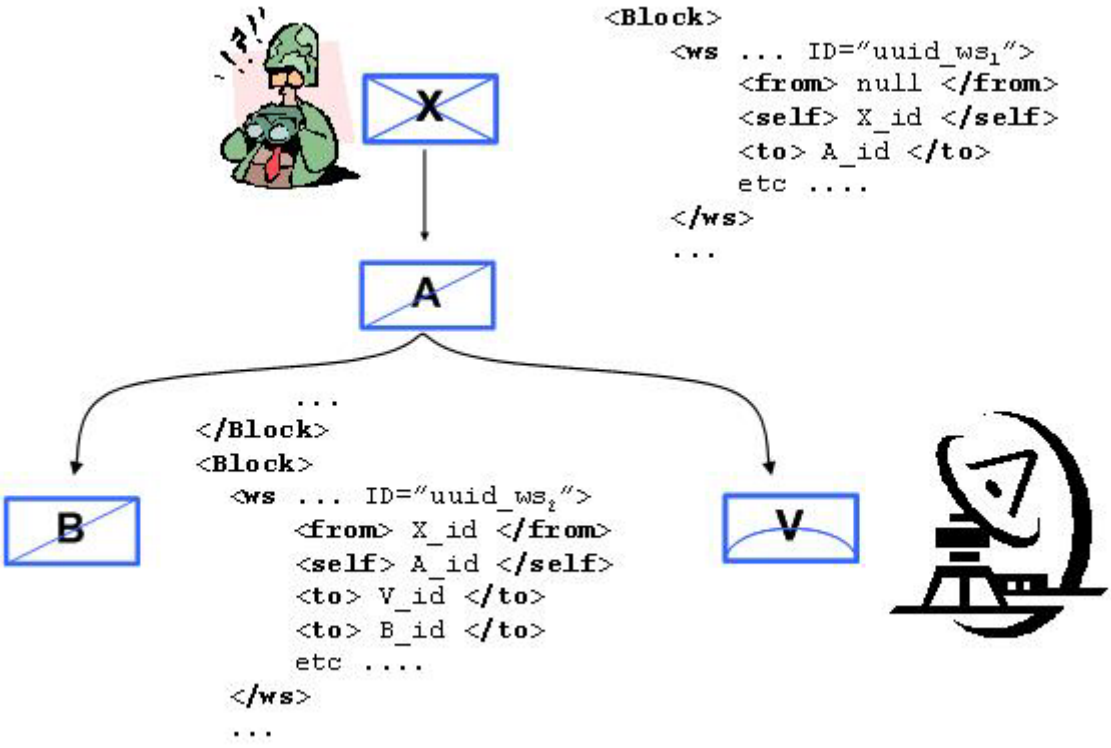
System av system - slutrapport

Alf Bengtsson, Lars Westerdahl



FOI-R-- 1830 --SE
ISSN 1650-1942

Ledningssystem
Användarrapport

December 2005

FOI är en huvudsakligen uppdragsfinansierad myndighet under Försvarsdepartementet. Kärnverksamheten är forskning, metod- och teknikutveckling till nytta för försvar och säkerhet. Organisationen har cirka 1350 anställda varav ungefär 950 är forskare. Detta gör organisationen till Sveriges största forskningsinstitut. FOI ger kunderna tillgång till ledande expertis inom ett stort antal tillämpningsområden såsom säkerhetspolitiska studier och analyser inom försvar och säkerhet, bedömningen av olika typer av hot, system för ledning och hantering av kriser, skydd mot hantering av farliga ämnen, IT-säkerhet och nya sensorers möjligheter.

FOI
Totalförsvarets forskningsinstitut
Ledningssystem
Box 1165
581 11 Linköping

Tel: 013-37 80 00
Fax: 013-37 81 00

www.foi.se

Alf Bengtsson, Lars Westerdahl

System av system - slutrapport

FOI-R--1830--SE
ISSN 1650-1942

Ledningssystem
Användarrapport

December 2005

Utgivare FOI - Totalförsvarets forskningsinstitut Ledningssystem Box 1165 581 11 Linköping	**Rapportnummer, ISRN** FOI-R--1830--SE	**Klassificering** Användarrapport
	Forskningsområde 4. Ledning, informationsteknik och sensorer	
	Månad, år December 2005	**Projektnummer** E7083
	Delområde 41 Ledning med samband och telekom och IT-system	
	Delområde 2	
Författare/redaktör Alf Bengtsson Lars Westerdahl	**Projektledare** Alf Bengtsson	
	Godkänd av Martin Rantzer	
	Uppdragsgivare/kundbeteckning FM	
	Tekniskt och/eller vetenskapligt ansvarig Jonas Hallberg	
Rapportens titel System av system - slutrapport		

Sammanfattning

Det framtida ledningssystemet för Försvarsmakten kommer sannolikt inte att bestå av ett homogent system utan av flera mindre och samverkande system. Lägger man där till att Försvarsmaktens ledningssystem skall kunna fungera tillsammans med andra system – civila och internationella – ökar kraven på interoperabilitet.

Att kunna samverka handlar inte enbart om att kunna kommunicera med andra system utan även att kunna utbyta och hantera säkerhetsinformation mellan system. Det handlar också om att kunna skapa tillit för de meddelanden som skickas mellan system och andra systems tjänster.

Projektet System av system har skapat en modell för att på ett säkert sätt spåra ett meddelande som rör sig mellan olika system. Den tekniska grunden är Web Services då tekniker och standarder därifrån har använts. Modellen etablerar förtroende dels för avsändaren av meddelandet, men även för de i svaret medverkande Web Services.

Den här rapporten är en sammanställning av de delrapporter som producerats under projektets tre år.

Nyckelord

System av system, säkerhet, Web Services

Övriga bibliografiska uppgifter	**Språk** Svenska
ISSN 1650-1942	**Antal sidor:** 32 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Issuing organization	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency Command and Control Systems P.O. Box 1165 SE-581 11 Linköping	FOI-R--1830--SE	User report
	Programme Areas 4. C4ISTAR	
	Month year December 2005	**Project no.** E7083
	Subcategories 41 C4I	
	Subcategories 2	

Author/s (editor/s)	Project manager
Alf Bengtsson Lars Westerdahl	Alf Bengtsson
	Approved by Martin Rantzer
	Sponsoring agency FM
	Scientifically and technically responsible Jonas Hallberg

Report title (In translation)

System of systems - final report

Abstract

The future command & control system for the Swedish Armed Forces will probably not be a single homogeneous system, but rather a collection of cooperating systems. Adding the need for the Armed Forces to cooperate with other systems – such as civilian or international - increases the demands for interoperability.

The ability to cooperate with other systems is not just about communication. It requires exchange of security information between different systems. It also involves establishing trust in messages coming from other systems, as well as for other systems services.

The project Systems of systems has created a model for securely transferring messages between systems. The technical basis is that of Web Services since techniques and standards have been collected from there. The model establishes trust in the sender of a message as well as for the participating Web Services.

This report is a summary of all the reports produced within the project during its three years.

Keywords

System of systems, security, Web Services

Further bibliographic information	Language Swedish

ISSN 1650-1942	Pages 32 p.
	Price acc. to pricelist

Innehåll

1 Inledning 7
2 Sammanfattande slutsatser 9
3 Varför System-av-system, förstudie 11
4 Resumé och slutsatser 15
4.1 Identitetsverifiering över systemgränser 15
4.1.1 Web Services 15
4.1.2 XKMS 18
4.1.3 SAML 19
4.1.4 Identitetsverifiering 19
4.2 Spårning vid samverkande Web Services 22
4.2.1 Scenario 22
4.2.2 Hierarkiskt träd 24
4.3 Implementation av spårning i kretsar av samverkande Web Services 26
4.3.1 Hotbild 26
4.3.2 Bibliotek och API:er 27
4.3.3 Implementerad modell vs. hotbild 28
4.3.4 Andra säkerhetsaspekter i modellen 29
5 Referenser 31
5.1 Litteraturreferenser 31
5.2 Webreferenser 31

Figurer

Figur 1. Protokollsnivåer inom Web Services. 16
Figur 2. Informationsflödet mellan aktörer i scenariot 23
Figur 3. Hierarkiskt träd av meddelanden. 25
Figur 4. Meddelandemodell med hierarkiskt träd. 25
Figur 5. Nod B döljer sig själv. 26
Figur 6. Nod B döljer nod A. 27
Figur 7. Java moduler. 28

1 Inledning

Den här rapporten är slutrapport för projektet ”System av system – säkerhetsfrågeställningar vid hopkopplingar”. Syftet med rapporten är att ge en kortfattad sammanfattning av de delprojekt som utgjort projektet under perioden 2003-2005.

Projektet har i huvudsak behandlat de tekniker som ligger till grund för koncepten inom Web Services. Det var främst inom affärsområdet Web Services utvecklades initialt, men efterhand har teknik och forskningssektorn kommit i kapp. I och med att Web Services från början främst är utvecklat mot affärstjänster, såsom e-handel, var det inte uppenbart om och i så fall hur teknologin kunde användas i militära system, till exempel det nya ledningssystemet. Projektets uppgift var således att utvärdera koncepten och tekniken runt Web Services för att se om dessa kunde utnyttjas i en militär miljö.

I kapitel 2 ges en kortfattade sammanfattning av de slutsatser som dragits i projektet. Kapitel 3 ger en översiktlig beskrivning av området ”System av system”. Avslutningsvis ger kapitel 4 en resumé av de delprojekt som ingått i projektet.

2 Sammanfattande slutsatser

Inom projektet definieras ett system som en autonom enhet med en egen policy. System av system definieras som flera system vilka samverkar men saknar gemensamma ägare och policys.

Projektet har studerat system av system på en tjänstenivå. Det innebär att vi främst studerat de tekniker och teknologier som möjliggör sammankoppling enligt definitionen av system av system.

Web Services med sina textbaserade XML-meddelanden har utgjort stommen i projektets arbete. XML är likt html i uppbyggnad men är avsett för att beskriva information för en applikation snarare än att vara ett presentationsspråk. Inom ramen för Web Services har ett flertal standarder och rekommendationer utvecklats. I och med att XML är beskrivande har de flesta funktioner från distribuerade system anpassats för Web Services. Det innebär att en XML-baserad standard har utvecklats för att kunna beskriva en funktion. Dessa standarder ska inte misstas för nya lösningar på gamla problem utan snarare ett sätt att beskriva hur en tjänst skall kunna användas av andra system och hur information kan överföras mellan olika system. Här ligger styrkan ur ett system av systemperspektiv. Att kunna beskriva, till exempel, säkerhetsinformation så att en användare från ett system kan utnyttja en tjänst i ett annat system utan att behöva vara lokal användare har stor betydelse för system av systemkonceptet.

Traditionell Web Services är utvecklad för att en mänsklig klient skall fråga efter en tjänst via en Web Service. Denna Web Service kan i sin tur kontakta ytterligare Web Services för att utföra den önskade tjänsten enligt ett klient-server maner. Det innebär i praktiken att en förfrågan låser upp delar av systemet och att en populär Web Service belastas hårt av att hålla reda på inkommande förfrågningar och inkommande svar på sända förfrågningar. Projektet har utvecklat en modell för att sända förfrågningar/meddelanden som asynkrona envägsmeddelanden samt att spåra medverkande Web Services. Modellen är även implementerad och testad.

Det finns flera utvecklingsverktyg och standarder för traditionell Web Services-utveckling. Det blev snabbt uppenbart att de verktyg som finns för standardiserad utveckling var mycket olämpliga för den kedja av signaturer vi ville använda oss av. Vill man skapa något som går lite utanför mönstret är det lämpligare att använda sig av de mer grundläggande bibliotek som finns att tillgå.

3 Varför System-av-system, förstudie

Projektet “System av system – säkerhetsfrågeställningar vid hopkopplingar” startade 2003. I projektplaneringen beskrevs mål och motiv för projektet enligt följande

I den målbild som beskriver målet för det framtida försvaret och dess ledningssystem återkommer kravet på flexibilitet på flera nivåer.

Dels finns kravet på flexibilitet med vid själva den tekniska systemutvecklingen. Det tekniska ledningssystemet måste byggas upp stegvis, så kallad evolutionär systemutveckling. Det totala systemet måste byggas av mindre och oberoende delsystem som var för sig kan bytas ut eller modifieras.

Dels finns kravet på flexibilitet vid användning av systemet. Man har krav på att delar skall kunna fungera självständigt utan beroende av centrala delar, att delsystem skall vara mobila, att man skall kunna koppla ihop delar med andra nationers system med mera. Systemet måste vara designat så att avknoppning och hopkoppling av delar möjliggörs.

Sammanfattningsvis innebär kraven på flexibilitet att designen måste bestå av "system av system". Detta gäller såväl den tekniska uppbyggnaden som systemet i stort. Detta leder i sin tur till ett flertal frågeställningar beträffande IT-säkerheten. Målet med projektet är att maximera den defensiva förmågan hos våra egna system - att få starkast möjliga skydd mot obehöriga intrång. Som delmål, inom huvudkonceptet "System av system", avgränsas olika delområden i olika etapper.

För att definiera vad vi menar med “System av system”, och för att avgränsa vår inriktning, inleddes projektet med en förstudie [Bengtsson et al., 2003]. Denna diskuterades i projektets referensgrupp. Den första uppgiften blev att konkretisera ”System av system”. Till exempel resonerade vi kring skillnaderna gentemot det vedertagna begreppet ”distribuerade system”.

Vår första avgränsning är att ”System av system” innebär att de ingående delsystemen skall vara helt kompletta och helt självförsörjande, och driftsläget i ett system skall inte vara beroende av driftsläget i ett annat system. Vid hopkoppling av system skall ingetdera systemet behöva ändras eller anpassas till det andra

systemet. Hopkopplingen skall ske via mäklare (broker, mediator, gateway …) där eventuell anpassning sker. Infrastrukturen inom varje delsystem skall vara fullständig.

Nästa steg vad gäller avgränsning rör vilken systemnivå som skall vara i fokus. Vi myntade två begrepp för två alternativa nivåer; koalitionsnivå respektive tjänstenivå. Den högre av dessa nivåer, koalitionsnivån, innefattar bland annat det svåra problemet att hantera (och inte minst beskriva) tilltro (eng. trust) mellan olika system eller organisationer. Tjänstenivån är mer av datorteknisk karaktär, hur man kan koppla ihop system utan att dessa måste förändras eller anpassas.

Koalitionsnivån – den skulle också kunna kallas federationsnivån – innebär att självständiga delsystem skall samverka och alltså utväxla information. Delsystemen är av samma karaktär och hanterar i huvudsak samma typ av information. Däremot skall delsystemen kunna ha olika sätt att beskriva sin respektive information. Till exempel kan det finnas olika sätt att beskriva delsystemens säkerhetspolicies. Den mäklare som finns mellan delsystemen skall kunna beskriva information, bland annat policys och andra regler, på ett sätt som kan tolkas och verifieras i alla delsystem.

Med begreppet tjänstenivå menar vi en starkare fokusering på datatekniska metoder att knyta ihop system. Detta ligger helt i linje med den skisserade arkitekturen i det nya FMLS, Försvarsmaktens Ledningssystem. Man ser hela FMLS som ett antal tjänster som skall kunna anropas av olika aktörer, inbegripet anrop av andra tjänster. Anropen måste på vanligt sätt regleras av säkerhetspolicyn – autentisering, auktorisering, tillgänglighet etcetera. Detta synsätt är också naturligt vid hopknytning av självständiga system. Den nämnda mäklaren kan ses som en typisk tjänst.

Tjänstebegreppet är ett ”innebegrepp” som flitigt används nästan synonymt med ett annat innebegrepp, Web Services (WS). Detta har växt fram ur det arbete, med gigantiskt kommersiellt intresse, som bedrivs för att få standarder för anrop till tjänster för e-handel. Det finns en oändlig mängd referenser till Web Services på Internet. En startpunkt är [OASIS].

En första fråga är vad som är skillnaden mellan Web Services och det man brukar kalla distribuerade system. Man har ju länge byggt system med delkomponenter som anropar varandra. Standard för anrop är till exempel DCOM i Windowsmiljö, Java RMI i Javamiljö och CORBA som är tänkt vara plattformsoberoende. Den viktigaste skillnaden

mellan Web Services och tidigare distribuerade system kan skrivas XML [W3Ca]. Alla meddelanden som finns i de olika kommunikationsprotokollen inom Web Services är strukturerade med hjälp av XML. XML ger en mycket mer öppen och enkel (men också ineffektivare) kodning än de andra standarderna. Därmed får man bättre förutsättningar för till exempel plattformsoberoende.

Vår förstudie om huvudinriktning inom ”System av system – säkerhetsfrågeställningar vid hopkopplingar” utmynnade i att vi, i samråd med referensgruppen, valde den mer datatekniskt inriktade tjänstenivån. Skälen till detta är framför allt två. Dels bedömde vi att arkitekturen som bygger på Web Services ligger helt i linje med arkitekturen inom FMLS. Dels innebär beskrivning och tolkning av policies vid koalitioner en tyngdpunkt på formella metoder och logik. Inom dessa områden måste vi lägga oproportionerligt stor del av tiden på kompetensutveckling.

4 Resumé och slutsatser

4.1 Identitetsverifiering över systemgränser

Förstudien resulterade i att Web Services valdes som avgränsning inom det generella området ”System av system”. Bidragande till valet var det gigantiska kommersiella intresse som finns inom området Web Services. Detta borgar för att utbudet av COTS-produkter blir stort, vilket i sin tur gör området intressant för användning inom FMLS.

Den första konkreta verksamheten inom projektet blev därför att genomlysa läget inom Web Services, med särskilt fokus på delar som har betydelse för säkerhetslösningar vid eventuell användning inom FMLS. Rapporten ”Identitetsverifiering över systemgränser” [Bengtsson et al., 2003] har denna inriktning. Den avhandlar alltså ingen egen forskning, utan rapporterar standardiseringsläget. Speciellt rapporteras om autentisering och identitetsverifiering.

4.1.1 Web Services

De två leverantörsoberoende sammanslutningar som har störst inverkan på utvecklingen av Web Services är *Organization for the Advancement of Structured Information Standards* [OASIS], samt *World Wide Web Consortium*, W3C, och då främst dess undergrupp med inriktning mot Web Services [W3Cb]. Av dessa arbetar W3C med den tekniska grundstandarden, som får samma status som övriga internetstandarder. OASIS inriktar sig mot tillämpningar, bland annat säkerhet i affärstransaktioner. Deras standardförslag är plattforms- och leverantörsoberoende, men har inte samma definitiva status som W3C. Utöver dessa två organisationer finns sammanslutningar med större leverantörsdominans. Deras rekommendationer blir därför än mindre definitiva.

Det finns lite skilda meningar om vad man egentligen lägger i begreppet Web Services. En definition, direkt citerad från [W3Cc] är

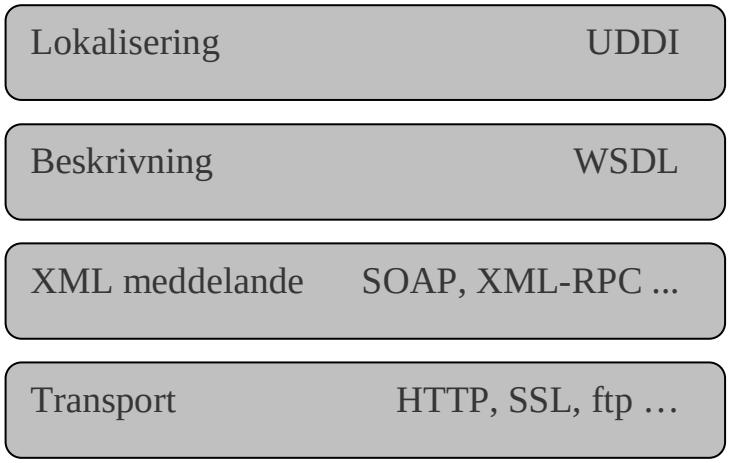
> *Definition: A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML based messages exchanged via internet-based protocols.*

En Web Service är en programvaruagent som skall kunna anropas av en annan Web Service (också i flera steg), eller annan programvara. Anropet skall ske via ett meddelande som skall vara uppbyggt med hjälp av XML (ofta enligt protokollet SOAP). Meddelanden skall överföras via något standardiserat internetprotokoll.

Vidare skall varje Web Service ha en globalt sett unik identitet – en URI, Uniform Resource Identifier. En Web Service skall kunna katalogiseras och lokaliseras (discovery) via en katalogtjänst (UDDI, Universal Description, Discovery and Integration) som själv är specificerad med hjälp av XML. Metoderna att anropa en Web Service, samt typerna av data som ingår i anrops- respektive svarsmeddelanden, skall beskrivas med hjälp av XML (WSDL).

Det som konstituerar Web Services är alltså

- En enhetlig standard, XML, för att bygga upp data och meddelanden. XML används också för att beskriva metadata, det vill säga beskrivningar av datatyper, objekttyper och -metoder, bindningar etcetera.
- Fyra stycken protokollsnivåer över TCP/IP-nivån enligt figur 1.



Figur 1. Protokollsnivåer inom Web Services.

Enligt definitionen är en Web Service tänkt att anropas från olika typer av agenter och andra program. Det bör dock understrykas att man i stort sett alltid har börjat med att utveckla modeller för den vanligaste typen av anrop, nämligen en person som via en webbläsare anropar en Web Service via http-protokollet. Utvecklingen av andra modeller har inte kommit lika långt.

I Figur 1 anges de fyra nivåer som konstituerar en Web Service. Den översta nivån är katalogtjänsten UDDI, Universal Description, Discovery and Integration [UDDI]. Denna är i sig själv en Web Service, det vill säga både innehållet i UDDI och kommunikationen till/från UDDI är beskrivet med hjälp av XML. UDDI beskrivs ofta som en katalogtjänst, liknande vita och gula sidorna. I en eventuell militär tillämpning, till exempel FMLS, känns det kanske inte naturligt att ha kataloger av typ gula sidorna. Men det finns vissa data som känns väsentliga, till exempel pekare till beskrivningar av Web Services.

Nästa nivå inom Web Service är beskrivningen WSDL, Web Service Definition Language. Detta är en XML-grammatik för att beskriva en Web Service. Tanken är att WSDL-beskrivningen skall vara så detaljerad och fullständig att den skall vara maskintolkbar av en klient som vill anropa Web Services. Klienten skall då kunna automatgenerera anropet, och automatiskt ta hand om svaret. Så långt har dock inte utvecklingen gått än. WSDL tjänar som en viktig beskrivning av gränssnittet för anrop av Web Services. Viktiga element i WSDL-beskrivningen är meddelandeformat, kommunikationsprotokoll, adresser etcetera.

SOAP är en standard för paketering av XML-meddelanden. Ett SOAP-meddelande skall, enligt reglerna för XML, bestå av ett enda element, ett `Envelope`. Inom detta kan det finnas ett element, `Header`, och det måste finnas ett element, `Body`. Inom såväl `Header` som `Body` får det finnas flera underelement. Elementet `Header` skall komma först. Däri läggs bland annat parametrar som rör data i `Body`, till exempel digitala signaturer.

De grundläggande säkerhetsfunktionerna inom Web Services är, liksom vid all informationshantering, den digitala signaturen och kryptering. Den digitala signaturen är användbar när det gäller identitetsverifiering och autentisering. Kryptering kan användas för åtkomstkontroll och för att säkerställa sekretess.

De grundläggande standarderna för Web Service-säkerhet anger inte tillvägagångssättet för signering respektive kryptering. De beskriver alltså inte algoritmer, nyckellängder, etcetera utan de beskriver hur man i XML-element paketerar dels de data som signerats respektive krypterats, dels anger vilka metoder man har använt. Det är väsentligt att detta görs på ett entydigt sätt så att konsumenten kan verifiera signaturen respektive dekryptera meddelandet.

Web Service-standarden för digital signatur är XML-Signature Syntax and Processing, [W3Cd]. Det intuitivt naturliga sättet att komplettera ett meddelande med en XML-Signature är via elementet `ds:Signature` med underelement som beskriver vilket, eller vilka, dataelement som har signerats och hur det gått till. Nedanstående är en skiss över element som kan eller måste finnas inom `ds:Signature`, hämtad från [W3Cd].

```
<Signature ID?>
   <SignedInfo>
      <CanonicalizationMethod/>
      <SignatureMethod/>
      (<Reference URI? >
          (<Transforms>)?
          <DigestMethod>
          <DigestValue>
      </Reference>)+
   </SignedInfo>
   <SignatureValue>
   (<KeyInfo>)?
   (<Object ID?>)*
</Signature>
```

4.1.2 XKMS

En naturlig, och viktig, Web Service rör hanteringen av certifikat och nycklar i system för PKI, Public Key Infrastructure. För att dölja skillnader mellan olika PKI-implementationer har man tagit fram standarden XKMS, XML Key Management Specification [W3Ce]. I denna specificeras en klients viktigaste anrop av ett PKI-system. Tanken är att olika PKI-system skall kunna anropas på samma sätt.

XKMS-standarden är indelad i två delar.

- XML Key Information Service Specification (X-KISS)
- XML Key Registration Service Specification (X-KRSS)

X-KISS stödjer förfrågningar rörande lokalisering och validering av publika nycklar. Förfrågningarna är av typen ”Vilka är kreditivhandlingarna för publika nycklar som jag skall använda för att kommunicera med X med hjälp av protokoll Y?” (lokalisering), respektive ”Är certifikatet C giltigt för användare X?” (validering).

X-KRSS specificerar via fyra olika tjänster en livscykelshantering av kreditiv för publika nycklar. Dessa fyra tjänster är registrering, återskapning, återutgivning respektive återkallande av kreditiv. Dessa är de delar av en PKI som har visat sig svårast att hantera. Detta återspeglas av att X-KRSS specificerar en begränsad mängd anrop.

Exempelvis specificeras återskapande enbart för PKI-genererade nyckelpar, inte för klientgenererade.

XKMS rör ett viktigt område. Det är dock iögonfallande att utvecklingen går långsamt. Inga kommersiella implementationer finns ännu. Anledningen till detta är oklar. Hela PKI-området verkar lida av att olika aktörer väntar ut varandra.

4.1.3 SAML

En annan viktig specifikation är SAML, Security Assertion Markup Language. Det är i första hand en specifikation över hur säkerhetsintyg, *assertions*, skall XML-formatteras så att de smidigt skall kunna hanteras av Web Services. Exempel på säkerhetsintyg är certifikat, Kerberos biljetter och auktorisationsbeslut.

I [Bengtsson et al., 2003] beskrev vi huvudsakligen SAML med avseende på autentisering. En vanlig missuppfattning rörande SAML är att det är en ny autentiseringsauktoritet. SAML är ett protokoll för autentisering med mera, men det ersätter inte existerande autentiseringsmekanismer, utan möjliggör en kommunikation mellan dem. En konsekvens av att använda SAML för att kommunicera säkerhetsinformation mellan två separata säkerhetsdomäner är att man kan bibehålla sina egna säkerhetslösningar, samtidigt som man kan nyttja utomstående resurser. System som använder Kerberos eller PKI bereds här en möjlighet att hantera varandras rutiner utan lokala överenskommelser.

Förutom själva formatspecifikationerna innehåller SAML ett antal use cases, profiler, som beskriver viktiga användarfall då SAML bör följas. Det mest utarbetade fallet är SSO, Single Sign On. Det finns tre olika profiler som beskriver hur man med SAML skickar intyg mellan Web Services om att en identitet har autentiserats.

4.1.4 Identitetsverifiering

Här sammanfattas tre olika lösningar på identitetshantering över systemgränser, nämligen Microsofts .NET Passport, Liberty Alliance och WS-Security. .NET Passport är en egenutvecklad produkt från Microsoft, medan Liberty Alliance och WS-Security är samarbete mellan flera företag. I Liberty Alliance:s fall var det Sun Microsystems som initierade arbetet, men nu är mer än 150 företag involverade. Även WS-Security är framtaget av samgående företag men de är betydligt färre jämfört med Liberty Alliance.

I grunden syftar alla tre systemen mot samma sak: att hantera webbaserad identifikation och autentisering. Lösningarna möjliggör för olika systemägare att förmedla och dela autentiseringsinformation. De huvudsakliga skillnaderna är att Microsoft .NET Passport är en produkt baserad på centraliserade servrar medan Liberty Alliance är en specifikation baserad på distribuerade servrar. WS-Security är ett stöd för att kunna skapa säkra SOAP-meddelanden.

Microsoft .NET Passport är ett centralt reglerat autentiseringssystem helt inriktat på människa-maskin. En användare har en global identitet vilken hanteras av en .NET Passport server. Anslutna tjänster har .NET Passport single sign-in (SSI) implementerat, vilket är ett program för att hantera den information som skickas via användarens webbläsare till .NET Passport servern och tillbaka igen.

Centrala tekniker för .NET Passport är HTTP redirect och cookies. En användare dirigeras om från den tjänst användare vill utnyttja till en .NET Passport server så fort autentisering, profilen eller plånboken (wallet) efterfrågas. Information avseende användaren skickas endast som krypterade cookies mellan .NET Passport och den aktuella tjänsten.

Stark kritik har riktats mot .NET Passport. Kritiken har främst handlat om centraliseringen av autentiseringsservrarna, plånboken (wallet) samt betydelsen av cookie:n. Centralisering av autentiseringstjänsten medför minskad kontroll både för användare och tjänsteförmedlare. Det medför också att systemet skulle drabbas hårdare om en av servrarna (i praktiken finns det mer än en server) skulle utsättas för en lyckad attack. Plånbokstjänsten bevisades 2001 vara mycket sårbar för en attack [Slemko, 2001].

Målsättningen med Liberty Alliance (vanligtvis bara kallad Liberty) skiljer sig inte från Microsofts – båda syftar till autentisering över skilda domäner. Liberty är dock en distribuerad lösning där ägandet av autentiseringsservrar inte är givet.

I Libertys specifikation finns det tre olika aktörer; Användare (principal), Identitetsförmedlare (Identification Provider) och Tjänsteförmedlare (Service Provider). Rollerna känns igen från SAML och det beror på att SAML är en viktig komponent. Liberty sätter dock in autentisering i ett sammanhang, till skillnad från den ursprungliga SAML specifikationen. SAML möjliggör autentisering över systemgränser medan Liberty är en komplett autentiseringsspecifikation.

Single Sign-On i lokala system är ofta skapade baserat på att en användare endast har en identitet. Microsoft har i Passport skapat en global identitet för varje enskild användare. Liberty har valt att låta användarna behålla sina lokala identiteter, det vill säga en användare kan ha skilda identiteter hos olika tjänster. Orsaken till detta beslut är underhåll av användarinformation. En användare uppger olika mängd information om sig själv beroende på vad tjänsten kräver och erbjuder. Genom att låta användaren själv kontrollera vilken information som förs vidare samt även själv kunna uppdatera information stärks användarens integritet samtidigt som repeterade uppdateringar undviks. För att kunna hantera olika lokala identiteter för samma användare utnyttjar Liberty vad man kallar identitetsfederation.

IBM, Microsoft, Verisign, BEA och RSA har gemensamt tagit fram en specifikation, WS-Security. Den handlar om hur man paketerar in krypterade och signerade dataelement, jämför XML Encryption respektive XML Signature ovan, i SOAP-meddelanden. Denna grund använder man sedan för att beskriva ett antal överliggande funktioner. En sådan funktion är WS-Federation, som i mångt och mycket handlar om samma frågeställningar som Liberty Alliance. Det är troligt att de sammansmälts till ett och samma koncept.

Slutsatsen i rapporten är att Web Services är en teknik i stark utveckling, men där också mycket återstår att utveckla. Beträffande autentisering över systemgränser har man kommit en bit på väg med autentisering mellan en mänsklig användare och en Web Service. Beträffande autentisering mellan Web Services är utvecklingen i sin linda.

Ett examensarbete [Heinonen & Manis, 2004] genomfördes inom området identitetshantering. Syftet var att få en egen uppfattning om hur väl standardisering inom Web Services avspeglas i olika programmeringsmiljöer. Därför gjordes examensarbetet som ett dubbelarbete, där ett scenario implementerades dels i Windows/.NET-miljö, dels i Linux/Java-miljö. Scenariots utformning var sådant att SAML hade varit användbart. De valda öppna programmerings-miljöerna understödde emellertid inte SAML, varför ett eget meddelandeformat användes. För övrigt bedrevs utvecklingen enligt Web Service metodik, till exempel användes från WSDL-filer. Slutsatsen blev att stöd finns i båda plattformarna för utveckling av vanliga funktioner. Det område som skapade mest interoperabilitetsbekymmer var certifikathantering.

4.2 Spårning vid samverkande Web Services

Slutorden ovan är att Web Services är en teknik i stark utveckling, men där också mycket återstår att utveckla. Beträffande autentisering över systemgränser har man kommit en bit på väg med autentisering mellan en mänsklig användare och en Web Service. Beträffande autentisering mellan Web Services är utvecklingen och standardiseringen i sin linda. Målet med vårt arbete under 2004, som redovisas i [Bengtsson, 2004], var att ta fram en metod att bygga upp historik och spårning när flera Web Services samverkar för att lösa en uppgift. Metoden möjliggör bland annat att det på ett säkert och oavvisligt sätt går att spåra identiteterna hos samtliga Web Services som samverkat, enligt en viss vald samverkansmodell. Egenskaper hos tre olika samverkansmodeller beskrevs – traditionell client/server, sammanhållna fråga/svar respektive asynkrona envägsmeddelanden. Den senare utvecklades vidare och en metod för historik och spårning av identiteter togs fram.

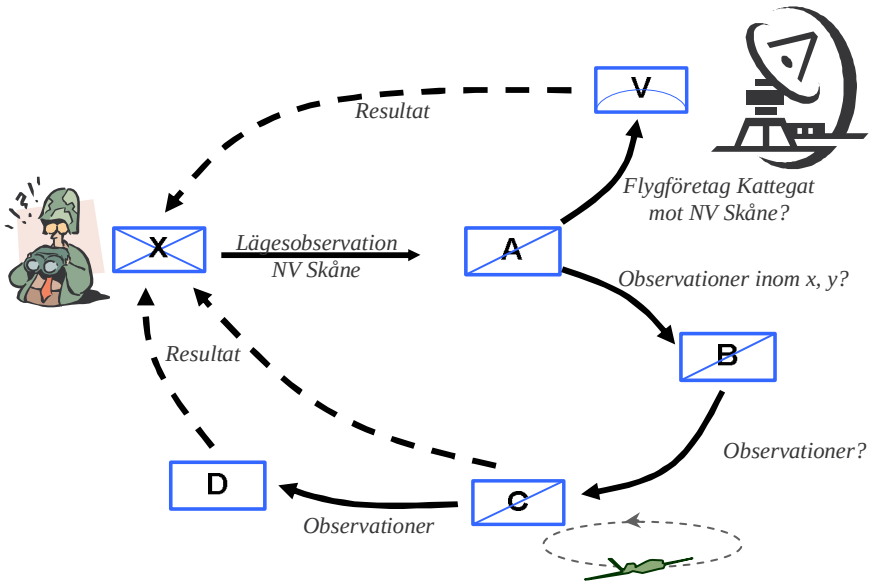
4.2.1 Scenario

Ett scenario, av typ informationssökning, målades upp enligt nedan. Scenariot har använts som illustration i flera lägen. En viktig observation är att scenariot, informationssökning, är av karaktären ”best effort”. Sådana modelleras ganska naturligt via asynkrona envägsmeddelanden. Scenarier av mer statisk karaktär, ”all or nothing”, modelleras naturligare på annat sätt.

Vi tänker oss att det finns ett stort antal klienter, en av dem kallas X, som vill ha uppgifter utförda. Det finns ett antal WS, färre än antalet klienter, som kallas A, B, C, ..., U, V. Låt oss anta att X vill ha utfört uppgiften ”Jag, X, behöver lägesbilden för NV-Skåne (angivet i koordinatform)”. Han ställer denna uppgift till A, som kan antas vara en WS som kan styrka X:s behörighet och som har kännedom om vilka tjänster som kan tänkas ge lägesbildsdata. A väljer att slussa ärendet vidare till V och B. V kan vara en luftövervakningscentral och får kanske meddelandet ”X, med adress si och så, behöver veta vilka flygföretag över Kattegat som har kurs mot NV-Skåne. Jag, A, styrker X:s behörighet”. B kan vara en spaningsresurs som kanske får frågan ”Vilka fientliga objekt finns inom området si och så? Meddela detta till X, vars behörighet jag styrker”. B kanske bedömer att han inte har resurser att besvara frågan, utan slussar den vidare till C. C tror sig veta att D har viss relevant information, och formulerar en fråga till D. Dessutom kan C själv ge information. Men det tar en viss tid, eftersom C först måste skicka ut en lokal spaningsresurs, innan C kan skicka sitt svarsbidrag till X.

Det är dessa identiteter – här kallade A, B, ..., V – som skall vara spårbara och omöjliga att manipulera. Särskilt viktigt är det att X, den som initierat uppgiften, kan verifiera identiteterna.

Scenariot kan illustreras med nedanstående figur, som återkommer senare i rapporten:



Figur 2. Informationsflödet mellan aktörer i scenariot.

Modellen innebär att varje Web Service skickar enbart envägsmeddelanden till efterföljande Web Service, respektive svar till X (streckade i figuren). Genom införandet av envägsmeddelanden kan varje Web Service göras helt tillståndslös. Varje Web Service bedömer själv vilka åtgärder som skall vidtagas, vem som skall kontaktas med nya meddelanden etcetera. När en Web Service har vidtagit åtgärderna kan den i princip glömma bort ärendet. Det är X som har uppgiften att hålla reda på ärendets tillstånd, vilka Web Services som har involverats, time-out med mera. Vissa lägen, när en Web Service behöver spara sitt eget tillstånd, kan modelleras som att den skickar ett meddelande till sig själv. Ett sådant exempel finns i figuren när C behöver vänta på sin lokala spaningsresurs.

Det bör påpekas att envägsmeddelanden lämpligen skickas via ett standard tvåvägs kommunikationsprotokoll, till exempel http. Man får

då direkt ifrån kommunikationslagret en kort kvittens på att meddelandet kommit fram.

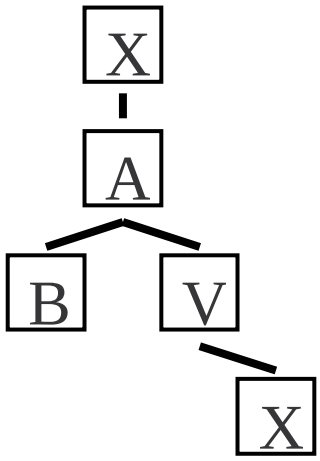
4.2.2 Hierarkiskt träd

Målet var att bygga upp en datastruktur, kodad i form av XML-element, som möjliggör spårbarhet av vilka Web Services som är involverade. Låt oss först mynta begreppet ärende. Ett ärende är allt som tillhör den uppgift som X vill att de samverkande Web Services skall lösa. Det är naturligt att datastrukturen skall innehålla ett ärendenummer, som skall vara en unik identifierare av ärendet. Det är ärendenumret som hjälper X att avgöra vilka svarsmeddelande som hör ihop, och som gör det möjligt för X att bygga upp det hierarkiska träd som avspeglar ärendets tillstånd.

Grundtanken är att datastrukturen följer med meddelandekedjan för ärendet. Varje involverad Web Service adderar underelement i datastrukturen, som därmed blir strikt växande. Varje Web Service signerar hela datastrukturen. Därigenom kan X rekursivt verifiera identiteterna för alla som varit involverade i meddelandekedjan.

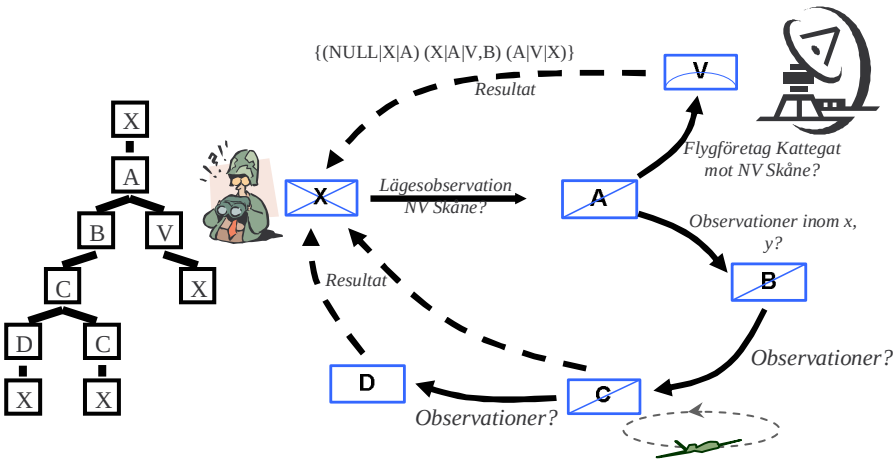
Eftersom målet är att åstadkomma en säker spårning av identiteter är det förstås identiteterna (X, A, B ... i vårt exempel, i verkligheten längre namn) som är de viktigaste elementen som varje Web Service adderar till datastrukturen. Varje nod adderar identiteten för den som anropat, för sig själv samt för de(n) som anropas. Med samma konventioner som i avsnitt 4.1.1 adderar varje Web Service elementen `<from>`, `<self>` och `<to>+`. I dessa element anges identitet för anropande Web Service, egen identitet, respektive identiteter för dem som kommer att anropas.

Delar av datastrukturen når X när X får svar från en Web Service (streckat i figur 2). X kan då bygga upp ärendets tillstånd i form av ett hierarkiskt träd. Figur 3 är ett exempel på trädets utseende baserat på den information som kommer via V till X. När X bygger upp detta träd syns det att det finns en oavslutad gren, via B. I detta läge kan X förvänta sig att det inkommer fler svar i ärendet. Kriteriet för att ärendet är avslutat är nämligen att alla löv i trädet är X. V signerar hela trädet, inklusive den signatur som A tillförde i föregående steg.



Figur 3. Hierarkiskt träd av meddelanden.

I och med att anroparen signerar vem som anropas (`<to>+`) och den anropade signerar vem som anropat (`<from>`) förhindras olika typer av man-in-the-middle attacker. Detta förstås under förutsättning att inte någon lyckats knäcka någon annans signeringsnyckel. Med en knäckt nyckel kan man alltid maskera sig. Inte heller kan en Web Service förneka sitt deltagande i kedjan. Man kan inte plocka bort sig själv, eftersom föregående Web Service har signerat mottagarna `<to>+`. Det går dock inte att hindra en Web Service från att dölja underaktiviteter. Vilken Web Service som helst kan ha frågat någon Q om information, utan att uppge detta. Likaså kan mycket väl C i exemplet ”glömma bort” anropet till sig själv.



Figur 4. Meddelandemodell med hierarkiskt träd.

4.3 Implementation av spårning i kretsar av samverkande Web Services

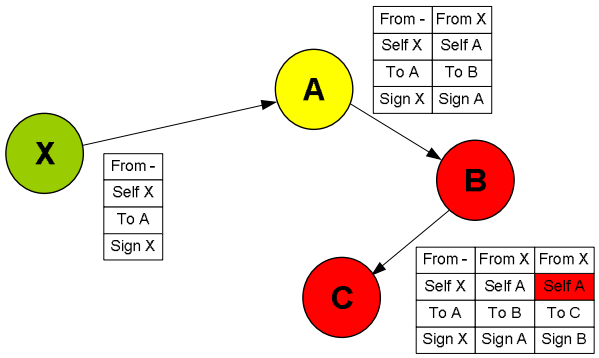
I [Heinonen & Manis, 2004] visades att det är förhållandevis enkelt att sätta upp Web Services även om man använder olika tekniska plattformar. De interoperabilitetsproblem som uppkom i samband med införandet av säkerhetsfunktioner gick att hantera med styrning av bland annat versioner av certifikat.

Den framtagna modellen för att spåra ett meddelande skiljer sig från de standarder som framkommit. De vanliga standarderna bygger på det traditionella antagandet att det är klienten som skall autentisera sig medan klienten förväntas lita på den tjänst som begärs. Även i vår modell skall klienten kunna styrka sin identitet men så skall även de ingående tjänsterna.

För att fastställa om tillgängliga bibliotek och API:er kan hantera vår modell gjordes en implementation av modellen [Bengtsson et al., 2005].

4.3.1 Hotbild

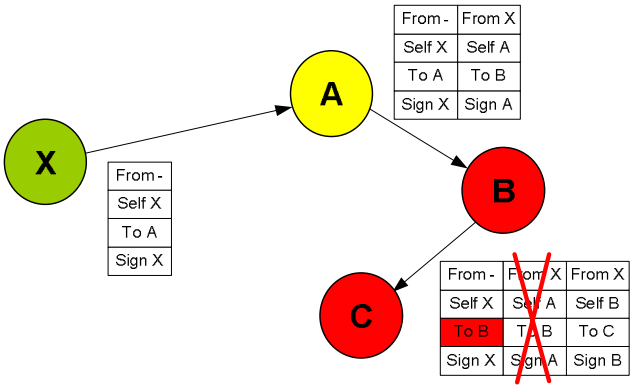
Syftet med implementationen var att visa att det även praktiskt inte var möjligt (så länge signeringsnycklarnas hemlighet är bevarade) att dölja en nod; antingen genom att dölja sin egen medverkan eller genom att dölja en annan nod. Två hotbilder togs fram för detta ändamål (figur 5 & 6).



Figur 5. Nod B döljer sig själv.

I figur 5 försöker nod B få nod C och därmed i slutändan klienten X att tro att nod B inte har medverkat i meddelandekedjan. Syftet kan

vara att föra in felaktig information och, i det här fallet, att få klienten X att tro att det är nod A som kommit med informationen.



Figur 6. Nod B döljer nod A.

Det kan också vara så att nod B vill reducera intrycket av andra noders medverkan. Genom att ange sig själv som avsändare redan på den initierande frågan och ta bort hela nod A:s signeringsblock försöker nod B visa för nod C att nod B blev direkt tillfrågad av klienten X. Syftet med detta kan till exempel vara att öka trovärdigheten för nod B.

4.3.2 Bibliotek och API:er

En målsättning var att bygga vidare på den Systinet Wasp server implementationen som skapades i samband med [Heinonen & Manis, 2004]. Tanken var att tillföra multipla signaturer till den ursprungliga koden. Det visade sig dock tämligen snabbt att detta skulle bli problematiskt.

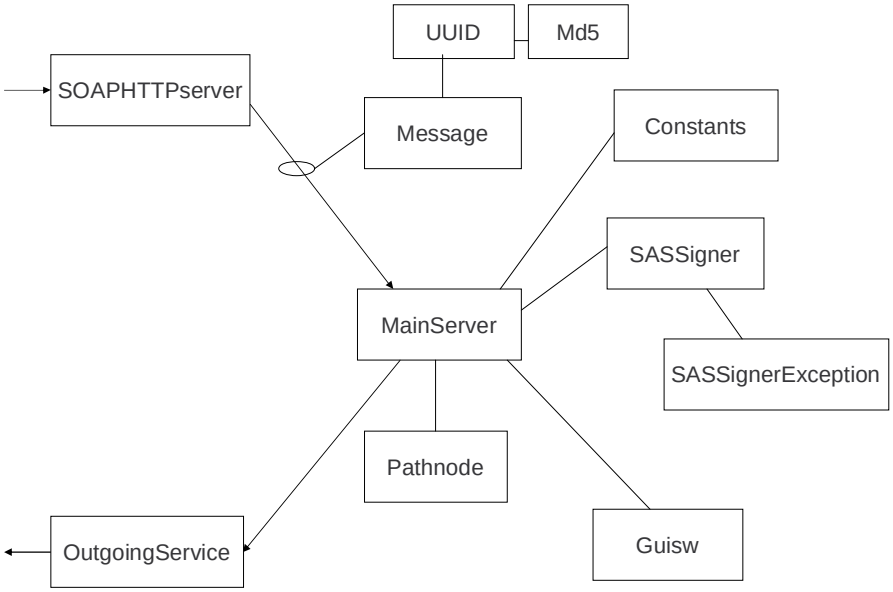
Wasp har tre abstraktionsnivåer för programmeraren att välja på; rå service, XML/SOAP service och java service. Varje nivå har sina egna klasser och objekt för att modellera data vilket medförde att det inte gick att flytta objekt mellan olika nivåer godtyckligt. En tänkbar lösning hade varit att använda samma nivå av funktionalitet för all programvara. Dock kan Wasp inte hantera multipla referenser eller signera flera referenser samtidigt.

Ett alternativ till Wasp som fortfarande bygger på WS-Security är Jakarta Tomcat Axis. Fördelen är att Axis stödjer multipla signaturer men nackdelen är att programmeraren har begränsad kontroll över var signaturen sätts in i meddelandet.

Det visade sig att WS-Security var för begränsat för att kunna ligga till grund för vår meddelandemodell. Andra lösningar var nödvändiga. Valet föll på Sun JWSDP (Java Web Services Development Pack). Sun JWSDP har flera utvecklade API:er för XMl, signering, DOM och SOAP.

Av smidighetssjäl skrevs en egen server. Både Wasp och Axix är relativt omfattande och innehåller många, i vårt fall, onödiga komponenter. En liten egenutvecklad server underlättar uppstart och minskar utnyttjandet av systemresurser.

En övergripande bild av implementationen med de ingående modulerna ges av figur 7.



Figur 7. Java moduler.

SASSigner är en nyckelmodul vilken både kan signera meddelanden och verifiera signaturer.

4.3.3 Implementerad modell vs. hotbild

I de hotbilder som beskrevs i figur 5 och 6 (kap. 4.3.1) inses enkelt hur signaturerna samarbetar för att säkerhetsställa integritet och konsistens. I figur 5 vill nod B dölja sin egen medverkan genom att påstå att meddelandet till nod C kommer direkt ifrån nod A. Nod B kan inte återskapa nod A:s signatur utan ändrar helt enkelt ursprunget

till nod X och sätter nod A som avsändare samt signerar blocket själv. På så sätt är alla signaturer korrekta men om man börja följa kedjan upptäcker man att den är felaktig.

Attacken i figur 6 är av en annan karaktär. Här vill nod B utesluta nod A helt och hållet genom att kasta nod A:s block samt sätta sig själv som mottagare från klienten (nod X). Här signaturerna och kedjan verifieras stämmer kedjan och nod B:s signatur, men signaturen för nod X är felaktig i och med att nod B har ändrat förutsättningarna.

Under implementationen upptäcktes även svagheter i identitetshanteringen. En signaturverifierare letar efter de attribut som refereras till i signaturen men kan inte avgöra om den har hittat rätt attribut. Detta möjliggör för en angripare att införa ett helt nytt block av XML-kod med korrekt referens och därefter modifiera den önskade referensen till något felaktigt. Lösningen på den här situationen är att inte bara verifiera signaturer utan även hela strukturen med ett XML-schema. Ett XML-schema upptäcker taggar som inte är definierade enligt dokumentstandarden.

4.3.4 Andra säkerhetsaspekter i modellen

Redan i scenariot (figur 2) introducerades en loop i systemet. I scenariot är loopen kontrollerad, det vill säga nod C loopar mot en instans (UAV:n) och skickar sedan ett resultat till nod X. En icke-kontrollerad loop skulle vara om nod C skickar en fråga till, till exempel, en nod K vilken i sin tur skickar tillbaka till nod C utan att kontrollera ursprung på meddelandet. Om både nod C och K skickar vidare meddelanden utan att kontrollera ursprung kommer meddelandet att skickas mellan C och K i oändlighet.

Den ursprungliga frågeställaren, nod X, kommer inte att märka av loopen på något annat sätt än att det inte kommer något svar från den delen av trädet där nod C ingår (om nod X vet att meddelandet har skickats till nod C).

Kontrollerade loopar kan hanteras enkelt med en loopräknare. Icke-planerade loopar kan man kontrollera genom att utnyttja http-lagrets OK continue/OK breake [Tanenbaum, 2003]. Detta är dock inte att rekommendera. Ett alternativ till http-acknowledge är att varje nod skickar ett vanligt tvåvägsmeddelande till nod X med förfrågan om att fortsätta eller avbryta (`Task ID=”uuid_tasknumber”continue/break`).

Sammanfattningsvis kan man säga att det finns möjligheter att bryta loopar men de behöver studeras mer så att man inte inför några beroenden vilka kan sätta en nod i en dålig situation.

5 Referenser

5.1 Litteraturreferenser

[Bengtsson, 2004] Bengtsson, A., *Spårning vid samverkande Web Services*, FOI-R--1399--SE, November 2004

[Bengtsson et al, 2003] Bengtsson, A., Hunstad, A. & Westerdahl, L., *Identitetsverifiering över systemgränser*, FOI-R--1025--SE, 2003

[Bengtsson et al, 2005] Bengtsson, A., Nordqvist, D., Persson, M. & Westerdahl, L., *Implementation of Tracing in a Circuit of Web Services*, FOI-R--1792--SE, 2005

[Heinonen & Manis, 2004] Heinonen, M. & Manis Sörensen, C., *Connecting Systems with Secure and Interoperable Web Services*, Linköping Institute of Technology, LiTH-ISY-EX3475-2004, 2004

[Tanenbaum, 2003] Tanenbaum A., *Computer Networks (4th edition)*, Prentice Hall, 2003

5.2 Webreferenser

[OASIS] Organization for the Advancement of Structured Information Standards
http://www.oasis-open.org/who/

[Slemko, 2001] Slemko, M., ”Microsoft Passport to Trouble”, 2001
http://alive.znep.com/~marcs/passport/, (6 november 2003)

[UDDI] Universal Description, Discovery and Integration protocol
http://www.uddi.org

[W3Ca] XML in 10 points
http://www.w3.org/XML/1999/XML-in-10-points.html, (13 november 2003)

[W3Cb] Web Services Activity
http://www.w3.org/2002/ws/, (13 november 2003)

[W3Cc] Web Services Architecture Requirements, Working Draft 19 August 2002
http://www.w3.org/TR/2002/WD-wsa-eqs-20020819, (13 november 2003)

[W3Cd] XML-Signature Syntax and Processing, W3C Recommendation, 12 februari 2002
http://www.w3.org/TR/xmldsig-core/, (13 november 2003)

[W3Ce] XML Key Management Specification (XKMS)
W3C Note 30 mars 2001
http://www.w3.org/TR/xkms/ (13 november 2003)

[WSDL] “Web Services Description Language (WSDL) Version 1.2 Part 1: Core Language”, W3C Working Draft, 11 June, 2003
http://www.w3.org/TR/2003/WD-wsdl12-20030611 (3 October, 2005)