EDSON TAVARES DE CAMARGO

TRANSPOSIÇÃO DE AUTENTICAÇÃO EM ARQUITETURAS ORIENTADAS A SERVIÇOS ATRAVÉS DE IDENTIDADES FEDERADAS

FLORIANÓPOLIS
2006

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

TRANSPOSIÇÃO DE AUTENTICAÇÃO EM ARQUITETURAS ORIENTADAS A SERVIÇOS ATRAVÉS DE IDENTIDADES FEDERADAS

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica.

EDSON TAVARES DE CAMARGO

Florianópolis, Junho de 2006.

TRANSPOSIÇÃO DE AUTENTICAÇÃO EM ARQUITETURAS ORIENTADAS A SERVIÇOS ATRAVÉS DE IDENTIDADES FEDERADAS

Edson Tavares de Camargo

'Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica, Área de Concentração em *Automação e Sistemas*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

Joni da Silva Fraga, Dr.
Orientador

Nelson Sadowski, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Joni da Silva Fraga, Dr.
Presidente

Michelle Silva Wangham, Dr.

Altair Olivo Santin, Dr.

Ricardo José Rabelo, Dr.

Frank Auguto Siqueira, Dr.

Aos meu pais, Maria e Carlito…

AGRADECIMENTOS

A Deus. Obrigado Pai pelo teu amor, amizade, força... Por tudo!.

Ao meu orientador, Joni Fraga, pela confiança, orientação, questionamentos e incentivo (em diversos sentidos). “Grande” Joni, muito obrigado!

Agradeço ao Departamento de Automação e Sistemas e aos respectivos professores. Ao professor Rabelo e ao Projeto ECOLEAD pelos 12 meses de bolsa.

Agradeço aos amigos do grupo de Segurança Computacional. Obrigado pela paciência, ajuda e compreensão. Obrigado principalmente ao Emerson e a Michelle que participaram deste trabalho.

Amigos, obrigado! Você simplesmente foram amigos. O nome de cada um de vocês não cabe aqui, mas o meu coração é grato pela vida de vocês. Obrigado à galera do GOU (Grupo de Oração Universitário), do LIDAS, do LCMI, da Ação Social do GOU, aos Sardinhas (viu, eu consegui!!!), aos que começaram como colegas de mestrado e se tornaram amigos no decorrer deste curso...

Agredeço ao galera do Ministério Universidades Renovadas (nos diversos níveis: diocesanos, estadual, nacional), por me incentivarem a sonhar. Obrigado pela amizade. Galera de Cascavel, valeu!!! Obrigado “maninhos”.

À minha família, pelo carinho e apoio financeiro e sobretudo afetivo. Pai, Mãe, sem vocês nada disso teria sentido. Obrigado por tudo!! Amo vocês!

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

TRANSPOSIÇÃO DE AUTENTICAÇÃO EM ARQUITETURAS ORIENTADAS A SERVIÇOS ATRAVÉS DE IDENTIDADES FEDERADAS

Edson Tavares de Camargo

Junho/2006

Orientador: Joni da Silva Fraga
Área de Concentração: Automação e Sistemas
Palavras-chave: Aquiteturas Orientadas a Serviço, Serviços *Web*, Segurança Computacional, Identidades Federadas
Número de Páginas: 1 + 132

A transposição de autenticação consiste, basicamente, em permitir o acesso a recursos controlados em um domínio diferente daquele do cliente, usando credenciais fornecidas no seu domínio de origem. No entanto, grandes são os empecilhos para a concretização de tais sistemas e vão desde a falta de interoperabilidade entre protocolos, linguagens de programação, plataformas operacionais, utilizadas no seu projeto, até as diferentes infra-estruturas de segurança subjacentes. Além disso, a segurança computacional (*computer security*) se apresenta como um requisito indispensável neste cenário.

Embora Serviços *Web* (*Web Services*) se configure como a tecnologia mais indicada para contornar os desafios da transposição, o grande número de especificações e padrões, destinados a Serviços *Web*, tanto impõe um certo nível de complexidade quanto dificulta a sua adoção. Algumas propostas ainda são imaturas e carecem de um estudo e implementação que abordem as soluções como um todo para uma boa compreensão do potencial e das limitações da tecnologia.

Para que a identidade do usuário seja reconhecida em diversos domínios administrativos, o gerenciamento de identidades federadas desponta como uma solução promissora. Além disso, devido a dinâmica do ambiente dos Serviços *Web*, no qual serviços muitas vezes são agregados de forma temporária para atender um determinado fluxo de negócios, o gerenciamento de identidade ganha cada vez mais destaque na literatura.

O objetivo deste trabalho é propor um modelo para a transposição de autenticação - e a conseqüente autorização descentralizada e transposição de atributos do cliente - entre domínios administrativos com diferentes infra-estruturas de segurança. Neste modelo, cada domínio agrupa clientes e provedores de serviço de acordo com a tecnologia de segurança usada. São assumidos, na construção do modelo, padrões relacionados a tecnologias de Serviços *Web* e os conceitos de gerenciamento de identidades federadas. Um protótipo é implementado a fim de verificar a aplicabilidade do modelo.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

AUTENTICATION TRANSPOSITION IN SERVICE ORIENTED ARCHITECTURE THROUGH FEDERATED IDENTITIES

Edson Tavares de Camargo

June/2006

Advisor: Joni da Silva Fraga
Area of Concentration: Automation and Systems
Key words: Service Oriented Arquitecture, Web Services, Computer Security, Federated Identity
Number of Pages: 1 + 132

Authentication transposition means, basically, to allow access to secure resources at a domain distinct from that client, using credential provided at your original domain. However, it's considerably hard to conclude those systems because of the lack of interoperability between protocols, programming languages, operational platforms, used in the project, and the distinct underlying security infra-structure. Also, the computer security is a crucial requirement in this scenario.

Though Web Services is the more indicated technology to enfold the challenges of transposition, the large amount of specifications and patterns, designated to Web Services, imposes a certain level of complexity and makes it harder to adopt this technology. Some propositions are still immature and need more study and implementation that approaches the solutions as a whole. This will allow a good understanding of the potential and limitation of the technology.

For the users' identity to be acknowledged in various administration domains, managing federated domains becomes a promising solution. Besides, because of the Web Services environments dynamic, which services usually are aggregated temporarily to attend a specific business flow, identity managing is cited more and more in the literature.

TThe purpose of this thesis is to propose a model for authentication transposition - and the consequent decentralized authorization and transposition of the client attributes. In this model, each domain groups clients and services providers according to the security technology in use. In the model definition, Web Services patterns and federated identity management concepts are used. A prototype is implemented in order to verify the applicability model.

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

AC Autoridades Certificadoras
AES Advanced Encryption Standard
AOS Arquitetura Orientada a Serviço
ASN.1 Abstract Syntax Notation One
B2B Business to Business
CORBA Common Object Request Broker Architecture
DCOM Distributed Component Object Model
DNS Domain Name System
EJB Enterprise JavaBean
HTTP HyperText Transfer Protocol
ICP Infra-estrutura de Chave Pública
IDL Interface Definition Language
IdP Identity Provider
IIOP Internet Inter-Orb Protocol
JMS Java Messaging Service
LDAP Lightweight Directory Access Protocol
MA Manipulador de Assinatura
MC Manipulador de Cifragem
MCA Manipulador de Controle de Acesso
MP Motor de Política
MPP Manipulador de Política de Proteção
OASIS Organization for the Advancement of Structured Information Standards
OSF Open Software Foundation
PDP Policy Decision Point
PEP Policy Enforcement Point
PGP Pretty Good Privacy
PKCS#7 Public Key Cryptography Standard #7
PIP Policy Information Point **RMI** Remote Method Invocation
RPC Remote Procedure Call
SAML Security Assertion Markup Language
SAP Serviço de Atributos e de Pseudônimos
SDSI Simple Distributed Security Infrastructure
SHA-1 Secure Hash Algorithm 1
SHAR SHibboleth Attribute Requester
SHIRE SHibboleth Indexical Reference Establisher
SMTP Simple Mail Transfer Protocol
SOA Service Oriented Architecture

SOAP Service Oriented Access Protocol
SPKI Simple Public Key Infrastructure
SSL Secure Sockets Layer
SSO Single Sign-On
STS Security Token Service
TCB Trusted Computing Base
TCSEC Trusted Computer System Evaluation Criteria
TCP Tranport Control Protocol
TLS Transport Layer Security
TPC Terceira Parte Confiável
UDDI Universal Description, Discovery and Integration
URI Uniform Resource Identifier
URL Uniform Resource Locator
WAYF Where Are You From?
WS Web Services
WSDL Web Services Description Language
WSS4J Web Service Security for Java
XACML eXtensible Access Control Markup Language
X-KISS XML Key Information Service Specification
XKMS XML Key Management Specification
X-KRSS XML Key Registration Service Specification
XML eXtensible Markup Language
XMLDSign XML Signature
XMLEnc XML Encryption
XSD XML Schema Definition

Capítulo 1

Introdução

A demanda pelo compartilhamento das informações de forma transparente e segura tanto para usuários quanto para sistemas é uma exigência cada vez mais atual, principalmente no contexto da Internet. Aliado a isto se vislumbra o cenário onde as informações do usuário, obtidas originalmente no seu domínio, são aceitas em domínios distintos, ou seja, um cenário onde as propriedades de "transposição" de autenticação e de autorização estejam presentes.

A idéia da transposição inclui o acesso a recursos controlados em um domínio diferente daquele do cliente, usando atributos e credenciais fornecidos no seu domínio de origem. Isto inclui o requisito de que principais se autentiquem apenas uma vez (*Single Sign On*) e usufruam desta nos demais domínios de um sistema distribuído complexo. No entanto, grandes são os empecilhos para a concretização de tais sistemas e vão desde a falta de interoperabilidade entre protocolos, linguagens de programação, plataformas operacionais, utilizadas no seu projeto, até as diferentes infra-estruturas de segurança subjacentes. Além disso, a segurança computacional (*computer security*) se apresenta como um requisito indispensável neste cenário. É através da segurança computacional que o acesso e o compartilhamento de informações se dará de forma segura, garantindo as propriedades de autenticidade, integridade e confidencialidade.

A tecnologia de Serviços *Web* (*Web Services*), baseada na arquitetura orientada a serviço (AOS), é uma das promissoras soluções para a integração de sistemas. Sua característica de fraco acoplamento e o uso de protocolos padronizados visa garantir a independência de plataforma e de linguagem de programação, indispensáveis para a comunicação de sistemas distribuídos distintos através da Internet. A capacidade de atravessar os *firewalls*, e assim os domínios administrativos, também é um forte atrativo para a adoção desta tecnologia. No entanto, algumas dificuldades permanecem. Segundo Vogels (Vogels, 2004), a segurança fim a fim é um problema para Serviços *Web*. O uso de tecnologias como o protocolo SSL (*Secure Socket Layer*) não garante segurança fim a fim nas trocas de mensagens quando há a presença de nós intermediários. Além disso, perante diferentes infra-estruturas de segurança (ICP X.509, SPKI, Kerberos) a tecnologia se mostra como limitada, uma vez que as especificações

se concentram principalmente em uma autenticação baseada em usuário/senha e ICP X.509. No sentido de atacar tais problemas, um grande conjunto de especificações de segurança para Serviços *Web*, em sua maioria, ainda estão em desenvolvimento ou foram recentemente lançadas. Apesar das especificações de segurança proporem soluções para os problemas de segurança, estas agregam complexidade à tecnologia.

Devido à dinâmica do ambiente dos Serviços *Web*, no qual serviços muitas vezes são agregados de forma temporária para atender um determinado fluxo de negócios, o gerenciamento de identidade desponta como uma solução promissora (de Mello, 2005). No gerenciamento de identidades federadas, cada empresa constrói um domínio, onde estão presentes os seus provedores de serviço, de identidades e de credenciais (Jøsang and Pope, 2005; Jøsang *et al.*, 2005). Os acordos estabelecidos entre os domínios permitem que identidades locais a um domínio sejam aceitas nos demais domínios participantes do acordo. Segundo Mello (de Mello, 2005), a idéia por trás de criar identidades federadas é basicamente permitir que o usuário, com uma identidade registrada em seu domínio, acesse recursos em outros domínios da federação, sem abertura de novo registro (e de nova identidade) no domínio que detém o recurso.

1.1 Motivação

Concretizar a autenticação distribuída, autorização descentralizada e troca de atributos do cliente de forma transparente em sistemas distribuídos de larga escala é uma tarefa muito árdua, diretamente afetada pela escalabilidade, e que exige grande cooperação entre domínios administrativos distintos e autônomos, principalmente a fim de garantir a interoperabilidade entre domínios. São necessários tanto padrões altamente difundidos, com abstrações suficientes para esconder as diferentes tecnologias, quanto modelos que contribuam para integração de tais tecnologias. Nesse trabalho, o foco principal é interoperabilidade entre diferentes infra-estruturas de segurança.

Embora Serviços *Web*, devido suas características, se configure como a tecnologia mais indicada para contornar os desafios da transposição, o grande número de especificações e padrões, destinados a Serviços *Web*, tanto impõe um certo nível de complexidade como dificulta a sua adoção. Algumas especificações e propostas ainda são imaturas e carecem de um estudo e implementação que abordem as soluções como um todo para uma boa compreensão do potencial e das limitações da tecnologia.

Este trabalho parte do pressuposto que as relações de confiança se encontram estabelecidas e assim, constrói um modelo para a transferência de autenticação e a conseqüente autorização descentralizada e a transposição de atributos do cliente. Outro trabalho do grupo, desenvolvido em (de Mello, 2005) preocupa-se com o estabelecimento dinâmico da confiança.

Diante da proposta de fazer uso de Serviços *Web* e do gerenciamento de identidades federadas para atingir a transposição de autenticação entre domínios que possuem diferentes infra-estruturas de segurança, algumas questões de pesquisa surgem indicando os desafios a serem enfrentados:

- Quais são as reais dificuldades em atingir a transposição entre domínios e como superá-la?
- É possível a interação entre um cliente e um provedor de serviços que pertencem a diferentes infra-estruturas de segurança?
- A tecnologia de Serviços *Web* cumpre com suas promessas de tornar a comunicação entre partes distintas interoperável, segura, aumentando as relações comerciais?
- Como deve ser o modelo para concretizar a transposição de autenticação e a conseqüente autorização distribuída?
- É possível avaliar este modelo sem conceber um protótipo, sendo este flexível, eficiente e de fácil configuração e uso para o projeto e implementação de aplicações seguras baseadas na Arquitetura Orientada a Serviços? Quais tecnologias devem ser utilizadas para construir o protótipo de forma a oferecer as características desejadas?

Estas questões de pesquisa motivaram a presente dissertação e orientaram a formulação de seus objetivos descritos na seção a seguir.

1.2 Objetivos

O objetivo deste trabalho é propor um modelo para a transposição de autenticação. Neste modelo, cada domínio agrupa clientes e provedores de serviço de acordo com a tecnologia de segurança usada. A autorização é local ao provedor de serviços e descentralizada. Na definição do modelo, parte-se de proposições consolidadas. São assumidos, na construção do modelo, padrões relacionados a tecnologias de Serviços *Web* e os conceitos de gerenciamento de identidades federadas.

Comprovar a aplicabilidade deste modelo em ambientes distribuídos de larga escala com a adoção da tecnologia orientada a serviços também é intenção deste trabalho. A partir das perspectivas citadas, o projeto de pesquisa perseguiu os seguintes objetivos específicos:

- definir um modelo para a transposição de autenticação apoiado na tecnologia de Serviços *Web* e no gerenciamento de identidades federadas;
- obter a interoperabilidade perante as diferentes infra-estruturas de segurança de forma transparente a clientes e provedores de serviços;

- verificar o potencial e as limitações da arquitetura de Serviços *Web* na construção do modelo;
- definir e implementar um protótipo que inclui o modelo proposto.

1.3 Organização do Texto

Esta dissertação está estruturada em sete Capítulos. Este capítulo descreveu o contexto geral do trabalho, a motivação e os objetivos. Os demais Capítulos estão dispostos da seguinte forma:

O Capítulo 2 apresenta os conceitos fundamentais relacionados a segurança computacional. Em um primeiro momento, apresenta-se os conceitos da segurança, com suas propriedades e as suas possíveis violações e ataques contra a segurança. Em seguida, as políticas e os modelos de segurança. Os mecanismos de segurança são brevemente analisados e, por fim, são apresentados os mecanismos de autenticação e autorização em sistemas distribuídos, juntamente com os estudos de casos relacionados as suas abordagens.

O Capítulo 3 apresenta os conceitos e as características da Arquitetura Orientada a Serviço e, em seguida, descreve a arquitetura de Serviços *Web*, com os padrões que formam a sua base. As especificações de segurança que objetivam tornar a tecnologia confiável, agregando diversos requisitos de segurança, são discutidos na seqüência. Por fim, descreve-se as preocupações em manter a interoperabilidade entre as diversas especificações de Serviços *Web*.

O Capítulo 4 tem por objetivo apresentar uma breve introdução aos conceitos relacionados ao gerenciamento de identidades e destacar a importância do estabelecimento das relações de confiança em Serviços *Web* para concretizar as relações de negócios. Além disso, este capítulo apresenta os esforços da literatura para atingir a transposição de autenticação - e também de autorização - através do conceito de identidades federadas e/ou para o estabelecimento da confiança entre Serviços *Web*.

O capítulo 5, apoiado na tecnologia de Serviços *Web*, apresenta o modelo proposto cuja função é compor um suporte para transposição em sistemas distribuídos complexos, que fazem uso da Internet na comunicação. Apoiado na identidade do usuário, através do conceito de identidades federadas, este modelo concretiza a transposição de autenticação no sistema como um todo mantendo propriedades de transparência e fácil uso nos acesso.

O Capítulo 6 descreve a implementação de um protótipo, que objetiva verificar a aplicabilidade do modelo, e descreve as ferramentas utilizadas na sua construção.

O Capítulo 7 apresenta as conclusões sobre esta dissertação, apresentando sugestões para trabalhos futuros.

Capítulo 2

Segurança Computacional

2.1 Introdução

Há algumas décadas atrás, proteger um computador era uma tarefa relativamente fácil se comparada à realidade atual. Estes eram grandes e tão poucos que simplesmente limitar o acesso físico já promovia um alto nível de segurança. Além disso, as redes de computadores e os sistemas distribuídos ainda não haviam se tornado uma realidade, o que tornava o compartilhamento das informações, as interações entre as máquinas e entre máquinas e usuários uma tarefa mais complexa. Atualmente, a situação é outra, computadores são encontrados em diversos lugares, tais como: casas, empresas, carros, pequenos dispositivos, etc. Com o surgimento da Internet e sua grande evolução, computadores constantemente comunicam-se entre si. Desta forma, é possível ter fácil acesso a informação, realizar negócios, compras, vendas de produtos, etc.

Ao mesmo tempo em que a evolução dos computadores trouxe benefícios, anteriormente inimagináveis, a segurança computacional (*computer security*) desperta como um requisito indispensável aos sistemas distribuídos. É através da segurança computacional que o acesso e o compartilhamento de informações se dará sem comprometer o sistema, seu funcionamento e sem revelar informações a usuários mal intencionados.

Em (Landwehr, 2001) encontra-se uma classificação relativa a segurança computacional, em termos de mecanismos, que ajuda a vislumbrar o avanço que a mesma tem alcançado. Nessa classificação, a segurança é dividida em 3 gerações. A primeira, onde os sistemas preocupados com a segurança eram de característica militar, limitava-se a prevenir violações de segurança. Exemplos são os uso de criptografia para proteger as informações contra pessoas mal intencionadas ou não autorizadas. A segunda geração de tecnologias de segurança, caracterizada pelo uso de *firewalls* e detecção de intrusão, não apenas prevenia, mas detectava e limitava as violações de segurança. Para a terceira geração, os esforços se concentram em tecnologias e arquiteturas que terão a habilidade de tolerar ataques.

Neste capítulo, aborda-se os conceitos fundamentais envolvidos na segurança. Em um primeiro momento, apresenta-se os conceitos fundamentais relacionados a segurança computacional, com suas propriedades e as possíveis violações e ataques contra a segurança. Em seguida, as políticas e os modelos de segurança, que fornecem as regras práticas e estabelecem os limites de operações dos usuários do sistema, são brevemente analisados. Os mecanismos de segurança, que concretizam a política de segurança, são colocados a seguir. Por fim, são apresentados os mecanismos de autenticação e autorização em sistemas distribuídos, juntamente com os estudos de casos relacionados as suas abordagens.

2.2 Conceitos Fundamentais

2.2.1 Propriedades da segurança

Na literatura, existem várias definições para segurança (*security*) e, em quase todas, esta é caracterizada como a necessidade de se manter no sistema um conjunto de propriedades. As seguintes propriedades são encontradas em (Bishop, 2003) e (Landwehr, 2001):

- **confidencialidade**: assegura que a informação computacional não será revelada sem autorização;
- **integridade**: assegura a não modificação indevida, causada intencionalmente ou acidentalmente, por usuários que não possuam direito para tal;
- **disponibilidade**: assegura que usuários legítimos terão acesso a informação e a recursos;

Outras duas propriedades de segurança, recentemente, tem sido adicionadas à definição clássica. São estas:

- **autenticidade**: garante que cada principal é quem diz ser;
- **não-repudiação**: garante que o participante de uma comunicação não possa negá-la posteriomente.

Sistemas em que essas propriedades são totalmente garantidas são difíceis de encontrar, principalmente nos sistemas de larga escala. Uma das dificuldades nesse sentido se dá pelo compartilhamento das informações. A informação compartilhada entre partes diversas abre possibilidades muitas vezes não previstas pelo administrador do sistema.

2.2.2 Violações e Ataques de Segurança

As definições de alguns conceitos são importantes antes de falar propriamente das violações e ataques de segurança. Entende-se por **risco** a medida do custo de uma vulnerabilidade que incorpora a probabilidade de uma violação ocorrer. A **vulnerabilidade** sempre este presentes em sistemas computacionais, sendo considerada uma característica indesejável (faltas ou defeitos) que pode ser explorada para concretizar uma violação de segurança. Um erro de programação, um erro na configuração ou mesmo um erro de operação, pode 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, 2003; Landwehr, 2001).

A **ameaça** é qualquer circunstância ou evento que forneça algum potencial de violação de segurança. Já o **ataque** é a ação tomada por um intruso [1] malicioso que envolve a exploração de certas vulnerabilidades, visando violações de segurança (Landwehr, 2001).

Quando os objetivos de segurança não são alcançados há uma violação da segurança, ou seja, uma revelação não autorizada ou modificação da informação ou, ainda, a negação de um serviço. A violação na segurança é obtida por meio de ataques de segurança. A Figura 2.1 ilustra, como exemplo, as quatro categorias de ataques normalmente identificados em sistemas distribuídos, segundo (Stallings, 2000). A princípio, considera-se o fluxo normal de informação, com uma máquina A se comunicando com uma máquina B (Figura 2.1 a).

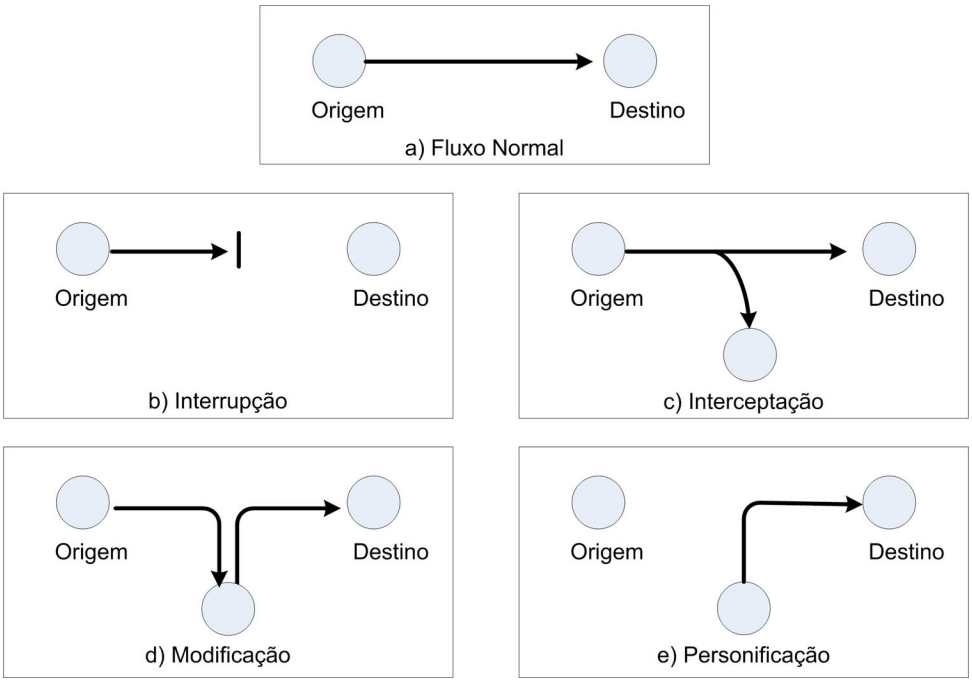

Figura 2.1: Categorias de Ataques, (Stallings, 2000)

- **interrupção**: caracteriza-se pela interrupção do fluxo da informação, tornando a comu-

[1] Refere-se às entidades não autorizadas pela política.

nicação indisponível ou o seu uso inviável. Esse é um ataque sobre a disponibilidade. Exemplos incluem a destruição de uma peça de hardware, como um disco rígido ou o corte da linha de comunicação.

- **interceptação**: uma parte não autorizada ganha acesso a informação. Compromete a confidencialidade, pois a informação é lida por pessoas indevidas. Exemplos são a captura de dados em uma rede.
- **modificação**: além de conseguir acesso a informação, uma parte não autorizada modifica a informação. Este é um ataque sobre a integridade. Exemplos são a mudança dos dados em um arquivo ou modificação na mensagem sendo transmitida em uma rede.
- **fabricação**: uma terceira parte insere dados falsos no sistema, podendo se passar por uma parte confiável. Este é um ataque sobre a autenticidade.

Os ataques são ainda classificados em ativos ou passivos, segundo (Stallings, 2000). A Figura 2.2 ilustra os tipos de ataques:

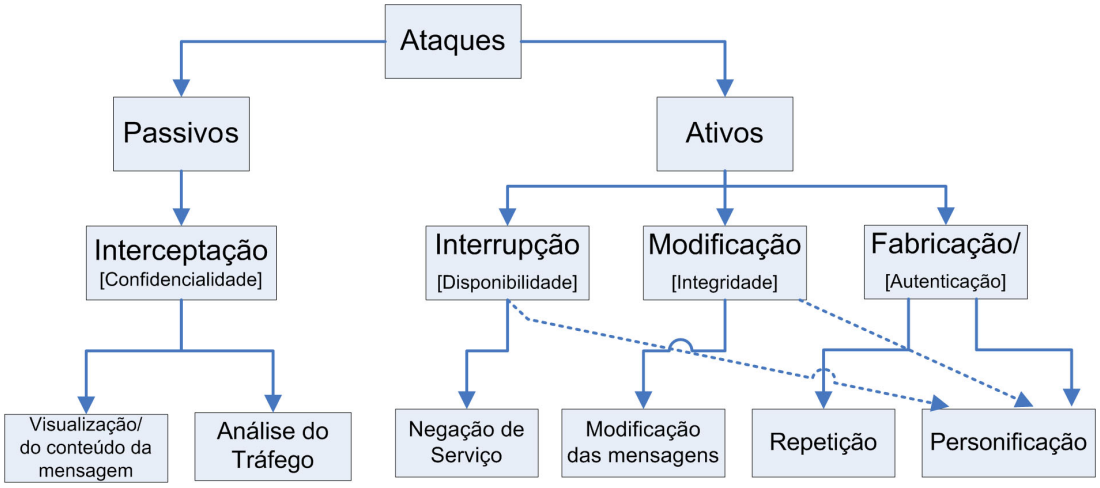

Figura 2.2: Categorias de Ataques (Stallings, 2000)

Os ataques ativos são geralmente divididos em quatro categorias. Estes envolvem alguma modificação ou criação de um fluxo de dados, ou seja, ameaçam a integridade e/ou disponibilidade da informação. Alguns exemplos desses ataques são:

- **personificação** (*masquerade/spoofing*): enviar ou receber mensagens usando a identidade de outro principal, sem sua autorização;
- **repetição ou mensagens antigas** (*replaying*): as mensagens interceptadas são armazenadas e enviadas posteriormente. Este ataque pode ser realizado mesmo quando a comunicação faz uso de autenticação e mensagens cifradas;
- **mensagens modificadas** (*message tampering*): é a interceptação e alteração das mensagens antes que estas cheguem ao seu destino. O ataque do homem do meio (*man-in-the-middle*) é uma forma de produzir tal ataque. Neste caso, um atacante intercepta

as primeiras mensagens da troca das chaves criptográficas e as substitui por chaves conhecidas que o habilitam a decifrar as mensagens subseqüentes antes de cifrá-las novamente com a chave correta e passá-las adiante;

- **negação de serviço** (*denial of service*): inundação de um canal de comunicação ou outro recurso com mensagens para negar acesso a outros.

Os ataques passivos ameaçam a confidencialidade. A monitoração de uma linha de comunicação é o exemplo de um ataque passivo. Estes podem ser categorizados de duas formas:

- **visualização ou divulgação do conteúdo da mensagem**: geralmente a escuta da comunicação é feita com um software *sniffer*. Nesse caso, informações importantes podem ser visualizadas por pessoas indevidas;
- **análise de tráfego**: saber que tipo de dados e a quantidade de informação que a rede transporta pode ser útil para ataques de negação de serviço.

Muitos outros tipos de ataques podem ser encontrados na literatura. O sucesso de tais ataques, na maioria das vezes, está condicionado ao descobrimento de entradas na segurança do sistema. Atualmente, é comum os ataques explorarem várias deficiências nas configurações dos sistemas. A CERT[2] (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil) fornece várias informações em seu sítio para prevenir e detectar tais incidentes.

2.3 Políticas e Modelos de segurança

2.3.1 Políticas

Políticas de segurança, segundo (Landwehr, 2001), fornecem as "regras do jogo" em segurança computacional. Uma política é um conjunto de regras que especifica como um sistema provê os seus serviços, estabelece os limites de operação dos usuários do sistema e determina a maneira pela qual as informações e os recursos são administrados, protegidos e distribuídos no interior de um sistema específico (15408-1:1999, 1999).

A política de segurança pode ser vista sob o aspecto físico, administrativo e lógico, segundo (Nicomette, 1996). A primeira está relacionada ao ambiente físico a ser protegido (incêndios, desastres, etc). A segunda se preocupa com os aspectos organizacionais, como a seleção de pessoas. Já a segurança lógica define as regras de acesso e de circulação das

[2] http://www.cert.br/

informações e define os controles para o acesso lógico às informações, especificando "quem" tem acesso à "que" e em "quais" circunstâncias.

A política de segurança lógica pode ser decomposta em política de identificação e política de autorização. O usuário necessita se identificar no sistema e, a partir de então, será autorizado a realizar as suas atividades segundo a descrição de sua política. A política de autorização é também chamada de política de controle de acesso por (Sandhu and Samarati, 1994). É importante lembrar que as políticas de segurança são independentes da tecnologia usada e são aplicadas via mecanismos de segurança, conforme será visto na Seção 2.4.

O termo **principal** se aplica as entidades responsáveis e autorizadas pela política de segurança para obter acesso a informação mantida pelo sistema, se refere a usuários, processos, máquinas atuando em nome dos usuários de um sistema.

2.3.2 Modelos de segurança

Os modelos de segurança correspondem a descrições formais do comportamento de um sistema atuando segundo regras de uma política de segurança. Segundo (Landwehr, 1981), os modelos formais de segurança permitem, de certa maneira, verificar se a política é coerente e serve como guia para implementações que correspondam às especificações contidas no modelo. Além disso, são os modelos que permitem demonstrar a corretude de uma política de segurança.

Os modelos são geralmente classificados em três tipos básicos: discricionários (*discretionary*) ou baseados em identidade (*identity-based*), obrigatórios (*mandatory*) ou baseado em regras e os baseados em papéis (*roles*).

2.3.2.1 Modelos discricionários

Este modelo sustenta a idéia de que é o proprietário da informação que livremente atribui a ele mesmo, a um grupo de usuários ou a outros usuários o direito (ler, escrever, executar) sobre a informação (arquivo, programa, etc).

O modelo de matriz de acesso é um modelo conceitual discricionário introduzido por (Lampson, 1974) como uma generalização dos mecanismos de proteção usualmente fornecidos por sistemas operacionais. Este modelo é utilizado para descrever e implementar uma grande variedade de mecanismos de segurança. Neste modelo, o estado de proteção do sistema é representado através de uma matriz, onde as linhas correspondem aos sujeitos[3] (principais) e as colunas correspondem aos objetos[4] do sistema. Uma célula M*ij* representa os direitos de acesso do sujeito *i* ao objeto *j*. A tabela 2.1 ilustra a matriz de acesso.

[3]Uma entidade ativa em um sistema computacional que inicia requisições por recursos; corresponde, via de regra, a um usuário ou um processo executando em nome de um usuário.

[4]É uma entidade passiva que armazena informações no sistema, como arquivos, diretórios e segmentos de memória.

Tabela 2.1: Exemplo de Matriz de Acesso

	Objeto 1	**Objeto 2**	**Objeto 3**
Sujeito 1	proprietário, ler, escrever	ler	ler, escrever
Sujeito 2	ler	...	proprietário, ler, escrever
Sujeito 3	escrever, ler	proprietário, ler, escrever	...

Na Tabela acima, o Sujeito 1 é o proprietário do objeto 1. Sendo assim pode ler e escrever no mesmo. O Sujeito 1 permite que o Sujeito 2 apenas leia o objeto 1; e ao Sujeito 3 permite a leitura e a escrita. Por sua vez, o proprietário do objeto 2 é o Sujeito 3, que atribui ao Sujeito 1 a permissão de ler o seu objeto. O objeto 3 é de propriedade do Sujeito 2, que atribui ao Sujeito 1 a permissão de ler e escrever.

Uma matriz de acessos não é fixa, mas evolui no tempo em função da criação de novos objetos e sujeitos. A evolução da matriz de acesso no tempo é definida em termos de estados de segurança e de regras de transição. A matriz de acesso corresponde à representação do seu estado atual de segurança no sistema. As mudanças de estado de um sistema são realizadas usando regras de transição do modelo. Essas regras são tipicamente: remover objeto/sujeito, criar objeto/sujeito, transferir direitos e suprimir direitos.

Como pode ser observado, a matriz de acesso pode atingir tamanhos consideráveis. Apresentando células vazias e dificultando sua implementação. Obelheiro, em (Obelheiro, 2001), descreve estratégias de implementação da matriz de acesso, que otimizam o seu uso.

2.3.2.2 Modelos obrigatórios

O modelo obrigatório baseia-se na administração centralizada da segurança, a qual dita regras incontornáveis de acesso à informação (Amoroso, 1994). Inicialmente proposto pelo Departamento de Defesa dos EUA (*Departament of Defense* - DoD), é um modelo pioneiro na segurança computacional. Surge da idéia de atribuir classificações à informações sigilosas, definindo assim **níveis de segurança**, ordenados segundo uma hierarquia. Os níveis mais usuais, ordenados em ordem crescente de importância, ou sensibilidade, são definidos como: NÃO-CLASSIFICADO, CONFIDENCIAL, SECRETO E ULTRA-SECRETO.

Além dos níveis de segurança, são definidas **categorias de segurança**, ou compartimentos de segurança, que correspondem a diferentes projetos, setores ou departamentos. Os indivíduos têm acesso a diferentes categorias na medida em que suas incumbências demandem este acesso. Assim, por exemplo, professores de um departamento não devem ter acesso a informação classificada como ULTRA-SECRETO pertencentes ao reitor da universidade.

Os reticulados (lattice[5]) são utilizados para expressar formalmente esse modelo, permitindo assim que estes sejam corretamente analisados e tenham sua segurança demonstrada.

No contexto computacional, um **rótulo de segurança** (*security label*) é um atributo que denota a sensibilidade das entidades em um sistema. Quanto atribuído a um objeto o rótulo recebe o nome de **classificação** do objeto. Já quando se refere a um sujeito, é chamado de **habilitação** (*clearance*) do sujeito.

Os modelos de fluxo de informação de (Denning, 1976) e os modelos de Bell e Lapadula e Biba (Landwehr, 1981) são descrições de modelos obrigatórios.

2.3.2.3 Modelos baseados em papéis

Segundo (Sandhu and Samarati, 1994), o modelo baseado em papéis, também chamadas de RBAC(*Role-Based Access Control Models*), regula o acesso dos usuários aos recursos com base nas atividades que estes desempenham no sistema. Os papéis podem ser definidos como um conjunto de ações e responsabilidades associadas com uma atividade particular. Desta forma, a atribuição de direitos de acesso é destinada a papéis (*roles*) e não a usuários, e os usuários recebem permissão para assumir e fazer parte de um ou mais papéis.

Segundo (Sandhu and Samarati, 1994, 1997) os modelos baseados em papéis possuem diversas características importantes, tais como:

- **Gerência de autorização mais simples**: a especificação de autorização é dividida em duas partes: associação de direitos a papéis e associação de direitos a usuários. Desta forma, a gerência de segurança se torna mais simples, pois facilita ajustar os direitos de acesso de um usuário em função de uma promoção ou transferência de setor na organização;
- **Suporte a hierarquia de papéis**: em muitas aplicações existe uma hierarquia natural de papéis baseada nas noções de generalização e especialização. Isto permite que permissões sejam herdadas e compartilhadas através da hierarquia;
- **Suporte a mínimo privilégio**: os papéis permitem que um usuário trabalhe com o mínimo de privilégio exigido para uma determinada tarefa. Usuários autorizados a exercer papéis poderosos só precisam exercê-lo quando forem absolutamente necessários, minimizando a possibilidade de danos por causa de erros inadvertidos;
- **Suporte a separação de tarefas**: os modelos baseados em papéis suportam separação de tarefas. Nestes modelos, a separação de tarefas é obtida através de restrições à autorização e/ou à ativação de papéis considerados mutuamente exclusivos;

[5]Corresponde a um conjunto matemático de elementos ou recursos do sistema parcialmente ordenados (relação transitiva e assimétrica) tal que, para qualquer dois elementos, existe um elemento que é maior no subconjunto de todos os elementos menores ou iguais a ambos e, um elemento que é o menor no subconjunto de todos os elementos que são maiores ou iguais a ambos.

- **Delegação da administração de segurança**: permitem que a administração da segurança seja descentralizada de maneira controlada. Isto significa que o administrador de segurança pode delegar parte de suas atribuições de acordo com a estrutura organizacional ou com a arquitetura do sistema computacional, permitindo, por exemplo, que administradores regionais gerenciem a segurança dos subsistemas locais.

2.4 Mecanismos de Segurança

Para assegurar que os sistemas implantem as políticas de segurança e sejam ditos seguros, existe a necessidade da adoção de mecanismos de segurança. Os mecanismos de segurança são os responsáveis efetivos pela garantia das propriedades e política de segurança. Para tanto, fazem uso de autenticação, autorização (ou controle de acesso) e controles criptográficos. A autenticação e a autorização serão vistas na Seção 2.5 e os controles criptográficos na Seção 2.4.1.

Amoroso (Amoroso, 1994) também cita controles adicionais de segurança, que mesmo não atuando diretamente nas requisições de acesso devem estar presentes no sistema. São estes: auditoria de vestígio, auditoria de segurança e detecção a intrusão. O primeiro está ligado à geração periódica de registros de eventos associados à segurança. Estes são coletados para uso potencial em detecção de intrusão e/ou auditoria de segurança. O segundo se refere à inspeção independente (por terceiros) dos procedimentos e registros do sistema como intuito de verificar a adequação da política de segurança e as possíveis violações do sistema. Por fim, a detecção de intrusão usa os registros da auditoria em métodos automatizados para analisar, em tempo real, se há atividades anormais no sistema.

2.4.1 Controles Criptográficos

Segundo (Stallings, 2000), os controles criptográficos fornecem a base para a maioria dos mecanismos de segurança. Desde sua origem militar, na década de 60, vem ganhando cada vez mais atenção no meio público. Um sistema criptográfico é constituído pela cifragem e decifragem. A cifragem (*encryption*) é o processo de codificar uma mensagem para ocultar o seu conteúdo. Já a decifragem (*decryption*) é aplicada sobre os dados cifrados para torná-los entendíveis, ou em claro, novamente. A criptografia atual fornece muitos algoritmos para a cifragem e decifragem. Todos estes são baseados no uso de segredos, chamados chaves. Uma chave criptográfica é um parâmetro usado em um algoritmo de cifragem de modo que a revelação do conteúdo só se faz perante o conhecimento da chave.

Há duas classes de algoritmos de cifragem: **simétricos** e **assimétricos** (ou algoritmos de chave pública). O primeiro compartilha a mesma chave para cifrar e decifrar a mensagem. A segunda classe usa uma chave pública para cifrar a mensagem e uma chave privada para

decifrá-la. A chave pública pode ser distribuída livremente, porém o conhecimento da chave privada deve se restringir ao seu proprietário. Em relação ao tempo de processamento, os algoritmos assimétricos requerem de 100 a 1000 vezes processamento que os simétricos, porém levam enorme vantagem em relação à distribuição das chaves (Coulouris *et al.*, 2001).

Os controles criptográficos provêem as propriedades de confidencialidade às informações, autenticidade de mensagens e de principais, integridade às informações armazenadas ou transmitida; e não repudiação. O processo de cifragem é que garante a confidencialidade. A autenticidade é alcançada via técnica de **assinatura digital**. Vale ressaltar que a assinatura digital baseada em chave pública vem sendo amplamente utilizada. Na assinatura digital, um resumo da mensagem (um *hash*) é cifrado usando a chave privada do emissor. Essa assinatura é enviada junto com a mensagem. Para verificar a assinatura, é preciso decifrar a assinatura com a chave pública do emissor e recalcular o *hash* da mensagem. Se a comparação dos resumos (o cifrado e o calculado) for igual, a assinatura será autêntica, preservando assim a sua integridade e impedindo que posteriormente o emissor negue o envio da mensagem (não repudiação) (Wangham, 2004).

Segundo (Stallings, 2000), em relação ao uso de chaves públicas, um problema encontrado é justamente na origem pública da chave pública. Assim, em um algoritmo RSA, por exemplo, qualquer participante pode enviar sua chave pública para outros participantes da comunidade. Embora isso pareça conveniente, há um grande problema: qualquer um pode falsificar um aviso público. Isto é, algum usuário poderia fingir ser um usuário A, por exemplo, e enviar a sua chave pública a outros. Até que A descubra a falsificação e avise aos outros participantes, o falsificador está habilitado a ler todas as mensagens codificadas para A e usar a chave falsa para autenticação. Uma das soluções para este problema é o **certificado digital** que vincula chaves públicas a principais. Para ter certeza que a chave pública no certificado pertence ao sujeito do certificado, o certificado é assinado por uma entidade confiável, chamada de Autoridade Certificadora (*Certificate Autority* - CA), que é confiável a todos os usuários da comunidade, tal como uma agência de governo ou uma instituição de finanças.

Tipicamente, em uma Infra-estrutura de Chave Pública (ICP)[6], pode haver dois tipo de certificados - um certificado de nomes e um certificado de autorização. O certificado de nomes liga uma chave pública a um nome. Já o certificado de autorização associa uma chave pública a atributos e significa uma autorização ou privilégio que tem significado ao assinante e a alguns principais.

Vários certificados estão em utilização. Alguns deles, como o *Pretty Good Privacy* (PGP) (Zimmerman, 1994), são patenteados. Outros certificados populares são específicos de um aplicativo como certificados usados no SET (*Security Electronic Transaction*) (Secure Electronic Transaction LLC, 1997) e no *Internet Protocol Security* (IPSec) (Markham, 1997). O

[6]Um PKI (*Public Key Infrastructure*) envolve um processo colaborativo entre várias entidades: a AC, uma autoridade de registro (*registration authority* - RA), um repositório de certificados, um servidor de recuperação de chaves e o usuário final.

certificado mais amplamente aceito é o que segue o padrão X.509 (Housley *et al.*, 2002). Há ainda propostas, como o SPKI/SDSI (Ellison, 1999) que visam retirar a complexidade imposta pelo padrão X.509.

2.5 Autenticação e Autorização em Sistemas Distribuídos

A **autenticação** é o processo de identificação de um principal perante um sistema (Bishop, 2003). Por meio desse processo é necessário que o usuário ou a entidade forneça uma prova de identificação, que pode ser manifestada através da posse de alguma informação, do conhecimento de um segredo ou ainda, de uma característica única (íris, impressão digital, etc).

Em sistemas distribuídos, onde podem ocorrer interações entre diversas máquinas, um possível cenário é imaginar que um usuário U, autenticado na maquina A, deseja ter acesso a uma máquina B, ou ainda, que a máquina A precisa trocar mensagens com a máquina B em nome do usuário U. Nesse caso, identifica-se duas situações:

- o usuário U se autentica novamente através da rede no sistema B ou;
- o sistema A envia informações de identificação do usuário pela rede ao sistema B.

Segundo (Stallings, 2000), em ambos os casos há a necessidade de controles criptográficos, como visto na Seção 2.4.1, para proteger a informação em trânsito na rede. Além disso, no primeiro caso, podemos imaginar a dificuldade do usuário em ter que se autenticar em diversas máquinas, sendo que em cada autenticação ele estaria enviando a sua senha pela rede. O segundo caso é mais simples, mas, os sistemas A e B deveriam compartilhar algum tipo de "segredo".

O caso 2 também pode ser resolvido com a utilização de uma terceira entidade (*trust third party*), ou seja, o serviço de autenticação promovendo uma relação de confiança. Partindo deste ponto, o servidor de autenticação é o fornecedor de informações necessárias na autenticação de dados sobre o principal como, por exemplo, através de certificados.

O uso de certificados evita normalmente a necessidade de mecanismos específicos para proteger informações de autenticação em trânsito pela rede, porque os mesmo envolvem assinatura digital ou outro mecanismo gerado pelo certificador, garantindo a integridade e a autenticidade do certificado. As mensagens trocadas também devem ser autenticadas para evitar que mensagens indevidas influenciem no sistema.

A **autorização**, ou controle de acesso, é um processo que decide se as requisições de acesso a objetos feitos por sujeitos devem ser ou não permitidas. Ou seja, é um processo

onde o objetivo é garantir que só tenham acesso aos recursos os sujeitos que tenham a legitimidade de fazê-lo. Tal procedimento é executado pelo monitor de referência (Figura 2.3) - introduzido por Anderson (Anderson, 1972) - que permite ou nega a tentativa de acesso a um destes objetos. Os monitores de referência atuam em vários níveis do sistema. As referências a segmentos de memórias são validadas nas camadas inferiores do sistema, por meio de *hardware* e do sistema operacional; por sua vez o serviço de diretório valida os acessos a arquivos.

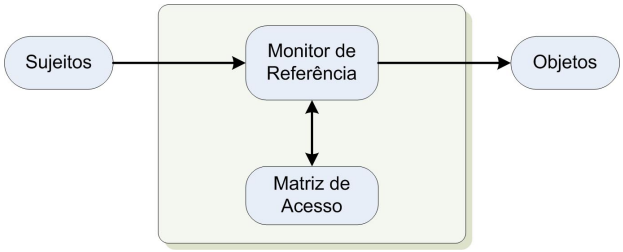

Figura 2.3: Monitor de Referência

O monitor de referência é implementado por mecanismos de controle que podem ser discricionários (DAC - *Discretionary Access Control*), obrigatórios (MAC - *Mandatory Access Control*) ou baseados em papéis (RBAC - *Role-Basead Access Control*), que são baseados nos modelos vistos na Seção 2.3.2.

A implementação dos mecanismos de autenticação e de autorização em sistemas distribuídos é uma tarefa complexa, que envolve vários domínios administrativos e tecnologias distintas. Além disso, serviços como autenticação, autorização, serviços de nomes, comunicação, entre outros, são totalmente afetados pela escalabilidade.

Segundo (Neuman, 1994) um sistema é dito escalável se tem a habilidade de tratar a adição de novos usuários e recursos sem sofrer uma perda notável de desempenho ou um aumento na complexidade de administração. Na literatura, encontra-se diferentes abordagens para a implementação dos mecanismos de autenticação e autorização: abordagem centralizada, autenticação centralizada e autorização descentralizada e abordagem descentralizada.

2.5.1 Abordagem Centralizada

Na abordagem centralizada, a autorização e a autenticação são implementadas de forma centralizada por uma única máquina no sistema distribuído. Costuma-se dizer que há uma única Base Computacional Confiável[7] (TCB - *Trusted Computing Base*) (Department of Defense, 1985), responsável pela autenticação e autorização em todo o sistema.

Embora apresente a vantagem de permitir uma política de autorização coerente e fácil de se implantar, tal abordagem traz a desvantagem de acarretar a perda de desempenho,

[7] O conjunto de mecanismos de segurança (softwares, hardwares, mecanismos criptográficos, etc, que implementam as políticas de segurança de um sistema (Lampson *et al.*, 1992).

haja vista uma única máquina ser a responsável por autorizar e autenticar todos os pedidos originados no sistema. Além disso, a concentração de funções em uma única máquina cria um ponto de vulnerabilidade capaz de comprometer toda a segurança do sistema, tornando essa abordagem pouco atrativa para sistemas distribuídos.

2.5.2 Autenticação Centralizada e Autorização Descentralizada

Nesta abordagem, largamente utilizada atualmente, a autenticação é centralizada, devido a sua ligação com o espaço de nomes. Já a autorização é controlada independentemente por cada sítio, ou domínio administrativo do sistema. No entanto, uma única máquina autenticando todos os seus membros ainda gera um ponto único de vulnerabilidades. Além disso, há a necessidade das máquinas envolvidas na autorização confiarem no servidor de autenticação. A própria autorização descentralizada também apresenta dificuldades na manutenção da sua coerência, já que neste caso a matriz de acesso apresenta-se dividida entre os domínios (de Mello, 2003).

A escalabilidade nesta abordagem é conseguida usando o conceito de domínio. O domínio é caracterizado por um servidor de autenticação impondo suas políticas de autenticação e autorização a seus membros.

O modelo X.509 (Adams and Farrell, 1999) e o projeto Athena desenvolvido no MIT (Neuman and Ts'o, 1994), que faz uso do Kerberos como serviço de autenticação, são baseados nesta abordagem.

2.5.3 Abordagem Descentralizada

Nesta abordagem cada máquina do domínio é responsável pela autenticação e autorização dos principais. A forma mais comum de se implementar esta abordagem é através das redes de confiança. Estas consistem em várias entidades confiáveis distribuídas no sistema, que se encarregam de implementar as políticas de autorização e autenticação. Embora essa abordagem retire o inconveniente da centralização de algumas atividades, o problema de gestão descentralizada continua comprometendo a coerência da política de autorização. O SPKI/-SDSI (*Simple Public Key Infrastructure/Simple Distributed Security Infrastructure*) (Ellison, 1998) e o TCSEC (*Trusted Computer System Evaluation Criteria*) (Department of Defense, 1985) são exemplos desta abordagem.

2.6 Estudo de casos

2.6.1 Kerberos

Kerberos[8] é um serviço de autenticação desenvolvido como parte do Projeto Athenas do MIT. Este trabalha em um ambiente distribuído aberto nos quais principais desejam acessar serviços em um servidor distribuído. Atualmente na sua versão 5 (Neuman and Ts'o, 1994), o Kerberos é um padrão IETF(*Internet Engineering Task Force*) (Kohl and Neuman, 1993).

O Kerberos possui **autenticação centralizada** e **autorização descentralizada**. Os principais se autenticam no *Authentication Service* (AS) do servidor Kerberos, conhecido como *Key Distribution Centre* (KDC). Uma vez autenticados, os principais recebem um *Ticket Granting Ticket* (TGT), que é uma credencial com um período de validade, para solicitar ao *Ticket Granting Services* (TGS) uma credencial para se comunicarem com os serviços ao qual se deseja fazer uso (Figura 2.4). Desta forma o serviço fica responsável por verificar se o principal pode ou não fazer uso do serviço.

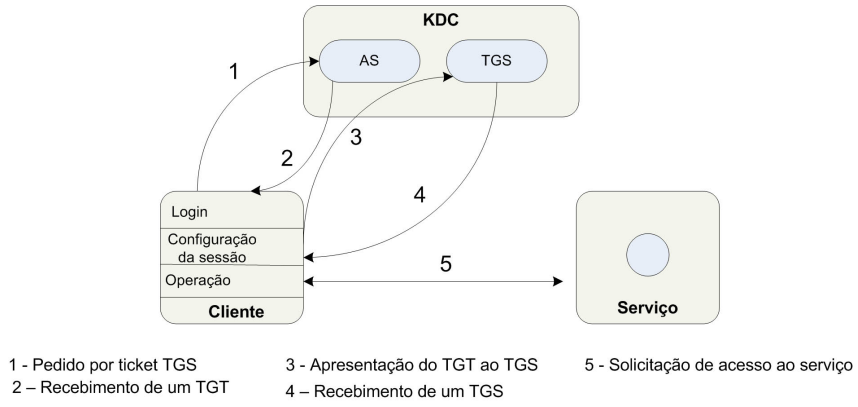

Figura 2.4: Modelo Kerberos

No passo 1 (Figura 2.4) o cliente envia seu nome e o nome do servidor ao qual deseja acesso e solicita ao servidor de autenticação (AS) o *ticket* para acessar o servidor de *tickets* (TGS). No passo 2 o cliente recebe um TGT do AS para se comunicar com o TGS. Em seguida, no passo 3, o cliente apresenta o pedido ao TGS (este, cifrado pela chave de sessão gerada pelo AS), que cria uma nova chave de sessão a qual será utilizada na comunicação entre o cliente e o servidor, passo 4. Por fim, no passo 5, o cliente envia ao servidor o *ticket* concedido pelo TGS e obtém acesso ao serviço.

Além da autenticação, o servidor Kerberos também oferece um serviço de comunicação com as propriedades de integridade e confidencialidade. A escalabilidade é atingida pela utilização de domínios separados de autenticação, chamados *realms*. Neste caso, cada domínio possui um servidor Kerberos e para que um cliente de um domínio tenha acesso a outro

[8]http://web.mit.edu/kerberos/www/

domínio é necessário que os servidores Kerberos possuam uma relação de confiança entre si, ou seja, compartilhem a chave secreta.

Além das desvantagens apontadas na Seção 2.5.2 em relação a esta abordagem, há também a dependência do compartilhamento de chaves secretas de cada cliente pertencente ao domínio com o servidor Kerberos.

2.6.2 X.509

O padrão X.509 foi publicado, originalmente, em 1988 como parte das recomendações sobre o diretório X.500[9]. Desde então, foi revisado duas vezes - em 1993 e novamente em 1995. Da revisão de 1993, que corrigiu vários problemas tanto com os protocolos e a estrutura do certificado, surgiu a versão referenciada como X.509v3 da *International Telecommunication Union* (ITU), padrão ISO (*International Orgazination for Standardization*) (BURNETT, 2002). As versões do certificado X.509 contêm os campos ilustrados na Figura 2.5.

Versão (*Version*)	Identifica as diversas versões de certificados	V.1
Número Serial do Certificado (*Certificate Serial Number*)	Valor inteiro único para cada certificado, gerado pela CA	V.1
Identificador do Algoritmo de Assinatura (*Signature Algorithm Identifier*)	Algoritmo utilizado na assinatura do certificado	V.1
Nome do Emissor (*Issuer Name*)	Nome distinto (distinguished name - DN) com qual a CA cria e assina o certificado	V.1
Validade - Não antes/Não depois (*Validity – Not before/Not after*)	Período de validade do certificado	V.1
Nome do Sujeito (*Subject Name*)	DN da entidade final a que o certificado se refere	V.1
Informação Chave Pública do Sujeito (*Subject Public Key Information*)	Valor da chave pública do sujeito, bem como algoritmo e parâmetro associados ao algoritmo	V.1
Identificador Único do Emissor (*Issuer Unique Identifier*)	Identificador único que pertence ao emissor	V.2
Identificador Único do Sujeito (*Subject Unique Identifier*)	Identificador único que pertence ao sujeito	V.2
Extensões (*Extensions*)	Extensões abragem as informações sobre a política, a chave, os atributos de sujeito e de emissor e as restrições do caminho de certificação	V.3
Assinatura (*Signature*)	Assinatura da CA	Todas as versões

Figura 2.5: Estrutura do certificado X.509 (BURNETT, 2002)

A autenticação, assim como no Kerberos, também é realizada de forma **centralizada**, porém o uso de nomes X.500 aumenta a aplicação do modelo em redes de larga escala, como a Internet. Já a **autorização é descentralizada**, cabendo aos servidores de aplicação efetuarem o controle de acesso (de Mello, 2003).

O X.509 baseia-se na infra-estrutura convencional de chave pública ou ICP, e define uma estrutura que provê serviços de autenticação através do diretório X.500. Cada cliente possui um par de chaves, no qual a chave privada é mantida em seu poder e a pública é armazenada em diretórios X.500 na forma de certificados de nomes. Estes são emitidos por

[9]É responsável por armazenar conjuntos de ligações entre nomes e atributos. São estruturados hierarquicamente para refletir aspectos geográficos e organizacionais (Coulouris *et al.*, 2001)

uma autoridade certificadora do domínio, também conhecida como Autoridade Certificadora - AC(*Certificate Authority* - CA).

O diretório X.500 fornece a convenção de nomes tanto para emissor quanto para o sujeito. Nomes distintos (DN) foram originalmente criados para identificar entidades dentro de uma árvore de diretório X.500. Um nome distinto relativo (*relative distinguished name* - RDN) é o caminho de nós para um nó subordinado. O DN inteiro atravessa um caminho de raiz da árvore a um nó final que representa uma entidade em particular. Um objetivo do diretório é fornecer uma infra-estrutura para nomear exclusivamente todas as entidades de comunicações em todos os lugares (por isso o termo "distinto" em "nome distinto"). Como resultado, nomes em certificados X.509 são mais complexos. Certificados X.509v3 concedem maior flexibilidade para os nomes, sem restringir unicamente as formas dos nomes de X.500. As entidades podem ser identificadas por um ou mais nomes como, por exemplo, endereço de *e-mail* e de Internet (BURNETT, 2002).

O X.509 é um modelo hierárquico. As comunidades X.509 são construídas na forma *top-down*, com base na confiança das chaves públicas das ACs. Uma AC controla sua chave privada que é usada para assinar os certificados que esta emite. O modelo se baseia, principalmente, em certificados formando cadeias de autenticação, partindo da chave privada de uma AC confiável até a chave pública do usuário (modelo adotado pelo SSL/TLS (Dierks and Allen, 1997)). O modelo também habilita a certificação cruzada entre duas ACs, que permite o estabelecimento de relações de confiança. Desta forma, uma AC pode ir até outra sem a necessidade de percorrer toda a estrutura hierárquica (Burr *et al.*, 1998). No entanto, esse tipo de alternativa de fluxo só é permitido entre autoridades certificadoras, o que não descaracteriza a AC como entidade centralizadora da certificação nem a rigidez e complexidade da estrutura hierárquica imposta por esta infra-estrutura.

Os certificados são criados com a crença de que serão válidos e usáveis por todo o tempo de vida esperado e indicado no campo "Validade" (Figura 2.5). Entretanto, em alguns casos, um certificado ainda em vigor não deverá ser mais utilizado. Por exemplo, quando há o comprometimento da chave privada correspondente a qual o certificado faz referência. Desta forma a AC utiliza-se de uma lista de revogação de certificados (*certificate revocation list* - CRL) para revogar o certificado. Em sua forma básica, a CRL é uma estrutura de dados assinada contendo uma lista de data/hora dos certificados revogados. O assinante de uma CRL é, em geral, a mesma entidade que originalmente a emitiu (a AC). Após criada e assinada, a CRL é distribuída livremente através de uma rede ou armazenada em um diretório da mesma maneira como os certificados são tratados. Porém, o uso das CRLs não resolve por completo o problema a qual foram designadas, pois o considerável e variável atraso entre a AC que deseja notificar a revogação de algum certificado e seus clientes/servidores invalidariam o uso das mesmas. O SSL/TLS faz grande uso de certificados X.509, porém não verifica as listas de revogação de certificados. Segundo (Gerck, 2000), as CRLs estão próximas do desuso completo.

O X.509 segue o padrão de formato de dados do ASN.1 (*Abstract Syntax Notation One*). O ASN.1 é um padrão poderoso, porém muito complexo, acarretando mais um dificuldade na implementação e uso do X.509, segundo (Clarke *et al.*, 2001).

2.6.3 SPKI/SDSI

O *Simple Distributed Security Infrastructure* (SDSI) foi projetado em 1996 por Ron Rivest e Butler Lampson no MIT (*Massachusetts Institute of Technology*). Seu principal objetivo, segundo (Clarke *et al.*, 2001), era "facilitar a construção sistemas distribuído seguros e escaláveis". Na mesma época, o *Simple Public Key Infrastructure* (SPKI), (Ellison, 1998), foi proposto por Carl Ellison para um modelo simples de autorização para chaves públicas. As duas proposta se uniram em 1998 para formar o SPKI/SDSI, seguindo assim a **abordagem descentralizada**.

Uma das principais razões para o surgimento do SPKI/SDSI foi fornecer um modelo flexível e de fácil implementação. Além do mais, o modelo trata principais como chaves públicas, tornandos-os verdadeiramente únicos (em contraste com o X.509) e, definindo assim, um certificado de nomes. Este modelo ainda combina uma chave pública com um conjunto de atributos para formar um certificado de autorização.

As principais características do SPKI são descritas a seguir, conforme (Nazareth, 2003; de Mello, 2003):

- **Orientado a chaves:** principais são chaves públicas. Todas as autorizações e verificações são realizadas em chaves públicas e atributos, não requerendo nomes.
- **Escalável**: não há uma hierarquia global ou infra-estrutura global. Cada principal é livre para emitir certificados a outros indivíduos, conduzindo assim a escalabilidade.
- **Tipos de certificados**: assim como o X.509 apresenta **certificados de nomes** e **autorização**. Os certificados de nomes, diferentemente do padrão X.509, ligam um nome local a uma chave pública. O nome é válido somente no espaço de nomes local e é usado pelos principais somente para identificação pessoal. Esse certificado apresenta 4 campos[10], sendo estes: (K, N, P, VS). K é a chave do emissor (*issuer*), N um identificador (*identifier*), P é o sujeito do certificado (*subject*) e VS uma especificação de validade (*valid specification*). Um **certificado de autorização** concede uma autorização específica, dada pelo emissor do certificado para o sujeito do certificado, sobre algum recurso que esteja protegido. O emissor do certificado é o principal, sendo este responsável pela guarda do serviço. As permissões dadas por este ao sujeito podem ser delegadas integralmente ou parcialmente para outros principais através de outros certificados de autorização emitidos por este sujeito, se o emissor do certificado especificar

[10]Em (Ellison, 1999) é definido como opcional o campo *comment* cujo conteúdo poderia ser um texto explicativo sobre o próprio certificado, como por exemplo, o objetivo do certificado em questão.

que o mesmo possa ser delegado. O certificado de autorização consiste de cinco campos: (E,S,A,D,V), sendo respectivamente, chave do emissor, sujeito do certificado, bit de delegação, *tag* e a especificação de validade. O emissor (*issuer*) é a chave que assina o certificado, é este que concede a autorização; o sujeito (*subject*) é a chave ou grupo que irá receber a autorização. Um sujeito pode ser uma chave pública ou um nome consistindo de uma chave pública seguida por um ou mais identificadores; *tag* é a especificação da autorização que será concedida pelo emissor ao sujeito; bit de delegação trata-se de um campo lógico, indicando se o certificado pode ou não ser delegado; e a especificação de validade (*validity specification*) indica o período de validade do certificado. Após este período, o certificado estará expirado e não mais será válido.

- **Delegação**: uma das principais vantagens do SPKI/SDSI é a habilidade de delegar autorizações. Isto é feito usando um *bit* de propagação no certificado de autorização. Se o *bit* é marcado então o sujeito do certificado pode delegar a sua permissão ou atributos a outros.

- **Cadeia de autorização**: uma cadeia de autorização é uma cadeia de certificados válidos que especificam um autorização. Segundo (Clarke *et al.*, 2001), uma cadeia de autorização pode consistir de apenas um certificado de autorização ou tanto um certificado de nome e autorização. A "raiz de confiança" de uma cadeia de autorização é o primeiro principal na cadeia que iniciou o processo de delegação. Ao contrário do X.509 onde a raiz de confiança é a AC que emite o certificado, a autorização baseada em cadeias de autorização exige que sempre haja caminhos ou cadeia de certificados, entre cliente e servidores, que garantam ao cliente o direito sobre o recurso (provido pelo servidor) desejado. Com isto a dificuldade neste modelo está na identificação das cadeias de certificados que levem um cliente obter a autorização necessária para acessar os recursos de um servidor. Tal dificuldade se dá porque há casos em que um cliente não possui relação de confiança com o servidor e assim sendo são necessários mecanismos que possibilitem a criação de tal relacionamento. Porém, há trabalhos dedicados a determinação de cadeias de certificados, como em (Santin, 2004).

- ***S-expressions***: certificados SPKI/SDSI fazem uso de *S-expressions* que são uma representação ASCII entendível por humanos. Isso permite que qualquer um facilmente leia um certificado e deduza o seu significado, ao contrário do X.509, que usa o padrão ASN.1.

O SPKI/SDSI ainda provê meios de tolerância a faltas através dos *Threshold Subjects*. Utilizados somente em certificados de autorização, os *Threshold Subjects* determinam que K dentre N sujeitos devem assinar um pedido ou uma delegação, para que o mesmo seja válido. Os sujeitos podem ser uma chave pública, um nome ou ainda um grupo.

2.7 Comparação entre os controles de autenticação e autorização

Na Seção 2.6, apresenta-se as características e limitações das configurações de autenticação de autorização em sistemas distribuídos. A Tabela 2.2 apresenta um resumo dos principais aspectos dos estudos de casos realizados.

Tabela 2.2: Resumo dos principais aspectos de Autenticação e Autorização

Abordagem	**Autenticação**	**Autorização**	**Espaço de nomes**	**Escalabilidade**	**Interoperabilidade**	**Tolerância a falhas**
Kerberos	centralizada	descentralizada	local	sim	sim	-
X.509	centralizada	descentralizada	global	sim	sim	-
SPKI	descentralizada	descentralizada	local	sim	sim	sim

A abordagem de autenticação centralizada e de autorização descentralizada, presente no Kerberos e no X.509, apresenta o problema de centralização da autenticação e desta forma possui um ponto único de falhas, vulnerabilidades e conseqüente comprometimento no desempenho. Tanto o Kerberos quanto o X.509 tentam minimizar os efeitos da centralização por meio de algum mecanismo particular. No caso do X.509 a adoção de chaves públicas e o uso do diretório X.500 minimizam o problema da centralização, porém a representação de nomes globais imposta pelo X.500 ainda constitui dificuldades devido a sua complexidade. O Kerberos já é mais limitado pela adoção de chaves secretas. Em um ambiente de larga escala a administração de todas as chaves secretas compromete a escalabilidade.

A abordagem descentralizada, presente no SPKI/SDSI, mostra-se mais adequada para controle da autenticação e autorização em ambientes de larga escala. O SPKI/SDSI apresenta uma poderosa solução para implementação da autenticação e autorização, porém a necessidade de conhecimento prévio das cadeias de certificados de autorização, em certas circunstâncias, impõe um problema de difícil solução.

2.8 Conclusão

Este capítulo apresentou os conceitos fundamentais de segurança em sistemas computacionais. Viu-se a urgência em adotar sistemas seguros e a complexidade imposta na adoção da segurança. Atingir as propriedades de segurança é requisito para alcançar a segurança computacional. Permitir que pessoas corretas acessem aos recursos corretos no lugar correto é função da política de segurança, que faz uso de modelos de segurança para descrever formalmente a política e avaliar a sua coerência. Os mecanismos de segurança são os meios utilizados para implementar as políticas e garantir as propriedades de segurança. Controles criptográficos, autenticação e autorização são mecanismos essenciais envolvidos na segurança computacional.

A autenticação e a autorização em sistemas de larga escala são determinantes para identificar e autorizar os principais do sistema. Diferentes abordagens propõem soluções variadas para tanto. Nesse sentido, devido a heterogeneidade das infra-estruturas de segurança, nem sempre a compatibilidade entre diferentes padrões utilizados.

Capítulo 3

Serviços *Web*

3.1 Introdução

O capítulo anterior abordou a segurança computacional, com enfoque nos mecanismos de autenticação e de autorização para sistemas distribuídos, apresentando alguns estudos de casos com as vantagens e desvantagens de cada abordagem. Verificou-se a grande heterogeneidade presente nos sistemas distribuídos e as características das diferentes tecnologias de Infra-estrutura de Chaves Públicas (ICPs).

Neste capítulo, aborda-se uma descrição das principais características da tecnologia de Serviços *Web* (*Web Services*). Esta tecnologia integradora permite que sistemas distintos se comuniquem através da Internet. O uso de protocolos padronizados garante a independência de plataforma e de linguagem de programação. Esta tecnologia é baseada na Arquitetura Orientada a Serviço e fornece meios para publicação, invocação e localização dos serviços. Sendo assim, é possível disponibilizar uma coleção de serviços remotos e permitir que estes sejam acessados por programas clientes. No entanto, apesar das vantagens oferecidas pelos Serviços *Web*, há grandes preocupações em relação à segurança das aplicações.

Inicialmente, este capítulo apresenta os conceitos e as características da Arquitetura Orientada a Serviço e, em seguida, descreve a arquitetura de Serviços *Web*, com os padrões que formam a sua base. As especificações de segurança que objetivam tornar a tecnologia confiável, agregando diversos requisitos de segurança, são discutidos na seqüência. Por fim, descreve-se as preocupações em manter a interoperabilidade entre as diversas especificações de Serviços *Web*.

3.2 Arquitetura Orientada a Serviço

Segundo (W3C, 2004a), a Arquitetura Orientada a Serviço (AOS) (*Service-Oriented Architecture* - SOA) pode ser definida como 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 participantes: o provedor de serviços, o diretório para registro do serviço e o cliente (solicitante do serviço). O provedor de serviços é a entidade que cria o serviço, é o responsável por publicar as interfaces dos seus serviços no registro de serviços e atender as requisições realizadas pelos clientes. O diretório para registro dos serviços é o repositório que publica e localiza as interfaces do serviço. O cliente é uma aplicação, ou um outro serviço, que realiza requisições a um serviço. As interações entre esses participantes, ilustrada na Figura 3.1, envolvem as operações de publicar, localizar e invocar (W3C, 2004a).

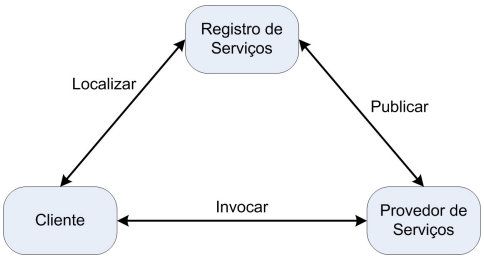

Figura 3.1: Interações entre os participantes da AOS (W3C, 2004a)

Na Figura 3.1, um provedor de serviços, primeiramente, constrói o seu serviço, o implementa em uma linguagem e em uma plataforma (Linux e C++, por exemplo) e disponibiliza a descrição deste serviço por meio de uma interface em um registro de serviços. A partir de então, qualquer cliente, após realizar uma busca no diretório de serviços e encontrar a descrição da interface do serviço, pode acessar essa interface e interagir com o serviço que implementa as funcionalidades descritas na sua interface. A interface fornece os mecanismos pelos quais os serviços se comunicam com os clientes, ou seja, esta descreve a assinatura de um conjunto de operações que são disponíveis para invocações dos clientes (Papazoglou, 2003).

3.3 Arquitetura dos Serviços *Web*

Serviços *Web* são consolidados sobre a AOS (Gutierrez *et al.*, 2004).O fraco acoplamento é a sua principal característica e significa que o principal não precisa conhecer qualquer estrutura ou rotina interna para fazer uso do serviço. O cliente apenas precisa ter acesso à interface do serviço, que é o meio pelo qual o provedor informa as características do seu serviço.

A interoperabilidade aparece também como uma grande vantagem fornecida pelos Serviços *Web* e se dá graças ao seu formato de dados na comunicação. A sua comunicação é baseada em formato texto, em um padrão chamado XML (*Extensible Marckup Language*)

(W3C, 1998). A linguagem XML oferece uma estrutura para criar diversos tipos de documentos. Além disso, pode ser facilmente lida pelos mais diversos dispositivos, sistemas operacionais e linguagens de programação.

Serviços *Web* são ideais para projetos B2B (*business-to-business*) e para outras aplicações de sistemas distribuídos que, através da Internet, requerem tecnologias integradoras. Além disso, cita-se como vantagens desta tecnologia à capacidade de atravessar os filtros de pacotes tradicionais *firewalls*, uma vez que, geralmente, utiliza o protocolo HTTP

Um grande número de definições pode ser encontrado para descrever Serviços *Web*. Segundo (W3C, 2004a):

> *Um Serviço Web é um sistema de software projetado para ser interoperável nas interações entre aplicações sobre uma rede. Sua interface é descrita em um formato processável pela aplicação (WSDL). Sistemas clientes interagem com os Serviços Web conforme descrito em sua interface, ou seja, a WSDL, e através de mensagens SOAP; tipicamente transportadas usando o protocolo HTTP sobre uma serialização XML e em conjunto com os outros padrões relacionados da Internet.*

Para tornar possível as três operações fundamentais de uma AOS - publicar, localizar e invocar - a arquitetura de Serviços *Web* apresenta as seguintes tecnologias, baseadas em XML, que constituem a sua base: a WSDL (*Web Services Description Language*), linguagem que descreve as funcionalidades dos Serviços *Web*; o UDDI (*Universal Description, Discovery and Integration*), serviço pelo qual a descrição do serviço é localizado; e SOAP (*Simple Object Access Protocol*), protocolo usado para a invocação do serviço. A Figura 3.2 ilustra a arquitetura dos Serviços *Web*. Inicialmente (passo 1), um provedor de serviços implementa o seu serviço, disponibiliza a sua interface (a WSDL) em um serviço UDDI (passo 2). A partir de então, o cliente obtém a WSDL, implementa a aplicação cliente, segundo a WSDL, e está hábil a fazer invocações ao serviço (passo 3).

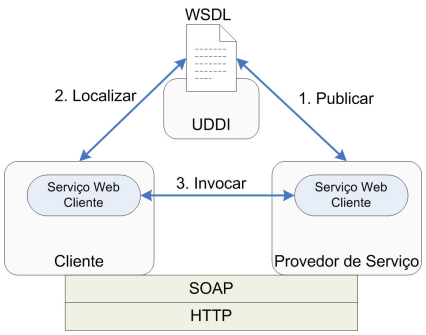

Figura 3.2: Arquitetura dos Serviços *Web*

Vogels (Vogels, 2004) apresenta uma série de conceitos equivocados que relacionam os Serviços *Web* à tecnologia de objetos distribuídos, como o padrão CORBA. Segundo ele, há

similaridades entre ambos, já que possuem uma linguagem de descrição para o serviço (IDLs em CORBA e WSDL para Serviços *Web*) utilizam de um mecanismo para registro e descoberta de componentes disponíveis. Porém, ele aponta várias limitações dos Serviços *Web* em relação a CORBA, como a dificuldade em assegurar a confiança fim a fim, precisarem rodar sobre um servidor *Web*, as chamadas dos serviços serem somente efetivas quando se usa a Internet, etc. Desta forma, apesar das várias vantagens apresentadas pela tecnologia de Serviços *Web*, algumas desvantagens ainda criam barreiras para a sua completa adoção, entre estas, pode-se citar (W3C, 2004a; Vogels, 2004):

- **Desempenho**. Em geral, os Serviços *Web* não apresentam um bom desempenho quando a quantidade de dados trocada por duas partes é muito grande. Isso se deve, basicamente, pelo fato das mensagens trocadas serem no formato XML e não em um formato binário.
- **Transações não Confiáveis**. Algumas negociações em Serviços *Web* são simples e em geral não requerem transações entre várias empresas. No entanto, em casos em que uma empresa vende um produto e precisa de outra empresa para entregá-lo e ainda uma terceira empresa para aprovar a transação financeira, um mecanismo padrão de transação confiável em Serviços *Web* se torna indispensável (OAIS, 2005).
- **Interoperabilidade**. Ainda que os Serviços *Web* sejam especificados e desenvolvidos para serem interoperáveis, algumas questões ainda em aberto e não definidas pelas especificações podem gerar dificuldades na integração de sistemas. Por exemplo, algumas especificações apontam como opcionais características que são obrigatórias para outras. A organização WS-I (, WS-I) é uma organização com destaque nos esforços para manter a interoperabilidade, reunindo as especificações, apontando possíveis problemas e sugerindo soluções (ver Seção 3.5).
- **Segurança**. Como apontado por (W3C, 2004a; P Kearney and He, 2004), a segurança fim a fim é um problema para Serviços *Web*. O uso de tecnologias como o protocolo SSL (*Secure Socket Layer*), não garante segurança fim a fim nas trocas de mensagens quando há a presença de nós intermediários.

As questões acima não são tratadas pelas especificações que formam a base da tecnologia de Serviços *Web* - UDDI, SOAP, WSDL. Por outro lado, diversas outras especificações surgem e estão sendo aprimoradas constantemente visando solucionar as dificuldades expostas acima. Serviços *Web* são atacados em várias frentes, dentre as quais, destaca-se (Figura 3.3):

- **as especificações base** que fornecem as tecnologias que compõe os Serviços *Web* e que serão abordadas nas subseções seguintes;

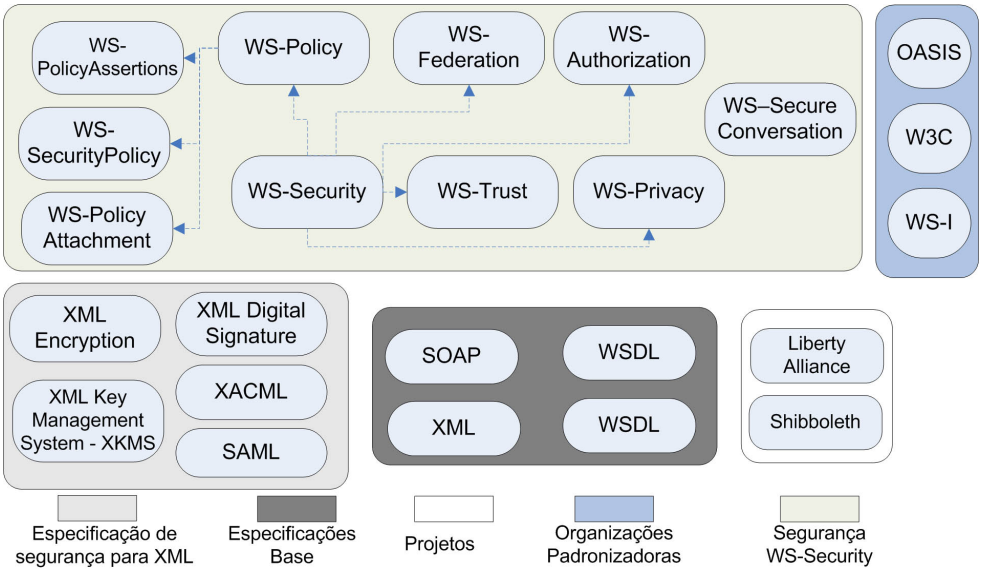

Figura 3.3: Visão geral das especificações, organizações e projetos sobre Serviços *Web*

- **as organizações padronizadoras**, nas quais se destacam a OASIS[1], a W3C[2] e a WSI[3];
- **os projetos** que se destacam na adoção da tecnologia e que inclusive colaboram no desenvolvimento das especificação, por exemplo, o projeto Liberty Alliance [4] e Shibboleth[5];
- os padrões de **segurança relacionados a XML** e já consolidados que são utilizados pelas especificações de segurança de Serviços *Web* para agregar segurança na troca de mensagens;
- a família de **especificações de segurança para Serviços Web** desenvolvida pela Microsoft e IBM (Corporation and Corporation, 2002) e padronizada pela OASIS (OASIS, 2004b) que serão abordadas na Seção 3.4, juntamente com outras questões envolvidas na segurança.

3.3.1 XML

A tecnologia XML (*eXtended Markup Language*) (W3C, 1998) oferece requisitos chaves para muitas aplicações. Esta oferece um formato de dados flexível e naturalmente extensível. O padrão XML foi definido como um formato de texto adequado para publicações eletrônicas, mas evoluiu para ser uma representação universal de documentos estruturados e de dados.

[1] http://www.oasis-open.org/
[2] http://www.w3.org/
[3] http://ws-i.org/
[4] http://www.projectliberty.org/
[5] http://shibboleth.internet2.edu/

Em se tratando de Serviços *Web*, os aspetos importantes de XML são sua sintaxe, o XML *Infoset*, o XML *Schema* (Fallside and Walmsley, 2004) e o espaço de nomes (XML *Namespaces*). A sintaxe do XML permite a definição de um número infinito de nomeadores, chamados *tags*. Desta forma, é possível criar *tags* para descrever dados estruturados. Um elemento XML pode ter dados declarados, por exemplo, como sendo preços de venda, taxas de preço, um título de livro, ou qualquer outro tipo de elemento de dado.

O XML *Infoset* (Cowan and Tobin, 2001) é um conjunto coerente de definições para componentes de um documento XML bem formado, incluindo as regras impostas pela extensão de espaço de nomes, e que são referenciadas por outras especificações relacionadas com XML. Como os padrões de Serviços *Web* estão estabelecidos sobre XML, as especificações fazem uso de XML *Infoset* para referenciar os documentos XML.

Em XML, um esquema (*schema*) define formalmente a estrutura que os documentos devem ter, ou seja, a ordem e aninhamento das *tags*. Desta forma, os documentos XML podem ser validados automaticamente, a partir de seu esquema correspondente. Além disso, a definição de esquemas em comum permite o intercâmbio de dados entre aplicações heterogêneas. Um esquema pode ser do tipo: DTD (*Document Type Definition*) ou XML *Schema* (XML Schema, 2001). O DTD apresenta uma série de limitações que são supridas pelo padrão *XML Schemas*. Em *XML Schema*, o modelo de conteúdo de um elemento pode ser especificado a partir da declaração de um tipo. Um tipo pode ser **simples** ou **complexo**. Um tipo simples pode ser um atributo ou um elemento simples, que possui somente texto e não possui elementos filhos. Já um tipo complexo é utilizado para dizer quais são os subelementos permitidos para um determinado elemento. Tipos simples são declarados com o `simpleType` e tipos complexos com o `complexType`. A declaração `element` liga um tipo a um nome de elemento. Elementos podem ser declarados dentro de um tipo complexo ou em um *schema*. Neste último caso, são considerados elementos globais.

Um espaço de nomes (*namespace*) serve para definir os nomes dos esquemas, ou seja, qualificam os nomes dos elementos e atributos para que não ocorram conflitos de nomes repetidos. Consiste de um prefixo e de um URL, que serve como um identificador único para o espaço de nomes. Desta forma, este mecanismo permite que um documento XML utilize elementos definidos por diferentes esquemas XML.

A Figura 3.4 ilustra a definição de um esquema. A linha 2 referencia o elemento raiz e o espaço de nomes padrão para construção de um esquema. Neste espaço de nomes está definido todo o vocabulário necessário para criação do esquema, como os elementos, tipo simples, complexos, etc. A linha 3 define o elemento `registro` que é um tipo complexo "pessoa". Um tipo complexo é como uma classe em orientação a objetos. A linha 4 define o tipo complexo "pessoa", que é formado por uma sequencia dos elementos `nome`, do tipo "string"; `idade`, do tipo "inteiro"; e `data`, do tipo "date".

A Figura 3.5 ilustra uma estrutura XML que obedece ao esquema definido na Figura 3.4.

```
1  <?xml version="1.0"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:element name="registro" type="pessoa"/>
4    <xs:complexType name="pessoa">
5        <xs:sequence>
6          <xs:element name="nome" type="xs:string"/>
7          <xs:element name="idade" type="xs:integer"/>
8          <xs:element name="data" type="xs:date"/>
9        </xs:sequence>
10     </xs:complexType>
11 </xs:schema>
```

Figura 3.4: Exemplo de um *XML-Schema*

```
1  <?xml version="1.0" encoding="ISO-8859"?>
2  <registro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="exemplo.xsd">
3    <nome>Edson</nome>
4    <idade>23</idade>
5    <data>1981-06-01</data>
6  </registro>
```

Figura 3.5: Exemplo de um XML

A linha 2, contêm os elementos que fazem referência ao *XML-Schema* definido anteriormente.

3.3.2 Linguagem para Descrição de Interfaces - WSDL

A WSDL (W3C, 2004b), definida pela W3C, é uma gramática XML, extensível, para descrever as interfaces dos Serviços *Web*. Através da WSDL é possível separar a descrição das funcionalidades oferecidas por um serviço dos detalhes concretos da sua implementação.

A WSDL começa descrevendo a troca de mensagens entre os agentes solicitantes e fornecedores de serviços. As mensagens são descritas de forma abstrata e então adequadas para um protocolo de rede e formato de mensagens. Desde que os agentes concordem com a WSDL, a implementação do Serviço *Web* pode ser concretizada em qualquer linguagem de programação ou plataforma, como dito anteriormente.

A linguagem para descrição de interfaces define o formato da mensagem, o tipo de dados, os protocolos de transporte e o formato de serialização de transporte que são usados entre o cliente e o serviço. Esta linguagem também especifica um ou mais locais de rede pelo qual o serviço pode ser invocado e pode indicar alguma informação sobre o padrão de troca de mensagens que é esperado. A idéia é similar a uma interface Java ou a uma IDL (*Interface Definition Language*) do CORBA. Em resumo, a descrição do serviço representa um acordo que rege as interações do cliente com o serviço.

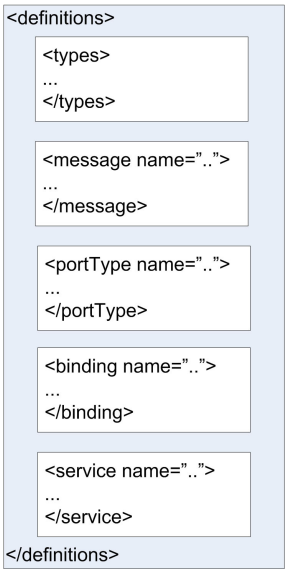

Figura 3.6: Estrutura sintática da WSDL (Weerawarana *et al.*, 2005)

Basicamente, quando o cliente deseja enviar uma mensagem para um determinado Serviço *Web*, este obtém a descrição do serviço (através da localização do respectivo documento WSDL), constrói a mensagem passando os tipo de dados corretos (parâmetros, etc), de acordo com a definição encontrada no documento que descreve a operação a ser invocada. Em seguida, a mensagem é enviada para o endereço onde o serviço está localizado, a fim de que possa ser processada. O serviço, quando recebe esta mensagem, a valida conforme as informações contidas no documento WSDL. O serviço remoto sabe como tratar a mensagem, como processá-la e como montar a resposta ao cliente.

A Figura 3.6 exibe a estrutura sintática da WSDL. Segundo (Weerawarana *et al.*, 2005), um documento WSDL possui um elemento raiz chamado `definitions`, que por sua vez contém uma parte dita abstrata e outra parte dita concreta. A parte abstrata descreve o que o Serviço *Web* faz em termos da mensagem que este consome e produz, sem considerar como e onde o serviço é oferecido. Tal descrição é responsabilidade dos elementos `<types>`, `<message>` e `<portTypes>`. Já a parte concreta, define o "como" e "onde". Isto se dá através dos elementos `<binding>` e `<service>`.

A partir dos elementos exibidos na Figura 3.6, qualquer interface pode ser representada pela WSDL. A Tabela 3.1 descreve os principais elementos que compõem a WSDL (Weerawarana *et al.*, 2005).

Para descrever melhor a função de cada elemento da Tabela 3.1, a Figura 3.7 contém a interface de um serviço, em linguagem Java, que será adequada segundo a linguagem WSDL (Weerawarana *et al.*, 2005). A interface possui o nome de `StockQuoteService` e possui um único método chamado `getQuote`. O método recebe como entrada um tipo `String` e retorna um tipo `float`. Um exemplo de aplicação de tal interface é conhecer o valor (preço) de determinado item em um estoque. O usuário entra com o nome do item e recebe o seu

Tabela 3.1: Elementos presentes na WSDL

Elemento	**Descrição**
`types`	A WSDL não define uma linguagem para descrever estrutura de dados, ao invés disso, utiliza-se de outras linguagens, como o *XMLSchema*. Desta forma, este elemento fornece informações sobre qualquer tipo de dado complexo usados na WSDL.
`message`	Permite descrever as mensagens que o Serviço *Web* está trocando. Na WSDL, mensagens podem consistir de uma ou mais partes, onde uma parte representa um único item que está sendo enviado ou recebido.
`portType`	Define o conjunto abstrato de operações mapeadas para um ou mais serviços, os quais são descritos como pontos finais de redes ou portas.
`binding`	Descreve como a operação é invocada pelo protocolo utilizado e o formato de dados para as *operações* e *mensagens*.
`service`	Declara o endereço das portas para os *bindings*. Ou seja, indica onde encontrar um serviço usando sua porta (`port`).

valor.

```
1 public interface StockQuoteService {
2     public float getQuote(String symbol);
3 }
```

Figura 3.7: Interface de um serviço em linguagem Java

A partir da interface da Figura 3.7 os tipos de dados são mapeados para o *XMLSchema* correspondente, ou seja, o elemento `types`, Figura 3.8. As linhas 3 e 6 contém o elemento `symbol`, do tipo `String`, e `getQuoteReturn`, do tipo `float`, respectivamente, definidos na interface do serviços, conforme Figura 3.7.

```
1 <wsdl:types>
2   <schema elementFormDefault="qualified" targetNamespace="...">
3     <element name="symbol" type="xsd:string"/>
4   </schema>
5   <schema elementFormDefault="qualified" targetNamespace="...">
6     <element name="getQuoteReturn" type="xsd:float"/>
7   </schema>
8 </wsdl:types>
```

Figura 3.8: XMLSchema descrito no elemento `types`

Após a definição do elemento `types`, o elemento `message`, correspondente a interface da Figura 3.7, é descrito na Figura 3.9. A função de tal elemento é descrever as mensagens trocadas por cliente e serviço. As linhas 1 e 4 indicam como se dá a resposta a um pedido do cliente e o como o cliente efetua o pedido, respectivamente.

O elemento *portType* define as operações que o serviço suporta. Uma operação é simplesmente um conjunto de mensagens que são trocadas entre clientes e serviço. Cada operação pode enviar ou receber no máximo uma mensagem em cada direção. Os tipos de operação

```
1 <wsdl:message name="getQuoteResponse">
2     <wsdl:part element="impl:getQuoteReturn" name="getQuoteReturn"/>
3 </wsdl:message>
4 <wsdl:message name="getQuoteRequest">
5    <wsdl:part element="tns1:symbol" name="symbol"/>
6 </wsdl:message>
```

Figura 3.9: Elemento `message`

suportados são:

- **one-way**: uma mensagem chega ao serviço e este não retorna nenhuma resposta;
- **request-response**: a mensagem chega ao serviço e este retorna uma mensagem em resposta;
- **solicit-response**: o serviço envia uma mensagem e obtém uma mensagem de retorno;
- **notification**: o serviço envia uma mensagem e não recebe retorno do cliente.

A Figura 3.10 exibe o **portType** correspondente a interface descrita acima. A operação é do tipo *Request-response* (linhas 3 e 4).

```
1 <wsdl:portType name="StockQuoteService">
2    <wsdl:operation name="getQuote" parameterOrder="symbol">
3       <wsdl:input message="impl:getQuoteRequest" name="getQuoteRequest"/>
4       <wsdl:output message="impl:getQuoteResponse" name="getQuoteResponse"/>
5    </wsdl:operation>
6 </wsdl:portType>>
```

Figura 3.10: Elemento `portType`

A função do elemento `binding` é descrever como as mensagens interagem com um serviço em particular. A WSDL não assume um protocolo padrão e permite que as mensagens sejam trocadas usando SOAP, HTTP ou MIME. A Figura 3.11 descreve a interface acima em termos do seu elemento `binding`. Define que o formato das mensagens obedece ao tipo `Soap:binding` sobre HTTP, linha 2.

Por fim, o elemento `service` descreve onde encontrar o serviço (ver Figura 3.12, linha 3).

3.3.3 O Protocolo SOAP

Definido pelo consórcio W3C, o SOAP (*Simple Object Access Protocol*) (SOAP, 2003) é um protocolo de comunicação baseado em XML para a troca de mensagens entre clientes e provedores de serviços no ambiente de Serviços *Web*. O SOAP é independente de linguagem,

```
<wsdl:binding name="stock-wss-01SoapBinding" type="impl:StockQuoteService">
   <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="getQuote">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getQuoteRequest">
         <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getQuoteResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
   </wsdl:operation>
</wsdl:binding>
```

Figura 3.11: Elemento `binding`

```
<wsdl:service name="StockQuoteServiceService">
   <wsdl:port binding="impl:stock-wss-01SoapBinding" name="stock-wss-01">
      <wsdlsoap:address location="http://localhost:8080/axis/services/stock-wss-01"/>
   </wsdl:port>
</wsdl:service>
```

Figura 3.12: Elemento `service`

trabalha com diversos sistemas operacionais e sobre protocolos de aplicação já consolidados, como o HTTP, o SMTP, o FTP, o RMI/IIOP, etc.

A estrutura do SOAP é composta por um elemento XML denominado **envelope** contendo dois elementos filhos, sendo que em um deles está o conteúdo do cabeçalho (*header*) e no outro o corpo da mensagem (*body*)(ver Figura 3.13). O elemento **envelope** está presente em toda a mensagem SOAP, é o elemento raiz do documento XML e pode conter as declarações dos espaços de nomes XML e também os atributos adicionais, como o que define o estilo de codificação[6] (linha 1). O elemento **cabeçalho** (linha 2) é opcional e carrega informações adicionais, tais como: os atores que devem processar a mensagem - uma mensagem SOAP pode ser processada por vários nós intermediários antes de chegar ao destino final - e informações sobre a segurança, como certificados X.509 ou *tickets* Kerberos. Já o elemento presente das linhas 7 a 9 é o responsável por retornar possíveis erros durante o processamento da mensagem.

A versão 1.2 do SOAP não mais reflete o seu acrônimo "*Simple Object Access Protocol*", não se tratando de um protocolo de acesso a objetos, mas sim corresponde à outra interpretação: "*Service Oriented Access Protocol*" que se refere a um protocolo para acesso a serviços. Uma mensagem SOAP representa a informação necessária para invocar um serviço ou refletir os resultados de uma invocação de um serviço, contendo a informação especificada na definição da interface. O exemplo (Figura 3.14), a seguir, ilustra o fragmento de uma interface. O método exibido recebe dois argumentos do tipo inteiro como entrada e deve retornar a soma de ambos, também um tipo inteiro. A Figura 3.15 exibe uma mensagem SOAP

[6] O "*Encoding Style*" define como os elementos são representados no documento XML

```
<soap:Envelope xmlns:soap="..." soap:encodingStyle="..."
   <soap:Header>
     ...
   </soap:Header>
   <soap:Body>
    ...
     <soap:Fault>
      ...
     </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

Figura 3.13: Exemplo de mensagem SOAP

que faz uma invocação ao serviço especificado na interface da Figura 3.14.

```
...
public int Soma (int num1, int num2){...}
...
```

Figura 3.14: Exemplo de um fragmento da interface do um serviço - Método Soma

```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
    <c:calculadora xmlns:c="http://www.schemas.exemplo/calculadora/">
       <c:Soma>
           <c:arg1> 2 </c:arg1>
           <c:arg2> 2 </c:arg2>
       <c:Soma>
    </c:calculadora>
  </soap:Body>
</soap:Envelope>
```

Figura 3.15: Exemplo de uma mensagem SOAP requisitando soma de dois termos

Na Figura 3.15, o corpo da mensagem vai da linha 4 a 11. O corpo da mensagem contém os elementos necessários para acesso ao serviço. Na linha 5, está o espaço de nomes definido para a calculadora e na linha 6 a indicação que o método a ser acessado será o "Soma". Nas linhas 7 e 8, estão os dois argumentos que são as entradas do método. Neste exemplo, como resposta à invocação SOAP, uma outra mensagem SOAP, com o resultado 4, deverá ser retornada.

3.3.4 O Serviço para Publicação e Localização de Serviços - UDDI

Como visto na Seção 3.3.2, um provedor de serviços, através da WSDL, define todos os detalhes do serviço. A partir de então, o serviço está pronto para ser usado pelos potenciais clientes. No entanto, é necessário que antes os clientes "encontrem" o serviço e que os

provedores exponham a WSDL de seus serviços. Um provedor de serviços pode desejar, por motivos comerciais, que muitos clientes façam uso do seu serviço e assim precisa encontrar uma maneira de publicá-lo. Mas, porventura, o provedor também pode restringir a descrição do serviço a apenas clientes previamente conhecidos.

O UDDI (*Universal Description, Discovery, and Integration)*, padrão OASIS (OASIS, 2002b), define uma forma para publicar, validar e invocar informações sobre os Serviços *Web*. Este é tanto um padrão para a descrição do formato quanto um protocolo para o descobrimento de Serviços *Web*. Um diretório UDDI pode conter metadados[7] para qualquer tipo de serviço. Há três tipos de diretórios UDDI: públicos, privados e afiliados. O primeiro é utilizado por diversas empresas com diferentes focos comerciais. O acesso aos dados dos diretórios é público, mas com garantias de administração próprias de uma organização. Os diretórios privados servem a um nicho comercial reduzido, com acesso restrito aos dados e funções de administração do diretório. Geralmente, ficam isolados da rede pública, atrás de um *firewall* ou *proxy*. Já os diretórios afiliados, permitem apenas que clientes autorizados o acessem. Características de administração podem ser delegadas para partes confiáveis e os dados podem ser compartilhados com outros diretórios de maneira controlada.

3.4 Segurança na arquitetura dos Serviços *Web*

A segurança aparece como um requisito essencial para a adoção de Serviços *Web*. Embora a tecnologia apareça como uma possível propulsora dos negócios realizados pela Internet, as empresas não desejam correr riscos com a exposição de suas aplicações e fluxos de negócios, sem antes garantir que não sofrerão prejuízos.

A Tabela 3.2 exibe os principais riscos de segurança aos quais os Serviços *Web* estão expostos (Demchenko *et al.*, 2005; Technology, 2003).

Os principais problemas de segurança em Serviços *Web* advêm justamente de algumas das vantagens apontadas para sua adoção como, por exemplo, o uso de XML sobre a Internet, a transposição das mensagens XML pelos filtros de pacotes (*firewalls*), entre outras. A Internet é uma infra-estrutura de rede pública insegura e não confiável, estando assim sujeita a diversos tipos de ataques de segurança tanto por profissionais quanto por amadores. O protocolo SOAP, na sua origem, não apresenta mecanismos de segurança para a troca de mensagens entre as aplicações (embora preveja o uso de extensões para tal). Desta forma, as mensagens são enviadas em claro, ou seja, sem a presença de cifragem ou assinatura. Essa prática expõe as mensagens trocadas e, conseqüentemente, os principais, aos ataques contra a segurança apontados na Seção 2.2.2. Além disso, o fato das mensagens XML atuarem na camada de aplicação, através do HTTP, por exemplo, permite o livre tráfego da informação sobre a porta 80, somando mais uma preocupação aos administradores de rede.

[7]Dados que descrevem outros dados. Metadados sobre um documento XML são definidos no DTD ou no documento XML propriamente dito.

Embora os protocolos SSL/TLS (Dierks and Allen, 1997), garantam a confidencialidade na comunicação, autenticando e cifrando os dados trocados entre duas partes, estes não oferecem segurança fim a fim. Em Serviços *Web*, frequentemente a mensagem, para atingir o destinatário final, passa por diversos nós intermediários. Se a cifragem for utilizada somente na camada de transporte, nós intermediários terão revelado as informações que passam por eles. Além disso, segundo Geer (Geer, 2003) essa tecnologia apresenta limitações de escalabilidade na presença de um grande volume de transações.

Tabela 3.2: Riscos de Segurança para Serviços *Web*

Vulnerabilidade	**Descrição**
Firewalls não fornecem proteção adequada	*Firewalls* (filtros de pacotes) são projetados para atuarem na camada de rede e não na camada de aplicação.
Exposição das funcionalidades	a WSDL exibe os parâmetros e métodos utilizados para acessar os serviços específicos, fornecendo assim informações sobre como acessar determinado Serviço *Web*. Além disso, as exceções e manipulação de erros descritos na WSDL fornecem importantes informações sobre a estrutura interna do serviço.
Estrutura XML	A análise da estrutura XML requer consumo de recursos e tempo computacional. Muitas aplicações podem permitir entradas XML volumosas e complexas, permitindo assim ataques de negação de serviço.
Conteúdo XML	Documentos XML podem conter instruções maliciosas (extensões *XML Schema*, instruções *XQuery* ou *XPath*) que podem alterar o processo de análise do XML ou conter comandos maliciosos que carregam ameaças às aplicações finais. Além disso, XML tem a habilidade de incluir referências a documentos ou tipos de dados externos, permitindo desta forma que recursos externos sejam manipulados.
SOAP/XML	A infra-estrutura das mensagens SOAP trabalha em cima da camada de transporte, faz uso de serviços para entrega e roteamento e assim está sucestível a ataques à infra-estrutura da rede.
Credenciais de segurança XML	XML faz uso de credenciais de segurança para autenticação, autorização e gerenciamento de sessão. O uso inapropriado deste recurso pode permitir que as credenciais sejam utilizadas por terceiros.
Chaves seguras e sessões	Uma implementação pobre das especificações de segurança para XML ou dos Serviços *Web*, a geração de chaves de criptográficas fracas e o descuido do gerenciamento de confiança, provocam sérios riscos à segurança do sistema

Como pode ser visto, há muitos desafios para a adoção de Serviços *Web* seguros. Verifica-se a necessidade de se assegurar o trânsito das mensagens na rede com ou sem a presença de nós intermediários. Além disso, os dados armazenados pelas aplicações também precisam ser protegidos. Diversos requisitos de segurança são propostos em W3C (2004a); Gutierrez *et al.* (2004), os quais se exibe a seguir:

- **mecanismos de autenticação**: tornam-se necessários para verificar a identidade do solicitante (cliente), sendo que, em alguns casos, o uso de autenticação mútua entre as partes pode ser necessária;

- **autorização**: verifica o que o principal pode acessar, ou seja, realiza o controle de acesso sobre os recursos, segundo as políticas da aplicação;
- **integridade e confidencialidade**: são necessários mecanismos para assegurar que os dados e as mensagens em trânsito na rede não serão alterados e nem visualizados por terceiros;
- **segurança fim a fim**: requer que apenas o destinatário tenha acesso ao conteúdo da mensagem, mesmo quando a mensagem passa por diversos nós intermediários até chegar ao seu destinatário;
- **não repudiação**: as partes envolvidas não podem negar a participação na transação;
- **trilhas de auditoria**: são necessárias para traçar o acesso e o comportamento do usuário.

Com o intuito de atender os requisitos de segurança impostos, diversas propostas e especificações de segurança para os Serviços *Web* foram e estão sendo lançadas (como apresentado na Figura 3.3). A seguir, apresenta-se os padrões de segurança para XML e, posteriormente, os padrões específicos para Serviços *Web*.

3.4.1 Especificações de Segurança para o padrão XML

3.4.1.1 *XML-Signature*

A assinatura digital XML (*XML Digital Signatures - XMLDSig*) (XML, 2002) é uma especificação da W3C fruto do trabalho da IETF e da W3C. Projetada para ser usada em transações XML, esta define uma estrutura (*schema*) XML para aplicar assinatura digital em qualquer documento digital (uma imagem JPG, um documento HTML, etc), e não apenas em documentos XML. Desta forma, é possível aplicar a assinatura digital sobre um arquivo texto e o resultado é a assinatura digital, sobre o conteúdo do documento, representada em uma estrutura XML.

A especificação *XMLDsig* herda todos os benefícios da assinatura digital convencional, ou seja, garante autenticidade, integridade e não repudiação, que são características essências em uma transação entre duas entidades.

Uma característica fundamental da especificação *XMLDSig* é a habilidade de assinar somente porções específicas do conteúdo da árvore XML, ao invés do documento completo. Isto é relevante quando um documento XML tem um longo tempo de vida e diferentes partes alteram ou adicionam alguma informação em tempos distintos nesse mesmo documento. Cada parte então assina somente o que é relevante a esta. Se a assinatura, ao contrário,

é realizada sobre o todo documento, qualquer alteração aos valores da assinatura poderia invalidar a assinatura original.

Neste sentido, as assinaturas podem ser representadas em três formas distintas: *enveloped*, *enveloping* e *detached* (ver Figura 3.16). As assinaturas do tipo ***enveloped*** e ***enveloping*** acompanham o documento XML ao qual referenciam, sendo a primeira utilizada em Serviços *Web*, inserida na mensagem SOAP. Já na forma ***detached***, a assinatura está separada dos dados assinados. Neste caso, o documento é apenas referenciado na estrutura XML da assinatura. Em qualquer uma dessas formas de assinatura, é necessário que o objeto assinado esteja acessível para a validação da assinatura.

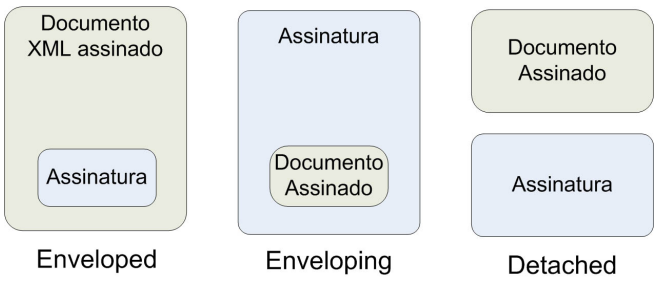

Figura 3.16: Forma de assinatura em *XML-Signature*

O cálculo da assinatura é sensível a pequenas modificações no documento XML, mesmo a uma simples mudança de um caracter ">" (simbolo que identifica o fechamento de um elemento XML). Diferentes aplicações XML, enquanto processam o conteúdo, podem tratar a representação física de forma diferente. Desta forma, é possível que ao verificar uma assinatura, esta venha a ser rejeitada mesmo se não sofrer alterações. Para resolver esse problema em Boyer (2001) foi introduzido um mecanismo de representar a XML na forma canônica. Este mecanismo tem por objetivo transformar um documento XML em um representação normalizada. Desta forma, durante a geração da assinatura, o resumo é calculado sobre a forma canônica do documento. Da mesma forma, ao receber o documento, a verificação também se dará sobre a forma canônica do documento, garantindo que a informação não foi alterada desde que foi assinada.

A estrutura que a especificação *XMLDSig* apresenta para armazenar as informações de assinatura é formada por vários elementos XML (Figura 3.17). O elemento raiz é chamado de `<Signature>` e o mesmo apresenta três elementos filhos, sendo estes: o `<SignedInfo>`, o `<SignatureValue>` e o `<KeyInfo>`. O elemento `<SignedInfo>`, linhas 2 a 10, representa os detalhes do processo que conduziu a assinatura. Na linha 3, está o elemento `<CanonicalizationMethod>`, que identifica o algoritmo usado para representar na forma canônica o documento assinado. O próximo elemento do `<SignedInfo>`, linha 4, identifica o algoritmo utilizado para produzir a assinatura criptográfica como, por exemplo, o RSA (RSA Laboratories, 2002). O elemento `<Reference>`, linhas 5 a 9, contem uma referência aos dados que foram assinados e várias outras informações sobre a assinatura, como o algoritmo utilizado para produzir o valor do resumo criptográfico (*digest*) e o corresponde valor

(linha 8).

```
1  <ds:Signature>
2     <ds:SignedInfo>
3        <ds:CanonicalizationMethod Algorithm="..." />
4        <ds:SignatureMethod Algorithm="..."/>
5        <ds:Reference URI="...">
6           <ds:Transforms .../>
7           <ds:DigestMethod Algorithm="..."/>
8           <ds:DigestValue> ... </ds:DigestValue>
9        </ds:Reference>
10    </ds:SignedInfo>
11    <ds:SignatureValue> ... </ds:SignatureValue>
12    <ds:KeyInfo> ....</ds:KeyInfo>
13 </ds:Signature>
```

Figura 3.17: Estrutura da XMLDsig

O valor da assinatura <SignatureValue> (linha 11) contém o valor do resumo criptográfico (linha 8) assinado com a chave privada do assinante. E, por fim, o elemento <KeyInfo> indica a chave que será usada para verificar a assinatura. Sendo assim, esse elemento pode conter um certificado X.509, por exemplo, ou qualquer outra informação sobre a chave do assinante. A especificação *XMLDSig* também prevê que outras infra-estruturas de chaves públicas possam ser utilizadas como, por exemplo, o PGP (com o elemento <PGPData>) ou o SPKI (com o elemento <SPKIData>).

Para verificar a integridade de um documento XML assinado, ou seja, assegurar que este não foi modificado, o receptor deve recalcular o resumo criptográfico e compará-lo com o resumo criptográfico (*digest*) que se encontra na mensagem. Se os dois se relacionam, então a integridade do documento está garantida.

3.4.1.2 *XML-Encryption*

A especificação para cifragem XML (*XML-Encryption - XMLEnc*) (Eastlake *et al.*, 2002) atende ao requisito de confidencialidade em mensagens XML. Esta oferece um modelo para cifragem, decifragem e representação do documento cifrado. Desta forma, duas entidades podem trocar documentos sem se preocuparem com a revelação do seu conteúdo e conseqüente uso indevido da informação por terceiros.

Da mesma forma que a especificação *XMLDsig*, a *XMLEnc* fornece uma estrutura XML que pode representar tanto um documento XML quanto qualquer outro documento digital. Além disso, também permite cifrar o elemento XML completo ou apenas o seu conteúdo, tanto para armazenamento local quanto para transporte em uma rede. Já os mecanismos SSL e TLS cifram toda a mensagem para transportá-la. Outra diferença em relação ao protocolo SSL é que a especificação *XMLEnc* permite permanecer com as informações cifradas mesmo após o final do seu trânsito na rede, o SSL, por sua vez, apenas garante a cifragem durante a sessão estabelecida para troca das informações.

A Figura 3.18 exibe a estrutura que especificação *XMLEnc* utiliza para representar as informações cifradas sobre um único elemento. O elemento raiz <EncryptedData> possui três elementos filhos e guarda informações como o espaço de nomes (linha 1) e, nesse caso, avisa que está aplicando o processo de cifragem apenas sobre um elemento, (linha 2). O primeiro filho do <EncryptedData> é o <EncryptionMethod>, que identifica o nome do algoritmo utilizado para cifragem que, no caso, poderia ser o DES (National Institute of Standards and Technology (NIST), 1992). O segundo filho <KeyInfo>, assim como em *XMLDsig* também guarda informações sobre a chave. O último filho (linha 7) possui o valor do processo de cifragem sobre o elemento, guardado no elemento <CipherValue>.

```
1  <xenc:EncryptedData xmlns:xenc="..."
2     Type="http://www.w3.org/2001/04/xmlenc#Element">
3     <xenc:EncryptionMethod Algorithm="..."/>
4     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
5          <ds:KeyName>...</ds:KeyName>
6     </ds:KeyInfo>
7     <xenc:CipherData>
8          <xenc:CipherValue>B457V645B45..</xenc:CipherValue>
9     </xenc:CipherData>
10 </xenc:EncryptedData
```

Figura 3.18: Estrutura da XMLEnc

O processo de decifragem consiste em capturar o conteúdo do elemento <CipherValue>, ler o algoritmo utilizado para cifragem (linha 3), verificar o elemento que foi cifrado (linha 2) e obter as informações sobre a chave no elemento <KeyInfo>. Por fim, reunir todas essas informações e recuperar o documento original, anterior ao processo de cifragem.

3.4.1.3 *XACML*

O processo de autorização determina se uma ação como, por exemplo, a leitura de um arquivo no servidor ou acesso a um recurso é permitida. Neste sentido, uma empresa pode criar regras de negócios para expor seus serviços conforme a importância de seus clientes, permitindo a uns a execução de alguns procedimentos e a outros privilégios maiores de execução de seus serviços.

Segundo Kafura *et al.* (2003), no contexto de sistemas distribuídos, há muitas dificuldades em relação as políticas de autorização. A maioria dos sistemas utiliza linguagens de políticas proprietárias ou que se aplicam a somente algumas aplicações, conduzindo assim a problemas de interoperabilidade. Além disso, qualquer decisão de autorização que transpõe dois ou mais domínios de autorização requer que cada participante seja capaz de, corretamente, produzir, aceitar e interpretar a informação de autorização proveniente de sistemas completamente diferentes. Todas essas capacidades requerem um acordo nos protocolos, sintaxes e semânticas para cada autorização. Adicionalmente, há a necessidade de se aplicar

as políticas de autorização a cada solicitação realizada, verificando desta forma se o principal que solicita o acesso pode realmente ter o pedido aceito.

A especificação XACML (OASIS, 2005a) (*eXtensible Access Control Markup Language*), da OASIS, é uma iniciativa para resolver os problemas apontados acima. Esta descreve tanto uma linguagem para expressar regras de políticas quanto um protocolo pedido/resposta para decisões de controle de acesso. A linguagem para expressar a política é usada para descrever os requisitos gerais de controle de acesso. O protocolo permite a interação para determinar se a ação poderia ser permitida, e desta forma a resposta é interpretada e o resultado - permitir ou negar - é aplicado.

A XACML define dois elementos principais, baseados em (Yavatkar *et al.*, 2000), para lidar com às políticas de autorização, o PEP (*Policy Enforcement Point*) e o PDP (*Policy Decision Point*). O PEP é o responsável por aplicar a decisão realizada pelo PDP sobre o pedido de acesso a determinado recurso. Além destes dois elementos, a especificação XACML também define um protocolo pedido/resposta, para as trocas de mensagens entre o PDP e o PEP, e mecanismos adicionais, como o PAP (*Policy Access Point*) e o PIP (*Policy Information Point*). O PAP é o ponto de administração da política de autorização e o PIP reúne informações sobre o sujeito, o ambiente e o recurso.

Um pedido de autorização parte do PEP, que cria o pedido e o envia ao PDP, que por sua vez avalia o pedido e retorna a resposta. O PDP toma a decisão depois de avaliar as políticas requeridas de acesso ao recurso e as informações sobre o sujeito, o recurso e o ambiente. Para acessar as políticas, o PDP consulta o PAP, que contém as políticas do sistema. As informações sobre o sujeito, o recurso e o ambiente são conseguidas via o PIP (*Policy Information Point*), que recupera os atributos relacionados.

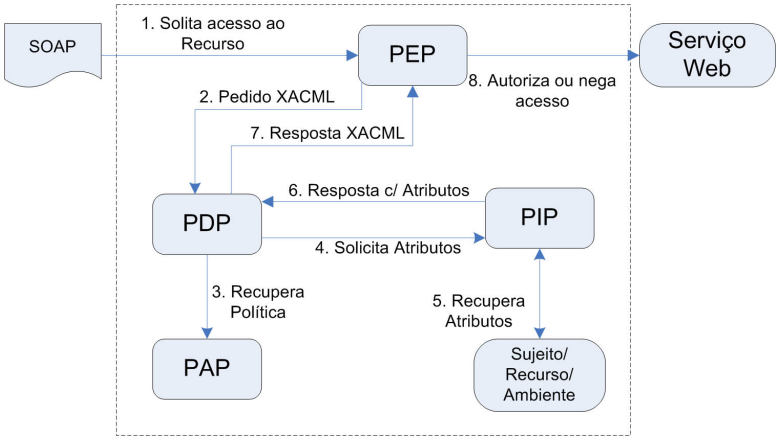

Figura 3.19: Dinâmica do XACML

A Figura 3.19 apresenta a dinâmica da especificação XACML em Serviços *Web*. Inicialmente (passo 1), a mensagem SOAP chega ao PEP, que a repassa ao PDP para avaliação. O PDP consulta a política relacionada ao recurso (passo 3) e em seguida, solicita alguns atributos ao PIP (passo 4). Atributos estes que podem ser o nome ou e-mail do sujeito (passo 5).

Após retornar os atributos (passo 6), o PDP repassa a decisão de autorização ao PEP (passo 7). Finalmente, o acesso ao Serviço *Web* é autorizado ou negado (no passo 8).

3.4.1.4 *SAML*

A especificação SAML (*Security Assertion Markup Language*) (OASIS, 2005b), da OASIS, é um conjunto de especificações e esquemas XML para troca de informações de segurança (OASIS, 2002a). Com o objetivo de ser interoperável, a informação de segurança é expressa na forma de **asserções** sobre sujeitos, no qual, por exemplo, um sujeito pode ser uma pessoa identificada pelo seu endereço de *e-mail*.

Esta especificação vem para cobrir as dificuldades de interoperabilidade entre as infraestruturas de segurança. Tais dificuldades residem tanto nas formas de autenticação quanto nas formas de autorização. Isto se deve ao uso de diferentes tecnologias de segurança, dificultando, por exemplo, a construção de federações de serviços. Uma federação de serviços consiste em entidades que, através da intermediação de confiança, trocam informações e realizam negócios através dos limites organizacionais (Dyke, 2004).

A SAML surge então como um importante padrão aberto [8] para federações formadas de diversos domínios de segurança através de ambientes distribuídos. Permite às companhias implementarem soluções com base na "confiança portável", no qual um usuário autentica-se uma única vez (*Single Sign On* - SSO) em uma entidade e usufrui dos direitos concedidos com esta ação em todas as demais entidades participantes, segundo políticas estabelecidas. O modelo básico de SSO é mostrado na Figura 3.20, no qual um cliente se autentica no seu provedor de identidade (*identity provider*), recebe uma asserção de autenticação, e é, conseqüentemente, reconhecido no provedor de serviços (*service provider*) de em outro domínio.

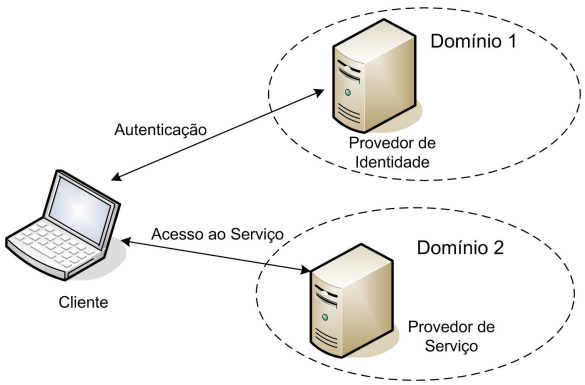

Figura 3.20: Cenário de autenticação única (SSO)

A especificação SAML também se consolida como um padrão altamente promissor para uso em gerenciamento de identidades federadas. No **gerenciamento de identidades fede-**

[8] Por ser um padrão aberto possibilita a adoção do padrão sem que as organizações se tornem dependentes de plataformas ou mecanismos proprietários.

radas cada empresa constitui um domínio, nos quais estão presentes os seus provedores de serviço, de identidades e de credenciais. Desta forma, são estabelecidos acordos entre tais domínios, que permitem que identidades locais a um domínio sejam aceitas nos demais domínios participantes do acordo (Jøsang and Pope, 2005; Jøsang *et al.*, 2005).

A Figura 3.21 exibe o modelo SAML. Os componentes do modelo são descritos a seguir (OASIS, 2002a):

- **entidade do sistema** (*System Entity*): um elemento ativo do sistema, por exemplo, um processo, um sistema ou uma pessoa ou grupo de pessoas, que possuem um conjunto de competências (*capabilities*);
- **coletor de credenciais**: responsável por coletar as informações sobre o principal, geralmente contidas em credenciais, e apresentá-las ou transferí-las para a autoridade de autenticação ou de atributos;
- **autoridade de autenticação**: assegura que um principal se autenticou em um tempo determinado, e usando um método de autenticação particular;
- **autoridade de atributos**: emite atributos sobre um principal; tal informação pode ser útil para decisões de autorização (grupos, funções, código de contrato, etc);
- **PDP**: local onde a decisão é tomada. Para tanto, avalia a identidade do solicitante, a operação e o recurso solicitado com base em uma política de segurança;
- **PEP**: aplica a decisão realizada pelo PDP.

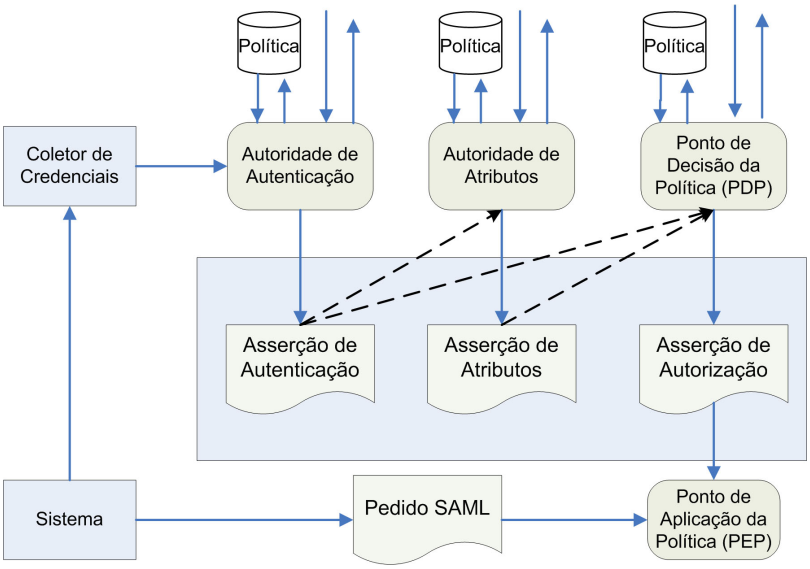

Figura 3.21: Modelo SAML, (OASIS, 2002a)

A Figura 3.21 não deve ser interpretada como um fluxo de mensagens ou um fluxo de processamento. A intenção é descrever quais entidades são responsáveis por produzir as saídas e quais entidades fazem uso de determinadas entradas. Por exemplo:

- um PDP recolhe várias asserções de suas origens para tomar uma decisão baseada na política;
- uma asserção de atributos é retornada à entidade que a solicitou.

Todas as entidades envolvidas podem estar em domínios de segurança distintos ou algumas delas podem residir no mesmo domínio. Normalmente dois domínios de segurança estão envolvidos nas interações e a disposição das entidades pode ser a seguinte, por exemplo:

- A autoridade de autenticação e de atributos em uma mesma entidade;
- PEP e PDP em um mesmo local.

As entidades podem ter múltiplas instâncias quando envolvem a transposição de autenticação e de autorização em domínios de segurança, várias autoridades de atributos ou múltiplos PDPs. Isto pode gerar relações não mostradas na Figura 3.21. Por exemplo, um PDP pode fornecer asserções a outros PDPs.

As asserções são realizadas com base na identidade, nos atributos e nos direitos de um sujeito, ou seja, estas podem transportar informações sobre:

- ações de **autenticação** realizadas por um sujeito, expressando que, por exemplo, a autenticação se deu sobre um mecanismo de segurança HTTPS, com autenticação baseada em senha e num instante específico;
- **atributos** do sujeito, como o seu endereço de *e-mail* ou número de cartão de crédito, etc;
- decisões de **autorização**, as quais podem permitir acesso a determinados recursos.

A Figura 3.22 apresenta um exemplo em que uma asserção de autenticação foi emitida no instante indicado na linha 3 por "www.example.com" (linha 4). O método utilizado para a autenticação foi usuário/senha (linha 7). A asserção foi emitida para o "usuário A" (linha 10) e está digitalmente assinada pelo emissor (linha 17 a 27). A asserção declara ainda outras informações sobre o sujeito, como sua chave pública (linhas 12 a 14).

As asserções são emitidas por uma autoridade SAML. A relação de confiança entre o cliente e a autoridade SAML é um requisito para a emissão das asserções. Uma única autoridade pode emitir os três tipos de asserções possíveis: autenticação, atributos e autorização. A especificação SAML também define um **protocolo** de *pedidos/resposta* pelo qual clientes podem solicitar asserções a autoridades SAML.

Além de asserções e protocolos, o conjunto de especificações SAML também define **mapeamentos** (*bindings*) e **perfis**:

- O **mapeamento** define como os protocolos SAML são organizado nos protocolos de transporte. Neste sentido, o protocolo para pedidos e respostas de asserções poder ser implementado, por exemplo, usando o SOAP sobre HTTP (SAM, 2003).
- Os **perfis** (*profiles*) são definidos para facilitar a interoperabilidade, especificando como as asserções são comunicadas usando determinados mapeamentos.

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
  AssertionID="abc123"
  IssueInstant="2005-11-08T01:10:25.796Z"
  Issuer="www.example.com">
  <AuthenticationStatement
      AuthenticationInstant="2005-11-08T01:10:25.765Z"
      AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
      <Subject>
          <NameIdentifier NameQualifier="www.example.com">
              usuarioA
          </NameIdentifier>
          <SubjectConfirmation>
             ...
          </SubjectConfirmation>
      </Subject>
  </AuthenticationStatement>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
     <ds:SignedInfo>
        ...
     </ds:SignedInfo>
     <ds:SignatureValue>...</ds:SignatureValue>
     <ds:KeyInfo>
        <ds:X509Data>
           ...
        </ds:X509Data>
     </ds:KeyInfo>
  </ds:Signature>
</Assertion>
```

Figura 3.22: Asserção SAML de autenticação

Segundo a OASIS (OASIS, 2005b), as vantagens da especificação SAML incluem:

- **Neutralidade de plataforma**: ao invés de implementações e arquiteturas particulares, a especificação SAML fornece uma forma simplificada de trabalhar com a estrutura de segurança, já que torna a segurança mais independente da lógica da aplicação, que é um importante princípio da Arquitetura Orientada a Serviço.
- **Fraco acoplamento de diretórios**: a especificação não requer que as informações do usuário sejam mantidas e sincronizadas entre diretórios.
- **Melhora a navegação para usuários finais**: asserções de autenticação SAML tornam possível autenticação SSO, permitindo a usuários se autenticarem em um provedor de identidade (*identity provider*) e então acessarem os serviços/recursos no provedor de serviços sem autenticação adicional. Alternativamente, asserções SAML podem

permitir a navegação com o anonimato do usuário. Isso significa que o usuário não terão as suas ações rastreadas na rede.

- **Redução de custos administrativos para provedores de serviços**: diminui os custos para manter informações de contas de usuários, transferido para os provedores de identidade distribuídos na rede.

Outra importante vantagem é a definida em OASIS (2004a) no qual é colocada a possibilidade das asserções serem emitidas como credenciais (*tokens*) de segurança em uma mensagem SOAP, fornecendo assim um importante meio de transferência de autenticação, de autorização e de atributos entre Serviços *Web*.

A versão 2.0 da especificação atende principalmente aspectos relacionados ao uso de identidades federadas. Esse e outros requisitos importantes presentes na versão 2.0 são destacados, a seguir (OASIS, 2005b):

- **Convergência**: a especificação 2.0 unifica as diferentes construções de identidades federadas baseadas em SAML 1.1. Essa nova versão recebeu contribuições tanto do projeto *Shibboleth* (Shibboleth, 2005) quanto da *Liberty Alliance* (Liberty, 2003). Sendo assim, a especificação SAML 2.0 é um passo importante em direção à completa convergência dos padrões de identidades federadas.

- **Pseudônimos**: define um identificador persistente opaco com nenhuma correspondência discernível com os identificadores do usuário (por exemplo, *e-mails* ou nome de contas), podendo ser usado entre provedores para representar os principais. Pseudônimos são a chave para habilitar a privacidade. Um usuário poderia ter diferentes pseudônimos, gerados de forma randômica, para cada associação que ele realizar. Desta forma, a privacidade é protegida, impedindo o rastreamento da identidade do usuário.

- **Gerenciamento de identificadores**: define como dois provedores poderão estabelecer e gerenciar os pseudônimos dos principais com quem operam.

- **Gerenciamento de sessão**: adiciona suporte para "*logout* único", isto é, a habilidade para diferentes sessões estabelecidas em diferentes sítios serem, automaticamente, terminadas. Por exemplo, se um usuário, depois de autenticado em uma autoridade SAML, estabeleceu uma sessão em múltiplos outros sítios, através do mecanismo de autenticação única, e se o usuário tiver qualquer uma das sessões encerradas, isso resultaria em todas as outras sessões serem terminadas.

- **Metadados**: os perfis SAML exigem acordos entre as entidades do sistema a respeito de identificadores, tipos de dados, suporte a ligações (*bindings*), certificados, chaves, e assim por diante. A especificação *metadata*, que acompanha a documentação de SAML 2.0, descreve essas informações de modo padronizado.

- **Cifragem**: permite que declarações de atributos, identificadores de nomes, ou asserções, além de assinadas, como na versão anterior, sejam também cifradas, ou seja, garante confidencialidade fim a fim em elementos da asserção.

- **Dispositivos móveis**: oferece suporte para o contexto de dispositivos móveis, tratando dos desafios impostos pelos dispositivos, dos limites de largura de banda e das oportunidades relacionadas ao surgimento de dispositivos ativos ou inteligentes.

- **Descoberta de provedores de identidade**: quando há um ou mais provedores de identidade relacionados a um provedor de serviços, estes necessitam de um meio para descobrir quais os provedores de identidades o principal usa. A descoberta do provedor de identidades conta com um *cookie* escrito em um domínio comum entre provedores e servidores de identidade.

3.4.1.5 *XKMS*

A complexidade em lidar com as diferentes infra-estruturas de chaves públicas (ICP) pode acarretar dificuldades no uso dos mecanismos criptográficos para as aplicações que trabalham com documentos XML. As operações de cifragem, assinatura, oferecidas pelas especificações *XMLEnc* e *XMLDSig*, respectivamente, são dependentes da ICP subjacente. Desta forma, quando a aplicação recebe um documento assinado com determinado certificado, como o X.509, por exemplo, esta deve ter a habilidade de fazer a verificação do mesmo. Em um cenário em que um provedor de serviços recebe requisições de vários clientes, sua comunicação é limitada a clientes que possuem a mesma ICP, o que não se configura como uma boa solução para ambientes heterogêneos.

A especificação XKMS (*XML Key Management Specification*) (Hallam-Baker and Ford, 2001), desenvolvida inicialmente pela VeriSign em conjunto com a Microsoft e WebMethods, tem como principal objetivo retirar dos desenvolvedores a complexidade em lidar com diferentes ICPs. A especificação consiste, basicamente, em definir um formato para as mensagens e protocolos para disponibilizar uma ICP em um Serviço *Web*. Desta forma, o registro, localização, validação, revogação e recuperação de chaves torna-se acessível em um Serviço *Web* através de serviços, como o **XKISS** (*XML Key Information Service Specification*) e o **XKRSS** (*XML Key Registration Service Specification*), eliminando da aplicação a necessidade de entender a sintaxe e a semântica da ICP.

O XKISS provê um mecanismo para o processamento de informações de chaves públicas associadas às assinaturas e cifragens XML ou qualquer outro uso do elemento chamado `<ds:KeyInfo>`. Como visto na Seção 3.4.1.1, o elemento `<ds:KeyInfo>`contém informações sobre a chave pública de quem assinou ou cifrou a mensagem. Informação esta que pode ser a chave em si, um nome referenciando a chave, um certificado X.509, um identificador de chave PGP, entre outras. Ao receber então um documento XML com alguma operação

criptográfica, a aplicação repassa as informação da chave, presentes em <ds:KeyInfo>, ao XKISS, que suporta duas operações:

- **Localizar** (*Locate*): a função desta operação é localizar informações relacionadas ao elemento <ds:KeyInfo>, uma vez que as informações recebidas inicialmente nesse elemento poderiam não ser suficientes para identificar uma chave (por exemplo, um endereço de *e-mail*). Para obter essas informações, o serviço XKMS pode usar dados locais ou fazer uso de outros serviços XKMS.
- **Validar** (*Validate*): permite realizar as mesmas funções da operação localizar, mas este verifica ainda se as informações relacionadas às chaves são válidas.

Enquanto o XKISS busca localizar e validar informações sobre a chave pública, o serviço XKRSS fornece operações para registro e gerenciamento destas informações. Ou seja, uma aplicação cliente pode requisitar a um serviço XKMS, por meio do XKISS, que registre uma chave pública associando a esta uma informação, como um nome ou qualquer outro identificador. A partir do registro, há também a necessidade da gerência dessa informação. Desta forma, quatro operações são definidas, (Hallam-Baker and Mysore, 2005):

- **Registrar** (*Register*): esta função associa uma informação a uma chave pública. A chave pode ser gerada pela aplicação cliente ou pelo serviço XKMS. Caso o par de chaves tenha sido gerado pela aplicação cliente, a mesma deve provar a posse da chave privada. Ao contrário, se a chave for gerada pelo serviço, o mesmo envia a chave privada para o usuário, podendo permanecer com uma cópia da mesma;
- **Reemitir** (*Reissue*): esta operação reemite associações de informações a chaves públicas, realizadas previamente na operação de registro. O principal objetivo desse serviço é permitir gerar novas credenciais na ICP subjacente, por exemplo, no caso de um certificado expirar;
- **Revogar** (*Revoke*): faz a revogação de um chave previamente registrada. Significa que a chave não é mais confiável para uso em nenhuma aplicação;
- **Recuperar** (*Recover*): recupera a chave privada associada anteriormente. É semelhante à operação de registro e só pode ser executada quando a chave privada tiver sido gerada pelo XKRSS.

O XMKS define modos de processamento destas mensagens, os quais podem ser: **síncronos**, **assíncronos** ou **pedido de duas fases**. No processamento síncrono, o serviço XKMS responde ao pedido sem emitir mais respostas com respeito à requisição realizada anteriormente. Já no processamento assíncrono, o serviço XKMS pode não conseguir completar a requisição imediatamente. Sendo assim, notifica que o mesmo ainda não foi satisfeito e que

respostas subseqüentes serão enviadas. O processamento assíncrono pode ser usado para possibilitar ao serviço intervir durante o processamento do pedido. O pedido de duas fases pode ser utilizado para se proteger de ataques de negação de serviço. Ao receber o pedido, o serviço envia um *nonce*[9] ao cliente, que o fornecerá no seu próximo pedido. Dessa forma, o serviço consegue verificar que o próximo pedido será realizado por uma origem já conhecida.

3.4.2 Especificações de Segurança para Serviços *Web*

3.4.2.1 *WS-Security*

A Figura 3.3 exibiu um conjunto de especificações relacionadas a Serviços *Web*. Destas, a principal especificação de segurança é a *WS-Security* (*WS-Sec*), (OASIS, 2004b) . Proposta inicialmente em 2002 por grandes empresas, como IBM e Microsoft, tornou-se um padrão OASIS em março de 2004. A WS-Sec propõe um conjunto de extensões ao SOAP com a intenção de construir Serviços *Web* seguros fornecendo confidencialidade, integridade e autenticação de mensagens SOAP.

A especificação visa ser flexível e permitir o uso com uma ICP, Kerberos, etc. Isso permite que as empresas utilizem as tecnologias que julguem mais adequadas a estas. Além disso, também suporta vários formatos de credenciais de segurança, domínios de confiança, assinatura e cifragem.

A *WS-Sec* fornece um mecanismo geral para associar credenciais de segurança com as mensagens, permitindo vários formatos de credenciais. As credenciais são classificadas em assinadas (por alguma autoridade específica) e não assinadas; e fornecem um conjunto de declarações sobre o proprietário da mensagem (nome, chave pública, algum direito associado, etc). Porém, os formatos e a semântica dessas credenciais são definidos em especificações particulares. Atualmente, é possível encontrar vários formatos de credenciais de segurança, entre estas o *UserNameToken* (OASIS, 2004c) (não assinada), X.509 (OASIS, 2004d) e Kerberos (OASIS, 2005c) (assinadas). Há ainda a especificação (OASIS, 2004a) que define como asserções SAML (seção 3.4.1.4) são conduzidas em mensagens SOAP.

A confidencialidade e a integridade são obtidas via *XML Encryption* e *XML Signature*, respectivamente (OASIS, 2004b). Tanto as informações de cifragem quanto as de assinatura são associadas às credenciais de segurança para assegurar que as mensagens são transmitidas sem modificação, podendo ainda suportar que vários atores cifrem ou assinem partes das mensagens. Desta forma, a especificação *WS-Sec* fornece uma segurança em nível de mensagem SOAP, evitando assim vários ataques a estas mensagens, como: modificação, interceptação ou personificação.

[9] Valor aleatório enviado de um servidor ou aplicativo.

A especificação *WS-Sec* inclui a segurança nas mensagens SOAP através de um elemento <wsse:Security>. A Figura 3.23 apresenta um exemplo de uma mensagem SOAP assinada com a ICP X.509. O certificado X.509 correspondente à assinatura é anexado como uma credencial à mensagem. Entre as linhas 3 e 17 estão as informações de segurança anexadas pelo padrão *WS-Sec*. Há uma credencial do tipo X.509v3, que identifica o assinante da mensagem (linhas 4 a 7) sendo que na linha 6 está a sua chave pública. A assinatura segue o padrão *XMLDsig* e está presente desde a linha 8 até a 16. Como pode ser observado na linha 13 a assinatura referencia a credencial anexada entre as linhas 4 e 7. A especificação (OASIS, 2004d) ainda prevê que as informações sobre a credencial podem estar em outras posições, como, por exemplo, dentro no elemento <ds:KeyInfo>ou em alguma URI especificada.

```
1  <soapenv:Envelope xmlns:soapenv="..." >
2    <soapenv:Header>
3      <wsse:Security xmlns:wsse="...">
4        <wsse:BinarySecurityToken xmlns:wsu=".." EncodingType="Base64Binary"
5            ValueType="X509v3" wsu:Id="CertId-2037176">
6              MIICTDCCTGVDF...
7        </wsse:BinarySecurityToken>
8        <ds:Signature xmlns:ds="...">
9          <ds:SignedInfo> ... </ds:SignedInfo>
10         <ds:SignatureValue>aTFx/jY4CoS...</ds:SignatureValue>
11         <ds:KeyInfo Id="KeyId-15014700">
12           <wsse:SecurityTokenReference>
13             <wsse:Reference URI="#CertId-2037176" ValueType="X509v3"></wsse:Reference>
14           </wsse:SecurityTokenReference>
15         </ds:KeyInfo>
16       </ds:Signature>
17     </wsse:Security>
18   </soapenv:Header>
19   <soapenv:Body>
20     ...
21   </soapenv:Body>
22 </soapenv:Envelope>
```

Figura 3.23: Modelo de SOAP com *WS-Security*

Desta forma, a integridade da mensagem da Figura 3.23 está garantida por meio da assinatura digital. Além disso, o receptor da mensagem também pode usar as informações presentes na mensagem para autenticar o usuário, uma vez que possui informações sobre o mesmo e pode validar a assinatura para verificar se pertence mesmo a quem está a enviando. Para garantir a confidenciabilidade bastaria cifrar a mensagem.

3.4.2.2 As especificações de Políticas (*WS-Policy*)

Mais um desafio a ser enfrentado por Serviços *Web* é expressar, trocar e processar as condições que governam as interações entre duas entidades. A WSDL é o documento que informa o que um serviço faz, através da descrição de sua interface e preocupa-se com os métodos acessíveis do serviço, o tipo de dados que o cliente deve fornecer, o tipo de

informação que terá de retorno, etc. Porém, um meio de informar ao cliente o que ele precisa fornecer, ou os requisitos que suas mensagens devem satisfazer, para então obter acesso ao serviço é essencial. Por exemplo, quais são as regras de negócios? o cliente deve se autenticar? É preciso que este assine suas mensagens? Qual o tamanho da chave que deve ser usado na cifragem? Desta forma, é necessário definir expressões que limitem e imponham as condições para as trocas de mensagens em Serviços *Web*. Em outras palavras é preciso informar ao cliente a política da aplicação.

A especificação *WS-Policy* (WS-Policy, 2004) proposta por empresas como Microsoft, IBM e Verisign, fornece um modelo de propósito geral para descrever políticas, através de uma gramática flexível e extensível que permite descrever uma ampla variedade de requisitos e habilidades para o ambientes dos Serviços *Web*. Desta forma, esta especificação permite que as entidades envolvidas nas interações definam suas escolhas para a comunicação fazendo uso de uma representação da política do serviço. A *WS-Policy* apóia-se em um conjunto de especificações, como: *WS-PolicyAssertion* (WS-PolicyAssertion, 2003), *WS-PolicyAttachment* (WS-PolicyAttachment, 2003) e *WS-SecurityPolicy* (WS-SecurityPolicy, 2003).

A WS-PolicyAssertion define um conjunto inicial de asserções de política a fim de abordar algumas necessidades básicas dos Serviços *Web*, sendo essas asserções de políticas o elemento básico da WS-Policy. Na WS-Policy uma política é definida como um conjunto de alternativas, onde cada alternativa é uma coleção de asserções de políticas. Algumas asserções de políticas, definidas pela especificação, representam requisitos e habilidades tradicionais, como seleção do protocolo de transporte e de método de autenticação.

Para conduzir as asserções de políticas de forma interoperável, a especificação *WS-Policy* descreve uma maneira para representar uma política, permitindo assim que as entidades processem a política em uma forma padrão. O elemento raiz da forma normal é o <Policy>, seguido do filho <ExactlyOne>. O elemento <ExactlyOne>descreve uma coleção de alternativas de políticas e possui o filho <All>, que por sua vez contém a alternativa de política. A Figura 3.24 ilustra um exemplo.

A linha 1 (Figura 3.24) inicia a descrição da política com o elemento <Policy>. Em seguida o elemento <ExactlyOne>(na linha 2) diz que ao menos uma alternativa de política, das duas alternativas de políticas presentes (linhas 3 a 12), terá que ser satisfeita. As alternativas de políticas expressam que a autenticação no serviço será realizada perante a apresentação de uma credencial do tipo Kerberos, (linha 5) ou uma credencial do tipo X.509v3 (linha 10).

Embora a especificação *WS-Policy* preocupe-se em descrever a política para os Serviços *Web*, ainda é necessário um meio para publicar a política para acesso dos clientes. Desta forma, os clientes, ao verificarem a política requerida para o serviço, saberão se estão ou não aptos para fazer uso do serviço. Neste sentido a especificação *WS-PolicyAttachment*, além de

```
<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 3.24: Exemplo de Política

apresentar meios para associar políticas aos serviços também permite anexá-la à WSDL ou ao UDDI. Na representação na WSDL, a política pode ser associada a uma porta específica (*PortType*). A Figura 3.25 ilustra um fragmento de WSDL com uma política anexa. A linha 3 indica que no documento WSDL há uma política anexa. A política, por sua vez, está representada por uma URI (na linha 7) e diz respeito ao método *GetPaper* definido na WSDL do serviço.

```
<wsdl:definitions ...>
 ...
 <wsp:UsingPolicy wsdl:Required="true" />
 ...
 <wsdl:operation name="GetPaper" parameterOrder="s">
   <wsdl:input message="impl:GetPaperReq" name="GetPaperReq"
      wsp:PolicyURIs="http://das.ufsc.br/policies#TOK"/>
   <wsdl:output message="impl:GetPaperResp" name="GetPaperResp"/>
 </wsdl:operation>
 ...
</wsdl:definitions>
```

Figura 3.25: Exemplo de política anexa a WSDL

As asserções de política são como o elemento básico de uma política. Sendo assim, a especificação *WS-SecurityPolicy* define como as asserções relacionadas à segurança de um serviço são usadas através da especificação *WS-Policy* em conjunto com a *WS-Sec*, *WS-Trust* e a *WS-SecureConversation*. Um exemplo de uma asserção de segurança associada à WS-Trust é ilustrada na Figura 3.26. Esta representa um fragmento de uma política de segurança no qual está definido que há a necessidade de apresentação de uma credencial de segurança do tipo SAML (linha 5) emitida por um STS (*Security Token Service*) especifico (linha 3).

```
<wsp:Policy ...>
   <sp:IssuedToken>
      <sp:Issuer>http://sts.exemplo.br</sp:Issuer>
      <sp:RequestSecurityTokenTemplate TrustVersion="1.1">
         <wst:TokenType="SAML">
      </sp:RequestSecurityTokenTemplate>
   </sp:IssuedToken>
</wsp:Policy>
```

Figura 3.26: Uso da política de segurança associada a WS-Trust

3.4.2.3 especificação *WS-Trust*

Ainda que a segurança nas mensagens esteja garantida por meio da *WS-Security*, as relações de negócios, ou as próprias interações entre duas entidades, geralmente se dão quando há a confiança entre estas. Um provedor de serviços pode ter como requisito - política - realizar negociações apenas se conhece e confia na entidade que faz solicitações. No paradigma de Serviços *Web*, a confiança é estabelecida através da troca de informações entre as partes envolvidas, e pode exigir a presença de uma terceira parte confiável.

A especificação *WS-Security* define como as informações de segurança são adicionadas às mensagens SOAP. Além de apresentar, por exemplo, como se dá a cifragem e a assinatura, a *WS-Security* também especifica como são inseridos as credenciais (*tokens*), ou atributos de segurança, nas mensagens, ou seja, a *WS-Security* aborda a segurança relacionada à mensagem em si. Esta supõe que se duas partes usam um tipo particular de credencial de segurança, essas partes estariam habilitadas a entender tal credencial. Entretanto, existem muitos tipos de credenciais de segurança (certificados X.509, tickets Kerberos, asserções SAML, etc.). Conseqüentemente, não existem garantias de que a entidade que está recebendo as mensagens irá compreender a sintaxe e a semântica da credencial inserida na mensagem. Quando se abrange entidades em diferentes domínios, essa situação se torna mais evidente. Um domínio A pode trabalhar com credenciais X.509 enquanto o domínio B só entende asserções SAML. Para que haja a interação entre estas duas entidades é necessário, além de mecanismos de segurança nas mensagens, meios para que a entidade compreenda a credencial e avalie a confiança na entidade que realiza o pedido. A credencial de segurança, inserida nas mensagens, é apresentada na negociação; é através desta que a entidade poderá avaliar se a entidade que está recebendo as mensagens confia ou não na entidade que as emite.

A *WS-Trust* (WS-Trust, 2005), desenvolvida por um conjunto de empresas lideradas por Microsoft e IBM, é uma complementação da *WS-Security*, que vem fornecer as respostas em relação a como duas entidades podem concordar na natureza e características das credenciais de segurança, ou seja, estabelecer uma **relação de confiança** entre as entidades. Esta especificação define, fazendo uso da *WS-SecurityPolicy*, uma sintaxe para que as credencias de segurança sejam emitidas, trocadas e validadas. A *WS-Trust* também trata a comunicação

entre diferentes domínios cujas relações de confiança foram estabelecidas, permitindo além da troca, da validação e da emissão de credenciais, a troca de atributos.

A especificação *WS-Trust* trata diretamente das seguintes questões:

- **formato da credencial**: a credencial anexa à mensagem SOAP pode estar em um formato no qual o seu receptor é incapaz de compreender; por exemplo, um receptor pode não possuir as funcionalidades para analisar ou interpretar uma credencial no formato de *ticket* Kerberos;
- **confiança**: o receptor pode ser incapaz de construir uma cadeia de confiança entre a sua autoridade (Kerberos, X.509) e o emissor que assina a credencial;
- **espaço de nomes**: as declarações presentes na credencial devem estar numa sintaxe entendível pelo receptor.

Para abordar as questões acima, a WS-Trust define um modelo de confiança. Nesse modelo, se um cliente não possui as credenciais solicitadas pelo provedor de serviços, este a solicita a uma autoridade que as possui, chamada de *Security Token Services* - STS. O STS forma a base para o estabelecimento das relações de confiança, sendo o responsável por emitir, trocar e validar as credenciais. Consiste de um Serviço *Web* que implementa uma interface WSDL, especificada pela WS-Trust, e processa mensagens SOAP seguras. Para a comunicação com o STS, a especificação também define, em sua WSDL, um protocolo pedido e responsta: *RequestSecurityToken* (RST) e *RequestSecurityTokenResponse* (RSTR).

A Figura 3.27 ilustra o modelo geral de segurança da WS-Trust. Este modelo, que inclui declarações sobre o sujeito, políticas e credenciais de segurança, suporta muito mais modelos específicos como autorização baseado em identidade (*identity-based authorization*), lista de controle de acessos e autorização baseado em competências (*capabilities-based authorization*). Permite o uso de tecnologias existentes como certificados de chave pública X.509, credenciais baseadas em XML, *tickets* Kerberos, e mesmo senhas (*password digest*). Segundo a especificação WS-Trust (WS-Trust, 2005), o modelo geral em conjunto com o *WS-Security* e *WS-Policy* é suficiente para construir um alto nível de trocas de chaves, autenticação, controle e acesso, auditoria e complexas relações de confiança.

No cenário da Figura 3.27, o cliente, após ter acesso a política de qualidade de proteção do serviço (WS-Policy), requer a credencial junto ao STS (passo 1). Após a obtenção da credencial, o cliente a apresenta ao Serviço *Web* (passo 2). O Serviço *Web*, por sua vez, verifica se confinça na credencial de segurança apresentada pelo cliente via consulta ao seu STS (passo 3).

Quando um serviço *Web* recebe a mensagem com a credencial anexa no seu cabeçalho, executa os seguintes passos:

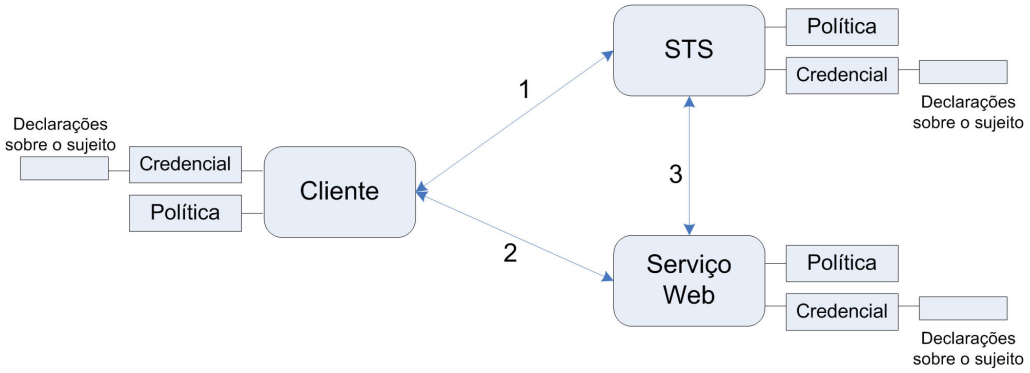

Figura 3.27: Modelo de confiança WS-Trust, (WS-Trust, 2005)

- verificar se a credencial, e o seu conteúdo, é suficiente para cumprir com a política e se a mensagem está adequada à política, ou seja, está assinada pelo assinante, cifrada, etc.
- verificar se os atributos do cliente são provados pela assinatura do STS. Em modelos de relação mediada, onde a confiança está em quem emite a credencial, a assinatura pode não verificar a identidade do requerente e sim a identidade do intermediário (STS), que simplesmente declara a identidade do cliente.
- verificar se o emissor das credenciais de segurança é confiável para emitir as declarações sobre o cliente. Caso não consiga fazer isso, deve solicitar ao seu STS que o faça.

O formato básico de uma mensagem SOAP solicitando por uma credencial pode ser visto na Figura 3.28. Entre as linhas 2 e 4 está o cabeçalho, que deve apresentar a informação necessária para o cliente se autenticar no STS (segundo a WS-Security), fornecendo assim as informações que o STS necessita para verificar quem está fazendo a solicitação e se o mesmo possui os privilégios para tanto, segundo a política definida no STS. O elemento <wst:RequestSecurityToken>(linha 6) inicia o pedido pela credencial definindo um contexto de segurança para a troca das mensagens subseqüentes. Nesse exemplo, o cliente solicita a emissão (linha 8) de uma credencial do tipo SAML (linha 7).

```
1  <soapenv:Envelope xmlns:soapenv="..." xmlns:wst="...">
2     <soapenv:Header>
3        ...
4     </soapenv:Header>
5     <soapenv:Body >
6       <wst:RequestSecurityToken Context= "...">
7          <wst:TokenType> SAML </TokenType>
8          <wst:RequestType> emissão </RequestType>
9      <wst:RequestSecurityToken>
10   </soapenv:Body>
11 </soapenv:Envelope>
```

Figura 3.28: Formato de solicitação por uma credencial de segurança junto ao STS

A Figura 3.29 exibe o fragmento da mensagem SOAP que solicita a credencial solicitada pelo cliente. O STS também poderia adicionar requisitos de segurança à mensagem (linhas 2 a 4) como, por exemplo, assinar e cifrar a mensagem. Além disso, também poderia anexar um certificado X.509. A resposta da credencial de segurança vem no elemento da linha 6, chamado <wst:RequestSecurityTokenResponse>, que contém a credencial solicitada (linhas 9 a 15). Como a credencial solicitada foi do tipo SAML, esta está em nome do cliente e assinada pelo STS (linha 12) .

```
1  <soapenv:Envelope xmlns:soapenv="..." xmlns:wst="..." xmlns:saml="...">
2     <soapenv:Header>
3        ...
4     </soapenv:Header>
5     <soapenv:Body >
6       <wst:RequestSecurityTokenResponse>
7         <wst:TokenType>SAML</TokenType>
8         <wst:RequestedSecurityToken>
9            <saml:Assertion AssertionID="..."
10              ...
11             <ds:Signature>
12               <-- assinada pelo STS -->
13             </ds:Signature>
14              ...
15           </saml:Assertion>
16         </wst:RequestedSecurityToken>
17    </wst:RequestSecurityTokenResponse>
18    </soapenv:Body>
19 </soapenv:Envelope>
```

Figura 3.29: Formato de resposta com credencial de segurança anexa

O modelo da Figura 3.27 ilustrou um caso no qual um simples protocolo pedido/resposta (RST/RSTR) entre cliente e STS foi suficiente para a aquisição da credencial. No entanto, há cenários no qual um conjunto de trocas de mensagens entre as partes se faz necessário antes de retornar uma credencial. A especificação *WS-Trust* define uma extensão ao protocolo base (pedido/resposta) para possibilitar negociações e desafios (Figura 3.30). Um desafio pode ser apresentado a uma entidade solicitante, no caso o cliente, para que o mesmo apresente alguma prova ou atributo para então obter a credencial solicitada.

No exemplo da Figura 3.30, um cliente inicialmente solicita ao STS a aquisição de uma credencial por meio de um RST. O STS analisa o pedido e propõe um desafio ao cliente. O desafio poderia ser: “apresente uma credencial emitida por uma entidade que confio, o STS A”. Neste caso, o cliente primeiro teria que adquirir tal credencial junto ao STS A, para então voltar a fazer o pedido. De posse da credencial o cliente volta a fazer o pedido (passo 3) e então recebe a credencial solicitada inicialmente (passo 4).

Embora a especificação WS-Trust defina um modelo de confiança com protocolos e várias possibilidade para o estabelecimento da confiança, esta não se preocupa com as tecnologias de segurança subjacentes e nem como a confiança pode ser estabelecida. A princípio esta utiliza relações de confiança já existentes para então criar novas relações. As relações

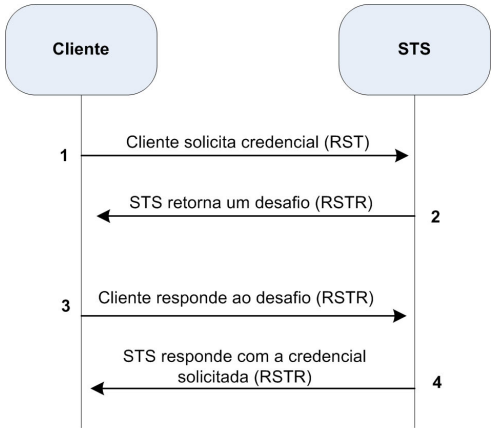

Figura 3.30: Protocolo desafio/resposta, WS-Trust (2005)

de confiança podem ser obtidas através de raízes fixas, onde a priori há um número fixo de entidades confiáveis; confiança hierárquica, onde a confiança se dá como numa estrutura de árvore, no qual os filhos confiam nos nós superiores; ou redes de confiança, onde cada entidade determina em quem confiar.

3.4.2.4 *WS-Federation*

O termo federações está relacionado ao processo no qual entidades intermediam confiança e trocam informações transpondo limites organizacionais (Dyke, 2004). Uma federação pode ser criada com vários propósitos como, por exemplo, para aumentar a rede de relacionamentos das empresas participantes que, conseqüentemente, conseguirão interagir com mais clientes. Diga-se que as empresas com interesse na área de turismo criem uma federação para atender aos seus clientes. Neste caso, um cliente, ao comprar sua passagem em uma empresa associada a essa federação, poderá usufruir das vantagens oferecidas por outras empresas do grupo, como locação de táxis, ingressos para shows, *resorts*, etc. Em um ambiente federado, um usuário de um domínio pode acessar de forma transparente a recursos localizados em outros domínios, que possuem políticas e infra-estruturas próprias.

A especificação *WS-Federation* (WS-Federation, 2003a), proposta pela Microsoft e IBM, define mecanismos para permitir que diferentes domínios de segurança se relacionem e, desta forma, usufruam da mediação de confiança para o compartilhamento de identidades, atributos e autenticação entre os participantes. A especificação *WS-Federation* define um modelo para a federação, porém outros dois perfis são definidos, em especificações separadas: *WS-Federation Passive Requestor Profile* (WS-Federation, 2003c), e *WS-Federation Active Requestor Profile* (WS-Federation, 2003b). A primeira define como trabalhar as funcionalidades da especificação *WS-Federation* em ambientes com clientes passivos, por exemplo, um navegador. A segunda define como a *WS-Federation* trabalha em ambientes com clientes ativos. Clientes ativos são aqueles que estão aptos a emitir mensagens e a reagir com respostas de um Serviço *Web*, ou seja, que possuem habilidades para interpretar uma mensagem

SOAP, verificar se está de acordo com a sua política e retornar um erro ou o resultado esperado, por exemplo.

Segundo (Jøsang and Pope, 2005; Jøsang *et al.*, 2005), o modelo da *WS-Federation* se enquadra no gerenciamento de identidade federado centralizado. Neste modelo, 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 (que será apresentado na Seção 4.2), porém com a diferença de não necessitar do mapeamento de credenciais.

Os modelos definidos em *WS-Security* (OASIS, 2004b), *WS-Trust* (WS-Trust, 2005) e *WS-Policy* (WS-Policy, 2004) fornecem a base para a federação. A *WS-Federation* utiliza as funcionalidades descritas por essas especificações e descreve como os modelos são combinados para permitir a construção de domínios de confiança. Esta parte da premissa de que a confiança já existe ao menos entre duas entidade que formam o domínio de confiança original e, a partir daí, indica como pode se dar o estabelecimento da confiança com outras entidades que desejam utilizar os serviços de uma entidade membro do domínio de confiança original.

A base para o estabelecimento da confiança na *WS-Federation* é o STS (*Security Token Service*), definido em *WS-Trust* (WS-Trust, 2005). Porém, a *WS-Federation* adiciona a este as funcionalidades de Provedor de Identidades (IdP) e o serviço de Atributos/Pseudônimos, combinados ou não em uma única entidade. Um provedor de identidade funciona como um serviço de autenticação, em que os membros se autenticam e fazem uso desta autenticação em todas os membros da federação, ou seja, o acesso aos serviços é feito de forma transparente, sem a necessidade de autenticação adicional (autenticação SSO).

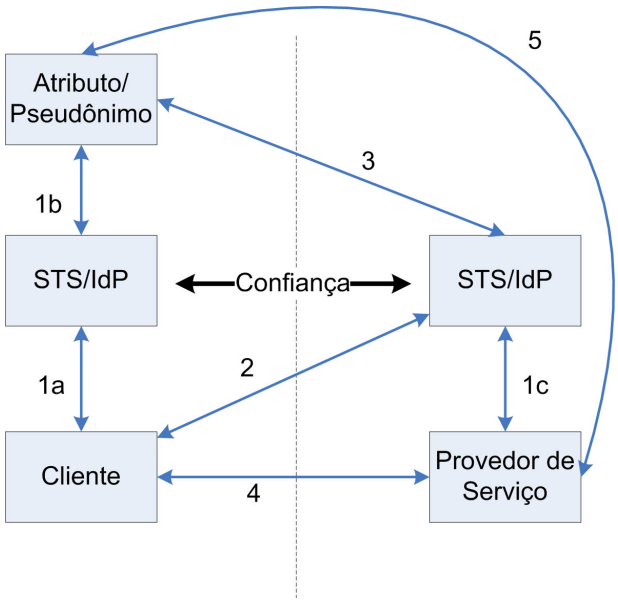

Figura 3.31: Modelo do Serviço de Atributos/Pseudônimos. WS-Federation (2003a)

O serviço de Atributos/Pseudônimos permite que as solicitações do cliente sejam realizadas de forma a proteger a privacidade do usuário, por meio de pseudônimos. O cliente se autentica no seu provedor de identidade, recebe um identificador opaco randômico (pseudônimo), que não é discernível por uma outra entidade, e o apresenta ao provedor de serviço. O provedor de serviço entra em contato com o serviço que emitiu o pseudônimo, caso necessite de informações adicionais sobre o cliente, sem, no entanto conseguir rastrear esse usuário posteriormente.

A Figura 3.31 mostra um cenário em que há a participação do provedor de identidade, do STS e do Atributo/Pseudônimo. Inicialmente o cliente se autentica no seu provedor de identidade (passo 1a). O provedor de identidade solicita um pseudônimo ao serviço Atributos/Pseudônimos (passo 1b). De posse da sua credencial, contendo o seu pseudônimo, o cliente a apresenta ao STS/IdP do provedor de serviço (passo 2). Este, por sua vez, verifica que possui uma relação de confiança com o STS do cliente e atesta se o pseudônimo emitido foi registrado no serviço de Atributos/Pseudônimos (passo 3). Agora o cliente, de posse de uma credencial emitida pelo STS/IdP do provedor de Serviço, invoca o provedor de serviço (passo 4). O provedor de serviço, como possui relação de confiança com o STS que emitiu a credencial ao cliente (passo 1c), poderia também requisitar atributos do cliente, caso o cliente permitisse em sua política (passo 5).

3.5 Interoperabilidade

Segundo (Weerawarana *et al.*, 2005), Serviços *Web* prometem significantes benefícios, tais como reduzir custos e complexidade de conexão de serviços e negócios. A grande alavanca para conseguir os benefícios prometidos e ganhar a atenção e credibilidade da comunidade tecnológica é a interoperabilidade. Entretanto, Serviços *Web* ainda estão em desenvolvimento, principalmente as especificações que tratam da sua segurança, e com isso várias organizações ainda trabalham na definição de especificações. Conseqüentemente, surgem barreiras à interoperabilidade, como questões relacionadas a ambigüidade na interpretação e pouco entendimento das interações entre as especificações ou padrões. Para o sucesso de Serviços *Web*, as especificações precisam estar em concordância para que haja realmente a interoperabilidade.

A *WS-I* (, WS-I) (*Web Service Interoperability Organization*)[10] é uma organização, aberta, que está liderando a busca de interoperabilidade em Serviços *Web*. Fazem parte da *WS-I* as empresas e organizações que desenvolvem as especificações e estão interessadas na tecnologia. A *WS-I* foi formada especificamente para criação, promoção e suporte de protocolos genéricos para troca de mensagens de forma interoperável. Quando surge uma nova especificação ou uma nova versão, a *WS-I* então a analisa e descreve as possíveis falhas encontradas quanto aos aspectos de interoperabilidade. Entre as responsabilidades da *WS-I* estão

[10] http://ws-i.org/

a criação de perfis (*profiles*), ferramentas de testes, cenários de caso de uso e exemplos de aplicações. Um perfil consiste de um conjunto de especificações de Serviços *Web*, numa versão específica, junto com linhas gerais recomendadas para uso ou exclusão de características inadequadas, ou seja, que comprometem a interoperabilidade. As ferramentas de testes monitoram e analisam as interações entre Serviços *Web* para garantir que a troca de mensagens esteja de acordo com determinado perfil. Os cenários de uso são verificados em aplicações exemplos, implementadas em diferentes linguagens e usando diversas ferramentas, para mostrar a interoperabilidade.

```
1 <wsse:UsernameToken
2    xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext
         -1.0.xsd' >
3    <wsse:Username>Bert</wsse:Username>
4    <wsse:Password>Ernie</wsse:Password>
5 </wsse:UsernameToken>
```

Figura 3.32: Exemplo incorreto de uso do *UsernameToken* - falta *PasswordText*

```
1 <wsse:UsernameToken
2    xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext
         -1.0.xsd' >
3    <wsse:Username>Bert</wsse:Username>
4    <wsse:Password>B5twk47KwSrjeg==</wsse:Password>
5 </wsse:UsernameToken>
```

Figura 3.33: Exemplo incorreto de uso do *UsernameToken* - falta *PasswordDigest*

As especificações que formam a base de Serviços *Web* (XML, WSDL, SOAP e UDDI) são analisadas no Perfil Básico (*WS-I Basic Profile*) (, WS-I) e as especificações de segurança estão em progresso no Perfil Básico de Segurança (*Basic Security Profile* - BSP) (, WS-I). O perfil de segurança reúne as especificações XML de segurança (XMLEnc, XMLDsig, SAML, etc), as de Serviços *Web* (WS-Security) e os diversos tipos de credencias (*UsernameToken*, X509Token, SAMLToken, etc) que são anexos nas mensagens SOAP. Desta forma, O BSP define como essas especificações são combinadas para atingir a interoperabilidade. Os exemplos a seguir (Figuras 3.32, 3.33, 3.34 e 3.35) são retirados do perfil de segurança e exibem a forma incorreta e a correta de como trabalhar com a credencial do tipo *UsernameToken* (OASIS, 2004c) nas mensagens SOAP.

O perfil *UsernameToken* define que a senha pode ser transportada em claro, sem resumo criptográfico (*PasswordText*) ou com resumo criptográfico (*PasswordDigest*). As duas primeiras Figuras exibem um exemplo incorreto porque falta o atributo tipo de senha utilizada e o seu valor. As duas próximas Figuras exibem a forma correta de se trabalhar com o *UsernameToken*.

```
<wsse:UsernameToken
   xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext
       -1.0.xsd' >
   <wsse:Username>Bert</wsse:Username>
   <wsse:Password    Type='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
       username-token-profile-1.0#PasswordText'>
      Ernie
   </wsse:Password>
</wsse:UsernameToken>
```

Figura 3.34: Exemplo correto de uso do *UsernameToken* - com *PasswordText*

```
wsse:UsernameToken xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-secext-1.0.xsd' >
   <wsse:Username>Bert</wsse:Username>
   <wsse:Password
      Type='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile
          -1.0#PasswordDigest'>
      B5twk47KwSrjeg==
   </wsse:Password>
</wsse:UsernameToken>
```

Figura 3.35: Exemplo correto de uso do *UsernameToken* com *PasswordDigest*

3.6 Conclusões

Os Serviços *Web* estão baseados na Arquitetura Orientada a Serviço e o oferecem grandes vantagens para uso na integração de aplicações e nas transações comerciais entre empresas através da Internet (*business-to-business*). Vantagens estas antes limitadas pelas tecnologias anteriores, como, por exemplo, CORBA, devido ao forte acoplamento desta última e incapacidade de atravesar os *firewalls*.

Se por um lado a tecnologia de Serviços Web motiva as organizações a oferecerem seus serviços no formato descrito pela tecnologia e assim potencializarem suas relações comerciais. Por outro lado, expor seus fluxos de negócios e trocar informações com clientes em um ambiente inseguro como a Internet é motivo de precauções. Vogels (Vogels, 2004), apontou uma série de limitações da tecnologia e entre estas a necessidade de garantir uma comunicação segura fim a fim. Os esforços realizados por organizações como a OASIS, W3C, IETF, em agregar segurança às especificações que forma a base da arquitetura de Serviços Web, tem resultado em um conjunto numeroso de especificações de segurança, prova da urgência de agregar propriedades de segurança à tecnologia de Serviços Web.

Este capítulo apresentou um conjunto de especificações de segurança projetados para Serviços *Web*. Tais especificações atendem vários requisitos de segurança necessários para a adoção de Serviços *Web* seguros. As especificações *WS-Security*, *WS-Policy* e *WS-Trust* em conjunto com as especificações de segurança para XML, trazem uma boa solução de segurança. No entanto, estas especificações agregam complexidade à tecnologia. Algumas

especificações de segurança carecem de uma maior maturidade e de desenvolvimento perante a dinamicidade dos Serviços Web. Por exemplo, as especificação *WS-Policy*, (WS-Policy, 2004) e *WS-SecurityPolicy* (WS-SecurityPolicy, 2003) até a data de conclusão deste trabalho ainda não haviam sido aprovadas por organizações padronizadoras, como OASIS ou W3C. Por trás da padronização está o interesses de gigantes do mundo da informática, como a Microsoft, Sun Microsystem, etc, o que pode trazer dificuldades de interoperabilidade entre as especificações devido ao jogo de interesses.

Os esforços para manter a interoperabilidade estão concentrados na organização *WS-I*. Apesar disso, a interoperabilidade frente a diferentes ambientes de segurança, com mecanismos de autenticação distintos ainda não está bem fundamentada. Como um domínio de segurança sobre uma tecnologia X.509 irá conversar com um domínio SPKI desconhecido? As credenciais de segurança definidas para o mundo de Serviços Web ainda não abrangem diferentes tecnologias. A credencial de segurança para o Kerberos se tornou um documento oficial OASIS apenas em fevereiro deste ano, conforme (for the Advancement of Structured Information Standards, 2006).

A diversidade de especificações pode também dificultar a adoção da tecnologia por parte de empresas e organizações. Estas empresas não querem maiores preocupações em gerenciar e entender diversas especificações e sim em utilizar a tecnologia.

Diante do exposto acima, fazer uso da tecnologia de Serviços Web para atingir a transposição de autenticação e autorização é ao mesmo tempo uma solução interessante e desafiadora. Interessante uma vez que a tecnologia apresenta meios para atravessar os diferentes domínios administrativos. Desafiadora porque agregar um cojunto de especificações de segurança a solução não é uma tarefa simples. Definir e implementar um modelo de transposição que faz uso da tecnologia de Serviços Web e dos conceitos de gerenciamento da identidades federadas é a proposta desta dissertação.

Capítulo 4

Gerenciamento de Identidades Federadas e as Relações de Confiança em Serviços *Web*

4.1 Introdução

O capítulo anterior apresentou a tecnologia de Serviços *Web* com suas especificações, vantagens, desvantagens e desafios. Desafios estes que, no contexto deste trabalho, se concentram principalmente em garantir a segurança nas interações entre duas entidades desconhecidas e, além disso, com diferentes infra-estruturas de segurança.

Devido à dinâmica do ambiente dos Serviços *Web*, onde serviços são agregados, às vezes de forma temporária, para atender um determinado fluxo de negócios, o gerenciamento de identidade ganha uma atenção cada vez maior. As especificações *WS-Trust* (WS-Trust, 2005), *WS-Federation* (WS-Federation, 2003a) e SAML (OASIS, 2002a) definem protocolos e mecanismos para auxiliar na tarefa de estabelecer uma relação de confiança e gerenciar as identidades dos usuários, conforme visto no capítulo anterior. Basicamente, estas especificações definem uma autoridade, ou seja, uma terceira parte confiável (*Trust Third Party* - TTP) (Kimery and McCord, 2002), responsável por emitir credenciais de segurança para um principal, assegurando que este é quem diz ser e que possui determinados atributos, como um identificador, uma chave pública, etc. Desta forma, as especificações apresentam meios para permitir a federação de identidades e assim obter a transposição dos mecanismos de autenticação e de autorização através da mediação da confiança.

Segundo (Jøsang and Pope, 2005; Jøsang *et al.*, 2005), no gerenciamento de identidades federadas cada empresa constrói um domínio, onde estão presentes os seus provedores de serviço, de identidades e de credenciais. Os acordos estabelecidos entre os domínios permitem que identidades locais a um domínio sejam aceitas nos demais domínios participantes

do acordo. A idéia por trás de criar identidades federadas é basicamente permitir que o usuário, com uma identidade registrada em seu domínio, acesse recursos em outros domínios da federação, sem abertura de novo registro (e de nova identidade) no domínio que detém o recurso.

Segundo (Jøsang *et al.*, 2005), a concretização de identidades federadas tem como ponto fundamental as relações de confiança entre provedores de serviços e clientes, e entre os próprios provedores de serviço. Em (Wu and Weaver, 2005), o estabelecimento de uma relação de confiança é apontado como crucial para concretizar as relações de negócios, seja em um ambiente de identidades federadas ou não. O estabelecimento de uma relação de confiança é definido por (Wu and Weaver, 2005) como o mecanismo pelo qual uma entidade confia em uma segunda entidade para executar um conjunto de ações ou emitir um conjunto de afirmações.

Este capítulo tem por objetivo apresentar uma breve introdução aos conceitos relacionados ao gerenciamento de identidades e destacar a importância do estabelecimento das relações de confiança em Serviços *Web* para concretizar as relações de negócios. Além disso, este capítulo apresenta os esforços da literatura para atingir a transposição de autenticação e autorização através do conceito de identidades federadas e/ou para o estabelecimento da confiança entre Serviços *Web*.

4.2 Formas de gerenciamento de identidades

Segundo (Shim *et al.*, 2005), a maioria dos provedores de serviços que disponibilizam recursos com um certo valor agregado, sejam estes Serviços *Web* ou não, são protegidos por algum mecanismo de autenticação. Isto obriga os provedores de serviços a manterem registro de todos os seus clientes. Desta forma, para acessar estes recursos, os clientes devem então fornecer a sua identidade digital ou então efetuarem o seu registro no sistema. Neste cenário, os problemas de gerenciamento de identidades estão presentes nos dois lados, uma vez que clientes e provedores de serviços precisam lidar com uma grande quantidade de informações.

O gerenciamento de identidades consiste de um sistema integrado de políticas, processos de negócios e tecnologias que permitem às organizações controlar o acesso aos recursos providos aos seus usuários de forma segura e se preocupando em garantir confidencialidade às informações dos usuários. Em (Jøsang and Pope, 2005; Jøsang *et al.*, 2005) o gerenciamento de identidades é apresentado em três diferentes modelos: **tradicional**, **federado** e **centralizado**.

No modelo **tradicional** cada provedor de serviço fornece as identidades e credenciais para acesso a seus serviços. Neste modelo, cada cliente do provedor possui identificadores únicos e específicos para cada serviço com o qual interajam. Conseqüentemente, para prove-

dores de serviços é custoso gerenciar as informações dos usuários e para estes últimos a dificuldade está em utilizar distintos identificadores para cada serviço do qual faz uso.

O modelo de gerenciamento de **identidades federadas** aparece para cobrir as limitações apresentadas pelo modelo de gerenciamento tradicional. Neste tipo de ambiente, cada empresa constitui um **domínio**, onde estão presentes os seus provedores de serviço, de identidades e de credenciais. No ambiente de identidades federadas, são estabelecidos acordos entre os domínios, os quais permitem que identidades locais a um domínio sejam aceitas nos demais domínios participantes do acordo. Neste caso, é estabelecido o mapeamento dos identificadores de um usuário em diferentes domínios. Por exemplo, ao adentrar em um outro domínio o identificador do usuário é traduzido para outro identificador.

O modelo **federado centralizado** parte do princípio que há apenas um 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. 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 onde especifica o *provedor de identidade* que tem o objetivo de autenticar os usuários, permitindo que estes usufruam desta autenticação em todos os serviços da federação.

Conforme (Damiani *et al.*, 2003), o gerenciamento de identidade também envolve aspectos relacionados com a definição, certificação e gerenciamento do ciclo de vida das identidades digitais, infra-estruturas para troca e validação dessas informações, juntamente com os aspectos legais. Em (Damiani *et al.*, 2003) é apresentado um estudo sobre os problemas inerentes ao gerenciamento de identidades, descrevendo os requisitos necessários que um sistema deste tipo deve atender. Dentre os requisitos apresentados, alguns estão diretamente preocupados com as necessidades de segurança dos clientes, visto que geralmente os conceitos de segurança estão direcionados à proteção dos provedores de serviços, deixando de lado os requisitos de segurança dos clientes. Nesse sentido, (Jøsang *et al.*, 2005) afirma que a concretização de identidades federadas tem como ponto fundamental o estabelecimento de relações de confiança entre provedores de serviços e clientes, e entre os próprios provedores de serviço, para auxiliar em interações confiáveis.

4.3 Estabelecimento das Relações de Confiança em Serviços *Web*

Um dos mais poderosos atrativos da tecnologia de Serviços *Web* é a possibilidade de manter relações de negócios com diversos parceiros através da Internet. Porém, como qualquer transação na Internet, a presença de serviços confiáveis é uma premissa para que tais relações ocorram de forma segura. O termo confiança em (Jøsang *et al.*, 2005) é apontado como o nível no qual um entidade aceita depender de alguma coisa ou alguém em uma dada situação,

tendo assim um sentimento de relativa segurança, mesmo que consequências negativas sejam possíveis.

Em relação ao termo confiança, (de Mello and da Silva Fraga, 2005) diz que este pode assumir diversos sentidos em uma aplicação distribuída. Em segurança, o mais usual é garantir que as informações foram enviadas por uma origem confiável. Nesse caso, há somente a preocupação em garantir as propriedades de autenticidade e de integridade das mensagens. Porém, a confiança não se restringe a garantir simplesmente tais propriedades. Em uma aplicação distribuída, as informações trocadas entre os clientes e os provedores de serviços possuem um certo valor e a manipulação indevida das mesmas pode acarretar prejuízos para ambos os lados.

Em se tratando de Serviços *Web*, a maioria dos trabalhos que tratam das relações de confiança (conforme será visto na Seção 4.4) fazem uso de credenciais de segurança emitidas por Autoridades de Asserções (*Authorities Assertion*), definidas no padrão SAML (ver Seção 3.4.1.4), ou Serviços de Credenciais de Segurança (*Security Token Services* - STS) (ver Seção 3.4.2.3), definidos na especificação *WS-Trust*. Sem um destes mecanismos, as entidades teriam que estabelecer e gerenciar a confiança individualmente, comprometendo a escalabilidade (Skogsrud *et al.*, 2003). Ao invés de manter confiança em pares com todos os potenciais parceiros, as entidades formam uma relação de confiança com o STS ou com a Autoridade de Asserção e então, com base nessa confiança, formam outras relações de confiança.

Diante disto, a confiança entre as entidades e estabelecida por uma autoridade distinta que emite credencias de segurança (por exemplo, X.509, asserções SAML, tickets Kerberos, etc). A credencial de segurança traz em si informações sobre a identidade do usuário ou outras características envolvendo os atores. Os provedores de serviço são capazes de relacionar um nível suficiente de confiança em uma credencial de segurança porque confiam na sua origem, quer dizer, conhecem e confiam na autoridade que emitiu uma credencial e podem verificar, através de mecanismos criptográficos, que a credencial foi emitida por tal autoridade. É através da confiança existente em uma terceira parte, emissora da credencial de segurança, que os provedores de serviços são capazes de atribuir confiança indireta ao possuidor da credencial de segurança. Porém, como já dito anteriormente, provedores de serviços estão sob uma tecnologia de segurança e nem sempre estão aptos a entender uma credencial de segurança em formato diferente da sua tecnologia.

4.4 Revisão da Literatura

Como apontado acima, 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. Nesta seção são apresentadas a revisão da literatura sobre gerenciamento de identidades federadas e o estabelecimento das relações de confiança.

Os projetos *Liberty Alliance* (Liberty, 2003) e *Shibolleth* (Shibboleth, 2005) são especificações abertas que seguem o modelo de gerenciamento de identidade federada e fazem largo uso do padrão SAML (são vistos nas Seções 4.4.1 e 4.4.2, respectivamente). O Cardea (Seção 4.4.3) é um sistema de autorização distribuído que dinamicamente avalia os pedidos de autorização de acordo com as características de quem solicita o acesso e do recurso.

Em Serviços *Web* há diversos trabalhos que abordam o estabelecimento de confiança, envolvendo também políticas de autorização para o acesso aos recursos. O Credex (Seção 4.4.4), faz uso de *WS-Trust* para criar um repositório de credenciais de segurança. Seu objetivo é facilitar o armazenamento e a troca dinâmica de diferentes credenciais de segurança. O trabalho de Dyke, na seção 4.4.5, da universidade de Virgínia, extende a especificação *WS-Trust* com a adoção de *trust level*, *trust groups* e *trust authorities*, tornando possível atribuir níveis de confiança a entidades que se utilizam de determinada forma de autenticação e pertencem a determinado grupo de confiança.

4.4.1 Projeto *Liberty Alliance*

O projeto *Liberty Alliance* (Liberty, 2003) é consórcio formado por mais de 160 empresas de diversas áreas como, por exemplo, governos, universidades, etc. Seu principal objetivo é criar especificações abertas para tratar o gerenciamento de identidades federadas por meio de Serviços *Web*. O trabalho do projeto *Liberty Alliance* envolve várias frentes de trabalho, simultâneas, que fazem uso de padrões e protocolos amplamente utilizados por Serviços *Web*. A Figura 4.1 ilustra os módulos que compõe arquitetura da *Liberty Alliance*.

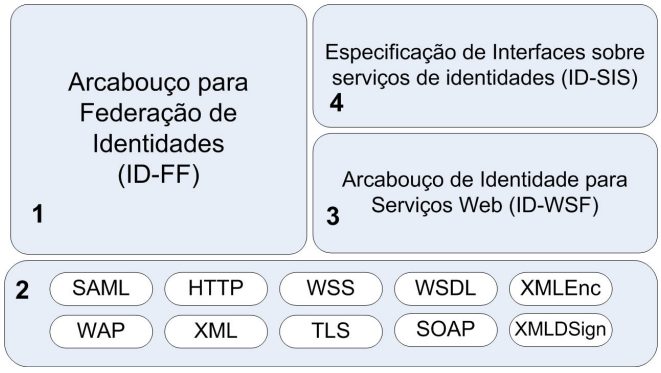

Figura 4.1: Arquitetura da *Liberty Alliance* (Liberty, 2003)

O módulo 1 do projeto *Liberty Alliance* (Figura 4.1), *Liberty Identity Federation Framework*(ID-FF) permite a gerenciamento de identidades federadas. Seu arcabouço é projetado para trabalhar com plataformas heterogêneas e com todos os tipos de dispositivos de rede, incluindo computadores pessoais, telefones celulares, PDAs, etc. Entre suas característica estão: a ligação entre diferentes contas ou identidades, autenticação única (SSO), e o gerenciamento de sessão de forma simplificada.

O módulo 2 demonstra a preocupação do projeto *Liberty Alliance* em adotar e estender padrões já consolidados e padronizados por diversas organizações como OASIS, W3C e IETF. Vale destacar a importância do padrão SAML que é amplamente utilizado pela *Liberty* para transportar as informações dos usuários. A própria *Liberty* foi a responsável por diversas características presentes na versão 2.0 do padrão SAML (seção 3.4.1.4).

O módulo 3, ID-WSF é uma camada fundamental que irá utilizar o módulo ID-FF e visa definir um arcabouço para criar, descobrir e processar serviços de identidade, sendo responsável por oferecer aos usuários serviços personalizados e mais adequados. Por fim, o módulo 4, o *Liberty Identity Services Interfaces Specifications*(ID-SIS) reúne uma coleção de especificações para construção de serviços interoperáveis sobre a ID-WSF. Este pode incluir serviços como registro, agenda de contatos, calendário, localização por GPS, etc.

O projeto *Liberty Alliance*, com a construção de identidades federadas, visa permitir ao cliente de uma federação uma única autenticação no seu provedor de identidade (SSO) para que este acesse aos diversos recursos espalhados pela federação e disponíveis via os provedores de serviços. As informações pessoais do cliente (os seus atributos) se encontram nos provedores de identidade da federação, ou seja, não precisam estar armazenados de forma centralizada. Para que provedores de identidades e de serviços interajam e compartilhem as informações do cliente estes precisam estabelecer uma relação de confiança através de contratos pré-estabelecidos entre as partes. Nesse contrato há a concordância em respeitar a privacidade do usuário, o compartilhamento das informações, etc. Ou seja, define-se a política que rege as interações entre as partes. O projeto *Liberty Alliance* chama o grupo de provedores de serviços que acordam em tais contratos de **círculos de confiança**, segundo (Liberty, 2004). Tais círculos consistem na federação de provedores de serviços e serviços de identidades, juntamente com os clientes.

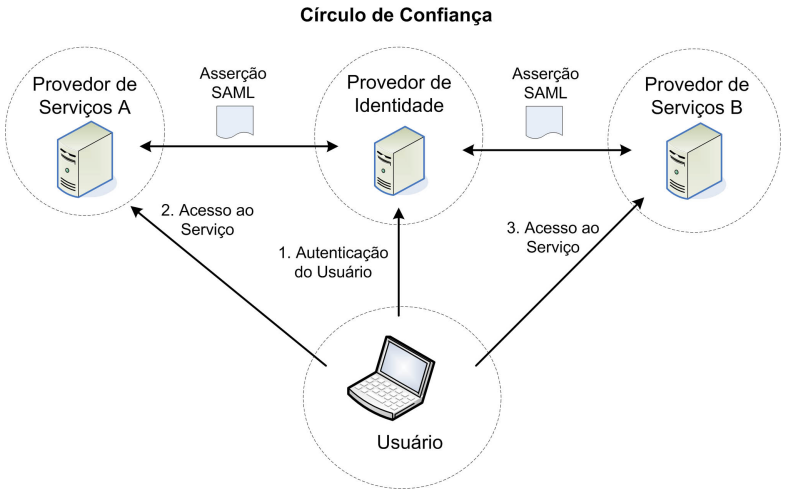

Figura 4.2: Exemplo de Círculo de Confiança

A Figura 4.2 apresenta o círculo de confiança e exibe, de forma simplificada, o funcionamento da federação de identidade no projeto *Liberty Alliance*. Um usuário, após se autenticar

no seu provedor de identidade (passo 1) faz uso dos provedores de serviço A e B sem a necessidade de nova autenticação (passos 2 e 3). A troca de informações entre provedores de identidade e provedores de serviço obedece o protocolo definido na especificação SAML. A asserção SAML é responsável por transportar as informações do usuário pelos diversos domínios do círculo de confiança.

4.4.1.1 Modelo de Confiança da *Liberty Alliance*

A *Liberty Alliance*, em um documento não normativo (Liberty, 2004), fornece um guia com uma variedade de modelos que podem ser aplicados para o estabelecimento da confiança. O consórcio prevê que interações entre entidades que não possuem um contrato pré-estabelecido poderão ocorrer. Um exemplo, é quando o usuário filiado a um determinado provedor de identidade tenta acessar recursos que não pertencem ao círculo de confiança do seu provedor de identidade. O documento introduz uma lista de negócios (*Business Anchor List* - BAL) e uma lista de confiança (T*rust Anchor List* - TAL). A primeira é uma lista que contém todas as entidades com as quais uma entidade estabeleceu contratos de negócios. Já a segunda é uma lista com as chaves criptográficas das entidades com as quais se relacionam e que são usadas para verificar a autenticação das entidades. Basicamente, essas listas objetivam verificar se existe um contrato de confiança pré-estalecido e/ou a confiança nas chaves criptográficas utilizadas na autenticação. A partir destas duas listas, uma variedade de modelos para o estabelecimento da confiança é sugerida (Liberty, 2004), conforme a Figura 4.3.

Confiança nas chaves criptográficos (autenticação)

Confiança para relações comerciais		Direta Chaves simétricas ou públicas	Indireta Confiança na Autoridade Certificadora ou KDCs	Nenhuma
	Direta Acordo bilateral	Confiança Direta	Confiança Direta	Operação Insegura
	Indireta Cadeia de acordos	Confiança Mediada	Confiança Mediada	Operação Insegura
	Nenhuma	Comunidade de confiança	Comunidade de confiança	Comunidade de confiança (fraca)

Figura 4.3: Modelos de Confiança da Liberty (Liberty, 2004)

Na Figura 4.3 as colunas se referem ao suporte a autenticação entre as entidades, assegurando que a identidade da entidade é autêntica. Caso alguma entidade não possua a chave pública de determinada entidade, poderá solicitá-la a alguma entidade do seu círculo de confiança. Já as linhas, referem-se ao contrato de negócios, que "firma o compromisso entre as entidades". Em relação à autenticação, esta pode ser direta ou indireta. Na autenticação direta, considera-se que há uma troca de chaves criptográficas - simétricas ou assimétricas

- e que estas são conhecidas ou as entidades envolvidas já as possuem. Já na autenticação indireta, o reconhecimento da identidade das entidades é possível através de intermediários confiáveis.

A partir do contrato de negócios e das chaves criptográficas, a *Liberty Alliance* define o modelo de confiança direta e o modelo de confiança mediada. O modelo de confiança direta (Figura 4.3) (*Pairwise Trust model*) estabelece uma forte confiança no contexto de negócio, mas tem a escalabilidade limitada. As relações entre todos os participantes são governadas por contratos de negócios assinados entre as partes. Isso gera uma comunidade fechada, no qual as entidades que não possuem o contrato estabelecido não podem ingressar.

O modelo mediado (*Brokered Trust model*) descreve o caso em que duas entidades não possuem um contrato mutuamente estabelecido, mas possuem acordos com partes intermediárias. Desta forma, há a possibilidade da construção de um caminho de confiança que envolva um intermediário e as entidades. Tal abordagem é semelhante à realizada pelo STS da especificação *WS-Trust* (Seção 3.4.2.3).

4.4.2 Projeto *Shibboleth*

Como parte do projeto do grupo de Internet2, o *Shibboleth* (Shibboleth, 2005) fornece um sistema de autenticação e autorização baseado na Internet através do navegador *Web* do usuário. Em relação a interações entre clientes e provedores de serviço, o *Shibboleth* permite apenas que esta seja via navegador do usuário. Um dos seus principais requisitos é permitir interações seguras e interoperação entre sítios *Web* educacionais, apesar de ser útil para qualquer ambiente que queira trabalhar com domínios de confiança. Desta forma, o sistema permite que quaisquer organizações criem federações e usufruam do gerenciamento de identidades federadas.

O projeto *Shibboleth* faz grande uso das especificações XACML e SAML (ver Seção 3.4.1). O primeiro é responsável pelo controle de acesso através das entidades PEP *Policy Enforcement Point*) e PDP (*Policy Decision Point*). Já o segundo, é utilizado para conduzir os atributos dos usuários entre os domínios de confiança.

O modelo conceitual do projeto *Shibboleth* é composto por usuários, sítio origem e sítio destino. Cada sítio que deseja fazer parte da federação *Shibboleth* deve decidir se sua atuação será como origem, destino ou ambos. O sítio origem é responsável tanto por autenticar o usuário, quanto por fornecer atributos sobre o usuário para o sítio destino. Os atributos fornecidos pela origem são assinados por esta, com o intuito de garantir a autenticidade dos mesmos.

A arquitetura do *Shibboleth* compreende um provedor de identidade (*Identity Provider* - IdP) e um provedor de serviço (*Service Provider* - SP) (Figura 4.4). Ambos são desenvolvidos separadamente, mas trabalham juntos para fornecer acesso seguro aos recursos. O

primeiro possui as informações sobre o usuário, é o domínio que o hospeda, responsável pela sua autenticação e o fornecimento de seus atributos. O provedor de serviços é quem hospeda o recurso a ser acessado. Para tanto, possui uma política relacionada ao recurso e que, se necessário, solicita informações extras sobre quem está requerendo acesso ao recurso. Alguns requisitos e definições sobre a arquitetura do *Shibboleth* são apresentados em (Morgan *et al.*, 2004):

- criação e gerenciamento da confiança entre sítios;
- gerenciamento sobre qual informação a origem irá transferir para cada sítio destino;
- definição de um esquema padrão para a comunicação entre domínios;
- prover meios para a transferência segura de atributos;
- definir requisitos que os sítios devem conhecer para participar do projeto *Shibboleth*;
- garantir a privacidade dos usuários, permitindo o acesso de usuários anônimos, porém autorizados, aos recursos.

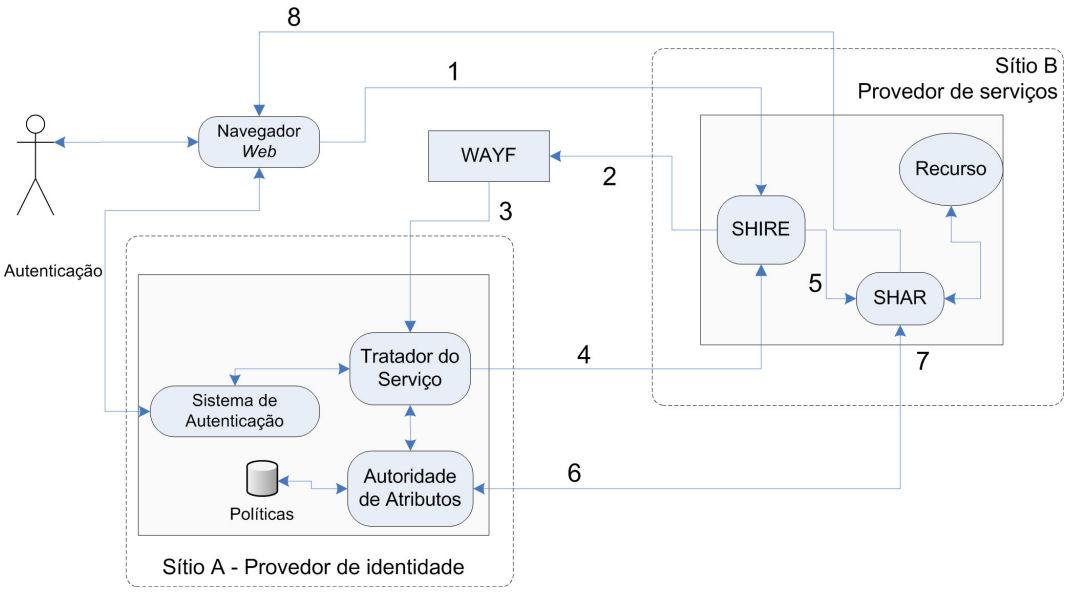

Figura 4.4: Arquitetura do Shibboleth

A Figura 4.4 exibe a arquitetura do *Shibboleth* e um possível cenário envolvendo interações entre os módulos da arquitetura. O provedor de identidade possui os módulos Tratador do Serviço (*Handle Service* - HS) e Autoridade de Atributos (*Atrribute Authority* - AA). O Tratador de Serviço é responsável por verificar se o usuário está autenticado localmente e por criar um tratador (*handler*) para determinar a Autoridade de Atributos que possui as informações sobre o usuário, uma vez que é possível a presença de várias AAs em um mesmo domínio. O *Shibboleth* não determina como isso é feito, apenas diz que faz parte da implementação do sistema. A função da Autoridade de Atributos é fornecer atributos sobre

um usuário, mas esta também pode especificar um meio do usuário determinar quais atributos este deseja dispor quando visita um sítio estrangeiro. Já o Provedor de Serviços possui os módulos SHIRE (*Shibboleth Indexical Reference Establisher*), WAYF (*Where are you from*) e SHAR (*Shibboleth Attribute Requester*). O SHIRE é o módulo responsável por descobrir qual o domínio de origem do usuário que está solicitando acesso ao recurso e criar um manipulador para este usuário. Para auxiliar o SHIRE, o WAYF questiona o usuário sobre o seu domínio de origem. O WAYF conhece o endereço dos Tratadores de Serviços pertencentes a federação e, após o usuário apontar de qual domínio pertence, entra em contato com o Tratador de Serviço do usuário. O SHAR tem por função interagir com a Autoridade de Atributos do usuário para obter os seus atributos.

As interações do cenário da Figura 4.4 iniciam quando um usuário solicita, através de seu navegador *Web*, acesso a um determinado recurso (passo 1). Por exemplo, um artigo hospedado em determinada universidade. Ao verificar o pedido de acesso, o SHIRE aciona o WAYF (passo 2) que questiona o usuário sobre o seu domínio de origem, por exemplo, a universidade à qual este pertence. O WAYF apresenta uma lista de universidades ou organizações ao usuário, que então informa a sua origem. Após isso, o WAYF entra em contato com o Tratador de Serviço do usuário (passo 3), que verifica se o usuário já está autenticado, caso contrário, realiza a autenticação do mesmo. A autenticação obedece a infra-estrutura subjacente do domínio, podendo ser baseada em senha ou via uso de uma ICP. Como parte da resposta ao SHIRE, o Tratador de Serviço retorna o endereço da Autoridade de Atributos do usuário (passo 4). O SHIRE então repassa o endereço ao SHAR (passo 5) que é o responsável por obter os atributos do usuário necessários para o acesso ao recurso (passo 6). A Autoridade de Atributos, ao receber o pedido por atributos, verifica a política de privacidade do usuário e então, conforme a política, retorna os atributos solicitados (passo 7). O SHAR verifica se o pedido por atributos foi atendido e então fornece ao usuário acesso ao recurso (passo 8). Lembrando que tanto a solicitação por atributos quanto a resposta com os atributos solicitados são realizadas via asserções e protocolos SAML.

O *Shibboleth* define uma forma padronizada para a troca de atributos, através do SAML, porém não especifica os atributos em si, deixando tal função livre para os desenvolvedores. Em um ambiente federado, a padronização destes atributos é tida como crucial (Morgan *et al.*, 2004). Para que a federação relacionada a serviços educacionais tenha uma forma padronizada o projeto Internet2 estabelece a InCommon Federation [1], (InCommon, 2006) que é responsável pela definição da gramática que rege a troca de atributos no *Shibboleth*.

4.4.3 Cardea

Cardea (Division, 2003) é uma sistema de autorização distribuído, desenvolvido como parte da Grade Computacional de Informações da NASA (*NASA Information Power Grid*)

[1] http://www.incommonfederation.org

(Foster *et al.*, 1998; Foster and Kesselman, 1998; Nasa, 2003). Construído sobre especificações como SAML, XACML e XMLDSig, este sistema avalia dinamicamente os pedidos de autorização, considerando as características dos recursos e do solicitante ao invés de avaliar apenas identidades locais (Figura 4.5). Neste sistema, os recursos dentro de um domínio administrativo são protegidos por uma política de acesso local, especificada com a sintaxe XACML, em termos de características do solicitante e do recurso. Além disso, os usuários são identificados por certificados *proxy* X.509 (Tuecke *et al.*, 2004).

Propostos no mundo de Grades Computacionais, os certificados X.509 *proxy* são criados a partir de um certificado X.509 padrão. O detentor do certificado X.509 padrão delega a um principal algum direito através do certificado X.509 *proxy*. Basicamente, certificados X.509 *proxy* são certificados X.509 assinados (ou emitidos) por uma entidade ao invés de uma autoridade certificadora (*Certificate authority* - CA). No campo `Issuer` (ver seção 2.6.2), o certificado *proxy* é identificado pela chave pública do certificado X.509 original ou por outro certificado X.509 *proxy*. Esta opção permite que o certificado seja criado, dinamicamente, pelo possuidor do certificado, sem a intervenção das autoridades certificadoras. Ao contrário do padrão X.509, onde o sujeito de um certificado é definido pela autoridade certificadora, o campo `Subject Name` (ver seção 2.6.2) do certificado X.509 *proxy* permite a definição de sujeitos dentro do escopo de nomes do emissor do certificado. A restrição do escopo deve ser feita para que os certificados sejam únicos. A chave pública do certificado X.509 *proxy* é distinta da chave pública do seu emissor. Uma característica importante dos certificados X.509 *proxy* é que estes possuem um tempo de vida limitado (aproximadamente de 2 a 168 horas), principalmente por questões de segurança.

A Figura 4.5 exibe a arquitetura do sistema Cardea. Esta é, basicamente, uma união das entidades definidas na especificação SAML e XACML (Seção 3.4.1), Sendo assim, é formada por uma ou mais entidades PEP (*Policy Enforcement Point*), PDP (*Policy Decision Point*), PAP (*Policy Access Point*), autoridades de atributos (AA) e um Serviço de Informações, o qual age como um repositório de atributos do usuário. Todas essas entidades e serviços trocam informações, de forma padronizada, para avaliar dinamicamente um pedido de autorização.

O Cardea define dois pontos de decisões de políticas, o SAML PDP e o XACML PDP. O SAML PDP é responsável pela emissão de asserções de atributos e de autorização. Para tanto, pode consultar um XACML PDP, que é o responsável por avaliar a política do sistema via consulta a pontos de administração da política (PAP) para só então emitir uma asserção de atributos ou de autorização. Além disso, o SAML PDP pode também consultar o Serviço de Informações, para obter atributos do solicitante e as informações sobre as autoridades de atributos. Lembrando que, neste sistema, considera-se que as informações do usuário não estão centralizadas.

A separação, das identidades dos usuários, das informações de autorização, divide os dados em diversas autoridades de atributos, que podem residir em domínios administrativos

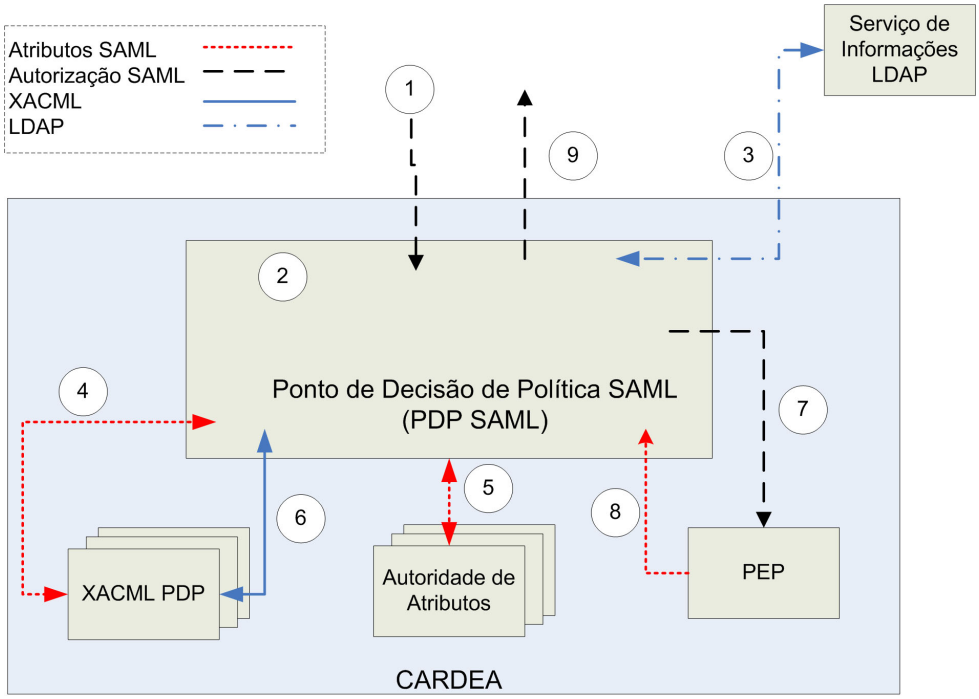

Figura 4.5: A arquitetura do Cardea

distintos. Essas autoridades recebem os pedidos de asserções de atributos e, uma vez que os PDPs locais podem não conhecer o sujeito de um pedido, é necessário que o pedido contenha informações suficientes para localizar qualquer autoridade, a fim de obter os atributos necessários para o processo de autorização.

Embora as entidades do sistema possam estar localizadas numa mesma máquina, estas também podem estar distribuídas e suas funcionalidades expostas como Serviços *Web*. A comunicação entre as entidades é feita via protocolo definido nos padrões XACML e SAML. A integridade das mensagens é atingida utilizando a especificação XMLDSig.

Cada decisão administrativa no Cardea é processada por um algoritmo geral, o qual requer um mínimo de conhecimento prévio dos participantes. Inicialmente, conforme passo 1 (Figura 4.5), o sistema recebe um pedido de acesso a algum recurso, por exemplo, que exige a apresentação de um conjunto de atributos do solicitante. O SAML PDP então precisa decidir se pode ou não fornecer os atributos para acesso ao recurso e, além disso, verificar se estes estão nas demais autoridades do sistema. Para tanto, o sistema realiza uma busca em seus registros (passo 2) para encontrar as possíveis autoridade de atributos que contenham as informações solicitadas no pedido inicial. No passo 3, o sistema solicita informações sobre tais autoridades no Serviço de Informações com o intuito de as localizar e, conseqüentemente, solicitar atributos. Após consultar a política de autorização do recurso, no XACML PDP (passo 4), o SAML PDP constrói uma mensagem SOAP solicitando informações as autoridades de atributos (passo 5). Cada autoridade de atributos consultada avalia se pode ou não fornecer os atributos via consulta às suas políticas locais. De posse dos atributos solicitados, o SAML PDP faz nova consulta à política de autorização do recurso, ou seja,

ao seu XACML PDP (passo 6), a fim de verificar se os atributos adquiridos são suficientes para o acesso. Após a avaliação, envia uma decisão de autorização ao PEP (passo 7), este responsável por aplicar a decisão do SAML PDP, aceitando ou negando acesso ao recurso.

4.4.4 CredEx

Em se tratando de Serviço *Web*, há várias formas de um principal se identificar. Este pode fornecer uma credencial (*token*) do tipo usuário/senha, uma mensagem assinada com um certificado X.509 anexo ou ainda através de um *ticket* Kerberos, etc. Um provedor de serviços também é livre para adotar o mecanismo de autenticação que desejar, obrigando o principal a se adequar ao mesmo. Conseqüentemente, o principal teria que ter a habilidade de gerenciar diversas credenciais, uma para cada serviço, o que trás incômodos e também acarreta complexidade nas invocações ao serviços.

O CredEx (Vecchio *et al.*, 2005) é um Serviço *Web* baseado na especificação *WS-Trust* (WS-Trust, 2005), com código aberto, projetado para utilização em Serviços *Web* e Grandes Computacionais (*Grid Services*) (Czajkowski *et al.*, 1999). Seu objetivo é o armazenamento e a troca dinâmica de diferentes credenciais de segurança, tirando dos principais a preocupação com o tipo de credencial requerida para acesso a cada serviço. Inicialmente, os principais recebem uma credencial padrão, que é utilizada para requerer os outros tipos de credenciais, armazenadas em um repositório central. Ou seja, o principal simplesmente armazena as credenciais que utiliza para acesso aos diversos serviços e, conforme a sua necessidade, utiliza a credencial padrão para solicitar a credencial correspondente a determinado serviço. A troca de credenciais é feita pelo sistema de forma transparente ao cliente, uma vez que este apenas informa a sua credencial padrão e o sistema recupera a credencial correspondente para acesso ao serviço.

Entre as características do Credex, estão (Vecchio *et al.*, 2005):

- suporte a autenticação única (SSO) mesmo perante múltiplos tipos de credenciais;
- suporte a uma credencial padrão, que é utilizada nos pedidos de troca por outras credenciais;
- uma implementação baseada em padrões abertos, como WS-Security (OASIS, 2004b) e *WS-Trust* (WS-Trust, 2005);
- armazenamento central de múltiplas credenciais do usuário;

O modelo do CredEx é composto, basicamente, por um repositório centralizado, denominado Servidor de Credencial. Este possui duas interfaces, chamadas *SecurityTokenService* e *CredencialMananger*. A primeira segue a especificação da *WS-Trust* e a segunda fornece

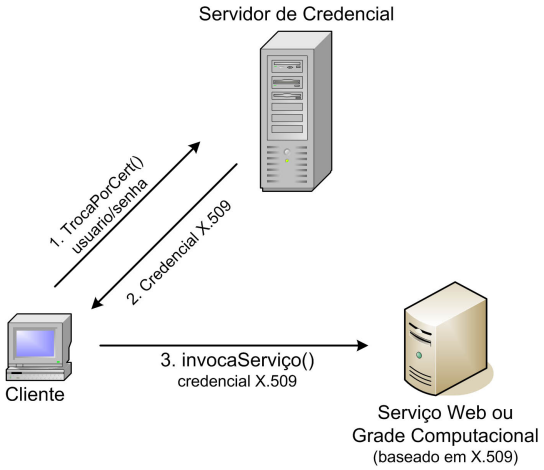

Figura 4.6: Cenário de troca de credenciais - CredEx

métodos e protocolos para armazenar, recuperar e remover credenciais do tipo usuário/senha e X.509. O armazenamento de uma nova credencial é feita pela associação desta com uma credencial padrão, através do método `storeCertInit`. Uma credencial do tipo usuário/senha, por exemplo, é associada a uma credencial do tipo X.509. Com isso, quando o cliente desejar acessar algum serviço basta trocar sua credencial padrão pela credencial solicitada pelo serviço, através do método `exchangeForCert`. O cliente passa como parâmetros usuário/senha (definido no momento do armazenamento), para se autenticar no Servidor de Credencial, um apelido e o intervalo de tempo em que deseja utilizar a credencial; e, como resposta ao pedido, obtêm a credencial solicitada para acesso ao serviço.

A Figura 4.6 exibe um cenário de troca de credenciais do CredEx. No passo 1, o cliente solicita a troca da sua credencial usuário/senha por uma credencial do tipo X.509, necessária para a acessar o Serviço *Web* (que também poderia ser uma Grade Computacional). O Servidor de Credencial devolve a credencial solicitada no passo 2. De posse da credencial X.509 requerida pelo Serviço *Web*, o cliente então realiza o acesso ao mesmo (passo 3).

A troca de credenciais também poderia ser realizada pelo provedor de serviços. Neste caso, quando o cliente realiza acesso a um serviço, este apresenta a sua credencial padrão. Caso um serviço não entenda a credencial fornecida, este pode então solicitar ao Servidor de Credencial que a troque por uma credencial que esteja de acordo com a sua tecnologia.

4.4.5 Redes de Confiança Federadas

O trabalho proposto em (Dyke, 2004) apóia-se nas especificações *WS-Trust* e *WS-Federation* e propõe uma arquitetura para construção dinâmica de redes de confiança entre Serviços *Web* no contexto de serviços médicos (*healthcare*). Redes de informações de serviços médicos são sistemas complexos que envolvem vários participantes: médicos, técnicos, equipe administrativa, banco de dados radiológicos, sistemas contábeis. Além disso,

usuários e serviços também se comunicam com entidades externas como companhia de seguros, farmácias e clínicas de saúde. Como o sistema envolve informações muitas vezes sigilosas e pessoais, o trabalho preocupa-se em mensurar a confiança através de níveis de confiança (*trust levels*), grupos de confiança (*trust groups*) e autoridades de confiança (*trust authorities*). Neste trabalho, cada domínio de confiança é composto por um STS que emite credenciais (*tokens*) aos seus diversos usuário para acesso aos serviços da federação. A federação então é composta por diversos domínios de confiança cada qual com seu STS, seguindo a especificação *WS-Federation*.

Os **níveis de confiança** são um valor numérico que representam a confiança ligada a um determinado método de autenticação, ou seja, expressam o nível de confiança que um STS tem no método de autenticação utilizado pelo usuário para declarar sua identidade e é contido na credencial emitida pelo STS ao usuário. O objetivo dos níveis de confiança é permitir que os STSs de cada domínio conheçam, ao receber uma credencial assinada por outro STS, qual o mecanismo de autenticação o usuário utilizou para se autenticar no STS do seu domínio e desta forma, segundo a sua política local, avaliem se podem ou não confiar no usuário. Um nível de confiança zero deveria identificar o método menos seguro (por exemplo, uso de usuário/senha).

Um **grupo de confiança** pode ser descrito como um conjunto de requisitos de segurança exigidos pelos administradores dos domínios, ou validados por uma autoridade de confiança, para que entidades externas venham a fazer parte do grupo previamente definido. Esses requisitos de segurança poderiam incluir critérios como nível de confiança mínimo, requisitos de política definidos através das especificações *WS-Policy* e *WS-SecurityPolicy* (requerer uso de determinado algoritmo para cifrar as mensagens, por exemplo), etc. Tais critérios são altamente subjetivos; entretanto, se membros do grupos de confiança os definem de tal forma que a autoridade de confiança pode validar seus membros, através de uma aplicação ou processo, estes podem ser úteis para auxiliar no estabelecimento dinâmico das relações de confiança.

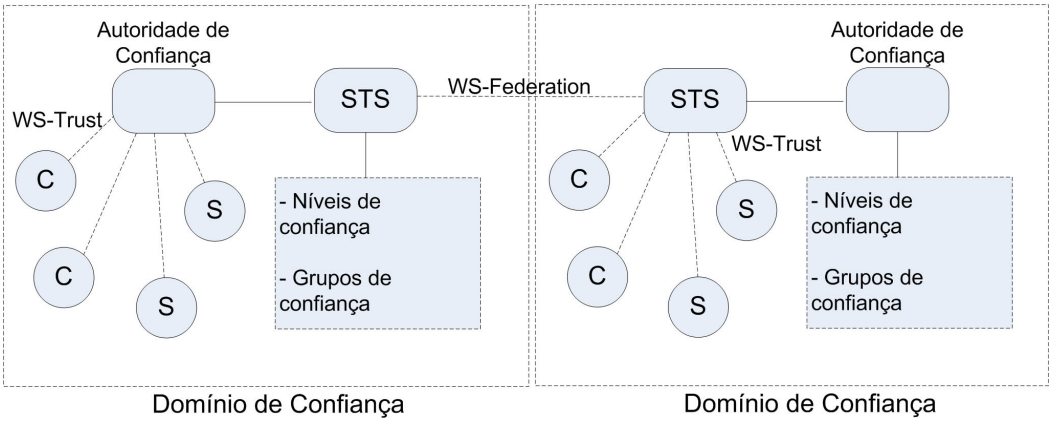

Figura 4.7: Modelo - Federações em Sistemas Médicos

As **autoridades de confiança** funcionam como entidades que gerenciam as relações de

confiança do domínio, atualizando as políticas, gerenciando os membros, etc. Para gerenciar a confiança estas mantém um repositório de níveis de confiança, grupos de confiança e definições de política para um ou mais domínios de confiança. Ao receber uma credencial, em um pedido de acesso por recurso, a autoridade de confiança é então responsável por verificar o nível de confiança e as exigências do grupo de confiança para permitir ou não acesso ao recurso. As autoridades de confiança podem estar junto ao STS ou separadas.

A Figura 4.7 ilustra modelo proposto. A cada STS está associada uma autoridade de confiança que possui grupos e níveis de confiança. Quando um serviço recebe um pedido de algum cliente, este verifica o nível e o grupo de confiança para então autorizar ou negar o acesso. Desta forma, as relações de confiança são criadas dinamicamente após a avaliação dos níveis de confiança e grupos de confiança pela autoridade de confiança.

4.5 Considerações sobre a Revisão da Literatura

A literatura apresentada neste capítulo representa um esforço no sentido de unir as vantagens oferecidas por Serviços *Web* e o gerenciamento de identidades federadas. O estabelecimento das relações de confiança aparece com um ponto fundamental nessa associação. Em todos os trabalhos, constata-se a preocupação em manter as propriedades de segurança sem comprometer a flexibilidade nas interações entre clientes e provedores de serviços. Além disso, os trabalhos visam que informações de autenticação e de autorização atravessem domínios e sejam reconhecidas pelo maior número de autoridades e provedores de serviço.

O projeto *Liberty Alliance* (Liberty, 2003) se apóia no conceito de gerenciamento de identidades federadas e em Serviços *Web* para definir um arcabouço com o objetivo de estabelecer federações nos mais diferentes ramos de negócios. O projeto permite a seus membros uma autenticação descentralizada e observa as propriedades de segurança, como privacidade e anonimato. Além disso, faz uso de padrões reconhecidos como o SAML, permitindo a autenticação única (SSO) a seus membros e a utilização de navegadores Web nas relações de negócios. No entando, as relações de confiança são estáticas e novas relações são criadas apenas por meio de contratos de negócios. Além disso, provedores de serviços são obrigados a adotarem o padrão SAML e a seguirem uma séria de normas para disponibilizarem seus serviços na federação definida pelo projeto *Liberty Alliance*.

O projeto *Shibboleth* (Shibboleth, 2005) está baseado no conceito de federações, permitindo a transposição dos mecanismos de autenticação e autorização e o compartilhamento de recursos. O projeto permite que diferentes infra-estruturas de segurança sejam utilizadas na federação, sem a necessidade de readequação destas. No entanto, o projeto *Shibboleth* apenas considera as interações do cliente com o provedor de serviços via navegador *Web* do usuário, não abrangendo cenários onde há serviços interagindo com outros serviços.

O sistema Cardea (Division, 2003) explora a autorização descentralizada com a união dos

padrões SAML e XACML. Está consolidado sobre o gerenciamento de identidades e traz boas idéias em como gerenciar um conjunto de recursos perante as identidades localizadas em diferentes domínios administrativos. Porém, o sistema é baseado na ICP X.509. Em Serviços *Web*, é desejável que diversas tecnologias de segurança sejam acomodadas. Além disso, o sistema Cardea está focado em Grades computacionais.

O Credex (Vecchio *et al.*, 2005) apresentou uma boa solução para lidar com a heterogeneidade das infra-estruturas de segurança. Como limitações, o CredEx permite apenas a troca entre dois tipos de credenciais: X.509 e usuário/senha. Desta forma, considera apenas a utilização de certificados de autenticação pelo cliente e não permite que o Servidor de Credencial atribua autorizações, através de uma asserção SAML, por exemplo, para o cliente acessar recursos específicos. Além disso, o cliente necessita de um prévio registro das suas credenciais no Servidor de Credencial para então solicitar as trocas. A natureza centralizada da autenticação definida no CredEx também oferece um ponto único de falha, trazendo problemas de escalabilidade e disponibilidade.

Já o trabalho Redes de Confiança Federadas (Dyke, 2004) tenta estabelecer a idéia de uma confiança estratificada (ou multi-nível) através de níveis de confiança. Tais níveis, juntamente com as exigências de segurança impostas nos grupos de confiança, devem definir os direitos de um membro em seus acessos a recursos da federação. A limitação deste trabalho está em definir as regras para estabelecer os níveis de confiança. Caso se considere os padrões da ICP X.509, as mudanças para comportar as informações de nível nas credenciais fornecidas por esta PKI devem comprometer a interoperabilidade de usuários fora destas redes federadas. Outro ponto é que o trabalho não especifica uma sintaxe comum aos domínios e, além disso, o trabalho não explora o potencial das asserções SAML, que naturalmente incluem informações sobre a autenticação do usuário e são extensíveis, comportando as necessidades de descrição de níveis de confiança.

4.6 Conclusão

Este capítulo apresentou uma breve introdução dos conceitos relacionados ao gerenciamento de identidades federadas e destacou a importância do estabelecimento das relações de confiança em Serviços *Web* para concretizar as relações de negócios. A literatura apresentada aborda os desafios no sentido de atingir a transposição dos mecanismos de autenticação e autorização.

Para os usuários, a integração entre os provedores de serviços, através do gerenciamento de identidades federadas, traz grandes benefícios. A facilidade de uma única autenticação (*Single Sign-On* - SSO), permite que o cliente efetue o processo de autenticação uma única vez, seja em um provedor de serviço qualquer ou em uma entidade autenticadora centralizada, e usufrua deste processo de autenticação nos demais sistemas e domínios, identificados

como parceiros de negócio. Para as empresas, o compartilhamento de identidades digitais provê uma certa facilidade no gerenciamento dos usuários, visto que usuários de outras empresas não precisarão estar presentes em sua base para que possam interagir com o sistema.

Porém, os trabalhos não são claros quanto a como ocorrem as interações entre os clientes e provedores que possuem diferentes infra-estruturas de segurança e que não desejam um padrão comum como, por exemplo, SAML . Geralmente, estes partem do princípio que o cliente está sob uma tecnologia de segurança comum ao do provedor de serviço ou que o provedor de serviço conhece previamente o cliente, impedindo assim o estabelecimento dinâmico de relações de confiança entre tecnologias distintas e a conseqüente transposição de autenticação, autorização e troca de atributos.

Capítulo 5

Modelo para Transposição de Autenticação através de Identidades Federadas

5.1 Introdução

Como visto na Seção 2.5, a autenticação e a autorização estão entre os principais mecanismos de segurança. Enquanto a autenticação identifica um principal (usuário, máquina, etc) perante o sistema, o processo de autorização decide se as requisições de acesso a objetos feitos pelo principal, devidamente identificado, devem ser ou não permitidas. Ou seja, o processo de autorização começa com uma avaliação de cada recurso protegido para determinar os requisitos que devem ser satisfeitos para permitir acesso a um dado recurso (et al, 2001). Tradicionalmente, a identidade é associada a um exclusivo conjunto de permissões para recursos do sistema. Credenciais (*tokens*) de segurança são criadas para expressar os direitos de um principal.

Quando se considera as possibilidades oferecidas no ambiente da Internet, pode-se vislumbrar uma gama de cenários e relações comerciais. No entanto, em tais cenários, geralmente é solicitado um conjunto mínimo de características a serem cumpridas entre as partes envolvidas para se concretizar as interações. Duas barreiras a serem transpostas são a heterogeneidade das infra-estruturas de segurança, presentes nos diversos domínios coorporativos e de usuários, e o estabelecimento da confiança entre partes desconhecidas.

Em sistemas distribuídos, os modelos usuais de autorização se apóiam em uma autoridade de autenticação para mediar a confiança entre partes desconhecidas. A confiança é alcançada com a apresentação de credenciais emitidas pela autoridade de autenticação conhecida pelas partes envolvidas na autorização. Em ambientes mais complexos, como os suportados via Internet, este modelo de simples intermediação por uma terceira parte confiável se apresenta

como limitado. Os principais, nestes casos, devem interagir com entidades localizadas em outros domínios administrativos. Cada domínio possui suas políticas de segurança, infra-estruturas de segurança próprias e uma forma particular de gerenciar identidades de usuários do domínio. Desta forma, a interoperabilidade [1] entre domínio não é garantida. Não há uma correlação entre as identidades locais de um domínio em outro domínio e cada domínio executa seus controles de autorização levando em consideração suas políticas locais, sem considerar os atributos de outros domínios.

Nestes ambientes distribuídos e complexos, autorizar cada novo usuário exige a criação de novas identidades locais para representar as permissões deste novo usuário em cada domínio. Claramente, o volume de dados para a autorização tende a ser grande e difícil de ser gerenciado.

Propriedades de "transposição" devem estar presentes nestas novas propostas. A idéia da transposição inclui o acesso a recursos controlados em um domínio diferente daquele do cliente, usando atributos de credenciais fornecidos no seu domínio de origem. Isto inclui a permissão para que principais se autentiquem apenas uma vez e usufruam desta permissão nos demais domínios de um sistema distribuído complexo. As autorizações de acesso a recursos em todo o sistema dependerão desta única autenticação em seu domínio de origem (*Single Sign On*), que diminui sensivelmente o fluxo de dados entre domínios e a complexidade da gerência destes dados no sistema distribuído como um todo. Usuários não precisam mais se registrar em vários domínios para usufruírem dos serviços na extensão do sistema como um todo. As estratégias de autorização passam a requerer que cada serviço mantenha um mínimo de conhecimento de seus potenciais colaboradores no sistema global. Cada decisão de autorização, que transpõe dois ou mais domínios, requer que cada entidade do sistema global produza, aceite e interprete corretamente informações de autorização de indivíduos, ou de um grupo de parceiros, provenientes de sistemas ou tecnologias diferentes (domínios heterogêneos).

O capítulo anterior exibiu tanto os esforços da literatura de Serviços *Web* para estabelecer as relações de confiança entre domínios distintos, quanto para permitir que identidades locais a um domínio sejam aceitas em outros domínios participantes de uma federação, no contexto do gerenciamento de identidades federadas (Seção 4.2). O gerenciamento de identidades federadas é uma promissora solução para concretizar a transposição de autenticação e de autorização em sistemas distribuídos. Estas transposições possuem como base identificadores aceitos pelos participantes de uma relação de confiança.

Concretizar a autenticação e a autorização distribuída de forma transparente nestes sistemas complexos é uma tarefa muito árdua, diretamente afetada pela escalabilidade, e que exige grande cooperação entre domínios administrativos distintos e autônomos, principalmente a fim de garantir a interoperabilidade entre domínios. São necessários tanto padrões

[1] interoperabilidade entre diferentes infra-estruturas de segurança

altamente difundidos, com abstrações suficientes para esconder as diferentes tecnologias, quanto modelos que contribuam para integração de tais padrões.

Este capítulo, apoiado na tecnologia de Serviços *Web*, apresenta um modelo cuja função é compor um suporte para transposição de autenticação em sistemas distribuídos complexos, que fazem uso da Internet na comunicação. Apoiado na identidade do usuário, através do conceito de identidades federadas, este modelo concretiza a transposição de autenticação no sistema como um todo mantendo propriedades de transparência e de fácil uso nos acesso. O modelo permite então que com base na autenticação realizada no seu domínio de origem o principal acesse recursos espalhados nos diferentes domínios pertencentes às suas relações de confiança, de forma transparente e interoperável. Desta forma, o modelo se concentra em lidar com diferentes tecnologias de segurança para permitir a autenticação e a autorização local. Para concretizar a autorização, parte-se da premissa que a transposição de atributos é necessária. Os atributos a que se refere estão relacionados a lógica do negócio e são exemplos destes o endereço, o cep, o*email*), etc. Não é intenção deste trabalho lidar com a transposição da autorização.

Neste modelo, cada domínio agrupa clientes e provedores de serviço de acordo com a tecnologia de segurança usada. Neste capítulo, para a definição do modelo, parte-se de uma das proposições da IETF e das especificações do SAML para a definição de um modelo de autenticação e de autorização. Este modelo é inicialmente introduzido em um sentido genérico, na seqüência toma formas mais concretas quando se assume as proposições de padrões e tecnologias de Serviços *Web*, bem como os conceitos de gerenciamento de identidades federadas. O desafio do modelo proposto é lidar com um grande conjunto de especificações de segurança para Serviços *Web* que, em sua maioria, ainda estão em desenvolvimento ou foram recentemente lançadas.

5.2 Modelo de uma Infra-estrutura de Segurança para Aplicações Distribuídas Orientadas a Serviço

O modelo aqui proposto é base de um projeto maior[2], o qual, entre outros, propõe uma infra-estrutura de segurança que faz uso de diversas especificações e propostas para garantir a segurança de aplicações orientadas a serviços. Essa infra-estrutura, além de garantir as propriedades básicas de segurança como integridade, confidencialidade, autenticidade e disponibilidade, irá possibilitar a transferência de autenticação entre diferentes domínios administrativos, mesmo que estes estejam utilizando diferentes tecnologias de segurança.

Este trabalho parte do pressuposto que as relações de confiança se encotram estabelecidas e assim, constrói um modelo para a transferência de autenticação e a consequente autorização

[2]Projeto CNPQ número 550114/2005-0

descentralizada e a transposição de atributos do cliente. Outro trabalho do grupo, desenvolvido em (de Mello, 2005) preocupa-se com o estabelecimento dinâmico da confiança.

5.2.1 Definição de um Domínio

Neste estudo, para atingir a transferência de autenticação, considera-se um ambiente formado por diversos domínios administrativos e apóia-se no conceito de identidades federadas. Clientes e provedores de serviços são agrupados em domínios que possuem um conjunto de entidades e serviços usados para manter as propriedades de segurança, fundamentados em uma tecnologia de segurança. No modelo proposto, como exemplo destas entidades, destacam-se as autoridades de autenticação e de atributos. As verificações de autorização são assumidas no modelo, a princípio, como locais aos provedores de serviço.

Como o trabalho proposto lida com sistemas distribuídos de larga escala, considera-se o mesmo composto por vários domínios usando diferentes tecnologias de segurança. Ou seja, leva-se em consideração, nestes sistemas complexos, a característica heterogênea dos domínios e, como conseqüência, a implantação de diferentes técnicas e serviços na autenticação e na autorização nos domínios do sistema. Domínios baseados em diferentes tecnologias como o Kerberos (Seção 2.6.1), a ICP X.509 (Seção 2.6.2), SDSI/SPKI (Seção 2.6.3), etc., determinam diferentes controles de segurança no acesso aos seus recursos. A compatibilidade entre diferentes domínios não é garantida como premissa do modelo. A total transparência destas diferenças e a transposição dos acessos vai depender da introdução no modelo de novos conceitos.

Outro aspecto importante das proposições apresentadas é que se assume que os controles de segurança no acesso a recursos segue modelos tradicionais. Ou seja, clientes devem provar sua identidade junto a uma autoridade de autenticação e, com base nessa autenticação, que atesta a identidade do usuário, são verificados os direitos associados para efetivar um acesso a um recurso do sistema.

A autenticação e a autorização, dependentes da tecnologia de segurança adotada no domínio, são ilustradas na Figura 5.1. Nesta figura são apresentadas, na execução destes controles, Autoridades de Autenticação, de Atributos e de Autorização, sendo esta última implementada junto ao provedor de serviço, segundo uma política de autorização.

Um típico domínio que faz parte de sistemas distribuídos de larga escala, como o ilustrado na Figura 5.1, pode ser entendido como um domínio corporativo. Serviços de Autenticação, de Atributos, e de Políticas de Segurança são assumidos no modelo como entidades autônomas no domínio que gerenciam suas respectivas bases de dados para as consultas durante as operações de autenticação e de autorização. O Serviço de Autenticação contém informações de registros de clientes e provedores de serviço do domínio. A autenticação sendo dependente da tecnologia de segurança do domínio envolve o uso de informações restritas a esta

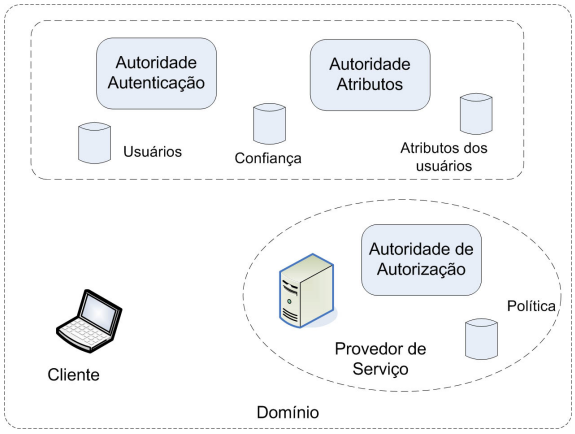

Figura 5.1: Domínio de segurança

tecnologia no processo de autenticação. Por exemplo, se este domínio está fundamentado sobre a ICP X.509, a autenticação será pela assinatura digital com a apresentação do respectivo certificado X.509. Os usuários, durante o processo de registro no domínio, informam atributos normalmente relacionados a lógica do negócio (endereço de *email*, cartão de crédito, CEP, etc) que farão parte das informações gerenciadas pela Autoridade de Atributos.

5.2.2 As Relações de Confiança no Modelo Proposto

As relações de confiança em um domínio podem ser diretas ou mediadas. Na primeira, há confiança entre duas partes, no qual uma confia nos atributos emitidos pela outra, não precisando de um mediador. Já na confiança mediada, a confiança entre duas partes é atingida por meio de uma terceira entidade confiável, responsável pela mediação. As relações de confiança entre autoridades e os seus clientes é do tipo direta. O modelo usufrui das relações de confiança já estabelecidas entre as autoridades de autenticação e de atributos do domínio para mediar a confiança entre clientes e provedores de serviço desconhecidos de um mesmo domínio.

Há também relações diretas entre domínios, neste caso as autoridades de um domínio confiam nos atributos emitidos pelas autoridades de outros domínios, abrindo assim um grande leque para estabelecimento de relações de confiança mediadas.

O modelo proposto neste trabalho tem por objetivo tanto permitir a confiança mediada entre clientes e provedores de serviços desconhecidos quanto permitir que a declaração de autenticação atravesse domínios administrativos, mesmo perante tecnologias de segurança distintas. As relações de confiança são a base para que as identidades e atributos de credenciais sejam reconhecidos nos diferentes domínios participantes destas relações de confiança, possibilitando então, desta forma, a transposição da autenticação. Neste trabalho, parte-se da premissa de relações de confiança previamente estabelecidas (relações estáticas) entre entidades de domínios diferentes. Em (de Mello, 2005), outro trabalho do grupo, é desenvolvido o mecanismo para o estabelecimento de relações dinâmicas de confiança.

5.2.3 Processo de Autorização no Modelo Proposto

A verificação da autorização que ocorre no provedor de serviços nesta proposta objetiva uma autorização descentralizada e está baseada no modelo proposto pela IETF na RFC 2743 (Yavatkar *et al.*, 2000). Definida em um contexto de Qualidade de Serviço (*Quality of Service* - QoS), a RFC descreve dois elementos fundamentais para o governo da política: o PEP (*Policy Enforcement Point*) e o PDP (*Policy Decision Point*) (ver Figura 5.2) [3]. O PEP é o elemento que concretiza o controle de acesso a um recurso. Sua função é aplicar as decisões de políticas. Já o PDP é a entidade que define a política a ser aplicada na requisição de serviço solicitada. É no PDP que as decisões de políticas são tomadas. Para tanto, pode-se usar mecanismos e protocolos adicionais a fim de acessar informações extras, tais como, de autenticação, as informações de política armazenada, etc.

Às proposições da IETF, adicionam-se outras entidades como ponto de partida do modelo. A Figura 5.2 mostra o Serviço de Políticas e as Autoridade de Atributos e de Autenticação como elementos adicionais que participam na autorização de uma requisição. Como se assume que as verificações de autorização no modelo são locais aos provedores de serviço, neste caso, tanto o PEP como o PDP devem possuir suas implementações junto aos provedores de serviços onde controlam as requisições a estes serviços (Figura 5.2).

A Figura 5.2 ilustra o modelo proposto destacando o processo de autorização. Inicialmente, são considerados clientes do domínio já registrados previamente, nos quais informações sobre seus atributos estão disponíveis segundo uma política de privacidade definida pelo próprio usuário. Tal política determina quais atributos o usuário deseja informar aos provedores de seu domínio e também a provedores de outros domínios. O cliente se autentica perante a sua Autoridade de Autenticação (passo 1, Figura 5.2) e recebe uma asserção, declarando sua identidade. Tal asserção declara, por exemplo, que o cliente denominado `cliente@dominoA.com.br` é pertencente ao domínio. A asserção o habilita a solicitar acesso aos recursos em seu domínio.

A concretização de uma política de autorização, neste modelo, se inicia no provedor de serviço com o seu PEP (passo 2, Figura 5.2). Este, ao receber uma requisição de um cliente, através de seu PEP, emite uma solicitação de decisão de política ao PDP (passo 3, Figura 5.2). O pedido do PEP para o PDP pode definir um ou mais elementos de política, além de informações sobre o acesso desejado. O PDP retorna a decisão de política e o PEP a aplica, aceitando ou negando o acesso (passo 9). O PDP pode também retornar informações adicionais ao PEP que incluem um ou mais elementos de política. Estas informações não precisam estar associadas com a decisão de controle de acesso. Preferencialmente, isto pode ser usado para gerar uma mensagem de erro ou passar a mensagem adiante.

O PDP, para tomar a decisão referente a requisição de acesso, pode fazer consultas (passo

[3] Apesar das entidades PEP e do PDP serem originalmente apresentados na RFC 2753 pela IETF, a RFC 3198 da DTMF e IETF é que acorda nos termos PEP e PDP, num glosário de termos relacionado à política.

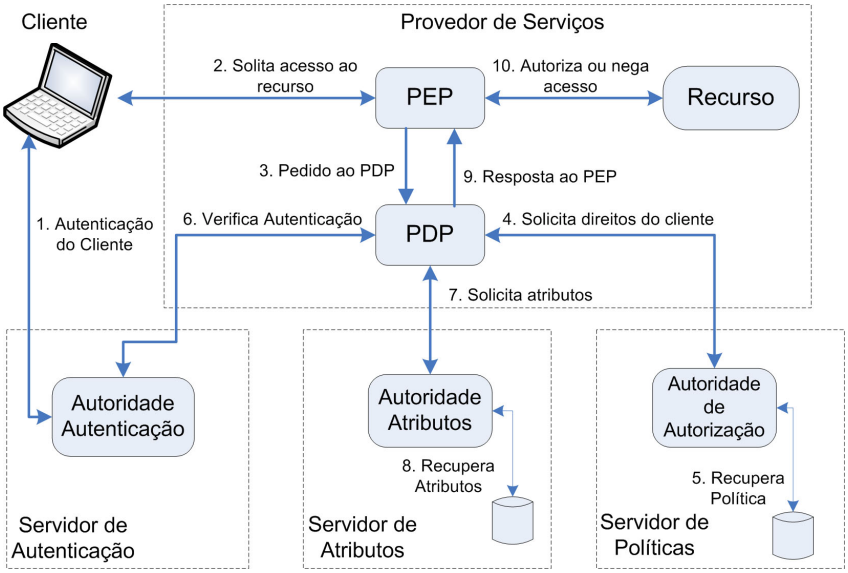

Figura 5.2: Processo de Autorização

4) a sua Autoridade de Autorização (Serviço de Políticas que centraliza as políticas do domínio; serve como um repositório no domínio de políticas) e às autoridades de autenticação e de atributos (passos 6 e 7). O que justifica as interações do PDP com estas últimas autoridades é que, nos controles da autorização, o provedor de serviços, através de seu PDP, necessita certificar a confiança da credencial apresentada pelo cliente e, se isto é suficiente para permitir o acesso. As autoridades de autenticação e de atributos são consultadas no sentido de garantir esta confiança. Mesmo que o cliente seja filiado ao domínio do provedor de serviço, pode haver a necessidade de buscar seus atributos na concretização de sua requisição. Por exemplo, se o cliente é desconhecido do provedor serviço este pode exigir que o cliente forneça o seu endereço e CEP para então autorizar uma compra.

5.2.4 Identidades Federadas no Modelo Proposto

Em grande parte das situações que são descritas neste capítulo, os atributos relevantes do cliente para o acesso ao recurso podem residir em outro domínio, ou seja, o cliente tem seus registros em outro domínio. Para poder implementar então a transposição de autenticação e de atributos neste modelo se faz o uso do conceito de identidades federadas (Seção 3.4.1.4).

No contexto de identidades federadas, os clientes podem estar filiados a mais de um domínio, com contas distintas. Conseqüentemente, possuem diferentes atributos espalhados em domínios distintos. No modelo proposto, ao se assumir identidades federadas também se assume relações de confiança entre domínios. As autoridades de Autenticação e de Atributos, conhecendo os domínios nos quais uma identidade de usuário está federada, fazem requisições sobre os atributos do cliente a tais domínios a pedido de um provedor de serviço. Vale ressaltar que a motivação para que o provedor de serviços busque as informações do usuário vem das propostas de identidades federadas, em que o cliente fornece apenas a sua

identidade ou pseudônimo e o provedor de serviços é quem procura, nos parceiros da sua federação, pelos atributos do cliente.

A abordagem de identidades federadas está então fundamentada em dois princípios: um domínio tem ciência sobre quais domínios a identidade de um cliente está registrada; e que as relações de confiança dos domínios federados se estendem entre suas autoridades de autenticação e de atributos. São relações diretas entre domínios, nos quais as autoridades de um domínio confiam nos atributos emitidos pelas autoridades de outros domínios.

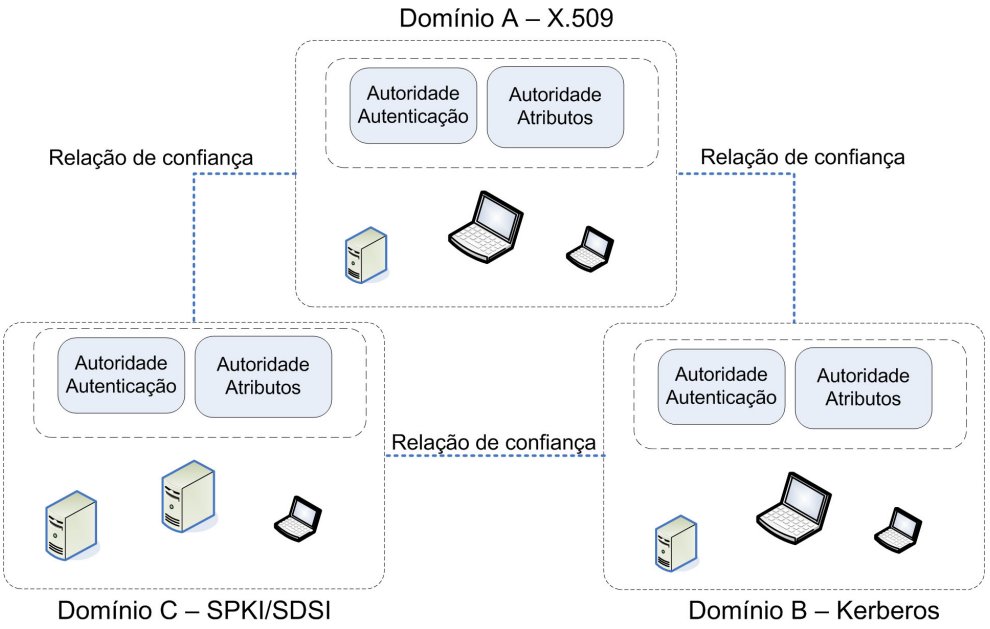

Figura 5.3: Domínios Federados

A Figura 5.3 ilustra um caso de domínios federados. Neste exemplo, o domínio "A" sabe que o cliente também é filiado dos domínios "B" e "C" e pode requerer informações sobre o cliente nestes outros domínios, seguindo, é claro, a política de privacidade do cliente. Os domínios possuem o conhecimento da identidade federada do cliente porque possuem relações de confiança com os demais domínios.

No modelo proposto neste trabalho, o repositório de confiança de um domínio (ilustrado na Figura 5.1) contém a relação de domínios confiáveis e é utilizado pelas autoridades (de Autenticação e de Atributos) do domínio para buscar informações no sentido de avaliar pedidos vindos de domínios externos. Se estes domínios pertencem as relações de confiança do domínio do provedor, será então oferecida ao usuário a possibilidade de federar sua identidade, ou seja, suas informações poderão ser consultadas e estas poderão trafegar entre domínios.

No cenário ilustrado na Figura 5.3, os domínios apresentam infra-estruturas de segurança diferentes. No exemplo, o Domínio "A" faz uso da tecnologia X.509, o "B" está fundamentado nos mecanismos do Kerberos e o "C" usa certificados SPKI/SDSI. As relações de confiança entre os domínios, previamente estabelecidas, devem permitir aos clientes e aos

provedores de serviço, a partir da mediação das autoridades de autenticação e de atributos, a concretização das requisições de serviço.

Nas seções seguintes são desenvolvidas as proposições aqui apresentadas.

5.2.5 Serviços *Web* para a Transposição de Autenticação

A solução proposta para concretizar a transferência de autenticação e de atributos no modelo apresentado se apóia em Serviços *Web* que permitem, entre outros, a integração de aplicações legadas e a interoperabilidade entre diferentes plataformas e sistemas. Fazendo o uso de proposições ou padrões visando a segurança, o modelo apresentado toma a forma orientada a serviços apresentada na Figura 5.4. Especificações como *WS-Trust* (WS-Trust, 2005), a *WS-Federation* (WS-Federation, 2003a), XACML (OASIS, 2005a), SAML (OASIS, 2002a), *WS-Policy* (WS-Policy, 2004), *WS-Security* (OASIS, 2004b), *XML-Encryption* (Eastlake *et al.*, 2002) e *XML-Signature* (XML, 2002) (apresentadas no capítulo 3), fornecem os conceitos e serviços que formam a base da solução proposta, que permite alcançar a transferência de autenticação e de atributos de negócio, respeitando os requisitos de segurança.

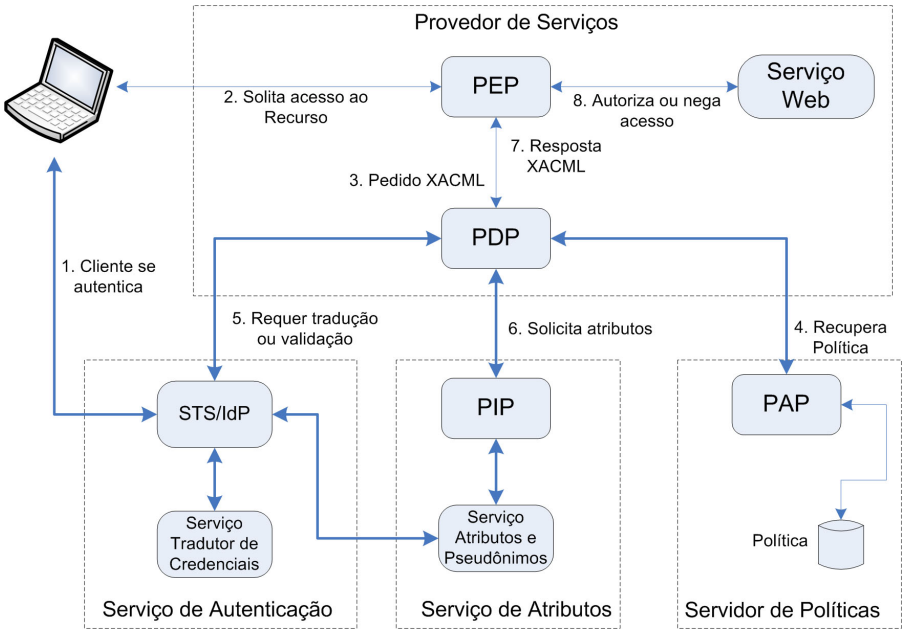

Figura 5.4: Representação do modelo em Serviços *Web*

A Figura 5.4 ilustra a extensão do modelo proposto em uma arquitetura orientada a serviço (AOS). A transposição dos controles de autenticação no modelo tem como base os serviços do *Security Token Service* (STS), definido na especificação *WS-Trust* (Seção 3.4.2.3), e ainda do Provedor de Identidade (*Identity Provider* - IdP) e Serviço de Atributos e Pseudônimos (*Attribute and Pseudonym Services* - APS), ambos propostos na especificação *WS-Federation* (Seção 3.4.2.4).

Cada domínio possui um STS, um IdP e um Serviço de Atributos e Pseudônimos (Figura

5.4). O STS é responsável por emitir, validar e trocar credenciais, já o IdP é um tipo especial de STS que visa declarar a identidade do principal em uma credencial. Nesta proposta, ambos são combinados em única entidade, chamada STS/IdP. Tanto a privacidade do usuário quanto o fornecimento de atributos são funções do Serviço de Atributos e Pseudônimos que age em conjunto com o STS/IdP. O STS/IdP e Serviço de Atributos e Pseudônimos agem, respectivamente, conforme a autoridade de autenticação e de atributos definidas no modelo inicial, reunindo vários clientes e provedores de serviços através de seus atributos de segurança (credenciais, certificados, etc).

A segurança nas trocas de mensagens na arquitetura proposta é garantida através da especificação WS-Security e obedece aos requisitos da política da qualidade de proteção definida na interface dos serviços. Para tanto, no próximo capítulo, apresenta-se o protótipo implementado que visa, além de reunir as soluções apresentadas para o modelo, agregar segurança nas trocas de mensagens de forma transparente às partes envolvidas.

Na seqüência, os componentes ilustrados na Figura 5.4 são discutidos em detalhes.

5.2.5.1 Padrões no Modelo para a Interoperação

Quando as relações de confiança acontecem em um único domínio, não há maiores implicações para a transposição de autenticação e de atributos, uma vez que, uma mesma tecnologia de segurança é usada no domínio. Basta o STS/IdP emitir algum atributo de segurança em nome da entidade que deseja acesso a um serviço de um provedor e este, avaliando se os atributos foram emitidos pelo STS que confia, concede acesso ao recurso. As especificações *WS-Trust* e *WS-Federation* ilustram vários destes casos. As dificuldades e barreiras estão presentes quando a requisição envolve a transposição entre domínios distintos. No modelo proposto, o STS/IdP, através de asserções SAML, assume então o papel de mediador das relações de confiança envolvendo diferentes domínios de segurança.

Como a especificação *WS-Federation* parte do princípio que todos os atributos do usuário se encontram centralizados em um domínio, uma extensão ao Serviço de Atributos e Pseudônimos é proposta para permitir que o Serviço de Atributos e Pseudônimos busque os atributos do usuário nos domínios em que este é filiado, ou seja, nos domínios que aceitam a identidade proveniente daquele domínio (identidade federada), conforme Figura 5.4. Desta forma, ao receber uma requisição por atributos, e não possuindo os atributos solicitados, o Serviço de Atributos e Pseudônimos os busca nos domínios em que a identidade do cliente está federada.

As características do STS/IdP, conforme a definição do domínio, dependem da tecnologia de segurança subjacente. Por exemplo, se este estiver em um domínio SPKI, o mesmo tem a capacidade de reconhecer autenticações baseadas em uma assinatura SPKI. Se, ao contrário, estiver num domínio no qual a tecnologia presente se baseia na emissão de *tickets* Kerberos, então tem a habilidade de entender as necessidades de tal infra-estrutura.

Para atingir a interoperabilidade entre domínios com tecnologias de segurança distintas e transpor as informações de autenticação e de atributos, faz-se neste trabalho largo uso de asserções SAML. O STS/IdP de um domínio emite estas asserções que podem transportar informações de autenticação e ainda atributos dos principais. Por exemplo, pode-se expressar em uma asserção que um principal "A" foi autenticado em um determinado período (via assinatura digital com uma chave RSA). Seus atributos como, *e-mails*, endereço, etc, são também transportados via asserções. Outra característica das asserções SAML é a capacidade de manter a privacidade do usuário por meio de pseudônimos. Neste caso, o provedor de serviços, ao receber uma asserção com o pseudônimo do principal, poderia solicitar ao Serviço de Atributos e Pseudônimos do STS/IdP de seu domínio que procure mais informações sobre o principal, sem violar a política de privacidade definida pelo usuário.

Apesar das asserções SAML serem flexíveis, permitindo carregar diferentes declarações (autorização, autenticação ou atributos), assume-se neste trabalho que ao se autenticar no STS/IdP do seu domínio o cliente recebe uma asserção contendo uma declaração de autenticação, que pode ser usada em qualquer requisição de serviço em domínios federados. Já nas interações entre domínios (entre STS/IdP de diferentes domínios), os mesmos se utilizam somente de asserções de atributos. Os atributos solicitados obedecem a uma sintaxe, definida para os domínios pertencentes à relação de confiança.

5.2.5.2 Uso de Protocolos Padrões na Concretização dos Controles de Autorização

O modelo proposto faz uso da especificação XACML para o controle de acesso local. Esta especificação também se apóia nos mecanismos PEP (*Policy Enforcement Point*) e o PDP (*Policy Decision Point*) da IETF (Seção 5.2.3), mas além disso define outras entidades, como o PAP (*Policy Access Point*) e o PIP (*Policy Information Point*). O PAP contém as políticas do sistema e o PIP informações sobre o sujeito, o ambiente e o recurso (e protocolos para interações entre estas entidades). Segundo a especificação XACML ambos podem ser representados por repositórios ou entidades remotas. É o uso que se faz nos serviços de Atributos e Pseudônimos e de Políticas apresentados na Figura 5.4. A solução proposta é uma extensão ao padrão XACML para suportar o protocolo pedido/resposta introduzido na especificação do *WS-Trust*, no qual o STS/IdP está baseado. Desta forma, quando o PDP necessita realizar requisições por atributos ou validar a credencial apresentada pelo cliente, o faz através do protocolo definido na especicação *WS-Trust*.

A autorização se concretiza por intermédio das autoridades do domínio quando há solicitações de um cliente desconhecido. Desta forma, o provedor não precisa ter conhecimento prévio dos seus potenciais clientes, agregando assim menor custo, pois não é necessário manter um diretório de usuários, e facilitando a utilização do serviço pelo usuário.

5.2.5.3 Serviço Tradutor de Credenciais

Uma das barreiras para transposição de autenticação está principalmente nas diferentes credenciais de segurança, provenientes das distintas infra-estruturas de segurança usadas nos domínios da transposição. Uma questão não clara nas especificações *WS-Trust* (WS-Trust, 2005) e *WS-Federation* (WS-Federation, 2003a) é como se dá a troca de uma credencial de uma infra-estrutura de segurança por uma outra credencial, em um formato particular. A *WS-Trust* coloca apenas a necessidade de troca de credenciais e na *WS-Federation* tem a citação "que o provedor de serviço, usa o STS para entender e validar as credenciais recebidas do cliente". Neste último, mesmo que o STS entenda duas tecnologias de segurança, como X.509 e SPKI, é difícil entender como as informações do cliente podem ser traduzidas de X.509 para SPKI. A especificação não abrange esta questão.

Na definição proposta o domínio, o STS/IdP entende apenas a tecnologia de segurança subjacente de seu domínio e representa informações como credenciais num formato interoperável, ou seja, asserções SAML. Sendo assim, ao receber uma asserção SAML, o STS/IdP terá que trocar a asserção SAML por uma credencial no formato de sua tecnologia de segurança.

Nesta proposta, inspirado nos trabalhos de Grades Computacionais (Foster *et al.*, 2005), a tradução de informações da asserção SAML para as necessidades da tecnologia subjacente é responsabilidade do Serviço Tradutor de Credenciais (STC) (ver Figura 5.4). Este entende a tecnologia do domínio e é capaz de tanto representar as informações da tecnologia em uma asserção SAML, quanto representar uma asserção SAML na informação entendida pela tecnologia do domínio. Desta forma, clientes e provedores de serviço mantém suas características de segurança e o STS, através das asserções SAML, faz a ligação entre ambos. O cliente recebe de seu STS/IdP a asserção SAML que representa suas informações e a apresenta ao serviço, que por sua vez solicita ao seu STS/IdP que represente a informação da asserção conforme sua tecnologia de segurança. Neste instante, o STS/IdP solicita a conversão da credencial ao Serviço Tradutor de Credenciais, que deverá retornar a credencial solicitada.

Neste trabalho, considera-se que o Serviço Tradutor de Credenciais apenas entende asserções SAML, certificados X.509 e SPKI. Vale lembrar que os provedores de serviço podem exigir que a credencial seja apresentada no formato da sua infra-estrutura de segurança.

A filosofia SPKI/SDSI permite apenas que o principal detentor da chave privada emita certificados (ver Seção 2.6.3). O proprietário de um recurso pode delegar acesso ao recurso a um outro principal. Sendo assim, quando em um domínio SPKI, o STS/IdP (representando o provedor) delega ao detentor da asserção SAML o direito de acesso ao recurso, apresentando as informações do cliente em *S-expression*, formato dos certificados SPKI/SDSI.

Para um domínio X.509, o STS/IdP, ao receber a asserção SAML, a repassa ao Serviço Tradutor. Este então retira da asserção as informações do cliente como, por exemplo, a sua

chave pública, e as devolve ao STS/IdP. O STS/IdP então assina a chave pública do cliente e repassa o certificado gerado ao provedor de serviços.

5.2.5.4 Solicitação de atributos

Tanto o padrão SAML, quanto a especificação *WS-Trust*, disponibilizam meios para a troca de atributos entre duas entidades. Vale lembrar que são atributos relacionados a lógica do negócio. A solução aqui proposta se baseia na *WS-Trust* para a troca de atributos. Quando o pedido parte do provedor de serviços, este o faz via `RequestSecurityToken` contendo os elementos (`Claims`) solicitados e a resposta é apresentada via `RequestSecurityTokenResponse`, com os respectivos atributos. Porém, quando o pedido por atributos parte do STS/IdP do provedor de serviços para o STS/IdP do cliente, a resposta a estes pedidos vem em uma asserção SAML de atributos.

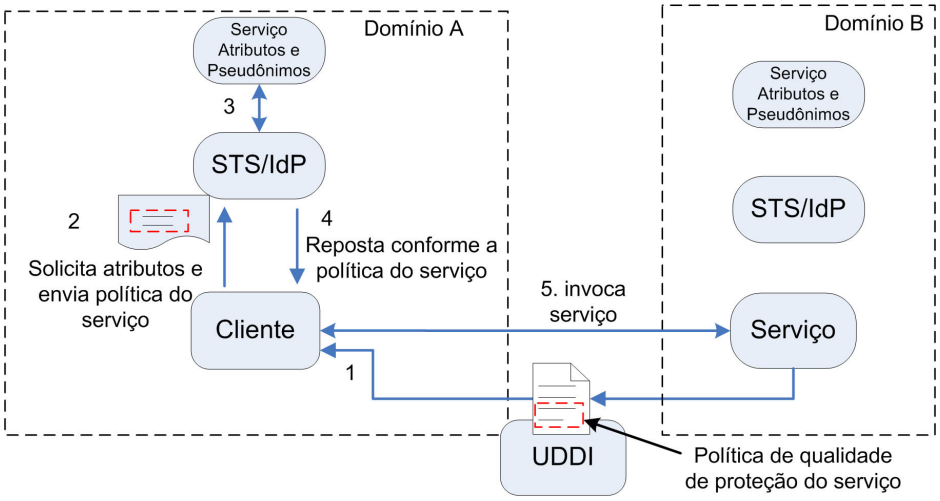

Figura 5.5: Cliente solicitando ao STS atributos

Duas abordagens são possíveis em relação a obtenção de atributos. Tanto cliente quanto serviço podem ser os responsáveis pelas requisições. Na primeira abordagem (ver Figura 5.5) uma vez de posse da política de proteção (passo 1), anexa na interface (WSDL) do serviço, o cliente pode verificar quais atributos são necessários para o acesso e solicitar que o seu STS/IdP (passo 2) obtenha tais atributos no seu Serviço de Atributos e Pseudônimos (passo 3). Após isso, o cliente recebe a resposta do seu STS/IdP (passo 4) e realiza a invocação ao serviço com os atributos descritos na WSDL do serviço (passo 5).

Por outro lado, o provedor de serviço pode, em outra abordagem, ir atrás dos atributos do cliente, tornando o acesso mais fácil e transparente ao cliente, conforme Figura 5.6. Esta é a abordagem adotada dentro do conceito de identidades federadas. Sendo assim, o cliente não fornece diretamente seus atributos ao provedor de serviço. Autentica-se normalmente em seu STS/IdP (passo 2), obtém um pseudônimo (passo 3) e realiza a invocação ao serviço (passo 5). Agora o serviço recorre a seu STS/IdP (passo 6) para recuperar os atributos do

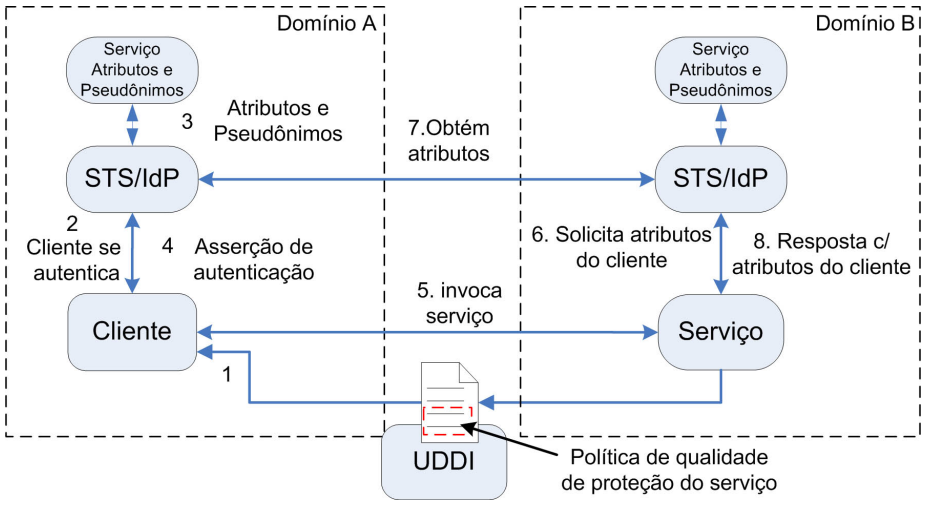

Figura 5.6: Serviço solicita atributos do cliente

cliente (passo 7). No presente trabalho, por empregar o conceito de identidades federadas, adota-se preferencialmente esta segunda abordagem na recuperação de atributos, porém não se descarta a possibilidade de solicitação de atributos por parte dos clientes.

Quando domínios administrativos distintos estão envolvidos, é necessário garantir que atributos emitidos em um domínio são válidos pela política de autorização em outro domínio. O projeto *Shibboleth* (Seção 4.4.2) define um conjunto padrão de atributos e também faz uso de SAML, visando garantir a interoperabilidade dentro de um ambiente federado. Nesse sentido, a solicitação de atributos requer que as partes envolvidas concordem em uma sintaxe e um protocolo, que devem ser comuns aos domínios para permitir o compartilhamento das informações e decisões sobre controle de acesso de forma interoperável. Diante disso, a solução proposta neste trabalho adota a idéia de atributos padrões.

Ainda que os domínios concordem em uma sintaxe, é o cliente que define qual dos seus atributos são disponíveis. O projeto maior desenvolvido por nosso grupo, vislumbra um serviço de filiação, que armazena diversas informações sobre o usuário, como seu certificado, nome, etc. Além disso, permite que o cliente especifique sua política de privacidade e desta forma escolha quais atributos estarão disponíveis.

Nesta dissertação, define-se uma sintaxe básica e extensível para permitir o compartilhamento dos atributos entre domínios através das asserções SAML. Como dito acima, não apenas o provedor de serviço pode requerer atributos do cliente, mas também a entidade STS/IdP pode precisar recuperar informações do cliente. Neste último caso, considera-se uma asserção SAML de autenticação fornecida pelo cliente como não sendo suficiente para montar a nova credencial e o STS/IdP então solicita os atributos necessários.

A Figura 5.7 exibe o esquema XML para a sintaxe. Não é intenção deste trabalho descrever todos os atributos possíveis, pois estes fazem parte da lógica da aplicação, mas sim os essenciais para a montagem dos certificados X.509 e para a delegação de certificados SPKI

pelo STS/IdP. Nesse sentido, para a especificação da sintaxe foi considerado, principalmente, a informação que o STS/IdP precisa obter do cliente para traduzir a asserção SAML na nova credencial X.509 ou SPKI.

```
1  <?xml version="1.0"?>
2  <xs:schema xmlns:xs="..." targetNamespace="..." >
3  <!-- Atributos conforme a lógica da aplicação-->
4  <xs:element name="name" type="xs:string"/>
5  ...
6  <!-- Atributos para geração de certificados-->
7  <element name="SubjectName" type="tr:DNType"/> <!-- Nome do sujeito -->
8  <complexType name="DNType">
9     <sequence>
10       <element name="commonName" type="string"/>
11       <element name="organizationUnit" type="string"/>
12       <element name="organizationName" type="string"/>
13       <element name=" localityname" type="string"/>
14       <element name="stateorprovincename" type="string"/>
15       <element name="lettercountry" type="string"/>
16    </sequence>
17 </complexType>
18 <element name="publickey" type="base64Binary"/> <!-- chave pública do sujeito -->
19 </xs:schema>
```

Figura 5.7: Esquema para solicitação de atributos

A seção 2.6 exibe os certificados X.509 e SPKI com os campos que são referentes ao sujeito. Quando o STS/IdP necessita traduzir uma asserção SAML para um certificado X.509 ou SPKI este considera principalmente a chave pública do sujeito, ou seja, o atributo da linha 18 (`publicKey`). As chaves públicas, inicialmente, não estão amarradas a um certificado, possibilitando seu uso tanto em novos certificados X.509 quanto nos SPKI. A responsabilidade por traduzir a asserção é do Serviço Tradutor de Credenciais. O atributo `SubjectName`, linha 7, (Figura 5.7) é utilizado para que o STS/IdP mantenha informações sobre o dententor da chave.

Como dito anteriormente, as solicitações de atributos entre os domínios são realizadas conforme o protocolo pedido/resposta da especificação *WS-Trust*. Por exemplo, a mensagem SOAP (ver Figura 5.8) é enviada por um STS/IdP que solicita uma asserção SAML contendo atributos referente ao sujeito da asserção SAML de autenticação (linhas 9 a 12 da Figura 5.8). Para comprovar que o cliente realmente está fazendo solicitação ao recurso, o STS/IdP anexa à asserção de autenticação (linha 14), apresentada pelo cliente ao serviço, no pedido pelos atributos do cliente.

A Figura 5.9 exibe a resposta ao pedido da Figura 5.9. Os atributos solicitados estão contidos na asserção SAML, conforme as linhas 14 e 17.

```
<soapenv:Envelope xmlns:soapenv="..." xmlns:wst="..." xmlns:att="..." >
   <soapenv:Header>
      ...
   </soapenv:Header>
   <soapenv:Body >
     <wst:RequestSecurityToken Context= "...">
       <wst:TokenType> SAML </TokenType>
       <wst:RequestType> emissão </RequestType>
       <wst:Claims wst:Dialect="...">
           <att:Claim URI=".../claims/name"/>
           <att:Claim URI=".../claims/organization"/>
       </wst:Claims>
       <Base>
           <!-- Asserção SAML de autenticação apresentada pelo cliente >
       </Base>
     <wst:RequestSecurityToken>
  </soapenv:Body>
</soapenv:Envelope>
```

Figura 5.8: Pedido por uma asserção SAML contendo determinados atributos

```
<soapenv:Envelope xmlns:soapenv="..." xmlns:wst="..." xmlns:saml="...">
   <soapenv:Header>
      ...
   </soapenv:Header>
   <soapenv:Body >
     <wst:RequestSecurityTokenResponse>
       <wst:TokenType>SAML</TokenType>
       <wst:RequestedSecurityToken>
          <saml:Assertion AssertionID="..."
           ...
           <saml:AttributeStatement>
              <saml:Subject> Cliente </saml:Subject>
              <saml:Attribute AttributeName="name">
                   <AttributeValue> Cliente Dominio A </AttributeValue>
              </saml:Attribute>
              <saml:Attribute AttributeName="organization">
                   <AttributeValue> Organização A </AttributeValue>
              </saml:Attribute>
           </saml:AttributeStatement>
              ...
          </saml:Assertion>
       </wst:RequestedSecurityToken>
   </wst:RequestSecurityTokenResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 5.9: Resposta contendo os atributos solicitados

5.2.5.5 Contrato de Confiança

No modelo proposto nesta dissertação, a confiança entre as entidades STS/IdP já está estabelecida. Tal estabelecimento se dá através de um *contrato*, fornecido para cada domínio, indicando ali os direitos e deveres que cada autoridade possui, expressos através de asserções SAML de atributos. Para expressar o *contrato*, foi desenvolvido um esquema XML que teve como base o trabalho de Maruyama *et al.* (2003).

A Figura 5.10 ilustra um *contrato* que estabelece a confiança entre duas partes. No *contrato*, podem estar contidos atributos (linhas 5 a 11) que indicam o tipo da relação de confiança (linhas 7 a 9), o papel a ser concedido ao outro STS/IdP, entre outros. Para o contrato ser considerado válido é necessário que ambos STS/IdP, envolvidos no processo, assinem-o (linhas 17–18). Uma vez munido das assinaturas, o contrato deverá ser entregue para ambos STS/IdPs, garantindo a estes os direitos ali expressos, direitos os quais não poderão ser repudiados.

O estabelecimento dinâmico da confiança entre as autoridades em domínios distintos, abordado em (de Mello, 2005), faz uso deste contrato para firmar a confiança.

```
1  <tr:contract xmlns:tr="..." xmlns:wsu="..." xmlns:saml="..."
2     xmlns:ds="..." xmlns:wsse="..." ID="..." >
3     <tr:partOne>
4     <saml:Assertion Issuer="...">
5        <saml:AttributeStatment>
6           <saml:subject>...</saml:subject>
7           <saml:Attribute AttributeName="trustlevel">
8              <saml:AttributeValue>strong</saml:AttributeValue>
9           </saml:Attribute>
10          ...
11       </saml:AttributeStatment>
12    </saml:Assertion>
13    </tr:partOne>
14    <tr:partTwo>
15       <!-- assertions to partOne -->
16    </tr:partTwo>
17    <ds:Signature><!-- Signed by partOne --></ds:Signature>
18    <ds:Signature><!-- Signed by partTwo --></ds:Signature>
19 </tr:contract>
```

Figura 5.10: Contrato de Confiança

5.2.5.6 Política de Qualidade de Proteção

A política que define os requisitos de segurança do serviço são expressas conforme define a especificação *WS-SecurityPolicy* (WS-SecurityPolicy, 2003) e é anexada a WSDL através da *WS-PolicyAttachment* (WS-PolicyAttachment, 2003) (ambas apresentadas na Seção 3.4.2.2). Ao obter a WSDL do serviço em um UDDI, ou de alguma outra forma, o cliente deve adequar suas mensagens SOAP à política de qualidade de proteção anexada a WSDL.

Na descrição da política de qualidade de proteção, o provedor pode especificar, além do tipo de credencial para acesso a seus serviço, um STS/IdP específico no qual o cliente deve efetuar a solicitação pela credencial. Caso o cliente seja incapaz de emitir a credencial solicitada ou não conheça o STS/IdP indicado pelo provedor de serviços, este então recorre ao seu STS/IdP para que o mesmo obtenha tais credenciais. No entanto, como neste trabalho é o provedor de serviços que vai atrás das informações do cliente, então o STS/IdP do cliente emite uma asserção de autenticação ao cliente, segundo as proposições deste trabalho.

Os serviços, fornecidos pelo provedor, apenas aceitam mensagens que obedeçam a sua política de qualidade de proteção. Cada pedido recebido é verificado para então ser repassado ao PEP (Figura 5.4). Na Figura 5.11, exibe-se um exemplo de uma política de qualidade de proteção de um serviço. Tal política expressa que são aceitas apenas credenciais de segurança do tipo X.509, emitidas por determinado STS/IdP (que agiria como um autoridade certificadora).

```
1  <wsp:Policy xml:base="http://stsidp.ufsc.br/policies" wsu:Id="TOK">
2     <sp:IssuedToken>
3        <sp:Issuer>http://sts.das.ufsc.br</sp:Issuer>
4        <sp:RequestSecurityTokenTemplate TrustVersion="1.1">
5           <wst:TokenType="X509Token">
6                   <wst:Claims wst:Dialect="...">
7                    <!-- Atributos definidos pelo serviço, conforme a gramática -->
8                   </wst:Claims>
9        </sp:RequestSecurityTokenTemplate>
10    </sp:IssuedToken>
11 </wsp:Policy>
```

Figura 5.11: Exemplo de política de qualidade de proteção do serviço

A linha 6 também define que o serviço obedece a uma sintaxe para a troca de atributos. O cliente não precisa entender tal política. Sua função é enviar o conteúdo das linhas 4 a 9 à seu STS/IdP para que esta emita a asserção de autenticação necessária para seu acesso.

5.2.6 Dinâmica do Modelo Proposto

A Figura 5.4 é apresentada novamente nesta seção, como Figura 5.12, para ilustrar a dinâmica do modelo proposto. Partindo das definições apresentadas, o cliente primeiro obtém uma asserção de autenticação (passo 1) e depois a envia ao provedor de serviço, junto com a sua requisição de serviço, conforme passo 2. Antes da concretização da requisição, o processo de autorização é efetivado. O PEP apresenta, através de um pedido no formato XACML, os dados do requisitante ao PDP, ou seja, a asserção de autenticação (passo 3, Figura 5.12).

Neste momento, o PDP precisa decidir se autoriza ou não acesso para o cliente. Como o cliente é desconhecido do provedor, então o PDP solicita informações como, por exemplo,

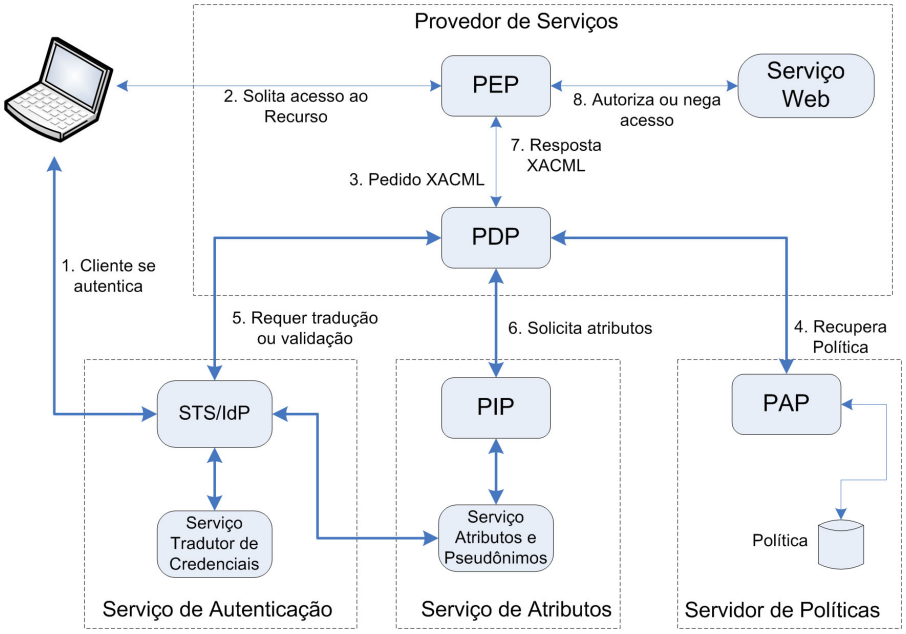

Figura 5.12: Representação do modelo em Serviços *Web*

atributos, a validação da credencial, ou mesmo a tradução da credencial, a entidades externas para então poder avaliar o pedido. Antes, porém, o PDP consulta a política de acesso associada ao serviço. A política do acesso é obtida através de interações com o PAP (passo 4).

De posse da política relacionada ao serviço, o PDP agora solicita a tradução (de SAML para X.509, por exemplo), e conseqüente validação da credencial, uma vez que o cliente apresentou uma asserção SAML ao invés da credencial exigida (X.509 ou SPKI, por exemplo) e é desconhecido pelo provedor (passo 5).

Como o provedor não detém um registro do cliente, faz-se então necessário entrar em contato com o domínio de origem do cliente, onde estão as suas informações. Desta forma, conforme o passo 6, o PDP requer atributos do cliente, segundo a política do serviço, para concretizar o acesso. Tais atributos podem ser, por exemplo, o endereço de *email* do cliente ou o CEP.

As informações obtidas pelo PDP definem o contexto de decisão que este necessita. A validação ou a tradução da credencial e a obtenção de atributos do cliente definem as interações entre o STS/IdP e o Serviço de Atributos e Pseudônimos, conforme mostra a Figura 5.12. Estas solicitações, por exemplo, podem assegurar ao provedor que o pseudônimo apresentado pelo cliente realmente foi emitido pelo seu domínio de origem.

Uma vez verificada a autorização, o PDP devolve ao PEP uma asserção de autorização, declarando os direitos de acesso do cliente, conforme passo 7. O PEP então aplica a decisão do PDP, negando ou autorizando o acesso, de acordo com o passo 8.

As interações entre PDP e as demais entidades externas ao provedor de serviços são

realizadas via protocolos definidos na especificação *WS-Trust*. Interessante perceber que o STS/IdP assume o papel do PIP em relação aos atributos do cliente. Vale observar também que, no passo 5, o PDP poderia apenas solicitar a validação da credencial, caso a credencial anexa ao pedido esteja de acordo com a sua infra-estrutura. A validação tem por objetivo confirmar se há uma relação de confiança entre o STS/IdP do cliente e o STS/IdP do provedor de serviços.

```
1  <soapenv:Envelope xmlns:soapenv="..." xmlns:wst="..." xmlns:att="..." xmlns:wsse="..." >
2     <soapenv:Header>
3        ...
4     </soapenv:Header>
5     <soapenv:Body >
6       <wst:RequestSecurityToken Context= "...">
7          <wst:TokenType> X.509 </TokenType>
8          <wst:RequestType> troca </RequestType>
9          <wst:OnBehalfOf>
10              <wsse:Security>
11                 <saml:Assercion AssertionID="...">
12                 ...
13                 </saml:Assercion>
14           </wsse:Security>
15         </wst:OnBehalfOf>
16         <wst:Claims wst:Dialect="...">
17            <att:Claim URI=".../claims/name"/>
18            <att:Claim URI=".../claims/organization"/>
19         </wst:Claims>
20      <wst:RequestSecurityToken>
21    </soapenv:Body>
22 </soapenv:Envelope>
```

Figura 5.13: Pedido de Troca de Credencial e por Atributos

Os passos 5 e 6 da Figura 5.12 exigem que o PDP formule um pedido ao seu STS/IdP requerendo a troca da credencial (ou a sua validação) e atributos do cliente, respectivamente. Desta forma, a Figura 5.13 ilustra um pedido XML, emitido pelo provedor de serviço, solicitando a troca de uma asserção SAML por uma credencial X.509, conforme a linha 7, e por atributos, linhas 17 e 18. Como resposta, o PDP recebe a credencial em nome do cliente no formato entendido pelo provedor de serviço (X.509, por exemplo) e os atributos "nome" e "organização".

A solicitação de atributos é baseada na definição da sintaxe apresentada na Figura 5.7 (Seção 5.2.5.4). Provedores de serviços solicitam ao STS/IdP atributos do cliente e esta, não conhecendo o cliente, pode transferir este pedido ao Serviço de Atributos e Pseudônimos do domínio de origem do cliente. Além dos atributos solicitados pelo provedor, o STS/IdP do serviço também pode solicitar ao Serviço de Atributos do cliente, os atributos necessários para transformar a asserção SAML de autenticação em uma credencial entendível pelo serviço.

5.2.6.1 Dinâmica no Modelo da Transposição

A fim de ilustrar a dinâmica do processo de transposição na solução proposta, uma requisição de serviço, envolvendo domínios distintos, é exposta a seguir, tomando como base a ilustração da Figura 5.14. Cliente e provedor de serviço não se conhecem, porém há um contrato de confiança entre suas entidades STS/IdP. Cada domínio possui sua própria infra-estrutura de segurança. O domínio do cliente tem como tecnologia de base o SPKI/SDSI e o do provedor usa a infra-estrutura X.509.

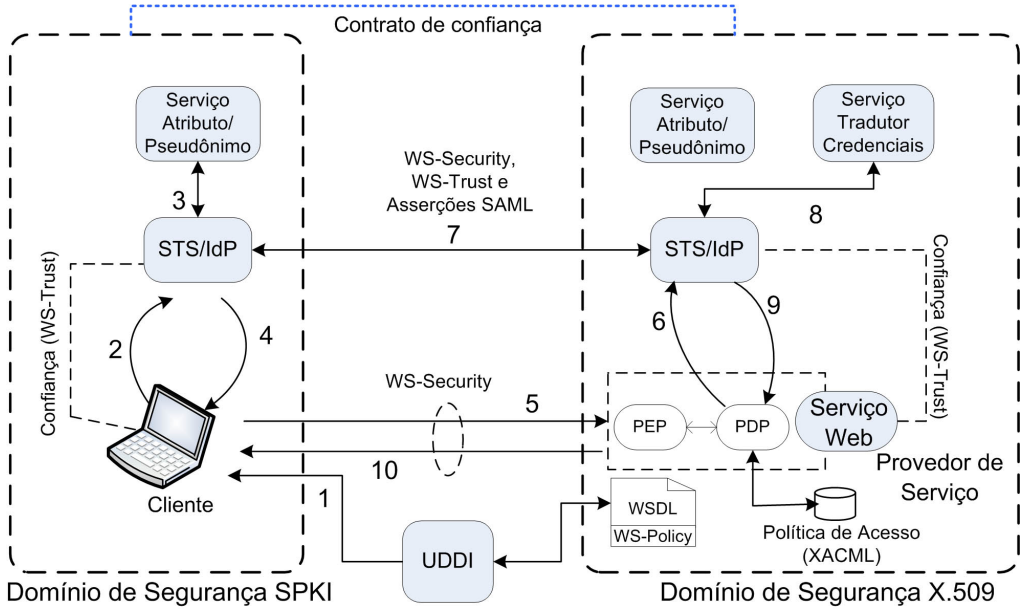

Figura 5.14: Transposição no Modelo

No passo 1, o cliente obtém acesso a WSDL do serviço. O cliente então lê a política de qualidade de proteção, anexa na WSDL do serviço, a qual requer credenciais no formato X.509 e emitidas pelo STS/IdP do provedor serviço. Além disto, também explicita o algoritmo e tamanho da chave para assinar e cifrar as mensagens. Como o cliente não entende o X.509, nem conhece o STS/IdP descrito na política de qualidade de proteção, este então solicita ao seu STS/IdP que faça a mediação na sua requisição.

No passo 2, o cliente efetua a autenticação para que o STS/IdP o reconheça como seu filiado. Como o provedor de serviços é a entidade ativa (abordagem do serviço solicitando atributos, Seção 5.2.5.4), o cliente apenas recebe uma asserção de autenticação declarando a sua identidade. No passo 3, o Serviço de Atributos e Pseudônimos (SAP) adquire um pseudônimo para o cliente. O pseudônimo é armazenado temporariamente no SAP para possíveis solicitações de atributos referentes ao pseudônimo do cliente.

Após receber a asserção de autenticação no passo 4, o cliente apresenta a mesma ao serviço, conforme o passo 5, juntamente com o pedido de acesso ao serviço. A requisição de serviço montada deverá atender requisitos da política de qualidade de proteção como assinar a mensagem e cifrá-la segundo algoritmos definidos. Ao receber a asserção, o provedor

de serviço verifica se a mesma contém os requisitos descritos em sua política de qualidade de proteção e, em caso afirmativo, a repassa ao seu PEP que então requer a decisão do PDP. Durante o processo de autorização, como descrito na seção anterior, através do PDP, o serviço realiza solicitações ao seu STS/IdP para que este traduza a asserção SAML no formato X.509 e também por atributos necessários para acesso ao serviço (segundo a política de acesso) (passo 6). O STS/IdP do provedor, tendo uma relação de confiança com o STS/IdP que emitiu a asserção, solicita os atributos ao STS/IdP do cliente (passo 7, Figura 5.14). A autoridade do cliente então recupera os atributos do Serviço de Atributos e Pseudônimos do domínio e os repassa ao provedor de serviço. Após receber os atributos (asserção SAML de atributos), o STS/IdP do domínio do Provedor pede ao seu Serviço Tradutor de Credenciais que monte um certificado X.509 com as informações do cliente para que o serviço possa conceder acesso ao recurso (passo 8). De posse da credencial X.509 e possíveis atributos o provedor de serviço, através de seu PDP pode tomar a decisão e a concretização do acesso ao serviço (passo 10, na Figura 5.14).

Vale lembrar que, no passo 2, se o cliente fosse uma entidade ativa, enviaria a política de qualidade de proteção do serviço para o seu STS/IdP conseguir os requisitos necessários.

5.3 Considerações sobre o Modelo Prosposto

Após reunir e analisar um conjunto considerável de especificações de segurança em Serviço *Web*, a solução proposta obtém uma forma interoperável de transferir autenticação e atributos entre domínios com infra-estruturas de segurança distintas e assim obter uma autorização descentralizada. O conceito de identidades federadas, no qual o trabalho é constituído, favorece uma eficaz e independente transposição da informação.

A definição do modelo sobre proposições já consolidadas favorece um melhor entendimento da dinâmica da transposição em sistemas distribuídos. Auxilia também a visualizar como a arquitetura orientada a serviços e o gerenciamento de identidades federadas definem meios para a integração entre os domínios. Integração esta que, além das barreiras relacionadas às dificuldades administrativas, como gerencia dos usuários, gerenciamento das políticas, etc, é dificultada principalmente devido a diferentes infra-estruturas de segurança.

No modelo, as credenciais de segurança são essenciais para provar a identidade dos usuários e os habilitar a utilizar os recursos espalhados em diferentes domínios. A asserção SAML traz uma maneira interoperável e eficaz de transportar esta informação. No entando, quando há a comunicação entre diferentes domínios e são apresentadas credenciais as quais um outro domínio não entende a comunicação entre estes domínios se torna mais custosa. A tarefa de traduzir uma credencial em outro formato não é uma tarefa simples.

As dificuldades em converter diferentes tipos de credenciais são evidentes. Para contornar tais dificuldades, deve-se contar com relações de confiança pré-estabelecidas entre

domínios distintos para que informações que entram em um domínio através destas relações de confiança sejam consideradas "confiável".

Os mecanismos de delegação dos certificados SPKI/SDSI soam como uma boa solução dentro de domínios SPKI. Para domínios X.509, o STS/IdP se torna um tipo de certificadora *online* e emite novos certificados com base na credencial recebida. Uma questão chave para tornar isso possível é que a confiança estabelecida entre o STS/IdP e seus membros deve ser tal a ponto dos membros confiarem na chave privada do STS/IdP. Para que a revogação dos certificados não seja um problema difícil de gerenciar, a validade dos certificados emitidos deve observar um curto período de vida. Um outro possível uso do Serviço Tradutor de Credencial é na conversão de uma autenticação baseada em usuário e senha num formato específico de credencial, aumentando a flexibilidade deste serviço.

A troca de atributos entre os serviços STS/IdP dos domínios é fundamental para o processo de autorização. Uma vez que o provedor de serviço desconhece o cliente é preciso formas de buscar as informações necessárias para o processo de autorização. O protocolo da especificação *WS-Trust*, aliado às asserções SAML, concretizam a troca de atributos entre os serviços STS/IdPs e, desta forma, uma autorização distribuída e descentralizada.

A união dos diversos padrões presentes na literatura de Serviços *Web*, como os protocolos SAML e XACML, permite ainda uma gama de possibilidades em relação à autenticação e autorização. Prova disto são as muitas e diferentes utilizações de tais padrões na literatura atual, como é visto na Seção 6.5.

Retirar do cliente a necessidade deste buscar todas as informações necessárias para obter acesso ao serviço aumenta a transparência para o cliente. Já a responsabilidade imposta ao provedor de serviço de validar e buscar informações dos usuários, apesar de lhe permitir alavancar as relações de negócios com potenciais clientes e diminuir a quantidade de informação gerenciada pelo provedor, pode gerar uma grande quantidade de troca de mensagens do lado deste provedor. Para diminuir possíveis ataques de negação de serviço ou mesmo uma baixa escalabilidade a RFC 2753 (Yavatkar *et al.*, 2000), que originalmente define o PEP e o PDP, prevê que as entidades podem estar em lugares distintos e serem tolerante a faltas.

5.4 Conclusão

Este capítulo exibiu um modelo genérico para permitir que com base na autenticação realizada no seu domínio de origem o principal acesse recursos espalhados nos diferentes domínios pertencentes a suas relações de confiança, de forma transparente e interoperável. Desta forma, o modelo, apoiado em Serviços *Web*, se concentrou em lidar com diferentes tecnologias de segurança para permitir a transposição de autenticação e de atributos.

Os Serviços *Web* se apresentaram com uma boa solução para a concretização da transposição no modelo proposto. As principais barreiras encontradas nesta concretização do modelo foram devido a grande quantidade de especificações, algumas ainda não totalmente consolidadas. Outra dificuldade é a falta de clareza nas especificações em lidar com diferentes infra-estruturas de segurança.

Capítulo 6

Implementação e Resultados

O capítulo anterior apresentou o modelo para transposição de autenticação através de identidades federadas em Serviços *Web*. De forma a verificar a aplicabilidade do modelo, este capítulo descreve a implementação de um protótipo, bem como as ferramentas utilizadas.

Como visto nos capítulos anteriores, as soluções propostas pelas especificações de segurança agregam complexidade no desenvolvimento e adoção de Serviços *Web* seguros. Desta forma, a construção do protótipo visa não apenas agregar segurança nas trocas de mensagens entre os principais envolvidos (clientes, provedores de serviços e STS/IdP), mas também amenizar a complexidade, inerente às especificações de segurança, de clientes e provedores de serviços. Nesse sentido, um dos requisitos observados é separar ao máximo possível as questões relacionadas à segurança da camada de negócios.

Para concretizar o modelo proposto, faz-se uso dos interceptadores (*handlers* do Axis), os quais são responsáveis por capturar a mensagem SOAP e aplicar algum tipo de processamento ou modificação na mesma. Neste trabalho, usa-se o termo "manipuladores" para se referir aos interceptadores implementados para cumprir com as exigências do modelo proposto. Os manipuladores implementados, além de atender as proposições do modelo, agregam as propriedades básicas de segurança, tais como integridade, confidencialidade e autenticidade. Também lidam com questões relacionadas à política de qualidade de proteção e realizam a comunicação com o STS/IdP a fim de permitir o estabelecimento dinâmico da confiança entre clientes e provedores de serviços, através da emissão e validação de credenciais e atributos.

Segundo a Seção 5.2.5.4, duas abordagens são possíveis em relação à solicitação de atributos. Na primeira, o cliente é o responsável pelas solicitações de atributos, ou seja, o cliente é dito "ativo". Na segunda, o serviço é que realiza solicitações por atributos, ou seja, o serviço é a entidade "ativa". No decorrer deste trabalho de dissertação, implementou-se um protótipo com a abordagem do cliente ativo. A aplicação definida em (de Melo E.R. *et al.*, 2004) foi adaptada conforme a abordagem do cliente ativo. Nesta aplicação é possível

tanto consultar quanto obter artigos científicos. A consulta é permita a qualquer cliente, mas a obtenção é autorizada somente a clientes que são de domínios confiáveis pelo provedor de serviço e que apresentam as características descritas na política de qualidade de proteção, como determinados tipos de credenciais e atributos (nome, *email*, endereço).

Este capítulo apresenta um protótipo que faz uso da abordagem na qual o serviço vai atrás das informações do cliente, conforme definição do modelo. Criou-se um serviço simples, que apenas devolve o texto enviado numa solicitação ao serviço, ou seja, um serviço "eco". Este serviço, assim como o adaptado na abordagem do cliente "ativo", apenas responde a solicitação do cliente se este último cumprir com os requisitos impostos pela política de qualidade de proteção.

6.1 Arquitetura do Protótipo

O modelo proposto no Capítulo 5 requer a união de mecanismos para autenticação, autorização, troca de atributos de segurança, etc. Neste sentido, um protótipo que atende às definições do modelo e os requisitos específicos de um sistema de larga escala é apresentado.

A Figura 6.1 ilustra a arquitetura do protótipo. Na camada de "Aplicação" estão localizados os clientes e os provedores de serviços. O módulo "Infra-estrutura para autenticação e autorização", a princípio, está baseado no uso de ICPs, como o SPKI/SDSI e o X.509, e comporta o STS/IdP (*Security Token Service/Identity Provider*), o Serviço de Atributos e de Pseudônimos (SAP) e o Serviço Tradutor de Credenciais (STC). Os manipuladores são responsáveis por capturar, de forma transparente à aplicação, as mensagens SOAP que são enviadas ou recebidas e realizar algum processamento na mesma. Por fim, o módulo "SOAP + HTTP" indica que as mensagens serão enviadas através do SOAP sobre o HTTP.

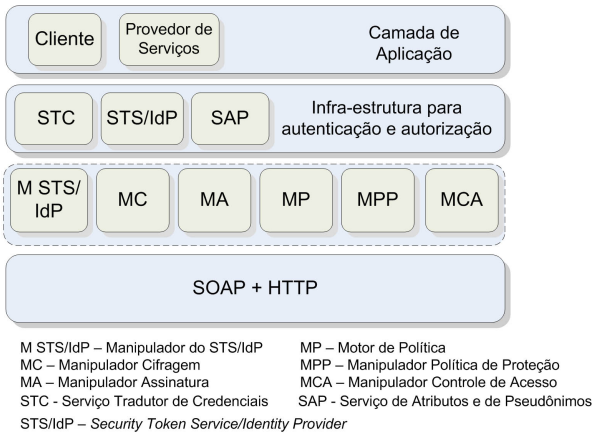

Figura 6.1: Arquitetura do protótipo

Para implementar a arquitetura do protótipo, estendeu-se os recursos presentes em bibliotecas de segurança desenvolvidas para os Serviços *Web* e implementaram-se os mani-

puladores de forma modular e com características próprias para permitir maior flexibilidade ao modelo. A separação entre as camadas de negócio e a de segurança, a conformidade com as especificações existentes, além de ser uma solução de código aberto, são requisitos observados no projeto.

As escolhas tecnológicas que compõe a arquitetura fazem uso de padrões abertos, tais como: o TomCat[1] versão 5.0, o *Axis*[2] versão 1.2.1, a biblioteca WSS4J[3] versão 1.0 e a biblioteca *XML-Security*[4] (que contém o *XML-Signature* e o *XML-Encryption*), todos da Apache Software Fundation. Também foram utilizadas as bibliotecas de código aberto SunXACML[5], OpenSAML [6], JSDSI 2.0 implementada por (Morcos, 1998) e Libsdsi, esta última desenvolvida dentro do projeto cadeias de confiança[7].

O TomCat é um servidor de aplicações Java para Internet. Desenvolvido dentro do projeto *Apache Jakarta* e oficialmente endossado pela Sun como a Implementação de Referência (RI) para as tecnologias *Java Servlet* e *JavaServer Pages* (JSP). O Tomcat tem a capacidade de atuar também como servidor *Web*/HTTP, ou pode funcionar integrado a um servidor *Web* dedicado como o Apache httpd ou o Microsoft IIS.

O Apache *Axis* (Apache, 2005)(*Apache eXtensible Interaction System*) é utilizado em conjunto com o TomCat e, desta forma, herda as funcionalidades deste último, hospedando e gerenciaando os Serviços *Web* de forma transparente.

A ferramenta Axis fornece a implementação do SOAP e é projetado de acordo com o conceito de mensagens encadeadas (Apache, 2005). Mensagens encadeadas são interceptadores (*handlers*) que implementam funções distintas de uma forma muito flexível. Um interceptador Axis é uma classe reutilizável responsável pelo processamento das mensagens SOAP. O motor do Axis invoca uma série de interceptadores para mensagens que seguem por fluxos de pedido e resposta (*request/response)*.

A arquitetura do Axis é formada por três tipo de interceptores, conforme exibe a Figura 6.2, e são relacionados ao transporte, ao serviço ou globais. Tomando como base o lado do serviço, inicialmente a mensagem é capturada no interceptador "transporte", este tem por função recuperar a mensagem SOAP segundo o protocolo utilizado para o transporte como por exemplo, HTTP ou SMTP. Posteriormente, a mensagem segue para o interceptador "global", que processa qualquer mensagem recebida, sem considerar se esta é destinada a um serviço específico. Por fim, a mensagem chega ao interceptador específico ao serviço. Pode existir mais de um interceptador de cada tipo e estes podem ser combinados, formando assim cadeias de interceptadores. A configuração dos interceptadores, tanto para cliente quanto

[1] http://tomcat.apache.org/
[2] http://ws.apache.org/axis/
[3] http://ws.apache.org/wss4j/
[4] http://xml.apache.org/security/
[5] http://sunxacml.sourceforge.net/
[6] http://www.opensaml.org
[7] http://www.das.ufsc.br/seguranca

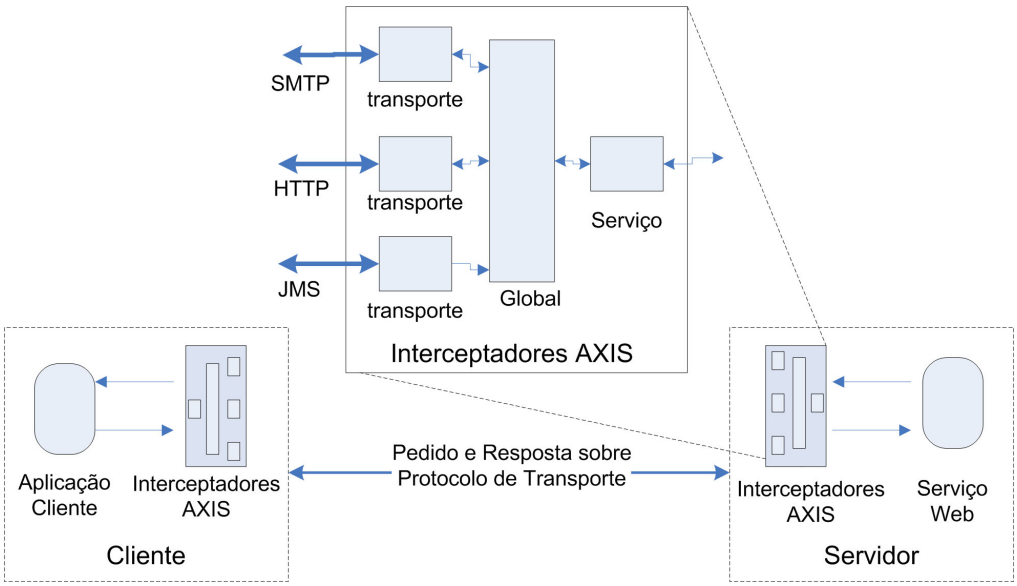

Figura 6.2: Arquitetura do Axis (Apache, 2005)

para serviço, se dá através de um arquivo chamado *Web Service Deployment Descriptor* - (*.`wsdd`). O arquivo permite declarativamente especificar todas as propriedades necessárias para controlar o fluxo de processamento das mensagens SOAP.

A biblioteca WSS4J (*Web Services Security for Java*) é uma implementação da especificação *WS-Security*. Esta faz uso de padrões e implementações abertas, como XML-Security[8] e a OpenSAML[9]. A biblioteca trabalha em conjunto com o Axis, explorando assim os benefícios dos seus interceptadores.

A implementação da WSS4J, vigente durante a implementação do protótipo, conta com um STS (*Security Token Service*, ver Seção 3.4.2.3) que emite apenas credenciais no formato X.509 e *UsernameToken*. A bilioteca também só realiza o processo de assinatura digital conforme a ICP X.509. Este trabalho agregou ao STS da WSS4J o suporte as credenciais no formato SAML, segundo a especificação (OASIS, 2004a). Além disso, adicionou-se à biblioteca a capacidade de assinar as mensagens utilizando os certificados SPKI, através das bibliotecas JSDSI e Libsdsi.

Além dos serviços, como STS/IdP, por exemplo, o protótipo é formado por um conjunto de manipuladores e um módulo chamado Motor de Políticas. Os manipuladores modificam a mensagem SOAP segundo a descrição da política de qualidade de proteção. Já o Motor de Políticas tem por função reunir a política de proteção do serviço.

A implementação dos manipuladores agrega as funcionalidades descritas nas especificações *WS-Security* (OASIS, 2004b), *WS-Trust* (WS-Trust, 2005), XACML (OASIS, 2005a), *WS-Policy* (WS-Policy, 2004) e SAML *Token Profile* (OASIS, 2004a). Por meio da *WS-Security* atinge-se as propriedades de integridade, confidencialidade e autenticidade. Através

[8]Implementação do XML-Encryption e XML-Signature. Disponível em http://xml.apache.org/security/
[9]Implementação aberta da SAML. Disponível em http://www.opensaml.org/

da especificação *WS-Trust* se concretiza a associação de credenciais de segurança às mensagens SOAP e a comunicação com o STS/IdP, que no modelo proposto é responsável pela autenticação dos clientes e pela mediação de confiança. O controle de acesso e a leitura da política de qualidade de proteção da aplicação são atingidos por meio da especificação XACML e *WS-Policy*, respectivamente.

Os manipuladores são acionados pelo arquivo descritor (`WSDD` do Axis). Este contém uma referência que, explicitamente, declara quais os manipuladores que estão ativos na aplicação. No protótipo, o arquivo descritor pode ser modificado pelo manipuladores a fim de acionar os manipulades correspondentes a descrição da política de qualidade de proteção (a segurança nas mensagens) indicada na WSDL. Caso tal política exija o uso de um ou outro manipulador o arquivo `WSDD` é automaticamente alterado para satisfazer a política. Nas seções a seguir, descreve-se as funções dos manipuladores e do motor de políticas.

6.1.1 Manipulador STS/IdP

A função do Manipulador STS/IdP é efetuar as trocas de mensagens entre o cliente e o seu STS/IdP. Desta forma, este é usado na aquisição, troca e validação das credenciais de segurança. Sendo assim, é por meio deste manipulador que se realiza a autenticação no STS/IdP e os pedidos por atributos. Para tanto, o protocolo utilizado é o *RequestSecurityToken* (RST) e *RequestSecurityTokenResponse* (RSTR), da especificação *Ws-Trust* (WS-Trust, 2005).

Todas as informações necessárias para as interações entre o cliente e seu STS/IdP são colhidas via consulta ao manipulador “Motor de Políticas” (MP). São exemplos de tais informações, o mecanismo de autenticação a ser utilizado, a URI do STS/IdP, o tipo de credencial requerida para acesso ao provedor de serviço, etc.

É importante ressaltar que tanto o cliente, quanto o provedor de serviço, possuem uma política de qualidade de proteção relacionada ao seu STS/IdP, pois este último também se trata de um Serviço *Web*. A Figura 6.3 apresenta como o manipulador lida com a política de qualidade de proteção. Inicialmente (passo 1 da Figura 6.3), consulta-se a política relacionada ao seu STS/IdP a fim de obter as informações referentes à autenticação, por exemplo. Em seguinda, no passo 2, a política de qualidade de proteção do serviço é obtida. De posse das duas políticas, o motor de políticas as processa e as anexa ao pedido ao STS/IdP, conforme o passo 3. O cliente não precisa entender a política de qualidade de proteção do serviço (WS-SecurityPolicy, 2003), apenas a envia para que o seu STS/IdP devolva a resposta conforme a política do serviço (passo 4).

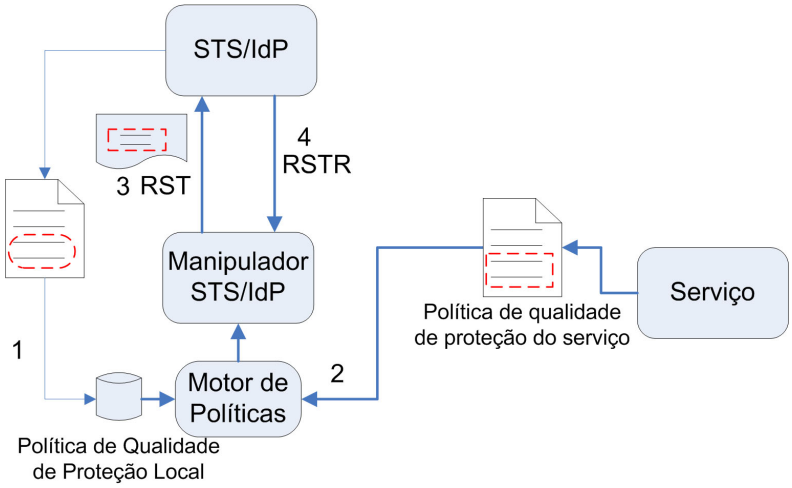

Figura 6.3: Pedido segundo a política de qualidade de proteção do cliente e do serviço

6.1.2 Manipulador de Cifragem e de Assinatura

Estes manipuladores agregam as propriedades básicas de segurança, como confidencialidade, integridade e autenticidade às mensagens. Também fazem consultas ao Motor de Políticas para conhecerem os requisitos impostos na cifragem e na assinatura. Tais requisitos podem ser, por exemplo, o tamanho da chave para cifrar a mensagem e o algoritmo a ser utilizado.

6.1.3 Manipulador de Política de Proteção

Sua função é examinar se a segurança descrita na política de qualidade de proteção é cumprida nas mensagens recebidas como, por exemplo, se a mensagem é ou não assinada pelo cliente. A Figura 6.4 apresenta uma mensagem SOAP que é captura por este manipulador. Partindo do princípio de que há uma asserção SAML de autenticação inclusa na mensagem (ou uma outra credencial) Se tudo estiver conforme a política de proteção descrita pelo serviço, a mensagem segue para o manipulador seguinte, ou seja, o manipulador de controle de acesso (MCA).

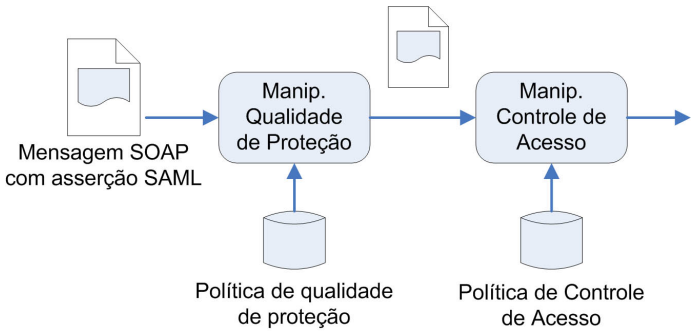

Figura 6.4: Atuação do manipulador política de proteção

A qualidade de proteção é verificada perante análise ao cabeçalho da mensagem. Caso alguma verificação de segurança falhe uma mensagem de erro é retornada. Desta forma, assegura-se que as mensagens recebidas são seguras e estão de acordo com a política de proteção. Vale lembrar que na presença de asserções SAML, anexadas como credenciais na mensagem, este manipulador a repassa ao manipulador de controle de acesso. Segundo as proposições do modelo, apresentadas no capítulo anterior, é possível o cliente obter acesso apenas perante a apresentação de uma asserção emitida por seu STS/IdP.

6.1.3.1 Manipulador de Controle de Acesso (PEP e PDP)

Como o próprio nome diz, é responsável pelo controle de acesso. Implementa a dinâmica de controle de acesso proposta no capítulo anterior. Desta forma, além das ações de permitir ou negar acesso a aplicação, este manipulador também é responsável por requerer a tradução da credencial de segurança (caso receba uma asserção SAML de autenticação e sua política de proteção exija um credencial X.509 ou SPKI, por exemplo) e por atributos do solicitante. Sempre que uma mensagem é recebida este manipulador é acionado a fim de verificar se o pedido pode ser atendido.

As políticas de acesso ao sistema estão armazenadas no formato XACML (OASIS, 2005a) e a decisão e aplicação das mesmas são feitas através do PDP (*Policy Decision Point*) e PEP (*Policy Enforcement Point*), respectivamente (Yavatkar *et al.*, 2000). A ação a ser tomada pelo PEP pode ser de três tipos: aceitar a mensagem ou não a mensagem, ou solicitar alguma ação ao seu STS/IdP.

A mensagem é aceita quando está de acordo com a política de autorização do serviço. Quando o serviço é incapaz de entender a credencial SAML, a asserção é repassada ao seu STS/IdP, que tem a responsabilidade de traduzí-la para uma credencial entendível pelo serviço. Nesta comunicação com o STS/IdP, o serviço também pode solicitar que o seu STS/IdP obtenha alguns atributos do sujeito para então permitir acesso ao recurso, conforme Seção 5.2.5.4 do modelo apresentado. A solicitação de atributos obedece a lógica da aplicação. Por exemplo, se é preciso o número do CEP do cliente, um pedido por tal atributo é realizado ao seu STS/IdP.

Este manipulador faz uso de duas classes, uma chamada PEP e outra classe chamada PDP. A asserção SAML é repassada como parâmetro para a classe PEP que então solicita que as declarações da asserção, como, por exemplo, o emissor, o sujeito, o periodo de validade, etc, sejam confrontadas com a política de autorização do serviço.

O protocolo utilizado para as interações entre o PEP e o PDP é o definido na especificação XACML. Já para a interação entre o PDP e o STS/IdP, o protocolo é o *RequestSecurityToken* (RST) e *RequestSecurityTokenResponse* (RSTR) (WS-Trust, 2005).

6.1.3.2 Motor de Políticas

Originalmente apresentado em (Baligand and Monfort, 2004), este manipulador é responsável por recuperar e aplicar as configurações relacionadas a política de qualidade de proteção. Quando a aplicação se comporta como cliente o MP é responsável por ler a política de qualidade de proteção do serviço via WSDL do serviço. Tal política pode estar explicitamente contida na WSDL ou referenciada em um URI. O MP também permite verificar se as políticas de qualidade de proteção definidas pelo cliente e pelo serviço são compatíveis. Para tanto, esta realiza a intersecção entre as políticas do cliente e do serviço para verificar se há a possibilidade de interação entre as partes envolvidas. No caso da impossibilidade de comunicação, a intersecção resulta em vazio.

6.2 Dinâmica do protótipo

A dinâmica do protótipo é apresentada em um cenário no qual o cliente apenas consegue acesso ao serviço se este cumpre com as exigências da política do serviços. Nesse sentido, o cliente obtém uma resposta do serviço perante a apresentação de uma mensagem assinada pelo cliente, contendo uma credencial segundo a infra-estrutura de segurança do serviço e assinada (emitida) por uma terceira parte confiável, no caso o STS/IdP do serviço. A Figura 6.5 apresenta os elementos envolvidos.

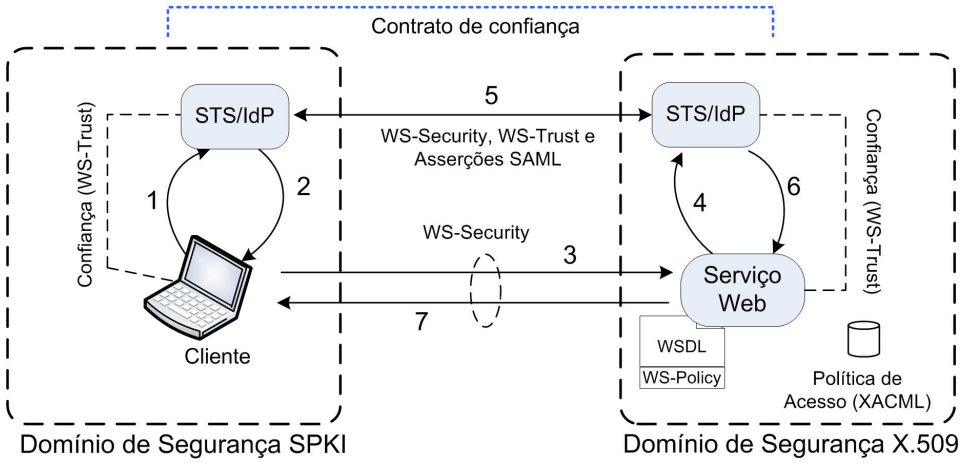

Figura 6.5: Transposição no Modelo

Para exibir a dinâmica do protótipo, a seguir ilustra-se o diagrama de seqüência de dois casos. O primeiro caso envolve um cliente e um STS/IdP, ambos em um domínio sob uma infra-estrutura de segurança SPKI. O cliente já está filiado ao seu STS/IdP, ou seja, os dois se conhecem e possuem uma relação de confiança direta. Desta forma, o cliente envia um pedido assinado para que o STS/IdP o autentique, e recebe uma asserção de autenticação que declara que o cliente é membro do domínio e foi autenticado perante o STS/IdP. Antes de enviar a asserção SAML ao cliente, o STS/IdP verifica se este é seu filiado. O Segundo caso

exibe a interação entre cliente e provedor de serviços. O cliente obtém a WSDL do serviço e realiza a leitura referente à qualidade de proteção.

O diagrama da Figura 6.6 apresenta o primeiro caso, no qual ocorre a aquisição da credencial de segurança pelo cliente. O cliente faz o pedido pela credencial e o manipulador STS/IdP é responsável por adquiri-la. Para tanto, o STS/IdP conta com o auxílio de dois objetos (*STS/IdP Client* e *STS/IdP Security*), conforme exibe a Figura 6.6, que são responsáveis por solicitar a credencial e adicionar propriedades de segurança à mensagem. No passo 5, a mensagem é enviada ao STS/IdP A. Antes, porém, de atingir o serviço, a mensagem é verificada pelos manipuladores MPP (Manipulador de Política de Proteção) e MCA (Manipulador de Controle de Acesso), os quais verificam se a política de qualidade de proteção está de acordo com as exigências do serviço e se o cliente é membro do domínio e tem direito a obtenção de uma credencial, respectivamente. No passo 8, o STS/IdP retorna a credencial solicitada pelo cliente e este último a utilizará para interagir com serviços localizados em outros domínios.

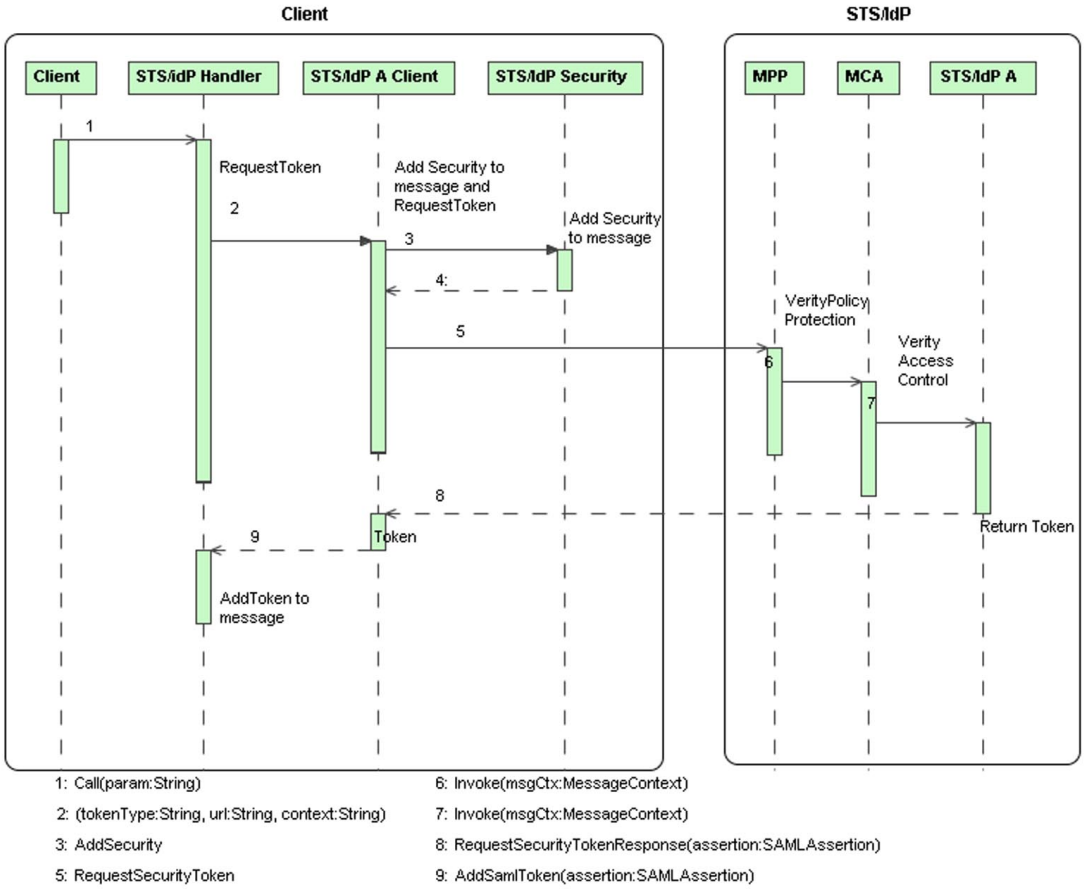

Figura 6.6: Primeiro Caso

O diagrama da Figura 6.7 apresenta o segundo caso, onde o cliente envia uma solicitação ao serviço. Como já possui a asserção SAML de autenticação, o cliente agora assina a mensagem por meio do manipulador MA (Manipulador Assinatura). No passo 2, a mensagem é

enviada ao serviço que, antes de retornar a resposta, a verifica por meio dos manipuladores MPP e MCA. O manipulador MCA, ao verificar a presença de uma asserção SAML de autenticação, solicita ao seu manipulador STS/IdP a troca da credencial, pois sua exigência é por uma credencial no formato X.509. O manipulador STS/IdP então aciona o STS/IdP do serviço (STS B) para que este faça a troca da credencial (passo 5). A credencial no formato X.509 é então recebido pelo MCA (passo 7). Este então avalia novamente a mensagem e a repassa ao serviço, o qual concede o artigo solicitado pelo cliente (passo 9).

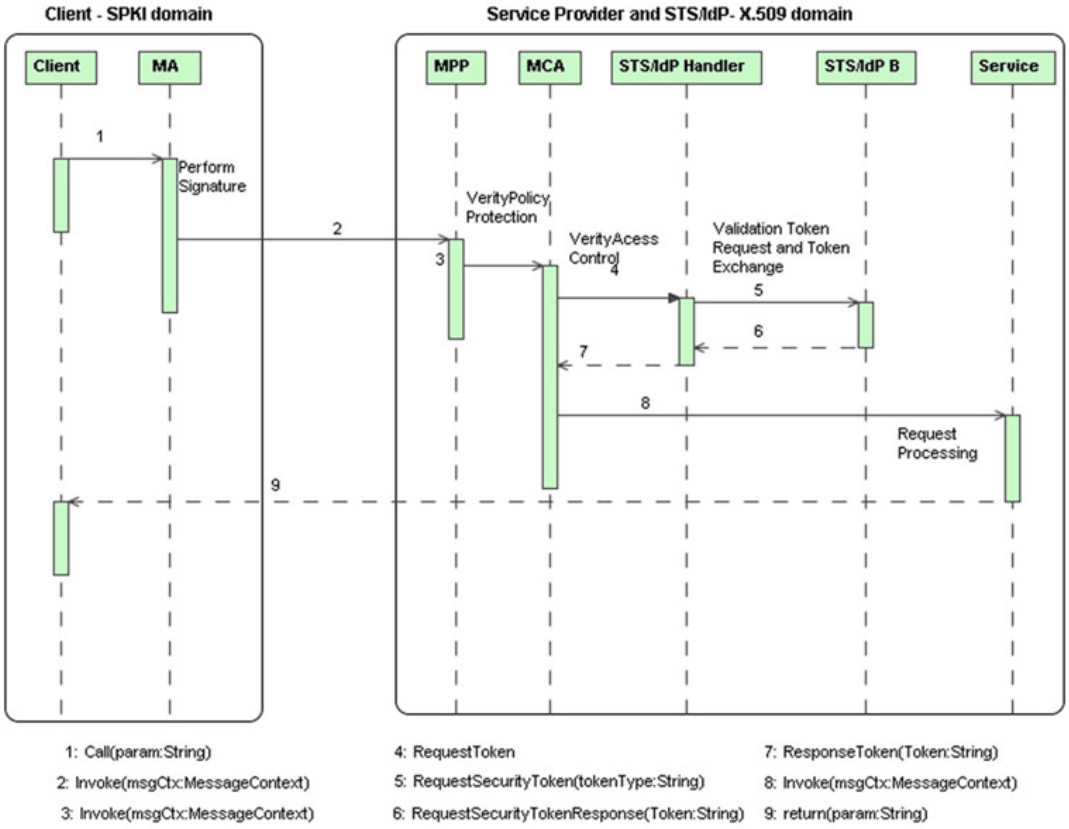

Figura 6.7: Segundo Caso

6.3 Considerações sobre a Implementação

O protótipo implementadado atingiu os objetivos na transposição das informações de autenticação. Um cliente concretiza a interação com um provedor de serviço desconhecido por meio dos conceitos apresentados nesse trabalho.

Os manipuladores implementados agregam ao cliente e ao serviço as propriedades básicas de segurança e fazem a ligação com o STS/IdP. O módulo Motor de Políticas, embora especificado neste trabalho, não foi implementado. O mesmo é basicamente um analisador sintático que observa a especificação *WS-Policy* (WS-Policy, 2004) e retira os elementos relacionados à segurança afim de configurar os manipuladores.

O STS/IdP implementando neste trabalho autentica o principal e realiza a emissão, validação e troca de credenciais de segurança. Implementações futuras ainda são necessárias para o Serviço de Atributos e Pseudônimos e o Serviço Tradutor de Credenciais.

Questões relacionadas à escalabilidade e tolerância a faltas devem ser melhor observadas em implementações futuras. Os testes realizados não demonstraram problemas de escalabilidade, porém, deve-se agregar ao manipulador política de proteção (MPP) formas deste perceber um ataque de negação de serviço.

6.4 Avaliação de Desempenho

O testes realizados objetivaram uma análise dos resultados considerando a latência média introduzida pelo processamento das mensagens por parte dos manipuladores.

Os testes foram realizados no LIDAS/DAS, em uma rede local (*Fast Ethernet* - 100 Mbps) e três computadores dedicados: (A) Pentium IV 2.4 GHz com 512 MB de memória RAM, tendo como sistema operacional o Windows XP; (B) Pentium IV 3.0 GHz com 1.0 GHz de memória RAM e o sistema operacional GNU/Linux; (C) Athlon XP 2.6 GHz com 512 de memória RAM e sistema operacional GNU/Linux. As máquinas dispunham da versão 1.5.0-01 do Java 2 Software Development Kit (J2SDK), da versão 5.5 do servidor de aplicação TomCat e versão 1.3 do Axis. O cliente foi hospedado na máquina "A", o provedor de serviços foi hospedado na máquina "B" e os seus respectivos STS/IdP alojados na máquina "C".

Os resultados foram obtidos em duas situações distintas. Na primeira, mediu-se o tempo de latência da autenticação do cliente em seu STS/IdP com diferentes mecanismos de autenticação, sendo estes: autenticação via usuário e senha e autenticação via assinatura digital[10] (X.509 e SPKI). O algoritmo utilizado para a assinatura digital foi o hash-one-way SHA-1 com o tamanho das chaves de 1024 *bits*. Na segunda situação, mediu-se a latência das trocas de mensagens entre cliente e provedor de serviços com a presença e ausência de propriedades de segurança. Para esta ultima situação se variou o tamanho das mensagens trocadas de 256 *bytes* e 512 Kb.

A Tabela 6.1 apresenta a primeira situação. Os resultados apresentados são a média sobre 10 interações entre cliente e STS/IdP. Como pode ser observado, a autenticação por meio de usuário e senha apresentou velocidade de processamento superior às assinaturas X.509 e SPKI. A autenticação por usuário e senha, no entando, deve contar com um canal seguro, conforme (OASIS, 2004c).

A Figura 6.8 apresenta as interações entre cliente e provedor de serviços. O cliente está sob a infra-estrutura SPKI, apresenta uma asserção SAML de autenticação a um serviço

[10] autenticação mútua

Tabela 6.1: Latência perante autenticação do cliente no STS/IdP

Mecanismo de autenticação - Rede Local	**Tempo Médio (ms)**
UsernameToken	204,7
Assinatura X.509	260,9
Assinatura SPKI	332,8

sob tecnologia X.509. Nesse caso, o cliente obtém a asserção no seu STS/IdP, assina a mensagem, e a envia ao serviço. O provedor de serviços então solicita a troca da asserção SAML por um certificado X.509.

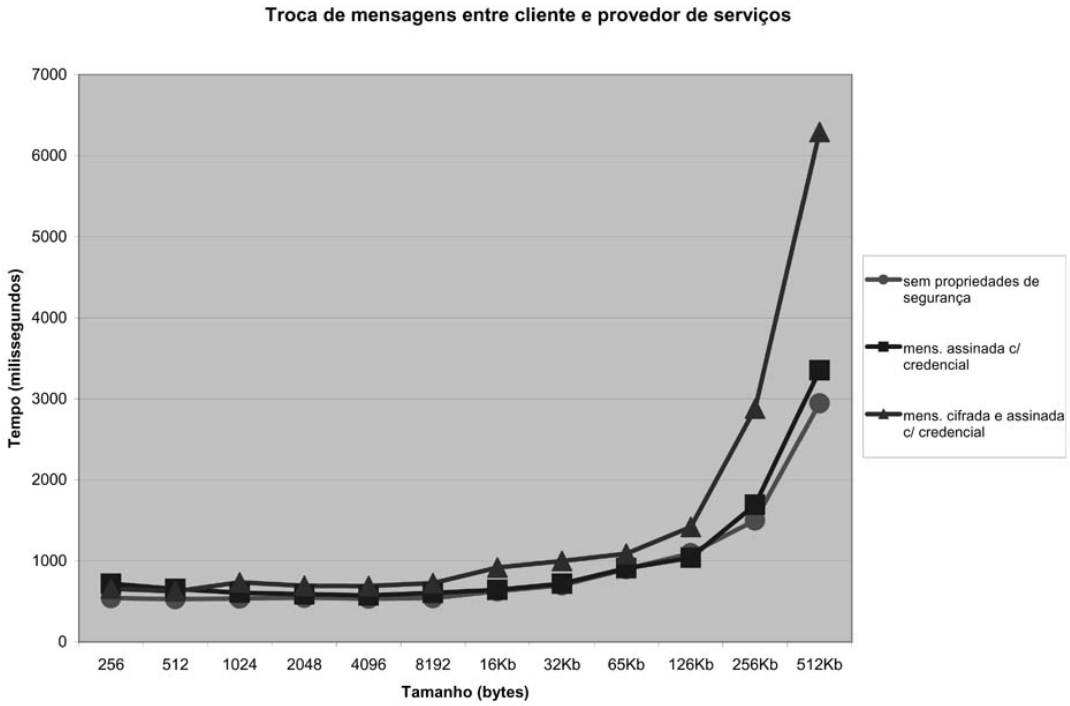

Figura 6.8: Resultados - Cliente e Provedor de Serviços em domínios distintos

Nos resultados da Figura 6.8, observa-se que o custo computacional das trocas de mensagens está principalmente no tempo de latência da rede e da serialização/deserialização XML. A operação de cifragem foi processada sobre a mensagem SOAP completa, o que elevou o tempo de processamento, principalmente para as mensagens com tamanho acima de 8 Kb. Para obter melhor desempenho no tempo de processamento, cifrar apenas partes das mensagens traria um melhor resultado.

6.5 Comparação com os Trabalhos Relacionados

Na literatura, alguns trabalhos se esforçam em reunir um conjunto de especificações relativas a Serviços *Web* para efetuar a transposição de autenticação. Entre estes, a maioria se apóia em relações de confiança previamente estabelecidas entre os domínios. Sendo que a

atenção está principalmente voltada a trabalhos que envolvem identidades federadas através de Serviços *Web*. A maioria dos trabalhos apenas considera o uso de certificados X.509. Além disso, não há clareza sobre como ocorre a interação entre domínios com diferentes infra-estruturas de segurança e em como manter as características de segurança do cliente ou do provedor de serviço nestas interações entre domínios com tecnologias distintas.

O CredEx (Vecchio *et al.*, 2005) facilita o gerenciamento de credenciais por parte do usuário por meio um repositório. Desta forma, o cliente armazena as diversas credenciais envolvendo várias tecnologias de segurança num repositório centralizado e as usa de acordo com os serviços invocados e de suas respectivas tecnologias de segurança. O CredEx em seus experimentos apenas considera a associação prévia de um usuário e senha a um certificado X.509 *proxy* ou vice-versa. Sendo assim, não lida com diferentes tecnologias de segurança, como SPKI ou Kerberos. Também não permite que credenciais sejam traduzidas dinamicamente.

Na solução aqui apresentada, há a conversão dinâmica da credencial pelo provedor de serviço, permitindo ao cliente manter seus atributos tomando como base a sua infra-estrutura de segurança completamente independente das possíveis interações com outros domínios. Através do Serviço Tradutor de Credenciais é possível que além do X.509 e do SPKI/SDSI, credenciais segundo o formato Kerberos possam ser traduzidas e assim ocorra a comunicação mesmo que os domínios envolvidos possuam diferentes infra-estruturas de segurança subjacentes.

A especificação *WS-Federation* (WS-Federation, 2003a) fornece os serviços de STS/IdP e Serviço de Atributos e Pseudônimos para as identidades federadas introduzidas no modelo apresentado. Porém, esta mesma especificação considera que os atributos do usuário são centralizados nos serviços descritos (seguem o modelo do *Passport* da Microsoft). O trabalho aqui proposto estende esse modelo para permitir a busca nos domínios em que o usuário tem a sua identidade reconhecida como federada, o que permite o usuário ser referenciado com mais de um identificador ou mesmo pseudônimos. O projeto *InfoCard*, da *Microsoft*, prevê tal extensão da *Ws-Federation*, mas não faz uso de asserções SAML para conduzir os atributos do usuário e sim de uma credencial de transferência chamada *InfoCard*. A *WS-Federation*, também não esclarece como se dá a troca de credenciais entre domínios distintos.

O trabalho de Dyke (Dyke, 2004) apóia-se em especificações como *WS-Trust* e *WS-Federation* e propõe uma arquitetura para construção dinâmica de redes de confiança entre Serviços *Web* no contexto de serviços médicos. O trabalho se preocupa principalmente com o estabelecimento dinâmico da confiança. Para tanto, propõe níveis de confiança (*trust levels*), grupos de confiança (*trust groups*) e autoridades de confiança (*trust authorities*) para auxiliar o STS na emissão de uma credencial a determinado cliente. Este trabalho trás uma importante contribuição no sentido de tentar estabelecer diferentes níveis de confiança. As relações dinâmicas seriam criadas após a avaliação dos níveis e grupos de confiança, segundo a política de segurança do serviço. No entanto, o trabalho não se preocupa com

outras relevantes questões, tais como lidar com diversas tecnologias de segurança presentes na federação, com atributos dos clientes nem com o processo de autorização. Além disso, incluir informações na credencial para transportar os níveis e grupos de confiança compromete a interoperabilidade, uma vez que o mesmo não define uma sintaxe para tanto. O trabalho também não explora o potencial das asserções SAML, que naturalmente incluem informações sobre a autenticação do usuário e são extensíveis.

O Projeto Shibboleth (Shibboleth, 2005) está baseado no conceito de federações, permitindo a transposição dos mecanismos de autorização e autenticação. Também prevê a troca de atributos através de asserções SAML, fazendo o uso de uma padronização dos mesmos. O Shibboleth assume que os usuários fazem uso de navegadores *web* tradicionais para acessar os recursos. Além disso, há uma grande preocupação do Shibboleth quanto à privacidade e anonimato dos usuários. As maiorias das organizações que adotam o Shibboleth fazem uso do padrão X.509 e prevêem que os usuários se autenticam perante usuário e senha. A proposta aqui apresentada difere do Shibboleth por dar mais liberdade aos membros da federação. Estes não precisam fazer uso de um navegador *Web* em suas interações com serviços, desta forma, permite também a comunicação direta entre provedores de serviços, ou seja, provedor a provedor. Provedores de serviços não preciso seguir uma rígida padronização para fazer parte da federação, basta estarem associados a um STS/IdP. Além disso, também podem permanecer com sua tecnologia de segurança e fazer uso do Tradutor de Credenciais quando há necessidade de lidar com diferentes tecnologias. Outro ponto, é que em Serviços *Web* as interações não se dão apenas entre o navegador *Web* do usuário e o provedor de serviço, mas também entre provedores de serviços

Cardea (Kafura *et al.*, 2003) é um sistema de autorização distribuído, desenvolvido como parte da Grade Computacional de Informações da NASA (NASA *Information Power Grid*) (Foster *et al.*, 1998; Foster and Kesselman, 1998; Nasa, 2003). Construído sobre padrões como SAML, XACML e XMLDSig, o sistema avalia dinamicamente os pedidos de autorização, considerando as características dos recursos e do solicitante ao invés de avaliar apenas identidades locais. Os usuários são identificados por certificados X.509 *proxy*. As informações necessárias para completar uma decisão de autorização são avaliadas e coletadas durante o processo de autorização. Essa informação é recolhida de forma apropriada e apresentada ao PDP, que retorna a decisão de autorização final para o pedido, junto com qualquer detalhe relevante. *Cardea* é implementado em JAVA com um conjunto de componentes independentes. Ao contrário do trabalho aqui proposto, o Cadea constrói a federação através do padrão SAML, ou seja, autoridades SAML, e não através das especificações *WS-Trust* e *WS-Federation*, além disso não considera diferentes tecnologias de segurança envolvidas no processo de autorização.

6.6 Conclusão

Este capítulo descreve as experiências de implementações e de avaliação do modelo proposto. Foi desenvolvido um protótipo cujo principal objetivo era validar a aplicabilidade do modelo de segurança proposto nesta dissertação. Nesse sentido, as ferramentas utilizadas - *Axis* e a biblioteca WSS4J - serviram de base na construção. A extensão a biblioteca WSS4J foi necessária para viabilizar o modelo proposto.

O protótipo construído atende grande parte das características impostas pelo modelo proposto. Os manipuladores, apesar de introduzirem latência nas trocas de mensagens realizadas, mostraram-se eficazes e de fácil adoção. Por meio dos manipuladores o modelo ganha flexibilidade.

Capítulo 7

Conclusão

Diante da proposta de fazer uso de Serviços *Web* e do gerenciamento de identidades federadas para atingir a transposição de autenticação entre domínios que possuem diferentes infra-estruturas de segurança, este trabalho acredita que, humildemente, colaborou para responder as questões da pesquisa que motivaram essa dissertação e alcançar os objetivos pretendidos na introdução.

Em relação as questões de pesquisa levantadas inicialmente, obte-se as seguintes conclusões:

- constatou-se que a principal dificuldade para atingir a transposição entre domínios está na necessidade de um acordo prévio entre os domínios para que haja um certo nível de padronização como, por exemplo, a definição da sintaxe para reger a troca de atributos;
- a interação entre domínios distintos e que estão sob deferentes infra-estruturas de segurança é possivel. O uso de asserções SAML permite tanto carregar informações dos principais quanto oferecer uma forma interoperável de reconhecer o principal em outro domínio. A autoridade do domínio, no caso o STS/IdP em um domínio X.509, atua como uma certificadora *online*. Isto exige uma forte relação de confiança estabelecida entre os domínios. Além disso, os provedor de serviços devem confiar na chave privada do seu STS/IdP;
- a confiança estabelecida entre as partes poderia ser base para a transposição das informações de autorização através das asserções SAML. No entanto, esforços de padronização das políticas de autorização e a sua atualização em todos os provedores filiados a federação necessitam ser realizados.
- a tecnologia de serviços *Web* é eficaz quanto a integração de clientes e provedores de serviços nas interações inter-domínio. Um grande esforço foi realizado no sentido de reunir as propostas de segurança em Serviços *Web*. Constatou-se que a tecnologia de Serviços *Web* cumpre com os seus objetivos de fraco acoplamento e que esforços

para manter a segurança computacional devem ser mantidos, perseguindo sempre a interoperabilidade entre as especificações;

- O modelo proposto, devido a sua fundamentação em padrões consolidados e sua forma inicialmente genérica, favoreceu uma melhor compreensão da dinâmica da transposição em sistemas distribuídos de larga escala no paradigma da arquitetura orientada a serviços e dos desafios para atingir a integração entre diferentes domínios.
- O protótipo implementado mostrou-se flexível, eficiente e de fácil configuração. O uso de manipuladores permite ainda que sejam agregadas ao protótipo outras definições. Uma questão que precisa ser melhor trabalhada em trabalhos futuros é em relação a administração da política de qualidade de proteção. Esta é crucial para a interação entre os domínios. Conhecer e administrar a política de qualidade de proteção dos serviços é necessário para uma boa interação entre cliente e provedor de serviços.

Os objetivos inicialmente traçados para esta dissertação foram atingidos. Foi concebido um modelo genérico para permitir que com base na autenticação realizada no seu domínio de origem o principal acesse recursos espalhados nos diferentes domínios pertencentes a suas relações de confiança, de forma transparente e interoperável. A comunicação entre diferentes domínios foi atingida por meio das asserções SAML.

Após o desenvolvimento modelo proposto e a implementação do protótipo, as principais contribuições desta dissertação podem ser assim identificadas:

- no decorrer deste trabalho de dissertação, implementou-se um protótipo com a abordagem do cliente ativo. A aplicação definida em (de Melo E.R. *et al.*, 2004) foi adaptada conforme a abordagem do cliente ativo e obedecendo o artigo publicado em (de Mello and da Silva Fraga, 2005);
- este trabalho definiu e implementou uma abordagem na qual o provedor de serviços é a entidade ativa;
- a implementação de características ausentes nas APIs de segurança para Serviços *Web*, como a emissão de asserções SAML no STS da API WSS4J.

Os documentos científicos produzidos nesta dissertação foram:

- MELLO, E. R. ; FRAGA, Joni da Silva ; CAMARGO, Edson.Transferência de autenticação e autorização através de Serviços Web. In: V Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, 2005, Florianópolis - SC. SBSEG 2005;
- Mini-curso sobre Segurança em Serviços Web submetido ao SBSeg 2006. FRAGA, Joni ; MELO, Emerson ; CAMARGO, E. T. ; Wangham, Michelle S. . Segurança em Serviços Web. In: Lau C. Lung. (Org.). Minicursos do SBSEG 2006. : , 2006, v. , p. -.

Como propostas de trabalhos futuros, sugere-se a implementação do motor de políticas. Este traria maior flexibilidade ao modelo e uma forma simples dos administradores de sistemas gerenciarem as diferentes políticas de qualidade de proteção dos serviços. Outra sugestão também está na implementação do Serviço de Atributos e Pseudônimos. As interações entre os STS/IdPs dos diferentes domínios para concretizar a troca de atributos também precisa ser efetuada.

A adaptação da aplicação (de Melo E.R. *et al.*, 2004) para o provedor de serviços ativo também um trabalho a ser realizado.

Referências Bibliográficas

(1998). Extensible markup language (XML) 1.0. Relatório Técnico REC-xml-19980210, W3C.

(2002). XML-signature syntax and processing. W3c recommendation, IETF / W3C. Also published as IETF Internet Request for Comment RFC 3275, Mar , 2002.

(2003). Bindings and profiles for the OASIS security assertion markup language (SAML). Relatório Técnico v1.1, OASIS.

15408-1:1999, I. (1999). *Introduction and general model*. ISO/IEC.

Adams, C. and Farrell, S. (1999). Internet X.509 public-key infrastructure — certificate management protocols. Internet proposed standard RFC 2510.

Amoroso, E. (1994). *Fundamentals of Computer Security Technology*. Prentice Hall, Englewood Cliffs, NJ.

Anderson, J. P. (1972). Computer security technology planning study. Relatório técnico, ESD-TR-73-51.

Apache (2005). *Axis Architecture Guide v1.2*. The Apache Software Foundation. `http://ws.apache.org/axis/java/architecture-guide.pdf`.

Baligand, F. and Monfort, V. (2004). A concrete solution for web services adaptability using policies and aspects. Em Aiello, M., Aoyama, M., Curbera, F., and Papazoglou, M. P., editors, *ICSOC*, pp. 134–142. ACM.

Bishop, M. (2003). *Computer Security: Art and Science*. Addison-Wesley, Boston, USA.

Boyer, J. M. (2001). Canonical XML version 1.0. World Wide Web Consortium, Recommendation REC-xml-c14n-20010315.

BURNETT, Steve; PAINE, S. (2002). *Criptografia e segurança : o guia oficial RSA. Tradução de Edson Furmankiewicz*. Campus, Rio de Janeiro - BR.

Burr, W., Dodson, D., Nazario, N., and Polk, W. T. (1998). Minimum interoperability specification for PKI components (MISPC), version. NIST Special Publication 800-15, NIST.

Clarke, D. E., Elien, J.-E., Ellison, C. M., Fredette, M., Morcos, A., and Rivest, R. L. (2001). Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, Vol. 9, No. 4, pp. 285–322.

Corporation, I. and Corporation, M. (2002). *Security in a Web Service World: A Proposed Architecture and Roadmap.* `http://msdn.microsoft.com/ws-security/`.

Coulouris, G. F., Dollimore, J., and Kindberg, T. (2001). *Distributed Systems, concepts and design, Edition 3*. Addison-Wesley, isbn0-201-62433-8 edição.

Cowan, J. and Tobin, R. (2001). XML information set. World Wide Web Consortium, Recommendation REC-xml-infoset-20011024.

Czajkowski, K., Foster, I., and Kesselman, C. (1999). Co-allocation services for computational grids. Proc. *Proc. 8th IEEE Symp. on High Performance Distributed Computing*. IEEE Computer Society Press.

Damiani, E., di Vimercati, S. D. C., and Samarati, P. (2003). Managing multiple and dependable identities. Proc. *IEEE Internet Computing*, pp. 29–37. IEEE.

de Mello, E. R. (2003). *Redes de Confiança em Sistemas de Objetos CORBA*. Tese (mestrado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.

de Mello, E. R. (2005). *Relações de Confiança e Segurança em Ambientes Orientados a Serviços*. Exame de qualificação (doutorado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.

de Mello, E. R. and da Silva Fraga, J. (2005). Mediation of trust across web services. Proc. *ICWS*, pp. 515–522. IEEE Computer Society.

de Melo E.R., G., A., and J., F. (2004). Integração da arquitetura de segurança dos serviços web com modelos de confiança igualitária. Proc. *In VI Simpósio de Segurança em Informática (SSI04)*, São José dos Campos - SP - Brasil.

Demchenko, Y., Gommans, L., and de Laat an Bas Oudenaarde, C. (2005). Web services and grid security vulnerabilities and threats analysis and model. Proc. *SC'05: Proc. The 6th IEEE/ACM International Workshop on Grid Computing CD*, pp. 262–267, Seattle, Washington, USA. IEEE/ACM.

Denning, D. E. (1976). A lattice model of secure information flow. *Communications of the Association of Computing Machinery*, Vol. .

Department of Defense (1985). *DoD 5200.28-STD: Department of Defense (DoD) Trusted Computer System Evaluation Criteria (TCSEC)*.

Dierks, T. and Allen, C. (1997). The tls protocol version 1.0. Internet Draft, TLS working group.

Division, N. A. S. N. (2003). *Cardea: Dynamic Access Control in Distributed System*. http://www.nas.nasa.gov/News/Techreports/2003/PDF/nas-03-020.pdf.

Dyke, J. W. V. (2004). Establishing federated trust networks among web services.

Eastlake, D. E., Reagle, J. M., Imamura, T., Dillaway, B., and Simon, E. (2002). XML encryption syntax and processing. World Wide Web Consortium, Recommendation REC-xmlenc-core-20021210.

Ellison, C. M. (1998). Spki certificate documentation. See http://www.clark.net/pub/cme/html/spki.html.

Ellison, C. M. (1999). *RFC 2692: SPKI requirements*. The Internet Society. See ftp://ftp.isi.edu/in-notes/rfc2692.txt.

et al, M. (2001). Akenti a distributed access control system.

Fallside, D. C. and Walmsley, P. (2004). *XML Schema Part 0: Primer Second Edition*. W3C. http://www.w3.org/TR/xmlschema-0.

for the Advancement of Structured Information Standards, O. (2006). *OASIS Web Services Security (WSS) TC*. OASIS. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

Foster, I., Hiro, H., Savva, A., Berry, D., Djaoui, A., Grimshaw, A., and et. al. (2005). The open grid services architecture v1.0. Relatório Técnico GFD-I.030, The University of Edinburgh.

Foster, I. and Kesselman, C. (1998). Globus: A toolkit-based grid architecture. Proc. *The Grid: Blueprint for a Future Computing Infrastructure*, pp. 259–278. MORGAN-KAUFMANN.

Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S. (1998). A security architecture for computational grids. Proc. *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS-98)*, pp. 83–92, New York. ACM Press.

Geer, D. (2003). Taking steps to secure web services. *IEEE Computer*, Vol. 36, No. 10, pp. 14–16.

Gerck, E. (2000). Overview of certification systems: X.509, PKIX, CA, PGP and SKIP. *The Bell*, Vol. 1, No. 3, pp. 8.

Gutierrez, C., Fernandez-Medina, E., and Piattini, M. (2004). Web services security: is the problem solved? Proc. *WOSIS*, pp. 293–304.

Hallam-Baker, P. M. and Ford, W. (2001). XML key management specification (XKMS). Proc. *WWW Posters*.

Hallam-Baker, P. M. and Mysore, S. H. (2005). XML key management specification (XKMS) — version 2.0. World Wide Web Consortium, Recommendation REC-xkms2-20050628.

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

InCommon (2006). Incomm federation: Common identity attributes.

Jøsang, A., Fabre, J., Hay, B., Dalziel, J., and Pope, S. (2005). Trust requirements in identity management. Proc. *CRPIT '44: Proceedings of the 2005 Australasian workshop on Grid computing and e-research*, pp. 99–108, Darlinghurst, Australia.

Jøsang, A. and Pope, S. (2005). User centric identity management. Proc. *AusCERT Asia Pacific Information Technology Security Conference 2005*.

Kafura, D., Lorch, M., Lepro, R., Proctor, S., and Shah, S. (2003). First experiences using XACML for access control in.

Kimery, K. M. and McCord, M. (2002). Third-party assurances: The road to trust in online retailing. Proc. *HICSS*, pp. 175.

Kohl, J. and Neuman, C. (1993). Rfc 1510: The kerberos network authentication service (v5). Status: PROPOSED STANDARD.

Lampson, Abadi, Burrows, and Wobber (1992). Authentication in distributed systems: Theory and practice. *ACMTCS: ACM Transactions on Computer Systems*, Vol. 10.

Lampson, B. W. (1974). Protection. *Operating Systems Review*, Vol. 8, No. 1, pp. 18–24.

Landwehr, C. E. (1981). Formal models for computer security. *ACM Computing Surveys*, Vol. 13, No. 3, pp. 247–278.

Landwehr, C. E. (2001). Computer security. *Int. J. Inf. Sec*, Vol. 1, No. 1, pp. 3–13.

Liberty (2003). *Introduction to the Liberty Alliance Identity Architecture*. Liberty Alliance. `http://www.projectliberty.org/`.

Liberty (2004). *Trust Models Guidelines*. Liberty Alliance.

Markham, T. (1997). Internet security protocol. *Dr. Dobb's Journal of Software Tools*, Vol. 22, No. 6, pp. 70–??

Maruyama, H., Nakamura, T., and Hsieh, T. (2003). Optimistic fair contract signing for web services. Proc. *ACM workshop on XML security (XMLSEC'03)*, pp. 79–85, New York, NY, USA. ACM Press.

Morcos, A. (1998). *A Java implementation of Simple Distributed Security Infrastructure*. Dissertação de mestrado, MIT, MIT.

Morgan, R. L., Cantor, S., Carmody, S., Hoehn, W., and Klingenstein, K. (2004). Federated security: The shibboleth approach. *EDUCAUSE Quarterly*, Vol. 27, No. 4, pp. 4–6.

Nasa (2003). The information power grid.

National Institute of Standards and Technology (NIST) (1992). The Digital Signature Standard, proposed by NIST. *j-CACM*, Vol. 35, No. 7, pp. 36–40.

Nazareth, S. P. (2003). SPADE: SPKI/SDSI for attribute release policies in a distributed environment. Relatório Técnico TR2003-453, Dartmouth College, Computer Science, Hanover, NH.

Neuman, B. C. (1994). *Scale in distributed systems*, pp. 463–489. IEEE Computer Society, Los Alamitos, CA.

Neuman, B. C. and Ts'o, T. (1994). Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, Vol. 32, No. 9, pp. 33–38.

Nicomette, V. (1996). *La Protection dans les Systèmes à Objets Répartis*. Phd thesis, Institut National Polytechnique, Toulouse.

OAIS (2005). *Web Services Transaction (WS-TX) TC*. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx`.

OASIS (2002a). *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*. Organization for the Advancement of Structured Information Standards (OASIS).

OASIS (2002b). *Universal Description, Discovery, and Integration (UDDI) version 2.0*. Organization for the Advancement of Structured Information Standards (OASIS).

OASIS (2004a). *Web Services Security: SAML Token Profile*. Organization for the Advancement of Structured Information Standards (OASIS). `http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf`.

OASIS (2004b). *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 (2004c). *Web Services Security UsernameToken Profile 1.0*. OASIS. `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0`.

OASIS (2004d). *Web Services Security X.509 Certificate Token Profile*. OASIS. `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0`.

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

OASIS (2005c). *Web Services Security Kerberos Token Profile 1.1*. OASIS. http://www.oasis-open.org/committees/download.php/15254/oasis-wss-kerberos-token-profile-1.1.pdf.

Obelheiro, R. R. (2001). *Modelos de Segurança Baseados em Papéis para Sistemas de Larga Escala: A Proposta RBAC-JacoWeb*. Tese (mestrado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.

P Kearney, J Chapman, N. E. M. G. and He, L. (2004). An overview of web services security. *BT Technology Journal*, Vol. 22, No. 1, pp. 27–42.

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. Proc. *WISE*, pp. 3–12.

RSA Laboratories (2002). *PKCS #1 v2.1: RSA Cryptography Standard*. pub-RSA, pub-RSA:adr.

Sandhu, R. S. and Samarati, P. (1994). Access control: Principles and practice. *IEEE Communications Magazine*, Vol. 32, No. 9, pp. 40–48.

Sandhu, R. S. and Samarati, P. (1997). Authentication, access controls, and intrusion detection. Proc. *The Computer Science and Engineering Handbook*, pp. 1929–1948.

Santin, A. O. (2004). *Teias de Federação: uma Abordagem Baseada em Cadeias de Confiança para Autenticação, Autorização e Navegação em Sistemas de Larga Escala*. Tese (doutorado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.

Secure Electronic Transaction LLC (1997). SET secure electronic transaction specification — version 1.0.

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

Shim, S. S. Y., Bhalla, G., and Pendyala, V. S. (2005). Federated identity management. *IEEE Computer*, Vol. 38, No. 12, pp. 120–122.

Skogsrud, H., Benatallah, B., and Casati, F. (2003). Model-driven trust negotiation for web services. *IEEE Internet Computing*, Vol. 7, No. 6, pp. 45–52.

SOAP (2003). Simple object access protocol (soap) 1.2. W3C Note.

Stallings, W. (2000). *Network Security Essentials: Applications and Standards*. Prentice-Hall, Englewood Cliffs, New Jersey.

Technology, W. (2003). *Guide to XML Web Services Security*. http://www.westbridgetech.com/.

Tuecke, S., Welch, V., Engert, D., Pearlman, L., and Thompson, M. (2004). Internet X.509 public key infrastructure (PKI) proxy certificate profile. Internet proposed standard RFC 3820.

Vecchio, D. D., Humphrey, M., Basney, J., and Nagaratnam, N. (2005). Credex: User-centric credential management for grid and web services. Proc. *ICWS*, pp. 149–156.

Vogels, W. (2004). Web services are not distributed objects. Proc. *Int. CMG Conference*, pp. 317–324.

W3C (2004a). *Web Services Architecture*. http://www.w3.org/TR/ws-arch/.

W3C (2004b). *Web Services Description Language Version 2.0 Part 1: Core Language*. http://www.w3.org/TR/2004/WD-wsdl20-20040803/.

Wangham, M. S. (2004). *Esquema de Segurança para Agentes Móveis em Sistemas Abertos*. Tese (doutorado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.

Weerawarana, S., Curbera, F., Leymann, F., Storey, T., and Fergunson, D. F. (2005). *Web Service Platform Architecture*.

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

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

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

(WS-I), T. W. S.-I. O. and of its Members, C. (2004). *Basic Profile Version 1.1*. WS-I. http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html.

(WS-I), T. W. S.-I. O. and of its Members, C. (2005). *Basic Security Profile Version 1.0*. WS-I. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2005-08-29.html.

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

WS-PolicyAssertion (2003). *Web Services Policy Assertions Language (WS-PolicyAssertions)*. http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-policyassertions.asp.

WS-PolicyAttachment (2003). *Web Services Policy Attachment (WS-PolicyAttachment)*. http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-policyattachment1202.asp.

WS-SecurityPolicy (2003). *Web Services Security Policy Language(WS-SecurityPolicy)*. http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-securitypolicy.pdf.

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

Wu, Z. and Weaver, A. C. (2005). Dynamic trust establishment with privacy protection for web services. Proc. *ICWS*, pp. 811–812. IEEE Computer Society.

XML Schema (2001). XML schema part 1: Structures, W3C recommendation.

Yavatkar, R., Pendarakis, D., and Guerin, R. (2000). *A Framework for Policy-based Admission Control*. Internet Engineering Task Force RFC 2753.

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