Capítulo 1

Segurança em Serviços *Web*

Emerson Ribeiro de Mello, Michelle S. Wangham,
Joni da Silva Fraga, Edson Camargo
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
email:{emerson,wangham, fraga, camargo}@das.ufsc.br

Abstract

The use of open standards and integrative nature are features that made Web Services an interesting area to academic research and to industry. This chapter introduces the concepts behind the Service Oriented Architecture, Web Services, in particular. This text shows, through a use case, the benefits of this architecture and its security challenges. Afterwards, we present some research projects and technologies that deal with these security challenges.

Resumo

Devido a sua característica integradora e por fazer uso de padrões abertos, os Serviços Web se tornaram uma área de grande interesse para pesquisa e para a indústria. Neste capítulo, pretende-se introduzir ao leitor os conceitos da arquitetura orientada a serviços, e em particular, a sua mais atual caracterização, os Serviços Web. Será mostrado, através de um cenário de uso, os benefícios em utilizar tal tecnologia e também serão apresentados os desafios de segurança associados a esta. Por fim, são apresentados alguns trabalhos de pesquisa e tecnologias voltadas para tratar tais desafios de segurança.

1.1. Introdução

Há tempos que a Internet se consolidou como um importante veículo de comunicação e não demorou muito para se tornar uns dos principais meios para a realização de negócios. A Internet também é conhecida por agregar os mais diversos sistemas computacionais que variam desde a arquitetura de máquina, sistema operacional até os aplicativos finais aos usuários. O sucesso deste ambiente tão heterogêneo foi possível devido ao uso de protocolos padronizados, que garantem a interoperabilidade entre as aplicações, não importando em qual sistema operacional ou arquitetura de máquina esta esteja rodando.

A Internet surgiu diante de empresas que já faziam uso de seus sistemas computacionais e que, geralmente, não foram desenvolvidos para serem interoperáveis, por exemplo, com os sistemas computacionais de seus clientes, fornecedores, etc. Diante da necessidade da interação entre as aplicações distribuídas de diferentes organizações, uma nova caracterização de sistemas distribuídos surgiu possibilitando assim a troca de informações e a integração com os sistemas legados existentes - os **Serviços *Web***.

Os Serviços *Web* seguem uma Arquitetura Orientada a Serviços (AOS) e as principais características que os tornam uma tecnologia integradora e promissora são: (1) possuem um modelo fracamente acoplado e transparente que garante a interoperabilidade entre os serviços, sem que estes necessitem ter o conhecimento prévio de quais tecnologias estão presentes em cada lado da comunicação; (2) são auto-contidos e auto-descritivos; (3) usam padrões abertos como o HTTP e o XML, permitindo assim que aplicações sejam integradas através de linguagens e protocolos amplamente aceitos, e (4) tornam mais fácil a composição ou a combinação de diferentes provedores, visando formar serviços mais complexos e sofisticados.

Para a realização de negócios através da Internet, por onde circulam informações importantes para as corporações e, muitas vezes, sigilosas, garantir a segurança das informações é uma necessidade crítica. Com os Serviços *Web*, as aplicações tornam-se mais visíveis, expondo assim seus fluxos de negócio, processos e arquiteturas internas. Mecanismos de segurança estão sendo propostos para Serviços *Web*, porém, tais mecanismos ainda não contemplam todas as necessidades exigidas para segurança em Serviços *Web* e alguns são propostas iniciais que ainda não se consolidaram como um padrão de fato. Este cenário torna esta área um excelente ambiente para pesquisa.

O objetivo deste capítulo é analisar as questões de segurança provenientes da adoção de Serviços *Web*, discutir os principais padrões que visam minimizar as ameaças que esta tecnologia está suscetível, bem como apresentar algumas questões de pesquisa em aberto ou que estão sendo atualmente exploradas pelas instituições de pesquisa, que tratam da segurança em Serviços *Web*.

O presente capítulo está dividido em sete seções. Nesta primeira seção foi descrito o contexto geral em que o trabalho está inserido, destacando os objetivos do documento e a motivação para a escolha do tema. A seção 1.2 introduz os conceitos e as características da Arquitetura Orientada a Serviços e, em seguida, a seção 1.3 apresenta a arquitetura dos Serviços *Web*, com os padrões que formam a sua base. Os principais conceitos relacionados com segurança de informação, as questões de segurança em Serviços *Web* e as principais especificações que objetivam tratar parte destas questões são analisados na

seção 1.4. Na seção 1.5, as questões de segurança não tratadas nas especificações são discutidas e alguns trabalhos que visam tratar destas questões são analisados. As ferramentas que, atualmente, podem ser para o desenvolvimentos de Serviços *Web* seguros são apresentadas na seção 1.6. Por fim, a seção 1.7 apresenta a conclusão do capítulo, destacando os principais aspectos da segurança em Serviços Web que serviram e serve de motivação para pesquisas nesta área.

1.2. Arquitetura Orientada a Serviços

Segundo [Papazoglou 2003], a Arquitetura Orientada a Serviços (AOS) (*Service Oriented Architecture* – SOA) é uma caracterização de sistemas distribuídos, em que as funcionalidades do sistema são expostas via descrição de uma interface, permitindo a publicação, localização e a invocação por meio de um formato padronizado.

A AOS é constituída de relações entre três tipos de participantes: o *diretório para registro de serviços*, repositório que é utilizado para publicar e localizar as interfaces dos serviços; o *provedor de serviços*, entidade responsável por publicar as interfaces dos serviços providos por esta no registro de serviços e também responsável por atender as requisições originadas pelos clientes; e o *cliente*, aplicação ou um outro serviço que efetua requisições a um serviço. Cada participante da arquitetura pode ainda assumir um ou mais papéis, podendo ser por exemplo, um provedor e um cliente de serviços.

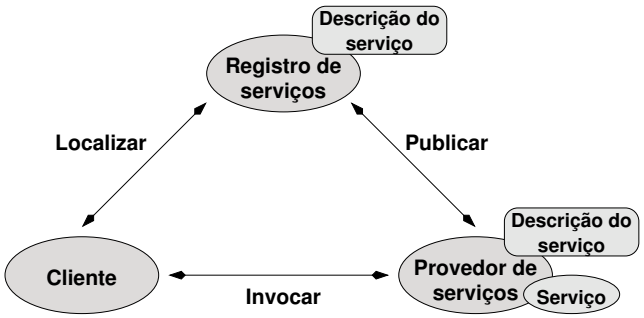


Figura 1.1. Interação entre entidades da AOS

Os participantes se relacionam através de três operações: *publicar*, *localizar* e *invocar*, como pode ser visto na Figura 1.1. Inicialmente, o provedor de serviço publica a interface do seu serviço junto ao diretório para registro de serviços. Desta forma, em algum momento posterior, o cliente pode efetuar uma busca por um determinado serviço (operação *localizar*), especificando as características desejadas, no diretório de registros. Se o serviço existir, a interface e a localização do respectivo serviço são retornados para o cliente. Por fim, o cliente efetua uma invocação ao provedor do serviço (operação *invocar*).

Os serviços estão baseados nas trocas de mensagens entre os provedores e os clientes, sendo assim, as mensagens seguem um formato padrão, garantindo aos serviços a neutralidade da tecnologia e permitindo que provedores e clientes utilizem diferentes implementações nas camadas inferiores. Os serviços também são definidos como *fracamente acoplados*, isso indica que possuem uma localização transparente e que não necessitam conhecer as estruturas internas presentes no lado do provedor e do cliente.

As interfaces dos serviços são auto-descritivas e baseadas em padrões abertos. Ou seja, a interface de um serviço define um conjunto de métodos públicos, juntamente com seus parâmetros, valores de retorno e meios para tratar possíveis exceções, porém não provê uma implementação. Com isto, pode-se assumir que a interface é um contrato entre o provedor do serviço e o cliente, sendo que o provedor deverá implementar todos os métodos ali descritos e o cliente só poderá invocar tais métodos.

Por estarem relacionados diretamente às funções de negócios, os serviços representam uma forma de modularidade diferente daquelas existentes nas linguagens de programação como os módulos, componentes e objetos. Componentes representam entidades e regras de negócio, já um serviço representa uma função de negócio completa, sendo composto por uma coleção de componentes. Serviços podem ser reutilizados e empregados em novas transações, na camada de negócios, dentro de uma organização ou através de organizações.

1.3. Arquitetura dos Serviços *Web*

Os Serviços *Web* (*Web Services* – WS) são classificados como um tipo específico de serviço, o qual é identificado através de um identificador uniforme de recursos (*Uniform Resource Identifier* – URI). Estes são independentes de linguagens de programação, de sistemas operacionais e das arquiteturas de máquinas. Através do uso de padrões abertos, como o XML e o HTTP, os Serviços *Web* conseguem garantir a interoperabilidade entre clientes e provedores de serviços, sem que os mesmos necessitem possuir o conhecimento prévio de quais tecnologias estão presentes em cada lado. Tal facilidade é ideal para que as relações de negócios entre empresas (*Business to Business* – B2B) sejam estabelecidas de maneira simples e dinâmica.

A definição para os Serviços *Web* dada em [W3C 2004a] é:

> *"Trata-se de uma aplicação identificada através de uma URI, que possui interfaces bem definidas e descritas em XML. As interações com outras aplicações se faz através de trocas de mensagens XML utilizando protocolos padrões da Internet."*

Um ponto importante a ressaltar é que os Serviços *Web* não são um outro tipo de objetos distribuídos, como aqueles presentes no CORBA[1], DCOM[2] e RMI[3] [OMG 2002, Brown e Kindel 1996, Sun 2002]. Em [Vogels 2003] é apresentada uma discussão sobre as semelhanças e diferenças entre Serviços *Web* e sistemas de objetos distribuídos. Para Vogels, os Serviços Web são um tipo de tecnologia de sistemas distribuídos que vem sendo utilizada em áreas em que as aplicações de objetos distribuídos falharam no passado.

As tecnologias de objetos distribuídos e de Serviços *Web* até possuem algumas características em comum, tais como: uma linguagem para descrição de interfaces (*Interface Definition Language* – IDL), que garante interações de rede bem definidas; e, mecanismos semelhantes para registro e localização de objetos ou serviços. Entretanto,

[1] *Common Object Request Broker Architecture*
[2] *Distributed Component Object Model*
[3] *Remote Method Invocation*

nos sistemas de objetos distribuídos, existe o conceito de *referência de objetos*, que não existe para os Serviços *Web*. A noção de *referência de objetos* é essencial dentro de um sistema de objetos distribuídos, visto que objetos, geralmente, possuem referências para outros objetos, possibilitando assim a computação com manutenção distribuída de estado . Todavia, a principal diferença entre os Serviços *Web* e os objetos distribuídos é o ciclo de vida dos mesmos. Um ciclo de vida de um objeto é composto pelas seguintes fases:

- diante de um pedido, uma fábrica cria uma instância de um objeto;
- o cliente que requisitou o pedido, executa operações no objeto instanciado;
- por fim, em algum momento posterior, o cliente remove a instância do objeto que não será mais utilizado.

Os Serviços *Web* não possuem um ciclo de vida com características como: objetos, referências e fábricas. Serviços *Web* não conseguem oferecer qualquer *facilidade para manter estado* na computação distribuída, característica básica de um sistema de objetos distribuídos. A arquitetura dos Serviços *Web* também não define relações entre as invocações realizadas em um mesmo serviço ou ainda em serviços relacionados, porém já estão sendo lançadas propostas para permitir tal interação, como a WS-Coordination [Cabrera et al. 2004].

Ambientes, como uma rede local, são caracterizados pela homogeneidade de plataforma e por possuírem um tempo de latência conhecido. Segundo [Vogels 2003], tal tipo de ambiente é ideal para a tecnologia de objetos distribuídos, visto que é uma tecnologia madura e, dentro de tal ambiente, bem robusta. Em ambientes como a Internet, em que a interoperabilidade e o suporte para plataformas e redes heterogêneas são essenciais, os Serviços *Web* demonstram ser os mais adequados.

A adoção dos Serviços *Web* não implica uso de qualquer aplicativo adicional no cliente ou no servidor. Para o cliente, basta uma linguagem de programação que dê suporte a XML e ao HTTP, por exemplo. Tal característica define os Serviços *Web* como *auto-contidos*. Serviços *Web* também são definidos como *auto-descritivos*, já que tanto o cliente como o servidor só precisam se preocupar com o formato e com o conteúdo das mensagens a serem trocadas, abstraindo assim os detalhes de implementação (fraco acoplamento). A arquitetura dos Serviços *Web* é composta basicamente por quatro elementos [Vogels 2003]:

- **serviço**: um aplicativo apto para processar documentos XML recebidos através de uma combinação de protocolos de transporte e de aplicação. Detalhes de como esse componente é construído, como técnicas de orientação a objetos, etc., não são especificados. O único requisito necessário para este tipo de componente, é que o mesmo esteja apto a tratar documentos XML;
- **endereço**: combinação entre protocolo e endereço de rede, utilizada para que um cliente possa acessar um serviço;
- **documento XML**: um documento que contém informações específicas à aplicação;

- **envelope**: encapsulamento que garante que documentos XML sejam processados de forma correta, separando as informações relacionadas a comunicação dos dados em si. Por exemplo, informações relacionadas a forma como a mensagem será cifrada ou assinada podem ser especificadas em um envelope sem que o documento XML original seja modificado.

Para tornar possível as três operações fundamentais de uma AOS - publicar, localizar e invocar - a arquitetura de Serviços *Web* adota as seguintes tecnologias baseadas em XML: a *Web Services Description Language* (WSDL)[W3C 2001], linguagem padrão usada para descrever as funcionalidades dos Serviços *Web*; o *Universal Description, Discovery and Integration* (UDDI) [OASIS 2004b], serviço padrão para publicação e localização de Serviços *Web*; e o SOAP [W3C 2003], protocolo usado para a invocação do serviço.

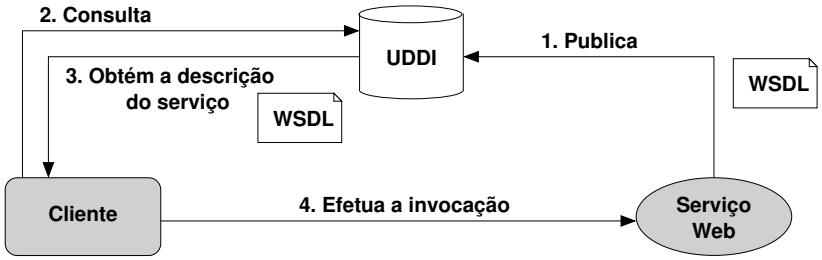


Figura 1.2. Colaboração típica na Arquitetura dos Serviços *Web*

A Figura 1.2 ilustra uma colaboração típica da arquitetura dos Serviços *Web*. O processo para tornar um Serviço *Web* publicamente disponível requer, inicialmente, que o provedor de serviços descreva a interface do serviço que deseja prover, utilizando a WSDL, e em seguida publique a interface em um serviço de busca público, como por exemplo o UDDI (passo 1 da Figura 1.2). A partir de então, o cliente pode localizar o serviço desejado e obter a sua WSDL (passos 2 e 3). A comunicação entre o provedor e o consumidor de um serviço é realizada através de trocas de mensagens XML, encapsuladas dentro de envelopes SOAP (passo 4).

A seguir, uma breve descrição das tecnologias empregadas na arquitetura dos Serviços *Web* é apresentada.

1.3.1. WSDL

A *Web Services Description Language* (WSDL) [W3C 2001] é uma gramática em XML, extensível, para especificar interfaces de Serviços *Web*. Um documento WSDL é independente de linguagem e de plataforma e tem por objetivo: (1) descrever quais são os serviços oferecidos; (2) mostrar como os clientes e provedores irão processar as requisições; e, (3) indicar em qual formato o serviço deve enviar as informações para um cliente.

Segundo [Weerawarana et al. 2005], um documento WSDL é composto por uma parte abstrata, que descreve o que o Serviço *Web* faz em termos de mensagem que este consome e produz, e por outra parte concreta que é referente a implementação e que define como e onde o serviços é oferecido. Os principais elementos XML presentes em um documento WSDL são:

- *types*: define os tipos de dados, utilizando o *XML Schema Definition* (XSD) ou ainda algum outro mecanismo para definição de tipos;
- *message*: define, de forma abstrata, as mensagens que serão trocadas;
- *operation*: define, de forma abstrata, a operação para uma mensagem;
- *portType*: descreve um conjunto abstrato de operações mapeadas para um ou mais serviços, os quais são descritos como pontos finais de rede ou portas;
- *binding*: especifica como mapear os elementos abstratos, *message* e *operation*, nos protocolos de rede que serão utilizados para transportar as mensagens até o destino (suas representações concretas);
- *port*: uma combinação entre o elemento *binding* e o endereço de rede, provendo assim um endereço único para acessar um serviço;
- *service*: declara o endereço das portas para os *bindings*. Ou seja, indica onde encontrar um serviço usando sua porta.

Os quatro primeiros elementos pertencem à parte abstrata da WSDL e os quatro últimos, à parte concreta. Cada um destes elementos pode ser descrito em diferentes documentos XML, e a combinação de todos estes formam uma descrição completa de um Serviço *Web*. A independência dos elementos principais traz uma grande flexibilidade para a disponibilização dos serviços. Uma mesma descrição de tipos de dados pode ser utilizada por diversos Serviços *Web* ou ainda múltiplos meios de transporte podem estar disponíveis para um serviço. A WSDL é uma linguagem flexível que também permite a inclusão de elementos não definidos pela especificação, possibilitando assim representar os atuais e os futuros formatos de mensagens.

1.3.2. UDDI

A especificação do *Universal Description, Discovery and Integration* (UDDI) [OASIS 2004b] define uma forma padronizada para publicação e descoberta de serviços dentro da Arquitetura Orientada a Serviço (AOS), que é parte fundamental na pilha de protocolos dos Serviços *Web*. A implementação de um servidor UDDI é composta por diversos Serviços *Web*, que provêem uma interface para que os clientes possam ter acesso as informações ali armazenadas. Os dados e meta-dados dos Serviços *Web* são armazenados em diretórios UDDI (*UDDI registry*), associando a cada estrutura de dados um identificador único, denominado *UDDI key*, criado de acordo com regras de classificação especificadas por cada organização. Tal classificação permite aos consumidores realizarem consultas mais refinadas, permitindo, por exemplo, buscar por provedores que forneçam determinado serviço dentro de uma localização geográfica específica.

Os diretórios UDDI não armazenam somente informações relativas a implementação de um Serviço *Web*, como a WSDL do serviço, estes também podem armazenar informações relacionadas diretamente a entidade que provê o serviço. O modelo de dados UDDI prevê os seguintes tipos de dados: *businessService*, descrições sobre a funções de negócio de um serviço; *businessEntity*, informações sobre a organização detentora do

serviço; *bindingTemplate*,informações técnicas do serviço, como por exemplo, endereço para invocação do mesmo; e, *tModel*, outros atributos, tais como taxonomia geográfica ou industrial. Nas especificações mais recentes do UDDI [OASIS 2002, OASIS 2004b], foram introduzidos dois novos tipos de dados voltados para a afiliação de registros, sendo estes: *publisherAssertion* e *subscription* [OASIS 2004a].

1.3.3. SOAP

Os Serviços *Web* adotaram diversas propostas para realizar trocas de mensagens entre clientes e provedores de serviços, mas foi o protocolo SOAP[4][W3C 2003] que surgiu como padrão de fato. Definido pelo consórcio W3C, o SOAP é um protocolo de comunicação baseado em XML para a troca de mensagens, independente de linguagem, que trabalha com diversos sistemas operacionais e sobre protocolos de aplicação já consolidados, como o HTTP, o SMTP, o FTP, o RMI/IIOP, etc.

O uso do SOAP sobre o protocolo HTTP se tornou comum nas atuais implementações de Serviços *Web* devido às facilidades providas pelo HTTP. Entre estas destacam-se: a infra-estrutura já existente dos servidores HTTP para disponibilizar os serviços e a facilidade em atravessar os limites de segurança impostos pelos *firewalls*, tendo em vista que o acesso à porta 80, utilizada por servidores HTTP, é geralmente liberada nestes mecanismos.

Uma mensagem SOAP é um documento XML que define o elemento ***envelope*** como sendo o elemento raiz do documento. O ***envelope*** SOAP contém as declarações dos espaços de nomes XML a serem utilizados, bem como as informações de codificação para a representação dos dados no documento e é composto pelos elementos ***header*** e ***body***. O ***header*** é um elemento opcional que contém informações, divididas em blocos, sobre como a mensagem deverá ser processada. Essas informações podem ser definições de roteamento, asserções de autenticação e autorização, entre outras. Já o elemento ***body*** é obrigatório e contém a mensagem em si. Qualquer tipo de informação que puder ser expressa em XML poderá fazer parte do corpo da mensagem. Dentro do elemento ***body*** pode estar contido o elemento ***fault***, que é usado para transportar informações sobre erros que possam vir a ocorrer no processamento das mensagens.

1.3.4. Uma Aplicação Exemplo

Um caso interessante para o uso dos Serviços *Web* é o de **portal de informações**. Um portal tem por objetivo agregar informações provenientes de diferentes origens em uma única e simples interface, se tornando um meio de fácil acesso para os usuários do sistema. Em [Wege 2002] são apresentadas algumas definições para portais, em que é possível destacar duas destas: os portais públicos e os portais corporativos. Os portais públicos são, geralmente, definidos por sítios que visam reunir informações de diferentes origens e aplicações, oferecendo uma interface padronizada e personalizável para seus usuários. Os portais corporativos também reúnem informações em uma interface padronizada, porém as informações ali reunidas só dizem respeito às necessidades da empresa em questão.

[4]Em sua criação, SOAP era um acrônimo para *Simple Object Access Protocol*, porém na versão 1.2 tal definição foi descartada pelo W3C, por achar que a mesma era equivocada. Assim, hoje SOAP é simplesmente o nome do protocolo e não mais um acrônimo.

Os portais de informações passaram por diversas evoluções desde o seu surgimento na década de 90, quando se resumiam em diretórios e máquinas de busca para catalogar sítios *Web*, até a mais atual versão que faz uso da tecnologia *Really Simple Syndication* (RSS) [RSS 2005]. O uso do RSS trouxe facilidades para provedores de informações e principalmente para os portais, pois apresenta uma forma padronizada e simples para disponibilizar resumos de informações. Tal tecnologia se constitui em um serviço de publicação e assinatura de notícias, porém não permite aos portais agregadores de notícias uma maior interatividade com seus usuários.

Os Serviços *Web* podem ser utilizados para a construção de um mesmo de tipo portal que hoje faz uso do RSS. Os provedores de serviços *web* disponibilizam uma interface de serviço padronizada para o fornecimento de informações para os portais e tais interfaces são publicadas em serviços como o UDDI. Desta forma, os clientes dos portais recorrem ao serviço UDDI para localizar as informações desejadas, assinando assim os respectivos serviços.Com os Serviços *Web* é possível obter um nível maior de interação entre todas as entidades participantes, seja um cliente interagindo com os provedores de serviços, ou seja estes últimos interagindo entre si.

O exemplo a ser apresentado neste capítulo consiste de um portal de informações voltado para o entretenimento. O objetivo do portal é reunir em uma única interface diversos provedores de serviços que tenham como área de atuação o entretenimento pessoal, como por exemplo, cinemas, parques de diversão, vídeo locadoras, teatros, etc. O portal também reunirá provedores de informações que não estão diretamente ligados ao entretenimento, mas que servem de base de apoio para a tomada de decisões dos usuários deste portal, como por exemplo, sítio de previsão do tempo, de resenhas de filmes, de companhias aéreas, de hotéis, de operadores de telefonia celular, etc.

Inicialmente, assim como a maioria dos serviços deste gênero, o portal exige um cadastro por parte de seus usuários, para que estes forneçam suas informações pessoais como nome, endereço, idade, sexo, etc. Após esta etapa, um usuário pode selecionar os provedores de serviços de sua preferência e assim configurar sua página pessoal no portal. Se for o caso, o usuário pode ainda indicar suas preferências para cada serviço selecionado. Por exemplo, no serviço de cinema, um usuário poderá informar os seus gêneros de filmes preferidos, o melhor dia da semana e horário, para ir ao cinema, etc. Em uma consulta, as informações retornadas podem ser ainda mais adequadas ao perfil do usuário se o serviço do cinema também souber a cidade e o bairro onde o usuário vive, sua faixa etária, podendo assim indicar ao usuário quais os cinemas mais próximos que estão exibindo os seus filmes prediletos.

Este cenário é totalmente possível de ser implementado quando se considera o ambiente dos Serviços *Web*. O serviço do cinema pode requisitar tais informações do usuário ao serviço de cadastro de usuários do portal, tendo em vista que ambos os serviços já possuem um acordo firmado para o compartilhamento de informação, acordo este que o usuário também possui ciência. A Figura 1.3 ilustra os relacionamentos entre os usuários, o portal e os provedores de serviços.

A listagem de filmes fornecidas pelo serviço do cinema já pode, por exemplo, vir acompanhada com a resenha de cada filme, informação esta obtida através da interação do serviço do cinema com o serviço de um sítio especializado em filmes. Enfim, com os

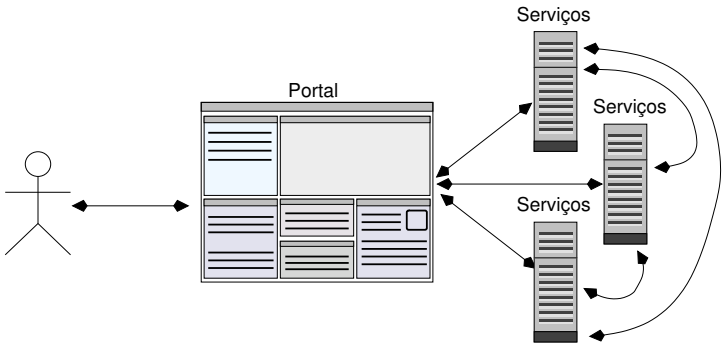


Figura 1.3. Portal de informações

Serviços *Web*, surge um novo tipo de interação, agora são aplicações se relacionando com aplicações sem a necessidade da intervenção dos usuários.

Nesta seção, preocupou-se em mostrar as facilidades que os Serviços *Web* podem trazer às aplicações de portais. Porém, sabe-se das inúmeras dificuldades e implicações associadas ao uso de Serviços *Web* neste domínio de aplicação. Este capítulo se resumirá em analisar os principais problemas relacionados à segurança computacional por trás desta aplicação de portal de informações.

1.4. Segurança em Serviços *Web*

Esta seção se inicia com uma breve introdução à segurança computacional, para que em seguida as questões e implicações de segurança relativas ao uso de Serviços Web possam ser adequadamente discutidas. Por fim, as principais especificações e trabalhos de segurança direcionados ao ambiente dos Serviços *Web* são descritos e analisados nesta seção.

1.4.1. Conceitos fundamentais de segurança

A segurança é vista como uma qualidade de serviço que garante o fornecimento do serviço, mesmo diante de ações de indivíduos não autorizados no sistema, sem que ocorram violações de segurança. Segundo [Russell e Gangeni 1991, Amoroso 1994], a segurança está fundamentada em quatro propriedades que devem ser garantidas:

- **confidencialidade**: a informação só deve ser revelada para usuários autorizados a acessá-la;
- **integridade**: a informação não poderá ser modificada, intencionalmente ou acidentalmente, por usuários que não possuam direito para tal;
- **disponibilidade**: o uso do sistema não poderá ser negado, de forma maliciosa, a usuários autorizados;
- **autenticidade**: o acesso ao sistema só deverá ser feito por usuários autênticos.

Algumas literaturas, como em [Landwehr 2001], também citam o ***não-repúdio*** como uma propriedade de segurança. O não-repúdio assegura que um usuário não poderá

negar a sua participação na ocorrência de um evento ou transação.

As violações de segurança ocorrem devido à exploração das vulnerabilidades existentes nos sistemas. As **vulnerabilidades** em sistemas computacionais sempre estiveram presentes. Um erro de programação, um erro de configuração ou mesmo um erro de operação, podem permitir que usuários não autorizados entrem no sistema ou mesmo que usuários autênticos executem ações não autorizadas, podendo assim comprometer o funcionamento correto do sistema [Bishop e Bailey 1996].

Uma **ameaça** consiste em uma possível ação que, se concretizada, poderá produzir efeitos indesejados ao sistema, comprometendo a confidencialidade, a integridade, a disponibilidade e/ou a autenticidade. Já o **ataque** é a concretização de uma **ameaça**, através da exploração de alguma **vulnerabilidade** do sistema, executado por algum intruso de forma maliciosa ou não. As quatro categorias de ataques, normalmente, identificados em sistemas distribuídos são:

- **interceptação**: uma parte não autorizada obtém acesso à informação (revelação não autorizada de informação);
- **interrupção**: o fluxo normal da mensagem é interrompida, impossibilitando que a informação chegue ao destino (negação de serviço);
- **modificação**: uma parte não autorizada modifica a informação recebida da origem e a transmite para o verdadeiro destino (modificação não autorizada da informação);
- **personificação**: entidade não autorizada transmite uma mensagem maliciosa pela rede, se passando por uma parte autêntica.

Em sistemas computacionais, as ameaças são constantes e uma maneira de evitar os ataques é identificar e corrigir as vulnerabilidades existentes nos sistemas, algo que não é tão simples quanto parece. Sistemas mais complexos tendem a possuir mais brechas de segurança, porém são nesses sistemas que a segurança é mais enfatizada, sabendo que o comprometimento desses sistemas gerariam enormes prejuízos financeiros.

A política de segurança de um sistema é um conjunto de diretrizes, normas e procedimentos, os quais estabelecem os limites de operação dos principais[5]. A política de segurança é concebida sob medida para um sistema específico, visto que cada sistema pode possuir diferentes necessidades. As diretrizes ditadas em uma política de segurança indicam o que cada componente do sistema (usuários, máquinas, etc) pode ou não pode fazer. As normas indicam o que cada componente está habilitado a fazer e como deverá ser feito.

As políticas de segurança de sistemas diferem em três ramos: a **segurança física**, objetiva proteger o meio físico em que opera o sistema (ex: imposição de restrições de acesso a determinadas áreas da empresas, medidas contra desastres, etc.); a **segurança**

[5]É universal que o termo **principal** identifique usuários, processos ou máquinas atuando em nome dos usuários de um sistema, que são considerados aptos pela política estabelecida (política de segurança lógica) em suas ações no sistema. Em contrapartida, usuários, processos e máquinas não autorizados pelas políticas são identificados como **intrusos**.

gerencial, que se ocupa com o ponto de vista organizacional, definindo processos para criação e manutenção das próprias políticas de segurança; e, a **segurança lógica**, que define quais usuários terão direitos de acesso ao sistema e quais os direitos que cada usuário possuirá.

1.4.2. Principais Questões de Segurança em Serviços *Web*

Os Serviços *Web* permitem que as aplicações se comuniquem sem a necessidade de qualquer tipo de interação com o usuário final. Voltando ao exemplo da Seção 1.3.4, o portal de entretenimento fornece aos seus usuários a opção para comprar ingressos para shows e ainda permite que o usuário utilize a mesma interface para comprar bilhetes aéreos e até mesmo para fazer a reserva em um hotel na cidade onde irá ocorrer o show. Neste caso, o usuário está interagindo diretamente com o portal e este, por sua vez, estaria interagindo com os demais sistemas, mediando assim a comunicação dos usuários com os sistemas da companhia aérea e do hotel. O problema aqui está em como garantir que as informações do usuário cheguem até o sistema da empresa aérea ou do hotel de forma segura, visto que as informações sensíveis do usuário, que só interessam a estes sistemas, estariam sendo roteadas e disponíveis pelo sistema do portal em si.

O roteamento entre múltiplos Serviços *Web* é comumente utilizado para obter escalabilidade e também para agir como uma ponte entre diferentes protocolos. Tecnologias como o TLS/SSL [Dierks e Allen 1999, Freier et al. 1996] permitem garantir a confidencialidade entre duas partes, porém não proporcionam segurança fim-a-fim, uma vez que a mensagem, para atingir o destinatário final, passa por diversos nós intermediários a nível de aplicação. Se a cifragem for empregada somente na camada de transporte, nós intermediários terão reveladas as informações que passam por eles, de forma proposital ou através das lacunas existentes entre uma sessão segura e outra.

As lacunas de segurança não ocorrem no transporte dos dados, mas sim quando os mesmos estão disponíveis nos nós intermediários. Assim, as informações confidenciais presentes nas mensagens SOAP, que deveriam permanecer confidenciais durante todo o percurso através dos nós SOAP intermediários, poderiam ficar expostas. Para tratar tal desafio, princípios de segurança devem ser aplicados em um contexto de segurança, que inclui muito mais que uma simples troca de mensagens SOAP. A Figura 1.4 ilustra diferentes contextos de segurança, em que o 1º contexto representa uma configuração ponto a ponto e o 2º uma configuração fim-a-fim.

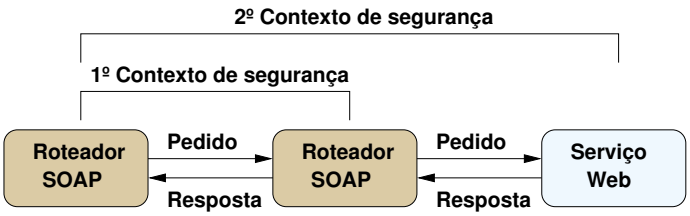


Figura 1.4. Contextos de segurança [IBM e Microsoft 2002]

Um outro desafio é como garantir os limites de segurança, antes determinados pelos *firewalls*. Os filtros de pacotes tradicionais se ocupam basicamente com a segurança na camada de rede, analisando se o pacote vem de uma origem confiável, porém não se

preocupa com o conteúdo dos pacotes. Assim, toda e qualquer requisição a um Serviço *Web* irá transpor o *firewall*. Os Serviços *Web* também estão suscetíveis a tipos de ataques já conhecidos como negação de serviço, mensagens antigas, estouro de pilha, entre outros, conforme apresentado nos trabalhos [Westbridge 2003, Demchenko et al. 2005]. Para garantir a segurança neste novo tipo de ambiente, novos mecanismos de segurança devem ser implantados também nas camadas superiores da pilha TCP/IP e devem operar em conjunto com os mecanismos presentes nas camadas inferiores (veja Figura 1.5).

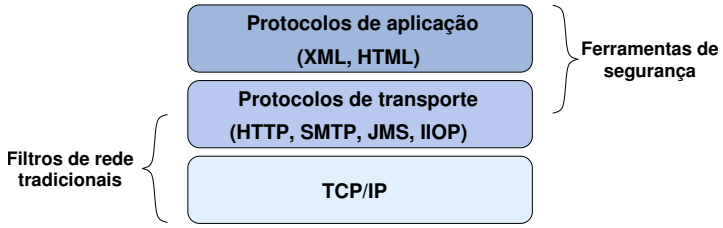


Figura 1.5. Segurança nas diferentes camadas

As interfaces dos Serviços *Web* são complexas e heterogêneas e é comum um Serviço *Web*, através de suas operações, acessar outros Serviços *Web*. A preocupação da negação de serviço que estava diretamente ligada ao sítio que hospeda o serviço, deve agora ser ampliada. Por exemplo, para uma instância de um Serviço *Web* é perfeitamente cabível atender 1000 requisições por segundo, porém essa instância pode fazer uso de serviço de terceiros, que para estes, o envio de mais de 10 requisições por segundo pode ser compreendido como um ataque de negação de serviço e assim interromper a comunicação.

Ao analisar o ambiente apresentado na aplicação exemplo da Seção 1.3.4, é possível notar que para a realização de um fluxo de negócio, a composição de diferentes Serviços *Web* se faz necessário. Quando se considera não mais um serviço único, mas sim uma orquestração ou coreografia de Serviços *Web*, constata-se que a segurança dos processos de negócios não foi ainda profundamente investigada e que ainda não existem soluções concretas e completas [Charfi e Mezini 2005].

Na aplicação exemplo, é dito que o usuário só precisa se cadastrar no portal, fornecendo suas informações pessoais, para que estas sejam propagadas por todos os serviços participantes. Como os limites administrativos precisam ser transpostos, as aplicações estarão sob diversos modelos administrativos e de segurança. Cada domínio transposto por um processo de negócio pode prover seu próprio conjunto de credenciais de segurança, tomando como base suas tecnologias subjacentes de segurança e suas políticas de segurança e de negócios. Por exemplo, em um determinado domínio, os usuários são autenticados através de um identificador único e senha; já em outro domínio parceiro, uma Infra-estrutura de Chave Pública (ICP) é usada para este fim. Para cada sistema, o cliente deverá possuir uma identidade e associada a esta diversos atributos de autorização.

Para o cliente o gerenciamento de tais informações pode se tornar muito custoso, visto que vai ter que gerenciar diferentes bases, fornecendo geralmente sempre as mesmas informações e ainda tendo que se preocupar em guardar os diferentes nomes de usuários e senhas. Já para os provedores de serviço, além da preocupação de ter que gerenciar inú-

meras identidades e credenciais, será necessário se preocupar também com a política de segurança que rege o sistema. Neste caso, a evolução das políticas é um fator que deve ser considerado, sabendo que uma política estática pode cobrir as necessidades de segurança de um determinado momento, porém pode já não conseguir suprir as necessidades que ainda estarão por vir. Assim, o provedor de serviço deverá considerar a necessidade da evolução de políticas, sabendo que tal ato não deverá desonrar as transações que estão em andamento, respeitando as políticas estabelecidas naquele momento.

Apesar de já existirem esforços para a definição de uma linguagem comum para expressar políticas, como a *WS-Policy* [WS-Policy 2004], ainda não existe nada padronizado para garantir a coesão destas políticas presentes em diferentes domínios. Em ambientes compostos por um número pequeno e conhecido de entidades, o gerenciamento das políticas de segurança não chega a ser um problema. Uma vez definida as políticas é possível que as mesmas continuem válidas por um longo período de tempo, visto que as entidades são conhecidas, bem como as tecnologias de segurança presentes no ambiente. Já os ambientes de larga escala, que são conhecidos pela sua dinâmica, apresentam o ingresso e egresso constante de entidades e ainda o uso de diferentes tecnologias de segurança. A gerência das políticas de segurança e a garantia de que as mesmas serão aplicadas são os grandes desafios em ambientes de larga escala.

Com o fluxo de negócios ultrapassando diversos domínios administrativos, a privacidade dos usuários também é um assunto que merece atenção. Em um cenário ideal, os usuários poderiam exercer o direito de determinar como suas informações serão manipuladas, indicando quais informações podem ser compartilhadas com terceiros, como esse compartilhamento deve ser feito e também indicando o período de tempo que essas informações podem ficar disponíveis nos sistemas. O projeto *Shibboleth* [Shibboleth 2005] apresenta uma preocupação com a privacidade das informações dos usuários, definindo como requisitos da arquitetura meios para gerenciar quais informações um sítio origem irá transferir para um sítio destino, com o consentimento do usuário.

Por fim, um dos pilares mais importantes para a construção de aplicações distribuídas e de processos de negócios é a confiança entre as entidades participantes. O termo confiança pode assumir diversos sentidos em uma aplicação distribuída. Em segurança, o mais usual é como garantir que as informações foram enviadas por uma origem confiável. No caso, a preocupação geralmente restringe-se a garantir as propriedades de autenticidade e integridade das mensagens. Para tratar tal problema diversos modelos foram propostos, como por exemplo, o X.509 [Housley et al. 2002], o PGP [Zimmerman 1994] e o SPKI/SDSI [Ellison et al. 1999, Rivest e Lampson 1996]. Porém, a confiança não se restringe simplesmente em garantir as propriedades de autenticidade e integridade. Em uma aplicação distribuída, as informações trocadas entre clientes e provedores de serviços possuem um certo valor e a manipulação indevida das mesmas pode acarretar em prejuízos para ambos os lados. Por exemplo, um cliente não gostaria de fornecer o número do cartão de crédito para qualquer provedor de serviço. A confiança entre clientes e provedores de serviço é algo que pode ser estabelecido com base, por exemplo, em uma base de reputações, o que poderia indicar que um determinado provedor de serviços sempre honrou suas comunicações.

Em alguns trabalhos, assume-se que o estabelecimento de confiança é um processo

manual que exige o cumprimento de diversos requisitos burocráticos antes da criação da relação de confiança. Por exemplo, para entrar na hierarquia das autoridades certificadoras do X.509 é necessário cumprir um conjunto de requisitos, sendo alguns destes relacionados a segurança física do local onde estará armazenada a chave privada da Autoridade Certificadora (AC). Outros trabalhos tratam a confiança de uma maneira mais dinâmica e volátil. Por exemplo, para um determinado fluxo de negócios é necessário que diversos provedores de serviço se agrupem e, uma vez que o fluxo tenha sido cumprido, tal relação é desfeita.

1.4.3. Especificações de Segurança para Serviços *Web*

Com o objetivo de tornar seguro o uso dos Serviços *Web* e assim garantir a sua ampla adoção, muitas propostas de segurança estão sendo submetidas a órgãos como: *World Wide Web Consortium* (W3C)[6] , *Organization for the Advancement of Structured Information Standards* (OASIS)[7] e *Web Services Interoperability Organization* (WS-I)[8]. As propostas visam cobrir diversas áreas de segurança e, em conjunto com as especificações de segurança para o padrão XML, estas permitem garantir alguns dos requisitos de segurança apontados na seção anterior.

XML Signature

O uso de assinaturas de digitais é uma forma para garantir as propriedades de integridade e autenticidade de informações digitais. A especificação *XML Signature* (XMLDSign) [Bartel et al. 2002], proposta conjunta entre W3C e IETF, define regras para gerar e validar assinaturas digitais expressas em XML. A XMLDSign possui pontos em comum com o *Public Key Cryptography Standard #7* (PKCS#7), porém apresenta formas para tratar os novos desafios em se trabalhar com documentos XML.

O desafio em criar assinaturas expressas em XML está justamente na forma de codificação dos documentos XML. Por exemplo, para interpretadores XML, o elemento `<Nome >` e o elemento `<Nome>` são tratados da mesma forma. Porém, quando aplicado um algoritmo para assinatura digital, duas assinaturas distintas seriam geradas.

A *XML Canonical* [Boyer 2001] define meios para representar documentos XML na forma canônica. Documentos XML, que sejam sintaticamente diferentes, porém logicamente equivalentes, serão representados por uma mesma forma canônica. Assim, o uso da forma canônica possibilita que os documentos XML possam ser assinados sem que haja preocupação com a sintaxe dos mesmos.

O uso da XMLDSign não está unicamente voltado para assinar documentos XML. É possível assinar qualquer tipo de documento eletrônico (arquivos binários ou textos), sendo que a assinatura será representada através de um documento XML. Também é possível assinar somente algumas partes de um documento XML, permitindo assim que outras partes de um documento XML sofram modificações, sem que isso invalide a parte assinada. A XMLDSign não define novos algoritmos criptográficos, mas faz uso dos

[6]http://www.w3.org
[7]http://www.oasis-open.org
[8]http://www.ws-i.org

algoritmos existentes, como o RSA [RSA 2002] e SHA-1 [Eastlake e Jones 2001]. As assinaturas podem ser representadas em três diferentes formas (veja Figura 1.6):

- ***enveloped***: a assinatura fica contida dentro do próprio documento XML a qual esta referencia. É ideal para ser utilizada com Serviços *Web*, inserida em mensagens SOAP;
- ***enveloping***: os dados assinados, em XML ou não ficam contidos dentro da própria estrutura do XMLDSig;
- ***detached signature***: a assinatura fica separada dos dados assinados. Isto é ideal para assinar documentos que não estão disponíveis localmente ou que sofrem constantes modificações.

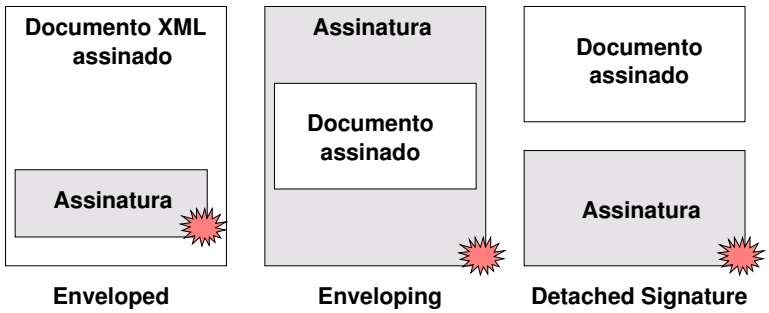


Figura 1.6. Formas de assinaturas XMLDSign

XML Encryption

A *XML Encryption* (XMLEnc) [Imamura et al. 2002] visa prover segurança fim-a-fim para aplicações que necessitem realizar troca de dados de forma segura. Diferentemente dos protocolos TLS/SSL [Dierks e Allen 1999, Freier et al. 1996], que só garantem a confidencialidade dos dados durante a sessão estabelecida entre duas partes, a XMLEnc garante confidencialidade persistente, garantindo assim a confidencialidade dos dados mesmo depois do término da sessão.

A XMLEnc provê soluções para algumas necessidades não cobertas pelo TLS/SSL, como a possibilidade de cifrar somente partes de um dado e o estabelecimento de sessões seguras entre mais de duas partes. Os dados cifrados são representados de uma forma estruturada e permitem que em um mesmo documento estejam presentes informações cifradas e não cifradas. Tal estrutura ainda possibilita o uso de diferentes chaves para cifrar partes de um documento, permitindo assim que um mesmo documento seja trocado entre diversas partes, sem que ocorra a revelação de informação para partes não autorizadas e garantam o acesso à informação, por partes autorizadas.

De forma análoga ao XMLDsig, o XMLEnc representa, de forma estruturada, dados cifrados e permite cifrar documentos XML ou não. A estrutura do XMLEnc, além de expressar os dados cifrados, também expressa detalhes sobre o tipo do documento

cifrado (jpeg, xml, etc.): a chave simétrica que será utilizada na sessão; informações sobre o tipo da chave simétrica; e o método de cifragem utilizado (ex: RSA para cifrar a chave secreta e AES [Daemen e Rijmen 2002] para cifrar os dados).

XACML

A autorização é uma propriedade básica de segurança que determina se um principal pode ou não executar alguma ação sobre algum recurso. Geralmente, cada sistema utiliza uma linguagem própria para definição das políticas, tornando assim um fator limitante para a concepção de sistemas distribuídos e abertos. Visando garantir a interoperabilidade entre os diversos sistemas, o órgão OASIS lançou a *eXtensible Access Control Markup Language* (XACML) [OASIS 2005a], um sistema de políticas de propósito geral, baseado em XML.

A XACML descreve uma linguagem para políticas de controle de acesso e também um formato para mensagens de *pedido* e *resposta*. A linguagem para política de controle de acesso é utilizada para definir quem possui direitos de acesso sobre o quê. O formato de *pedido* e *resposta* descreve como as consultas sobre o sistema de políticas deverão ser realizadas (pedido) e como deverão ser as respostas.

O formato de *pedido* e *resposta* define as trocas ente o *Policy Decision Point* (PDP) [Yavatkar et al. 2000], ponto este que efetua o processamento da política, e o *Policy Enforcement Point* (PEP) [Yavatkar et al. 2000], ponto este que concretiza as decisões de política. A XACML foi desenvolvida para garantir a interoperabilidade entre diversas aplicações. Assim, uma camada de abstração entre o ambiente da aplicação e a linguagem núcleo do XACML é feita através de um Contexto XACML. Um Contexto XACML é definido através de um esquema XML, que descreve uma representação canônica das entradas e saídas do PDP [OASIS 2005a].

Um pedido é composto: (1) por atributos associados ao sujeito que está originando a requisição; (2) pela identificação do recurso desejado; (3) pelas ações que serão executadas no recurso; e também (4) pelos atributos do ambiente. Já na resposta são contidas decisões como: *permit* – para acesso garantido; *deny* – para acesso negado; *not applicable* – para a inexistência de política ou de regras associadas ao recurso; ou ainda *indeterminate* – para a ocorrência de erros durante o processamento [Lorch et al. 2003].

A Figura 1.7 ilustra o fluxo de dados entre um cliente tentando acessar um recurso, utilizando-se do XACML. No passo 1 o sujeito (cliente) lança um pedido ao PEP, que monta um pedido XACML e encaminha ao Tratador de contexto (passo 2). O tratador de contexto encaminha o pedido para o PDP para que o mesmo decida sobre a tentativa de acesso (passo 3). O PDP pode requisitar ao Tratador de contexto atributos relacionados ao recurso e ao sujeito (passos 4, 5 e 6). De posse dos atributos, o PDP requisita as políticas associadas com as entidades envolvidas (passo 7) e assim gera uma resposta sobre a decisão tomada (passo 8). O Tratador de contexto gera uma resposta XACML e envia ao PEP (passo 9). E por fim, o PEP garante ou não o acesso ao recurso (passo 10).

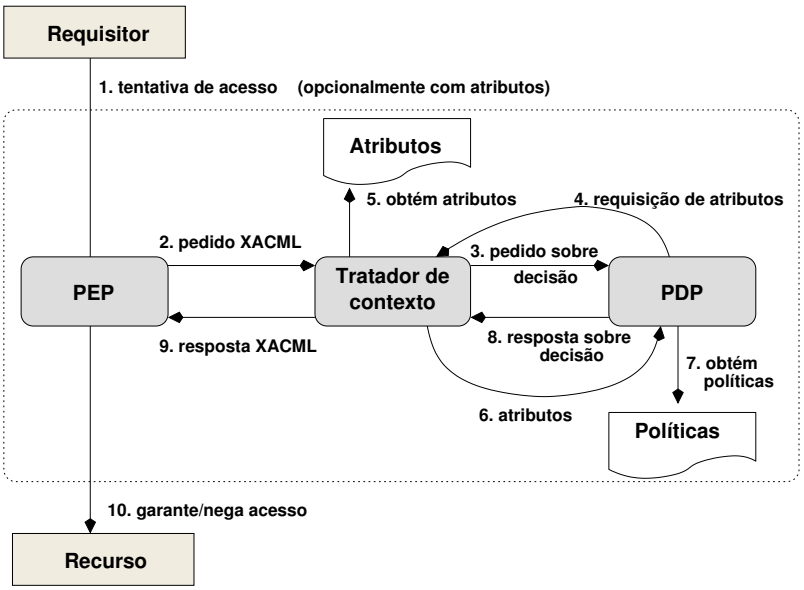


Figura 1.7. Fluxo de dados com o XACML [OASIS 2005a]

SAML

A *Security Assertion Markup Language* (SAML) [OASIS 2005c] consiste de um conjunto de especificações e esquemas XML, que juntos definem uma forma padrão para criar, trocar e interpretar asserções de segurança entre entidades de uma aplicação distribuída. No caso, são definidos meios para expressar, em XML, informações sobre autenticação, autorização e atributos de um sujeitos, porém as especificações da SAML não definem uma nova tecnologia ou forma para autenticação, mas sim uma tecnologia que visa garantir a interoperabilidade entre os diferentes sistemas de autenticação[9].

Uma *asserção de segurança* é um conjunto de afirmações, concedidas por um emissor SAML, sobre determinadas informações de um principal. Na especificações são definidas três tipos de asserções: de **autenticação**, fornecida pelo emissor SAML após o ato de autenticação com sucesso do usuário, que contém informações relacionadas ao emissor, o principal autenticado, o período de validade, etc.; de **atributo**, que contém detalhes específicos sobre o principal em questão, por exemplo, um papel que o principal desempenha dentro do sistema; e de **autorização**, que indica os direitos que um principal possui sobre um determinado recurso, sendo que esta asserção pode levar como base as asserções de autenticação e de atributos. Apesar do modelo de uso do SAML prever o uso de autoridades responsáveis pela emissão dessas asserções, as especificações não fazem qualquer menção sobre as mesmas. Todavia, as especificações definem os protocolos para que se possa interagir com essas autoridades.

Em sua primeira versão, o principal objetivo do SAML era permitir a transferência de autenticação e autorização entre aplicações *Web* (confiança portável). Já a versão

[9]A especificação da SAML prevê o uso de diferentes mecanismos para a autenticação: usuário e senha, Kerberos [Kohl e Neuman 1993], *Secure Remote Password* [Wu 1998], certificados TLS/SSL, chave pública (X.509 [Housley et al. 2002], SPKI [Ellison et al. 1999], XKMS [Hallam-Baker e Mysore 2005]), XMLDSign e ainda, possibilita o uso de mecanismos não definidos na especificação.

1.1 foi lançada com o intuito de melhorar a interoperabilidade e garantir uma melhor integração com o XMLDSign. Por fim, com base nas iniciativas do projetos *Liberty Alliance* (ver seção 1.5.1) e Internet2 Shibboleth [Carmody 2001], a versão 2.0 da SAML, recentemente lançada, tem como foco principal o uso de identidades federadas e ainda apresentando as seguintes características [OASIS 2005b]:

- **pseudônimos**: pseudônimos, ou identificadores opacos, permitem que principais interajam com o sistema sem a necessidade de revelar qualquer informação que o identifique, como e-mail, nome, etc. O uso de pseudônimos impede que provedores entrem em comum acordo para cruzar informações de um determinado principal e assim ferir sua privacidade;

- **gerenciamento de identificadores**: define como dois provedores poderão estabelecer e, em conseqüência, gerenciar os pseudônimos dos principais, com quem operam;

- **metadados**: estes definem como expressar dados de configuração e dados de confiança, para tornar mais simples o uso do padrão SAML, visto que as entidades participantes devem aceitar os mesmos papéis, identificadores, perfis, URL e certificados;

- **cifragem**: possibilita que atributos, identificadores ou toda a asserção seja cifrada. Tal característica permite garantir a confidencialidade fim-a-fim;

- **perfis de atributo**: estes simplificam a configuração e a implantação de sistemas que trocam dados de atributos. Definem como os atributos poderão ser transportados nas asserções SAML. Definem um perfil básico, que utiliza os tipos primitivos do XML para expressar os atributos e também define perfis como X.500/LDAP, UUID[10] e XACML;

- **manutenção da sessão**: o SAML 2.0 provê um protocolo que permite que todas as sessões, providas por uma autoridade de sessão, possam ser facilmente encerradas simultaneamente;

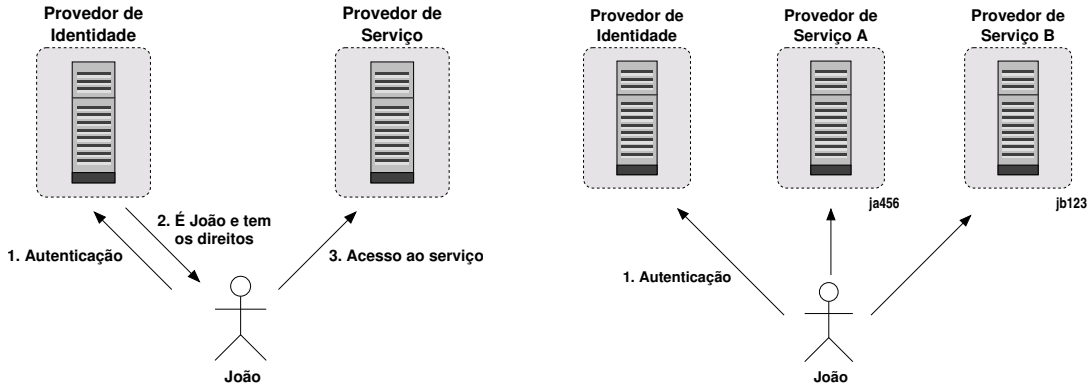
- **suporte a dispositivos móveis**: trata com as restrições de processamento dos dispositivos e com a largura de banda;

- **mecanismos de privacidade**:permitem expressar as configurações e políticas de privacidade dos provedores e principais, com relação ao uso da informação;

- **descoberta do provedor de identidade**: permite uma forma para localizar provedores de identidades, em ambientes em que existam mais de um provedor de identidade;

[10]Identificador único universal, definido pela *Open Software Foundation* (OSF) como parte do *Distributed Computing Environment* (DCE), uma vez criado por alguém, tem-se a garantia que o mesmo não será reutilizado por mais ninguém [OpenGroup 1997].

Nas versões 1.0 e 1.1 da SAML, o principal objetivo era transpor domínios através do uso da autenticação única (*Single Sign-On* – SSO), possibilitando que usuários autenticados em um domínio de segurança pudessem usufruir dessa autenticação em serviços presentes em outros domínios, sendo isto transparente para o usuário. Para isto, o conceito de identidade federada é utilizado. Neste caso, as entidades *Provedor de Identidades* e *Provedor de Serviços* entram em um acordo sobre os atributos dos usuários, como por exemplo, o nome do usuário e atributos de sessão, cabendo ao Provedor de Identidades garantir a autenticidade dos mesmos ao Provedor de Serviços. A Figura 1.8(a) ilustra um caso de identidade federada.



(a) Atributos federados (b) Ligação entre contas

Figura 1.8. Identidade federada

Com a SAML 2.0, surgiu uma nova forma de uso de identidade federada, que permite a "*ligação entre contas*". Neste caso, as diferentes identidades de um usuário, presentes em diferentes Provedores de Serviço, podem ser associadas de forma que possibilite o SSO, porém sem ferir a privacidade do usuário. A SAML 2.0 propõe o uso de pseudônimos, o que evita que Provedores de Serviços entrem em acordo, visando rastrear as informações de um determinado usuário.

No caso apresentado na Figura 1.8(b), o Provedor de Identidade estabeleceu diferentes pseudônimos com os Provedores de Serviços A e B, para referenciar um mesmo usuário, no caso João. Assim, João ao apresentar a asserção ao Provedor A, é reconhecido como o usuário local *ja456*; e ao apresentar a asserção ao Provedor B, é reconhecido como o usuário local *jb123*. Dessa forma, os provedores A e B não terão meios para rastrear o usuário João.

XKMS

Desenvolvida inicialmente pela VeriSign, em conjunto com a Microsoft e WebMethods, o padrão *XML Key Management Specification* (XKMS) [Hallam-Baker e Mysore 2005] é uma especificação aberta que define interfaces, baseadas em Serviços *Web*, visando retirar dos desenvolvedores de aplicações a complexidade em se trabalhar com Infra-estrutura de Chave Pública (ICP), podendo esta ser X.509, SPKI ou mesmo PGP [Zimmerman 1994]. A especificação é dividida em duas sub-especificações, *XML Key Information Service*

Specification (XKISS) e *XML Key Registration Service Specification* (XKRSS), que juntas definem meios para gerar pares de chaves, armazenar e localizar informações sobre chaves públicas, bem como para validar assinaturas.

A especificação XKISS define os serviços que visam retirar das aplicações a complexidade em se trabalhar com assinaturas expressas em XMLDSign [Bartel et al. 2002]. Informações como nome da chave, ou um certificado X.509, ou a própria chave, são descritas dentro do elemento XML `<ds:KeyInfo>` de uma assinatura em XMLDSign. Porém, a informação fornecida juntamente com a assinatura pode ser insuficiente para que o receptor possa validar a mesma, ou ainda, a informação pode estar em um formato no qual o receptor não é capaz de compreender. Por exemplo, dentro do elemento `<ds:KeyInfo>` só é fornecido um nome para a chave utilizada, mas não a própria chave em si. O XKISS define dois serviços: um para permitir a localização de informações relacionadas às chaves (*XKISS Locate*) e outro para verificar se estas informações relacionadas às chaves são válidas (*XKISS Validate*).

O objetivo do serviço *XKISS Locate* é de somente localizar informações relacionadas ao elemento `<ds:KeyInfo>`. Tais informações podem ser obtidas em uma base local de dados ou através do encaminhamento de um pedido a outros servidores. Por exemplo, dentro de um elemento `<ds:KeyInfo>` poderia estar contido somente o e-mail do criador da assinatura. Com essa informação, o *XKISS Locate* poderia localizar qual chave está associada com o e-mail e assim permitir que a assinatura seja validada. Porém as informações retornadas pelo *XKISS Locate* não são validadas, sendo tal tarefa atribuída ao serviço *XKISS Validate*. O *XKISS Validate* possibilita realizar as mesmas funções do *XKISS Locate*, porém o cliente pode obter uma asserção garantindo a validação das informações por este retornadas.

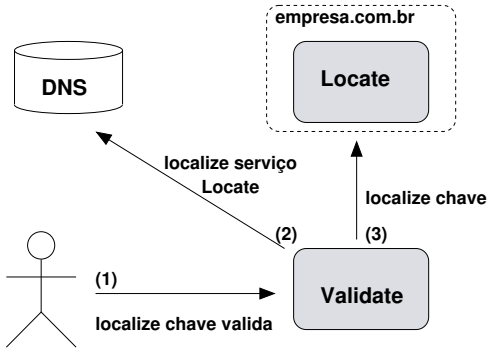


Figura 1.9. Uso combinado dos serviços *Locate* e *Validate* [Hallam-Baker e Mysore 2005]

A Figura 1.9 ilustra um exemplo, em que os serviços *XKISS Locate* e *Validate* são combinados com o intuito de localizar e validar uma assinatura. Neste exemplo, o usuário João aciona o seu serviço *XKISS Validate* para encontrar, de forma confiável, a chave pública do usuário `maria@empresa.com.br` (passo 1). No caso, o *XKISS Validate* utiliza o serviço de nomes (DNS) para localizar o serviço *XKISS Locate* responsável pelo domínio `empresa.com.br`[11] (passo 2). Por fim, o serviço *XKISS Validate* aciona o

[11] A especificação 2.0 do XKMS [Hallam-Baker e Mysore 2005] define meios para incluir informações do XKISS nos registros do DNS.

serviço *XKISS Locate* do domínio `empresa.com.br` para obter a chave pública do usuário *Maria* (passo 3).

Como visto, o objetivo dos serviços propostos na especificação XKISS é de localizar e validar as informações associadas com as chaves públicas, sendo que o registro e o gerenciamento destas informações estão dentro do contexto das facilidades providas pela especificação XKRSS. Tal especificação define serviços para: (1) o registro de informações; (2) a reemissão das informações associadas a chaves, permitindo gerar novas credenciais na ICP subjacente, por exemplo, no caso de um certificado expirar; (3) a revogação das informações associadas; e, (4) para a recuperação de uma chave privada, associada anteriormente. Neste último caso, só é possível recuperar a chave privada, somente se o par de chaves em questão foi gerado anteriormente pelo próprio XKRSS.

Cada protocolo definido pela XKMS provê suporte a diversas opções, incluindo opções de processamento das mensagens trocadas. Cabe ao cliente especificar quais opções este está apto a tratar e assim o serviço XKMS poderá decidir se aceita ou não o pedido, sendo que tal decisão dependerá de sua própria política de negócio. As opções para o processamento das mensagens podem ser: síncrona – o serviço responde ao pedido assim que o processamento for finalizado; assíncrona – o serviço pode não conseguir responder ao pedido imediatamente, mas notifica o cliente que o pedido ainda não foi satisfeito, posteriormente, o cliente poderá novamente invocar o serviço com o objetivo de obter a resposta final; pedidos de duas fases – diferente do assíncrono, nesta opção não há atraso entre o pedido inicial e o envio da resposta final. Tal forma de processamento é, basicamente, utilizada como um tipo de proteção contra ataques de negação de serviço.

A Figura 1.10 ilustra as opções de processamento previstas para o XKMS. No caso do processamento síncrono, o serviço, ao receber o pedido *P*, executa a operação requisitada e já envia uma resposta, composta pelo resultado e por um código (*Final*), o qual indica que a transação foi encerrada com sucesso.

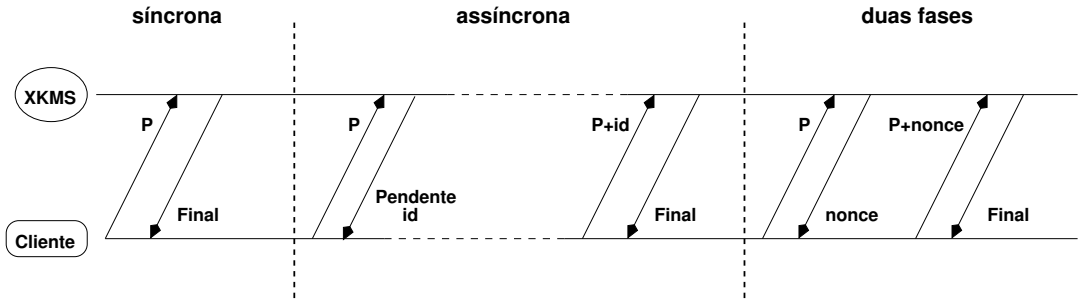


Figura 1.10. Tipos de processamento os pedidos XKMS

No processamento assíncrono, o cliente envia um pedido inicial *P* ao serviço para registrar uma chave, por exemplo. Segundo a política do serviço em questão, um pedido de registro de chaves requer a interação com o administrador do serviço, assim o pedido fica pendente até que o administrador aprove ou não o registro. Neste caso, o serviço responde o pedido inicial, feito pelo cliente, indicando que o processamento está pendente e fornece também um identificador para esta pendência (*id*). O cliente com este identificador em mãos pode, em um próximo pedido, verificar se o pedido inicial *P* já foi

processado e em caso afirmativo o serviço responde ao pedido *P* juntamente com o código *Final* para indicar o término da transação.

A política de segurança do serviço XKMS pode determinar que o serviço só aceitará requisições de clientes autenticados, estes já presentes na lista de controle de acesso do serviço, por exemplo. Dessa forma, todos os pedidos que chegam ao serviço devem ser assinados e cabe ao serviço verificar tais assinaturas. Sabe-se que as verificações de assinaturas digitais possuem um certo custo de processamento e intrusos poderiam inundar o serviço XKMS com requisições falsas, com o objetivo de provocar uma negação de serviço. O processamento de duas fases visa coibir tal tipo de ataque. Para isso, o serviço ao receber um pedido inicial *P* de um cliente (ver Figura 1.10) responde para este enviando um *nonce*[12] e indicando que o cliente deverá executar um novo pedido, juntamente com este *nonce*. O objetivo do *nonce* é garantir uma autenticação fraca, ou seja, o serviço só irá realmente verificar a assinatura de pedidos que estejam acompanhados de *nonces*, emitidos por ele. Assim, evita-se processar assinaturas de pedidos falsos. Pedidos, que forem novamente encaminhados com o *nonce*, serão processados e serão respondidos juntamente com o código *Final* para indicar o término da transação.

WS-Security

Proposta apresentada inicialmente pela IBM e Microsoft, a *WS-Security* [OASIS 2004c] é hoje uma especificação padronizada pela OASIS que tem como objetivo a proposição de extensões ao SOAP para permitir a construção de Serviços *Web* seguros. A especificação visa garantir a segurança fim-a-fim no nível de mensagem e não somente no nível de transporte, tendo três principais pontos:

- *credenciais de segurança*: incluir nas mensagens SOAP credenciais de segurança com informações de autenticação;
- *integridade da mensagem*: incluir nas mensagens SOAP informações relacionadas a assinaturas digitais de toda ou de parte da mensagem;
- *confidencialidade da mensagem*: mensagens SOAP podem ser cifradas, totalmente ou somente partes dela.

A WS-Security define um esquema XML, o qual possibilita incluir de forma padronizada as informações relacionadas a assinatura e a cifragem dos dados da mensagem SOAP em questão, fazendo uso das especificações XMLDSign [Bartel et al. 2002] e o XMLEnc [Imamura et al. 2002]. Tais informações são inseridas dentro de elementos XML `<wsse:Security>` e cada mensagem SOAP poderá conter um ou mais destes elementos. Isso se justifica devido ao fato que o caminho percorrido por uma mensagem SOAP, da origem até o destino final, pode ser composto por diversos nós SOAP intermediários. Neste caso, a WS-Security consegue garantir que somente determinadas partes de uma mensagem SOAP possam ser lidas, modificadas por determinados nós intermediários.

[12]Número pseudo-aleatório utilizado uma única vez (do inglês: *number used once*).

Dessa forma, a WS-Security permite a inclusão de múltiplas assinaturas e cifragens nas mensagens SOAP. Cada elemento `<wsse:Security>` deverá identificar, através do atributo `SOAP1.2:role`, o nó a qual aquela informação está direcionada. Não é permitido que haja dois elementos `<wsse:Security>` que tenham como alvo um mesmo nó SOAP, porém informações como a assinatura do emissor inicial podem ser interessantes para todos os nós SOAP intermediários ou final. Desta forma, é possível definir um único elemento `<wsse:Security>` sem que necessite indicar o nó relacionado com este elemento. Isso permite que todos os nós SOAP possam tratar tal elemento.

Cada nó intermediário só pode processar o elemento `<wsse:Security>` direcionado a ele, podendo assim removê-lo ou mesmo adicionar novos elementos `<wsse:Security>`, antes de encaminhar para o próximo nó, presente no caminho da mensagem SOAP. É possível também que cada nó intermediário adicione novos subelementos a um elemento `<wsse:Security>` já existente.

```
1  <soapenv:Envelope
2    xmlns:soapenv="..." xmlns:wsse="...">
3    <soapenv:Header>
4
5      <wsse:Security>
6        <wsse:UsernameToken wsu:Id="...">
7          <wsse:Username>joao</wsse:Username>
8        </wsse:UsernameToken>
9      </wsse:Security>
10
11   </soapenv:Header>
12   <soapenv:Body>
13     ...
14   </soapenv:Body>
15 </soapenv:Envelope>
```

Figura 1.11. Mensagem SOAP ilustrando o *WS-Security*

A Figura 1.11 apresenta um exemplo de uma mensagem SOAP com o cabeçalho da *WS-Security*. O exemplo consiste em enviar uma simples credencial, no caso "joão" (linha 7), sem qualquer tipo de proteção. Na linha 2 da figura, são informadas as URI[13] para os espaços de nomes XML do SOAP e da WS-Security. Cada elemento `<wsse:Security>` (linhas 5 a 9) pode expressar informações sobre a cifragem, a assinatura e sobre as credenciais de segurança. As linhas 6 a 8 expressam detalhes sobre uma credencial de segurança, porém os elementos `<wsse:Security>` podem conter mais de uma credencial de segurança, se desejado for.

Em um cenário de uso para a mensagem apresentada na Figura 1.11, um nó SOAP, após receber uma invocação de um cliente, autenticado através de um mecanismo presente nas camadas subjacentes (como TLS/SSL), encaminha tal mensagem para outro nó SOAP, sendo que ambos os nós estariam presentes dentro de um mesmo ambiente considerado confiável e seguro. Assim, o objetivo da mensagem é indicar ao nó SOAP final que, em um nó SOAP mais externo, a autenticação do cliente já foi realizada e esta informação está

[13] A URI foi suprimida para facilitar a visualização do código.

sendo repassada através do elemento `<wsse:Username>` (linha 7). Supõe-se também que a segurança da comunicação entre os nós SOAP é garantida através, por exemplo, do TLS/SSL. A Figura 1.12 ilustra tal cenário.

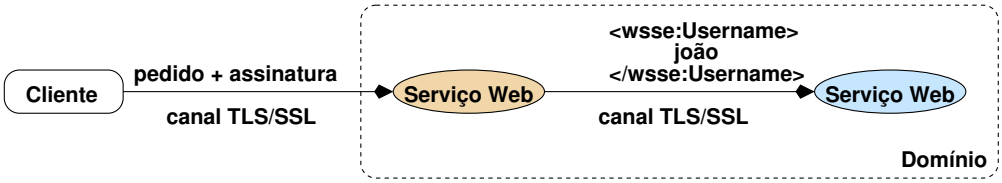


Figura 1.12. Encaminhando a identificação do cliente [Weerawarana et al. 2005]

No exemplo apresentado na Figura 1.12, a confidencialidade e a integridade das mensagens são garantidas através do uso do TLS/SSL, ou seja, no nível de transporte. Porém, pode-se ainda usar os padrões XMLEnc e do XMLDSign para garantir tais propriedades, evitando assim a necessidade do TLS/SSL. Outra forma ainda é a combinação do TLS/SSL com o XMLEnc e o XMLDSign.

Atualmente, a especificação *WS-Security* provê suporte a dois tipos de credenciais de segurança: credenciais `UsernameToken` e credenciais `BinarySecurityToken`. Um uso para a credencial `UsernameToken` foi descrito anteriormente e apresentado na Figura 1.12. Já a credencial `BinarySecurityToken` apresenta uma forma padrão para anexar a um pedido SOAP qualquer credencial de segurança codificada em forma binária; por exemplo, certificados X.509, *tickets* Kerberos, etc.

Políticas para os Serviços *Web*

Como visto anteriormente, a especificação WSDL surgiu da necessidade de um padrão para especificar as funcionalidades presentes em um Serviço *Web*. A WSDL permite aos provedores de serviços especificar quais são os serviços oferecidos, quais informações são necessárias para invocar um serviço e como deverá ser o formato para troca de informações com os clientes.

A WSDL só se preocupa em descrever as propriedades funcionais de um serviço, porém, é necessário uma forma padronizada e interoperável, para descrever as habilidades e requisitos não funcionais de um serviço. Tais habilidades não funcionais podem estar relacionadas com a segurança provida ou exigida pelo serviço. Os clientes, com base nessas informações, podem determinar qual serviço escolher, visando, por exemplo, serviços que apresentem uma política de privacidade bem definida, ou ainda, que garantam confidencialidade nas transações.

Como representar e anexar tais informações aos serviços ou recursos dentro do ambiente dos Serviços *Web*, serviu de motivação para o surgimento de documentos como o *WS-Policy* [WS-Policy 2004] e o *WS-PolicyAttachment* [WS-PolicyAttachment 2004]. A especificação *WS-Policy* provê um modelo de propósito geral para descrever políticas. Provê uma gramática flexível e extensível que permite descrever uma ampla variedade de requisitos e habilidades para o ambiente dos Serviços *Web*. A especificação *WS-PolicyAttachment* descreve como associar políticas com os determinados recursos e

também define como associar políticas a elementos XML que compõem um documento WSDL e a elementos UDDI.

Juntamente com o WSDL, o *WS-Policy* provê uma descrição declarativa dos requisitos que o serviço possui, os quais deverão ser cumpridos pelos requisitante. Porém, o uso de políticas não se limita somente aos serviços. Dentro do ambiente dos Serviços *Web*, existe uma ampla variedade de recursos, como documentos XML, sessões de mensagens confiáveis nas quais se podem associar políticas.

A especificação *WS-Policy* apresenta uma estrutura para descrição de políticas dividida em três principais componentes: *asserção de política* – expressa a habilidade do recurso, específica a um domínio, por exemplo, permitir a troca confiável de mensagens; *alternativas de políticas* – descrevem as combinações aceitáveis de obrigações e requisitos (conjunto de *asserções de política*), para a interação entre o serviço e um requisitor ou ainda para o acesso a um recurso; *política* – expressa um conjunto de alternativas de políticas válidas.

Segundo a *WS-Policy*, as *políticas* são representadas através de documentos XML, cujo elemento raiz do documento é o elemento `Policy`. Dentro deste elemento são representadas *coleções de asserções*, que quando combinadas representam um conjunto válido de *alternativas de políticas* . As *asserções* são combinadas através de dois tipos de *operadores de políticas*: `ExactlyOne` – indica que somente uma das *asserções* contidas na política poderá fazer parte de uma *alternativa de política*; `All` – permite a combinação de todas as *asserções* apresentadas como uma *alternativa de política*.

```
1 <wsp:Policy ...>
2   <wsp:ExactlyOne>
3     [ <wsp:All> [ <Assertion> ... </Assertion> ]* </wsp:All> ]*
4   </wsp:ExactlyOne>
5 </wsp:Policy>
```

Figura 1.13. Forma normal para expressar políticas [WS-Policy 2004]

Os operadores `ExactlyOne` e `All` podem ser combinados de diversas formas. Visando facilitar a interoperabilidade das políticas expressas pela *WS-Policy*, a especificação definiu uma *forma normal* para expressar as políticas. A Figura 1.13 apresenta a estrutura que uma política deve seguir para se adequar à *forma normal*. Dessa forma, cada alternativa de política válida fica contida dentro de um elemento `All` (linha 3) e todas as alternativas de políticas deverão estar contidas dentro de um único elemento `ExactlyOne` (linhas 2 a 4)[14]. Isso indica que só é possível escolher uma única alternativa e esta alternativa consiste na expressão completa de todas as asserções ali descritas [Weerawarana et al. 2005]. A especificação da *WS-Policy* também define um algoritmo para a tradução de qualquer expressão de política para a *forma normal*.

A Figura 1.14 ilustra uma política expressa de acordo com a *forma normal* da WS-Policy. No exemplo, a política indica que credenciais Kerberos ou X.509 podem ser utilizadas para prover a autenticação em um determinado recurso. As linhas 3 a 7 e 8 a

[14]O símbolo "*", de acordo com a notação do XML, indica a presença de 0 ou *mais* elementos.

12 apresentam duas *alternativas de política* e somente uma das duas alternativas poderá ser selecionada. Se a primeira for selecionada, indica que somente credenciais Kerberos serão aceitas e no caso de ser selecionada a segunda, então somente credenciais X.509 serão aceitas.

```
<wsp:Policy>
  <wsp:ExactlyOne>
    <wsp:All>
      <wsse:SecurityToken>
        <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
      </wsse:SecurityToken>
    </wsp:All>
    <wsp:All>
      <wsse:SecurityToken>
        <wsse:TokenType>wsse:X509v3</wsse:TokenType>
      </wsse:SecurityToken>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Figura 1.14. Uma política expressa de acordo com a WS-Policy [WS-Policy 2004]

Geralmente, o provedor de um Serviço *Web* expõe sua política com o objetivo de indicar sob quais condições irá prover seu serviço, ou seja, informa suas habilidades e seus requisitos. Um possível cliente, após analisar a política, pode decidir se está apto ou se deseja acessar o serviço ou não. A especificação da *WS-Policy* define somente uma gramática para expressar políticas, porém não especifica como associar tais políticas aos Serviços *Web* ou mesmo como divulgá-las, permitindo que outras especificações determinem como associar políticas de acordo com uma tecnologia específica. Essa separação da definição das políticas com a associação aos recursos permite que as políticas possam ser reutilizadas.

A *WS-PolicyAttachment* [WS-PolicyAttachment 2004] define diversos mecanismos para associar as políticas aos recursos. Dentro do ambiente dos Serviços *Web*, os recursos poderão ser uma troca de mensagens, um serviço, uma coleção de serviços, etc. A *WS-PolicyAttachment* define mecanismos que permitem que as políticas sejam anexadas diretamente dentro de documentos XML ou ainda permitem associar as políticas com os recursos de forma que não necessitem que as políticas e os recursos estejam presentes dentro de um mesmo documento XML. Diversos tipos, presentes nos documentos WSDL, podem constituir um recurso, como os elementos *messages*, *portType*, *binding*, *service*, entre outros. As políticas podem ser anexadas aos documentos WSDL, através de elementos `PolicyReference`.

Também vale citar a proposta *WS-SecurityPolicy* [WS-SecurityPolicy 2005] que tem por objetivo descrever como deverá ser a segurança no nível de mensagens, utilizando para isso as especificações *WS-Security* [OASIS 2004c], *WS-Trust* [WS-Trust 2005] e *WS-SecureConversation* [WS-SecureConversation 2005].

WS-Trust

Desenvolvida por um conjunto de empresas, lideradas pela Microsoft e IBM, a especificação *WS-Trust* [WS-Trust 2005] define serviços e protocolos visando a troca de atributos de segurança (p.ex.: asserções SAML), para possibilitar a comunicação entre diferentes domínios administrativos e de segurança. A especificação WS-Trust apesar de ser bastante citada em trabalhos acadêmicos e aparecer em algumas ferramentas de desenvolvimento para Serviços *Web*, ainda não recebeu aval de entidades padronizadoras. Porém, recentemente o órgão OASIS criou um comitê técnico, *OASIS WS-SX TC*[15], que objetiva definir padrões para a troca confiável de mensagens SOAP e o trabalho envolvido consistirá em refinamentos das propostas WS-Trust, *WS-SecurityPolicy* [WS-SecurityPolicy 2005] e *WS-SecureConversation* [WS-SecureConversation 2005], trazendo assim importância para tais propostas, que brevemente poderão se tornar padrões de fato.

O serviço de atributos de segurança (*Security Token Service – STS*) é definido pela *WS-Trust* como a autoridade responsável por emitir, renovar e validar os atributos de segurança, sendo este a base do modelo de confiança. O STS consiste de um Serviço *Web* que implementa uma interface WSDL, especificada pela *WS-Trust*, e que processa mensagens SOAP seguras, ou seja, que está de acordo com a especificação *WS-Security* [OASIS 2004c]. A interface do STS define duas operações, a `RequestSecToken` para realizar o pedido e a `RequestSecTokenResp` para a obtenção dos atributos de segurança.

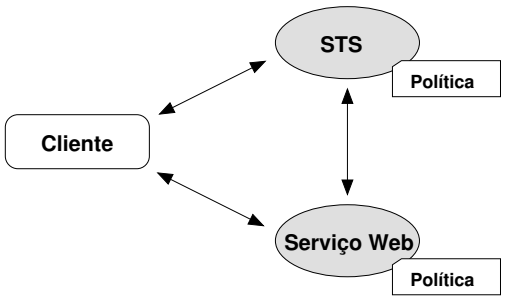


Figura 1.15. O uso do STS na mediação de confiança

A Figura 1.15 ilustra um caso típico de confiança mediada através do STS. O cliente deseja invocar o Serviço *Web*, porém de acordo com as políticas deste serviço (ex: expressas de acordo com a *WS-Policy*), é necessário que o cliente apresente credenciais de segurança emitidas pelo STS. Assim, o cliente deve obter as credenciais junto ao STS para que depois possa novamente invocar o serviço. É possível que o STS também exija algum tipo de autenticação do cliente.

Uma vez que o STS tenha analisado as credenciais fornecidas pelo cliente e verificado que as mesmas cumprem os requisitos necessários, o STS responde com a mensagem `RequestSecTokenResp`, que contém as credenciais necessárias para que o cliente consiga acessar o Serviço *Web*. A autenticidade da resposta pode ser garantida através de assinaturas digitais e a resposta ainda pode conter informações adicionais, como tempo

[15] `http://www.oasis-open.org/committees/ws-sx`

de vida da credencial e mecanismos para proteção contra ataque de mensagens antigas. Por fim, o cliente efetua um novo pedido ao Serviço *Web*, juntamente com as credenciais de segurança obtidas junto ao STS. O Serviço *Web* verifica as credenciais apresentadas e assim garante o acesso ao recurso.

A WS-Trust não se preocupa com as tecnologias de segurança subjacentes, deixando a escolha para o desenvolvedor da aplicação. O uso combinado das tecnologias de segurança presentes nas camadas de rede ou de transporte pode ser utilizado para garantir a autenticação do pedido em si. Por exemplo, o uso de assinatura de digital para provar que o cliente realmente é o detentor dos atributos apresentados ao serviço.

A especificação da WS-Trust não se preocupa com o estabelecimento das relações de confiança, mas usufrui das relações já estabelecidas, possibilitando assim que partes que não possuem relações estabelecidas possam se comunicar. As relações de confiança podem ser obtidas: (1) através de raízes fixas, em que é definido um conjunto fixo de entidades em quem se confia; (2) através de confiança hierárquica, em que a confiança é dada através de uma árvore e os nós inferiores confiam nos nós superiores; ou ainda, (3) através do uso das redes de confiança, em que cada entidade determina em quem confiar.

WS-Federation

A *WS-Federation* é uma proposta lançada pela Microsoft e IBM que apresenta meios para permitir que diferentes domínios de segurança possam criar federações de identidades e usufruir da mediação de confiança destas para permitir o compartilhamento de identidades, atributos e autenticação entre os participantes. Para isso, a *WS-Federation* faz uso dos modelos definidos nas especificações *WS-Security* [OASIS 2004c], *WS-Policy* [WS-Policy 2004] e *WS-Trust* [WS-Trust 2005], sendo que o STS, definido pela *WS-Trust*, assume também o papel de um *provedor de identidades*.

Ainda são definidas duas sub-especificações para a *WS-Federation*. A especificação *WS-Federation Passive Requestor* [WS-Federation 2003c] detalha como implementar as funcionalidades da federação em ambientes com clientes passivos. Os navegadores *Web* são considerados clientes passivos, pois não estão aptos a tratar as respostas SOAP enviadas por um Serviço *Web*. Desta forma, o processamento das mensagens deverá ter como base as funcionalidades do protocolo HTTP 1.1 (*GET*, *POST*, *redirects* e *cookies*). Já a especificação *WS-Federation Active Requestor* [WS-Federation 2003b] define como implementar as funcionalidades da federação em ambientes com clientes ativos. Clientes ativos são aqueles que estão aptos a emitir mensagens e reagir com as respostas de um Serviço *Web*, fazendo uso dos mecanismos definidos nas especificações *WS-Security*, *WS-Trust* e *WS-Federation*. Em geral, clientes ativos podem obter uma política, enviada através de uma mensagem de erro de um Serviço *Web*, processar essa política, obter as credenciais de segurança necessárias e refazer um novo pedido ao Serviço *Web* inicial.

1.5. Revisão da Literatura de Segurança em Serviços *Web*

Nesta seção serão apresentados alguns trabalhos que juntos cobrem diversas necessidades de segurança para a concepção de aplicações distribuídas, como o exemplo do portal de informações apresentado na seção 1.3.4. Os trabalhos estão focados em três dife-

rentes áreas, sendo alguns destinados aos problemas de gerenciamento de identidades, o que envolve questões de privacidade e anonimato; outros focados aos problemas de gerenciamento de políticas; e por fim alguns trabalhos voltados para o gerenciamento de confiança.

1.5.1. Gerenciamento de Identidades

Uma identidade digital consiste na representação de uma entidade em um domínio específico e geralmente está relacionada a domínios do mundo real. Uma entidade pode possuir múltiplas identidades, em que cada identidade é constituída por um conjunto de características, podendo estas serem únicas ou não a um domínio.

A identidade pode ser temporária ou permanente e pode assumir diferentes conotações, dependendo do contexto no qual esta se encontra. Para uma pessoa a identidade pode estar associada ao nome, endereço, documento de identidade. Já no contexto de uma empresa, a identidade pode estar associada com funções, privilégios, direitos e responsabilidades [Parr e Villars 2001].

O *gerenciamento de identidades* consiste de um sistema integrado de políticas, processos de negócios e tecnologias que permite às organizações controlar o acesso aos recursos providos aos seus usuários de forma segura, provendo confidencialidade às informações dos usuários. Diversos modelos foram propostos para o gerenciamento de identidades e em [Jøsang e Pope 2005, Jøsang et al. 2005] é apresentada uma breve descrição de alguns modelos.

O ***modelo tradicional*** de gerenciamento trata a identificação de forma isolada, sendo que o provedor de serviço também atua como o provedor de identidades e de credenciais (senhas associadas com os identificadores). Neste modelo, os usuários possuem identificadores únicos e específicos para cada serviço com o qual interagem, resultando assim em diferentes credenciais associadas com cada identificador.

O modelo de ***gerenciamento de identidades federadas*** surgiu para suprir as necessidades apresentadas pelo modelo de gerenciamento tradicional. Neste tipo de ambiente, é definido o conceito de **domínios**, nos quais estão presentes os provedores de serviço, de identidades e de credenciais, por exemplo, relacionados a uma determinada empresa. Assim, cada empresa constitui um domínio. O projeto *Liberty Alliance* (ver seção 1.5.1) e o projeto *Shibolleth* [Carmody 2001] são implementações abertas de modelos de gerenciamento de identidade federada.

No ambiente de identidades federadas, são estabelecidos acordos entre os domínios, os quais permitem que identidades locais a um domínio sejam reconhecidas nos demais domínios participantes do acordo. Neste caso, é estabelecido o mapeamento dos identificadores de um usuário em diferentes domínios. Por exemplo, o identificador `joao.pedro@empresa`, oriundo do domínio `empresa`, dentro do domínio `universidade`, será mapeado para o identificador `jp@universidade`. A federação de domínios de identificação dá a impressão aos usuários de possuírem um identificador único para todos os domínios que compõem a federação. Os usuários poderão continuar a manter identificadores locais a cada serviço ou mesmo domínio, porém o simples fato de possuírem tal identificador permite que estes usuários possam acessar serviços presentes

em qualquer domínio da federação.

No ***modelo centralizado*** de gerenciamento, considera-se a existência de um único provedor de identidades e de credenciais em uma federação, o qual é utilizado por todos os provedores de serviços da mesma. Neste modelo um usuário pode acessar todos os serviços presentes na federação utilizando um mesmo identificador. Em tese, o modelo se assemelha ao modelo de identidade federada, porém com a diferença de não necessitar do mapeamento de credenciais. A *WS-Federation* [WS-Federation 2003a] é um exemplo deste tipo de modelo.

[Damiani et al. 2003] apresenta um estudo sobre os problemas inerentes ao gerenciamento de múltiplas identidades, descrevendo os requisitos necessários que um sistema de gerenciamento de identidades deve atender. Dentre os requisitos apresentados, alguns estão diretamente preocupados com as necessidades de segurança dos clientes [W3C 2002, Rannenberg 2000, Asokan et al. 1997], como a privacidade, o anonimato, a responsabilidade, etc.

A seguir, serão apresentados alguns trabalhos que buscam atender estes requisitos dentro da arquitetura dos Serviços *Web*.

Privacidade

A especificação [W3C 2004a] apresenta algumas considerações sobre a privacidade na arquitetura dos Serviços *Web*, indicando que tal assunto ainda não está completamente solucionado e necessita de um estudo mais aprofundando. Em [W3C 2004b] são apresentados alguns requisitos para a arquitetura dos Serviços *Web*, necessários para garantir a proteção da privacidade dos clientes de um Serviço *Web*, sendo estes:

- a arquitetura deve permitir expressar políticas de privacidade sobre os Serviços *Web*;
- a política de privacidade de um Serviço *Web* deve ser expressa de acordo com a *Platform for Privacy Preferences* (P3P) [W3C 2002];
- a arquitetura deve prover um meio para que os clientes possam verificar as políticas de privacidade dos Serviços *Web*;
- a arquitetura deve permitir a propagação e a delegação da política de privacidade;
- os Serviços *Web* devem permitir interações onde uma ou mais partes são anônimas.

A *Platform for Privacy Preferences* (P3P) [W3C 2002] é um projeto do W3C que permite que os sítios *web* expressem suas políticas de privacidade de forma padronizada utilizando XML, dando aos usuários o conhecimento sobre como seus dados pessoais serão tratados.

A P3P provê ferramentas que permitem que o administrador de um sítio *web*, através de uma lista de requisitos, preencha como será a política de privacidade do sítio. Uma vez indicados todos os requisitos, a ferramenta retorna um código XML que pode ser facilmente associado a um sítio *web*. Navegadores *web*, compatíveis com a P3P, podem

obter a política dos sítios, comparar com as preferências de privacidade do usuário e decidir sobre a continuidade da transação com o sítio.

A P3P não define um padrão mínimo para garantir a privacidade e também não provê meios para monitorar se os sítios *web* estão honrando suas políticas. A P3P só define uma forma padrão para que sítios e clientes *web* possam informar e verificar as políticas de privacidade aplicadas naquele sítio *web*.

Segundo [Hung et al. 2004], o uso do P3P não pode ser diretamente aplicado no contexto dos Serviços *Web*, visto que o P3P foi projetado para que usuários de sítios *web* possam ter controle sobre suas informações pessoais. Outro problema é que os vocabulários da P3P estão direcionados principalmente para descrever as práticas de privacidade dos sítios *web*, sobre quais dados irão coletar dos usuários e o que irão fazer com essas informações. Dessa forma, é possível concluir que os requisitos apresentados em [W3C 2004b] ainda não atendem a real necessidade existente no ambiente dos Serviços *Web*, o que exige a criação de novas soluções para a área.

Anonimato

O anonimato é uma propriedade que está diretamente relacionada com a privacidade, porém com significado distinto. O acesso anônimo de um usuário a um sistema indica que o usuário não será identificado, garantindo assim a privacidade de sua identidade real.

Segundo [Cattaneo et al. 2004], em cenários onde os recursos podem ser acessados de forma anônima, porém por usuários autorizados pelas políticas de acesso do sistema, as soluções triviais poderiam seguir duas linhas:

- Restringir o acesso aos recursos com base no endereço de rede de um determinado domínio. Dessa forma, todos os usuários que estiverem dentro da rede, dita confiável, terão completo acesso ao recurso e de forma anônima. Porém, tal solução possui como pontos fracos a possibilidade de um usuário não autorizado conseguir acesso aos recursos pelo simples fato de estar na rede confiável, e negar o acesso a um usuário autorizado, pelo fato deste estar em uma rede não confiável.
- Fazer uso de credenciais de grupo. Por exemplo, o provedor do serviço poderia emitir certificados de grupo para todos os usuários autorizados, indicando que o usuário pertence ao grupo de usuários "confiáveis". Porém, um usuário, dito confiável, poderia ceder tal certificado a terceiros, comprometendo assim a segurança do sistema, sendo que em alguns casos fica impossível determinar qual dos usuários "confiáveis" delegou o certificado de grupo.

Em [Cattaneo et al. 2004] é apresentada uma extensão ao SOAP [W3C 2003] para permitir o acesso anônimo aos Serviços *Web*. A solução está baseada no fato de que os usuários só precisam provar, para um provedor de serviços, que pertencem a um determinado grupo, autorizado pelas políticas do sistema, evitando assim revelar sua identidade pessoal. A proposta dos autores consiste de uma variação do modelo para identificação

anônima de grupo introduzida em [Santis et al. 1998]. Este modelo apresenta um protocolo com *prova de conhecimento zero* (*zero-knowledge proof*) [Goldwasser et al. 1989] que permite um usuário se identificar, de forma anônima, para um sistema remoto.

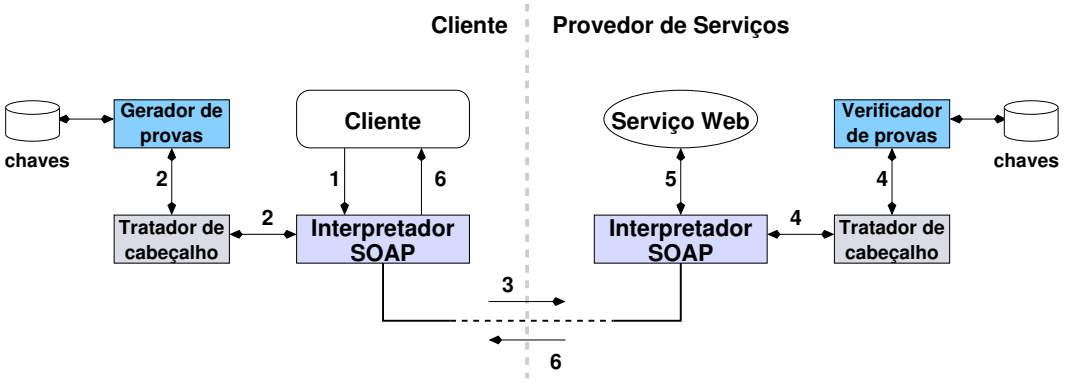


Figura 1.16. Pedido com identificação de grupo anônima [Cattaneo et al. 2004]

Para adicionar o anonimato de forma transparente para os Serviços *Web*, a proposta em [Cattaneo et al. 2004] define os componentes *gerador de provas*, presente no lado do cliente, e o *serviço para verificação de provas*, presente no lado do Serviço *Web* (veja a figura 1.16). As mensagens SOAP, originadas pela aplicação cliente são interceptadas, implicitamente ou explicitamente, pelo *gerador de provas* o qual irá gerar uma credencial de identificação anônima e incluir no cabeçalho SOAP, juntamente com um marcação temporal (passo 2). Da mesma forma, no lado do Serviço *Web* o pedido é interceptado e encaminhado ao *serviço para verificação de provas*, o qual verifica se a credencial é válida (passo 4).

Projeto Liberty Alliance

O projeto *Liberty Alliance* consiste em um conjunto de especificações produzidas por um consórcio de empresas atuantes nas mais diferentes áreas, como em telecomunicações, transportes, universidades, bancos, empresas de *software*, etc. Tem como principal objetivo criar especificações abertas para tratar o gerenciamento de identidades, usufruindo do conceito de *federação de identidades*. Os principais objetivos do projeto são [Liberty 2003a]:

- permitir aos usuários garantir a privacidade e a segurança de suas informações pessoais;
- prover um padrão aberto para permitir uma única autenticação (SSO), o que inclui a autenticação descentralizada e a autorização em múltiplos provedores de serviços;
- prover especificações, compatíveis com uma grande variedade de dispositivos;
- utilizar, em suas especificações, sistemas, padrões e protocolos existentes e amplamente aceitos;

- prover meios para que as empresas respeitem os requisitos de segurança e a privacidade dos clientes.

Esses objetivos podem ser alcançados quando provedores de serviços e clientes agrupam-se baseados em acordos comerciais e nas tecnologias propostas pela *Liberty*, formando assim os *círculos de confiança*. Tais círculos consistem na federação de provedores de serviços e serviços de identidade, juntamente com os clientes. A Figura 1.17 ilustra a arquitetura da *Liberty Alliance* dividida em quatro módulos.

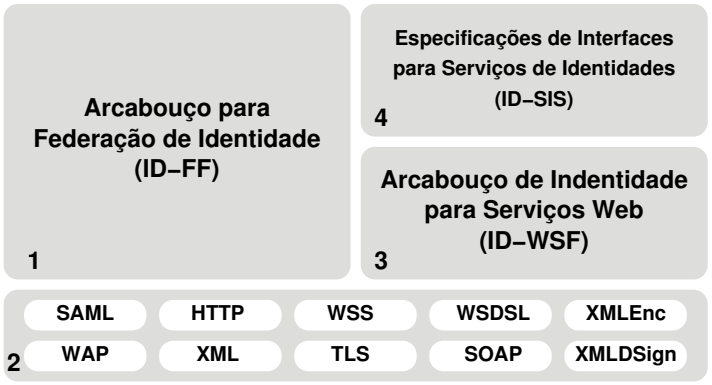


Figura 1.17. Arquitetura da *Liberty Alliance* [Liberty 2003a]

O módulo 1, *Liberty Identity Federation Framekwork* (IDFF), da figura 1.17 visa permitir a federação de identidades e o gerenciamento das mesmas através de características como ligação entre diferentes contas ou identidades, autenticação única (SSO) e o gerenciamento de sessões de forma simplificada. O módulo 2 ilustra a preocupação da *Liberty* em adotar e estender, de forma apropriada, os padrões da indústria ao invés de inventar novas especificações que atuariam de maneira semelhante às já existentes. Algumas das extensões da *Liberty* propostas para o SAML, por exemplo, foram aceitas pelo órgão OASIS e estão presentes na versão 2.0 da especificação do SAML [OASIS 2005c].

No módulo 3, *Liberty Identity Web Services Framework* (IDWSF), é definida uma estrutura para criação, descoberta e acesso aos serviços de identidade. A possibilidade de que empresas ofereçam serviços personalizados para os clientes, de acordo com os atributos e preferências que estes clientes escolheram compartilhar, é uma das principais características deste módulo.

No módulo 4, *Liberty Identity Services Interface Specifications* (IDSIS) , é definida uma coleção de especificações para a construção de serviços interoperáveis sobre a ID-WSF. Tais especificações foram definidas para permitir que organizações possam facilmente criar ou estender serviços sobre a estrutura da ID-WSF. Uma das especificações propostas foi a *ID-Personal Profile*, que define um serviço para obter informações pessoais de um usuário como nome, endereço, telefone, etc. Tal especificação permite que todas organizações que estejam de acordo com a *Liberty* possuam um conjunto de campos e valores conhecidos, tendo assim um dicionário e uma linguagem padrão para que possam interagir entre si. Já estão sendo definidas especificações para obter informações de um usuário relacionadas ao seu emprego, sobre sua geo-localização, etc.

Algumas especificações do projeto *Liberty Alliance* estão direcionadas para garantir a segurança e a privacidade dos clientes. Em [Liberty 2003b], um guia de "boas práticas" para serem seguidas pelos provedores de serviços que estejam de acordo *Liberty* é apresentado, frisando que cada empresa ainda deverá estar de acordo com a jurisdição a qual está submetida.

No guia é descrito que os serviços deverão informar, de forma clara, aos usuários quem está coletando suas informações pessoais, quais informações estão sendo coletadas e de que forma estão sendo coletadas. Os serviços deverão acatar as escolhas do usuário, com relação a privacidade de suas informações pessoais. O usuário deve ter o direito de escolher quais atributos um provedor de serviço terá acesso, bem como os meios para permitir gerenciar e indicar o tempo de vida das informações fornecidas. Deve-se também prover mecanismos para resolução de conflitos para o caso de um usuário acreditar que suas informações não estejam sendo manuseadas de forma incorreta. O guia também apresenta a necessidade de mecanismos que garantam o acesso às informações pessoais de outros usuários, principalmente quando amparado por uma ação judicial.

Os *identificadores opacos* ou *pseudônimos* foram propostos nas especificações da *Liberty* com o intuito de garantir a privacidade dos usuários dos serviços. Para cada provedor de serviço, o provedor de identidade poderá atribuir diferentes pseudônimos relacionados a um mesmo usuário. Dessa forma, o mesmo usuário seria representado por diferentes pseudônimos para cada serviço que fosse acessar, garantindo assim a proteção contra o rastreamento de suas transações. Identificadores opacos permitem aos provedores de serviços identificar quem são seus clientes, relacionando em suas contas locais, porém não possibilita que os provedores de serviços obtenham informações pessoais dos clientes de forma que possa comprometer a privacidade do mesmo.

1.5.2. Gerenciamento de Políticas de segurança

As políticas de segurança descrevem as necessidades e as obrigações de segurança para um dado domínio de segurança. No ambiente dos Serviços *Web*, é comum que um fluxo de negócios seja composto por diversos serviços presentes em diferentes domínios administrativos e de segurança. Dessa forma, um fluxo de negócios pode ser regido por diferentes políticas de segurança e o gerenciamento das mesmas para que a transação ocorra com sucesso e forma segura é um desafio.

Por exemplo, a comunicação de todos os nós de um determinado domínio deverá ser cifrada e assinada, garantindo as propriedades básicas de segurança. Visando remover a complexidade dos nós em ter que trabalhar com uma Infra-estrutura de Chave Pública (ICP), um único nó do domínio poderia ficar responsável pelos processos de cifragem, decifragem, assinatura e verificação de assinaturas. Já em um outro caso, tal solução não seria ideal, visto que é desejado garantir uma segurança fim-a-fim, ou seja, uma vez que a mensagem tenha sido cifrada e/ou assinada pela origem, somente o nó destino poderá ler e/ou modificar a mesma (diferentes contextos de segurança, apresentados na figura 1.4).

Neste caso, a flexibilidade para localização das operações de segurança também deve ser considerada como um requisito da especificação da política de segurança. Em ambiente multi-salto, a política deve ser flexível o suficiente para permitir regras que definam onde deverá ocorrer a cifragem, a decifragem, a assinatura ou a verificação da

assinatura.

Em [Chang et al. 2003], é apresentado um modelo para o gerenciamento de políticas de segurança para ambientes de larga escala compostos por Serviços *Web*. Segundo [Chang et al. 2003], para que uma política de segurança garanta um acordo fim-a-fim, deve-se considerar a interoperabilidade entre as versões das políticas de segurança; garantir a privacidade das partes envolvidas e visar o estabelecimento dinâmico das políticas de segurança, visto que as políticas definidas estaticamente podem se tornar inseguras depois de um certo tempo.

Porém, se por um lado há necessidade de uma evolução dinâmica das políticas, por outro lado tal fato pode trazer um problema. Em ambientes de larga escala, uma transação pode possuir uma longa duração e mudanças nas políticas de segurança não deveriam interferir nessa transação. Dessa forma, é importante também considerar a interoperabilidade entre as versões que uma política pode assumir.

A proposta de [Chang et al. 2003], baseada no uso de um *contrato interoperável* (*Interoperability Contract Document* – ICD), permite a colaboração entre as partes para estabelecer políticas de segurança dinâmicas e individuais para cada operação do serviço e provê ainda medidas para o controle de versão e interoperabilidade destas políticas. A Figura 1.18 ilustra os passos envolvidos em uma transação de acordo com o modelo.

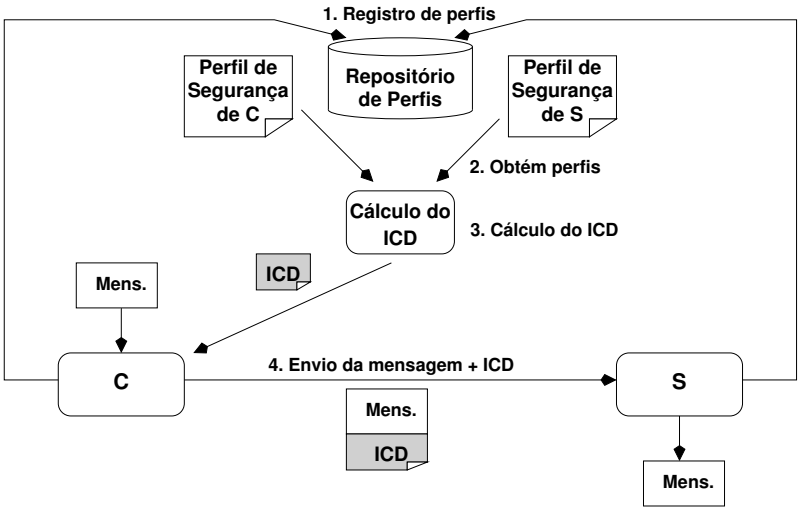


Figura 1.18. O uso do ICD [Chang et al. 2003]

Cada parte, no caso *C* e *S*, registram seus perfis de segurança no repositório de perfis (passo 1). Cada perfil pode estar relacionado a um grupo de serviços ou a serviços individuais e contém as políticas de segurança, suas preferências, etc. Tal repositório está acessível somente para as partes registradas, no exemplo, para *C* e *S*, respectivamente.

Uma vez que *C* queira enviar uma mensagem para *S*, este aciona o módulo para o cálculo do ICD, o qual obtém os perfis de segurança de *C* e de *S* no repositório de perfis (passos 2 e 3). O cálculo do ICD consiste de uma intersecção das preferências de segurança de ambas as partes. Se o resultado for um conjunto vazio, então o processo é abortado através de uma exceção. Por outro lado, se existir mais de um elemento dentro do conjunto, a seleção será baseada em uma ordem de prioridade, a saber: as preferências

do receptor; as preferências do emissor; o nível de segurança; e o desempenho.

Para todas as mensagens que forem enviadas por *C*, será anexado o ICD, indicando assim a política de segurança aplicada especificamente àquela mensagem (passo 4). A mensagem SOAP é personalizada de acordo com as informações de segurança baseadas no ICD. Por exemplo, se para um dado nó a credencial de autenticação for requerida, esta será inserida na parte de autenticação do cabeçalho SOAP. E, se a integridade for requerida, a mensagem será assinada e o algoritmo utilizado será identificado pelo ICD, o mesmo para a cifragem. Por fim, na recepção da mensagem, através do ICD, *S* consegue verificar se está de acordo com o combinado.

O ICD anexado à mensagem possibilita uma evolução da aplicação sem que isso acarrete problemas de segurança. Dessa forma, quando a versão da política de segurança de qualquer parte avançar, um novo ICD é recalculado e será anexado nas próximas mensagens que sairão. Com isso, uma parte consegue aplicar diferentes versões da política para interoperar com diferentes parceiros sem ambigüidades.

Conforme apresentado na Figura 1.18, os perfis de segurança de *C* e de *S* estão armazenados em um mesmo repositório, porém é previsto no modelo a presença de diversos repositórios, por exemplo, um por domínio administrativo e assim, no cálculo do ICD os perfis de segurança podem ser obtidos de diferentes repositórios. No trabalho [Chang et al. 2003] não é apresentada uma forma para localizar tais repositórios, bem como para saber em qual repositório estará armazenado o perfil de segurança de cada parte.

1.5.3. Gerenciamento de Confiança

A federação de identidades tem como ponto fundamental as relações de confiança entre provedores de serviços e clientes, e entre os próprios provedores de serviço. O fornecimento de informações pessoais de um cliente a um provedor de serviço só ocorre depois que o cliente tenha certeza de que suas informações serão manipuladas de maneira correta. Por outro lado, o provedor de serviço só irá conceder ao cliente o acesso ao recurso, se o mesmo confiar nas informações fornecidas pelo cliente. O mesmo ocorre nas relações de confiança entre provedores de serviço. Tais relações permitem, por exemplo, que um usuário autenticado em um determinado provedor possa usufruir dos recursos providos por outro provedor, já que ambos possuem um acordo indicando o compartilhamento de recursos para os usuários de ambos provedores.

A seguir, serão apresentados alguns trabalhos que tratam diretamente com o problema da confiança, com um enfoque voltado na negociação da confiança dentro do ambiente dos Serviços *Web*.

TrustBuilder

Em [Winslett et al. 2002], é apresentado o sistema *TrustBuilder* que tem por objetivo permitir o estabelecimento dinâmico da confiança entre partes estranhas dentro do contexto da Internet. O *TrustBuilder* permite que as partes, envolvidas na negociação, revelem gradualmente suas credenciais e políticas de controle de acesso para estabelecer a confiança

necessária para a realização da comunicação efetiva entre as partes. Este trabalho está focado na definição de estratégias e protocolos para o estabelecimento da confiança.

O estabelecimento da confiança leva em consideração que cada parte possui políticas de controle de acesso sobre os recursos que deseja proteger. Tais recursos podem ser os serviços providos por uma determinada parte ou ainda as credenciais de segurança, como o número do documento de identidade, número do passaporte, número do cartão de crédito, etc. As políticas também definem quais credenciais específicas, devem ser apresentadas para que se obtenha acesso ao recurso desejado.

As estratégias controlam quais e quando as credencias serão reveladas e também quando a negociação será finalizada. Neste caso, as estratégias estarão trabalhando juntamente com as políticas de controle de acesso. Já o protocolo indica a ordem das mensagens a serem trocadas, bem como quais informações deverão estar contidas nas mensagens.

A revelação gradual das credenciais trata de uma medida preventiva contra partes com quem ainda não exista uma relação de confiança. Por exemplo, João deseja comprar um produto na empresa XYZ. Para que se possa confirmar a compra, é necessário que João forneça o número do seu cartão de crédito, porém João não fará isso sem que antes a empresa XYZ forneça informações de que a mesma trata-se de uma empresa idônea, informação esta que pode ser emitida por um órgão no qual João já confia. Neste caso, o problema está em como revelar as credencias de cada parte, sem que isso venha trazer prejuízos caso alguma parte não honre suas obrigações.

Uma solução para o problema consiste na utilização de uma Terceira Parte Confiável (TPC), em que ambas as partes envolvidas na comunicação, revelam suas credenciais e políticas para a TPC e delegam a esta a tarefa de determinar a confiança entre cada parte. Porém, tal solução torna-se um gargalo em ambientes de larga escala e ainda um ponto único de vulnerabilidade. O uso do conceito de *prova de conhecimento zero* conseguiria provar que as credenciais respeitam as políticas sem que seja preciso revelar tanto as credenciais quanto as políticas (o trabalho de [Cattaneo et al. 2004], apresentado anteriormente faz uso deste conceito). Porém, segundo apresentado em [Winslett et al. 2002], tal solução é difícil de ser implementada de forma eficiente.

O *TrustBuilder* baseia-se na negociação direta entre as partes, através da revelação parcial das credenciais e políticas. Sabendo que cada parte pode possuir diversas políticas e credenciais de segurança, as quais podem ou não ser utilizadas em determinadas negociações, a medida, considerada por [Winslett et al. 2002] como a melhor alternativa, consiste em somente revelar as políticas necessárias para a comunicação em questão.

A Figura 1.19 ilustra um exemplo apresentado em [Winslett et al. 2002]. No passo 1, João deseja comprar um produto da empresa XYZ, indicando que quer um desconto no valor final do produto. A empresa XYZ definiu em suas políticas que somente os clientes que comprovarem que são "revendedores" poderão obter desconto, dessa forma, no passo 2, é enviada a política $P2$ para João, requerendo uma "credencial de revendedor" e também o número do cartão de crédito para que se possa efetivar a venda.

Por sua vez, João também possui um política local a qual indica que só fornecerá seu número de cartão de crédito a instituições que estejam devidamente regulamentadas

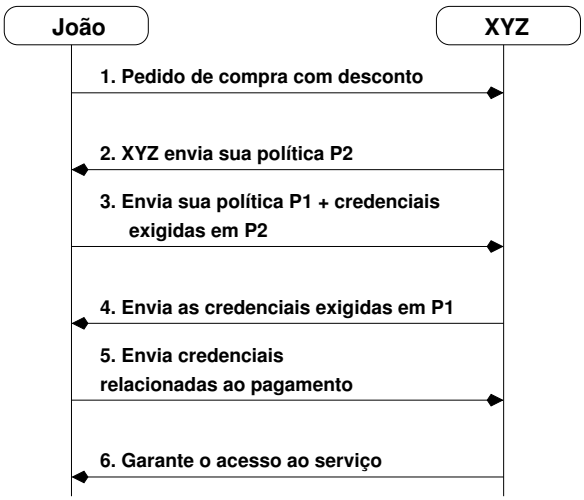


Figura 1.19. *TrustBuilder*: negociação de confiança

na *Federação da Indústria e do Comércio*. Assim, no passo 3 João fornece a credencial de que é um revendedor juntamente com a sua política P1, indicando que XYZ prove que faz parte da *Federação da Indústria e do Comércio*. No passo 4, XYZ fornece a credencial exigida por P1 e assim, no próximo passo, João fornece o número do cartão de crédito. Por fim, após a empresa XYZ confrontar as credenciais fornecidas com a sua política de controle de acesso, esta garante a João a venda do produto.

Trust-Serv

O *Trust-Serv* [Skogsrud et al. 2003] é uma infra-estrutura voltada para o estabelecimento de confiança dentro do ambiente dos Serviços *Web*. O trabalho apresenta um modelo de políticas para o estabelecimento de confiança baseado em máquinas de estado [Hopcroft e Ullman 1979]. Neste trabalho, um modelo para o gerenciamento do ciclo de vida das políticas o que permite a evolução, bem como a migração, das políticas sem que isso interrompa as negociações que já estejam em andamento é apresentado. A estratégia proposta também permite compensar o requisitor, caso os direitos de acesso concedidos a este sejam revogados em uma migração para uma nova política. Segundo [Skogsrud et al. 2003], o *Trust-Serv* é um trabalho complementar ao *TrustBuilder* [Winslett et al. 2002] e pode ser utilizado para prover o gerenciamento do ciclo de vida das políticas.

No *Trust-Serv*, os *estados* de uma máquina de estados representam o nível de confiança atingido pelo requisitor e para cada novo estado que o requisitor atingir o acesso a novos recursos será garantido. Os recursos são definidos como operações dos Serviços *Web* ou credenciais do próprio provedor de serviços. O *Trust-Serv* ainda adota o conceito de *papéis* [Ferraiolo et al. 2001] e, ao invés de associar os recursos diretamente aos *estados*, associam-se papéis. No modelo, os papéis são acumulativos e assim, os papéis ativados em um estado anterior não serão desativados ao se atingir um novo estado. Já as *transações* indicam as condições que um requisitor deve cumprir para que possa sair de um estado e ir para outro. Em [Skogsrud et al. 2003], são propostas extensões às tran-

sações de uma máquina de estado tradicional para capturar as abstrações de segurança, necessárias para o estabelecimento da confiança.

A arquitetura do *Trust-Serv* é dividida em camadas o que permite separar os mecanismos para o estabelecimento da confiança e o controle de acesso (nível de controle) da lógica de aplicação (nível de serviço). Para cada Serviço *Web*, é associado um *controlador*, o qual intercepta de forma transparente todas as mensagens direcionadas a este serviço. Os controladores podem aceitar ou recusar uma invocação ou ainda iniciar uma iteração com o *controlador* da outra parte para estabelecer um nível de confiança, antes de aceitar a invocação. A Figura 1.20 ilustra a disposição dos *controladores* na arquitetura do *Trust-Serv*.

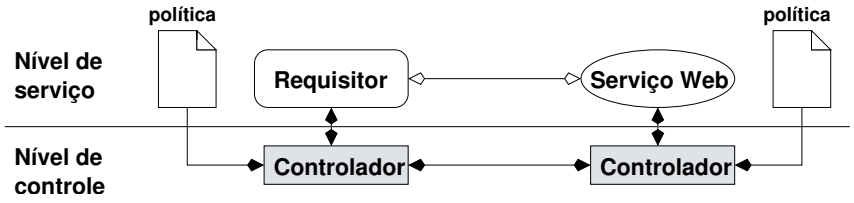


Figura 1.20. *TrustServ*: Níveis de serviço e de controle [Skogsrud et al. 2004]

O trabalho também propõe soluções para o gerenciamento do ciclo de vida das políticas, no caso, o foco do trabalho está direcionado ao contexto das políticas para o estabelecimento da confiança entre os Serviços *Web*. A substituição direta de uma política que já está sendo empregada por uma nova política pode não ser o ideal, visto que isso poderia acarretar no reinício de todas as negociações que já estão em andamento. O *Trust-Serv* provê diferentes estratégias para lidar com estes problemas:

- **coexistência**: permite que a negociação em andamento seja finalizada de acordo com a política antiga, porém requer que todas as novas negociações obedeçam a nova política;
- **abortamento**: aborta todas as negociações que estejam em andamento;
- **migração**: migra todas as negociações existentes para a nova política, desde que a política antiga esteja contida dentro da nova política. Dessa forma, todos os estados já visitados e todas as transações já disparadas na política antiga deverão estar presentes na nova política. As negociações em andamento que estejam de acordo com essa medida serão migradas para a nova política. Já as negociações que não estejam de acordo serão retornadas até atingirem um estado que esteja de acordo com a nova política.

Em alguns casos, as estratégias *Coexistência* e *Abortar* podem não ser ideais. A coexistência de duas políticas operando ao mesmo tempo pode não ser desejado, por exemplo, do ponto de vista legal. Clientes estariam recebendo tratamento diferenciados diante, por exemplo, de um mesmo pagamento. Já abortar todas as transações existentes poderiam trazer prejuízos para o provedor do serviço, visto que diversos clientes poderiam desistir de tentar recomeçar toda a negociação novamente. Por outro lado, a estratégia da

migração consegue unir as vantagens das duas estratégias anteriores. A regressão até um estado que seja comum às duas políticas evita que toda a transação tenha que ser refeita e também garante que os próximos estados estarão de acordo com a nova política, evitando assim a coexistência de diferentes políticas.

1.6. Ferramentas para o Desenvolvimento de Serviços *Web* Seguros

Um Serviço *Web* consiste de um componente de software que permite a interação entre aplicações através de uma rede [W3C 2004a]. Em resumo, qualquer aplicação que consiga enviar, receber e processar mensagens SOAP trocadas através de algum protocolo de transporte pode ser considerada um Serviço *Web*.

Existem hoje diversas ferramentas para o desenvolvimento de aplicações baseadas na arquitetura dos Serviços *Web*. Algumas destas são ambientes completos para o desenvolvimento e disponibilização dos serviços, como a *Java Web Services Development Pack* (JWSDP)[16], já outras são só implementações do SOAP, como é o caso do Apache Axis[17] e necessitam de outras ferramentas para permitir a disponibilização dos serviços ou mesmo para agregar características de segurança. As licenças de distribuição destas ferramentas também são outro ponto de divergência. Algumas estão sob licença de código fechado (proprietário) e exigem o pagamento para o uso de ferramentas, outras possuem uma licença de distribuição gratuita, porém não disponibilizam os códigos fontes, e, por fim, existem ainda as ferramentas sob licença de *software* livre. Esta seção aborda algumas ferramentas sob licença de *software* livre, devido à natureza destas ser mais adequada ao meio acadêmico e por que estas permitem a divulgação do conhecimento através da disponibilização do código e da possibilidade de modificar e redistribuir o mesmo.

O Apache Axis é uma ferramenta de código aberto que disponibiliza um servidor SOAP juntamente com um conjunto de APIs que facilita o desenvolvimento de aplicações clientes e de Serviços *Web*. Apesar do Axis apresentar um servidor HTTP próprio para a disponibilização dos serviços, os desenvolvedores geralmente fazem uso do servidor de aplicação Apache TomCat[18], que também possui uma licença de código aberto, já que este último é mais robusto e completo.

Atualmente, o Axis possui duas versões que estão sendo desenvolvidas em paralelo. A versão 1 possui implementações em Java e em C++, e além de ser um interpretador SOAP, apresenta ferramentas e APIs para tratar diretamente com documentos WSDL (ver seção 1.3). Já a versão 2 do Axis[19], recentemente lançada, só possui a implementação em Java e segundo sua documentação, trata-se de uma completa reestruturação da versão 1, possuindo uma melhor modularidade, mais focada no XML e mais eficiente que a versão 1. Porém, a principal diferença é que a versão 2 não se resume apenas na implementação das especificações SOAP 1.1 e SOAP 1.2. Esta versão apresenta também uma melhor integração com as propostas para os Serviços *Web* (como a *WS-Security* [OASIS 2004c], *WS-Coordination* [Cabrera et al. 2004], entre outras), permitindo a integração destas através de módulos de *software*.

[16] `http://java.sun.com/webservices/jwsdp`
[17] `http://ws.apache.org/axis`
[18] `http://tomcat.apache.org`
[19] `http://ws.apache.org/axis2`

Porém, o Axis não apresenta soluções para prover segurança nas aplicações desenvolvidas com ele. Para isso, a própria fundação Apache lançou diversas outras ferramentas que implementam as principais especificações de segurança descritas na seção 1.4. A Apache XMLSecurity[20] é uma implementação de código aberto para as especificações *XMLDSign* e *XMLEncryption*. Trata-se de um trabalho bem maduro e amplamente aceito, sendo até mesmo utilizado por outras plataformas de desenvolvimento, como a JWSDP da empresa Sun.

O projeto Apache WSS4J (*WS-Security for Java*) é uma implementação de código aberto da especificação *WS-Security* [OASIS 2004c], que consiste de uma biblioteca Java que pode ser usada para assinar e verificar mensagens SOAP que contenham informações expressas de acordo com a *WS-Security*. Diferentemente da biblioteca Apache XMLSecurity, a WSS4J está diretamente ligada ao Apache Axis, a XMLSecurity e ainda a biblioteca OpenSAML[21], uma implementação de código aberto para a SAML[OASIS 2005c]. Estão surgindo alguns esforços para agregar à WSS4J os conceitos definidos na especificação *WS-Trust* [WS-Trust 2005], mas a implementação ainda não está tão madura quanto as outras bibliotecas apresentadas aqui.

1.7. Conclusão

Pode-se dizer que o grande sucesso das aplicações para Internet se deu devido ao alto nível de abstração. Isto permitiu garantir a interoperabilidade entre as mais diversas aplicações, sistemas operacionais e equipamentos. Os Serviços *Web* exploram esse nível de abstração associado a uma lógica de negócios.

Neste capítulo buscou-se introduzir os conceitos que regem a arquitetura orientada a serviços, tendo como foco os Serviços *Web*. Foram apresentados alguns dos desafios de segurança relacionados à arquitetura dos Serviços *Web*, bem como alguns trabalhos que visam tratar tais desafios.

Como visto, além das propriedades básicas de segurança, a concepção de aplicações baseadas nos Serviços *Web* deve considerar pontos como a transposição de domínios administrativos e de segurança, o que acarreta em preocupações com a privacidade, o anonimato, a evolução das políticas de segurança, e principalmente, a interoperabilidade. Para muitos destes desafios, já foram lançadas diversas propostas com o aval de órgãos padronizadores, fornecendo assim um ponto de partida comum para que desenvolvedores possam criar suas aplicações e que as mesmas serão interoperáveis.

Porém, muitas especificações para o ambiente dos Serviços *Web* são projetadas para serem estendidas e ainda apresentam diversas formas para expressar a mesma função, o que permite diferentes interpretações e como conseqüência, tais características tornam-se uma barreira contra a interoperabilidade [WS-I 2005]. Por exemplo, a especificação *WS-Security* introduz muitas opções e escolhas. Se diferentes empresas selecionam diferentes opções, estas podem não mais interoperar, apesar de estarem seguindo a mesma especificação. Já existem preocupações a respeito e o órgão WS-I já está apresentando algumas recomendações para o uso de padrões e tecnologias de segurança para os Serviços

[20] `http://xml.apache.org/security/`
[21] `http://www.opensaml.org`

Web, de forma a garantir a real interoperabilidade entre as implementações.

Enquanto em algumas áreas de segurança já existem especificações consolidadas, mesmo diante de algumas ambigüidades, em outras áreas, como o gerenciamento de políticas e de confiança, embora este capítulo tenha apresentado alguns trabalhos, não existem ainda padrões de fato, sendo esta um bom ambiente para pesquisa.

No sítio `http://www.das.ufsc.br/seguranca/webservices` estão disponíveis informações sobre as recentes pesquisas[22] feitas pelo Grupo de Computação Segura e Confiável (GCSeg) da UFSC, no contexto do projeto "Infra-estrutura de Segurança para Aplicações Distribuídas Orientadas a Serviço", que tem apoio do CNPq. Além disso, também estão disponíveis documentos técnicos que detalham como implantar a infra-estrutura necessária para o desenvolvimento e provimento de aplicações baseadas nos Serviços *Web*, e alguns exemplos de código, estes disponíveis sob licenças de *software* livre.

Referências

[Amoroso 1994] Amoroso, E. G. (1994). *Fundamentals of Computer Security Technology*. Prentice Hall.

[Asokan et al. 1997] Asokan, N., Schunter, M., e Waidner, M. (1997). Optimistic protocols for fair exchange. In *CCS '97: 4th ACM conference on Computer and communications security*, pages 7–17, New York, NY, USA. ACM Press.

[Bartel et al. 2002] Bartel, M., Boyer, J., e Fox, B. (2002). *XML-Signature Syntax and Processing*. W3C. `http://www.w3.org/TR/xmldsig-core`.

[Bishop e Bailey 1996] Bishop, M. e Bailey, D. (1996). A critical analysis of vulnerability taxonomies. Technical Report CSE-96-11, Department of Computer Science at University of California, Davis.

[Boyer 2001] Boyer, J. (2001). *Canonical XML*. W3C. `http://www.w3.org/TR/xml-c14n`.

[Brown e Kindel 1996] Brown, N. e Kindel, C. (1996). *Distributed Component Object Model Protocol – DCOM/1.0*. Microsoft.

[Cabrera et al. 2004] Cabrera, F., Copeland, G., Freund, T., Klein, J., Langworthy, D., Orchard, D., Shewchuk, J., e Storey, T. (2004). *Web Services Coordination*. Web Services Interoperability Organization. `http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-Coordination.pdf`.

[Carmody 2001] Carmody, S. (2001). *Shibboleth Overview and Requirements*. Shibboleth Working Group.

[Cattaneo et al. 2004] Cattaneo, G., Faruolo, P., e Petrillo, U. F. (2004). Providing privacy for web services by anonymous group identification. In *International Conference on Web Services (ICWS'04)*. IEEE.

[22] [de Mello et al. 2005, de Mello e da Silva Fraga 2005, Wangham et al. 2005, Wangham et al. 2006]

[Chang et al. 2003] Chang, S., Chen, W., e Hsu, M. (2003). Managing security policy in a large distributed web services environment. In *27th International Computer Software and Applications Conference (COMPSAC'03)*. IEEE.

[Charfi e Mezini 2005] Charfi, A. e Mezini, M. (2005). Using aspects for security engineering of web service compositions. In *Proceedings of the 2005 IEEE International Conference on Web Services, Volume I*, pages 59–66.

[Daemen e Rijmen 2002] Daemen, J. e Rijmen, V. (2002). *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag.

[Damiani et al. 2003] Damiani, E., di Vimercati, S. D. C., e Samarati, P. (2003). Managing multiple and dependable identities. In *IEEE Internet Computing*, pages 29–37. IEEE.

[de Mello e da Silva Fraga 2005] de Mello, E. R. e da Silva Fraga, J. (2005). Mediation of trust across web services. In *3rd IEEE International Conference on Web Services (ICWS'05)*, pages 515–522, Orlando, Flórida - EUA.

[de Mello et al. 2005] de Mello, E. R., Wangham, M., da Silva Fraga, J., e Rabelo, R. (2005). A secure model to establish trust relationships in web services for virtual organizations. In Camarinha-Matos, L. M., Afsarmanesh, H., e Ortiz, A., editors, *6th IFIP Working Conference on Virtual Enterprises (PRO-VE'05)*, pages 183–190, Valência, Espanha. Springer.

[Demchenko et al. 2005] Demchenko, Y., Gommans, L., de Laat, C., e Oudenaarde, B. (2005). Web services and grid security vulnerabilities and threats analysis and model. `http://www.uazone.org/demch/analytic/draft-grid-security-incident-04.pdf`.

[Dierks e Allen 1999] Dierks, T. e Allen, C. (1999). *The TLS Protocol – Version 1.0*. IETF RFC 2246.

[Eastlake e Jones 2001] Eastlake, D. e Jones, P. (2001). *US Secure Hash Algorithm 1 (SHA1)*. Internet Engineering Task Force RFC 3174.

[Ellison et al. 1999] Ellison, C. M., Frantz, B., Lampson, B., Rivest, R., Thomas, B. M., e Ylonen, T. (1999). *SPKI Certificate Theory*. Internet Engineering Task Force RFC 2693.

[Ferraiolo et al. 2001] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., e Chandramouli, R. (2001). Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274.

[Freier et al. 1996] Freier, A. O., Karlton, P., e Kocher, P. C. (1996). *The SSL protocol - v.3*. Internet Draft.

[Goldwasser et al. 1989] Goldwasser, S., Micali, S., e Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208.

[Hallam-Baker e Mysore 2005] Hallam-Baker, P. e Mysore, S. H. (2005). *XML Key Management Specification (XKMS 2.0)*. W3C – Proposed Recommendation.

[Hopcroft e Ullman 1979] Hopcroft, J. e Ullman, J. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

[Housley et al. 2002] Housley, R., Polk, W., Ford, W., e Solo, D. (2002). *Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. IETF RFC 3280.

[Hung et al. 2004] Hung, P. C. K., Ferrari, E., e Carminati, B. (2004). Towards standardized web services privacy technologies. In *International Conference on Web Services (ICWS'04)*. IEEE.

[IBM e Microsoft 2002] IBM e Microsoft (2002). *Security in a Web Services World: A Proposed Architecture and Roadmap*. IBM Corporation and Microsoft Corporation. `http://msdn.microsoft.com/ws-security/`.

[Imamura et al. 2002] Imamura, T., Dillaway, B., e Simon, E. (2002). *XML Encryption Syntax and Processing*. W3C. `http://www.w3.org/TR/xmlenc-core`.

[Jøsang et al. 2005] Jøsang, A., Fabre, J., Hay, B., Dalziel, J., e Pope, S. (2005). Trust requirements in identity management. In *CRPIT '44: Proceedings of the 2005 Australasian workshop on Grid computing and e-research*, pages 99–108, Darlinghurst, Australia. Australian Computer Society, Inc.

[Jøsang e Pope 2005] Jøsang, A. e Pope, S. (2005). User centric identity management. In *AusCERT Asia Pacific Information Technology Security Conference 2005*.

[Kohl e Neuman 1993] Kohl, J. e Neuman, C. (1993). *The Kerberos Network Authentication Service (v5)*. Internet Engineering Task Force RFC 1510.

[Landwehr 2001] Landwehr, C. E. (2001). Computer Security. In *International Journal of Information Security*, volume 1, pages 3–13. Springer-Verlag Heidelberg.

[Liberty 2003a] Liberty (2003a). *Introduction to the Liberty Alliance Identity Architecture*. Liberty Alliance.

[Liberty 2003b] Liberty (2003b). *Privacy and Security Best Practices*. Liberty Alliance.

[Lorch et al. 2003] Lorch, M., Proctor, S., Lepro, R., Kafura, D., e Shah, S. (2003). First experiences using xacml for access control in distributed systems. In *ACM Workshop on XML Security*.

[OASIS 2002] OASIS (2002). *Universal Description, Discovery and Integration v2 (UDDI)*. Organization for the Advancement of Structured Information Standards (OASIS).

[OASIS 2004a] OASIS (2004a). *Introduction to UDDI: Important features and functional concepts*. Organization for the Advancement of Structured Information Standards (OASIS). `http://uddi.org/pubs/uddi-tech-wp.pdf`.

[OASIS 2004b] OASIS (2004b). *Universal Description, Discovery and Integration v3.0.2 (UDDI)*. Organization for the Advancement of Structured Information Standards (OASIS).

[OASIS 2004c] OASIS (2004c). *Web Services Security: SOAP Message Security 1.0*. OASIS. `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf`.

[OASIS 2005a] OASIS (2005a). *eXtensible Access Control Markup Language (XACML) version 2.0*. Organization for the Advancement of Structured Information Standards (OASIS). `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`.

[OASIS 2005b] OASIS (2005b). *SAML Executive Overview*. Organization for the Advancement of Structured Information Standards (OASIS).

[OASIS 2005c] OASIS (2005c). *Security Assertion Markup Language (SAML) 2.0 Technical Overview*. Organization for the Advancement of Structured Information Standards (OASIS).

[OMG 2002] OMG (2002). *The Common Object Request Broker Architecture v3.0.2*. Object Management Group (OMG).

[OpenGroup 1997] OpenGroup (1997). *DCE 1.1: Remote Procedure Call*. Open Group Technical Standard, AE Specification C309.

[Papazoglou 2003] Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In *Fourth International Conference on Web Information systems Engineering (WISE'03)*.

[Parr e Villars 2001] Parr, B. e Villars, R. (2001). Digital identity: The coming struggle for the future of the net. Boletim 24929, IDC.

[Rannenberg 2000] Rannenberg, K. (2000). Multilateral security a concept and examples for balanced security. In *Workshop on New security paradigms (NSPW'00)*, pages 151–162, New York, NY, USA. ACM Press.

[Rivest e Lampson 1996] Rivest, R. L. e Lampson, B. (1996). SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession.

[RSA 2002] RSA (2002). *PCKS#1 v2.1: RSA Cryptography Standard*. RSA Laboratories. `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf`.

[RSS 2005] RSS (2005). *Really Simple Syndication*. `http://www.rssboard.org/rss-specification`.

[Russell e Gangeni 1991] Russell, D. e Gangeni, G. (1991). *Computer Security Basics*. O'Reilly Associates Inc.

[Santis et al. 1998] Santis, A. D., Crescenzo, G. D., e Persiono, G. (1998). Communication-efficient anonymous group identification. In *5th A.C.M. Conference on Computer and Communications Security (ACM CCS'98)*, pages 73–82, San Francisco, California, U.S.A.

[Shibboleth 2005] Shibboleth (2005). *Shibboleth Architecture*. http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf.

[Skogsrud et al. 2003] Skogsrud, H., Benatallah, B., e Casati, F. (2003). Modelo-driven trust negotiation for web services. In *IEEE Internet Computing*, pages 45–52. IEEE Computer Society.

[Skogsrud et al. 2004] Skogsrud, H., Benatallah, B., e Casati, F. (2004). Trust-serv: model-driven lifecycle management of trust negotiation policies for web services. In *WWW 2004*, pages 53–62. ACM.

[Sun 2002] Sun (2002). Java remote method invocation specification. Revision 1.8 Java 2 SDK.

[Vogels 2003] Vogels, W. (2003). Web services are not distributed objects. *Internet Computing*, 7(6):59–66.

[W3C 2001] W3C (2001). *Web Services Description Language 1.1*. W3C Working Group.

[W3C 2002] W3C (2002). *The Platform for Privacy Preferences 1.0 (P3P1) Specification*. W3C Recommendation. http://www.w3c.org/TR/P3P.

[W3C 2003] W3C (2003). *SOAP 1.2 – W3C Recommendation*. W3C. http://www.w3.org/TR/soap12.

[W3C 2004a] W3C (2004a). *Web Services Architecture*. W3C Working Group. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211.

[W3C 2004b] W3C (2004b). *Web Services Architecture Requirements*. W3C Working Group. http://www.w3.org/TR/2004/NOTE-wsa-reqs-20040211.

[Wangham et al. 2006] Wangham, M., da Silva Fraga, J., de Mello, E. R., e Milanez, J. (2006). Um modelo para o gerenciamento federado do spki/sdsi através do serviço xkms. In *VI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSEG'06)*, Santos, SP - Brasil.

[Wangham et al. 2005] Wangham, M. S., Mello, E., Rabelo, R., e da Silva Fraga, J. (2005). Provendo garantias de segurança para formação de organizações virtuais. In Gerrini, F. M., editor, *Gestão Avançada de Manufatura*, volume 2, pages 75–84. Editora Novos Talentos.

[Weerawarana et al. 2005] Weerawarana, S., Curbera, F., Leymann, F., Storey, T., e Ferguson, D. F. (2005). *Web Services Plataform Architecture*. Prentice Hall.

[Wege 2002] Wege, C. (2002). Portal server technology. *IEEE Internet Computing*, 6(3):73–77.

[Westbridge 2003] Westbridge (2003). *Securing and Managing XML Web Services – Guide to XML Web Services Security*. Westbridge Technology Inc.

[Winslett et al. 2002] Winslett, M., Yu, T., Seamons, K. E., Hess, A., Jacobson, J., Jarvis, R., Smith, B., e Yu, L. (2002). Negotiating trust on the web. In *IEEE Internet Computing*, number 6 in 6, pages 30–37. IEEE Computer Society.

[WS-Federation 2003a] WS-Federation (2003a). *Web Services Federation Language*. `http://msdn.microsoft.com/ws/2003/07/ws-federation`.

[WS-Federation 2003b] WS-Federation (2003b). *WS-Federation: Active Requestor Profile*. `ftp://www6.software.ibm.com/software/developer/library/ws-fedact.pdf`.

[WS-Federation 2003c] WS-Federation (2003c). *WS-Federation: Passive Requestor Profile*. `ftp://www6.software.ibm.com/software/developer/library/ws-fedpass.pdf`.

[WS-I 2005] WS-I (2005). *Basic Security Profile Version 1.0*. Web Services Interoperability Organization. `http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2005-08-29.html`.

[WS-Policy 2004] WS-Policy (2004). *Web Services Policy Framework*. `http://msdn.microsoft.com/ws/2004/09/policy/`.

[WS-PolicyAttachment 2004] WS-PolicyAttachment (2004). *Web Services Policy Attachment*. `http://msdn.microsoft.com/ws/2004/09/policyattachment`.

[WS-SecureConversation 2005] WS-SecureConversation (2005). *Web Services Secure Conversation Language*.

[WS-SecurityPolicy 2005] WS-SecurityPolicy (2005). *Web Services Security Policy Language*.

[WS-Trust 2005] WS-Trust (2005). *Web Services Trust Language (WS-Trust)*. `http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-Trust.asp`.

[Wu 1998] Wu, T. (1998). The secure remote password protocol. In *Internet Society Network and Distributed System Security Symposium*, pages 97–111.

[Yavatkar et al. 2000] Yavatkar, R., Pendarakis, D., e Guerin, R. (2000). *A Framework for Policy-based Admission Control*. IETF RFC 2753.

[Zimmerman 1994] Zimmerman, P. (1994). *PGP User's Guide*. Massachusetts Institute of Technology.