REDES DE CONFIANÇA EM SISTEMAS DE OBJETOS CORBA

Emerson Ribeiro de Mello*
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
88040-900 Florianópolis - SC
emerson@das.ufsc.br

Joni da Silva Fraga
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
88040-900 Florianópolis - SC
fraga@das.ufsc.br

Altair Olivo Santin
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
88040-900 Florianópolis - SC
santin@das.ufsc.br

Frank Siqueira
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina
88040-900 Florianópolis - SC
frank@inf.ufsc.br

RESUMO

Este trabalho apresenta um modelo de autenticação e autorização, resultado da integração da infra-estrutura SPKI/SDSI com o CORBAsec. No artigo são apresentadas as principais facilidades providas pelo modelo proposto, demonstrando as vantagens do uso da infra-estrutura SPKI/SDSI. O CORBA adiciona ao modelo as vantagens de objetos distribuídos interoperáveis em ambientes heterogêneos. A idéia sustentada neste artigo é a maior adaptação de redes de confiança, como o SPKI/SDSI, com as características da rede mundial.

ABSTRACT

This work presents an authentication and authorization model that results from the integration of the SPKI/SDSI infrastructure with CORBAsec. The paper presents the main facilities provided by the proposed model, showing the advantages of using the SPKI/SDSI infrastructure. CORBA provides to the model the advantages of interoperable distributed objects in heterogeneous environments. The idea defended in this paper is the better suitability of trust chains, as the SPKI/SDSI, with the characteristics of the world-wide network.

1 Introdução

A Internet provê uma poderosa infra-estrutura de comunicação possibilitando assim o surgimento dos sistemas distribuídos. A ampliação da rede pública e o crescimento cada vez maior do número de aplicações que fazem uso deste meio, tornaram da Internet peça fundamental para os sistemas com abrangência global. Porém, juntamente com as facilidades providas pelas Internet, surgiram novos desafios para o desenvolvimento de aplicações distribuídas, como a flexibilidade, modularidade e escalabilidade, exigindo assim a criação de novas tecnologias. A arquitetura CORBA (*Common Object Request Broker Architecture*), um padrão aberto para sistemas abertos e distribuídos baseados em objetos, surgiu para solucionar tais dificuldades. O *middleware* CORBA fornece funcionalidades para a interoperabilidade e a portabilidade em aplicações distribuídas, tais como a transparência de localização e heterogeneidade de plataforma e de linguagens de programação.

Com o amplo uso de sistemas distribuídos, a segurança da informação tornou-se imprescindível. Aplicações distribuídas devem contar com mecanismos de segurança que garantam propriedades como integridade, confidencialidade, autenticidade e disponibilidade, para que as mesmas possam operar de forma correta e eficaz. Na forma de objetos de serviço, a especificação CORBAsec (*CORBA Security*) surgiu para atender tais requisitos de segurança em implementações de objetos distribuídos suportado por *middlewares* CORBA. O CORBAsec é um modelo capaz de fornecer funcionalidades como identificação e autenticação de principais, controle de acesso na invocação, entre outras.

O modelo clássico de autenticação e autorização em sistemas distribuídos define, em um domínio de nomes, uma autenticação centralizada precedendo a autorização, o qual poderá ter seus controles distribuídos. Tal modelo é adequado para ambientes cujo cliente é conhecido de antemão, porém se mostra inadequado quando isso não é verdade, como no caso de aplicações na Internet.

Desenvolvido para facilitar a concepção de sistemas computacionais escaláveis e seguros, o SPKI/SDSI provê um fino controle de acesso, utilizando espaços de nomes locais e um modelo simples de autorização, baseado em redes de confiança (modelo equalitário).

O presente trabalho mostra como integrar o mo-

*Bolsista CNPq

delo de segurança do CORBA com a infra-estrutura SPKI/SDSI, propiciando um controle descentralizado, de fina granularidade, de políticas de autenticação e de autorização.

Este artigo está organizado da seguinte forma. A seção 2 descreve a infra-estrutura de chave pública SPKI/SDSI. A seção 3 introduz o serviço de segurança do CORBA. Na seção 4 é apresentado o modelo de autorização proposto, integrando as duas tecnologias citadas. Na seção 5 são abordados os aspectos da implementação do modelo. A seção 6 discute os trabalhos correlatos e na seção 7 são feitas as conclusões sobre este trabalho.

2 SPKI/SDSI

A criação do SPKI/SDSI (*Simple Public Key Infrastructure / Simple Distributed Security Infrastructure*) foi motivada pela imperfeição e pela complexidade das infra-estruturas de chaves públicas baseadas em hierarquias de nomes globais, como o X.509. O SPKI/SDSI é a união de duas propostas. Projetado no MIT por Ronald Rivest e Butler Lampson, o SDSI [19] é uma infraestrutura de chave pública que possui como principal característica o espaço de nome local. Criado por Carl Ellison e outros, o SPKI [4] surgiu com o intuito de ser um modelo de autorização simples, flexível, bem definido e de fácil implementação. A união SPKI/SDSI resultou em um sistema de autenticação e autorização para aplicações distribuídas.

No SPKI o *principal*[1] é representado por uma chave pública, e não por um nome de indivíduo[2]. Pois, uma chave pública é mais estável que um nome, em sistemas de larga escala, por exemplo.

Cada *principal* SPKI é detentor de um par de chaves o que lhe permite criar, assinar e divulgar certificados tal como uma autoridade certificadora (CA) X.509. A versão 2.0 do SPKI/SDSI define dois tipos de certificados: certificado de nome e certificado de autorização. No SPKI/SDSI para cada chave pública existe um espaço de nomes local associado. Um nome local é um par constituído de uma chave pública e um de identificador arbitrário, o certificado de nome (veja figura 1 - 1.a) tem como objetivo publicar nomes locais, permitindo assim que outros principais possam chegar à chave pública do nome definido localmente.

Um certificado de nome pode associar um nome à uma chave pública, a outro nome dentro do espaço

Figura 1: Certificados SPKI/SDSI

Emissor	Chave pública que representa o emissor do certificado, de quem a assinatura deve acompanhar o certificado
Nome	Nome local arbitrário o qual irá identificar o sujeito
Sujeito	Uma chave pública ou um nome constituindo de uma chave pública seguida de um ou mais identificadores
Validade	Período pelo qual o certificado é considerado válido

1.a – Certificado de nome

Emissor	Chave pública que representa o emissor do certificado, de quem a assinatura deve acompanhar o certificado
Sujeito	Uma chave pública ou um nome constituindo de uma chave pública seguida de um ou mais identificadores
Tag	É a especificação da autorização que será concedida pelo emissor ao sujeito
Bit de delegação	Campo lógico que indica se o certificado poderá ser propagado ou não
Validade	Período pelo qual o certificado é considerado válido

1.b – Certificado de autorizacão

de nomes local do próprio emissor, ou ainda, a outro certificado de nome em um espaço de nomes de outro *principal*, formando assim as correntes de certificados. A divulgação de nomes no SPKI pode ser feita através de redes de confiança, formadas por certificados de nomes ligados por encadeamento de referências (*linked names*).

O certificado de autorização (veja figura 1 - 1.b) concede uma autorização específica, dada pelo emissor do certificado, ao sujeito do certificado. A autorização concedida pelo emissor ao sujeito pode ser delegada, por este sujeito, integralmente ou parcialmente para outros principais caso o *bit* de delegação esteja ativo. Ao permitir a delegação do certificado, o emissor assume que o sujeito é de extrema confiança, pois o mesmo estará apto a propagar os direitos recebidos a qualquer outro *principal*, sem que haja a necessidade de consultar o emissor.

Assim, esse modelo de confiança possibilita a construção de correntes de autorização que partem da chave do guarda do serviço e terminam nas chaves dos principais que desejam obter acesso ao serviço. Um *principal* que deseja propagar um direito recebido, deverá anexar o certificado recebido à uma seqüência de certificados e enviar a mesma para o sujeito, o qual deseja conceder o direito.

O SPKI é uma infraestrutura orientada a chaves e assim sendo, para o processo de controle de acesso é necessário que a cadeia de nomes seja resolvida a fim de se obter a chave pública do *principal* em questão. Para que a cadeia seja considerada válida é necessário que a primeira chave da cadeia seja a chave pública do detentor do recurso e que a última chave da cadeia seja a chave pública do *principal* que deseja obter acesso ao recurso.

[1] O termo *principal* geralmente se refere à pessoa que é representada pela chave pública, mas pode referenciar uma instituição, uma máquina, etc, a serviço do primeiro. O *principal* é sempre reconhecido como a entidade autorizada pelas políticas de segurança do sistema.

[2] Conhecido como *keyholder* (detentor da chave) no SPKI

Uma lista de controle de acesso (ACL - *Access Control List*) SPKI/SDSI é composta por várias entradas, as quais assemelham-se a um certificado de autorização tendo o diferencial de não possuir o emissor do certificado, logo, também não são assinadas. Uma entrada em uma ACL é dita como uma delegação de direitos do emissor, dono do serviço, para o sujeito da entrada [11]. O período de validade da entrada é especificado pelo campo de validade e caso o campo não seja especificado, assume-se o período de validade indo de $-\infty$ até $+\infty$.

3 Serviço de Segurança CORBA (CORBAsec)

3.1 CORBA *security*

A especificação CORBAsec [15] foi projetada para atender requisitos de segurança em sistemas distribuídos. O modelo CORBA de segurança é especificado na forma de objetos de serviço (COSS), e provendo segurança para as informações e aplicações expressas como objetos distribuídos.

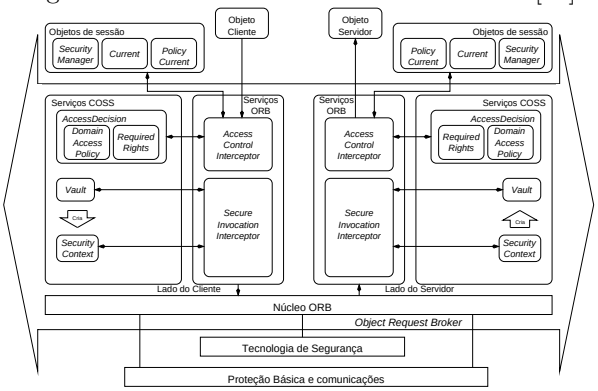
A especificação de segurança CORBA define um modelo capaz de fornecer funcionalidades como identificação e autenticação de principais, controle de acesso na invocação de um método remoto, comunicação segura (incluindo cifragem para garantir a confidencialidade bem como a integridade), não-repudiação[3], auditoria e administração.

No modelo CORBA de segurança os objetos são relacionados em quatro níveis (figura 2): nível de aplicação, o qual compreende os objetos de aplicação; nível de *middleware*, composto pelos objetos de serviço, serviços ORB e núcleo do ORB; nível de tecnologia de segurança, formado pelos serviços de segurança subjacentes; e o nível de proteção básica, composto por uma combinação de hardware e sistemas operacionais locais.

Note que o CORBAsec em si não provê qualquer mecanismo de segurança , por exemplo cifragem de dados. Pelo contrário, o CORBAsec provê somente uma arquitetura e interfaces padronizadas que podem ser utilizadas para dar segurança em um ambiente de objetos distribuídos CORBA. Os mecanismos de segurança devem ser providos separadamente. Como mecanismos de segurança podemos ter o Kerberos, SPKM, SESAME e SSL. O estudo destes mecanismos foge do escopo deste documento e uma introdução a respeito pode ser obtida em [6].

A segurança no modelo CORBAsec é provida em dois níveis: nível 1 (*security level 1*), também conhecido como nível do ORB, o qual fornece segurança para aplicações não cientes da segurança, garantindo controle de acesso aplicado pelo ORB e auditoria de eventos do sistema; nível 2 (*security level 2*), também conhecido como nível de aplicação, que além de suportar as funcionalidades do nível 1 possui outras interfaces de aplicação e administração. O nível 2 é utilizado por aplicações cientes da segurança, permitindo que estas implementem a segurança de forma personalizada.

Figura 2: Modelo estrutural do CORBAsec [22]



Um *principal* deve primeiro estabelecer seus direitos para então acessar objetos remotos. O objeto *PrincipalAuthenticator* implementa o serviço de autenticação no CORBAsec, retornando as credencias para o contexto do *principal*. O nível 2 permite que o *principal* mude os privilégios nas credencias e ainda, permite que escolha quais credenciais serão utilizadas na invocação.

O controle de acesso no nível do ORB é realizado de forma transparente para as aplicações. Durante uma invocação, ou durante o tempo de ligação (*bind time*), o interceptador de controle de acesso aciona o objeto *AccessDecision* que, através do método chamado *access_allowed*, determina se a invocação do cliente deverá ser permitida. Para isto é feito um confronto dos direitos fornecidos pelo cliente, através do objeto *Credentials*, com os direitos necessários (objeto *RequiredRights*).

Além dos interceptadores de controle de acesso que atuam durante uma invocação, o modelo CORBAsec também define os interceptadores de chamada segura, os quais fazem uma interceptação de mais baixo nível no sentido de fornecer propriedades de integridade e de confidencialidade nas transferências de mensagens correspondentes à invocação.

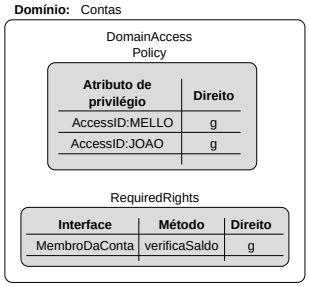
3.2 Considerações sobre o CORBAsec

A forma como é realizado o controle de acesso no CORBAsec impõe ao modelo certas restrições de

[3]Esta é opcional e está disponível somente para o nível 2, ou seja, somente para aplicações cientes da segurança

granularidade. O controle de acesso consiste basicamente no confronto dos direitos fornecidos aos atributos de privilégio (objeto *DomainAccessPolicy*), contra os direitos requeridos (objeto *RequiredRights*). No objeto *DomainAccessPolicy* são definidos os direitos que cada atributo de privilégio possui dentro de um determinado domínio. Atributos os quais são fornecidos aos principais logo após estes ingressarem no sistema de objetos CORBAsec. Já o objeto *RequiredRights* define os direitos necessários para que a invocação, a um determinado método de uma interface, seja honrada. Veja o exemplo, ilustrado pela figura 3.

Figura 3: Controle de acesso no CORBAsec



Aos atributos de privilégio "*AccessID:MELLO*" e "*AccessID:JOÃO*" é dado o direito "*G*" sobre o domínio "*Contas*". No objeto *RequiredRights* é definido que o direito "*G*" se faz necessário para invocação do método "*verificaSaldo*" da interface "*MembroDaConta*". Desta forma, somente os principais da interface "*MembroDaConta*", dentro do domínio "*Contas*" e que tenham um atributo de privilégio que contenha o direito "*G*", poderão invocar o método "*verificaSaldo*".

Porém, no caso do exemplo, nada impede que principais autorizados acessem todas as contas do domínio. Todas as contas, no que se refere ao método "*verificaSaldo*", pertencentes ao domínio estariam acessíveis a qualquer *principal* pertencente ao mesmo domínio que possuísse o atributo "*G*". Tal fato ocorre porque, os direitos fornecidos visam o domínio e os direitos requeridos visam os métodos, ocasionando um problema de granularidade. Em [7] são propostas algumas soluções para tal problema:

1. A criação de um domínio separado para cada cliente. Tal maneira permite que o controle de acesso seja realizado completamente pelo CORBAsec, porém não se trata de uma solução escalável e modificações nas políticas de segurança geram modificações em cada um dos domínios, tornando assim complexa a tarefa de administração.

2. Impor um controle de autorização na lógica das implementações dos objetos. Desta forma um único domínio conteria todos os objetos e não mais teria os problemas relacionados com a escalabilidade. Porém, com a transferência da responsabilidade sobre o controle de autorização para os desenvolvedores de aplicação, qualquer alteração das políticas de segurança implicaria na atualização dos objetos de aplicação.

3. A imposição de um servidor de autorização, como por exemplo, o RAD (*Resource Access Decision*) [16]. Assim, os objetos delegam as verificações ao servidor de autorização centralizado. Com a centralização lógica das regras de autorização ganha-se facilidades para modificar as políticas de segurança, sem que esta afete as aplicações. Contudo, por se tratar de uma abordagem centralizada, esta estará submissa a problemas de escalabidade, desempenho e disponibilidade.

4 Modelo de integração SPKI/SDSI CORBAsec

Nesta seção são descritos os meios que fornecerão os controles de autorização e autenticidade SPKI no CORBAsec. Neste trabalho o SPKI/SDSI é estendido com o uso do conceito de Federações [20], cuja finalidade é agrupar principais com objetivos comuns, para que possam compartilhar certificados SPKI/SDSI.

4.1 Federações SPKI/SDSI

Em [20], [21] foi definido um modelo de confiança que visa o agrupamento de principais que possuam interesses afins. O objetivo das Federações SPKI/SDSI é propiciar facilidades na redução de certificados e na construção de novas cadeias, utilizando para isso o compartilhamento de certificados entre seus membros.

As federações são compostas por três entidades: clientes, servidores (de aplicação) e o gerente de certificados da federação. O papel do gerente é facilitar a interação entre clientes e servidores. O gerente provê mecanismos de armazenamento, recuperação bem como mecanismos para criação de cadeias de certificados de autorização, necessários para que os clientes consigam efetivar o acesso nos servidores, sem no entanto se caracterizar como uma chave intermediária.

Um *principal*, sendo ele um cliente ou um servidor, ao ingressar em uma federação fornece os certificados de nomes e de autorização (pertencentes a ele e que possam ser delegados) que julgar úteis para a federação em questão. Dessa forma, o repositório do gerente da federação conterá todos os certificados, delegáveis, dos membros da federação. Essa

base de certificados serve de apoio nas buscas realizadas por principais que desejam obter acesso a um determinado e recurso porém não possuem os direitos necessários. Uma vez encontrado o *principal* detentor dos direitos é feito então uma negociação entre os principais, o que deseja obter direitos e o que possui os direitos, para que estes direitos sejam delegados. A negociação pode ser constituída de uma simples transferência dos direitos até algo mais complexo como a venda dos mesmos; isto tudo dependendo da semântica envolvida na aplicação.

Um *principal* pode filiar-se a quantas federações ele for aceito, sendo que deverá fornecer um *threshold certificate*, assinado por k-de-n membros já filiados a federação em questão, acompanhado do certificado de nome deste *principal*, o qual está solicitando a admissão. O objetivo do cliente em filiar-se a várias federações é obter as facilidades de acesso propiciadas pelos certificados de autorização dos membros de cada federação. A filiação do cliente em várias federações pode ser necessária para que o mesmo obtenha uma maior presença na rede mundial.

Contudo, em um sistema de larga escala juntamente com as diversas filiações surge o problema da escalabilidade. O modelo trata o problema da escalabilidade através de cadeias de confiança estabelecidas pelas associações entre os gerentes de diferentes federações. As associações entre as federações são feitas visando suprir as necessidades de escala dos membros de cada federação. Dessa maneira, não mais seria necessário que clientes e servidores associem-se a inúmeras federações ao mesmo tempo para possuírem abrangência na rede mundial.

4.2 O uso de elementos CORBAsec nesta proposta

A arquitetura CORBA e seu modelo de segurança (*CORBAsec*) foram utilizados para obter as facilidades de comunicação em sistemas distribuídos, garantindo a interoperabilidade e a segurança na comunicação.

Implementado no nível de aplicação, o modelo de autorização proposto utiliza como suporte alguns objetos CORBAsec no nível do ORB: *SecurityManager*, *PrincipalAuthenticator*, *Credentials*, *Current* e *AccessDecision*. São utilizados também os objetos do nível do ORB responsáveis por estabelecer uma comunicação segura: interceptadores de chamada segura, *Vault* e *SecurityContext*. Porém, os controles de acesso são concentrados no nível da aplicação, sugerindo então o uso das especificações CORBAsec referentes ao nível 2 de segurança.

O modelo proposto para a integração do SPKI/SDSI com o CORBAsec, visa respeitar por completo a filosofia do modelo SPKI/SDSI 2.0, onde o *principal* que deseja obter acesso a algum recurso é inteiramente responsável pela busca das cadeias de certificados que lhe forneça o direito de acesso ao recurso, deixando ao *principal* (provedor de recursos) apenas a função de verificar a autoridade e autenticidade do pedido. Essa verificação é feita da seguinte forma: na tentativa de acesso ao recurso, os direitos fornecidos juntamente com o pedido são confrontados pelo *guarda do serviço* (monitor de referência) com os direitos requeridos, expressos em uma lista de controle de acesso (*ACL*), determinando assim se o acesso ao recurso deve ou não ser garantido ao solicitante.

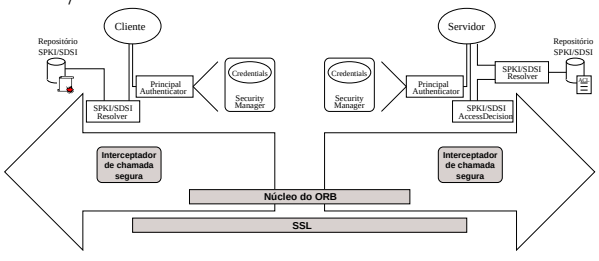
O modelo proposto, portanto, usa as especificações do nível 2 do CORBAsec no sentido de manter as características do SPKI/SDSI intocadas (controles de segurança a nível de aplicação). Este modelo de integração SPKI/SDSI-CORBAsec constitui também uma solução para os problemas de granularidade identificados no CORBAsec, apresentados na seção 3.2

4.2.1 Dinâmica do trabalho proposto

De acordo com o modelo proposto, um objeto cliente ao ingressar no sistema de objetos *CORBAsec* deverá autenticar-se. Essa autenticação é feita através do objeto *PrincipalAuthenticator*, que por sua vez cria o objeto *Credentials*, o qual conterá o conjunto de atributos de privilégios do cliente que está se autenticando e que ficará armazenado no objeto de sessão *SecurityManager*. Todos esses objetos são ilustrados na Figura 4.

A fim de garantir a integridade e a confidencialidade das mensagens, é utilizado como tecnologia de segurança subjacente, o SSLv3. Assim, o processo de autenticação no modelo de segurança do CORBA é feito através da leitura de certificados SSL, sendo que estes serão certificados auto-assinados traduzidos de certificados de nomes SPKI/SDSI, conforme o especificado em [4]. Os certificados SSL, obtidos desta maneira, são usados na autenticação mútua e no estabelecimento de uma sessão segura (canal criptográfico) entre cliente e servidor.

Figura 4: Objetos do modelo de integração SPKI/SDSI - CORBAsec



Toda a habilidade de trabalhar com objetos SPKI/SDSI, como por exemplo, gerar cadeias de

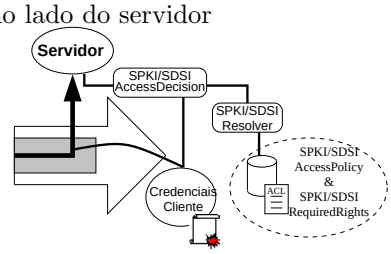
certificados ou verificar a autoridade de uma cadeia, é conseguida pela aplicação através do objeto Java *SPKI/SDSI Resolver* (figura 4). Este objeto encapsula a biblioteca 2.0 do SPKI/SDSI.

No modelo de autorização proposto o confronto entre os atributos de privilégios do objeto requerente com os atributos de controle, os quais protegem o objeto requisitado é feito de uma forma diferenciada, pois a autenticação no sistema de objetos e as credenciais criadas por este processo não serão utilizadas para garantir a autorização de acesso. A autorização de acesso será feita a partir de certificados de autorização SPKI/SDSI, os quais utilizarão credenciais CORBAsec para serem transportados do cliente para o servidor. De posse destes certificados o servidor aciona o guarda do serviço o qual é o responsável por fazer o confronto entre os direitos requeridos e os direitos fornecidos.

No lado do servidor, o objeto *SPKI/SDSI AccessDecision* (figura 5) é responsável por decidir se as requisições ao objeto servidor deverão ser honradas. As *ACLs* SPKI/SDSI estariam fazendo o papel tanto do objeto *RequiredRights* do CORBAsec, pois definem os direitos requeridos ou necessários para possibilitar a invocação de cada operação na interface, quanto do objeto *AccessPolicy*, pois fornecem a um conjunto de principais, um conjunto de direitos na execução de operações dos objetos de um domínio.

No nível 2 do CORBAsec [15], aplicações cientes da segurança podem controlar as opções de segurança utilizadas nas invocações [15]: podem escolher a qualidade da proteção das mensagens, mudar os privilégios contidos nas suas próprias credenciais, escolher quais credenciais serão utilizadas na invocação de um objeto e se estas credenciais só poderão ser utilizadas no objeto destino ou se também poderão ser delegadas. Com base nestas afirmações optou-se em disponibilizar os certificados de autorização, pertencentes ao cliente e necessários para obtenção de direito de acesso aos recursos distribuídos, na forma de credenciais. As credenciais do cliente são reconstruídas no lado do servidor e este as obtém através do objeto *ReceivedCredentials* (veja figura 5). A flexibilidade oferecida pelo nível 2 do CORBAsec no manejo com credenciais é plenamente adequada às necessidades de processamento sobre certificados SPKI/SDSI.

Figura 5: Objetos utilizados para o controle de acesso no lado do servidor



4.2.2 Adicionando e recuperando certificados nas credenciais

Um objeto ao ingressar no sistema de objetos CORBAsec fará sua autenticação no sistema através do objeto *PrincipalAuthenticator*. A autenticação é feita através da leituras de certificados SSL, uma vez que a tecnologia de segurança utilizada no protótipo é o SSLv3. Porém, as credenciais criadas no processo de autenticação no momento da entrada no sistema de objetos do CORBAsec (veja seção 3.1) não serão utilizadas para o controle de acesso.

No nível 2, o cliente tem a habilidade de definir quais credenciais utilizar em suas invocações. Para cada cadeia de autorização, montada pelo cliente para cumprir os requisitos dos desafios lançados a ele, é criado um objeto *SPKI/SDSI Credentials*[4] onde as cadeias, transformadas em uma simples seqüência de *bytes*, serão adicionadas como um atributo de privilégio nas credenciais definidas no CORBAsec.

No lado do servidor, o objeto *SPKI/SDSI AccessDecision* é automaticamente chamado a cada requisição de método. Este objeto faz o confronto dos direitos fornecidos pelo cliente com os direitos necessários, especificados na ACL. Os direitos do cliente (cadeia de certificados) são obtidos através do objeto *SPKI/SDSI ReceivedCredentials*, o qual está contido no objeto de sessão *SecurityManager* (figura 4).

4.2.3 Lista de controle de acesso

O SPKI/SDSI é um sistema prático dirigido ao problema de assegurar que um usuário esteja autorizado a executar uma ação e não apenas ao problema de identificar o usuário. Este foco permite uma maior flexibilidade no compartilhamento de recursos através das delegações, indo em contraste com os sistemas de autenticação baseados em infraestruturas convencionais de chaves públicas juntamente com a autorização através de *ACLs* convencionais, construídas a partir de nomes de principais.

Em infraestruturas convencionais de chaves públicas, se um servidor S deseja verificar se o *principal* U realmente possui direitos para acessar o serviço, é necessário que uma Autoridade Certificadora (CA), a qual o servidor tenha uma relação de confiança, emita um certificado de autorização que contém as permissões e o nome do *principal* pretendente ao acesso, *(U)*, e que, do lado do servidor, a identificação única do mesmo

[4]Trata-se de um objeto CORBAsec *SecurityLevel2:Credentials*, mas com funcionalidade específica para o controle de acesso com certificados SPKI/SDSI

(nome global do cliente) esteja contida na *ACL* do servidor.

No SPKI não existe uma Autoridade Certificadora, e cada *principal* está apto a emitir certificados de autorização, podendo então o próprio servidor *S* emitir um certificado de autorização para o *principal U* e assim o uso da *ACL* se torna desnecessário [13], já que no momento do acesso *U* informa os certificados delegados necessários que possui, e com isto basta ao servidor verificar a autenticidade do pedido e os direitos que este possui.

O modelo proposto utiliza *ACLs* a fim de propiciar uma maior facilidade na busca das cadeias de certificados válidas que deverão ser encontradas pelo cliente, dentro do âmbito das *Federações SPKI/SDSI* proposta em [20]. Uma *ACL* é composta por várias entradas e cada entrada é composta por uma chave pública e uma *tag*. As *tags* poderiam ser, por exemplo, os nomes dos respectivos métodos que o recurso disponibiliza, permitindo ao servidor garantir um controle individual de seus métodos.

Os certificados de autorização são utilizados de forma que possam ser delegados, permitindo assim que outras entidades tornem-se emissoras de autorização, idéia que é utilizada nas *Federações SPKI/SDSI*, onde um membro ao ingressar em uma federação fornece todos os seus certificados de autorização (relevantes à associação), recebidos de outras entidades e que possam ser delegados, através do gerente da federação. O gerente, sendo passivo (não caracterizando uma chave pública), não participa da cadeia de confiança. Sua função é a de um simples repositório.

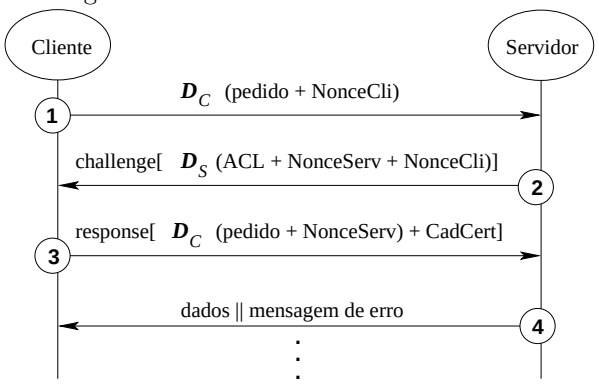
4.2.4 Autenticação mútua e autorização no modelo proposto

Um dos princípios básicos da segurança é garantir que as informações protegidas só serão reveladas a entidades autorizadas e autênticas. Sistemas criptográficos, sendo eles simétricos ou de chaves públicas, provêem meios diferentes para garantir estes princípios. Uma vez garantida que a entidade envolvida na comunicação está autorizada e o pedido é autêntico, ou seja, realmente originado por esta entidade, as informações protegidas deverão ser fornecidas à entidade solicitante.

O estabelecimento de uma comunicação segura exige um conjunto de trocas entre as partes comunicantes. O protocolo *challenge/response*, ou desafio e resposta, é um protocolo de autenticação mútua normalmente empregado em infraestruturas de chaves públicas. Este protocolo dá ao cliente a certeza de estar conectando-se ao servidor desejado, e ao servidor a certeza de estar recebendo requisições de um cliente válido.

A comunicação entre cliente e servidor no modelo proposto, utiliza como base para a autenticação dos seus respectivos principais, o protocolo *challenge/response* baseado em certificados de autorização SPKI/SDSI e ACLs. Considere o seguinte cenário: um cliente deseja invocar um método de um objeto servidor. Esse método está protegido por uma ACL SPKI/SDSI e, caso o pedido original não possua os requisitos necessários para o acesso, o cliente deverá vencer um desafio para que a invocação seja honrada. Os passos envolvidos neste protocolo, entre cliente e servidor, são ilustrados pela figura 6 e descritos logo abaixo.

Figura 6: Protocolo de controle de acesso



No passo 1, o cliente faz uma requisição acompanhada de um *nonce*[5] (*NonceCli*) ao servidor sem fornecer qualquer cadeia de autorização, pois o cliente desconhece os direitos requeridos. De posse do pedido, o servidor aciona o guarda do serviço, que é o responsável por confrontar os direitos requeridos, contidos na ACL, com os direitos fornecidos. Como o cliente não forneceu os direitos necessários, o servidor gera um desafio (*challenge*), assinado por sua chave privada, ao cliente (passo 2). O desafio é composto pela ACL[6] que protege o recurso, pelo *nonce* enviado pelo cliente (*NonceCli*) e por um outro *nonce* (*NonceServ*), este gerado pelo servidor.

Com a chave pública do servidor [7], o cliente verifica a autenticidade do desafio e, caso essa seja garantida o cliente aciona os dispositivos necessários para gerar a cadeia de certificados que lhe garanta o direito de acesso ao recurso provido pelo servidor.

Uma vez gerada a cadeia de autorização válida, o cliente faz a resposta (*response*) ao servidor (passo 3). A resposta é composta pelo pedido original, pelo *NonceServ*, ambos assinados pela chave privada do cliente, e também pela cadeia de certificados de autorização.

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

[6] O envio da ACL não traz problemas de segurança, pois as entradas da ACL são compostas por chaves públicas e estas não identificam os seus detentores.

[7] A chave pública do servidor é conhecida de antemão pelo cliente, podendo ser através de um certificado de nome.

O servidor, de posse da chave pública do cliente (essa pode ser obtida pegando-se a última chave da cadeia de certificados fornecida ao servidor), verifica a autenticidade da resposta. Garantida a autenticidade, o servidor aciona o guarda do serviço para que o mesmo verifique se a cadeia realmente concede a autorização desejada, e assim no passo 4 o pedido é honrado, retornando a informação desejada.

É importante salientar que no protocolo descrito acima, a autenticação mútua está toda baseada sobre certificados de autorização SPKI/SDSI. Esta é uma característica da infra-estrutura SPKI/SDSI.

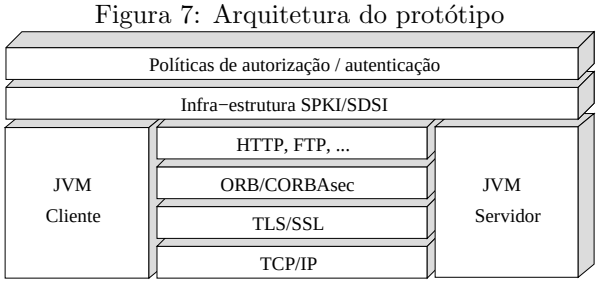
5 Implementação do protótipo

São discutidos nesta seção, os aspectos relacionados à implementação do protótipo de autorização utilizando certificados SPKI/SDSI. A arquitetura do protótipo (figura 7) é composta por ferramentas amplamente utilizadas na internet, ambiente o qual é do contexto do trabalho.

5.1 Arquitetura

A base de suporte SPKI/SDSI é conseguida pelo protótipo através da biblioteca JSDSI2.0 implementada por [12]. Os códigos dos clientes e servidores são interpretados por máquinas virtuais Java (*JVM*). O uso do Java é bem difundido em aplicações distribuídas, pois o mesmo provê facilidades no desenvolvimento de aplicações amigáveis a partir de protocolos de aplicação na Internet.

Figura 7: Arquitetura do protótipo



O CORBA e o CORBAsec fornecem facilidades para a interoperabilidade de aplicações distribuídas, construídas a partir de objetos de sistemas heterogêneos. No protótipo foi utilizado o ORBacus [9] juntamente com o ORBAsec SL2 [1]. O ORBAsec SL2 implementa somente alguns objetos do nível 2 da especificação CORBAsec, mas esses já satisfazem as necessidades do protótipo.

A proteção das mensagens em trânsito é garantida pela camada TLS/SSL (*Transport Layer Security / Secure Sockets Layer*). Com essa camada é garantida a confidencialidade e a integridade na comunicação em rede entre componentes situados em máquinas físicas diferentes. Foi utilizado, no protótipo, como módulo SSL junto ao ORB o iSaSiLk [8], porém outras implementações do SSL podem ser utilizadas.

5.2 Busca da cadeia de certificados

O primeiro passo do cliente ao receber um desafio é buscar em seu repositório local uma cadeia de certificados que ligue a chave pública do servidor à sua chave pública. Caso tal cadeia não exista, o segundo passo é a busca de cadeias que liguem principais contidos na ACL, enviada pelo servidor com a chave pública do cliente.

A busca da cadeia é feita pelos seguintes parâmetros: *ChavePublicaDoPrincipalDaACL*, *ChavePublicaDoCliente*, *TAG*. A princípio as buscas são feitas na base local de certificados do cliente. Essa busca é repetida N vezes, onde N é o número de chaves públicas contidas na ACL.

Dentro do âmbito das Federações SPKI/SDSI, temos o terceiro passo, caso no repositório local não exista a cadeia de autorização necessária para o satisfazer o desafio, a busca então é feita a partir do gerente de certificados da federação, onde o cliente é membro. Os mecanismos utilizados para busca e geração das cadeias de certificados utilizados pelo gerentes fogem do escopo deste texto; os mesmo podem ser encontrados em [20].

5.3 Repositório de certificados

Os repositórios de objetos SPKI/SDSI podem ser implementados de inúmeras maneiras. Objetos SPKI/SDSI são constituídos por *S-expressions* [18], e assim podem ser armazenados na forma ASCII. Em [17] foi proposta uma forma padronizada de transformar objetos SPKI/SDSI codificados em *S-expressions* para documentos XML (*eXtensible Markup Language*).

Os repositórios locais foram construídos com base no documento citado acima, ou seja, os repositórios locais armazenam documentos XML, estes gerados a partir de objetos SPKI/SDSI codificados em *S-expressions*. Os objetos SPKI/SDSI são transformados em documentos XML para que possam ser armazenados, mas uma vez recuperados da base de dados eles são novamente codificados em *S-expression* para que possam ser trocados entre os objetos de aplicação (objetos CORBA) envolvidos na comunicação, propiciando assim a compatibilidade com qualquer aplicação que trabalhe com objetos SPKI/SDSI.

6 Trabalhos relacionados

Em [3], tem-se um trabalho com um conteúdo bem completo e a implementação da versão atual do SPKI/SDSI. A implementação teve como aplicação um servidor HTTP. O protocolo de controle de acesso adotado, define desafios lançados pelos servidor ao cliente. Foram citadas algumas variações do protocolo visando diminuir o número de trocas de mensagens e o tamanho das mesmas.

Ainda é descrito uma forma para proteger a ACL do recurso contra principais ainda não autenticados, sendo a proteção feita através de uma outra ACL. Dessa forma, o sigilo sobre os direitos que cada *principal* possui sobre o recurso estaria garantido contra usuários ainda não autenticados. Uma análise mais criteriosa verifica que ainda assim é possível que usuários não autenticados obtenham a lista de principais que possuem direitos sobre o recurso. Isto porque a chave dos principais deverá estar contida nas duas ACLs. O mesmo não é verdade para os direitos, pois esses podem ser diferentes em cada ACL.

A abordagem descrita acima possui a inconveniência das inúmeras trocas de mensagens entre o cliente e o servidor, pois cada requisição de método gera dois desafios e duas respostas. O ganho de segurança obtido por tal abordagem não é muito significativo, pois como as chaves públicas SPKI/SDSI não revelam quem são os seus proprietários, impõe ao invasor a dificuldade de localizar os pontos a serem atacados.

Em [5] é proposto um modelo de segurança distribuído para a tecnologia Jini [2], onde o autor cita os desafios de segurança presentes devido à independência de protocolos, destacando também que a arquitetura do Jini não propõe qualquer mecanismo de segurança em adição às facilidades padrões do Java, como por exemplo, a proteção da máquina virtual Java (JVM) do cliente contra códigos maliciosos provenientes do *proxy*.

Eronen e Nikander [5] propoêm uma solução interessante para os problemas de segurança o qual a arquitetura Jini não trata. O modelo aborda basicamente a autenticação de clientes e de serviços, tendo o foco da autorização voltado ao nível das chamadas de método. Neste trabalho é feita uma modificação dos certificados SPKI, onde o campo *TAG* é reescrito para adaptar-se às necessidades do modelo, tornando inválida a interoperabilidade com outros certificados SPKI provenientes de outras aplicações.

Em [10] é apresentada uma forma de implementar autorização em aplicações distribuídas CORBA com certificados SPKI. São discutidas as vantagens da implementação quando comparadas com o controle de acesso do CORBAsec. Lampinen [10] utilizou a versão 1.0 do SPKI, onde a proposta SDSI ainda não tinha sido unida com a proposta do SPKI, formando assim a versão SPKI/SDSI 2.0. O cliente, ao ingressar no sistema de objetos CORBA, disponibiliza todos os certificados de autorização pertencentes a ele em um único objeto *Credentials*. Ao receber uma invocação, o servidor obtém todos os certificados de autorização pertencentes ao cliente e faz a busca por alguma cadeia que forneça ao cliente a autorização desejada.

Tal abordagem além de fugir dos conceitos do SPKI/SDSI 2.0, sofre com o problema da escalabilidade, visto que o cliente poderá possuir uma grande quantidade de certificados, dificultando assim o transporte e o processamento dos mesmos.

Nikander e Partanen em [14] propoêm uma solução escalável para o gerenciamento de controle de acesso seguro e distribuído no JDK 1.2. Para isto são utilizados certificados de autorização para delegar permissões para os módulos Java. No JDK 1.2, as permissões são atribuídas aos módulos de execução através dos domínios de proteção. No trabalho de Nikander, para cada domínio de proteção poderão ser anexados um ou mais certificados SPKI, onde tais certificados descrevem diretamentes as possíveis permissões do domínio. Cada certificado SPKI denota um número de permissões que o emissor do certificado deseja conceder ao domínio. Para isto, fora feita uma expansão do campo *TAG* dos certificados de autorização SPKI, tornando tais certificados úteis somente para o ambiente.

No modelo proposto, focou-se na versão atual do SPKI/SDSI. Onde o *principal*, tido como cliente da comunicação, é o único responsável pela busca da cadeia de autorização necessária ao acesso desejado, restando ao servidor apenas a tarefa de verificar se a cadeia é válida. Propôs-se um modelo de autorização mais flexível para objetos distribuídos. O nível 2 do CORBAsec é utilizado e a questão da granularidade de acesso mal resolvida neste modelo de segurança proposto pela OMG (ver seção 3.1) é resolvida pela integração com o SPKI/SDSI.

O controle de acesso é feito de forma granular e descentralizada. Todos os aspectos da versão atual do SPKI/SDSI são respeitados bem como a utilização dos objetos do nível 2 de segurança do CORBAsec. As facilidades propiciadas aos principais, clientes e servidores, não impõem restrições para implementações do mesmo, garantindo assim o mínimo esforço para os mesmo. O protocolo de autenticação garante proteção contra ataques de mensagens antigas e que as partes envolvidas na comunicação são autênticas.

7 Conclusão

Neste texto são mostrados os esforços na proposição de um modelo de autenticação e autorização flexível

e escalável para sistemas com as características da rede mundial, utilizando certificados SPKI/SDSI como uma alternativa de controle de acesso no modelo de segurança do CORBA. O CORBAsec serve de maneira concreta como um veículo para a introdução do conceito de redes de confiança em programação de objetos distribuídos interoperáveis em sistemas distribuídos de larga escala.

O nível 2 de segurança propiciou as facilidades necessárias para o desenvolvimento do trabalho. Tais liberdades de implementação da segurança na aplicação foram de extrema importância no modelo. O uso de certificados SPKI/SDSI mostrou-se uma ótima escolha para tratar os problemas de escalabilidade e flexibilidade em sistemas distribuídos de larga escala. A implementação dos repositórios em XML propiciou facilidades na busca e geração das cadeias de certificados.

Este trabalho é parte de um projeto que visa um suporte de autenticação e autorização para a distribuição de documentos na Internet.

Referências

[1] *ORBAsec SL2 User Guide*, Julho 2000. Version 2.1.4.

[2] Ken Arnold, Bryan O'Sullivan, Robert W. Schifler, Jim Waldo, and Ann Wollrath. *The Jini Specification*. Addison-Wesley, 1999.

[3] Dwaine E. Clarke. *SPKI/SDSI HTTP Server / Certificate Chain Discovery in SPKI/SDSI*. PhD thesis, MIT, 2001.

[4] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. *SPKI Certificate Theory*. Internet Engineering Task Force RFC 2693, Setembro 1999.

[5] Pasi Eronen and Pekka Nikander. Decentralized Jini Security. In *Network and Distributed System Security Symposium - (NDSS 2001)*, San Diego, California, Fevereiro 2001.

[6] Dieter Gollmann. *Computer Security*. John Wiley & Sons, 1999.

[7] Bret Hartman, Donald J. Flinn, and Konstantin Beznosov. *Enterprise Security with EJB and CORBA*. John Wiley & Sons, 2001. ISBN 0-471-40131-5.

[8] Institute for Applied Information Processing and Communications (IAIK). *iSaSiLk 3 - Reference Manual*, 2000. Version 3.

[9] *ORBacus User Guide*, 2001. Version 3.3.4.

[10] Tuomo Lampinen. Using SPKI certificates for authorization in CORBA based distributed object-oriented systems. In *Proceedings of the 4th Nordic Workshop on Secure IT Systems (NordSec '99)*, pages 61–81, Kista, Sweden, Novembro 1999.

[11] Ningui Li. Local Names in SPKI/SDSI. In *Proceedings of The 13th Computer Security Foundations Workshop*, pages 2–15. IEEE Computer Society Press, 2000.

[12] Alexander Morcos. A Java implementation of Simple Distributed Security Infrastructure. Master's thesis, MIT, Maio 1998.

[13] P. Nikander and L. Viljanen. Storing and Retrieving Internet Certificates. In *Proceedings of the Third Nordic Workshop on Secure IT Systems*, 1998.

[14] Pekka Nikander and Jonna Partanen. Distributed policy management for JDK 1.2. In *Network and Distributed System Security Symposium - (NDSS'99)*, pages 91 – 101, San Diego, California, Fevereiro 1999.

[15] Object Management Group - Security Service v1.7. OMG Document 01-03-08, Março 2001.

[16] OMG. Resource access decision facility specification. OMG Document 01-04-01, Abril 2001.

[17] Xavier Orri and Joan-Maria Mas. *SPKI-XML Certificate Structure*. Internet Engineering Task Force - Internet Draft, Novembro 2001. http://xml.coverpages.org/ni2001-12-18-a.html. Visitado em 02/04/2003.

[18] Ronald L. Rivest. SEXP (S-expressions). Internet Engineering Task Force - Internet Draft, Maio 1997. http://theory.lcs.mit.edu/ rivest/sexp.html.

[19] Ronald L. Rivest and Butler Lampson. SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession, 1996. http://citeseer.nj.nec.com/rivest96sdsi.html.

[20] A. Santin, J. Fraga, E. Mello, and F. Siqueira. Um modelo de autorização e autenticação baseado em redes de confiança para sistemas distribuídos de larga escala. In *Anais SSI 2002*, pages 101–110, São José dos Campos, SP - Brazil, 2002. ITA, SSI.

[21] A. Santin, J. Fraga, E. Mello, and F. Siqueira. Teias de Federações como extensões ao modelo de autenticação e autorização SDSI/SPKI. In *Anais XXI Simpósio Brasileiro de Redes de Computadores*, pages 553 – 568, Natal, RN - Brazil, 2003. SBRC.

[22] Carla Merkle Westphall. *Um esquema de autorização para a segurança em sistemas distribuídos de larga escala*. PhD thesis, Universidade Federal de Santa Catarina - Brasil, 2000.