. beskyttelse mot at en utenforstående bevisst har forfalsket det.

Ikke-benekting av opphav:

> At den som har signert dokumentet ikke senere kan benekte det, dvs. beskyttelse mot forfalskning (bedrageri) fra denne parten selv.

Ikke-benekting kan oppfattes som en sterkere form for autentisitet. Legg merke til at for tilfellet autentisering er det ikke noen fundamental mistillit mellom partene. Krav om ikke-benekting er alltid begrunnet i en viss mistillit mellom partene selv.

I praksis er grensen mellom disse to tjenestene temmelig flytende. I virkelige tilfeller vil graden av ikke-benektbarhet avhenge av hvilken motivasjon underskriveren har for svik, og hvilke sanksjonsmuligheter en tredjepart (f.eks. via domstolene) kan gjøre gjeldende. Også når det gjelder autentisering er det selvfølgelig i siste instans snakk om å stille den som signerer til ansvar. Det er altså noen tekniske forutsetninger som må være tilstede for å gi ikke-benekting, men **styrken** vil avhenge av andre forhold (lovverk, avtaleforhold mellom partene o.l.)

For at en digital signatur skal kunne erstatte en håndskrevet signatur, vil lovverket stille krav om en såkalt *kvalifisert* digital signatur (egentlig: at signaturen kan bekreftes ved et *kvalifisert sertifikat*). Dette betyr i realiteten at behandlingen av nøkler, signaturgenereringen, sertifikattjenesten og sertifikatutsteders ansvar tilfredsstiller visse krav. En kvalifisert signatur gir alltid sterk ikke-benekting, siden den pr. definisjon skal være gyldig når lover og forskrifter krever signatur. Omvendt, kvalifisert signatur er **ikke** en forutsetning for å oppnå en viss grad av ikke-benektbarhet, bare at uten kvalifisert signatur avhenger ikke-benektbarheten av sertifiseringpolicy og avtaler mellom aktørene.

Forskrifter for kvalifisert signatur er ikke på plass ennå, og det gjenstår å se hvor sterke krav til utstyr og programvare som i praksis blir stilt, og i neste omgang, hvordan domstolene vil stille seg i saker der noen bestrider en kvalifisert signatur. Det er imidlertid grunn til å tro at kravene blir svært strenge, og f.eks. en vanlig PC med «hyllevare»-type programvare bør antakelig ikke være en akseptabel plattform. Med en kvalifisert signatur vil man kunne «signere fra seg gård og grunn», og det sier seg selv at det da stilles høye sikkerhetskrav.

Samtidig vil vi bemerke at det trolig er mange situasjoner der man tradisjonelt har krevd en håndskrevet signatur, men hvor det reelle behovet egentlig bare er for autentisering, eller at en elektronisk versjon av transaksjonsgangen kan utformes slik at krav om ikke-benekting kan nedgraderes til autentisering. En skal altså ikke se seg blind på kvalifisert digital signatur og stille urimelig sterke krav til en løsning.

Kryptering av innhold gir sikkerhetstjenesten:

Konfidensialitet:

> At innholdet i dokumentet ikke har blitt/kan bli kjent av uvedkommende.

Konfidensialitet og autentisering stiller tilsvarende tekniske sikkerhetskrav til utstyr og programvare, og derfor behandler vi disse sikkerhetstjenestene under ett nedenfor. (I og for seg kan konfidensialitet og autentisititet oppfattes som duale egenskaper: Konfidensialitet gir sikkerhet for at bare rette vedkommende mottar innholdet, mens autentisitet gir sikkerhet for at bare rette vedkommende kan være opphavet til innholdet.) Felles for disse to tjenestene, og til forskjell fra ikke-benekting, er at avsender og mottaker har tillit til hverandre.

Meldingssikkerhet er i alminnelighet en applikasjonslagstjeneste, og partene (avsender og mottaker) er ofte virkelige personer.

Meldingssikkerhetstjenester har en effekt som strekker seg ut i tid, i prinsippet ubegrenset, i praksis så lenge sertifikater er gyldige eller en kan anta at nøkler er hemmelige.

A.1.2 Transport- og nettverkssikkerhet

Sikkerhetstjenestene konfidensialitet og autentisering kan også anvendes på lavere lag i protokollhierarkiet, typisk på (dvs. rett over) transportlaget, nettverkslaget eller linklaget. Sammenlignet med meldingssikkerhet er det noen karakteristiske forskjeller:

- Sikkerhetstjenestene beskytter en kommunikasjonskanal, altså trafikk mellom to endepunkter/aksesspunkter. Partene i forhold til sikkerhetstjenestene er de som har kontroll over endepunktene, ikke nødvendigvis de som kommunikasjonen skjer på vegne av.
- Sikkerhetstjenestene er efemeriske (flyktige, midlertidige), dvs. de beskytter innholdet bare i det utvekslingen foregår.

Autentisering generelt, men spesielt på transportlagsnivå, brukes som basis for *aksesskontroll* (tilgangskontroll).

Virtuelle private nett (VPN) realiseres gjerne ved sikkerhetstjenester på nettverkslaget (eller linklaget).

A.2 Forutsetninger for autentisitet og konfidensialitet

A.2.1 Grunnleggende sikkerhetskrav

Den mest grunnleggende forutsetningen, både for autentisitet og konfidensialitet, er full kontroll over det utstyret og den programvaren som på noen måte kan påvirke sikkerhetskritiske

funksjoner. (Kryptering og dekryptering, generering og verifikasjon av digital signatur.) Det er stor oppmerksomhet på styrken av kryptografiske algoritmer og muligheten for forfalskning av nøkler. Dette er nødvendige ledd i kjeden (se nedenfor). Men andre ledd er like viktige og, dessverre, lettere å overse.

- Den som utsteder en autentisert melding (eller sender autentiserte data) forholder seg aldri like direkte til den som til et papirdokument. Meldingen produseres og presenteres av en eller annen applikasjon, selv i det tilfellet hvor applikasjonen gir illusjonen av direkte redigering (WYSIWYG). Utstederen er avhengig av at programvaren presenterer for signering noe som er en korrekt representasjon av utstederens intensjoner og ikke kan tolkes på noen måte som strider mot dette. (For så vidt gjelder dette også kryptering, når det gjelder for utstederen å kontrollere at ikke annen konfidensiell informasjon slipper ut: Jfr. problemene med skjult eller slettet tekst i populære tekstbehandlingsprogrammer.)
- Ved sending av konfidensielt innhold er utstederen avhengig av at programvaren ikke lekker innhold til andre enn de tiltenkte mottakerne.
- Nøkkel som benyttes ved signaturgenerering eller dekryptering må holdes hemmelig, både ved generering, lagring og bruk. All håndtering og bruk av nøkkelen innebærer en viss risiko for lekkasje av informasjon eller for misbruk.
- Ved signaturverifisering (kryptering) kreves tillit til verifikasjonsnøkkelen, dvs. til nøkkelens integritet, til at signeringsnøkkelen (krypteringsnøkkelen) er hemmelig, til at koblingen mellom nøkkel og partsidentitet er korrekt, og, ikke minst, det som de tekniske systemene *ikke* kan sikre: at brukeren forsikrer seg om at partsidentiteten (et DN, en epost-adresse etc.) virkelig refererer til rette vedkommende.
- Mottakeren er på sin side avhengig av programvare for presentasjon og viderebehandling av mottatte data, at de presenteres som de ble sendt, at datainnholdet ikke byttes ut med noe annet etter verifikasjon av signatur, at innholdet faktisk beholdes konfidensielt etter dekryptering.

Alle disse stegene er i prinsippet like viktige. *Den digitale signaturen eller krypteringen vil aldri kunne gi større tillit enn den tilliten en har til hvert enkelt av disse stegene*. Truslene er:

Implementasjons- eller konstruksjonsfeil i programvare og utstyr

(altså feil som ikke bevisst er introdusert av en angriper). I mange tilfeller vil slike feil føre til at autentisering eller dekryptering feiler, og dermed oppdages uten at noen vesentlig skade er skjedd. Men slike feil kan også føre til at hemmelige nøkler kompromitteres, at ugyldige verifikasjons- eller krypteringsnøkler aksepteres som ekte, at svakere kryptosystemer blir brukt i stedet for sterke, at falske dokumenter aksepteres som autentiske eller at konfidensielle dokumenter avsløres for uvedkommende. *Spesielt er det en risiko for at kjente feil kan utnyttes av utenforstående angripere.*

Bevisste angrep (trojanske hester, virus, m.m.)

kan modifisere eller rekonfigurere programmer, databaser, kataloger osv. på en måte som i verste fall *kompromitterer hvilket som helst av stegene ovenfor*, og slik at angrepet ikke uten videre blir oppdaget. Tilliten avhenger altså av hvor godt beskyttet mot slike angrep avsenders og mottakers systemer er og hvor effektivt de er i stand til å begrense konsekvensene.

Brukerfeil

kan føre til at hemmelige nøkler kommer på avveier eller brukes til å signere dokumenter uten skikkelig kontroll, til at ugyldige sertifikater aksepteres og brukes ved signaturverifikasjon eller kryptering, til at sikkerhetspolicy i web-leser eller epost-klient konfigureres for løst m.m.

Populære plattformer, som Microsoft Windows (95, 98, men også NT), er svært dårlig beskyttet: *Mistanke om ett eneste angrep av makrovirus, ondsinnede Active-X-kontroller eller lignende er i prinsippet nok til å ødelegge tilliten til sikkerhetstjenester på den aktuelle maskinen.* Brukes programvareløsninger for nøkkeloppbevaring, er også tilliten til nøklene ødelagt. Spesielt gjelder det hemmelighold av signerings- og dekrypteringsnøkler, men også for krypterings- og verifikasjonsnøkler siden angrepet kan ha merket forfalskede nøkler som gyldige. (Det siste kan man imidlertid oppdage i etterkant.) Gode smartkortløsninger hindrer riktignok at hemmelige nøkler kommer på avveier, men kan ikke hindre at de samme nøklene blir misbrukt til å signere forfalskede dokumenter (dvs. dokumenter som er konstruert av angriperen, og som undertegneren blir lurt til å signere).

Plattformer med flernivå sikkerhet (og en løsning med nøkler i smartkort er egentlig et eksempel på det) gir større muligheter til å begrense konsekvensene av et angrep, gitt at mekanismene blir korrekt brukt.

A.2.2 Krav til kryptografiske løsninger

Kryptografiske algoritmer, protokoller og nøkler som brukes ved kryptering, dekryptering, signaturgenerering, -verifikasjon, nøkkelgenerering, -utveksling og sertifisering må være (tilstrekkelig) sikre. Ved å velge anerkjente løsninger kan vi stort sett forutsette at dette kravet er oppfylt. Dette er også en rent teknisk problemstilling som kan vurderes nokså uavhengig av de andre forholdene vi tar opp, og hvor analysene i den generelle faglitteraturen på området gjelder, slik at det ikke er noen grunn til å gå dypere inn på dette her. Vi vil bare påpeke:

Det er knapt noe teknisk problem å velge sterke nok løsninger -- de er ikke så mye mer ressurskrevende i bruk at det har noen praktisk betydning. (Men politiske reguleringer av kryptografi gjør dem mindre kommersielt tilgjengelig.)

Selv om medieoppslagene rundt «knekking av krypto» fokuserer på disse faktorene, utgjør de andre forholdene vi diskuterer i praksis vel så stor risiko. (Det er ingen grunn til å bruke ressurser på å bryte kryptonøkler hvis andre svakheter i systemene kan gi angriperne adgang til den samme informasjonen ukryptert...)

A.2.3 Krav til sertifikattjeneste

En sertifikattjeneste er ikke absolutt nødvendig, verken for konfidensialitet eller når digital signatur bare benyttes for å verifisere autentisitet, så lenge partene kjenner hverandre på forhånd og er i stand til å utveksle nøkler på en sikker måte. I de tilfellene der det bare er krav om autentisitet eller konfidensialitet, har partene i utgangspunktet ikke noen grunn til mistillit til hverandre, men bare til å frykte angrep fra utenforstående.

Det er likevel gode **praktiske** grunner til å bruke en sertifikattjeneste. I praksis kan en ikke forutsette at partene kjenner hverandre, og i hvert fall ikke at de møtes direkte eller har andre kommunikasjonskanaler som kan brukes for å utveksle nøkler på en sikker måte. Når en etablert PKI-infrastruktur finnes, er dette en god mekanisme for å utveksle nøkler.

Det mest fundamentale kravet til en sertifikattjeneste er de grunnleggende sikkerhetskravene:

- Bindingen mellom identitet og nøkkel gjennom sertifikatet er korrekt.
- Mottakere av sertifikater har en effektiv måte å verifisere at sertifikatet er gyldig (gjennom katalogtjeneste og/eller tilbakekallingslister).
- Hvis private nøkler genereres hos sertifikatautoriteten, at de holdes hemmelig.
- At sertifikatautoritetens interne systemer opereres sikkert.

Når partene ikke kjenner hverandre, har sertifikatautoriteten et annet (ikke-teknisk) ansvar. Det er sertifikatautoriteten (evt. indirekte via en registreringsautoritet) som går god for den parten som blir sertifisert:

- At sertifikat bare utstedes til parter som har krav på de rettighetene eller den tilliten som sertifikatet impliserer.
- At parten selv opererer sine systemer for signaturgenerering henholdsvis behandler konfidensielle data på en betryggende måte.

Nøyaktig hvordan sertifikatautoriteten utøver dette ansvaret er definert i sertifikatpolicy og -praksis. I praksis vil sertifikatautoriteten gi en garanti overfor dem som mottar og stoler på sertifikater. Den som vil ha utstedt et sertifikat, må på sin side inngå en avtale med sertifikatautoriteten, som gir sertifikatautoriteten en sanksjonsmulighet hvis sertifikateieren misligholder sin del av avtalen (dvs. et regresskrav). Sertifiseringen er f.eks. ikke noe verd dersom den som har fått et sertifikat ved uaktsomhet lar sin private nøkkel komme på avveie, så sertifikatautoriteten må ha betingelser som legger ansvaret på sertifikateieren i dette tilfellet.

A.2.4 Transport- og nettverkssikkerhet

Transport- og nettverkslagssikkerhet stiller stort sett de samme kravene som meldingssikkerhet, både når det gjelder generelle sikkerhetskrav til utstyret, og krav til kryptosystemer og sertifisering. En av hovedforskjellene ligger nettopp i at det som blir beskyttet av sikkerhetsmekanismene er trafikk på et lavere nivå i protokollhierarkiet, mellom maskiner eller programmer (prosesser). Det er ikke alltid like lett å knytte det som skjer på dette nivået til ansvarlige personer og meningsfylte operasjoner på anvendelsesnivået. En praktisk konsekvens av dette er f.eks. at en ansvarlig bruker ikke kan gi en bekreftelse hver gang en nøkkel brukes for autentisering, autentiseringen skjer automatisk når det er behov for det, og en baserer seg om mulig enda mer på tillit til programvarens og utstyrets integritet. Men omvendt betyr det også at autentiseringen eller andre sikkerhetstjenester på lavere lag ikke kan oppfattes som like sterke. Spesielt betyr det at ikke-benekting på lavere lag er meningsløst.

Relasjonen mellom autentisering og konfidensialitet faller litt annerledes ut på lavere lag, spesielt for transportlaget, nettopp fordi konteksten for sikkerhetstjenestene er selve datakanalen. Konfidensialitetstjenesten i forbindelse med meldingssikkerhet brukes nærmest for å «autentisere» mottakeren, dvs. sikre at meldingens innhold bare når rette vedkommende. På transportlaget setter man opp en datakanal mellom to parter og bruker en eller annen form for autentiseringstjeneste for å sikre at man vet hvem som er i den andre enden. (Kryptografiske mekanismer, som kryptering, digital signatur eller autentiseringskoder er i seg selv mer underordnede, de sikrer at kanalen ikke «lekker» verken ut eller inn.)

A.3 Forutsetninger for ikke-benekting

For å oppnå ikke-benekting må alle forutsetninger for autentisitet først være oppfylt. Dersom det er tvil om at et signert dokument er autentisk, vil den som angivelig signerte dokumentet uten videre kunne påberope seg dette og påstå at et omstridt dokument må være forfalsket.

Dette er et mønster vi finner igjen i flere punkter nedenfor: Enhver uønsket mulighet i systemet gjør det umulig for en tredjepart å avgjøre hva som faktisk skjedde, dvs. hvilken av partene som snakker sant. Den ene parten kan direkte utnytte hullet til sin fordel, f.eks. forfalske et dokument som utgir seg for å komme fra den andre. Men siden denne muligheten ikke kan utelukkes, kan den andre parten på sin side påstå svik, selv når det ikke er tilfelle: Den som har sendt et dokument kan etterpå påstå det må være forfalsket av mottakeren, og løpe fra ansvaret.

I avsnittene nedenfor tar vi for oss de punktene der ikke-benekting stiller sterkere eller andre krav enn autentisering.

A.3.1 Generelle sikkerhetskrav

Sikkerhetskravene generelt er ikke vesensforskjellige for ikke-benekting og autentisitet, men de vil i alminnelighet være sterkere for ikke-benekting. Grunnen er at systemet skal produsere bevis som er holdbare overfor en tredjepart (i verste fall, en domstol), når en må regne med at den ene eller andre parten kan komme til å bestride dem.

Oversikt og kontroll

Ikke-benekting krever at underskriveren der og da får presentert dokumentet som skal signeres og utløser signaturgenereringen. Dokumentet må være til stede og presenteres lokalt på den samme maskinen som signaturen faktisk beregnes. (Jfr. neste avsnitt).

Det er et opplagt krav at selve dokumentpresentasjonen på alle måter er korrekt og får frem dokumentets betydning. Men dokumentet må også presenteres i korrekt sammenheng, slik at det er klart hva konsekvensene av å signere er. (F.eks.: selv om dokumentet er det samme, er det en vesensforskjell mellom en kontrasignatur og en signatur som forplikter en person til innholdet -- eller en forpliktende signatur og en signatur som bare tjener til å beskytte innholdet under overføring og lagring.)

Programvare- og utstyrsplattform

Underskriveren er alltid i siste instans den som er ansvarlig for integriteten til all programvare og utstyr som brukes til signeringen, altså at signaturløsningen virkelig signerer det (og bare det) som det var hensikten å signere. Siden det i praksis aldri er mulig for underskriveren selv å kontrollere at alle ledd i kjeden gjør det de er forutsatt, er det et tillitsforhold mellom brukeren og leverandører (av programvare og utstyr), driftspersonell (som installerer programvare og utstyr), nettverksoperatører (som setter opp brannvegger og forhindrer angrep) og andre som kan ha mulighet til å påvirke de komponentene som korrekt signaturgenerering avhenger av.

For ikke-benekting er dette forholdet spesielt viktig. I prinsippet delegerer underskriveren signaturmyndigheten sin til programvaren som genererer signaturen. **Spesielt betyr det at signaturen ikke kan genereres av programvare som leveres av den andre parten, eller på utstyr (f.eks. tjenere) som den andre parten opererer.** Dette ville undergrave hele hensikten med ikke-benekting, siden mottakeren nå kan anklages for selv å være skyld i feil i dokumentet som ble signert eller signaturen selv -- i verste fall bevisst forfalskning.

A.3.2 Kryptografisk løsning

For autentisitet er symmetrisk krypto tilstrekkelig (dvs. en felles, hemmelig nøkkel). Ikke-benekting krever asymmetrisk krypto, ellers ville mottaker kunne fabrikere forfalskede, signerte dokumenter (og den som angivelig har signert vil kunne vise til denne muligheten og benekte signaturen).

Den litt tilfeldige betegnelsen «sterk kryptografi» viser til kryptosystemer (algoritmer, nøkkellengder) der det ikke er noen kjente, praktisk gjennomførbare angrep på selve kryptosystemet. For ikke-benekting og spesielt kvalifiserte signaturer er alt som ikke faller inn under dette begrepet uaktuelt. Spesielt er «eksport-grad»-kryptosystemene i Netscape og Internet Explorer for svake.

Nøkkel-bruksområder

Nøkler for bruksområdene autentisering og ikke-benekting må normalt holdes strengt adskilt. For ikke-benektingsnøkler vil en normalt kreve spesielt sterke sikkerhetsforanstaltninger:

- Underskriveren skal bekrefte (f.eks. ved PIN-kode) hver enkelt signatur.

- Separat autentiseringsmekanisme til nøkkelmediet for å aktivere nøkkelen (f.eks. en egen PIN-kode til smartkort, slik at underskriveren ikke er i tvil om hvilken nøkkel som brukes når kortet inneholder flere).
- Nøkkelen må bare brukes på tiltrodde utstyrsplattformer.

For autentiseringsnøkler er dette ofte overflødig, og i verste fall i konflikt med måten de brukes på.

En annen grunn er: De fleste autentiseringsprotokoller opererer ved at den som krever autentisering av den andre parten sender en tilfeldig utfordring, som den som skal autentiseres krypterer med sin private nøkkel. Digital signatur utføres på sin side ved å kryptere en hash-kode beregnet over dokumentet. Hvis samme nøkkel er gyldig både for autentisering og ikke-benekting, kunne motparten i visse tilfeller sende noe som angivelig var en tilfeldig utfordring, men i virkeligheten var hash-koden for et forfalsket dokument, og dermed oppnå å få tilbake noe som kunne gis ut for å være en bindende signatur.

A.3.3 Sertifikattjeneste

En form for nøytral tredjepart er essensiell for å oppnå ikke-benektbarhet, mens sertifikatautoriteten bare er en praktisk ordning ved autentisering. I praksis er grunnlaget for ikke-benekting at den som signerer og siden løper fra ansvaret risikerer sanksjoner (f.eks. erstatningskrav eller straff) eller andre ubehagelige konsekvenser (f.eks. tap av tillit, ugyldighet av andre signaturer), i ytterste konsekvens ved at saken blir brakt inn for en domstol. Den som har fått utstedt et sertifikat (sertifikateier), har inngått en avtale med sertifikatautoriteten. Denne avtalen stiller som nevnt krav til sertifikateier, og er grunnlag for å stille sertifikateieren økonomisk ansvarlig ved mislighold. For et sertifikat som er gyldig for ikke-benekting vil bl.a. avtalen (ved å henvise til sertifikatpolicy og -praksis) stille krav om forsvarlig håndtering av private nøkler og til å melde fra ved mistanke om at privat nøkkel er kommet på avveie. Hvis sertifikateieren da i etterkant benekter en signatur ved å påstå at privat nøkkel må være kompromittert, er det samtidig en innrømmelse av ikke å ha oppfylt sin del av avtalen.

Hvis saken bringes frem for en domstol, har sertifikatautoriteten rollen som nøytral tredjepart som frembringer bevismateriale.

Tidsavhengighet

Av tekniske årsaker, og på grunn av muligheten for tilbakekalling av sertifikater (i tilfelle den private nøkkelen blir kompromittert) er gyldigheten av sertifikater, og dermed digitale signaturer, alltid tidsbegrenset. For autentisering er dette sjelden noe problem, fordi mottaker av en signatur vanligvis verifiserer signaturen umiddelbart. Ved ikke-benektbarhet er problemet større, fordi signaturen må kunne verifiseres av en tredjepart når det oppstår en tvist, og dette kan bli nødvendig å gjøre på et tidspunkt da sertifikatet ikke lenger er gyldig.

Dette problemet kan løses ved tidsstempling og logging hos tredjepart. (Tidsstemplet er i prinsippet bare en ny signatur, utført av tredjeparten, som dekker dokumentet eller den opprinnelige signaturen pluss en angivelse av tidspunktet, altså en attest fra tredjeparten om at dokumentet er sett og logget på et visst tidspunkt. Loggingen er det primære; tredjepartens grunnlag for å kunne bekrefte signaturen på et senere tidspunkt.) Tredjeparten kan dermed bevitne at dokumentet eller signaturen forelå på et tidspunkt da sertifikatet signaturen viser til var gyldig.

Entiteten som signeres

- Entiteten (dokumentet) som signeres må ha en veldefinert betydning, som også en tredjepart kan etterprøve. Hvis ikke er ikke-benekting meningsløst, signaturen bekrefter bare at partene har utvekslet en viss bitstreng, men de er ikke enige om hva denne bitstrengen betyr.

- Syntaks og semantikk (data- eller dokumentformatet) for det som blir signert må være definert (i en standard eller spesifikasjon som begge parter slutter seg til).
- Betydningen av det som er signert kan ikke avhenge av noe utenfor dokumentet selv (som ikke er omfattet av signaturen eller avtaler som partene har inngått). Det er nettopp for å understreke at det som er signert må være et sluttet hele at vi bruker ordet dokument.

Dette betyr at ikke-benekting vanskelig kan gjøres gjeldende for transportlags- eller nettverkslags-sikkerhet (f.eks. SSL). For det første: På dette nivået oppfattes innholdet i pakkene eller datastrømmene bare som en bitsekvens (selv om det kan være mulig å rekonstruere protokollen på applikasjonsnivå). Dessuten vil hver enkelt pakke normalt signeres hver for seg, mens betydningen av innholdet i den avhenger av hva som tidligere har vært utvekslet (og ikke nødvendigvis bare i denne sesjonen).

B. Mekanismer for web-leser-utvidelser

Mange web-baserte anvendelser krever tilpasning eller utvidelse av web-lesernes funksjonalitet. Det finnes forskjellige mekanismer for å gjøre dette gjennom nedlastet kode, men mange av disse har iboende svakheter eller restriksjoner som gjør at de ikke uten videre kan brukes i sikkerhetskritiske sammenhenger. Samtidig er sikkerhetsmekanismer og -protokoller et typisk eksempel på tilfeller hvor enkelte funksjoner nødvendigvis **må** utføres lokalt, for eksempel digital signatur hvor dokumentet som skal signeres må finnes lokalt og kunne kontrolleres av den ansvarlige brukeren før signaturen genereres.

Et felles problem for alle klientside-løsninger er sikkerhetsrisikoen (virus og trojanske hester) ved å laste ned kode over nettet og kjøre den. En løsning på dette er å hindre nedlastet kode i å utføre sikkerhetskritiske funksjoner, altså behandle koden som ikke-tiltrodd. Denne modellen brukes i JavaScript og Java applets. Dette impliserer imidlertid at koden ikke kan utføre kritiske operasjoner som bl.a. signaturgenerering, som vi har behov for. (Selv om sikkerhetsmodellen i JavaScript og Java i teorien er trygg, er det dessuten mange eksempler på feil i implementasjonene som åpner mer eller mindre alvorlige hull i «sandkassen».)

Den andre løsningen er å autentisere koden slik at den (til en viss grad) kan behandles som tiltrodd, enten ved at koden som lastes ned er digitalt signert, eller ved at den lastes ned via en trygg kanal (SSL). Men dette er heller ikke problemfritt: Sikkerhetsfunksjoner som utføres av slik kode blir aldri sterkere enn autentiseringen av koden selv. Det er også verd å merke seg at de etablerte oppleggene for kodesignering vanligvis ikke innebærer noe krav om kvalitetskontroll av koden. (Det går i prinsippet an å sette opp sertifikatautoriteter som i sertifikatpolicy krever en definert grad av kvalitetskontroll for signert kode, men vi kjenner ikke til at noen slike er operative.) En slik kvalitetskontroll **kunne** være vilkårlig sterk, men det er en iboende begrensning at den bare sertifiserer den nedlastbare koden isolert sett, ikke koden i den omgivelsen den skal kjøre. Dermed er det uklart om den f.eks. kunne sertifisere en nedlastbar kodemodul som del av et «godkjent signaturgenereringsprodukt» for kvalifisert signatur.

De to typene mekanismer som er nevnt ovenfor bør kombineres, slik at autentisert kode ikke får fulle rettigheter, men at rettighetene graderes etter hvilken tillit en har til ansvarlig utgiver (og evt. garantier i sertifikatpolicy).

Installasjon av programvare fra nettet uten å bruke noen av disse sikkerhetsmekanismene er (dessverre) vanlig praksis, men **bør i det hele tatt ikke forekomme på en maskin som utfører noe mer enn minimalt sikkerhetskritiske funksjoner**. Det er i alle fall uakseptabelt å gjøre det med programvare som er direkte involvert i sikkerhetskritiske funksjoner som signaturgenerering og lignende.

B.1 HTML-scripting

Det finnes en håndfull forskjellige scriptspråk som kan brukes for scripting i HTML-sider. Netscapes *JavaScript* og Microsofts *JScript* er noenlunde kompatible, og nærmer seg en felles standard *ECMAScript* [ECMA-262] (se f.eks. Cetus-links.) *WMLScript* i WAP er også i denne familien. I Internet Explorer finnes dessuten *VBScript*, som er en delmengde av Visual Basic.

Alle script-språkene har en begrenset grad av aksess til elementene i HTML-dokumentet (bl.a. felter i skjema), og til web-leser-vinduet (bl.a. mulighet for å åpne dialogbokser eller åpne andre web-sider). Grensesnittet mot HTML-dokumentet og web-leseren betegnes ofte som DOM (Document Object Model), som strengt tatt ikke er en del av scriptspråket selv, og ikke er like standardisert som språket.

Script er **ikke-tiltrodd** kode, siden brukeren kan motta og utføre script fra ukjente og potensielt ondsinnede kilder. Derfor er det svært begrenset hvilke funksjoner som kan utføres fra script. Generelle

signatur- eller kryptofunksjoner, eller aksess til smartkort/-leser bryter med sikkerhetsmodellen for script, og ville vært et alvorlig sikkerhetshull (på linje med muligheten for å lese og modifisere lokale filer.)

Netscapes JavaScript (i Communicator 4.x) har en utvidelse for å kunne generere digital signatur på en begrenset måte. Den samme utvidelsen er også foreslått for WMLScript [WAP-161] . Funksjonen `crypto.signText` signerer en tekst (generert av scriptet). Hver gang funksjonen kalles, presenteres teksten i en dialogboks for brukeren, som eksplisitt må godkjenne signaturgenereringen. Denne mekanismen håndterer bare korte tekster som kan presenteres i dialogboksen, og er ikke praktisk brukbar for å signere et stort skjema.

Den har imidlertid noen nøkkelegenskaper som gjør den rimelig trygg:

- Signeringsmekanismen er en innebygd del av den lokale web-leseren (som i seg selv kan forutsettes å være tiltrodd).
- Mekanismen kobler presentasjon av det som skal signeres uløselig sammen med at brukeren bekrefter signaturen, uten at scriptet kan påvirke dette på noen måte.
- Scriptet selv håndterer ikke PIN-koder eller tilsvarende.

Alle andre mekanismer for signaturgenerering fra script må ha disse egenskapene for å være akseptable. Såvidt vi vet er dette den eneste ferdige signaturgenereringsmekanismen som finnes i noen av de aktuelle web-leserne.

Teksten som signeres med `crypto.signText` kan formateres med HTML-oppmerking (lagt inn av scriptet). Vi har ikke undersøkt nøyaktig hva som er mulig, men dette kan være en sikkerhetsmessig svakhet dersom formateringen kan utnyttes til å villede brukeren om hva som faktisk signeres. (Legg merke til at den grafiske presentasjonen av teksten potensielt kan påvirke betydningsinnholdet: En angriper kan kanskje generere en tekst med formatering -- font, farge etc. -- som gjør deler av teksten usynlig på en normalkonfigurert web-leser. I enkelte typer dokumenter brukes grafisk presentasjon til å skille ut kommentarer og annen tekst som ikke er bindende. Kreative triks med skjult eller alternativ tekst er også tenkelig.)

B.2 Script-språk med utvidelser

Mekanismene som i tilfelle må brukes for å få til signaturgenerering finnes i DOM, og er altså til en stor del forskjellige mellom forskjellige web-lesere. Det å hente ut innhold fra et HTML-skjema ser ut til å være mulig å gjøre på en felles måte, mens selve signaturgenereringen må gjøres på helt forskjellige måter for forskjellige web-lesere.

Det finnes noen muligheter til å installere utvidelser (dvs. nye objekter/funksjoner i objektmodellen) som kan gi anledning til å aktivere en digital signatur-mekanisme. I praksis ville dette måtte skje gjennom Active-X-kontroller (VBScript, ActiveX-objekt i JScript) eller via Java (Netscapes LiveConnect, Microsofts Java/Active-X-bro), altså plattformspesifikke løsninger. Merk at implementasjonen av utvidelsen må være tiltrodd, m.a.o. lokalt installert av brukeren selv eller som signert kode via nettet. (Se underkapitlene om signert Java og Active-X.)

For ikke-tiltrodde script vil selve signaturmekanismen i sin helhet måtte legges i en utvidelse, f.eks. en Active-X-kontroll merket «safe for scripting». Selve mekanismen måtte være utformet tilsvarende `crypto.signText` ovenfor. Dette vil kunne være en akseptabel løsning, men en må være oppmerksom på risikoen for at scriptet aktiverer signaturfunksjonen på noe som ikke skulle vært signert.

B.3 Signerte script

JavaScript 1.2 og høyere (Netscape) tilbyr også *signerte* script. Sikkerhetsmodellen er avledet av Java, men har vesentlige forskjeller pga. fundamentale forskjeller i språkene. Stort sett de samme sikkerhetsbetraktningene som for signert Java-kode gjelder (se nedenfor.)

JavaScript (dvs. JavaScripts DOM) inneholder ikke noen mekanismer som kan brukes til signaturgenerering (bortsett fra `crypto.signText`), og signaturgenereringen vil stadig måtte basere seg på egenutviklede utvidelser (f.eks. Java-objekter via LiveConnect). Fordelen med signerte script vil i tilfelle være større frihet i hvordan disse utvidelsene blir utformet, f.eks. at de kunne bruke Active-X-kontroller som ikke generelt er «safe for scripting».

B.4 Java-applet

På samme måte som script er usignerte Java-applets **ikke-tiltrodd** kode. Sikkerhetsmodellen for Java blokkerer for signaturgenerering, aksess til smartkort etc. Det finnes heller ikke noen standard mekanisme i Java tilsvarende Netscapes `crypto.signText`.

Applets kan brukes for dokumentpresentasjon. Men presentasjon av et dokument for at underskriveren skal kunne kontrollere det før signering er en sensitiv operasjon som **ikke** bør utføres av ikke-tiltrodd kode. Vinduer som presenteres av en applet er identifisert som sådan, nettopp på grunn av appletens mulighet til å presentere et brukergrensesnitt som gir seg ut for noe annet enn det er. Ingen sikkerhetsbevisst bruker må gjøre noen sikkerhetskritiske aksjoner bare på grunnlag av informasjon presentert i slike vinduer, eller taste inn sensitiv informasjon i dem.

B.5 Signert Java-applet

Signerte applets har potensielt alle rettigheter som lokalt installert Java-kode kan ha. Avhengig av hvem som har signert en applet, kan brukeren selv tildele rettigheter, dvs. på kontrollert vis åpne veier ut av Java-«sandkassen».

Brukeren må altså:

- (Importere sertifikatet til sertifikatautoriteten som har sertifisert applet-utgiveren.)
- (Importere sertifikatet til applet-utgiveren.)
- Åpne Java «privilege manager»-dialogen for applet-utgiverens sertifikat og krysse av for de navngitte rettighetene som kreves, f.eks. det som gir adgang til smartkort og signaturfunksjoner.
- Alternativt ber appleten selv om de privilegiene den trenger, og brukeren må bekrefte at dette er godtatt.
- Alle disse operasjonene bør bekreftes manuelt, slik at brukeren er klar over hva han gjør og konsekvensene av det.

Når først koden er signert er det sikkerhetsmessig nesten irrelevant hvilken tjener den faktisk lastes ned fra. Merk: **Kodesignering er ingen kvalitetskontroll**. Formålet er å autentisere koden. Bruker eller lokal systemansvarlig må uavhengig av mekanismen bestemme policy, altså hvilke privilegier som er knyttet til hvert enkelt sertifikat.

Noen av plattformene (Communicator 4.x) kan, som et alternativ til å signere koden, håndtere kode som er lastet ned via en SSL-forbindelse (`https:`) som tiltrodd: Privilegiene tildeles da på basis av tjenerens SSL-sertifikat (altså: som om koden var signert med dette sertifikatet).

Problemet med signerte Java-applets er at det er svært varierende støtte i ulike Java-plattformer, og Java-koden må tilpasses plattformen den skal kjøre på.

- JVM 1.1 utfører signerte applets som om de er lokale (dvs. full tillit, samme modell som for Active-X-kodesignering, med de samme problemene, se nedenfor.)
- Microsoft Java svarer noenlunde til JVM 1.1.
- Netscape Communicator 4.x har en fleksibel mekanisme for å be om og tildele rettigheter, basert på opphavet til koden. Systemet av rettigheter er dessverre ikke integrert med kryptografi-støtten i Java.
- JVM 1.2 (Java 2) har en mekanisme som i prinsippene er ganske lik Communicators, men forskjellig i detaljene.
- For å få en uniform plattform, er det mulig at signerte applets må kjøres i Suns Java-omgivelse, som kan installeres som en plugin i både Netscape og Internet Explorer. Dette er en ekstra komplikasjon.

Java-løsningen kan utføre kryptografiske operasjoner på to ulike måter, og reguleringen av privilegier blir litt forskjellig for de to tilfellene:

- Ved å benytte kryptografistøtte i Java Cryptography Extension. I JVM 1.2 styres adgangen til signaturgenerering ved å tildele *SecurityPermission(getSignerPrivateKey)*-tillatelsen. (Dette gir generell adgang til å bruke -- ikke lese -- alle private nøkler som er tilgjengelig for JCE på maskinen.)
- Ved å kalle metoder definert i eksterne (egenutviklede) DLL-er gjennom JNI. Disse DLL-ene går utenom Java-maskinens sikkerhetsmekanismer, og har dermed de samme potensielle sikkerhetsproblemene som plugins eller Active-X-kontroller. De bør altså installeres lokalt, ikke over nettet. Adgang til å laste og bruke en bestemt DLL med JNI-metoder tildeles gjennom *RuntimePermission(loadLibrary.{libraryName})*-tillatelsen. (Hvis det ikke er kontroll med hvor dette biblioteket stammer fra, er dette i praksis det samme som å gi alle tillatelser.)

B.6 Lokalt installert Java-kode

Java-kode (klasse-filer eller JAR-filer), evt. DLL-er som implementer «native»-metoder, kan selvfølgelig også installeres lokalt på klienten. Slik kode vil kunne få alle nødvendige privilegier, og er ikke i prinsippet forskjellig fra annen lokalt installert programvare. Lokal Java-kode blir en del av det biblioteket av kode som er kallbar fra en applet, og kan brukes for å gi kontrollert tilgang fra applets til sikkerhetskritiske funksjoner, f.eks. til å implementere en funksjon tilsvarende *Crypto.signText*.

Kode som skal brukes til å gi mindre priviligert kode adgang til mer priviligerte funksjoner må eksplisitt programmeres til dette. Mekanismene som brukes er stort sett de samme som for å tildele privilegier til signert kode, og dermed forskjellige i forskjellige Java-versjoner. I Java arves tillit i utgangspunktet fra kalleren, slik at lokal kode som kalles fra en applet ikke har større privilegier enn applet-en selv, dersom den da ikke eksplisitt overstyrer dette. (Sammenlign sikkerhetsmodellene i Java og Active-X: Active-X-kontroller består av maskinkode og får dermed med en gang alle rettigheter som den lokale brukeren har, slik at adgangen til å bruke dem må kontrolleres med flaggene «safe for scripting» og «safe for untrusted data». Lokale Java-klasser skal pr. def. være «safe for scripting» og «safe for untrusted data», og blir det automatisk hvis ikke koden selv eksplisitt «tar over ansvaret», dvs. ber om utvidede rettigheter.)

B.7 Netscape-plugin

En Netscape-plugin består av maskinkode (altså: en DLL eller delt bibliotek) og kan dermed nå alle lokale ressurser med de samme rettigheter som den brukeren som kjører web-leseren. En plugin vil altså kunne generere signaturer m.v. Plugins kan installeres over nettet på en måte som er ganske enkel for brukeren, men dette er, som all annen programvareinstallasjon over nettet, **en stor sikkerhetsrisiko**. Dermed vil vi ikke anbefale løsninger basert på plugins, med mindre disse kan distribueres på en sikker måte. (Dvs.

installeres lokalt fra diskett eller CD-ROM, evt. i signert form.) I såfall er løsningen sikkerhetsmessig tilsvarende Active-X (se nedenfor).

Siden plugins består av maskinkode, må de finnes i separate versjoner for hver enkelt plattform som Netscape kjører på. Enkelte plugins finnes bare for noen av plattformene.

B.8 Active-X-kontroller

En Active-X-kontroll er et Microsoft COM-objekt som (bl.a.) kan integreres i en web-side. (Dessuten i andre applikasjoner som opptrer som «Active-X containers», bl.a. det meste av Office-suiten.) Active-X er en Microsoft-spesifikk mekanisme, og støttes heller ikke av Netscape på Windows. Active-X-kontroller består av maskinkode, i form av en DLL, og har alle muligheter og rettigheter som ligger til den prosessen (f.eks. Internet Explorer) som kontrollen kjører inne i. Dette er spesielt problematisk på énbruker-operativsystemene Windows 95 og Windows 98, hvor Active-X-kontrollen dermed får adgang til absolutt alt på maskinen; men også for Windows NT fordi typiske NT-installasjoner er uforsiktige med å tildele rettigheter som normalt bare burde være tilgjengelige for administrator.

Active-X-kontroller kan være enten lokalt installerte eller lastes ned automatisk over nettet. På grunn av sikkerhetsrisikoen ved nedlastet kode, er nedlastede Active-X-kontroller i praksis alltid signerte. (Brukeren har mulighet til å overstyre dette og akseptere ikke-signerte Active-X-kontroller, selv om de ikke stammer fra lokal maskin.) Men det er svært få muligheter til å gradere tillit, ettersom Active-X-kontrollen selv består av maskinkode. **Microsofts opplegg for kodesignering innebærer ingen kvalitetskontroll av koden, og gir ingen garanti for at en Active-X-kontroll ikke inneholder sikkerhetshull.**

Bruk av Active-X-kontrollen styres av to definerte sikkerhetsegenskaper ved hver enkelt Active-X-kontroll:

- «Safe for untrusted data», dvs. Active-X-kontrollen kan ikke gjøre noe farlig selv om den startes opp på ukjente og ikke-tiltrodde data. En Active-X-kontroll som presenterer et rasterbilde er typisk «safe», mens f.eks. en som presenterer PostScript-kode eller et Word-dokument (med makroer) ikke er det.
- «Safe for scripting», dvs. Active-X-kontrollen kan ikke på noen måte provoseres til å gjøre noe farlig dersom den styres (mottar metodekall) fra ikke-tiltrodde script.

(Et ikke lite antall av sikkerhetshull i forbindelse med Internet Explorer eller Outlook skyldes Active-X-kontroller som feilaktig var merket med disse egenskapene!)

Active-X-kontroller kan utføre alle operasjoner som er aktuelle, og signerte eller lokalt installerte Active-X-kontroller kan være en akseptabel løsning for eksempel for signaturgenerering. Dette krever imidlertid at brukeren konfigurerer Internet Explorer til å akseptere Active-X-kontroller. Selv om standardinnstillingene åpner for det, vil vi nok anbefale IE-brukere at de **ikke** aksepterer Active-X-kontroller fra ukjente kilder, og ikke tillater scripting av Active-X-kontroller fra HTML-sider med ukjent kilde. **Spesielt** gjelder dette når maskinen skal brukes til signaturgenerering og andre sikkerhetskritiske funksjoner. Dermed må brukeren legge inn de aktuelle tjenermaskinene som «trusted site» for å kunne motta og bruke Active-X-kontroller derfra.

C. Digital signatur i web-løsninger

Behov for digital signatur (dvs. en signatur som gir ikke-benekting), kommer av og til opp i forbindelse med web-løsninger. Dette er et ganske typisk eksempel hvor sikkerhetskritiske funksjoner må utføres lokalt, og diskusjonen av løsningene nedenfor gjelder langt på vei for andre tilsvarende situasjoner også.

C.1 Forkastede løsningsalternativer

Vi vil først gå gjennom noen løsningsmønstre som kan synes plausible i utgangspunktet, men som av forskjellige grunner må forkastes. (Først og fremst fordi de ikke gir reell ikke-benektbarhet, dvs. signaturfunksjonen oppfyller ikke kravene til en kvalifisert signatur.) Likefullt er det instruktivt å se på hvorfor de ikke fungerer. Merk at disse løsningene er fullt ut brukbare dersom det bare kreves **autentisering**.

C.1.1 Tradisjonell web-løsning, sikret med SSL

Dette er antakelig den enkleste løsningen å realisere, og i hvert fall den som brukere, utviklere, driftspersonell og tjenesteleverandører er mest kjent med. Dokumentene som skal signeres foreligger som HTML-skjemaer som fylles ut i brukerens web-leser, og mottas av CGI-programmer eller andre slags transaksjonsprogrammer, avhengig av hva som er tilgjengelig og praktisk på den valgte implementasjonsplattformen. Sikkerheten ivaretas ved å bruke SSL med klientautentisering på sesjonen mellom web-leseren og web-tjeneren.

Denne løsningen har flere alvorlige ulemper, og er derfor uaktuell:

- SSL gir ikke noen form for ikke-benekting. (Det kan hende det ville være mulig å logge alle data som utveksles over sesjonen, og at dette kunne gi tilstrekkelig sporbarhet i tilfelle tvister, men det er høyst tvilsomt. Merk at det brukes symmetrisk kryptografi for å autentisere data som utveksles i sesjonen, så autentiseringskodene alene er ikke tilstrekkelig bevis for innholdet som ble utvekslet.)
- Det er helt uaktuelt å installere kvalifiserte sertifikater som klientautentiseringssertifikater for SSL. Vi må gå ut fra at dette vil være i strid med enhver realistisk sertifikatpolicy for kvalifiserte sertifikater. Brukeren vil altså misligholde avtalen med sertifikatautoriteten og utsette seg for sanksjoner (ved siden av risikoen for misbruk av nøklene).
- Formatet for HTML-skjema er ikke spesielt velegnet for å bli signert over. I det minste kreves en del omtanke i design og spesifikasjon av detaljene i skjemainnholdet for at betydningen skal være veldefinert og entydig, og brukerne må også være innforstått med dette.
- Hvis web-tjeneren selv opereres av den parten som mottar skjemaene og som krever ikke-benektbarhet, er det uheldig at presentasjonen av skjemaet styres av HTML-koden som kommer fra web-tjeneren.

Dersom en tredjepart skal kunne dømme i en tvist, må HTML-koden som definerer (presentasjonen av) skjemaet kunne knyttes til resultatet av utfyllingen slik at det ikke er tvil om hva brukeren så på skjermen. Dette er ikke praktisk gjennomførbart med standard verktøy. Stilark vil forverre problemet, ettersom presentasjonen av skjemaet både avhenger av stilark og oppsett lokalt hos brukeren, og av stilark som følger skjemaet fra web-tjeneren. I prinsippet må alt dette logges, ellers kan en ikke utelukke f.eks. at tekst på skjemaet ble usynlig pga. uheldige kombinasjoner av farger, størrelser på skjemaelementer o.l.

Det gjør ikke noen vesentlig forskjell om andre innholdsformater enn HTML-skjema blir brukt.

C.1.2 Tradisjonell web-løsning med uavhengig tredjepart

Denne løsningen er den samme som den ovenfor, bortsett fra at web-tjeneren kjører hos en uavhengig tjenesteleverandør. Hovedproblemet består, at SSL i seg selv ikke gir ikke-benekting.

Denne løsningen totalt sett gir imidlertid en viss grad av ikke-benekting likevel, på grunn av at den går gjennom en uavhengig tredjepart som kan loggføre all trafikk. Spesielt vil tredjeparten være i stand til å koble skjemadefinisjonen (dvs. eksakt den HTML-koden som opprinnelig ble sendt til klienten) med det utfylte skjemaet som klienten returnerte. Tredjeparten har altså i prinsippet samme rolle som sertifikatautoriteten i en løsning med digital signatur.

C.1.3 Signaturgenerering i klient

Selve signaturgenereringen **må** utføres i klienten, ute hos underskriveren. (Hvis ikke ville det vært som om den som signerer skulle gitt den andre parten blankofullmakt til å signere.) En nøkkel som kan generere en ikke-benektbar og, spesielt, kvalifisert signatur må eieren aldri slippe utenfor sin kontroll, hvis ikke må den umiddelbart betraktes som kompromittert. Vi kunne tenke oss en variant av signaturgenereringen der hash-koden ble generert på web-tjeneren, overført til klienten, signert der og så returnert til web-tjeneren.

Denne løsningen må vi også forkaste:

- Brukeren har ingen garanti for at det som web-tjeneren presenterer som dokumentet som skal signeres samsvarer med hash-koden som oversendes. (Web-klienten ser bare den representasjonen av dokumentet som web-tjeneren velger å generere, f.eks. HTML-siden som er resultatet av å presentere et XML-dokument via XSL og kan ikke selv kontrollere hash-koden.)
- Hash-koden er bare beskyttet av SSL-autentiseringsnøkkelen/sertifikatet under overføring, og som nevnt ovenfor er dette vesentlig svakere enn et kvalifisert sertifikat.

(Denne løsningen kan vi sammenligne med å få lest opp kontrakten av motparten over telefon, og deretter få oversendt den siste siden for å undertegne...)

Løsningen vil være noe mer spiselig dersom tjeneren blir operert av en uavhengig tredjepart, men uansett vil løsningen langt fra oppfylle kravene til kvalifisert signatur.

C.2 Løsningsalternativer

C.2.1 Krav til akseptable løsninger

Følgende sikkerhetsrelaterte krav må oppfylles:

1. Signaturen må oppfylle kravene til kvalifisert signatur, og sertifikatet må dermed tilfredsstille kravene til kvalifisert sertifikat. (I så fall vil sertifikatpolicy antakelig stille krav som er ekvivalente til kravene nedenfor.)
2. Signeringsnøkkel må være under eierens kontroll til en hver tid. Signeringen må foregå inne i et smartkort som besittes av eieren (underskriveren), og hvor PIN-kode eller lignende eksplisitt må gis inn for hver signatur som genereres.
3. Originaldokumentet som skal signeres, ikke bare en transformert presentasjonsform av det, må være til stede på klientmaskinen der signeringen foregår. Underskriveren må være i stand til å lese dokumentet på skjerm, eller på annen måte forsikre seg om at dokumentet som blir signert er det rette.
4. Programvare som brukes for presentasjon av dokumentet og generering av signatur opererer på vegne av underskriveren, og må stamme fra en uavhengig tredjepart. (Det er ikke akseptabelt at

den kontrolleres av mottakeren som krever signatur. F.eks. vil **ikke** plugins, applets eller annen programvare nedlastet fra mottakerens web-tjener kunne brukes.)

5. Nedlasting og installasjon av ekstra programvare (plugins etc.) må ikke skje på en måte som gir risiko for kompromittering av klientmaskinen (virus, trojanske hester).

Punkt 3 betyr at for å få generert en signatur på et dokument, må originaldokumentet lastes ned til klienten. Formatet på originaldokumentet må være et som kan presenteres for underskriveren på klienten. Det betyr at det enten må være i HTML, XML, PDF eller andre formater som en kan anta at underskriveren har eller kan skaffe presentasjonsmoduler for i sin web-leser, eller at det må utvikles og legges til rette for at underskriveren kan få skaffet applets, plugins eller lignende fra en uavhengig tredjepart for å håndtere dokumentformatet.

Det er ikke nødvendig å returnere hele det signerte dokumentet fra klienten, såfremt dokumentet ikke er endret. Det er tilstrekkelig å returnere selve signaturen, som kan kombineres eller assosieres med resten av dokumentet på tjeneren.

Web-leserne inneholder ikke noen direkte støtte for å generere signaturer på web-sider eller web-skjema. (En praktisk funksjon, som altså ikke finnes, ville være HTTP «file upload» eller «form submit» med automatisk S/MIME-signatur.) Funksjonene for S/MIME-signering som finnes både i Netscape Messenger og Microsoft Outlook gjelder epost-meldinger. Dette innebærer ett av to:

Hvis ned- og opplasting av skjema/signatur skal skje interaktivt, ved HTTP, må signaturgenereringen utføres i en eller annen form for ekstern programvare, som applets, plugins etc.

Alternativt overføres meldinger til epost-klienten for signering, og signerte meldinger returneres pr. epost.

C.2.2 Valg av protokoll/sikkerhetssyntaks

S/MIME [RFC 2633] (PKCS#7 [PKCS#7] hhv. CMS [RFC 2630] innkapslet i MIME) er den mest utbredte sikkerhetssyntaksen for meldingssikkerhet. Denne (dvs. S/MIME versjon 3) blir Internett-standard, og brukes i andre standarder/profiler som EDIINT. Dessuten er implementasjoner av S/MIME integrert i mange utbredte produkter. Det er derfor liten grunn til å velge noe annet for web-løsningen. M.a.o., den MIME-innholdsdelen som inneholder det utfylte skjemaet sikres ved hjelp av S/MIME. (Det kan være aktuelt å følge EDIINT-rekommendasjonen for EDI over HTTP. Denne definerer bl.a. bruk av kvitteringsmeldinger, som kan være praktisk, men dette er ikke strengt nødvendig. I så fall kan HTTP-responsen inneholde kvitteringsmeldingen.)

Protokollen S-HTTP [RFC 2660] sikrer utvekslingen på HTTP-nivå med kryptering og/eller digital signatur. (Denne protokollen er også basert på CMS-formatet.) Denne typen protokoll er ikke spesielt velegnet for vårt formål siden den signerer over HTTP-transaksjonen, ikke bare selve skjemaet. Det fantes en Netscape-plugin for denne protokollen, men såvidt vi kan se er denne ikke lenger tilgjengelig, og vi kjenner ikke til andre implementasjoner.

SSL og andre mekanismer som gir sikkerhet på transportlag eller lavere er ikke egnet. Som tidligere nevnt gir de ikke god nok støtte for ikke-benektbarhet.

C.2.3 Løsningsmønstre

Nedenfor har vi gjennomgått noen typer av løsninger som kan være aktuelle. Merk at vi ikke har gjennomført noen fullstendig evaluering av de produktene som er nevnt, og det at det er nevnt nedenfor betyr ikke at vi går god for sikkerhet og brukbarhet for hvert enkelt produkt. Produktene er tatt med rett og slett som eksempler på hva som finnes. Vi regner heller ikke med at vi har fått med **alle** aktuelle produkter.

C.2.4 Signaturgenerering knyttet til submit-aksjon

HTML-scripting kan brukes til å aktivere et script umiddelbart før «submit» av skjema (`onsubmit`). Dette scriptet kan i prinsippet lese innholdet i skjemaet og generere en signerbar representasjon av dette. (Det kan også i og for seg generere en EDIFACT-melding eller tilsvarende standardisert form av skjemaet.)

Det er en par poeng med denne løsningen som ikke er helt opplagte:

- Det lar seg ikke gjøre å sende det signerte skjemaet i script-funksjonen selv, så signaturen må legges i et skjult felt, som så implisitt blir sendt av den innebygde submit-funksjonen i web-leseren. (`onsubmit`-aksjonen er grei fordi den kjøres før submit-funksjonen samler innholdet i skjemaet og utfører HTTP POST/GET.) Signaturgenereringen kunne også gjøres i en annen script-aksjon, f.eks. onclick på en egen «Signer»-knapp. I dette tilfellet ville brukeren måtte klikke først «Signer» og så «Send».
- For å unngå å sende dobbelt opp (både skjemafeltene selv + den konverterte og signerte versjonen) kan signaturfeltene og submit-knappen legges på et **annet** skjema i samme HTML-side. (Det skjemaet som inneholder de opprinnelige feltene har da ikke noen egen submit-knapp.)

Denne løsningen har flere problemer:

- Mulighetene for signaturgenerering fra script ble diskutert ovenfor, men er generelt svært begrenset.
- Den representasjonen som signeres blir generert av scriptet, og for å få rimelig sikkerhet må brukeren finne seg i å kontrollere at den er korrekt før signeringen faktisk blir utført. (M.a.o., brukeren må forholde seg både til to representasjoner av skjemaet, både den normale som web-leseren presenterer på skjermen, og den tekstlige som er den som signeres.) Sammenhengen mellom de to må være rimelig opplagt, uten noen semantiske spissfindigheter. Hvis ikke er det stor risiko for at brukeren kan villedes til å signere noe annet enn det som var hensikten.

C.2.5 Signaturgenerering av proxy

Denne løsningen går ut på at web-klienten ikke sender skjemaet direkte til mottakstjeneren, men via en lokal HTTP-tjener (proxy) som tilføyer signaturen og sender den videre. Proxies er en vanlig teknikk ved brannvegger, og web-klientene kan settes opp til å sende all HTTP-trafikk via en proxy (for sikkerhetskontroll av innholdet). Dersom proxyen kjører på en separat maskin, som er det vanlige i brannveggløsninger, er dette ikke noen hensiktsmessig løsning.

Det finnes imidlertid noen produkter (bl.a. Algorithmic Research PrivateWire) med signering av web-skjema, som faller i denne kategorien. Ut fra produktbeskrivelsene ser dette ser egentlig ut til å være VPN/brannvegg/transportlagssikkerhetsløsninger som installerer seg i TCP-stakken (lokalt på samme maskin) og fanger opp og «filtrerer» HTTP POST-operasjoner, hvor de bl.a. kan føye til signatur på innholdet. Disse filtrene på transportlaget har i prinsippet samme funksjon som proxies, selv om de ikke er realisert som separate tjenerprosesser på en annen maskin.

En slik løsning har også flere uheldige sider i forhold til vår anvendelse:

- Det er uheldig å skille dokumentpresentasjonen (i web-leseren) og signaturgenereringen (i et eget produkt), når de to delene henger sammen på en såpass løs måte. (Igjen: Det blir for lett å lure brukeren til å signere noe annet enn det som vises i web-leseren.)
- Det er en ganske vidløftig og gjennomgripende ting for sluttbrukere å installere denne typen produkter -- spesielt når det er for én eneste anvendelses skyld.
- Det er neppe mulig å installere mer enn ett produkt av denne typen på en enkelt maskin. Brukeren får med andre ord et problem hvis han/hun skal samarbeide med flere parter som krever signerte skjema og baserer seg på forskjellige produkter.

C.2.6 Signaturgenerering i epost-brukeragent

Denne typen løsning innebærer at web-løsningen bare brukes til funksjoner som ikke krever ikke-benekting, f.eks. til å navigere i en skjemabase og velge ut hvilke skjema som skal oversendes. SSL-klientautentisering er tilstrekkelig for å implementere aksesskontroll. Dokumenter som skal signeres lastes ned, og signeres med S/MIME i epost-klienten. Poenget med denne løsningen er at S/MIME-støtte i epost-klienter er nokså utbredt.

Denne løsningen har flere varianter:

- Mest direkte ville være å sette submit-aksjonen til en mailto-URL, slik at skjemaet blir sendt via epost i stedet for HTTP. Dette strander imidlertid på at web-leserne oppfører seg forskjellig. (Alt annet enn HTTP som protokoll for submit-aksjonen er eksplisitt udefinert i HTTP 4.0-spesifikasjonen.) Ingen av dem vi har prøvd har akkurat den ønskede oppførselen. Communicator sender innholdet i skjemaet som innholdet i meldingen (som ønsket), men uten å åpne epost-klienten, så brukeren får ikke noen sjanse til å signere. Internet Explorer åpner epost-klienten, så meldingen kan signeres, men får ikke med innholdet i skjemaet.
- Epost-klienten aktiveres fra script. Dette krever plattformspesifikke utvidelser og/eller tiltrodde script. Problemene er for det første at det å aktivere en epost-klienten i seg selv er plattform- og klient-avhengig, og for det andre bør scriptet aktivere den epost-klienten brukeren prefererer (ikke bare en bestemt klient på hver plattform). På Windows finnes trolig Active-X-kontroller som kan brukes fra script og gir adgang til epost via MAPI el.l.
- Brukerens epostadresse legges inn i skjemaet (eller gjøres tilgjengelig for tjeneren på en annen måte). Transaksjonsprogrammet sender skjemaet pr. epost til brukeren, og brukeren må signere og sende tilbake. (Evt. plukke ut vedlegget som er det faktiske skjemaet og kjøre dette gjennom en frittstående signeringsapplikasjon.) Denne løsningen krever mest manuelle operasjoner, men krever ikke noe annet enn at epost-klienten har signeringsmuligheter, evt. håndterer binære vedlegg på en rimelig måte.

C.2.7 Signaturgenerering i hjelpeprogram

Skjemapresentasjon og signering skjer inne i et lokalt hjelpeprogram, og er egentlig temmelig uavhengig av web-leseren. Hjelpeprogrammet kan realiseres som signert Java-applet, Active-X-kontroll, Netscape-plugin eller lignende.

En slik løsning kan utvikles fra bunnen av, men det finnes også en del ferdige løsninger for elektroniske skjema som det er mulig å starte fra, bl.a.:

- UWI InternetForms Netscape-plugin og Active-X-kontroll, bruker CryptoAPI på Windows for digital signering. (Kan laste ned demo-eksemplar, som også tillater digital signering.)
- JetForm Netscape-plugin og Active-X-kontroll, signering v.hj.a. kryptoløsninger fra Entrust og RSA.
- Shana Informed: Signering v.hj.a. kryptoløsning fra Entrust.
- Adobe Acrobat har også støtte for web-integrerte skjema med signatur (også her kryptoløsning fra Entrust).

Merk: Det kan være eksportrestriksjoner på de versjonene av produktene som inkluderer digital signatur.

Grunnleggende i denne løsningen er at skjemadefinisjon og utfylt skjema er nye innholdstyper (MIME-typer), slik at web-leseren aktiverer hjelpeprogrammet når web-tjeneren leverer et skjema. I praksis må det defineres proprietære innholdstyper, selv om mange av løsningene vil være basert på standarder som XML. Produktene ovenfor har hver sine MIME-typer og er ikke uten videre interoperable. (Det finnes

forslag til standarder for representasjon av skjema-maler, utfylte skjema -- XForms -- og digital signatur generelt -- XML-DSig -- i XML.)

Merk også at måten signaturen er representert (sikkerhetssyntaksen) varierer. Ingen av de produktene som er nevnt ovenfor benytter såvidt vi kan se S/MIME som signaturformat, men fletter i stedet signaturen inn i selve skjemaformatet på en format-avhengig måte. Det betyr at for hvert slikt format som aksepteres, må signaturverifisering skreddersys på tjenersiden.

Hjelpeprogrammet som installeres i web-leseren presenterer skjema og genererer signatur. Dette programmet er i noen tilfeller (f.eks. Acrobat Reader) gratis tilgjengelig, men for de fleste er det reelle lisenskostnader som det ikke alltid er fornuftig å velte over på klienten. For universitets- og høyskolesektoren, evt. for det enkelte lærested, bør det være mulig å forhandle frem avtaler som gjør det mulig å distribuere klientprogramvaren fritt til ansatte og studenter. (Detaljer om prising og lisensbetingelser er stort sett ikke tilgjengelig gjennom web-sidene, og må antakelig forhandles med leverandøren for hvert enkelt tilfelle.) Verktøyene som brukes for å utforme skjema(maler) og annen støtteprogramvare på tjenersiden er stort sett proprietære (og antakelig relativt kostbare.)

De ferdige løsningene over styrer altså i stor grad skjemaformat og sikkerhetssyntaks. For å få frihet til å velge dette selv, er alternativet et egenutviklet hjelpeprogram, f.eks. i Java. I såfall kan programmet lastes ned som signert applet.

D. Referanser

[1999/93/EC]
Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. http://europa.eu.int/eur-lex/en/lif/dat/1999/en_399L0093.html

[CDSA]
Open Group Technical Standard: CDSA and CSSM, Version 2 (with corrigenda), C914, The Open Group, mai 2000, http://www.opengroup.org/publications/catalog/c914.htm

[COSS]
OMG formal/98-12-09: *CORBAservices: Common Object Services Specification*, Object Management Group, Inc., desember 1998.

[draft-aboba-pppext-eapgss]
B. Aboba: *PPP EAP GSS Authentication Protocol*, work-in-progress Internet draft, juli 2000, ftp://ftp.nordu.net/internet-drafts/draft-aboba-pppext-eapgss-01.txt

[draft-ietf-ipsec-pki-req]
R. Thayer, C. Kunzinger og P. Hoffman: *A PKIX Profile for IKE*, work-in-progress Internet draft, juli 2000, ftp://ftp.nordu.net/internet-drafts/draft-ietf-ipsec-pki-req-05.txt

[ECMA-262]
ECMA-262: *ECMAScript: A general purpose, cross-platform programming language*, ECMA, juni 1997.

[EMV-APPL]
EMV'96. Integrated Circuit Card Application Specification for Payment Systems, versjon 3.0, Europay, MasterCard, Visa, juni 1996. http://www.emvco.com/specifications.cfm

[EMV-CARD]
EMV'96. Integrated Circuit Card Specification for Payment Systems, versjon 3.1.1, Europay, MasterCard, Visa, mai 1998. http://www.emvco.com/specifications.cfm

[EMV-TERM]
EMV'96. Integrated Circuit Card Terminal Specification for Payment Systems, versjon 3.0, Europay, MasterCard, Visa, juni 1996. http://www.emvco.com/specifications.cfm

[Freier, Karlton og Kocher 1996]
A. O. Freier, P. Karlton og P. C. Kocher: *The SSL Protocol Version 3.0*, Netscape, 1996, http://www.netscape.com/eng/ssl3/draft302.txt

[FSP-1]
FSP-1:1.0, Forvaltningsnettsamarbeidets sertifiseringspolicy -- nr 1 (høysikkerhets sertifiseringspolicy for utstedelse av X.509-baserte sertifikater for personer og virksomheter i norsk offentlig forvaltning, Arbeids- og administrasjonsdepartementet, Forvaltningsnett-samarbeidet rapport 8/99, august 1999. http://forvaltningsnett.dep.no/ftp/pdf/rapporter/rapport_8-99_sert_policy.PDF

[ISO 7816-1]
ISO/IEC 7816-1:1998: *Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 1: Physical characteristics*, ISO, 1998.

[ISO 7816-2]
ISO/IEC 7816-2:1999: *Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 2: Dimensions and location of contacts*, ISO, 1999.

[ISO 7816-3]
ISO/IEC 7816-3:1997: *Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 3: Electronic signals and transmission protocols*, ISO, 1997.

[ISO 7816-4]
ISO/IEC 7816-4:1995: *Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 4: Interindustry commands for interchange*, ISO, 1995 + Amd.1:1997, 1997.

[ISO 7816-5]
ISO/IEC 7816-5:1994: *Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 5: Numbering system and registration procedure for application identifiers*, ISO, 1994 + Amd.1:1996, 1996.

[ISO 7816-6]
ISO/IEC 7816-6:1996: *Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 6: Interindustry data elements*, ISO, 1996 + Amd.1:2000, 2000.

[ISO 7816-8]
ISO/IEC 7816-8:1999: *Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 8: Security related interindustry commands*, ISO, 1999.

[ISO 7816-9]
ISO/IEC 7816-9:2000: *Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 9: Additional interindustry commands and security attributes*, ISO, 2000.

[NHD 2000]
Elektroniske signaturer, myndighetsroller og regulering av tilbydere av sertifikattjenester, rapport fra utvalg oppnevnt av Nærings- og handelsdepartementet, januar 2000. http://odin.dep.no/nhd/norsk/publ/rapporter/024005-994081/index-dok000-b-n-a.html

[PCSC-1]
Interoperability Specification for ICCs and Personal Computer Systems. Part 1. Introduction and Architecture Overview, revisjon 1.0, PC/SC Workgroup, http://www.pcscworkgroup.com/

[PCSC-5]
Interoperability Specification for ICCs and Personal Computer Systems. Part 5. ICC Resource Manager Definition, revisjon 1.0, PC/SC Workgroup, http://www.pcscworkgroup.com/

[PCSC-6]
Interoperability Specification for ICCs and Personal Computer Systems. Part 6. ICC Service Provider Interface Definition, revisjon 1.0, PC/SC Workgroup, http://www.pcscworkgroup.com/

[PKCS#7]
PKCS#7 Cryptographic Message Syntax Standard, versjon 1.5, RSA Security, Inc., november 1993, http://www.rsalabs.com/pkcs/pkcs-7/index.html

[PKCS#11]
PKCS#11 Cryptographic Token Interface Standard, versjon 2.10, RSA Security, Inc., desember 1999, http://www.rsalabs.com/pkcs/pkcs-11/index.html

[PKCS#15]
PKCS#15 Cryptographic Token Information Format Standard, versjon 1.1, RSA Security, Inc., juni 2000, http://www.rsalabs.com/pkcs/pkcs-15/index.html

[Platform SDK]
Platform SDK, Microsoft Corp., http://msdn.microsoft.com/library/psdk/portals/mainport.htm

[RFC 1424]
RFC 1424: B. Kaliski: *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services*, februar 1993.

[RFC 1968]
RFC 1968: G. Meyer: *The PPP Encryption Control Protocol (ECP)*, juni 1996.

[RFC 1994]
RFC 1994: W. Simpson: *PPP Challenge Handshake Authentication Protocol (CHAP)*, august 1996.

[RFC 2222]
RFC 2222: J. Myers: *Simple Authentication and Security Layer (SASL)*, oktober 1997.

[RFC 2246]
RFC 2246: T. Dierks og C. Allen: *The TLS Protocol Version 1.0*, januar 1999.

[RFC 2251]
RFC 2251: M. Wahl, T. Howes og S. Kille: *Lightweight Directory Access Protocol (v3)*, desember 1997.

[RFC 2284]
RFC 2284: L. Blunk og J. Vollbrecht: *PPP Extensible Authentication Protocol (EAP)*, mars 1998.

[RFC 2401]
RFC 2401: S. Kent og R. Atkinson: *Security Architecture for the Internet Protocol*, november 1998.

[RFC 2408]
RFC 2408: D. Maughan, M. Schertler, M. Schneider og J. Turner: *Internet Security Association and Key Management Protocol (ISAKMP)*, november 1998.

[RFC 2409]
RFC 2409: D. Harkins og D. Carrel: *The Internet Key Exchange (IKE)*, november 1998.

[RFC 2411]
RFC 2411: R. Thayer, N. Doraswamy og R. Glenn: *IP Security Document Roadmap*, november 1998.

[RFC 2419]
RFC 2419: K. Sklower og G. Meyer: *The PPP DES Encryption Protocol, Version 2 (DESE-bis)*, september 1998.

[RFC 2420]
RFC 2420: H. Kummert: *The PPP Triple-DES Encryption Protocol (3DESE)*, september 1998.

[RFC 2459]
RFC 2459: R. Housley, W. Ford, W. Polk og D. Solo: *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, januar 1999.

[RFC 2487]
RFC 2487: P. Hoffman: *SMTP Service Extension for Secure SMTP over TLS*, januar 1999.

[RFC 2510]
RFC 2510: C. Adams og S. Farrel: *Internet X.509 Public Key Infrastructure Certificate Management Protocols*, mars 1999.

[RFC 2511]
RFC 2511: M. Myers, C. Adams, D. Solo og D. Kemp: *Internet X.509 Certificate Request Message Format*, mars 1999.

[RFC 2527]
RFC 2527: S. Chokhani og W. Ford: *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*, mars 1999.

[RFC 2560]
RFC 2560: M. Myers, R. Ankney, A. Malpani og S. Galperin: *Internet X.509 Public Key Infrastructure Online Certificate Status Protocol -- OCSP*, juni 1999.

[RFC 2595]
RFC 2595: C. Newman: *Using TLS with IMAP, POP3 and ACAP*, juni 1999.

[RFC 2616]
RFC 2616: R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach og T. Berners-Lee: *Hypertext Transfer Protocol -- HTTP/1.1*, juni 1999.

[RFC 2630]
RFC 2630: R. Housley: *Cryptographic Message Syntax*, juni 1999.

[RFC 2633]
RFC 2633: B. Ramsdell (red.): *S/MIME Version 3 Message Specification*, juni 1999.

[RFC 2637]
RFC 2637: K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little og G. Zorn: *Point-to-Point Tunneling Protocol*, juli 1999.

[RFC 2660]
RFC 2660: E. Rescorla og A. Schiffman: *The Secure HyperText Transfer Protocol*, august 1999.

[RFC 2661]
RFC 2661: W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn og B. Palter: *Layer Two Tunneling Protocol "L2TP"*, august 1999.

[RFC 2716]
RFC 2716: B. Aboba, D. Simon: *PPP EAP TLS Authentication Protocol*, oktober 1999.

[RFC 2759]
RFC 2759: G. Zorn: *Microsoft PPP CHAP Extensions, Version 2*, januar 2000.

[RFC 2764]
RFC 2764: B. Gleeson, A. Lin, J. Heinanen, G. Armitage og A. Malis: *A Framework for IP Based Virtual Private Networks*, february 2000.

[RFC 2817]
RFC 2817: R. Khare og S. Lawrence: *Upgrading to TLS Within HTTP/1.1*, mai 2000.

[RFC 2818]
RFC 2818: E. Rescorla: *HTTP Over TLS*, mai 2000.

[RFC 2830]
RFC 2830: J. Hodges, R. Morgan og M. Wahl: *Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security*, mai 2000.

[RFC 2865]
RFC 2865: C. Rigney, S. Willens, A. Rubens og W. Simpson: *Remote Authentication Dial In User Service (RADIUS)*, juni 2000.

[RFC 2888]
RFC 2888: P. Srisuresh: *Secure Remote Access with L2TP*, august 2000.

[Saarinen 1999]
M.-J. Saarinen: Attacks against the WTLS Protocol, *Fourth Joint Working Conference on Communications and Multimedia Security (CMS'99)*, Leuven, Belgia, september 1999, http://www.jyu.fi/~mjos/wtls.ps

[Schneier og Mudge 1999]
B. Schneier og Mudge: Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2), *Secure Networking -- CQRE [Secure] '99*, Springer Verlag, 1999, http://www.counterpane.com/pptpv2-paper.html

[Solaris-smartkort]
Solaris Smart Card Administration Guide, 806-4223, Sun Microsystems, Inc., februar 2000, http://docs.sun.com

[SS 614330]
SS 614330: *Electronic ID Application*. http://www.seis.se/funktion.htm

[SS 614331]
SS 614331: *Electronic ID Certificate.* http://www.seis.se/funktion.htm

[SS 614332]
SS 614332: *Electronic ID Card -- Swedish Profile*. http://www.seis.se/funktion.htm

[Steiner, Neuman og Schiller 1988]
J. Steiner, C. Neuman og J. Schiller: Kerberos: An Authentication Service for Open Network Systems, *Proceedings of the USENIX Winter Conference*, 1988.

[TS 100 977]
TS 100 977 V8.3.0: *Digital cellular telecommunications system (Phase 2+): Specification of the Subscriber Identity Module -- Mobile Equipment (SIM -- ME) interface* (GSM 11.11 version 8.3.0, Release 1999), ETSI, August 2000.

[TS 101 267]
TS 101 267 V8.3.0: *Digital cellular telecommunications system (Phase 2+): Specification of the SIM Application Toolkit for the Subscriber Identity Module -- Mobile Equipment (SIM -- ME) interface* (GSM 11.14 version 8.3.0 Release 1999), ETSI, August 2000.

[UNISA-policy]
UNISA certificate policy (draft version), UNINETT/Norsk Regnesentral, desember 1999.

[WAP-160]
WAP-160: *Wireless Identity Module*, Wireless Application Forum, Ltd., juni 2000, http://www.wapforum.com/what/technical.htm

[WAP-161]
WAP-161: *WMLScript Cryptographic API Library*, Wireless Application Forum, Ltd., juni 2000, http://www.wapforum.com/what/technical.htm

[WAP-163]
WAP-163: *Wireless Transport Layer Security Specification*, Wireless Application Forum, Ltd., juni 2000, http://www.wapforum.com/what/technical.htm

[X.509]
ITU-T X.509 | ISO/IEC 9594-8:1997: *Information Technology -- Open Systems Interconnection -- The Directory: Authentication Framework*, ISO, 1997.

[X.521]
ITU-T X.521 | ISO/IEC 9594-7:1997: *Information Technology -- Open Systems Interconnection -- The Directory: Selected Object Classes*, ISO, 1997.