JOSIANE MILANEZ

MODELO DE GERÊNCIA PARA O SPKI ATRAVÉS DO XKMS

FLORIANÓPOLIS
2005

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

MODELO DE GERÊNCIA PARA O SPKI ATRAVÉS DO XKMS

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

JOSIANE MILANEZ

Florianópolis, Outubro de 2005.

MODELO DE GERÊNCIA PARA O SPKI ATRAVÉS DO XKMS

Josiane Milanez

'Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica, Área de Concentração em *Controle, Automação e Informática Industrial*, 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.

Carla Merkle Westphal, Dr.

Ricardo José Rabelo, Dr.

Para Odete Steiner Milanez, minha mãe.....

AGRADECIMENTOS

Agradeço à Deus, meus pais, pela vida.

Agradeço meu orientador Joni e minha co-orientadora Michelle, pela enorme ajuda.

Agradeço ao Emerson Mello a ajuda na definição do algoritmo de buscas de certificados.

Agradeço também aos bolsistas Laura Carrijo e Rafael Deitos e a todos que tornaram possível a realização deste mestrado.

Agradeço principalmente aos familiares, meus amigos e amigas pela paciência comigo durante este período.

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

MODELO DE GERÊNCIA PARA O SPKI ATRAVÉS DO XKMS

Josiane Milanez

Outubro/2005

Orientador: Joni da Silva Fraga, Dr.
Área de Concentração: Controle, Automação e Informática Industrial
Palavras-chave: Federações SPKI, XKMS, Segurança Computacional, Sistemas Distribuídos
Número de Páginas: xi + 67

O propósito do *XML Key Management Specification* (XKMS) é facilitar o gerenciamento da PKI, transferindo a complexidade da mesma para um serviço *web* de confiança. Os serviços *web* formam uma tecnologia emergente e promissora para a automatização de interações inter-organizacionais. Estes serviços, fornecem um nível de abstração para diferentes plataformas e linguagens de programação, permitindo que sistemas de organizações diferentes se comuniquem de forma aberta, através de padrões *de facto* como o XML e o HTTP. Entretanto, o XKMS está fortemente focado na PKI X.509 que define um modelo hierárquico de confiança baseado na nomenclatura do X.500. Estes modelos apresentam efeitos negativos devido a esta centralização como a escalabilidade limitada e a falta de flexibilidade, indispensáveis em ambientes distribuídos de larga escala. O SPKI, se mostra mais adequado a estes sistemas comparado ao X.509. Esta PKI está baseada em uma estrutura de nomes locais e um modelo simples de autorização, baseado em redes de confiança. Este trabalho propõe um modelo de gerência para o SPKI através do XKMS. São apresentadas as principais facilidades providas pelo modelo proposto, como auxiliar na localização de certificados de autorização para construir os caminhos ligando o cliente ao servidor. Um algoritmo é proposto para a localização destes certificados e um protótipo é implementado e integrado a uma aplicação de forma a validar o modelo proposto.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Internet provê uma poderosa infra-estrutura de comunicação, que com seu crescimento têm impulsionado a demanda por aplicações distribuídas. Entretanto, a interação entre estas aplicações advindas de organizações diferentes torna-se uma tarefa difícil, já que cada organização utiliza os mais diferentes tipos de *hardware* bem como são usadas as mais diversas linguagens de desenvolvimento.

A arquitetura orientada a serviços (AOS) concentra-se no fraco acoplamento e ligação dinâmica entre serviços, onde estes podem ser: aplicativos, informações e outros recursos de TI (Weerawarana et al., 2005). Através deste fraco acoplamento e ligação dinâmica de serviços, a AOS permite uma rápida integração entre serviços podendo misturá-los e combiná-los de diferentes formas, criando assim flexíveis processos de negócios. Como benefícios desta arquitetura destacam-se: (1) uma diminuição de tempos de ciclo de desenvolvimento e implementação utilizando serviços reutilizáveis, (2) facilidades de integração entre estes serviços, (3) automatização de transações usuais (como localização de serviços, por exemplo), além de (4) facilidades nas interações inter-organizacionais.

Os serviços *Web*, uma tecnologia emergente e promissora para a automatização destas interações inter-organizacionais baseada na AOS, fornece um nível de abstração para diferentes plataformas e linguagens de programação. Estes serviços também permitem que sistemas de organizações diferentes se comuniquem de forma aberta, através de padrões *de facto* como o XML e o HTTP.

Vários padrões de segurança como *XML Signature*(Bartel, 2002), *XML Encryption*(Imamura, 2002), SAML(Cantor et al., 2005), entre outros, são utilizados para prover segurança ao XML e aos serviços *Web*. Estes padrões estão fundamentados em PKIs na manipulação de chaves públicas e emissão de certificados. Dentre estas PKIs podem ser citadas: X.509, PGP e SPKI.

1.1 Motivação

O propósito do *XML Key Management Specification* (XKMS) é facilitar o gerenciamento de PKIs, abstraindo a sua complexidade dos usuários. Para isso, a especificação XKMS define um serviço *Web* de confiança, responsável pelo gerenciamento da PKI. Sendo assim, a complexidade da PKI é

transferida da aplicação para um serviço *Web*, poupando a mesma da necessidade de gerenciamento da PKI utilizada. As informações e operações envolvendo a PKI ficam acessíveis através de um protocolo padrão, independendo da tecnologia de segurança usada.

O XKMS está fortemente focado na PKI X.509 que define um modelo hierárquico e centralizado de confiança baseado na nomenclatura do X.500. Entretanto, estes modelos apresentam efeitos negativos devidos a esta centralização. As dificuldades nestes modelos, normalmente citadas, são a escalabilidade limitada e a falta de flexibilidade, indispensáveis em ambientes distribuídos de larga escala.

Já os modelos de autenticação e autorização distribuída não apresentam este problema. A SPKI/SDSI (*Simple Public Key Infrastructure/Simple Distributed Security Infrastructure*) (Rivest e Lampson, 1996) (Ellison et al., 1999) é uma infra-estrutura de segurança baseada em chaves públicas que segue esta abordagem e teve seu desenvolvimento motivado devido a complexidade e imperfeição das infra-estruturas de chaves públicas baseadas em uma hierarquia de nomes globais, como o X.509. O SPKI/SDSI tem um modelo igualitário, no qual, os principais são chaves públicas e cada chave pública pode divulgar e assinar certificados (Clarke, 2001). Como o SPKI trabalha com uma estrutura de nomes locais, sua implementação torna-se mais simples, quando comparado ao X.509. Entretanto, o esquema de autorização do SPKI impõe restrições à localização de um determinado direito porque não oferece nenhum mecanismo através do qual seja possível buscar principais detentores de certificados com o direito de acesso desejado (Santin, 2004).

Em (Santin, 2004) foi proposta uma extensão do modelo de confiança do SPKI/SDSI chamada Federação SPKI, que é uma entidade que reúne principais com interesses afins e atua como um agente facilitador na localização de certificados e principais, pois permite o compartilhamento do acesso ao seu repositório de certificados. Através deste compartilhamento, os clientes passam a ter uma alternativa a recorrer quando da falta de cadeias apropriadas para o acesso desejado.

Porém, este modelo de gerência ao preservar a filosofia adotada no modelo SPKI/SDSI, no qual o principal que deseja obter acesso a algum recurso é inteiramente responsável pela busca das cadeias de certificados que lhe forneçam o direito de acesso, sobrecarrega o cliente. Além disso, este modelo não provê um protocolo padrão de acesso as funções de gerenciamento e estabelecimento de confiança providos pelo mesmo.

1.2 Objetivos

O objetivo deste trabalho é propor uma extensão ao modelo de Federações SPKI proposta em (Santin, 2004) através de serviços XKMS. O modelo aqui proposto oferece um esquema alternativo para a criação de cadeias de autorização ligando um cliente a um servidor de forma padronizada. Neste modelo, a responsabilidade pela localização de cadeias de autorização é transferida para o serviço XKMS, que também é o responsável pelo gerenciamento de confiança entre os membros e associados da Federação.

Para que este objetivo seja atingido é necessária uma integração do SPKI com o XKMS, o que não é uma tarefa fácil, já que o XKMS é fortemente focado no X.509. Também é necessária a criação de um novo algoritmo para a localização de cadeias de autorização que liga um cliente a um servidor.

Pretende-se também comprovar a aplicabilidade deste modelo em ambientes distribuídos de larga escala. A partir das perspectivas citadas, o projeto de pesquisa perseguiu os seguintes objetivos específicos:

- Definir um modelo de gerenciamento federado para o SPKI.
- Propor um algoritmo flexível para a localização de cadeias de autorização.
- Definir e implementar um protótipo que inclui o modelo de gerenciamento federado e o algoritmo de localização.
- De forma a comprovar a aplicabilidade deste modelo, integrar o protótipo a uma aplicação distribuída.

1.3 Organização do Texto

Este capítulo descreveu o contexto geral, a motivação e os objetivos da dissertação, sendo que os próximos capítulos encontram-se assim divididos:

O capítulo 2 apresenta os fundamentos sobre segurança computacional, dentre estes estão as políticas e os mecanismos de segurança. Por fim, os modelos apresentados são comparados.

O capítulo 3 descreve as especificações de segurança para a *Extensible Markup Language* (XML), tais como: *XML Signature*, *XML Encryption*, SAML, XACML e XKMS. Além das especificações de segurança em XML, serão descritos os conceitos de serviços *Web* e as especificações para segurança destes serviços, tais como: *WS-Security*, *WS-Policy*, *WS-Trust*, *WS-SecureConversation* e *WS-Federation*.

O capítulo 4 apresenta um modelo proposto de gerenciamento para o SPKI através do XKMS. Este modelo é uma extensão das Federação SPKI. Também será apresentado como a Teia de Federações é formada e o algoritmo de localização de certificados pelas Teias de Federações através do XKMS.

O Capítulo 5 apresenta uma descrição detalhada do protótipo implementado, descrevendo as ferramentas utilizadas, os cenários de aplicação e as considerações sobre os resultados encontrados.

No capítulo 6 são feitas conclusão sobre o texto e também serão expostos os trabalhos futuros.

Capítulo 2

Segurança em Sistemas Distribuídos

2.1 Introdução

A evolução da informática e das redes de computadores tem aumentado o uso de sistemas computacionais nas organizações e, em geral na vida das pessoas. O barateamento de alguns recursos como *hardware* e meios de comunicação têm contribuído para expandir o desenvolvimento e utilização de sistemas distribuídos.

Juntamente com este aumento, surge a preocupação com a segurança destes sistemas para garantir, por exemplo, que não haja interferência indevida entre o envio e o recebimento das mensagens, ou mesmo reservando recursos apenas à entidades autorizadas. Uma entidade, também chamada de **principal**, pode ser um usuário, um processo ou ainda uma máquina em uma rede de computadores.

Neste capítulo serão discutidos alguns aspectos da segurança computacional em sistemas distribuídos. Serão revisados os conceitos fundamentais, as políticas de segurança, as técnicas nas quais os mecanismos de segurança estão baseados e as abordagens de implementação dos mecanismos de autenticação e de autorização.

2.2 Segurança Computacional

Um sistema computacional é seguro se atende às propriedades básicas de segurança. Conforme a literatura, estas propriedades de segurança são (Landwehr, 2001):

- **Confidencialidade:** assegura que as informações não serão reveladas a pessoas que não tenham acesso autorizado.
- **Integridade:** assegura que as informações não serão modificadas por pessoas sem acesso autorizado.
- **Disponibilidade:** assegura que as informações estarão sempre disponíveis à usuários legítimos.

A violação das propriedades de confidencialidade, integridade e disponibilidade são chamadas de revelação não-autorizada ou vazamento de informação, modificação não autorizada e negação de serviço, respectivamente.

Outras propriedades podem ser adicionadas a estas citadas (Landwehr, 2001):

- **Autenticidade:** assegura que cada principal é realmente quem diz ser.
- **Não repudiação:** assegura que um emissor ou receptor de uma comunicação não possa negá-la posteriormente.

2.3 Ameaças, Ataques e Vulnerabilidades a Sistemas Distribuídos

Uma vulnerabilidade é um erro em um sistema de computadores (Landwehr, 2001), sendo este um erro de programação, configuração ou ainda de operação. Uma ameaça pode ser definida como um conjunto de ações que forneçam potencial violação das propriedades de segurança (Bishop, 2003). Ou seja, é uma intenção de causar danos em um sistema, que podem comprometer a confidencialidade, integridade e disponibilidade.

Quando uma violação da segurança ocorre através de uma ação, esta ação é chamada de ataque. O ataque consiste na exploração de alguma vulnerabilidade do sistema, que se bem sucedido, causa danos ao mesmo.

Ataques de segurança em uma rede ou sistema de computadores podem ser caracterizados analisando o fluxo de informação de uma fonte, como um arquivo ou uma região de memória principal, de uma máquina para um destino, que pode ser um outro arquivo ou usuário. Conforme pode ser verificado na Figura 2.1, existem quatro categorias gerais de ataques (Stallings, 2000):

- **Interrupção:** Um bem (*asset*) do sistema é destruído ou se torna indisponível ou inutilizável. Este é um ataque contra a disponibilidade. Este bem pode ser uma parte do *hardware*, o corte na linha de comunicação ou a desabilitação do gerenciamento do sistema.
- **Interceptação:** Um principal não autorizado obtém acesso a informação. Este é um ataque contra a confidencialidade.
- **Modificação:** Um principal não autorizado ganha acesso a um bem e o modifica. Este é um ataque contra a integridade. Exemplos incluem trocas de valores em um arquivo, alteração de um programa que passa a executar de forma diferente, modificando o conteúdo de mensagens transmitidas na rede.
- **Personificação:** Um principal não autorizado insere objetos falsificados no sistema. Este é um ataque contra a autenticidade.

Para um sistema ser denominado seguro é necessário corrigir os pontos vulneráveis, diminuindo assim, o risco de ataques. Entretanto, esta correção não é fácil, principalmente em sistemas mais complexos que tendem a apresentar maior número de vulnerabilidades.

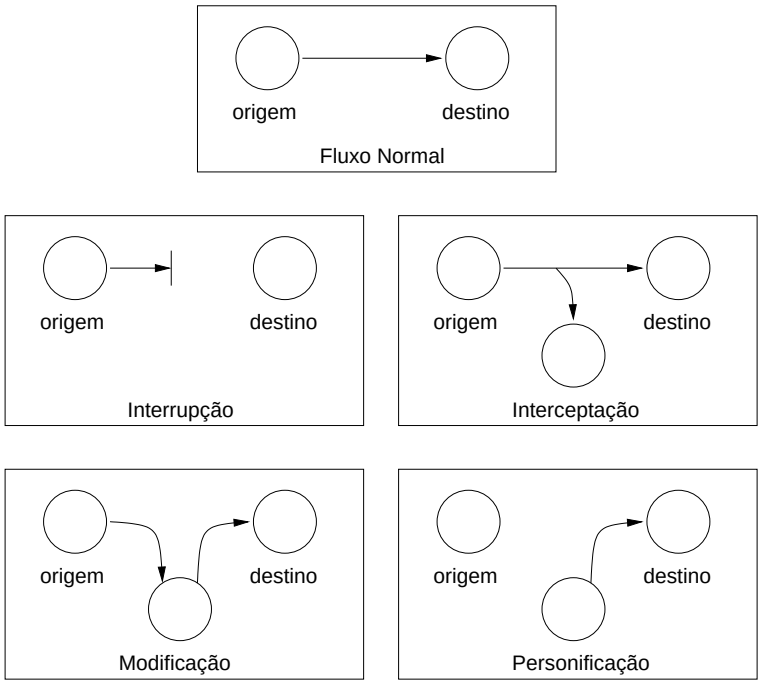

Figura 2.1: Ataques de segurança (Stallings, 2000)

2.4 Políticas de Segurança

Uma política é um simples conjunto de regras definidas para ir ao encontro de um objetivo principal, neste caso, a segurança de um computador ou da informação que este processa (Landwehr, 2001). Estas regras são específicas para cada sistema. As políticas de segurança de sistemas diferem em três ramos: segurança física, segurança administrativa e segurança lógica (Nicomette, 1996).

- **Política Física:** ocupa-se em proteger o meio físico em que opera o sistema. Nestas políticas são definidas medidas contra desastres, como por exemplo: incêndios, alagamentos, terremotos, entre outros. Também são definidas medidas para proteger o acesso físico ao provedor do sistema, fornecendo meios de proibir o acesso de pessoas não autorizadas. No caso de sistemas distribuídos, as políticas físicas não são muito eficazes, já que existem muitos pontos de acesso ao sistema.

- **Política Administrativa:** ocupa-se em proteger o sistema sob o ponto de vista organizacional. Define o modo através do qual, os responsáveis pela segurança dos sistemas informatizados são selecionados e os modos de como criar e manter as políticas de segurança.

- **Política Lógica:** ocupa-se em proteger os controles de acesso lógico ao sistema. Estes controles de acesso definem ”quem”tem acesso a ”que”e em ”quais”circunstâncias. Esta política pode ser decomposta em **políticas de autenticação**, em que o usuário necessita se identificar para obter o acesso ao recurso e **políticas de autorização** em que o usuário precisa provar que possui direitos sobre o recurso o qual ele deseja acessar.

2.5 Mecanismos de Segurança

Mecanismos de segurança são as implementações das políticas expressas pelos modelos de segurança. Para isso, os mecanismos fazem uso dos controles de acesso e controles criptográficos.

Em 1985, foi definido pelo Departamento de Defesa dos EUA (DoD) o conceito de Base Computacional de Segurança ou TCB (*Trust Computing Base*) como um conjunto que contém os elementos do sistema responsáveis pelo suporte às políticas de segurança (Department of Defense, 1985). Este conjunto inclui *hardware*, *software* e *firmware*[1] críticos. Um TCB é composto por um núcleo de segurança, que implementa o conceito de monitor de referência (responsável por aplicar o controle de acesso) e também pelos controles adicionais, como os controles criptográficos entre outros.

2.5.1 Autenticação e autorização

A autenticação consiste em um conjunto de procedimentos e mecanismos que permitem a um sistema computacional assegurar que a identificação de um principal esteja correta. Um exemplo de autenticação é a identificação de um principal através de um nome de usuário (*login*) e uma senha. Existem também outras maneiras de provar a identificação de um principal, como cartões magnéticos (*smartcards*), ou através do reconhecimento de características físicas, por exemplo: impressão digital ou íris.

Nos sistemas distribuídos, o serviço de autenticação preocupa-se também em assegurar a autenticação mútua dos parceiros da comunicação. Esta preocupação se faz necessária para que os indivíduos se assegurem da identidade dos envolvidos no processo da troca de informações. Além da autenticação mútua também existe a necessidade da autenticação dos dados, tornando possível provar a identidade da origem da mensagem.

A autorização é o processo que controla quais recursos e operações o principal autenticado terá permissão de acesso. Esses recursos incluem arquivos, bancos de dados, tabelas, entre outros. No caso de um sistema multi-usuário, o administrador do sistema define quais usuários terão permissão para acessar algum recurso do sistema. Sendo assim, somente os usuários com os devidos direitos especificados poderão acessar os recursos protegidos.

O monitor de referência consulta uma base de dados com as informações de políticas de autorização para determinar se a demanda de um principal está autorizada ou não no sistema. As políticas nesta base de dados são definidas e mantidas por um administrador de segurança. (Sandhu e Samarati, 1994).

2.5.2 Controles criptográficos

Os controles criptográficos são usados em vários níveis de um sistema distribuído, envolvendo a confidencialidade, a integridade, a autenticidade e o não repúdio nestes sistemas. Os algoritmos

[1] Programação em *hardware*; programa ou dados de computador que são armazenados permanentemente em um chip de memória de *hardware*, como uma ROM ou EPROM.

criptográficos usados podem ser simétricos ou assimétricos. Um algoritmo simétrico utiliza a mesma chave para cifrar e decifrar mensagens. Já o assimétrico utiliza uma chave pública e uma privada para cifrar e decifrar um mensagem, respectivamente.

A criptografia baseada em chave pública também pode ser usada no processo de assinatura digital, que permite garantir a autenticidade de quem envia a mensagem, associada à integridade do seu conteúdo. A assinatura digital é executada em duas etapas. Como primeiro passo, o emissor, através de um algoritmo gera um *hash* dos dados da mensagem que quer enviar. O *hash* é então cifrado com a chave privada do emissor, resultando a assinatura digital. Em seguida, o emissor envia a mensagem ao destinatário, com a assinatura digital e este, por meio da chave pública faz a decifragem e recalcula o *hash* da mensagem. Se os *hashes* forem iguais, a assinatura será autêntica. A não repudiação também é garantida pois o emissor da mensagem é o único com acesso à chave privada.

Com um sistema de chave pública, existe a necessidade de garantir a entrega da chave pública de forma segura. Por esta razão, um sistema de gerenciamento de chaves visa amenizar as preocupações de armazenamento e distribuição de chaves criptográficas. A forma de distribuição de chaves criptográficas mais empregada na atualidade são os certificados digitais, que vinculam chaves públicas a principais. Para garantir que a chave pública contida no certificado realmente pertence ao sujeito, é comum que o mesmo seja assinado por uma terceira parte confiável, chamada de autoridade certificadora (CA).

2.5.3 Controles de acesso

Os mecanismos de controle de acesso são usados para garantir que o acesso a um recurso seja limitado aos usuários devidamente autorizados. Formam a base para a implementação dos mecanismos de autorização. Os modelos de controle de acesso são geralmente classificados como (Sandhu e Samarati, 1994):

- **Controle de Acesso Discricionário (*Discretionary Access Control* - DAC):** O responsável pela informação, geralmente o proprietário, é quem manipula o direito de acesso a informação conforme a sua discrição.
- **Controle de Acesso Obrigatório (*Mandatory Access Control* - MAC):** A cada usuário e a cada objeto é atribuído um rótulo (*Security label*). Este rótulo associado com um objeto reflete a sensibilidade da informação contida no objeto, enquanto aos usuários refletem níveis de habilitação. Um controle obrigatório administra o acesso com base na habilitação dos sujeitos e na classificação dos objetos do sistema.
- **Controle de Acesso Baseado em Papéis (*Role-Based Access Control* - RBAC):** O acesso dos usuários à informação é regulamentado tendo como base os papéis. Estes papéis são atribuídos aos usuários conforme as atividades que estes executam no sistema. Os papéis podem ser definidos como um conjunto de ações e responsabilidades associadas com uma atividade de trabalho em particular (Sandhu e Samarati, 1994). Sendo assim, a autorização é dada aos

papéis, ao invés de especificar o que cada principal está autorizado a fazer, o que facilita o remanejamento de papéis.

2.6 Autenticação e Autorização em Sistemas Distribuídos

Em sistemas distribuídos, os recursos computacionais estão distribuídos em computadores ligados de forma dinâmica através de uma rede. Conforme a escala do sistema aumenta, maior é a dificuldade de implementar os mecanismos de autenticação e autorização. Um sistema é dito escalável se este puder tratar a adição de usuários e recursos sem sofrer uma perda notável de desempenho ou um aumento na complexidade de administração (Neuman, 1994).

A seguir serão apresentadas abordagens para implementação dos mecanismos de autenticação e autorização de forma escalável em sistemas distribuídos. Também serão apresentados alguns estudos de caso.

2.6.1 Abordagem Centralizada da Autenticação e da Autorização

Nesta abordagem, os serviços de autenticação e autorização são implementados de forma centralizada por uma única máquina no sistema distribuído, a Base Computacional de Segurança (TCB - *Trust Computing Base*).

A vantagem desta abordagem é a facilidade para a implantação da política de autorização. Porém, a centralização dos controles em um sistema distribuído provoca uma perda de desempenho já que todos os usuários que desejam utilizar os serviços de autenticação e autorização terão uma única máquina que executará este serviço.

Um outro problema é que esta máquina torna-se um ponto central de falhas, ou seja, caso uma falha ocorra, a segurança de todo o sistema é comprometida. Devido a estes problemas, esta abordagem não se torna atrativa para sistemas distribuídos e escaláveis.

2.6.2 Abordagem de Autenticação Centralizada e de Autorização Descentralizada

Diferindo da abordagem anterior, o sistema de autorização e de autenticação são separados, não sendo mais realizados na mesma máquina. Os controles de autenticação continuam centralizados, porém a autorização é realizada de forma local. Esta abordagem deixa o sistema menos vulnerável pois, apesar da autenticação ainda ser feita em uma única máquina, tornando-se um ponto central de falhas, a autorização é feita em máquinas diferentes. Entretanto observa-se que a descentralização da autorização pode provocar problemas de coerência na política de autorização.

A escalabilidade dos sistemas que seguem esta abordagem é obtida através do conceito de domínios: os vários domínios em sistemas de larga escala devem impor suas políticas de autenticação e de autorização aos seus respectivos membros.

Um servidor de um sistema de autenticação pode possuir relações de confiança com outros servidores, sendo que estas relações permitem a troca entre domínios nas certificações de autenticação e no controle de autorização. O modelo X.509 (ITU-T, 1993) e o Kerberos (Kohl e Neuman, 1993) são modelos baseados nesta abordagem.

2.6.3 Controle Descentralizado da Autenticação e da Autorização

Nesta abordagem, a autenticação e a autorização são descentralizadas. Ou seja, os principais se encarregam da implantação das políticas de autenticação e de autorização, sem que haja a necessidade de uma entidade confiável central, podendo criar relações de confiança sucessivas, formando redes de confiança (de Melo, 2003).

Apesar da dificuldade em se manter a coerência das políticas de autorização, não há mais o ponto central de falhas. Duas propostas, o TCSEC (Department of Defense, 1985) e o SPKI (*Simple Public Key Infrastructure*) (Ellison et al., 1999) baseiam-se nesta abordagem.

2.7 Estudo de Casos

2.7.1 Kerberos

Kerberos (Kohl e Neuman, 1993) é um serviço de autenticação centralizado, que provê autenticação entre aplicações clientes e servidoras. Este serviço teve seu modelo de autenticação baseado no protocolo de distribuição de chaves de Needham e Schroeder (Needham e Schroeder, 1978) e está enquadrado na abordagem de controle de autenticação centralizada e de autorização descentralizada.

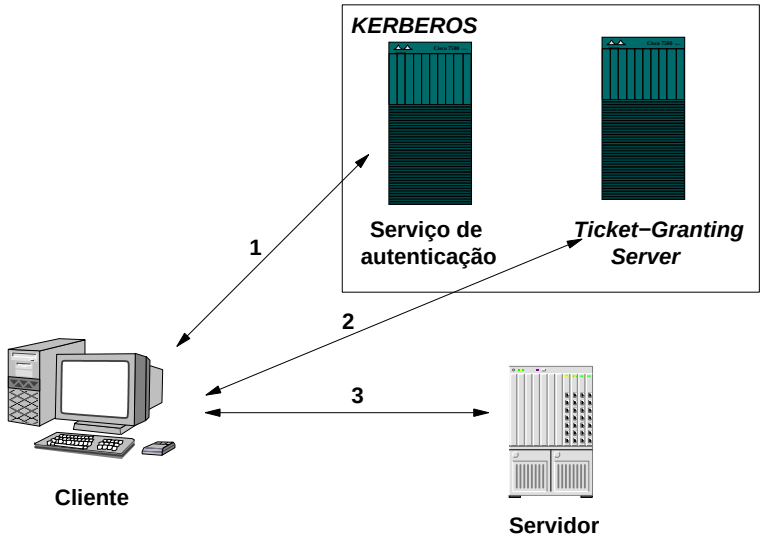

Figura 2.2: Kerberos

A Figura 2.2 ilustra, de forma simplificada, as trocas de mensagens no processo de autenticação. No passo 1, o cliente envia seu nome e o nome do servidor com o qual este deseja se comunicar e

solicita ao servidor de autenticação (AS) o *ticket* do servidor de tickets (TGS). O cliente de posse do *ticket*, fornece o mesmo ao TGS, que cria uma chave de sessão que será utilizada na comunicação entre o cliente e o servidor desejado (passo 2). No passo 3, o cliente envia ao servidor o *ticket* concedido pelo TGS.

O servidor Kerberos possui dependência no compartilhamento de chaves secretas com cada cliente pertencente ao domínio e com cada servidor de domínio com quem possui relacionamento de confiança.

A escalabilidade do Kerberos é obtida através do uso de domínios separados de autenticação chamado de *realms*. Cada domínio possui o seu próprio servidor Kerberos, e todos os clientes deste domínio confiam neste servidor. Estabelecendo relações de confiança entre servidores, através de compartilhamento de uma chave secreta, um cliente autenticado em um domínio pode utilizar esta autenticação em domínios diferentes. Para a unicidade de nomes, nomes locais são concatenados a nomes de domínio.

2.7.2 X.509

A recomendação ITU-T X.509 (ITU-T, 1993) define uma estrutura para certificados de chave pública e descreve dois níveis de autenticação. Um modo chamado de fraco, que usa conta e senha e outro chamado de forte, que envolve credenciais consolidadas por técnicas criptográficas.

O nível forte de autenticação define uma estrutura para o fornecimento de serviços de autenticação controlado de forma centralizada representado por um serviço de diretório X.500. O X.500, através de uma hierarquia de nomes, permite uma nomeação global e geograficamente distribuída.

Como no modelo Kerberos, o X.509 define uma autorização descentralizada, ou seja, após a autenticação o cliente fica sujeito ao controle de acesso do servidor de aplicação. Além disso, tem como base uma infra-estrutura de chave pública ou PKI (Public Key Infrastructure). Cada cliente possui, em particular, um par de chaves: uma chave privada que fica em seu poder, e outra, chave pública, que fica armazenada em diretórios X.500 na forma de certificados emitidos pela CA do domínio.

O modelo de confiança X.509 é hierárquico e as comunidades X.509 são construídas de forma *top-down* com base na confiança das chaves privadas das CAs. Este modelo baseia-se em certificados providos com a cadeia de autenticação, partindo da chave privada de uma CA confiável até a chave pública do usuário (Clarke, 2001). Através destes certificados é possível vincular nomes globais a chaves públicas.

Os certificados permitem uma associação entre um nome chamado identificador de nome único, ou DN para um usuário e a sua chave pública. É interessante notar, entretanto, que um mesmo usuário pode ter diferentes DNs em diferentes CAs, ou usar o mesmo DN em diferentes CAs, mesmo se este não é o primeiro a utilizá-lo. Desta forma, diferentes DNs em diferentes CAs não necessariamente significam diferentes usuários (Gerck, 2000). A Figura 2.3 apresenta o conjunto padrão de campos

de um certificado X.509. No último campo encontra-se a assinatura dos campos anteriores utilizando a chave privada da CA. Na seqüência segue a descrição de cada campo:

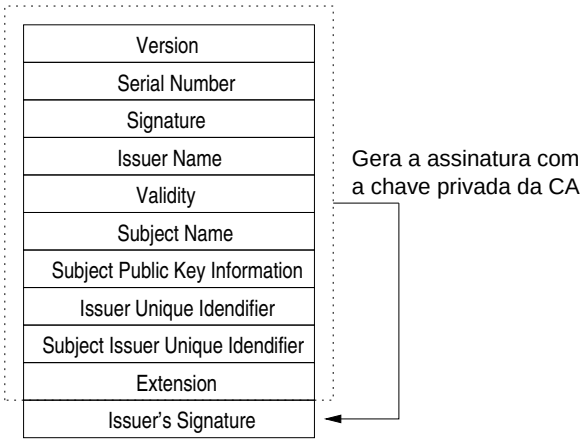

Figura 2.3: Certificado X.509 versão 3

- **Versão (*Version*):** identifica qual a versão do X.509 usada no certificado em questão, o que afeta qual informação pode ser especificada neste.
- **Número de Série do Certificado (*Serial Number*):** valor inteiro único atribuído pela CA a cada certificado.
- **Assinatura (*Signature*):** identificador do algoritmo usado para assinatura.
- **Nome do Emissor (*Issuer Name*):** o nome único da entidade que assinou o certificado.
- **Validade (*Validity*):** período de validade do certificado.
- **Nome do Sujeito do Certificado (*Subject Name*):** Identificação do principal associado à chave pública.
- **Informação da Chave Pública do Sujeito do Certificado (*Subject Public Key Information*):** a chave pública do principal, parâmetros opcionais e o identificador do algortimo.
- **Identificador único do Emissor do Certificado (*Issuer Unique Identifier*):** o nome da CA que publicou o certificado.
- **Identificador Único do Sujeito do Certificado (*Subject Unique Identifier*):** campo idealizado para ser único em toda a Interntet; este é composto de várias partes, como por exemplo: DN=JosianeM CN=Josiane, OU=DAS, O=UFSC, Inc., C=BR.
- **Extensão (*Extension*):** Proporciona um meio para associar dados adicionais para informações pessoais, chaves públicas e gerência de chaves, entre outras.
- **Assinatura do Emissor (*Issuer's Signature*):** a assinatura usando a chave privada da CA.

O modelo X.509 também permite construir relações de confiança através de certificação cruzada entre duas CAs, dispensando assim a necessidade de percorrer toda a estrutura hierárquica para ir de uma CA até outra. Esta alternativa só pode ser adotada entre CAs, o que evidentemente não descaracteriza a CA como entidade centralizadora da certificação.

O conceito de Listas de Revogação de Certificados (CLR - *Certificate Revocation Lists*), que invalidam certificados antes de sua data de expiração, também são descritas no padrão X.509. Esta característica pode ser interessante em caso de comprometimento da chave, ou em casos em que a informação sobre o sujeito do certificado é invalidada. Um certificado X.509 só pode ser revogado por uma CA. As CLRs, constantemente emitidas, datadas e assinadas digitalmente, informam quando a próxima CRL será emitida. Infelizmente CRLs não resolvem os problemas que lhes foram designadas, pois pode haver um atraso considerável e variável entre a CA que deseja notificar a revogação de algum certificado e a a reflexão desta necessidade nos clientes e nos servidores. Além disso, a maior aplicação de segurança X.509 atual, SSL/TLS (Freier et al., 1996a) não verifica as CRLs (Gerck, 2000).

Um dos maiores problemas com o X.509 é seu nível de dificuldade para ser entendido e adotado. Os formatos dos dados não são humanamente legíveis e são expressos em uma notação de sintaxe abstrata (ASN.1). Esta notação, que é um padrão de interconexão de sistemas abertos (OSI), é muito poderosa, porém muito complexa para ser usada (Clarke, 2001).

2.7.3 SPKI/SDSI

O SPKI/SDSI, conforme citado anteriormente, segue a abordagem descentralizada. Teve seu desenvolvimento motivado devido a complexidade e imperfeição das infra-estruturas de chaves públicas baseadas em uma hierarquia de nomes globais, como o X.509. O SPKI/SDSI surgiu da união das propostas SDSI (*Simple Distributed Security Infrastructure*) (Rivest e Lampson, 1996) e SPKI (*Simple Public Key Infrastructure*) (Ellison et al., 1999).

O SDSI, projetado no MIT por Rivest e Butler Lampson (Rivest e Lampson, 1996), é uma infra-estrutura de segurança com o objetivo principal de facilitar a construção de sistemas distribuídos seguros e escaláveis. A proposta SPKI, criada por Carl Ellison e outros (Ellison et al., 1999) é um padrão IETF que surgiu com o intuito de ser um modelo de autorização flexível, simples e de fácil implementação.

O SPKI/SDSI está baseado em um modelo igualitário. Os principais são chaves públicas e cada chave pública pode funcionar similarmente a uma entidade certificadora (Clarke, 2001). Não há uma entidade centralizadora para o registro de chaves públicas e emissão de certificados como a autoridade certificadora do X.509. Não existe uma infra-estrutura global hierárquica, ou seja, as comunidades podem ser construídas de forma distribuída sem a necessidade de uma entidade raíz (na qual todos confiam).

O SPKI/SDSI foi projetado para ser simples de entender, adotar e usar (Clarke, 2001). Por estes motivos foi concebido tendo como base uma adaptação da linguagem *S-expressions* (sintaxe abstrata

utilizada para representar dados) proposta em (Rivest e Lampson, 1996). Estas *S-expressions* são como estruturas LISP que encapsulam elementos com parênteses, cujos elementos podem ser cadeias de caracteres ou outras *S-expressions*. Ou seja, listas vazias não são permitidas e o primeiro elemento deve ser uma *string*.

Existem dois tipos de certificados no SDSI/SPKI: certificados de nomes e certificados de autorização (Clarke, 2001). Um certificado de nome define um nome local no espaço de nomes do emissor do certificado e é formado pelos seguintes campos:

- **Emissor (*Issuer*):** chave pública que assina o certificado.
- **Identificador (*identifier*):** identificador que determina o nome local que está sendo definido.
- **Sujeito (*subject*):** uma chave pública ou um nome, que identifique uma entidade definida em outro espaço de nomes e que está sendo redefinida no espaço de nomes do emissor.
- **Data de validade (*Validity Specification*):** período de tempo em que o certificado é válido.

Um certificado de nomes define um nome local e pode ser expresso da seguinte forma:

$$\text{K ”nome”} \longrightarrow \text{S}.$$

Onde K é a chave do emissor, ”nome”é o nome dado a chave no espaço de nomes local e S representa o sujeito que está sendo ligado ao ”nome”. Por exemplo, Bárbara detentora da chave K_B, quer emitir um certificado de nomes para sua mãe Carmen, detentora da chave K_C. Então Bárbara cria no seu espaço de nomes um nome local (mãe), ou seja, emite um certificado de nome para a chave pública de Carmem:

$$K_B\text{”mãe”} \longrightarrow K_C.$$

Débora, detentora da chave K_D deseja definir um nome para a chave pública da mãe de Bárbara. Para isso, Débora associa um nome (BárbaraMãe) ao certificado emitido por Bárbara:

$$K_D\text{”BárbaraMãe”} \longrightarrow K_B\text{”mãe”},$$

formando assim uma cadeia de certificados de nome. Se uma cadeia de certificados de nome for reduzida chega-se a uma chave pública, neste caso, a chave pública de Carmen.

No SPKI/SDSI é possível definir grupos de principais, no qual cada grupo possui um nome que é identificado com um conjunto de membros. O nome é local ao emissor de certificados de nome de grupo, que é o responsável pelas alterações nas definições do mesmo. Para definir um grupo, o emissor emite um certificado de nomes definindo o nome local do grupo no seu espaço de nomes. O sujeito deste certificado pode ser a chave de um membro ou um nome (Clarke, 2001).

Um certificado de autorização no SPKI/SDSI concede uma autorização específica dada pelo emissor ao sujeito do certificado e é formado pelos seguintes campos:

- **Emissor (*Issuer*):** chave pública que assina o certificado; ou seja, o principal que garante a autorização.
- **Sujeito (*subject*):** uma chave ou um grupo que recebe a autorização.
- **Bit de delegação (*Delegation bit*):** Valor lógico que indica se o certificado pode ou não ser delegado.
- **Autorização (*tag*):** especifica a autorização que será concedida ao sujeito pelo emissor.
- **Data de validade (*Validity Specification*):** período de tempo em que o certificado é válido.

O emissor destes certificados corresponde a um principal que, através dos mesmos, concede permissões de acesso a outros principais no sistema. O emissor de um certificado pode permitir que o sujeito (principal receptor) delegue as permissões recebidas a outros principais, através do *bit* de delegação.

Este sujeito, no papel de emissor, pode delegar estes direitos recebidos a outros principais, construindo assim, uma cadeia de autorização. Esta cadeia é enviada ao sujeito deste novo certificado.

Como a autorização baseada em redes de confiança 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, o sujeito do novo certificado deverá sempre apresentar a cadeia recebida quando quiser acessar o serviço.

Uma Lista de Controle de Acesso ou ACL SPKI/SDSI consiste de uma lista de entradas, onde cada entrada é considerada um certificado de autorização com o emissor sendo o proprietário da mesma. Os campos requeridos para cada entrada na ACL são: um sujeito, que pode ser um nome ou uma chave; uma tag; e o bit de delegação, que são os mesmos campos de um certificado de autorização (Clarke, 2001).

O SPKI/SDSI provê um mecanismo de tolerância a falhas através dos *threshold subjects* (Aura, 1998), que podem ser usados somente como certificados de autorização. São utilizados para especificar K de n sujeitos (sendo estes uma chave pública, um nome ou ainda um grupo), que devem assinar uma requisição ou uma delegação para que esta seja honrada.

O modelo SPKI/SDSI apresenta, como dificuldade, a identificação das cadeias de certificados que levem um cliente a obter autorização necessária para acessar os recursos do servidor desejado, podendo haver casos em que o cliente não possua caminho de confiança que o autorize diante do servidor. Para solucionar este problema, existem diversos trabalhos que tratam da procura de cadeias de certificados que viabilizem acessos aos recursos desejados, dentre estes destacam-se a abordagem de Federações SPKI, proposta em (Santin, 2004).

2.7.4 Federações SPKI

Em (Santin, 2004) foi proposta uma extensão do modelo de confiança do SDSI/SPKI (que impõe restrições à localização de determinado direito) através da introdução da noção de Federação. A

federação é uma entidade que reúne principais com interesses afins e atua como um agente facilitador na localização de certificados e principais, pois permite o compartilhamento do acesso ao seu repositório de certificados. Através deste compartilhamento, os clientes passam a ter uma alternativa a recorrer quando da falta de cadeias apropriadas para o acesso desejado.

As principais funções de uma federação estão centradas em um gerente que provê mecanismos de armazenamento, recuperação e criação de cadeias de certificados de autorização, necessários para que os clientes consigam efetivar o acesso nos servidores, sem no entanto caracterizar-se como uma chave intermediária. Um principal, sendo um cliente ou um servidor, ao ingressar em uma federação fornece os certificados de nomes e de autorização delegáveis que julgar úteis para a federação.

Um principal qualquer pode filiar-se a quantas federações este queira. Sendo que para se registrar terá que fornecer um *threshold certicate*, assinado por k-de-n membros já filiados à Federação em questão, acompanhado do certificado de nome deste principal que está solicitando a admissão. Quando registrado, um certificado de nome é incluído no repositório da federação e será mantido pelo gerente de certificados para auxiliar no processo de identificação deste principal. Para cada novo membro incluído na federação é também emitido um ”certificado de grupo”(certificado de nome expressando participação no grupo SDSI) para fins de comprovação da filiação (*membership*).

A partir da integração da federação ao modelo de confiança do SDSI/SPKI, os clientes passam a ter uma alternativa a recorrer na falta de cadeias apropriadas para o acesso desejado (Santin, 2004). Na Figura 2.4 pode ser observado um cenário em que um *cliente A* recebe de um servidor de aplicação um certificado de autorização delegando um direito ao mesmo (passo 1). O *cliente A* armazena este certificado no seu repositório local e no repositório de certificados da Federação que pertence (passo 2). Desta forma, um *cliente B*, que não possui o direito de acesso, ao fazer a busca deste certificado na Federação, recebe o certificado emitido pelo servidor ao *cliente A* (passo 3). De posse deste certificado, o *cliente B* pode negociar o direito com o *cliente A* (passo 4). Obtido o direito através do certificado negociado, o *cliente B* pode fazer requisições de acesso ao Servidor de Aplicação (passo 5).

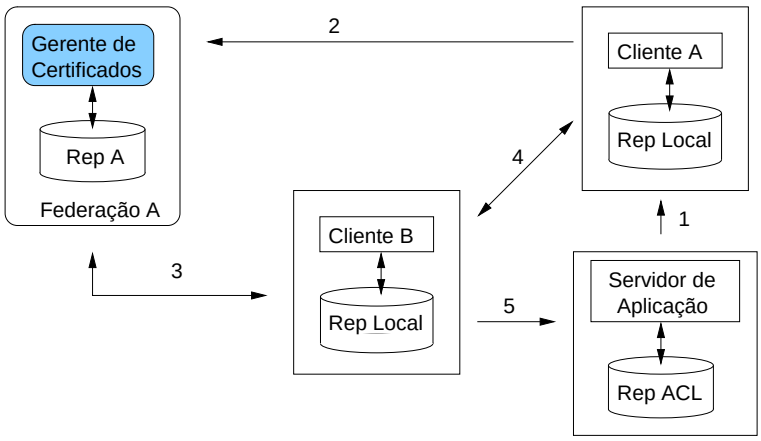

Figura 2.4: Modelo de confiança estendido do SPKI/SDSI

Apesar de facilitar a localização de certificados de autorização, uma federação geralmente tem

a sua abrangência limitada ao fim a que se presta. Para evitar que membros de uma federação necessitem filiar-se a várias outras federações para atingir um certo nível de presença na rede, surgiu a proposta que os gerentes de certificados das federações se associem formando teias de federações.

A teia de federações é formada arbitrariamente e cria caminhos de confiança entre as federações associadas de tal modo que um cliente pode fazer consultas a federações as quais não é filiado. Para poder realizar uma busca em uma outra federação, o gerente de certificados da federação de origem deve ser associado a federação onde será realizada a busca. Ao realizar uma busca em uma outra federação, o cliente deve apresentar à federação associada o certificado de membro da sua federação de origem e o certificado de associação entre as duas federações. Desta forma, a criação de novas cadeias de certificados pode ser realizada através da busca de certificados em uma teia de federações e da negociação da concessão do privilégio com o possuidor do mesmo.

Conforme pode ser visto na Figura 2.5, um principal da Federação A pode realizar a busca na Federação B, já que os gerentes das duas Federações são associados, ou seja, possuem uma relação de confiança. Desta forma, para que esta busca se realize, basta que o principal da Federação A apresente ao gerente da Federação B, o seu certificado de membro da Federação A.

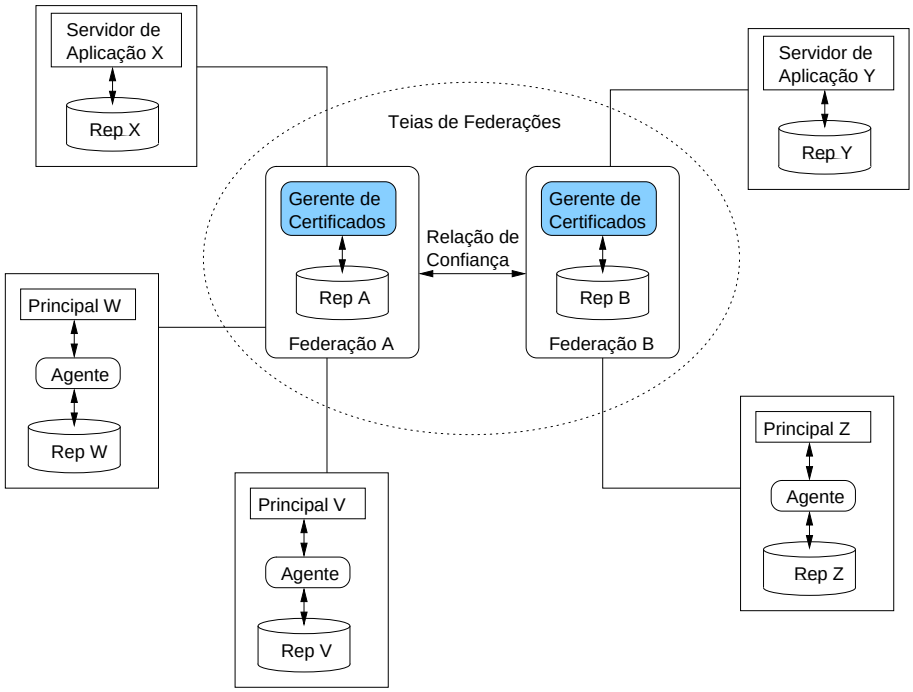

Figura 2.5: Teias de federações

A extensão do modelo de confiança do SDSI/SPKI proposta em (Santin, 2004), visa preservar a filosofia adotada no modelo SPKI/SDSI versão 2.0, na qual o principal que deseja obter acesso a algum recurso é inteiramente responsável pela busca das cadeias de certificados que lhe forneçam o direito de acesso ao recurso. Entretanto, esta filosofia sobrecarrega o cliente, já que é este quem realiza a busca por certificados na teia de federações.

2.8 Conclusão

Neste capítulo foram apresentados os conceitos fundamentais de segurança, juntamente com os conceitos de políticas e de mecanismos de segurança. Foram vistas as principais abordagens clássicas para a autenticação e a autorização em sistemas distribuídos.

Conforme pode ser observado no texto, modelos onde a relação de confiança pode ser estabelecida de maneira distribuída, como no SPKI/SDSI, a escalabilidade e a flexibilidade são favorecidas. Foi também apresentado o trabalho proposto em (Santin, 2004) que introduz o conceito de Federação que visa auxiliar o modelo de confiança do SPKI/SDSI, que apresenta como dificuldade a identificação das cadeias de certificados.

No capítulo 4, será apresentado um modelo também baseado em federações que possui um serviço que encapsula o gerente de certificados, além de realizar buscas para o cliente. Além desta facilidade para o cliente, o novo modelo também está baseado em serviços *Web*.

Capítulo 3

Segurança através do XML e Serviços Web

3.1 Introdução

No capítulo anterior foram expostas diferentes abordagens de segurança utilizadas em sistemas distribuídos, tendo como enfoque as características de autenticação e de autorização. Foi exposta uma alternativa para recorrer quando há a falta de cadeias apropriadas para o acesso desejado, construída sobre a infra-estrutura de chaves públicas chamada SPKI/SDSI proposta em (Santin, 2004).

Neste capítulo serão apresentadas as especificações de segurança para a *Extensible Markup Language* (XML) (Bray et al., 2004) ou, linguagem de marcação de dados, que provê um formato para descrever diversos tipos de dados. Estas especificações definem vocabulários em XML para representar informações seguras permitindo a aplicação de segurança em documentos XML, em elementos do documento, bem como no seu conteúdo total. Por serem interoperáveis e flexíveis, as especificações de segurança XML também facilitam o gerenciamento de direitos digitais e o controle de acesso distribuído. As especificações a serem descritas são: *XML Signature* (Bartel, 2002), *XML Encryption* (Imamura, 2002), *Security Assertion Markup Language* (SAML)(Cantor et al., 2005), *Extensible Access Control Markup Language* (XACML)(Moses, 2005) e *XML Key Management Specification* (XKMS)(Hallam-Baker, 2004).

Além das especificações de segurança em XML, serão apresentados os conceitos de serviços *Web* e as especificações para segurança destes serviços, tais como: *WS-Security* (Nadalin et al., 2004), *WS-Policy* (Anderson et al., 2005b), *WS-Trust*, *WS-SecureConversation* (Anderson et al., 2005a) e *WS-Federation* (Bajaj, 2003).

A XML, desenvolvida por um grupo de trabalho formado através do *World Wide Web Consortium* (W3C) em 1996, é um subconjunto do *Standard Generalized Markup Language* (SGML)[1] otimizado,

[1] O SGML é um padrão internacional (ISO 8879), portável, publicado em 1986, que descreve um padrão para o uso de marcações descritivas mescladas ao documento, permitindo a criação de documentos independentes do tipo de máquina e dos programas usados.

principalmente para a distribuição de dados através da *Web*.

Dentre os principais objetivos de projeto do padrão XML (Bray et al., 2004) estão o suporte a um grande número de aplicações e a usabilidade de documentos XML em diversos contextos (Bray et al., 2004). Para descrever os dados em um documento XML, um documento de definição de tipos (*Document Type Definition* - DTD) pode ser adotado. O DTD é um documento que especifica exatamente o contexto e a estrutura dos elementos permitidos em um documento XML. Como uma alternativa ao DTD, pode ser utilizado um esquema XML (XML Schema) (David C. Fallside, 2004), que é uma evolução do DTD. Uma das grandes vantagens do esquema XML sobre o DTD é o suporte a tipos de dados, que facilita a descrição do conteúdo de documentos XML permitindo assim a sua validação e a sua conversão, além da definição de restrições dos dados. Por ser definido através da própria linguagem XML, um esquema XML pode ser facilmente estendido, reutilizado, transformado através das ferramentas já definidas e implementadas para o XML.

3.2 XML Signature

A especificação *XML Signature* (XMLSig) (Bartel, 2002) é um padrão desenvolvido em conjunto pela W3C e IETF (RFC 2807 e 3275), que define regras para gerar e validar assinaturas digitais expressas em XML. As assinaturas digitais servem como prova persistente da integridade e da autenticidade de dados de um documento, assim como evidencia a origem de quem a criou (garante a não repudiação).

As assinaturas digitais somente funcionam se os cálculos de verificação são executados nos mesmos *bits* que os usados para o cálculo da assinatura. A especificação *XML Signature* utiliza mecanismos para representar documentos XML na forma canônica[2] (Boyer, 2001). A forma canônica assegura que duas instâncias de um mesmo texto produzam o mesmo resumo e a mesma assinatura, mesmo que as instâncias sejam um pouco diferentes. Por exemplo, uma destas instâncias possui um espaço em branco a mais que a outra, ainda assim, o resumo será o mesmo (Naedele, 2003). A forma canônica de um documento XML é uma representação física e invariante em respeito a todas as possíveis diferenças sintáticas que um mesmo documento pode sofrer, por exemplo, um espaço em branco, a ordem dos atributos, entre outros (O'Neill, 2003).

Não somente documentos XML podem ser assinados, como também outros tipos de dados. Os dados assinados podem estar dentro ou fora do documento XML que contém uma assinatura (arquivos binários ou textos). A especificação prevê ainda a possibilidade de assinar somente porções específicas da árvore XML, ao invés do documento completo. Isto é importante quando um único documento XML deve ser assinado múltiplas vezes por uma única ou múltiplas partes. Esta flexibilidade pode assegurar a integridade de certas porções de um documento XML, enquanto deixa aberta a possibilidade para que outras porções do documento possam ser modificadas.

Segundo a especificação, existem três tipos de assinatura XML (Bartel, 2002):

[2]Forma canônica é aquela que é geral e que as outras formas podem ser geradas da mesma por processos de redução. Documentos XML que sejam sintaticamente diferentes, porém logicamente equivalentes, podem ser assumidos como reduções de uma forma canônica e representados por esta última

- ***Enveloped***

  Quando a assinatura está contida no próprio documento assinado, esta é chamada de *Enveloped XML Signature*. Obviamente, com assinaturas *enveloped*, deve-se tomar cuidado para que não seja incluído o seu valor no cálculo do *SignatureValue*. Uma representação deste tipo de assinatura pode ser visto na Figura 3.1.

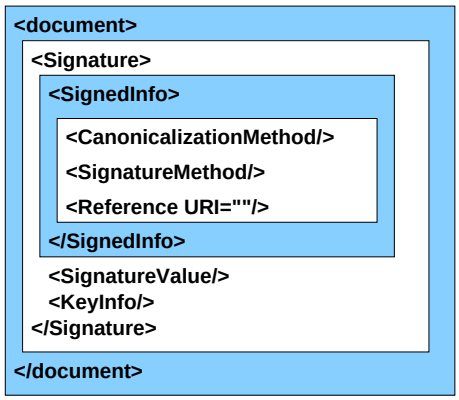

Figura 3.1: Enveloped XML Signature

- ***Enveloping XML Signature***

  Quando o dado assinado (em XML ou não) está contido na própria estrutura da assinatura XML, a assinatura é chamada de *Enveloping XML Signature*. Este dado é identificado via um elemento *Reference*, conforme ilustrado na Figura 3.2.

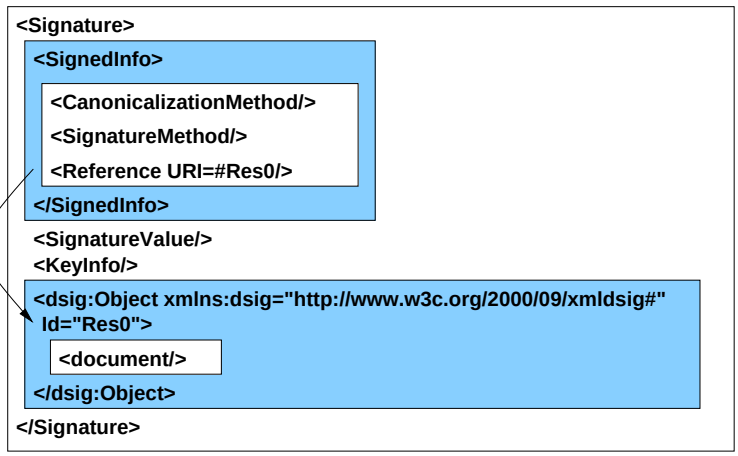

Figura 3.2: Enveloping XML Signature

- ***Detached Signature***

  Quando uma assinatura está separada dos dados que foram assinados, esta é chamada de *Detached Signature*, conforme apresentado na Figura 3.3. Este tipo de assinatura é geralmente aplicada em objetos, ou arquivos de dados separados de sua assinatura. Mas também podem incluir casos em que a assinatura e o dado assinado estão no mesmo documento XML, porém

em elementos separados. O dado assinado pode ser referenciado através de uma URI, podendo ser localizado na *Web* e não limitado aos dados disponíveis em arquivos locais.

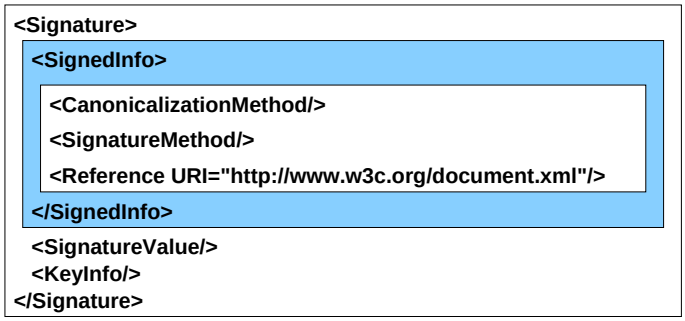

Figura 3.3: Detached Signature

3.3 XML Encryption

A especificação *XML Encryption* (Imamura, 2002), desenvolvida pela W3C, é um padrão que descreve o processo de cifragem e representação de dados cifrados em documentos XML. O conteúdo cifrado e o processamento adicional da informação são representados em XML, de modo que o resultado possa ser processado por outras ferramentas XML. A *XML Encryption* não pretende substituir os protocolos SSL/TLS (Freier et al., 1996b) (Dierks e Allen, 1999), mas sim prover um mecanismo de segurança fim-a-fim que oferece algumas características não cobertas pelo SSL/TLS, tais como: cifrar somente parte dos dados.

A especificação *XML Encryption* define uma sintaxe (definida através de um esquema XML) e regras de processamento para proteção da confidencialidade de dados. Estes dados podem ser, documentos XML como um todo, partes de um documento XML, ou ainda um dado qualquer. O resultado da cifragem é um elemento *EncryptedData*, que contém o dado cifrado.

3.3.1 Cenários de Cifragem

Cifragem XML pode ser aplicada em três casos amplos: cifragem de um elemento XML, cifragem do conteúdo de um elemento XML e cifragem de um dado qualquer (que pode ser XML) (O'Neill, 2003). Estes casos são exemplificados nos cenários descritos em *http://www.w3.org/TR/xmlenc-core*, que serão aqui analisados conforme apresentado na Figura 3.4. Nesta Figura, podem ser observados dados de uma pessoa representados em elementos XML. Dentre estes elementos estão o nome da pessoa e os dados de seu cartão de crédito.

3.3.1.1 Cifragem de um elemento XML

O número do cartão de crédito de Smith é uma informação sensível. Se a aplicação deseja conservar a confidencialidade da informação, esta pode cifrar o elemento *CreditCard* que pode ser visu-

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

Figura 3.4: Exemplo

alizado na Figura 3.4(Imamura, 2002) (Figura 3.5).

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
   xmlns='http://www.w3.org/2001/04/xmlenc#'>
    <CipherData>
      <CipherValue>A23B45C56</CipherValue>
    </CipherData>
  </EncryptedData>
</PaymentInfo>
```

Figura 3.5: Cifragem de um elemento XML

O elemento *CreditCard*, desde o seu início até seu fim, foi escondido no elemento *CipherData* que contém a cifragem do mesmo.

3.3.1.2 Cifragem do conteúdo de um elemento XML

Pode-se ainda considerar o cenário no qual toda informação exceto o número do cartão de crédito real deve estar disponível. Neste caso, somente o conteúdo do elemento *Number* é cifrado (Figura 3.6).

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>
      <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
       Type='http://www.w3.org/2001/04/xmlenc#Content'>
        <CipherData>
          <CipherValue>A23B45C56</CipherValue>
        </CipherData>
      </EncryptedData>
    </Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

Figura 3.6: Cifragem do conteúdo de um elemento XML

3.3.1.3 Cifragem de um dado qualquer

Se o cenário da aplicação requer que toda informação seja cifrada, todo documento é cifrado como uma seqüencia de octetos. Isto aplica-se a dados arbitrários, ou seja, qualquer tipo de dado, podendo ser ou não um documento XML (Figura 3.7).

```
<?xml version='1.0'?>
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
 MimeType='text/xml'>
  <CipherData>
    <CipherValue>A23B45C56</CipherValue>
  </CipherData>
</EncryptedData>
```

Figura 3.7: Cifragem de um dado qualquer

3.4 XACML

Devido ao alto custo de gerenciamento de políticas em grandes empresas (que definem estas políticas em diferentes pontos de efetivação e de forma não padronizada) surge a necessidade de uma linguagem comum para expressar as políticas de segurança. A *Extensible Access Control Markup Language* (XACML) (Moses, 2005) é uma linguagem que especifica esquemas para políticas de controle de acesso, além de um formato para requisições e respostas de decisões de autorização, garantindo assim, a interoperabilidade destas políticas. Através da XACML, uma empresa pode gerenciar a efetivação de todos os elementos de suas políticas de segurança em todos os componentes dos seus sistemas de informação (Moses, 2005). A XACML é usada para definir se o acesso ao recurso requisitado é permitido baseado em conjuntos de regras ou políticas (Damiani et al., 2002). A XACML define três componentes principais (Moses, 2005):

- **Regras (*Rule*):** contém expressões booleanas que podem ser acessadas através de um Ponto de Decisão de Política (PDP).
- **Política (*Policy*):** contém um conjunto de elementos de regras.
- **Conjunto de Políticas (*PolicySet*):** contém um conjunto de políticas (*policies*) ou conjuntos de políticas e um procedimento especificado pela combinação dos resultados de suas avaliações.

3.4.1 Regras em XACML

Uma regra é a unidade mais elementar de uma política. Possui três principais componentes: um *target*, um efeito (*effect*) e um conjunto de condições (*conditions*) (Moses, 2005). Um *target* é definido como o espaço de pedidos de decisão que referenciam ações em recursos (documentos, serviços ou sistemas) por sujeitos (pessoas ou computadores). Efeito é permitir ou negar. Para a efetivação

de uma regra, condições devem avaliadas para a aplicação do efeito a regra. Estas condições podem envolver cálculo de atributos de um sujeito, de um recurso de uma ação ou de um ambiente (O'Neill, 2003).

3.4.2 Políticas em XACML

Uma política compreende quatro principais componentes: *targets*, algoritmo de combinação de regras (*Rule-combining algorithm*), regras (*rules*) e obrigações (*obligations*) (Moses, 2005). Um *target* especifica um sujeito (*subject*), recursos e ações e ambientes aos quais a mesma se aplica. O algoritmo de combinação de regras especifica o procedimento através do qual os resultados das avaliações dos componentes das regras são combinados quando se avalia a política. Ou seja, o valor de decisão dado como resposta pelo Ponto de decisão de políticas (PDP) é o valor da política conforme definido no algoritmo de combinação de regras. Uma obrigação é definida como uma ação para ser executada uma vez que a decisão de autorização está completa. Quando um PDP avalia uma política que contém obrigações, este retorna estas obrigações ao Ponto de Efetivação de Políticas (PEP).

3.4.3 Fluxo de dados no modelo e XACML

Para definir se o acesso ao recurso requisitado é permitido, a XACML utiliza um contexto que é facilmente mapeado em requisições SAML (Damiani et al., 2002) (seção 3.5). O contexto XACML é definido em um esquema XML que descreve a representação para entradas e saídas do pondo de decisão de políticas (PDP). Na Figura 3.8, onde é mostrado o modelo de fluxo de dados do XACML, pode ser observado o tratador do contexto (*handler*) . O modelo realiza passos que podem ser observados na tabela 3.1 (Moses, 2005). Ao fim, se o acesso é permitido, o PEP libera o acesso ao recurso, caso contrário o acesso é negado.

3.5 SAML

A *Security Assertion Markup Language* (SAML) (Cantor et al., 2005), desenvolvida pelo comitê técnico de serviços de segurança da OASIS (*Organization for the Advancement of Structured Information Standards*), é um conjunto de especificações e esquemas XML para expressar informações de autenticação, direitos e atributos de usuários (Madsen, 2004). Visando garantir a interoperabilidade, o SAML provê uma plataforma neutra, abstraindo o *framework* de segurança de uma implementação ou arquitetura em particular.

A SAML 2 introduz um novo suporte ao mundo móvel além de mecanismos de privacidade (Madsen, 2004). Além disso, define como pseudônimos podem ser utilizados, estabelecidos e gerenciados entre provedores para representar principais (gerenciamento federado). Outra característica é o gerenciamento de sessão provido pelo protocolo SAML 2, através do qual as sessões providas por uma autoridade de sessão podem ser simultaneamente terminadas. O SAML também não exige que a

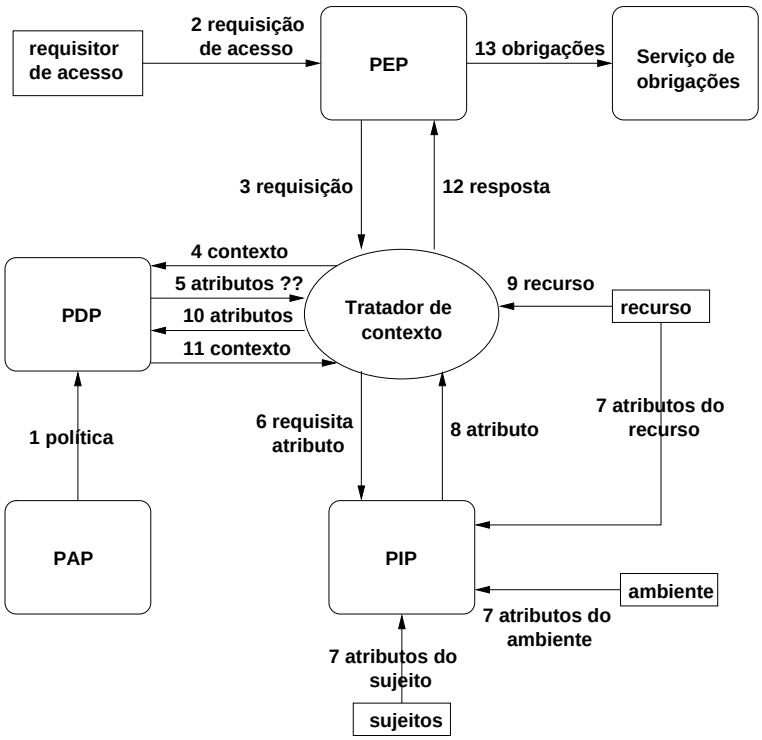

Figura 3.8: Modelo de Fluxo de dados (Moses, 2005)

informação do usuário seja gerenciada ou sincronizada em todos os diretórios das entidades participantes (acoplamento entre diretórios).

A maioria dos produtos atuais que provêem a propriedade *Single Sign On*[3], evitando a necessidade de uma nova autenticação, são normalmente implementados em formas proprietárias. Além disso, estes produtos utilizam *cookies* para armazenar informações do usuário, que não são transferidas entre diferentes domínios DNS.

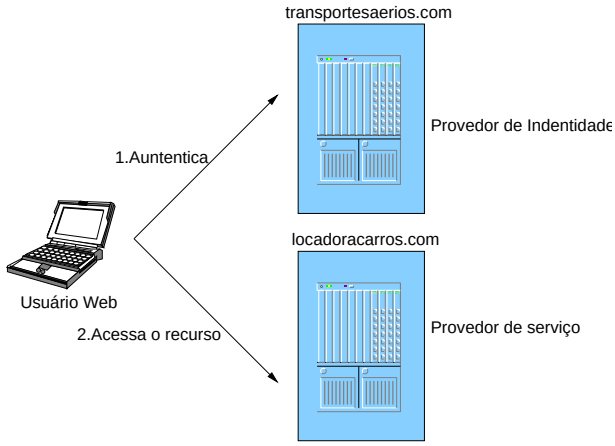

Figura 3.9: *Cenário SSO - Identidade Federada*

[3]Um usuário autentica-se uma única vez em um mesmo domínio e mesmo usando serviços ou servidores diferentes não necessita se autenticar novamente.

1	O Ponto de Administração de Políticas (PAP) define as políticas e as deixa disponíveis no Ponto de Decisão de Políticas (PDP).
2	O requisitor de acesso envia a requisição de acesso ao Ponto de Efetivação de Políticas (PEP).
3	O PEP envia a requisição de acesso ao tratador de contexto no seu formato de requisição nativo, opcionalmente incluindo os atributos do sujeito, do recurso, da ação e do ambiente.
4	O tratador de contexto constrói o contexto de uma requisição XACML e o envia ao PDP.
5	O PDP requisita os atributos adicionais necessários ao tratador de contexto.
6	O tratador de contexto requisita ao Ponto de Informações de Políticas (PIP) os atributos necessários.
7	O PIP obtém estes atributos requisitados.
8	O PIP retorna estes atributos ao tratador de contexto.
9	Opcionalmente o tratador de contexto adiciona o recurso ao contexto.
10	De posse dos atributos, o tratador de contexto os envia ao PDP.
11	O PDP retorna ao tratador de contexto o contexto de resposta que contém a decisão de autorização.
12	O tratador de contexto traduz o contexto de resposta para o formato de resposta nativo e envia o resultado desta tradução para o PEP.
13	O PEP cumpre as obrigações.

Tabela 3.1: Modelo de Fluxo de dados

As asserções de autenticação SAML possibilitam o *Single Sign On* através de uma autenticação em um provedor de identidade de forma que o acesso a serviços e recursos nos provedores de serviço, dispersos em vários domínios administrativos, sejam realizados sem que uma autenticação adicional seja necessária. Ou seja, a responsabilidade de gerenciamento de identidades é repassada para o provedor de identidade, já que este gerenciamento é mais compatível com o mesmo do que com o provedor de serviço, permitindo assim que usuários consolidem muitas identidades locais em uma única (ou pelo menos um conjunto reduzido de) identidade federada que atravesse domínios administrativos.

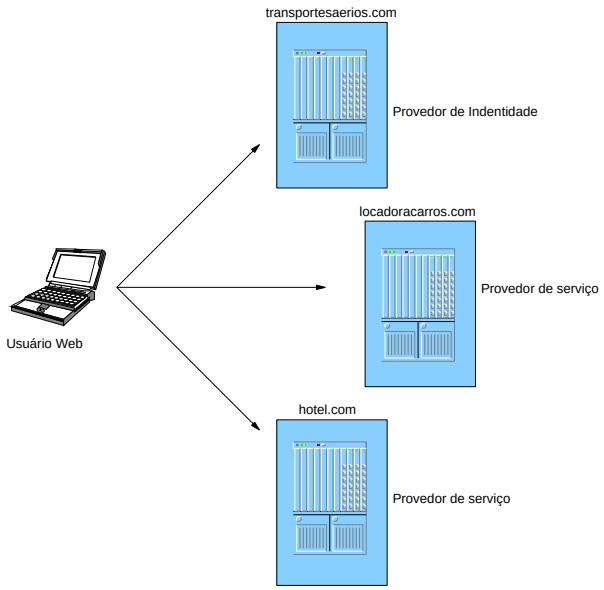

Figura 3.10: Ligação entre contas

O caso de uso original suportado pela SAML 1.0, ilustrado na Figura 3.9, é o caso de uso *sin-*

gle sign on[4], em que um usuário possui uma sessão de logon (um contexto de segurança) em um determinado *web site* e esta sessão é repassada de forma transparente ou explícita para outro *web site* em um domínio diferente (Madsen, 2004). Para isso o *web site* original ou provedor de identidade declara ao provedor de serviço que o usuário é conhecido e provê o nome ou os atributos de sessão do usuário. O provedor de serviço (locadoracarros.com) que confia no provedor de identidade (transportesaereos.com) cria uma sessão para o usuário com base no nome ou nos atributos de sessão repassados.

Um outro de caso de uso é a ligação entre contas, ilustrado na Figura 3.10. Neste caso de uso, o usuário é registrado em dois provedores de serviços diferentes (locadoracarros.com e hotel.com), porém com nomes diferentes. No locadoracarros.com, o usuário está registrado como deveria e no hotel.com como daniellav.

3.5.1 Arquitetura SAML

A SAML consiste de um conjunto de componentes que permitem a transferência e troca de informações de identidade, autenticação e autorização entre organizações autônomas. A especificação SAML define a estrutura e o conteúdo de asserções SAML, que transportam declarações sobre um principal. Estas asserções são um conjunto de declarações sobre características e atributos de uma entidade, dadas por uma autoridade SAML. São definidos três tipos de asserções SAML (Cantor et al., 2005):

- **Autenticação:** O emissor da asserção declara que o sujeito foi autenticado.
- **Atributo:** Declara informações específicas do sujeito (como por exemplo, limite de crédito para comércio eletrônico entre outros) e credenciais do mesmo.
- **Autorização:** Declarar se o sujeito pode ou está autorizado a acessar um determinado recurso, com base nas asserções de autenticação e de atributos.

Como estas asserções SAML são transportadas é definido nos protocolos SAML, que definem os pares de requisição e resposta para a obtenção das asserções SAML e o gerenciamento federado (Hughes e Maler, 2004). Os *bindings* definem como estes protocolos SAML são transportados sobre os protocolos de mensagens (HTTP ou SOAP) e a combinação entre os protocolos SAML juntamente com os *bindings* e as estruturas de asserções criam os perfis SAML que suportam os casos de uso. A Figura 3.11 ilustra como os componentes SAML se relacionam.

3.5.2 A SAML e outros padrões e iniciativas

A Liberty Alliance (*www.projectliberty.org/*) é um consórcio industrial de definições de padrões para identidade federada. O *ID-Web Services Framework*, uma plataforma de segurança para serviços

[4]Atravessam domínios administrativos diferentes através da autenticação única (SSO)

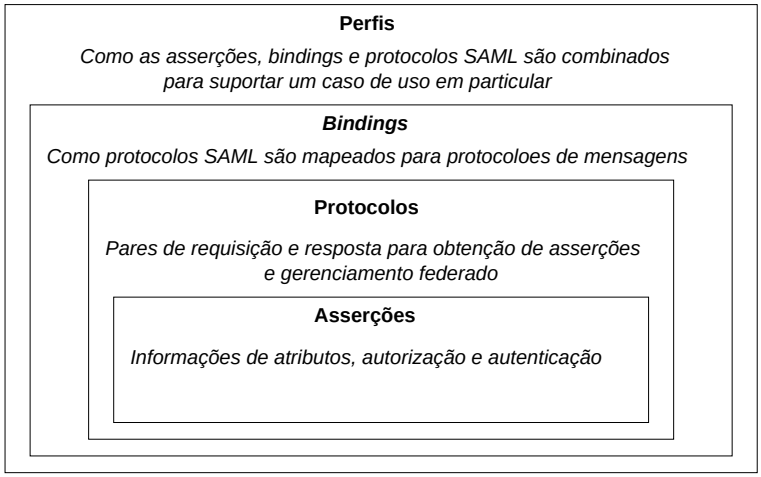

Figura 3.11: Componentes SAML (Hughes e Maler, 2004)

Web definido pela Liberty Alliance, usa asserções SAML como formatos de *tokens* de segurança através dos quais as informações de autenticação e autorização associadas a atores de serviços *Web* são comunicadas entre estes.

Shibboleth (*shibboleth.internet2.edu*) é uma iniciativa da Internet2 para compartilhamento de recursos entre pesquisadores e estudantes de instituições de ensino. O *Shibboleth* traça o perfil de seus requisitos inserindo a gerência de privacidade em asserções SAML.

A XACML (Moses, 2005) (seção 3.5) é uma linguagem baseada em XML para controle de acesso padronizada pela OASIS que descreve uma linguagem de controle de acesso e uma linguagem de requisições de respostas. A linguagem de política é usada para expressar políticas de controle de acesso e a linguagem de requisição resposta expressam consultas sobre um acesso em particular e as respostas para estas consultas. A XACML e a SAML são complementares. Uma política XACML pode especificar o que um provedor deve fazer para receber uma asserção SAML.

WS-Security (Nadalin et al., 2004) (seção 3.7.4) é um padrão OASIS que especifica extensões de segurança SOAP provendo integridade e confidencialidade de dados. WS-Security define um *framework* que torna segura estas mensagens SOAP. Existem diferentes perfis do WS-Security para diferentes formatos de *tokens* de segurança, por exemplo: certificados X.509, *tickets* Kerberos e asserções SAML.

3.6 XKMS

Aplicações que fazem uso da cifragem e assinaturas XML utilizam uma solução PKI baseada em X.509, PGP, ou SPKI. Estas infra-estruturas de chaves públicas (PKIs) permitem ligar um nome a uma chave pública. Entretanto, quando uma aplicação necessita interagir com uma outra aplicação que use uma PKI diferente da sua, as duas aplicações terão que entender as duas soluções PKI adotadas

(Verma, 2004). Por estes motivos, é desejável que a complexidade da PKI seja transferida de um lugar onde é difícil a gerência (a aplicação cliente) para o serviço especializado em gerência da PKI.

A *XML Key Management Specification* (XKMS) (Hallam-Baker, 2004) define protocolos para o registro, localização e validação de informações de chaves públicas, além da possibilidade de geração de pares de chaves. O propósito do XKMS é facilitar o gerenciamento da PKI, abstraindo a sua complexidade. Para alcançar este objetivo o XKMS define uma interface baseadas em serviços *Web* para uma PKI (Naedele, 2003), conforme pode ser observado na Figura 3.12:

Figura 3.12: Interface ao PKI (O'Neill, 2003)

Esta interface elimina a necessidade da aplicação de entender a sintaxe e a semântica complexa da PKI, podendo assim utilizar diferentes soluções de PKI sem a necessidade de modificar a aplicação. A especificação XKMS se divide em duas partes: especificação XML do serviço de informação de chaves (XML Key Information Service Specification X-KISS) e a especificação XML do serviço de registro de chaves (XML Key Registration Service Specification X-KRSS).

3.6.1 Especificação XML do Serviço de Informação de Chaves (X-KISS)

O X-KISS provê um mecanismo de processamento de informações de chaves públicas associadas às assinaturas e cifragens XML ou qualquer outro uso do elemento *ds:Keyinfo* (Hallam-Baker, 2004). O elemento *ds:KeyInfo* é definido na especificação de assinatura XML (Bartel, 2002) como um elemento opcional que contém informações sobre a chave pública que assinou o documento XML. Esta informação pode ser a chave em si, um nome de chave, um certificado X.509, um identificador de chave PGP entre outras informações de gerenciamento. Através do conteúdo deste elemento o receptor do documento XML assinado ou cifrado pode obter os dados necessários para validar a assinatura.

Ao processar a informação contida em um elemento *ds:KeyInfo* para a aplicação cliente, um serviço XKMS libera a mesma da sintaxe e complexidade das subcamadas PKI usadas para estabelecer relacionamentos de confiança (Hallam-Baker, 2004). Para isso, a aplicação cliente repassa ao serviço XKMS a informação da chave que assinou ou cifrou estes dados e o mesmo processa a informação ou responde se a chave é válida ou não. O X-KISS suporta duas operações (Verma, 2004):

- **Localizar:** esta operação localiza informações relacionadas ao elemento *ds:KeyInfo*, porém esta informação não é validada. Para obter estas informações, o serviço XKMS pode usar dados locais ou pode fazer uso de outros serviços XKMS.
- **Validar:** esta operação além de localizar as informações relacionadas a um elemento *ds:KeyInfo*, também valida estas informações. O serviço XKMS pode fornecer ao cliente uma declaração especificando o estado da ligação da chave pública a outros dados, por exemplo, um nome ou um conjunto de atributos estendidos (Hallam-Baker, 2004).

3.6.2 Especificação XML do serviço de Registro de Chaves (X-KRSS)

O X-KRSS provê operações para o registro e gerenciamento de uma chave pública. Ou seja, uma aplicação cliente pode requisitar a um serviço XKMS que registre uma chave pública associando uma informação, sendo esta um nome, um identificador ou atributos estendidos, à mesma. O gerenciamento desta informação associada a chave pública é também realizado através deste mecanismo. O X-KRSS suporta quatro operações (Hallam-Baker, 2004):

- **Registrar:** esta operação associa uma informação a uma chave pública. A chave pode ser gerada pela aplicação cliente ou pelo serviço XKMS. Caso a chave tenha sido gerada pela aplicação cliente, a mesma deve provar a posse da chave. Se a chave for gerada pelo serviço, o mesmo envia a chave privada para o usuário. O serviço XKMS também permite que uma chave privada seja registrada, caso o par de chaves tenha sido gerado pelo mesmo.
- **Recuperar:** esta operação recupera chaves privadas que tenham sido registradas previamente. É semelhante a operação de registro e só pode ser executada quando a chave privada tiver sido registrada previamente e gerada pelo XKRSS.
- **Reemitir:** esta operação reemite associações de informações a chaves públicas, realizadas previamente na operação de registro. A reemissão permite a geração de novas credenciais na PKI subjacente.
- **Revogar:** esta operação revoga uma chave previamente registrada e qualquer credencial cifrada associada a esta. Revogar esta associação significa que esta não é mais confiável para nenhum uso da aplicação.

3.6.3 Protocolos de Trocas de Mensagens

O protocolo de trocas de mensagens XKMS consiste de pares de requisição e resposta. Os modos de processamento destas mensagens podem ser síncronos ou assíncronos. No processamento **síncrono**, conforme pode ser visto na Figura 3.13, o serviço XKMS responde ao pedido sem emitir mais respostas com respeito a esta requisição.

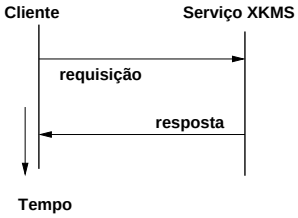

Figura 3.13: Processamento Síncrono

Já no processamento **assíncrono** (Figura 3.14), o serviço XKMS não completa a requisição imediatamente, notifica a aplicação cliente de que o pedido ainda não foi satisfeito e que respostas

subseqüentes serão enviadas. O processamento assíncrono pode ser usado para permitir ao serviço XKMS intervir durante o processamento do pedido.

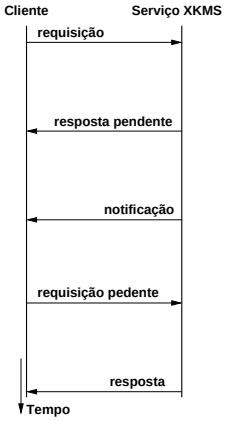

Figura 3.14: Processamento Assíncrono

Por exemplo, pode ser solicitado ao mesmo a verificação e aprovação de todos os pedidos de registro X-KRSS antes que sejam processados. Uma aplicação cliente pode notificar a um serviço XKMS de que esta aceita processamento assíncrono de um pedido na mensagem de requisição.

O serviço XKMS que recebe este pedido pode responder imediatamente ou não. Caso não responda imediatamente, este avisa a aplicação cliente na mensagem de resposta. O modo através do qual a aplicação cliente toma conhecimento de que o serviço completou o processamento não é especificado pelo XKMS. Uma forma seria através de uma mensagem de notificação. Após a aplicação cliente receber a notificação esta envia a requisição pendente ao serviço XKMS, que retorna a resposta da mesma. Um serviço XKMS só deve retornar uma mensagem pendente quando especificado no pedido correspondente. Ou seja, se o serviço receber um pedido que não pode ser processado imediatamente este deve notificar o cliente na mensagem de resposta.

O serviço XKMS pode utilizar um protocolo de **requisição de duas fases** para se proteger contra negação de serviço. Este protocolo permite ao serviço XKMS executar uma autenticação fraca da fonte de requisições XKMS (aplicação cliente). O Protocolo de duas fases consiste nas seguintes trocas, que podem ser observadas na Figura 3.15:

- Na primeira fase a aplicação cliente envia a requisição e o serviço XKMS responde a esta com o *nonce*[5].
- Na segunda fase a aplicação reapresenta a requisição original juntamente com o *nonce* emitido previamente pelo serviço XKMS.

Uma aplicação cliente pode avisar ao serviço XKMS que suporta o protocolo de requisição de duas fases de forma semelhante ao processamento assíncrono. Um serviço XKMS só deve retornar o *nonce* para a aplicação cliente quando especificado no pedido correspondente. Ou seja, se o serviço

[5] Valor aleatório que é enviado de um servidor ou aplicativo, solicitando autorização do usuário.

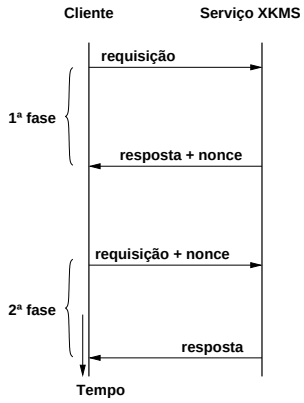

Figura 3.15: Protocolo de Duas Fases

XKMS receber um pedido que exija o protocolo de duas fases e a aplicação cliente não especificou que aceita o mesmo, o serviço XKMS deve enviar uma mensagem de resposta notificando a aplicação cliente que este protocolo é necessário. O serviço também deve verificar se o valor do *Nonce* especificado na segunda fase da requisição foi gerado recentemente pelo serviço XKMS, ou se este já não foi enviado anteriormente.

Um serviço XKMS também suporta **requisições compostas**, permitindo que múltiplas requisições sejam feitas ao mesmo tempo. Estas consistem de uma requisição externa e uma ou mais requisições internas. Não há uma ordem implícita entre as requisições internas. A semântica de um conjunto de requisições enviadas em separado ou a de uma única requisição composta é a mesma. O retorno de uma requisição composta é uma resposta composta que consiste de uma resposta externa e zero ou mais respostas internas.

3.7 Serviços *Web*

Conforme definido em (Booth, 2004), um serviço *Web* é um sistema de *software* desenvolvido para suportar interoperabilidade de interação entre máquinas sobre a rede. Os serviços *Web* definem pontos de entrada para os sistemas internos das organizações, além de empacotar componentes intra-organizacionais provendo conectividade entre aplicações inter-organizacionais autônomas e heterogêneas (Medjahed et al., 2003).

Para isso, estes serviços, provêem um sistemático e extensível *framework* para interação entre aplicações, construído sobre os protocolos *web* e baseado nos padrões XML. O *framework* de serviços *Web* é dividido em três áreas (Curbera, 2002): protocolos de comunicação (SOAP), descrições de serviços (WSDL) e descoberta de serviços (UDDI).

3.7.1 SOAP

O SOAP (*Simple Object Access Protocol*) provê um mecanismo simples e leve para troca de informação estruturada e tipada em um ambiente distribuído (Gudgin et al., 2003). Este protocolo utiliza tecnologias XML como suporte para a construção de mensagens, fazendo uso de uma variedade de protocolos subjacentes para transporte das mesmas. Este suporte é desenvolvido para ser independente de qualquer modelo de programação em particular, ou de qualquer outra semântica específica de implementação (Gudgin et al., 2003).

A princípio, o SOAP é um paradigma de troca de mensagens de um só sentido, porém aplicações podem criar padrões de interações mais complexas, como por exemplo requisição/resposta, requisição/múltiplas respostas, entre outras. Estes padrões podem ser obtidos através da combinação de trocas *one-way*, adicionando ainda as características providas pelo protocolo subjacente ou informações específicas de cada aplicação. Conforme pode ser visualizado na Figura 3.16, o formato de uma mensagem SOAP contém um elemento chamado envelope SOAP que por sua vez contém dois subelementos: cabeçalho (elemento opcional) e corpo (elemento obrigatório). Dentro destes dois elementos ainda podem haver outros subelementos chamados blocos de cabeçalho e subelementos do corpo, respectivamente.

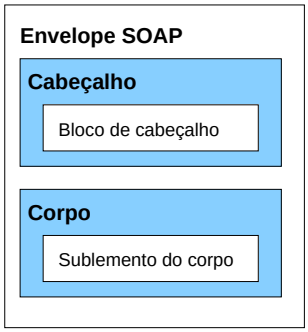

Figura 3.16: Mensagem SOAP

No elemento cabeçalho estão contidas informações de controle que incluem a passagem de diretivas ou informações contextuais relacionadas ao processamento da mensagem. Isto permite que uma mensagem SOAP possa ser estendida para atender requisitos de uma aplicação específica (Mitra, 2003). O elemento corpo define a troca de informação entre o emissor inicial e o receptor final, ou seja, é nos subelementos do corpo onde estão contidas as informações trocadas entre as partes emissora e receptora. Estes subelementos do corpo também podem ser utilizados para transportar um erro, ou informações obrigatórias esperadas pelo receptor final da mensagem. O SOAP 1.2 também comporta funcionalidades para a chamada remota de procedimento usando a extensibilidade e flexibilidade do XML (Gudgin et al., 2003).

3.7.2 WSDL

A *Web Services Description Language* (WSDL) (Chinnici, 2005) provê um modelo e um formato XML para descrição de serviços *Web*. Esta linguagem permite separar a descrição das funcionalidades oferecidas por um serviço das representações concretas que descrevem, por exemplo ”como”e ”onde”tal funcionalidade é oferecida. Esta separação é realizada para promover a reusabilidade de descrições e separar interesses de projetos independentes.

WSDL define uma descrição abstrata em termos de mensagens trocadas na interação de um serviço. Há três principais componentes nesta descrição (Curbera, 2002): vocabulário, a mensagem e a interação. No vocabulário encontram-se as definições de tipo usadas na WSDL. As mensagens (*message*) são descritas em um formato independente, usando um sistemas de tipos, como os esquemas XML. Um exemplo de uma descrição abstrata pode ser observada na Figura 3.17. Um tipo de porta (*portType*) é uma coleção de operações que são suportadas por um ponto de acesso, onde cada operação (*operation*) representa o padrão de trocas de mensagens que o serviço *Web* suporta. As mensagens (*message*) são combinadas nos elementos operação (*operation*) e tipo de porta (*portType*) para definir interações.

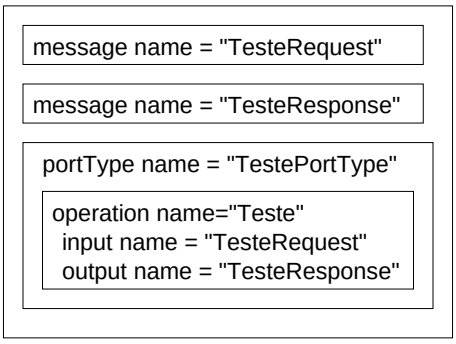

Figura 3.17: Descrição abstrata

As informações concretas de ligação definem quais protocolos de comunicação utilizar na transferência de mensagens (SOAP sobre HTTP, por exemplo), como concretizar interações com o serviço sobre este protocolo e os endereços de rede onde o mesmo está disponível. Na descrição concreta, conforme pode ser visto na Figura 3.18, um elemento ligação (*binding*) especifica o transporte e o formato detalhado para uma ou mais mensagens, um ponto de acesso (*port*) associa um endereço de rede a uma ligação (*binding*) e um serviço (*service*) agrupa os pontos de acesso que implementam uma interface comum.

3.7.3 UDDI

Para que um serviço *Web* seja utilizado, é necessário que os potenciais usuários tenham conhecimento de sua interface, da sua semântica, da sua chamada e a sua localização. O objetivo da *Universal Description Discovery and Integration* (UDDI) (Clement et al., 2004) ou Serviço para a Localização e Publicação de Serviços é a definição dos conjuntos de serviços que suporta a descrição do negócio,

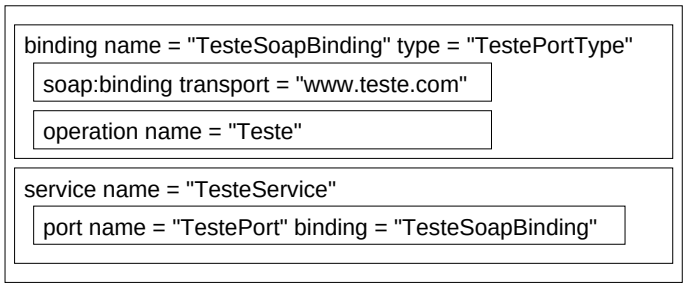

Figura 3.18: Descrição concreta

a descrição de como o serviço está disponível e as interfaces técnicas utilizadas para acesso a estes serviços. O Serviço para a Localização e Publicação de Serviços está baseado em um conjunto de padrões (HTTP, XML, esquema XML e SOAP) e provê uma infra-estrutura fundamental e interoperável para ambientes de *software* baseados em serviços *Web* disponíveis ao público ou expostos internamente nas organizações.

Um registro UDDI oferece um mecanismo para classificar, catalogar e gerenciar serviços *Web* para que os mesmos possam ser descobertos e utilizados. Esta classificação dos serviços *Web* é feita através do esquema XML do UDDI. Este esquema define quatro estruturas para a classificação de informações técnicas que uma pessoa necessita saber a fim de usar serviços *Web* parceiros. Na Figura 3.19 encontra-se a hierarquia da informação e os nomes dos elementos XML chaves que são utilizados:

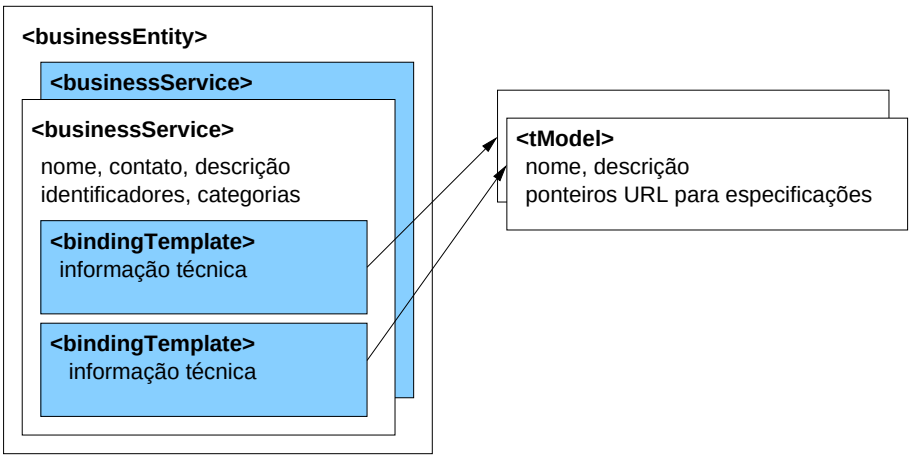

Figura 3.19: Estrutura de dados de um registro UDDI (Clement et al., 2004)

- **businessEntity:** Elemento XML que suporta publicação e descoberta de informação sobre os registros de negócio UDDI. Estas informações permitem que pesquisas possam ser realizadas para localizar negócios que servem a uma indústria em particular, categoria de produto, ou quais estão localizado dentro de uma região geográfica específica.

- **businessService:** Elemento XML que é utilizado para agrupar serviços *Web* relacionados a processos de negócio ou categorias de serviços. Exemplos de processos de negócio que podem

ser agrupados, são os serviços de compra e os serviços de transporte.

- **bindingTemplate:** Elemento XML que contém informação relevante aos aplicativos que são conectados e se comunicam com serviços *Web* remotos. Esta informação inclui endereço de contato, assim como suporte a opções de informação que podem ser utilizados para descrever serviços hospedados e serviços que requerem valores adicionais para serem descobertos antes da invocação do serviço. Características adicionais são definidas para permitir opções de roteamento complexo como balanço de carga.

- **tModel:** Elemento contém dados sobre a especificação, incluindo seu nome, organização que o publicou e ponteiros URL para sua especificação real. O conjunto destes elementos formam uma impressão digital que pode ser utilizada para reconhecer um serviço *Web* que implementa um comportamento em particular e uma interface de programação.

3.7.4 WS Security

O projeto de serviços *Web* provê um nível de abstração para diferentes plataformas e linguagens de programação, permitindo que sistemas de organizações diferentes se comuniquem de forma aberta. Além de utilizar plataformas e linguagens de programação diferentes, estas organizações podem utilizar diferentes tecnologias de segurança. Para prover um nível de abstração às organizações que utilizam diferentes tecnologias surgiu a especificação *Web Services Security* (*WS Security*) (O'Neill, 2003).

Uma mensagem SOAP pode sofrer vários tipos de ataques, entre estes: modificação (uma mensagem modificada), interceptação (leitura da mensagem por uma pessoa não autorizada) ou ainda, personificação (uma pessoa não autorizada envia mensagens falsas a um serviço *Web*). Para proteger-se destes ataques e autenticar as mensagens SOAP, a *WS Security* especifica um modelo de segurança de mensagens abstratas unindo *tokens* de segurança[6] em conjunto com assinaturas digitais.

O padrão WS *Security* propõe extensões às mensagens SOAP que podem ser usadas na construção de serviços *Web* seguros implementando integridade e confidencialidade de mensagens (Nadalin et al., 2004). A especificação provê três principais mecanismos: a propagação de *tokens* de segurança como parte de uma mensagem, a integridade e a confidencialidade de mensagens. A integridade e a confidencialidade das mensagens são garantidas pelas assinatura e cifragem XML (XML Signature e XML Encryption seção 3.2 e 3.3) (Bartel, 2002) (Imamura, 2002) respectivamente. Estes serviços podem ser usados em conjunto com outras extensões de serviços *Web* e protocolos de alto nível específicos da aplicação para acomodar uma larga variedade de modelos e tecnologias de segurança, entretanto, não garantem uma solução completa para a segurança de serviços *Web*. A seguir serão apresentadas outras especificações sobre o protocolo SOAP para garantir a segurança de serviços *Web*.

[6] Um *Token* de segurança representa uma coleção de um ou mais direitos. (Nadalin et al., 2004)

3.7.4.1 WS-Policy

O *WS-Policy* (Bajaj, 2004) permite que organizações especifiquem requisitos de segurança em seus serviços *Web*. Estes requisitos de segurança incluem os algoritmos de suporte a cifragem e assinatura digital, atributos de privacidade (como quais parâmetros devem ser cifrados) e como esta informação pode ser ligada a um serviço *Web*. O objetivo principal da *WS-Policy* é prover mecanismos necessários para permitir que aplicações *web* especifiquem informação de política. A *WS-Policy* também define uma política como uma coleção de políticas alternativas, onde cada política alternativa é uma coleção de asserções.

3.7.4.2 WS-Trust

A especificação *WS-Trust* (Anderson et al., 2005b) define mecanismos gerais para trocas de mensagens durante a aquisição ou comunicação de *tokens* dentre diferentes domínios de confiança. Esta especificação usa os mecanismos básicos de mensagens seguras definidos no *WS-Security* e define primitivas adicionais e extensões para troca de *token* de segurança permitindo a emissão e disseminação de credenciais dentro de diferentes domínios de confiança. Entretanto, cada uma das partes precisa determinar se pode confiar nas credenciais declaradas pela outra parte.

A *WS-Trust* descreve um modelo para que se obtenha um relacionamento de confiança, tanto direto, quanto por meio de agentes (incluindo terceiros e intermediários). Este modelo é baseado em um processo no qual um serviço *Web* indica ao cliente um conjunto de direitos, descritos através do *WS-Policy*. Se a mensagem enviada por este cliente não provar o conjunto de direitos, a mesma é ignorada pelo serviço. Caso o cliente não possua os *tokens* necessários para provar a um serviço os direitos exigidos, este pode entrar em contato com o serviço de *tokens* de segurança e requisitá-los. O serviço de *tokens* também pode exigir seu próprio conjunto de direitos para autenticar e autorizar a requisição destes *tokens*.

Além disso, o serviço de *tokens* pode formar a base de confiança para emissão e alcance dos *tokens* que podem ser usados para suportar relacionamentos de confiança entre diferentes domínios. Ou seja, o serviço não necessita confiar na autoridade que emitiu o *token* original, desde que confie no serviço que está comprometido na troca do novo *token*.

3.7.4.3 WS-SecureConversation

A *WS-Secure Conversation* (Anderson et al., 2005a) define extensões da *WS-Security* e da *WS-Trust* para prover comunicação segura, além de mecanismos para estabelecer e compartilhar contextos de segurança[7], assim como estabelecer e compartilhar as chaves derivadas do estabelecimento destes contextos. Além disso, a *WS-Secure Conversation* também descreve como um serviço pode trocar de contexto de uma forma segura.

[7] coleções de direitos sobre atributos de segurança e dados relacionados

Ao estabelecer um contexto, um serviço pode suportar além de *tokens* de segurança usando tecnologia de chaves assimétricas, *tokens* de segurança utilizando chaves simétricas. Ou seja, inicia-se o processo com o uso de uma chave assimétrica para negociar uma chave simétrica, que pode ser utilizada por uma série de mensagens SOAP. Isto significa que cada mensagem SOAP não tem que passar por um processo longo e intenso de autenticação em nível de mensagens (O'Neill, 2003).

3.7.4.4 WS-Federation

A especificação *WS-Federation* (Bajaj, 2003) define mecanismos para permitir que diferentes domínios de segurança sejam unidos usando mecanismos diferentes ou semelhantes permitindo confiança intermediada de identidades, atributos e autenticação entre serviços *Web* participantes nestes domínios.

A *WS-Federation* age uma camada acima das especificações *WS-Policy*, *WS-Trust* e *WS-SecureConversation*, que são usadas para determinar quais *tokens* são consumidos e como requisitá-los de um serviço de emissão de *tokens* de segurança, indicando como relacionamentos de confiança são gerenciados.

3.8 Conclusão do Capítulo

Neste capítulo foram apresentados os padrões de segurança para o XML. Foram vistas as especificações para garantir a integridade, confidencialidade de documentos XML, além da especificação de um serviço que funciona como uma interface para PKIs. Esquemas XML para a troca de informações de autenticação e autorização e para a definição de políticas de autorização também foram apresentados. Além disso, foram expostos os padrões de segurança para serviços *Web* definidos sobre o protocolo SOAP.

Capítulo 4

Modelo de gerência para o SPKI através do XKMS

4.1 Introdução

O modelo de confiança, fundamentado na formação de cadeias de autorização do SPKI/SDSI, define políticas de autorização distribuídas através da cadeia de confiança formada pela delegação de direitos, propagada através dos certificados de autorização de um servidor até um cliente. Este modelo é fortemente focado no principal que deseja acessar a um recurso (Santin, 2004).

No capítulo 2, uma extensão do modelo de confiança SDSI/SPKI chamada de Federação SPKI (Santin, 2004) foi apresentada. A Federação SPKI é uma entidade que reúne principais com interesses afins e atua como um agente facilitador na localização de certificados e principais, pois permite o compartilhamento do acesso ao seu repositório de certificados. Neste modelo foi introduzido o gerente de certificados (GC), que é principalmente um repositório de cadeias de certificados de autorização SPKI/SDSI, que serve apenas ao grupo de principais de sua Federação e não participa ativamente de nenhuma cadeia de autorização (Santin et al., 2003).

Ao integrar a noção de Federação SPKI a infra-estrutura do SPKI/SDSI, os clientes passam a ter uma alternativa para a busca de cadeias apropriadas de modo a obter o acesso desejado. Este modelo de gerência visa preservar a filosofia adotada no modelo SPKI/SDSI no qual o principal que deseja obter acesso a algum recurso, é inteiramente responsável pela busca das cadeias de certificados que lhe forneçam o direito de acesso. No entanto, esta filosofia sobrecarrega o cliente, já que este necessita entender a complexidade da PKI, além de ser o responsável pela busca de certificados.

Por outro lado, a especificação XKMS define um serviço *Web* de confiança para a gerência de PKIs. O XKMS através de protocolos padrões independentes da tecnologia de segurança usada, viabiliza aplicativos com um foco maior nas atividades de negócios. A complexidade do gerenciamento da PKI é então deixado para um serviço *Web* de confiança.

O modelo de gerência proposto nesta dissertação é uma extensão do modelo de Federação SPKI

apresentado em (Santin, 2004), que também provê suporte à gerência de certificados. Neste modelo, as funções de gerenciamento e estabelecimento de confiança entre os elementos da Federação SPKI, assim como a busca de certificados, antes atribuídas ao gerente de certificados, são repassados a um serviço XKMS. O serviço XKMS, em outras palavras, deve encapsular muitas das funções de gerenciamento introduzidas em (Santin, 2004) para o SPKI/SDSI.

Ao livrar o cliente destas necessidades, o serviço torna as aplicações menores e mais facilmente implementadas. Além disso, estas aplicações tornam-se independentes da tecnologia de segurança subjacente, já que, a responsabilidade pela localização das cadeias de certificados passa também a ser responsabilidade do serviço XKMS.

Apesar das facilidades citadas, a integração do XKMS ao SPKI não é simples, pois o XKMS está fortemente focado na PKI X.509 e esta infra-estrutura difere do SPKI. A PKI X.509 segue uma abordagem centralizada de autenticação e define um modelo hierárquico de confiança baseado na nomenclatura X.500. Já o SPKI trabalha com uma estrutura de nomes locais e define um modelo igualitário, onde cada chave pública torna-se uma entidade certificadora, que pode divulgar e assinar certificados.

Neste capítulo, o nosso modelo de gerência que faz uso do XKMS nas buscas de cadeias de certificados SPKI nas Teias de Federações, é apresentado como um Serviço XKMS provendo as funções de gerenciamento para uma Federação SPKI.

4.2 Gerenciamento Federado do SPKI através do XKMS

No modelo proposto neste trabalho, as tarefas de localização e validação de certificados de cadeias de autorização da PKI usada são repassadas para um serviço XKMS. Desta forma, o serviço XKMS facilita a interação entre o cliente e o servidor de aplicação, tornando disponível o suporte de uma Federação SPKI. Além disso, o serviço XKMS provê operações de armazenamento, e de recuperação, necessários para que os clientes consigam efetivar o acesso nos servidores.

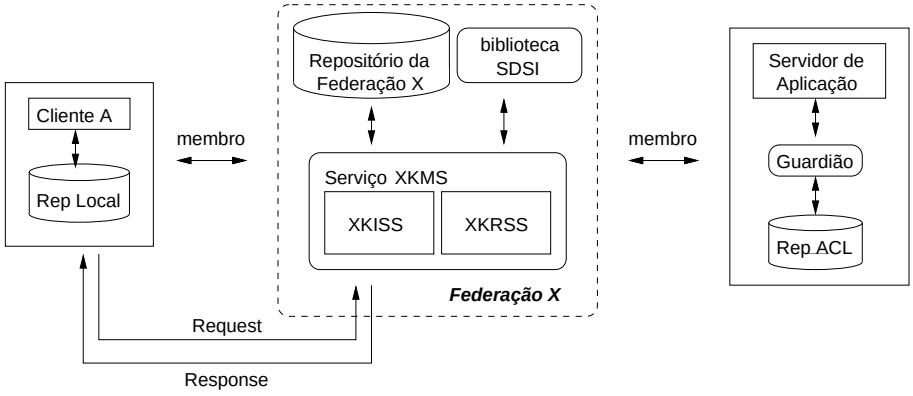

Figura 4.1: Elementos da Federação

Conforme pode ser observado na Figura 4.1, o serviço XKMS provê dois serviços: XKISS e XKRSS. Neste modelo, o XKISS é responsável pelo gerenciamento das informações SPKI e prevê as operações de localização e validação. Já o XKRSS é responsável pelo gerenciamento das Federações SPKI e prevê a operação de registro.

A especificação do XKMS prevê para o XKRSS além da operação de registro, as operações de reemissão, recuperação e revogação. As operações de reemissão e recuperação não são utilizadas neste modelo, pois estas são específicas para o gerenciamento de chaves privadas e na filosofia SPKI ninguém além do detentor da chave privada pode emitir certificados. Também não foi utilizada a operação de revogação, já que o SPKI/SDSI é caracterizado por certificados com períodos de validade bem definidos, o que descaracteriza de certa forma a necessidade de CRLs descrevendo políticas negativas como nos outros modelos de controle de acesso (Santin, 2004).

Cada serviço XKMS serve apenas aos grupos de principais de sua Federação SPKI (as chaves públicas de seus integrantes formam um grupo SDSI), não participando ativamente de nenhuma cadeia de autorização.

A forma como o modelo de gerência, suportado através das Federações SPKI, é integrado ao serviço XKMS está representado no cenário da Figura 4.2. Quando ativada uma operação de localização de certificados no repositório da Federação, o serviço XKMS retorna a identificação do principal, detentor do acesso, através de uma URI [1] (*Uniform Resource Identifier*) (BERNERS-LEE et al., 1998). De posse desta URI, o principal que deseja obter o acesso pode negociar a delegação da permissão com o principal detentor do direito. A negociação de permissões pode ser realizada de maneira simples (através de uma delegação), já que os principais são membros de uma mesma Federação SPKI. Entretanto, dependendo da aplicação pode haver a necessidade de uma negociação mais complexa.

No cenário da Figura 4.2, podem ser observados os clientes A e B, que são membros da Federação X. No passo 1, o servidor de aplicação emite e envia um certificado de autorização delegável ao cliente A. Este cliente então passa a ter o direito de acesso ao recurso protegido pelo guardião, além de poder delegar este direito. No passo 2, o cliente A armazena este certificado no repositório da Federação, para que fique disponível aos demais membros da mesma. No passo 3, o cliente B tenta acessar o recurso, porém o guardião do servidor de aplicação verifica que este cliente B não possui o direito de acesso e responde com um desafio (passo 4). O cliente B realiza uma busca no seu repositório local e não encontra a cadeia de certificados necessária. Entretanto, o cliente B, através das Federações SPKI, tem uma alternativa para localizar o certificado desejado. No passo 5, o cliente B faz uma requisição de localização para o serviço XKMS que procura no repositório da Federação X e encontra o certificado que o cliente A armazenou no passo 2. No passo 6, o serviço XKMS retorna a URI do cliente A de forma que o cliente B possa localizá-lo e negociar com o mesmo a delegação do direito de acesso (passo 7). Após a negociação, o cliente A emite um certificado de autorização ao cliente B (passo 8). Finalmente o cliente B, de posse de uma cadeia de certificados de autorização que o liga ao serviço desejado, assina esta cadeia junto a uma requisição e envia ao servidor de aplicação (passo 9), que por sua vez libera o acesso ao recurso (passo 10).

[1] A URI fornece basicamente as seguintes informações: o mecanismo de acesso, a máquina hospedeira e o nome do recurso sendo acessado.

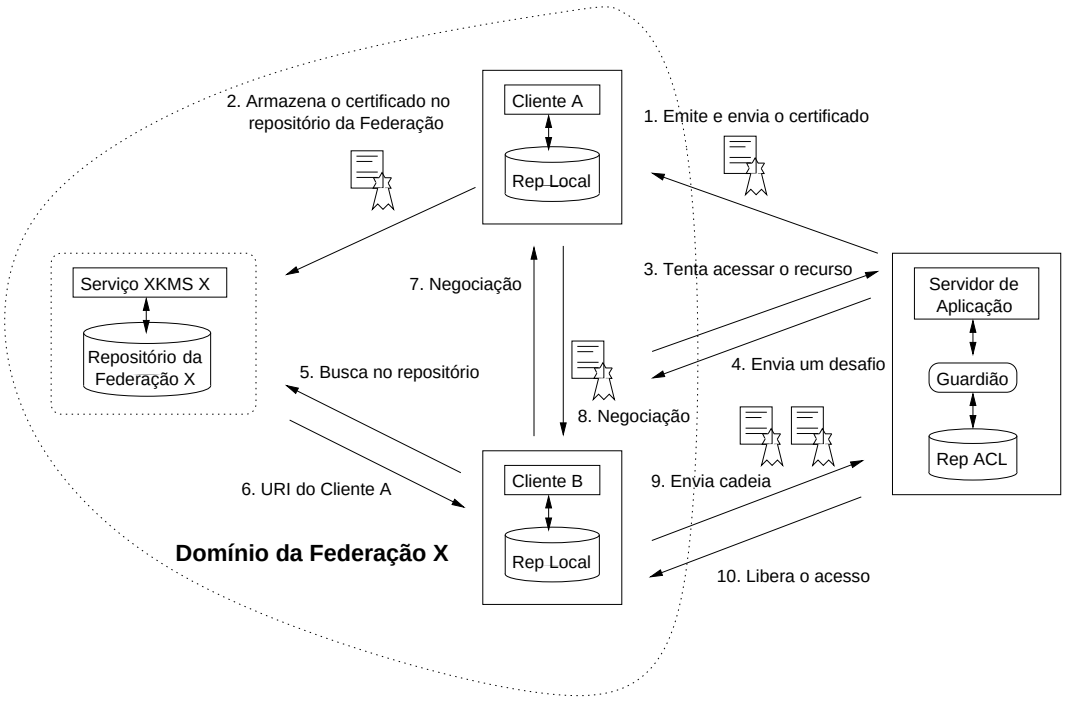

Figura 4.2: Gerenciamento Federado do SPKI através do XKMS

4.3 Processo de Filiação e de Registro de certificados no repositório da Federação

Um principal pode filiar-se a quantas Federações SPKI ele for aceito. Para isso, conforme definido em (Santin, 2004), o mesmo deve fornecer um endosso efetivado através da apresentação de um *threshold certificate*, assinado por *k*-dentre-*n* membros definidos pela Federação em questão,[2] juntamente com seu o certificado de nome, que será incluído no repositório da mesma. Este certificado de nomes é utilizado na identificação de principais. A cada novo membro da Federação SPKI é emitido um novo certificado de grupo, expressando a participação no grupo SDSI, para fins de comprovação da filiação (*membership*).

Neste trabalho, o processo de filiação se dá através da operação de Registro do XKRSS. Esta operação pode ser utilizada tanto para a afiliação de membros quanto para envio de certificados delegáveis para o repositório da Federação SPKI. Desta forma, quando um serviço XKMS recebe uma requisição de registro, o mesmo verifica se no conteúdo da mensagem encontra-se um *threshold certificate* ou uma cadeia de certificados de autorização. Se no conteúdo da mensagem encontra-se uma cadeia de certificados de autorização delegável, o serviço XKMS verifica se o principal que enviou a requisição é membro da Federação SPKI. Se este não for membro, o serviço XKMS simplesmente ignora a requisição, caso contrário o serviço armazena este certificado no repositório da Federação SPKI, que serve como base de consulta para futuras requisições de localização. Caso no conteúdo da mensagem encontra-se um *threshold certificate*, o serviço XKMS da Federação SPKI em questão ve-

[2]Esta estratégia facilita o processo de filiação no sentido de permitir que um subconjunto (k) de principais do conjunto (n) possa atuar no papel do administrador da federação.

rifica se o endosso, além de válido, é para afiliação. Em caso afirmativo é emitido um novo certificado de grupo de filiados para o principal que fez a solicitação.

4.4 Provendo suporte as Teias de Federações SPKI

Um principal, ao filiar-se a uma Federação SPKI pode fazer uso do repositório da mesma como auxílio na localização de cadeias e/ou certificados de autorização. Porém uma Federação SPKI tem sua abrangência limitada, havendo a necessidade de que os membros de uma Federação se filiem a várias outras Federações para alcançar um certo nível de presença na rede. Para solucionar este problema, é proposto em (Santin, 2004) que a Federação SPKI através de seus Gerentes de Certificados se associem a outras Federações SPKI criando relacionamentos mútuos de confiança que permitam ampliar o modelo de confiança administrativo, formando Teias de Federações SPKI.

De forma semelhante ao proposto em (Santin, 2004), neste modelo, uma Federação SPKI pode se associar a outras Federações formando Teias de Federações SPKI, porém esta associação é realizada através dos Serviços XKMS e não mais pelos Gerentes de Certificados.

Uma Teia de Federações SPKI é formada arbitrariamente e cria caminhos de confiança entre as Federações SPKI associadas de tal modo que um membro pode fazer consultas ao Serviço XKMS de sua Federação original e este, por sua vez, realiza em nome de seu membro a busca nos Serviços XKMS aos quais está associado.

No cenário da Figura 4.3 pode ser observado um cliente A, pertencente a Federação X, que deseja acessar um servidor de aplicação pertencente a Federação Y. O serviço XKMS de X é associado aos serviços XKMS das Federações W e Z, e o serviço XKMS de Z, por sua vez é associado ao serviço XKMS da Federação Y. Neste caso, o cliente A pode recorrer ao Serviço XKMS da Federação X a qual pertence, para que este realize a localização através das Teias de Federações SPKI. Quando uma cadeia de certificados de autorização é localizada, o membro pode negociar a concessão do privilégio com o detentor do mesmo.

Um Serviço XKMS é associado com os Serviços XKMS aos quais tenha uma relação de confiança. Esta relação de confiança pode ser de dois tipos: ligação fraca de confiança ou ligação forte de confiança. O tipo da ligação de confiança é definido através da regra do negócio. Por exemplo, uma ligação forte de confiança pode ocorrer entre uma Federação de companhias de transportes aéreos e uma Federação de Hotéis. Estas Federações podem apresentar fortes ligações de confiança por terem interesses comuns, como por exemplo uma venda casada de passagens aéreas com estadias em hotéis. Caso as Federações não apresentem esta relação forte de negócios, a ligação de confiança entre estas é fraca. Por exemplo, uma Federação de companhia de transportes aéreos com uma Federação de *pet shops*.

As relações podem ser estabelecidas por questões de escala e de abrangência nas procuras de certificados; o que deve diferenciar as várias associações de uma federação são os diferentes graus de confiança que são quantificados como fracos e fortes.

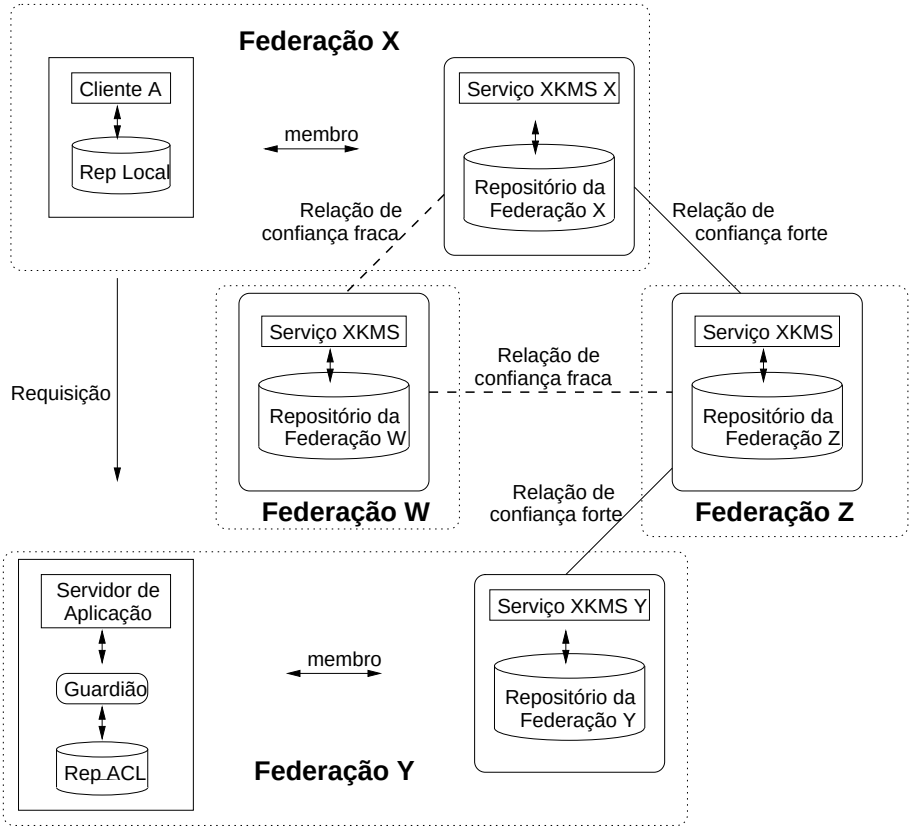

Figura 4.3: Teias de Federações

Associações entre Federações SPKI também são interpretadas como membros pela gerência de uma Federação. Neste caso, o serviço XKMS que se associa a uma outra Federação SPKI, torna-se membro da mesma. Esta associação é materializada pela emissão de um certificado de grupo SDSI de associados.

Ao serviço XKMS cabe então, a manutenção das informações referentes aos membros e associados de sua Federação SPKI, removendo ou adicionando membros e associações com outras Federações SPKI, sem promover conflito de interesses.

As federações e suas teias formam o suporte para a geração de novas cadeias de certificados de autorização SPKI. Para ilustrar este processo de formação de novas cadeias, pode-se observar o cenário da Figura 4.4, que mostra a troca de mensagens entre o Cliente, o Servidor de Aplicação, o Serviço XKMS e o detentor do privilégio. No passo 1, o Cliente envia uma requisição ao Servidor de Aplicação. Como o Cliente não possui a autorização de acesso, o Servidor envia ao mesmo um desafio (passo 2). Este desafio, neste caso é a ACL SPKI/SDSI que protege o recurso.

De posse da ACL, o Cliente chama o seu agente para que o mesmo faça uma busca no seu repositório local por uma cadeia e/ou certificado de autorização que o ligue ao servidor de aplicação e permita o acesso desejado. Se a busca local não for bem sucedida, o Cliente então recorre ao Serviço XKMS (passo 3) da Federação SPKI a qual é membro. Este Serviço XKMS tenta localizar esta cadeia e/ou certificado no repositório da Federação SPKI, se novamente não houver sucesso na

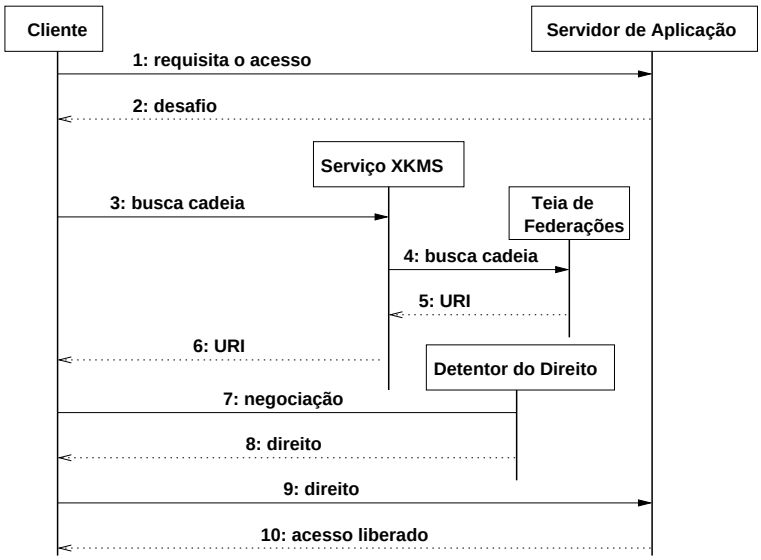

Figura 4.4: Cenário de busca de certificados

busca, o Serviço XKMS inicia a localização nos seus Serviços XKMS associados (passo 4), ou seja, a localização é feita na Teia de Federações. Após o término da busca realizada pelo serviço, o mesmo retorna a identificação do detentor do direito, contida no certificado de autorização (passo 5). De posse da identificação do detentor, a Federação SPKI de origem repassa a mesma ao seu membro (Cliente), (passo 6), que parte para a negociação da delegação do direito requerido (mensagem 7 e 8). Conforme citado anteriormente a negociação da delegação pode ser realizada de forma simples ou complexa, entretanto neste cenário a negociação foi realizada com uma simples delegação do direito desejado. Com o certificado que delega o direito ao Cliente, o mesmo responde ao desafio proposto pelo Servidor (passo 2) e por fim, (passo 9) o Servidor de Aplicação libera o acesso ao Cliente (passo 10).

A associação entre serviços, conforme definida na seção 4.3, se dá através da operação de registro do XKMS, assim como a filiação e o registro de certificados, conforme pode ser visualizado na Figura 4.5. Um grupo inicial de Federações com interesses comuns se associam através de seus Serviços XKMS.

Estas associações formam teias que assumem topologias arbitrárias, uma vez que as mesmas são estabelecidas e removidas de forma dinâmica e aleatória. Um serviço XKMS, se torna um associado de outro ao fornecer o endosso assinado por *k*-dentre-*n* associados, juntamente com seu certificado de nome. Um Serviço XKMS também pode se associar a quantos serviços XKMS quiser.

Em uma requisição de associação, o Serviço XKMS da Federação SPKI em questão verifica se na mensagem recebida existe um *threshold certificates* e se este endossa uma associação. Neste caso o serviço XKMS emite um certificado de grupo de associados e o envia para o requisitor da operação. Da mesma forma, se o endosso é para a afiliação, é emitido um certificado de grupo de membros que é posteriormente enviado ao requisitante.

A operação de localização de um serviço XKMS é utilizada pelos membros da Federação e pelos

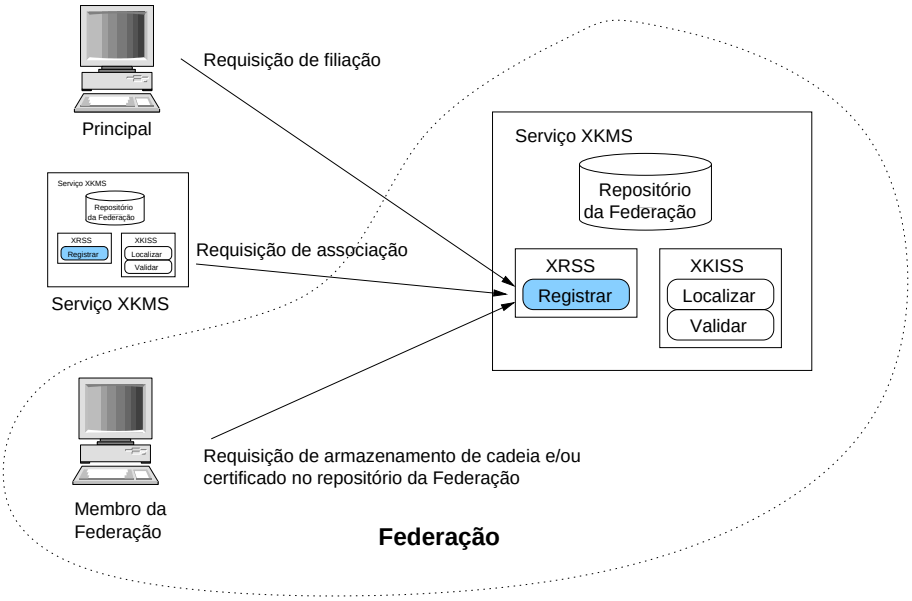

Figura 4.5: Operações do Serviço XKMS

serviços XKMS associados para requisitar uma busca de cadeias de certificados de autorização na sua Federação. Um membro também pode requisitar a um Serviço XKMS que valide uma cadeia de certificados, por exemplo, de autorização, através da operação de ”Validar”livrando o membro da Federação da necessidade de implementar esta operação.

4.5 Algoritmo de Busca de Certificados nas Teias de Federações

O uso dos serviços XKMS facilita a localização de certificados de autorização através de buscas realizadas nas teias de Federações SPKI. Nesta seção será apresentada a forma como a localização de novas cadeias de certificados de autorização nas Teias de Federações SPKI acontece.

Conforme observado anteriormente, uma ligação de confiança entre serviços XKMS pode ser de dois tipos: fraca e forte. Quando a relação de confiança é fraca, o funcionamento do algoritmo de busca de certificados é realizado de forma interativa. Ou seja, quando um membro faz uma requisição de localização de cadeias de certificados de autorização ao Serviço XKMS na Federação SPKI, este Serviço, caso o certificado desejado não se encontra no repositório da Federação SPKI em questão, faz uma requisição aos Serviços XKMS associados. Se o Serviço XKMS que recebeu a requisição verifica que a sua Federação também não possui o certificado, este retorna as URIs dos Serviços XKMS associados ao Serviço XKMS requisitante para que este possa continuar a busca na Teia de Federações.

No fim da busca, o Serviço XKMS da Federação de origem retorna uma lista de zero ou mais URIs, onde cada URI contém a identificação de um principal que possui o direito de acesso ao recurso desejado ao cliente. Se a lista de URIs não for vazia, o cliente parte para a negociação da concessão com os principais que possuem este direito.

Algoritmo 1 locate (recurso, sign, *ttl*)

```
Require: T = {Conjuto de serviços com associação forte.}
Require: U = {Conjuto de serviços com associação fraca.}
 1: if (ttl > 0 ) and (signature.getPublicKey() ∈ (T ∪ U)) then {A requisição deve ser proveniente de um serviço com associação forte ou
    fraca e ttl tem que ser maior que zero.}
 2:    ttl ← ttl-1
 3:    uris ← buscaRepositorio(recurso)
 4:    if (uris ≠ ∅) then {Algum membro possui o recurso}
 5:       locateHit(uris, "members")
 6:    else if (signature.getPublicKey() ∈ U) then {A requisição é de um serviço de associação fraca}
 7:       uris = associatedsURIs(T ∪ U)[URIs dos associados]
 8:       locateHit(uris, "associateds")
 9:    else if (ttl > 0 ) then {Busca nos assoiados de relação forte e fraca.}
10:       N ← T ∪ U
11:       while (N ≠ ∅) do
12:          X ← firstElement(N)
13:          X.locate(recurso, sign(), ttl)
14:          N ← N \ X Remove o elemento X do conjunto N
15:       end while
16:    end if
17: else
18:    locateHit(∅, "")
19: end if
```

Se a relação de confiança entre os serviços associados é forte, a busca é feita de forma recursiva. Neste caso, quando um membro faz um requisição ao Serviço XKMS da sua Federação SPKI e este não possuindo a permissão, deve repassar a procura a seus associados diretos. A busca em questão só é repassada para um Serviço XKMS associado se a relação entre ambos for forte. Ou seja, o Serviço XKMS que recebeu a requisição de busca verifica se a requisição veio de um serviço com quem este tenha uma relação de confiança forte. Em caso afirmativo, a busca é assumida por este. Desta forma, a busca é realizada como se um dos membros da Federação ao qual pertence a tivesse requisitado. Na localização recursiva, o serviço XKMS ao delegar a tarefa de busca de cadeias de certificados a um serviço associado, diminuem a sua sobrecarga no processamento desta localização.

Visualizando a Teia de Federações SPKI como um grafo, a busca de certificados pode ser de duas formas: a busca por amplitude e a busca por profundidade. Neste modelo, assim como em (Santin, 2004), optou-se pela busca em amplitude, pois quanto mais distante o membro e o principal detentor do direito, maior a dificuldade de negociação, pois estes pertencem a Federações distantes, por conseqüência a probabilidade de os membros possuírem interesses afins é menor. Para que a busca não seja infinita foi inserido também um controle de saltos. Este controle de saltos é passado na requisição de busca do certificado (valor *ttl*). O cliente também apresenta um *timeout* para que este não fique esperando indefinidamente por uma resposta do serviço XKMS.

O protocolo que segue este modelo possui duas mensagens: *locate*, usada para efetuar a localização da cadeia e/ou certificado; e a *locateHit*, informando que a cadeia e/ou certificado procurado(o) foi encontrado(a). A mensagem *locate* é composta por três elementos principais, sendo estes: o recurso, a assinatura e o *ttl*. O valor *ttl*, conforme citado anteriormente é usado para o controle de saltos, o recurso é quem identifica a cadeia de certificado que está sendo buscada, e através da assinatura é obtida a chave pública para a identificação do requisitor (se é membro, ou se é Federação com associação forte ou fraco).

O Algoritmo 1 descreve como esta localização é realizada. Primeiramente o Serviço XKMS verifica através da chave pública contida na assinatura se o cliente que fez a requisição é um Serviço

XKMS associado ou se é um membro da Federação e se o valor de *ttl* é maior que zero(linha 1). Se não é nem membro nem associado e o valor de *ttl* não é maior que 0, o serviço XKMS não responde a requisição. Caso contrário, o valor de *ttl* é diminuído de 1 e a busca é realizada normalmente. Se o serviço XKMS encontrar algum membro que detenha o recurso, sua(s) identificação(s) através da(s) URI(s) é(são) enviada(s) na mensagem *locateHit*, juntamente com um valor identificando que as mesma(s) são URI(s) de membros que detém o recurso (”members”, linha 5).

Caso não seja encontrado nenhuma cadeia de certificados no repositório da Federação, é então verificado se a chave púbica do requisitor pertence a uma Federação associada de relação fraca de confiança (linha 6). Em caso afirmativo, o serviço XKMS retorna na mensagem *locateHit*, contendo as URIs dos seus Serviços XKMS associados (de associação forte e fraca), juntamente com um valor identificando que as mesma(s) são URI(s) de associados (”associateds”) para que o serviço que fez a requisição possa continuar a busca (linha 8).

Caso a chave pública pertença a algum Serviço XKMS com relação de confiança forte ou a um membro da Federação, o serviço XKMS assume a localização. Como os Serviços XKMS de relações de confiança forte assumem a busca, o serviço XKMS em questão somente armazena os resultados da busca para posteriormente enviar a quem lhe fez a requisição. Quando o valor de *ttl* for igual a 0, o serviço XKMS responsável pára a localização e envia uma mensagem *locateHit* contendo os resultados obtidos até então (linha 18).

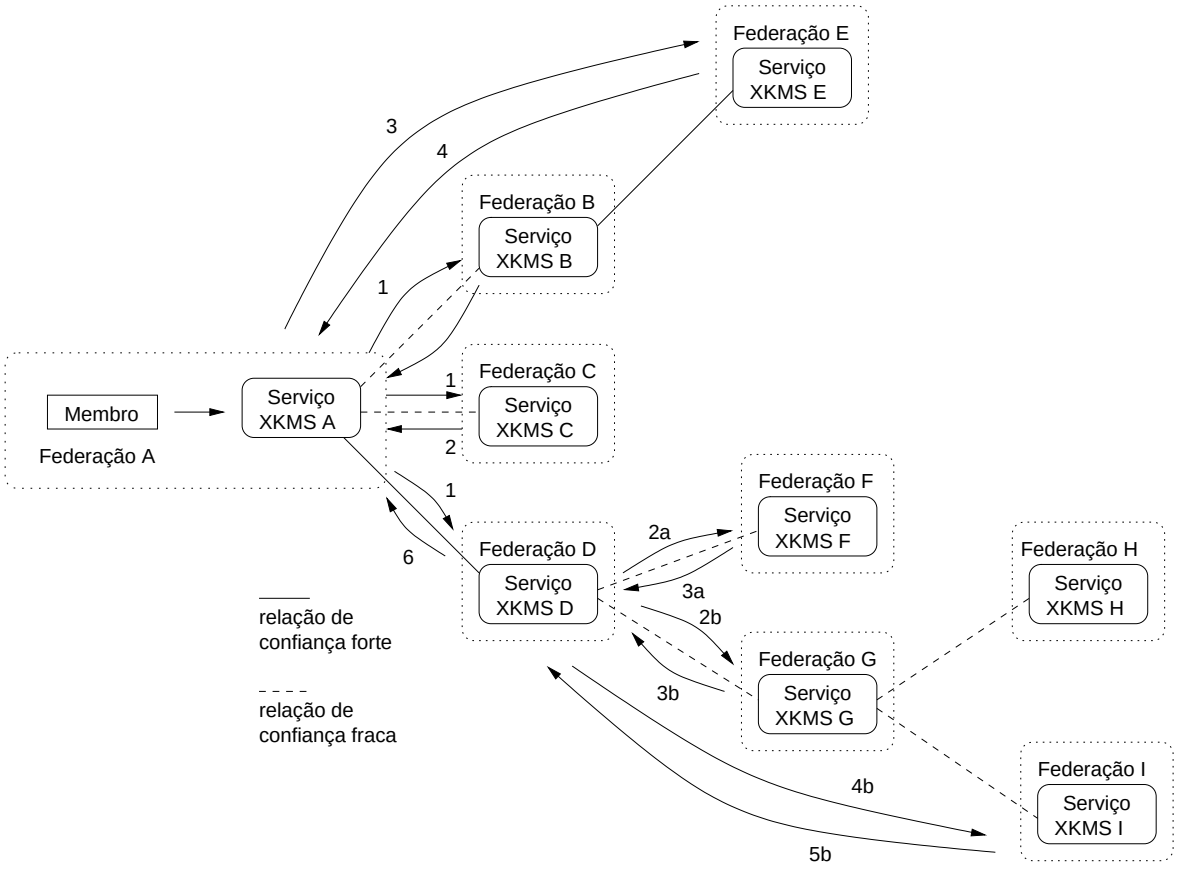

Figura 4.6: Algoritmo de busca

Um cenário de exemplo de uma busca de uma cadeia de certificados é ilustrado na Figura 4.6. Um membro da *Federação A* faz uma requisição ao *serviço XKMS A*, que por sua vez, verifica no repositório da Federação e não encontra a cadeia requisitada. Neste exemplo, os dois *serviços XKMS B* e *C*, com os quais este possui uma relação de confiança fraca, a cadeia também não é encontrada. Desta forma, o *serviço XKMS C* retorna a URI contendo a localização do *serviço XKMS E* (passos 1 e 2), para que o serviço XMKS A realize a busca.

Se a relação de confiança entre os serviços XKMS for forte, a busca é realizada de forma recursiva. Neste caso, o *serviço D*, que possui uma relação de confiança forte com o *serviço A*, assume a busca e procura a cadeia em seus associados (passos 2a e 2b). No passo 3b, o *serviço XKMS G* associado ao *serviço XKMS D* que assumiu a busca, não encontra a cadeia e repassa as URIs dos *serviços XKMS H* e *I* de seus associados. Através destas URIs, conforme passos 4b e 5b, o *serviço XKMS D* de associação forte envia uma requisição de localização de certificados ao serviço XKMS I, que finalmente retorna uma mensagem indicando que a cadeia foi encontrada e informa a URI do membro da sua federação que detém o recurso. Ao delegar a tarefa de busca de cadeias de certificados aos serviços associados de confiança forte, o serviço XKMS diminui a carga de processamento para realizar a operação de localização

4.6 Trabalhos Relacionados

Existem alguns trabalhos na literatura fazem uso de um serviço XKMS. No entanto, estes trabalhos utilizam somente a infra-estrutura de chaves pública (PKI) X.509, herdando os problemas de uma abordagem de autenticação centralizada.

Um serviço de validação de certificados usando XKMS para *grids* computacionais é apresentado em (Park et al., 2003). Este serviço introduz um módulo de validação de certificados (CVM), que é um componente que valida certificados para o cliente. Vários protocolos são utilizados para a validação de certificados, tais como, OSCP[3], SCVP[4] e LDAP[5]. O fluxo de validação de certificados funciona da seguinte maneira: o cliente gera uma requisição para a validação de um caminho de certificados e envia esta ao servidor. O servidor reconstrói o caminho usando a sua tabela de certificados. Se o caminho já se encontra na tabela local, o servidor responde ao cliente que o caminho é válido. Caso o caminho não seja encontrado na tabela local, o serviço faz a validação usando a tabela da CA em que está registrado. Se o caminho do certificado é válido, o serviço armazena este caminho na sua tabela local e responde ao cliente que o caminho é válido. Caso contrário, retorna uma resposta ao cliente indicando que o caminho de certificado é inválido. Nos testes de validação de certificados devem ocorrer também as verificações das CRLs com as políticas negativas.

O modelo proposto no nosso trabalho também apresenta a operação de validação de cadeias de certificados onde o serviço XKMS responde ao cliente o resultado da validação do mesmo. Porém,

[3]Protocolo que verifica a validade de um certificado em tempo real

[4]Protocolo de validação de certificados simples, que pode prover mais informações que somente o status de um certificado

[5]*Lightweight Directory Access Protocol*. Como o nome sugere, é um protocolo leve para acessar serviços de diretório. O LDAP roda em cima do protocolo TCP/IP ou outras conexões de transferência de serviços.

conforme citado no texto, o SPKI/SDSI é caracterizado por certificados com períodos de validade bem definidos, o que descaracteriza de certa forma a necessidade de CRLs descrevendo políticas negativas, facilitando assim a implementação desta operação.

Em (Kraft, 2002) é apresentando um modelo geral de um processador de controle de acesso para serviços *Web*. Neste trabalho o XKMS, juntamente com as extensões XML de assinatura e de cifragem, são citados como exemplo de ferramentas a serem usados para garantir a integridade, não repudiação e a confidencialidade das mensagens SOAP.

Um *framework* para implementação de segurança em serviços *Web* através de extensões do WSDL e UDDI é proposto em (Adams e Boeyen, 2002). Nestas extensões um elemento chamado *security-Parameters* e seus subelementos são incluídos no esquema do UDDI e do WSDL para prover um modo de estabelecer uma infraestrutura de política de confiança. Através destes elemento um serviço consumidor obtém os certificados e as chaves, necessários para acessar o serviço provedor, que por sua vez, pode utilizar um serviço XKMS para determinar se as chaves ou certificados recebidos são válidos.

Em (Bilykh et al., 2003) é apresentado um *grid* de informações de saúde chamado *HealthInfo*Grid que possui um componente chamado *Medical Exchange Agency* (MEA). Este componente, usa PKI X.509 e assume o papel de CA para os demais componentes do GRID. Para cada um destes componentes, o MEA emite um certificado de rede usado para a identificação e assinatura digital e implementa a interface XKMS para o gerenciamento destes certificados.

Entretanto, nestes modelos citados acima ((Kraft, 2002), (Adams e Boeyen, 2002) e (Bilykh et al., 2003)) o XKMS provê somente suporte para a PKI X.509.

Já em (Daniel J. Polivy, 2002) é apresentado uma arquitetura para autenticação de respostas assinadas de requisições feitas a réplicas de dicionários autenticados usando serviços *Web* e assinaturas XML. Se a informação da chave pública contida na assinatura necessária para a validação da mesma não estiver no documento XML, esta arquitetura cita, como exemplo de um meio de busca desta informação, um serviço XKMS. Novamente o sistema obriga a utilização da PKI X.509 nas réplicas de dicionários.

Um modelo, proposto em (Kim e Moon, 2005), define uma extensão ao XKMS para que este gerencie não somente informações de chaves públicas, mas também outras informações de segurança, tais como: chave privada, chave secreta, informações biométricas, *tokens* de segurança para serviços *Web*, entre outros. Este tipo de extensão não é interessante para o nosso modelo devido a filosofia do SPKI onde o detentor da chave é o responsável por sua chave privada, e pelos seus certificados delegáveis.

O SPKI necessita de um modelo de gerência para que principais possam localizar uma cadeia e ou certificado com direito de acesso necessário para o acesso a determinado recurso. Em (Santin, 2004), foi proposto um modelo de gerência, porém neste modelo há uma sobrecarga no cliente que além de ter que entender a complexidade da PKI, também é o responsável pela tarefa de localização de cadeias de certificados. Além disso, o acesso às funcionalidades providas pelo mesmo não apresentam uma

forma padronizada. Desta forma, neste trabalho, optou-se pelo uso de um serviço XKMS de forma que a localização seja realizada pelo mesmo, além de apresentar uma padronização de acesso as funcionalidades providas.

Na literatura não existe nenhum esforço do uso do XKMS com o SPKI. Nisto o nosso trabalho é original. A explicação para a ausência de abordagens concorrentes com a nossa é que a especificação do XKMS embora, possua elementos nos quais é possível a extensão dos mesmos, tornando-a adequada para o uso com o SPKI, na prática, esta integração é difícil porque a especificação do XKMS foi toda definida visando uma PKI hierárquica como o X.509. No nosso trabalho de integração do XKMS com o SPKI não houve nenhuma alteração ou extensão que alterasse os padrões XKMS ou comprometesse a conformidade da nossa proposta.

Para o desenvolvimento do algoritmo de localização foram observados na literatura alguns protocolos *peer-to-peer*. O Gnutella (Services, 2001) é um protocolo de compartilhamento de arquivos *peer-to-peer* que usa inundação para rotear suas mensagens através dos nodos de forma a implementar as funcionalidade de busca e procura de pares. Para reduzir o *overhead* da rede causado pelo protocolo Gnutella foi introduzido o conceito de *ultrapeer* e nós folhas (*leaf nodes*). Um nó *Ultra peer* mantém conexão com nós folhas, evitando assim que estes nós folhas recebam muitas mensagens de tráfego introduzidas pelo protocolo Gnutella. Ou seja, o *ultranode* é quem responde as requisições de busca no lugar dos nós folha conectados a este.

Fast Track (FastTrack, 2001) é um protocolo *peer-to-peer* proprietário usado no *Kazaa*, *Kazaa Lite*, *Grokster*, *iMesh*. Sua versão aberta é chamada de OpenFT. A arquitetura do OpenFT é um pouco diferenciada da arquitetura proposta no *Fast Track* e introduz três tipos de nós:

- Nós indexados (*Index nodes*): São os hosts mais confiáveis. Mantém índices disponíveis para nós de busca, coleta de estatísticas e monitoração da estrutura da rede.
- Nós de busca (*Search nodes*): São os servidores da rede que mantém os índices dos arquivos compartilhados pelos nós usuários. A configuração padrão permite a um nó de busca gerenciar informação de arquivos armazenados em 500 nós usuários.
- Nós usuários (*User nodes*): nós clientes que mantém conexões com um grande número de nós de busca.

Quando uma mensagem de busca chega a um nó de busca esta não é mais propagada. Os nós clientes enviam mensagens de busca a todos os nós de busca com os quais mantém conexão. Pode-se economizar largura de banda através de critérios para a escolha dos nós de busca para os quais a mensagem será enviada.

Estes algoritmos apesar das modificações para a diminuição de mensagens enviados pela rede, ainda apresentam um grande consumo de banda nos nós intermediários. Por este motivo, optou-se por um algoritmo no qual o serviço XKMS, ao qual foi requisitado a localização do recurso, é quem realiza a busca em cada serviço XKMS associado. Caso exista um serviço XKMS com uma associação forte, ao qual pode ser repassada a busca, o algoritmo de busca funciona de forma semelhante ao Gnutella, de forma que o serviço XKMS de associação forte assume a tarefa de localização.

4.7 Conclusão do Capítulo

Neste capítulo foi apresentado um modelo que estende o modelo de gerência baseado nas teias de federações proposto em (Santin, 2004). Nesta proposta, o Serviço XKMS é o responsável pela localização distribuída e pela validação de cadeias e/ou certificados de autorização através de teias de federações, poupando assim o cliente desta tarefa. Além disso, o Serviço XKMS também assume as funções administrativas da Federação SPKI. Quando a cadeia de autorização não existe é possível criar novas cadeias através do processo de negociação.

Capítulo 5

Implementação

5.1 Introdução

No capítulo anterior foi apresentado um modelo de gerenciamento federado para o SPKI através de serviços XKMS. Para verificar a aplicabilidade do modelo, neste capítulo serão apresentados alguns aspectos da implementação de um protótipo deste serviço e as ferramentas utilizadas no desenvolvimento do mesmo.

O protótipo desenvolvido é integrado a uma aplicação que localiza publicações científicas em um Serviço *Web* fictício. Esta aplicação foi desenvolvida com o propósito de evidenciar as vantagens da integração do XKMS ao SPKI. Outros aspectos envolvendo a análise dos resultados e a comparação dos mesmos com a literatura relacionada também serão objeto de considerações neste capítulo.

5.2 Arquitetura do Protótipo

De modo a avaliar o modelo de gerência proposto para o SPKI através do uso do XKMS, uma arquitetura que atende às especificações do modelo e aos requisitos específicos de um sistema de larga escala, foi definida como se pode ver a partir da Figura 5.1. Nesta arquitetura encontram-se os serviços de Transporte e Mensagens, onde se dá a comunicação entre o cliente e o serviço XKMS. A camada de serviços de Transporte provê o fundamento para a comunicação entre serviços *Web*, já a camada dos serviços de Mensagens forma a base interoperável dos mesmos (Weerawarana et al., 2005).

Também podem ser observados, nesta arquitetura, os serviços de descrição que definem metadados para a descrição das características dos serviços *Web* em funcionamento na rede (Weerawarana et al., 2005) e, na camada de qualidade de serviço as especificações associadas com a qualidade destas interações (Weerawarana et al., 2005). As escolhas tecnológicas que compõem cada camada desta arquitetura serão expostas na seqüência.

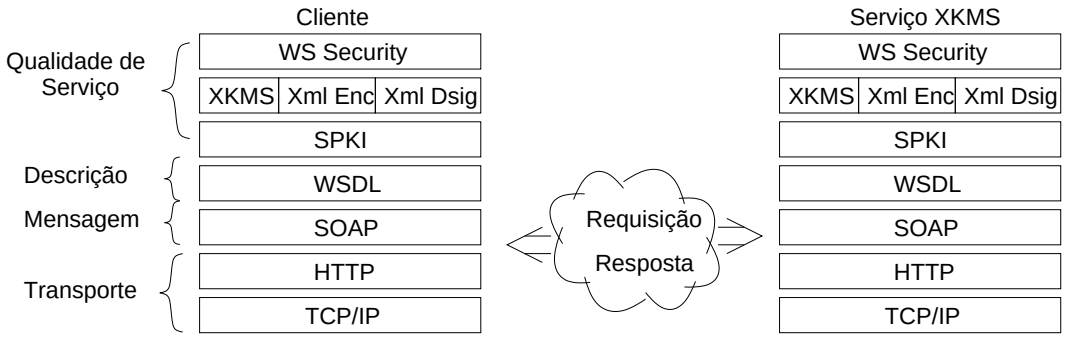

Figura 5.1: Arquitetura do Protótipo

5.2.1 Camada de transporte, mensagens e descrição

O apache Tomcat (*http://jakarta.apache.org/tomcat/*) é um *servlet container* como Implementação de Referência oficial para as tecnologias Java Servlet e JavaServer Pages (JSP), que são especificações desenvolvidas pela Sun. O Tomcat é também um servidor de aplicações Java para *Web*, robusto e eficiente o suficiente para ser utilizado mesmo em um ambiente de produção. Sua escolha deve-se ao fato deste ser software livre e de código aberto desenvolvido dentro do conceituado projeto Jakarta da Fundação Apache, disponível, seja para fins comerciais ou não.

O Apache Axis[1] é um *framework* através do qual é possível criar serviços *Web* e os clientes desses serviços. É implementado nas linguagens Java e C++. Para a versão 1.2, o Apache vem focando o suporte ao WS-I Basic Profile 1.0 [2] e à especificação JAX-RPC 1.1 [3]. Por ter seu desenvolvimento orientado a componentes, possibilita a reutilização de *Handlers*. A tecnologia de *Handlers* age como um *Firewall* XML, geralmente utilizado para a geração e validação de extensões XML de assinaturas e de cifragens (*XML Signature* e *XML Encryption*) da mensagem SOAP no cliente e no servidor. Esta tecnologia foi escolhida para a implementação desta aplicação devido a sua facilidade de integração com aplicações *Web*, independente do *container* (Tomcat, JBoss, outros) e por ser um *framework* de código aberto. O *framework* de mensagens possui uma abstração simples e clara para desenvolvimento de mensagens (*SOAP senders and listeners*). O núcleo é completamente independente da camada de transporte. As requisições são passadas em documentos XML, empacotados em envelopes SOAP (compatível com o SOAP 1.1/1.2). Estas mensagens, ao chegar ao seu destino, tem seus envelopes SOAP processados pelo Axis que entrega para as aplicações apenas o seu conteúdo. O Axis também provê suporte aos tipos de dados básicos e a serialização/desserialização automática e também realiza a conversão automática entre *Java Collections* e *SOAP Arrays*.

Dentre suas funcionalidades, o Axis possui um extenso suporte a *Web Service Description Languange* (WSDL 1.1) através do qual as interfaces Java dos serviços podem ser geradas automaticamente. Da mesma forma podem ser gerados, os documentos WSDL com as descrições das interfaces,

[1] http://ws.apache.org/axis/

[2] O WS-I Basic Profile 1.0 consiste em um conjunto de especificações para serviços *Web* juntamente com emendas às especificações que promovem interoperabilidade.

[3] JAX-RPC 1.1 é uma API JAVA para RPC baseado em XML que permite aos desenvolvedores Java construir serviços *Web* utilizando a funcionalidade RPC de acordo com a especificação SOAP.

uma vez criadas as interfaces Java dos serviços. Juntamente com este *framework*, existe uma ferramenta chamada SOAPMonitor, que intercepta mensagens SOAP e as torna visíveis aos usuários.

Porém a versão utilizada do Axis apresenta alguns problemas na geração de classes através de WSDL que utilizam o elemento de grupo *choice* usado para definir esquemas XML, que permite que somente um dos seus subelementos esteja no documento XML definido por este esquema[4].

5.2.2 Infra-estrutura SPKI

O suporte a infra-estrutura SPKI/SDSI é obtida através da biblioteca JSDSI2.0, implementada por (Morcos, 1998). Esta biblioteca JSDSI2.0, uma implementação na linguagem Java do SPKI/SDSI, contém classes que fornecem meios de converter e criar *S-expressions* e implementar os objetos fundamentais do SPKI/SDSI, como certificados e chaves. Esta biblioteca foi escolhida por se tratar de uma implementação mais completa, já que existe a possibilidade de criação de pares de chaves sem a necessidade de utilizar outras ferramentas, e principalmente por esta ter sido desenvolvida em Java. Além da criação dos pares de chaves, a biblioteca também apresenta uma ferramenta gráfica que permite o gerenciamento de certificados, a criação de assinaturas, como também a verificação de cadeias de certificados.

Para que a infra-estrutura SPKI/SDSI se tornasse mais flexível e eficiente para ser usada pelo serviço XKMS de forma a agilizar a busca de cadeias de certificados SPKI e a validação das mesmas, esta biblioteca foi acrescida de algumas extensões. Toda a habilidade de se trabalhar com objetos SPKI/SDSI no protótipo é garantida através da classe `SPKIResolver` e esta foi desenvolvida dentro do projeto *Cadeias de Confiança (Conteúdos Digitais - CNPq/PROTEM)* [5].

Para o suporte criptográfico, a biblioteca utiliza o *Cryptix32*, que é uma implementação das extensões de criptografia do Java (JCE - *Java Cryptography Extensions*). Porém, o JSDSI só trabalha com chaves RSA, embora a especificação do SPKI/SDSI prevê o uso de outras chaves, como chaves DSA (*Digital Signature Algorithm*).

Como os objetos SPKI/SDSI são descritos em *S-expressions* (Rivest, 1997), estes podem ser armazenados no formato ASCII, podendo estar em um arquivo texto ou até mesmo em um banco de dados relacional. A ferramenta presente na biblioteca JSDSI2.0 utiliza como repositório um simples arquivo texto. Para facilitar o gerenciamento dos objetos SPKI/SDSI e agilizar a busca de certificados a biblioteca *parserSxxS*, desenvolvida no projeto Cadeias de Confiança, foi implementada de maneira que todos os objetos suportados pela biblioteca JSDSI pudessem ser convertidos para documentos XML e vice-versa. Para isso, é seguido o DTD (*Document Type Definition*) especificado em (Apparao et al., 1998) e pode ser utilizada como ferramenta para construir aplicações que trabalhem com a infra-estrutura SPKI/SDSI. Assim, a biblioteca permite a completa adoção de documentos XML como uma forma para armazenar objetos SDSI ao invés de *S-expressions*, tornando mais ágil a manipulação destes documentos e ainda permite que aplicações SPKI/SDSI que utilizam documentos

[4] Para contornar este problema foi feita uma modificação na versão contida no CVS do projeto AXIS.
[5] www.das.ufsc.br/seguranca

XML trabalhem perfeitamente com aplicações SPKI/SDSI que só conheçam *S-expressions* (de Melo, 2003).

No contexto das Federações SPKI/SDSI, implementadas no projeto Cadeias de Confiança, há dois tipos de repositórios: os repositórios locais e os repositórios globais (pacotes `repository.local` e pacote `repository.global`) (de Melo, 2003). O pacote `repository.local`, desenvolvidos em Java, situam-se junto de cada membro da federação. São implementados em uma estrutura de arquivos, a qual representa os documentos XML armazenados no repositório e ainda mecanismos para manutenção desta base de dados como a inclusão, modificação, exclusão de arquivos bem como a busca por alguma informação dentro destes arquivos, tendo para isto utilizado o parser SAX (*Simple API for XML*) (Megginson, 1998), presente no J2SE 1.5.

Já para o repositório global, que situa-se junto aos serviços XKMS, foi utilizado o Apache Xindice (Staken, 2002). O Xindice é um banco de dados que armazena documentos XML de forma nativa e utiliza o *XPath* (Clark e DeRose, 1999) como linguagem para recuperação de documentos. Assim o pacote `repository.global` consiste basicamente da implementação de mecanismos que propiciem fazer consultas, inclusões, remoções e modificações rápidas e simples na base de dados, implementada pelo *Xindice*.

Também foi desenvolvida uma classe chamada `RepositorioResolver` para auxiliar na gerência de certificados membros e associados da Federação. Esta classe provê funcionalidades que encapsulam o acesso ao repositório facilitando assim as operações executadas no mesmo.

5.2.3 Qualidade de Serviço

A segurança das interações entre serviços está incluída na camada de qualidade de serviço. Para prover suporte a esta camada, foi escolhido o Apache WSS4J (*http://ws.apache.org/wss4j/*) que provê uma implementação em Java de código da *WS-Security* (Nadalin et al., 2004).

O WSS4J provê recursos para a construção de serviços *Web* de forma interoperável usando o padrão *WS-Security*. O WSS4J utiliza a *engine* do Axis e invoca uma séries de *handlers* na entrada e saída de mensagens SOAP, permitindo assim, uma clara separação da lógica de negócio da lógica do processamento da segurança. Os *handlers* permitem o desenvolvimento e a adição de novas características de segurança com o mínimo de impacto no código e configurações existentes.

Para prover o suporte a assinaturas XML (*XML Signature*), utilizadas nestes *handlers*, é utilizada a biblioteca implementada no projeto *Apache XML Security*. Este projeto foi escolhido por ser um projeto de código aberto, que implementa a especificação W3C de assinatura XML e permite a implementação de extensões da especificação de assinatura XML.

Este projeto requisita a biblioteca *Java Cryptography Extensions* (JCE), que não está incluída com a distribuição padrão. Também é necessária uma versão estável do *Xalan*, que é um processador usado com a XSLT para transformar documentos XML criado pelo *Apache XML Project*. Por ser uma implementação desenvolvida pela *apache foundation*, a confiabilidade e prosseguimento na correção de eventuais *bugs* é garantida.

A especificação W3C de assinatura XML prevê que o elemento *ds:KeyInfo* contenha um subelemento chamado *ds:SPKIData*. Este subelemento é usado para transmitir pares de chave, certificados ou outro dado SPKI. Entretanto a especificação não define como estes dados SPKI são inseridos no *ds:SPKIData*. Por este motivo, neste trabalho, foi definida uma extensão ao esquema XKMS de modo a restringir quais elementos da infra-estrutura SPKI/SDSI podem ser inseridos em uma assinatura XML (Figura 5.2).

```
<element name="SPKIData" type="ds:SPKIDataType"/>
<complexType name="SPKIDataType">
  <sequence maxOccurs="unbounded">
    <element name="SPKISexp" type="base64Binary"/>
    <any namespace="##other" processContents="lax" minOccurs="0"/>
  </sequence>
</complexType>
<element name="SPKISexp" type="SPKISexpType">
<complexType name="SPKISexpType">
  <sequence maxOccurs="unbounded">
    <choice>
      <element name="keyValue" type="ds:KeyValueType"/>
      <element name="tag" type="xsd:tag-content"/>
      <element name="uris" type="xsd:uris"/>
      <element name="authorization-cert" type="xsd:authorization-cert"/>
      <element name="name-cert" type="xsd:name-cert"/>
      <element name="sequence" type="xsd:sequence"/>
    <choice>
  </sequence>
</complexType>
```

Figura 5.2: Extensão do XML Signature Schema

O elemento *ds:SPKIData* contém o elemento *ds:SPKISexp* que, por sua vez contém os subelementos, indicados na Tabela 5.1.

KeyValue	Valor de uma chave pública
tag	Expressão real que pode transmitir autorizações.
uris	Lista de URI.
authorization-cert	Certificado de autorização SPKI.
name-cert	Certificados de nome SPKI
sequence	Seqüência SPKI.

Tabela 5.1: Subelementos do *ds:SPKISexp*

5.2.4 XKMS

A especificação do XKMS prevê diferentes modos de trocas de mensagens entre os serviços XKMS e seus clientes. Neste trabalho, para a troca de mensagens entre o serviço XKMS e os seus serviços XKMS associados, o protocolo assíncrono a duas fases é usado (seção 3.6.3). Este protocolo foi escolhido para que o serviço não fique bloqueado toda vez que recebe uma requisição de um outro serviço XKMS associado e para a proteção contra negação de serviço, respectivamente.

Já para as trocas de mensagens entre os serviços XKMS e seus membros foi usado o protocolo síncrono a duas fases. Novamente o protocolo a duas fases foi escolhido para prover a proteção contra

negação de serviço e a escolha do modo síncrono deve-se a limitação da versão utilizada do Axis que só provê processamento síncrono.

Foi desenvolvida uma classe chamada `RepositorioResolver`, que traz como facilidades: a busca por URIs de detentores de direitos no repositório e as URIs de associados. Também fazem parte de suas funcionalidades o teste, determinar através de uma chave pública se o detentor da mesma é um membro, um associado forte, ou um associado fraco. Estas funcionalidades são necessárias para a implementação do algoritmo implementado na operação de localização. Além destas facilidades, esta classe também provê as funções de gerenciamento de membros, associados e certificados, sendo responsável pelo registro dos mesmos no repositório da Federação.

No capítulo 4, para o serviço XKMS foram introduzidas as operações de localização e de validação e registro. Estas operações foram implementadas no protótipo e, abaixo encontram-se os diagramas de seqüência e alguns detalhes da implementação de cada uma destas operações.

5.2.4.1 Cenários que fazem uso da operação de Localização

O primeiro cenário ocorre quando um cliente requisita ao serviço XKMS a localização do principal que detém a cadeia de certificados necessária para o acesso ao recurso. Neste caso, o cliente envia uma requisição ao serviço XKMS contendo as informações necessárias para a localização e o serviço responde com os valores das uris dos sujeitos aos quais foram delegados os direitos. No segundo cenário, o cliente tenta acessar um recurso e recebe um desafio, contendo uma ACL, para que o mesmo provê a posse do direito de acesso a este recurso. Entretanto, este cliente não sabe como processá-la. Nesta circunstância, o mesmo deve enviar esta ACL ao serviço XKMS. O serviço processa a ACL e a partir de cada entrada, executa a localização da cadeia desejada do mesmo modo que no cenário anterior.

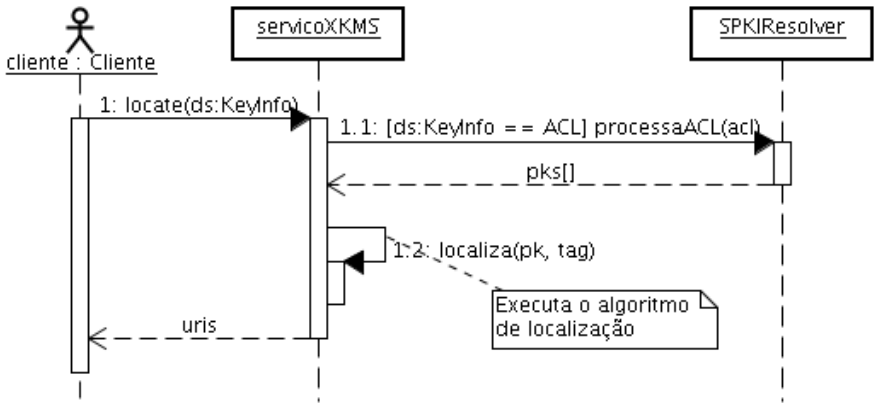

Figura 5.3: Diagrama de Seqüência de Localização

Os detalhes da implementação desta operação podem ser observados no diagrama de seqüência da Figura 5.3. Primeiramente é feita uma verificação do dado recebido, caso seja uma ACL esta é tratada através de uma chamada a classe `SPKIResolver` antes do início da localização do certificado.

5.2.4.2 Cenários que fazem uso da operação de Validação

Este cenário trata da validação de certificados SPKI. Um cliente faz uma requisição ao serviço XKMS enviando uma seqüência contendo uma cadeia de certificados, a chave do sujeito e o direito delegado. Como resposta o serviço XKMS envia o resultado da validação. Na Figura 5.4, encontra-se o diagrama de seqüência desta operação, onde também faz-se o uso do `SPKIResolver` para a validação da cadeia.

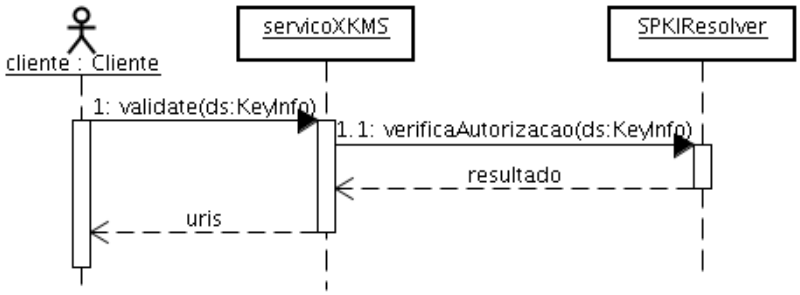

Figura 5.4: Diagrama de Seqüência de Validação

5.2.4.3 Cenários que fazem uso da operação de Registro

Conforme citado no capítulo 4, a operação de Registro pode ser utilizada em três cenários diferentes: filiação de membros, associação de serviços XKMS ou ainda, envio de certificados delegáveis para serem armazenados no repositório da Federação SPKI.

Quando um cliente deseja se filiar a Federação SPKI, o mesmo envia uma requisição de registro contendo o endosso efetivado através do *threshold certificate* assinado por *k*-dentre-*n* membros da Federação em questão. Ao verificar que o conteúdo da mensagem é um endosso para filiação, o serviço XKMS verifica a validade do mesmo. Caso o endosso seja válido, o serviço XKMS emite um certificado de grupo de filiados para o cliente e os demais membros e armazena a chave pública deste novo membro.

Se o endosso é para uma associação, o serviço XKMS verifica a validade do mesmo. Caso o endosso seja válido, o serviço XKMS emite um certificado de grupo de associados para o serviço XKMS solicitante.

Porém se um cliente faz uma requisição ao serviço XKMS enviando uma seqüência contendo uma cadeia de certificados, o serviço XKMS verifica se este é membro da Federação SPKI e, em caso afirmativo, armazena a cadeia no repositório da Federação. A Figura 5.5 mostra aspectos de como esta implementação foi realizada.

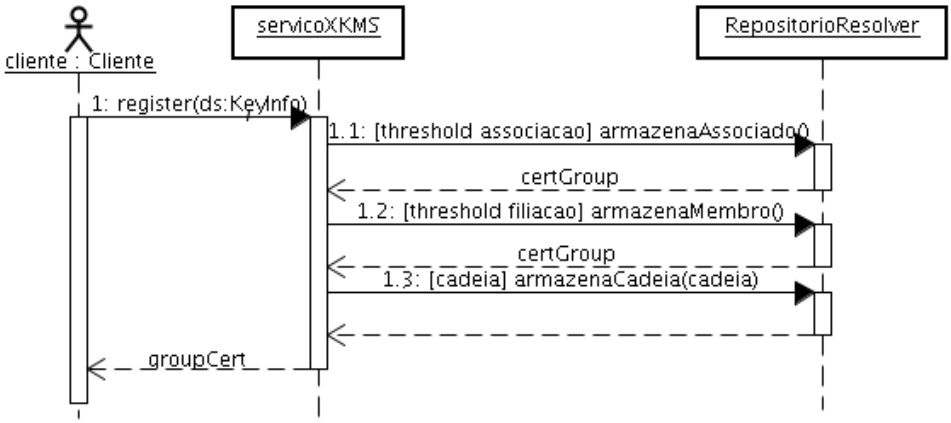

Figura 5.5: Cenário de Registro 2

5.3 Integração do Protótipo a uma Aplicação Distribuída

A idéia da aplicação exemplo foi proposta em (de Melo et al., 2004) e surgiu da atual necessidade em possuir um repositório distribuído de artigos onde se pudesse realizar consultas a textos científicos, informando nome do autor, título do trabalho, etc. e que fosse retornado a maior quantidade de indicações de artigos possíveis sobre um determinado assunto.

O usuário de um aplicativo *stand-alone* pode se autenticar somente uma única vez nesta base distribuída, realizando consultas e, assim, recebendo uma resposta única de informações provenientes de diversos serviços que compõem esta base de informações distribuída. Uma vez autenticado, o usuário é poupado da necessidade de se autenticar ou conhecer os mecanismos de busca nas várias organizações que cooperam neste sistema de informações distribuídas.

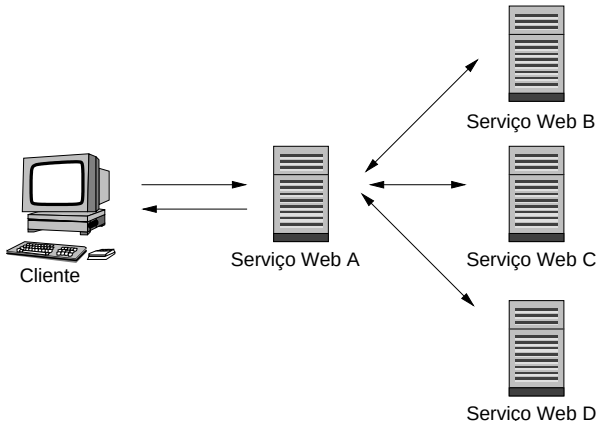

Figura 5.6: Dinâmica da aplicação (de Melo et al., 2004)

Para que isso seja possível é necessário que os serviços *Web* envolvidos possuam alguma relação de confiança, ou seja, que um usuário autenticado em algum destes serviços esteja autorizado a acessar os recursos providos pelos demais, desde que possua os direitos necessários. Essa relação de

confiança pode ser simples, estabelecida em algum momento do passado entre os administradores dos sistemas envolvidos, através do cadastro dos serviços associados, ou poderiam adotar modelos com políticas de negócios complexas as quais poderiam definir taxas para o uso de seus serviços(de Melo et al., 2004).

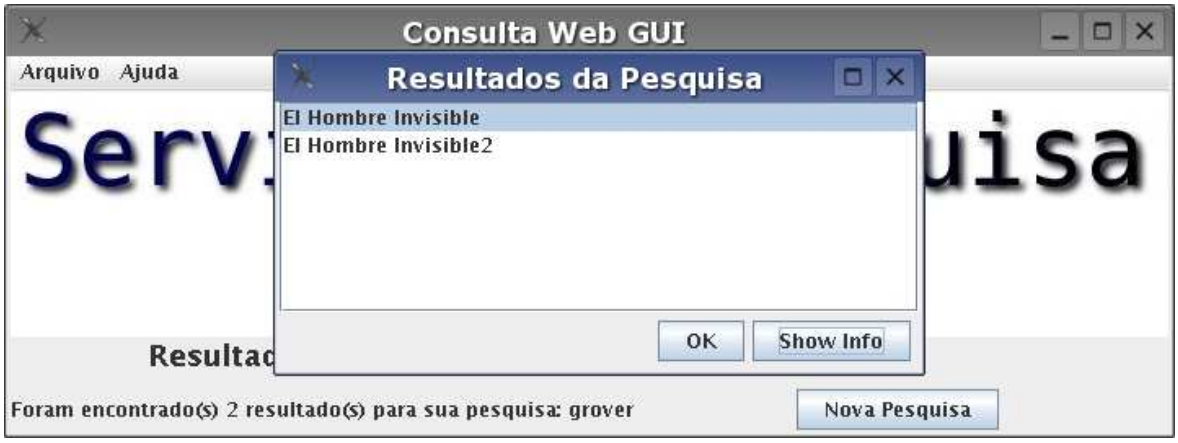

Figura 5.7: Retorno da busca

O serviço *Web* montado para pesquisas de artigos consiste basicamente de uma interface, a qual interage com o usuário, e um motor de buscas, responsável por realizar consultas em bases locais, remotas ou ainda invocando outros serviços *Web*. A dinâmica da aplicação pode ser vista na Figura 5.6. O cliente realiza a busca através da interface do serviço *Web* A e este efetua a consulta em seu repositório local e propaga a consulta para os demais serviços *Web*. Por fim retorna ao aplicativo do usuário uma única lista de ponteiros para todos os artigos encontrados (Figura 5.7).

Quando o usuário selecionar um destes ponteiros retornados, o aplicativo envia uma mensagem de requisição ao serviço *Web* que possui em seu repositório o artigo desejado. Ao receber esta mensagem o serviço *Web* lança um desafio para este usuário para que este prove a posse do direito de acesso a este artigo.

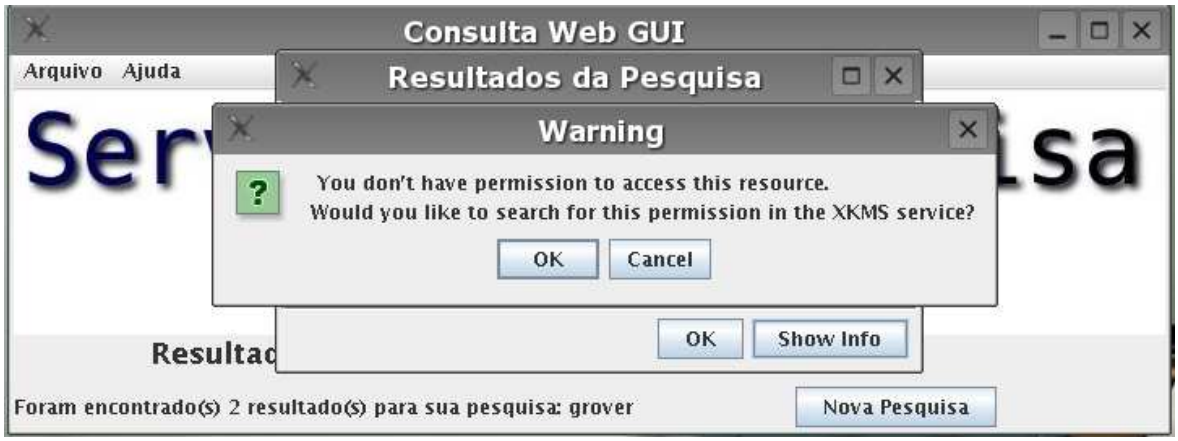

Figura 5.8: Opção de localização no repositório da Federação SPKI

Se a aplicação não encontrar em seu repositório local o direito necessário para o acesso, esta dá

a opção de localização no repositório da Federação o qual o usuário é filiado (Figura 5.8). Caso o usuário aceite esta opção a aplicação envia uma requisição de localização ao serviço XKMS para que este realize a busca da cadeia de certificados que ligue a aplicação cliente ao servidor. O serviço retorna esta cadeia e o usuário pode enfim visualizar o conteúdo do ponteiro retornado pelo serviço *Web* de pesquisa de artigos.

5.4 Considerações

Os resultados observados a partir de textos desenvolvidos e da integração com a aplicação citada comprovou a validade do modelo proposto em aspectos que foram levantados neste texto, como o isolamento das aplicações em relação a gerência de chaves e o uso de padrões.

As operações definidas no capítulo 4 foram implementadas e seguem o comportamento desejado, conforme descrito na seção anterior. Na operação de localização, foi implementado o algoritmo apresentado na seção 4.5. Para auxiliar o serviço XKMS no acesso ao repositório da Federação, foi desenvolvida uma classe `RepositorioResolver` que encapsula o acesso ao mesmo. Também foi utilizada a `SPKIResolver`, para a implementação do cenário 2 da operação de localização, como auxílio no processamento da ACL.

A operação de validação foi implementada de forma que ao receber as seqüências assinadas, é então realizada a validação das mesmas. Novamente foi utilizada a classe `SPKIResolver` como auxílio para a validação. A implementação da operação de registro também faz uso da classe `SPKIResolver` e de funcionalidades da classe que encapsula repositório. Esta classe proveu o auxilio no registro de cadeias de certificados, membros e associados no repositório da Federação.

Existem serviços XKMS, tais como o SQLData XKMS Server v2.0 (SQLData XKMS Server v2.0, 2005), que implementa a especificação XKMS 1.0 e foi desenvolvida em C++. O servidor desta implementação é empacotado juntamente com uma autoridade certificadora e é capaz de emitir, validar e revogar certificados de forma síncrona e assíncrona.

Richard Salz (Richard Salz, 2003) também mostra como implementar um serviço XKRSS da especificação do XKMS 1.0 com as operações *register* e *revoke* na linguagem Python. Nesta proposta que não tem uma CA integrada, para o suporte da PKI é necessário um certificado SSL ou uma chave privada, que sirva de mestre nas emissões de certificados a partir deste serviço. Estas implementações citadas acima, além de suportarem apenas a PKI X.509, seguem a versão 1.0 da especificação XKMS.

O projeto *Markup Security Project* implementa um serviço XKMS que provê suporte para as PKIs X.509 e PGP. Este projeto também incorpora suporte para *hardware* criptográfico, permitindo assim o uso de *smartcards* que provêem a interface PKCS 11[6] tanto no lado cliente como no lado servidor.

Uma outra proposta de serviço XKMS pode ser encontrado em (XKMS Prototype Server, 2005). Este protótipo suporta somente as operações XKISS (*locate* e *validade*) sobre os protocolos síncrono,

[6]Padrão de criptografia de chaves públicas que controla dispositivos de segurança, como cartões inteligentes.

assíncrono, de 2 fases e requisições compostas. Existe também uma outra implementação que envolve a operação *Validate* do XKMS, dirigida à testes de interoperabilidade apresentada em (XKMS 2 Interop Test Site, 2005). Estes protótipos do serviço somente geram o XML das mensagens XKMS, já que o objetivo dos mesmos é apenas um serviço para testes de interoperabilidade.

A ferramenta PHAOS XKMS (*http://www.phaos.com/products/xkms/xkms.html*) implementa a especificação XKMS 2.O e provê suporte para as operações XKISS e XKRSS. Nesta implementação a localização e validação de certificados utiliza entre outras ferramentas como o LDAP e o OSCP respectivamente. Porém esta ferramenta, que dá suporte ao X.509, foi comprada pela ORACLE, que não o apresentou ainda como produto comercial.

Existe também o *Trust Service Integration Kit* (Trust Service Integration Kit, 2005), que é um *kit* de integração de confiança que provê uma API Java para minimizar a complexidade no desenvolvimento de aplicações de confiança. Este *kit* provê uma API para a construção de um cliente XKMS que faz requisições de registro, localização e validação de certificados e chaves. Este *kit* também é baseado na PKI X.509.

Conforme pode ser observado, todas estas implementações encontradas não integram o SPKI ao XKMS, sendo que o protótipo desenvolvido é a primeira implementação a realizar esta integração. Este protótipo traz como benefício além de uma extensão à especificação do XKMS para integração com o SPKI, o suporte à localização de certificados SPKI de forma padronizada.

É importante salientar que nenhuma das experiências, envolvendo o XKMS, citadas acima está disponível em código aberto. Para desenvolver o XKMS foi utilizada a especificação XKMS 2.0 (Hallam-Baker, 2004), porém este não contempla requisições compostas, além das operações revogar, reemitir e de recuperar informações associadas a chaves públicas.

Estas omissões na nossa implementação, como já foi explicado (seção 4.2), se deve ao fato de que não são aplicáveis no manuseio de certificados SPKI.

5.5 Conclusão

Este capítulo descreve a implementação do protótipo e da sua integração a uma aplicação. O objetivo do protótipo era validar a aplicabilidade do modelo de segurança proposto nesta dissertação.

Foram apresentados os cenários de uso do protótipo bem como apresentadas implementações existes na literatura e a comparação entre estas com a implementação do protótipo.

Capítulo 6

Conclusões

Com a crescente evolução da Internet impulsionando a demanda por aplicações distribuídas, criou-se a necessidade de uma arquitetura que fornecesse um meio de integrar diferentes tecnologias e linguagens de programação. Através de serviços *Web*, que seguem a arquitetura AOS, esta integração tornou-se possível, já que estes não impõem restrições de algum tipo de implementação e as trocas de mensagens entre aplicações de corporações que utilizam tecnologias diferentes são realizadas através de protocolos padrões.

Nesta dissertação foram discutidos alguns aspectos da segurança computacional em sistemas distribuídos. Foram também revisados os conceitos fundamentais, as políticas de segurança, as técnicas nas quais os mecanismos de segurança estão baseados e as abordagens de implementação dos mecanismos de autenticação e de autorização.

Foi observado que a escalabilidade limitada e a falta de flexibilidade, indispensáveis em ambientes distribuídos de larga escala são as principais dificuldades encontradas nos modelos que seguem uma abordagem de autenticação centralizada. A habilidade de definir grupos e de delegar autorizações, bem como as facilidades para o desenvolvimento de sistemas computacionais distribuídos escaláveis e seguros, fazem do SPKI/SDSI uma boa opção de suporte para o desenvolvimento de aplicações distribuídas baseadas em redes de confiança. Porém foi observado que o modelo SPKI/SDSI apresenta, como dificuldade, a localização de cadeias de certificados que levem um cliente a obter a autorização necessária para acessar os recursos de um servidor desejado, podendo haver casos em que o cliente não possua o caminho de confiança necessário que o autorize diante do servidor do recurso alvo de acesso.

Para solucionar este problema foi proposta em (Santin, 2004) uma extensão do modelo de confiança do SPKI/SDSI chamada Federação SPKI, que é uma entidade que reúne principais com interesses afins e atua como um agente facilitador na localização de certificados e principais, pois permite o compartilhamento do acesso ao seu repositório de certificados. Através deste compartilhamento, os clientes passam a ter uma alternativa a recorrer quando da falta de cadeias apropriadas para o acesso desejado. Entretanto pôde ser concluído que este modelo de gerência ao preservar a filosofia adotada no modelo SPKI/SDSI no qual o principal que deseja obter acesso a algum recurso é inteiramente

responsável pela busca das cadeias de certificados que lhe forneçam o direito de acesso, sobrecarrega o cliente. Além disso, este modelo não provê um protocolo padrão de acesso as funções de gerenciamento e estabelecimento de confiança providos pelo mesmo.

Neste trabalho foi apresentada uma abordagem para integrar o conceito de Federação SPKI/SDSI aos padrões de serviços *Web* e de extensões XML. Os estudos apresentados sobre serviços *Web* e XML e das extensões de segurança mostraram a necessidade de extensões principalmente nas especificações de assinaturas XML para a adequação destas tecnologias ao SPKI.

Os esforços realizados levaram ao resultado de oferecer a partir do modelo de Federações SPKI/SDSI, funções de gerenciamento aos certificados SPKIs através da tecnologia orientada a serviços, via XKMS. A localização de certificados atribuída ao serviço XKMS poupa o cliente desta tarefa. Este modelo, por se tratar de um serviço XKMS, apresenta um protocolo padrão de acesso as funções de gerenciamento e estabelecimento de confiança providos pelo mesmo.

O protótipo implementado, baseado em uma aplicação que faz uma busca de textos científicos em um serviço de publicação *Web* fictício, enfatiza a viabilidade e a adequação do modelo proposto com a tecnologia de serviços *Web*.

Concluindo então, acredita-se que os objetivos apresentados na seção 1.2 que nortearam este trabalho tenham sido alcançados e que este tenha contribuído de forma simples para o avanço do uso de técnicas e padrões em aplicações programados segundo a ótica de orientação a serviço.

Referências Bibliográficas

Adams, C. e Boeyen, S. (2002). Uddi and wsdl extensions for web service: a security framework. Em *Proceedings of the 2002 ACM workshop on XML security*, páginas 80–89.

Anderson, S. et al. (2005a). Web Services Secure Conversation Language (WS-SecureConversation). Disponível na Internet, última visita em julho de 2005. ftp://www6.software.ibm.com/software/developer/library/ws-secureconversation.pdf.

Anderson, S. et al. (2005b). Web Services Trust Language (WS-Trust). Disponível na Internet, última visita em julho de 2005. ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf.

Apparao, V., Byrne, S., Champion, M., Isaacs, S., Jacobs, I., Hors, A. L., Nicol, G., Robie, J., Sutor, R., Wilson, C., e Wood, L. (1998). Document Object Model Level 1 Specification version 1.0 - W3C recommendation. Disponível na Internet, última visita em agosto de 2005. http://www.w3.org/TR/RECDOM-Level-1.

Aura, T. (1998). On the structure of delegation networks. Em *11th IEEEE Computer Security Foundations Workshop, Rockport, MA USA*.

Bajaj, S. (2003). Web Services Federation Language (WSFederation). Disponível na Internet, última visita em julho de 2005. www-106.ibm.com/developerworks/ webservices/library/ws-fed.

Bajaj, S. (2004). Web Services Policy Framework (WSPolicy). Disponível na Internet, última visita em setembro de 2004. ftp://www6.software.ibm.com/software/developer/library/ws-policy.pdf.

Bartel, M. (2002). XML-Signature Syntax and Processing. Disponível na Internet, última visita em julho de 2004. http://www.w3.org/TR/xmldsig-core/.

BERNERS-LEE, T., FIELDING, R. E., e MASINTER, L. (1998). *Uniform Resource Identifiers (URI): Generic Syntax*. Internet Engineering Task Force RFC 1396.

Bilykh, I., Bychkov, Y., Dahlem, D., Jahnke, J. H., McCallum, G., C. Obry, A. O., e Kuziemsky, C. (2003). Can grid services provide answers to the challenges of national health information sharing? Em *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*, páginas 01–15.

Bishop, M. (2003). *Computer Security*. Addison-Wesley. ISBN 0-201-44099-7.

Booth, D. (2004). Web Services Architecture. Disponível na Internet, última visita em junho de 2004. http://www.w3.org/TR/ws-arch/.

Boyer, J. (2001). Canonical XML. Disponível na Internet, última visita em julho de 2004. http://www.w3.org/TR/xml-c14n.

Bray, T. et al. (2004). Extensible Markup Language (XML) 1.0 (Third Edition). Disponível na Internet, última visita em julho de 2005. http://www.w3.org/TR/REC-xml/.

Cantor, S. et al. (2005). Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0. Disponível na Internet, última visita em julho de 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

Chinnici, R. (2005). Web Services Description Language (WSDL) Version 2.0. Disponível na Internet, última visita em julho de 2005. http://www.w3.org/TR/wsdl20.

Clark, J. e DeRose, S. (1999). XML Path Language (XPath). Disponível na Internet, última visita em julho de 2005. http://www.w3.org/TR/xpath.

Clarke, D. E. (2001). *SPKI/SDSI HTTP Server / Certificate Chain Discovery in SPKI/SDSI*. Tese de Doutorado, MIT.

Clement, L. et al. (2004). UDDI Version 3.0.2. Disponível na Internet, última visita em julho de 2005. http://uddi.org/pubs/uddiv3.htm.

Curbera, F. (2002). Unraveling the web services web: An introduction to soap, wsdl, and uddi. Em *IEEE Internet Computing*, páginas 86–93. IEEE Educational Activities Department.

Damiani, E., di Vimercati, S. D. C., e Samarati, P. (2002). Towards securing xml web services. Em *Proceedings of the 2002 ACM workshop on XML security*, páginas 90 – 96, New York, USA.

Daniel J. Polivy, R. T. (2002). Authenticating distributed data using web services and xml signatures. Em *Proceedings of the 2002 ACM workshop on XML security*, páginas 80–89.

David C. Fallside, P. W. (2004). XML Schema Part 0: Primer Second Edition. Disponível na Internet, última visita em julho de 2005. http://www.w3.org/TR/xmlschema-0/.

de Melo, E. R. (2003). *Redes de confiança em sistemas de objetos CORBA*. Tese de Doutorado, UFSC.

de Melo, E. R., Acajima, G., e Fraga, J. (2004). Integração da arquitetura de segurança dos serviços web com modelos de confiança igualitária. *SSI 2004 - 6° Simposio Seguranca em Informatica*.

Department of Defense (1985). Trusted computer system evaluation criteria. DOD 5200.28-STD.

Dierks, T. e Allen, C. (1999). *The TLS Protocol*. Internet Draft.

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

FastTrack (2001). FastTrack Peer-to-Peer technology company. Disponível na Internet, última visita em agosto de 2005. http://www.fasttrack.nu.

Freier, A. O., Karlton, P., e Kocher, P. C. (1996a). *The SSL protocol - version 3*. Internet Draft.

Freier, A. O., Karlton, P., e Kocher, P. C. (1996b). *The SSL protocol - version 3*. Internet Draft.

Gerck, E. (2000). Overview of certification systems: X.509, CA, PGP and SKIP. Visitado em 03/02/2003.

Gudgin, M. et al. (2003). SOAP Version 1.2 Part 1: Messaging Framework. Disponível na Internet, última visita em julho de 2005. http://www.w3.org/TR/soap12-part1/.

Hallam-Baker, P. (2004). XML Key Management Specification (XKMS 2.0). Disponível na Internet, última visita em julho de 2005. http://www.w3.org/TR/xkms2.

Hughes, J. e Maler, E. (2004). Security Assertion Markup Language (SAML) 2.0 Technical Overview. Disponível na Internet, última visita em julho de 2005. http://xml.coverpages.org/SAML-TechOverviewV20-Draft7874.pdf.

Imamura, T. (2002). XML Encryption Syntax and Processing. Disponível na Internet, última visita em agosto de 2004. http://www.w3.org/TR/xmlenc-core/.

ITU-T (1993). ITU-T recommendation x.509. http://www.mcg.org.br/mirrors/97x509final.doc. Visitado em 22/08/2005.

Kim, J. e Moon, K. (2005). Design of unified key management model using xkms. Em *Advanced Communication Technology, 2005, ICACT 2005*, páginas 77–80.

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

Kraft, R. (2002). Designing a distributed access control processor for network services on the web. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):36–52.

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

Madsen, P. (2004). Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0. Disponível na Internet, última visita em agosto de 2005. www.oasis-open.org/committees/download.php/9886/sstc-saml-exec-overview-2.0-draft-02.pdf.

Medjahed, B., Benatallah, B., Bouguettaya, A., Ngu, A. H. H., e Elmagarmid, A. K. (2003). Business-to-business interactions: issues and enabling technologies. *The International Journal on Very Large Data Bases*, 12(1):59–85.

Megginson, D. (1998). The Simple API for XML. Disponível na Internet. Disponível na Internet, última visita em agosto de 2005. http://www.saxproject.org/.

Mitra, N. (2003). SOAP Version 1.2 Part 0: Primer. Disponível na Internet, última visita em julho de 2005. http:http://www.w3.org/TR/2003/REC-soap12-part0-20030624/.

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

Moses, T. (2005). eXtensible Access Control Markup Language (XACML) Version 2.0. Disponível na Internet, última visita em julho de 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

Nadalin, A. et al. (2004). Web Services Security: SOAP Message Security 1.0 (WS-Security 2004). Disponível na Internet, última visita em julho de 2005. docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

Naedele, M. (2003). Standards for xml and web services security. Em *Computer*, páginas 96 – 98. IEEE Educational Activities Department.

Needham, R. M. e Schroeder, M. D. (1978). Using encryption for authentication in large networks of computers. Em *Communications of the ACM 21(12)*, páginas 993–999.

Neuman, B. C. (1994). *Readings in Distributed Computing Systems*, chapter Scale in distributed systems, páginas 463–489. IEEE Computer Society, Los Alamitos, CA.

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

O'Neill, M. (2003). *Web Services Security*. Brando A. Nordin. ISBN 0-07-222471-1.

Park, N., Moon, K., e Sohn, S. (2003). Xml security: Certificate validation service using xkms for computational grid. Em *Proceedings of the 2003 ACM workshop on XML security*, páginas 112–120.

Richard Salz (2003). Developing a X-KRSS Web Service. Disponível na Internet, última visita em setembro de 2005. http://webservices.xml.com/pub/a/ws/2003/11/25/salz.html.

Rivest, R. (1997). *SEXP (S-expressions)*. Internet Engineering Task Force - Internet Draft.

Rivest, R. L. e Lampson, B. (1996). SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession. http://citeseer.nj.nec.com/rivest96sdsi.html.

Sandhu, R. S. e Samarati, P. (1994). Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40–48.

Santin, A. (2004). *Teias de Federações: uma Abordagem baseada em Cadeias de Confiança para Autenticação, Autorização e Navegação em Sistemas de Larga Escala.* Tese de Doutorado, UFSC.

Santin, A., Fraga, J., Mello, E., e Siqueira, F. (2003). Federation web: A scheme to compound authorization chains. *The 22nd Symposium on Reliable Distributed Systems*.

Services, C. D. S. (2001). The Gnutella Protocol Specification v0.4. Disponível na Internet, última visita em agosto de 2005. http://dss.clip2.com.

SQLData XKMS Server v2.0 (2005). Disponível na Internet, última visita em junho de 2005. http://www.sqldata.com/xkms.htm.

Staken, K. (2002). Developers Guide 1.1. Disponível na Internet, última visita em agosto de 2005. http://xml.apache.org/xindice/guide-developer.html.

Stallings, W. (2000). *Network Security Essentials - Applications and Standards*. Prentice Hall.

Trust Service Integration Kit (2005). Disponível na Internet, última visita em junho de 2005. http://www.verisign.com/developer/xml/.

Verma, M. (2004). XML Security: The XML Key Management Specification. Disponível na Internet, última visita em julho de 2005. http://www-128.ibm.com/developerworks/xml/library/x-seclay3/.

Weerawarana, A., Curbera, F., Leymann, F., Storey, T., e Ferguson, D. (2005). *Web Services Platform Architecture*. Prentice Hall. ISBN 0-13-148874-0.

XKMS 2 Interop Test Site (2005). Disponível na Internet, última visita em junho de 2005. http://vsinterop.entrust.com:7001/verificationserver/index.html.

XKMS Prototype Server (2005). Disponível na Internet, última visita em junho de 2005. http://www.wingso fhermes.org/xkms.html.