Infraestrutura de Autenticação e Autorização Baseada em *SmartCards* com Controle de Atributos Centrado no Usuário

Davi da Silva Böger[1], Luciano Barreto[1], Joni da Silva Fraga[1], André Santos[2], Davi Teles França[2]

[1]Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina

[2]Centro de Ciência e Tecnologia, Universidade Estadual do Ceará

{dsboger, lucianobarreto, fraga}@das.ufsc.br,
Andre.Moura@ece.uprm.edu, davi.teles@insert.uece.br

Abstract. *This paper presents an Authentication and Authorization Infrastructure that extends OpenID protocols in order to accomplish SmartCard based authentication and user-centric attribute release control. An Identity Selector component was designed to mediate the communication between the Identity Provider and the SmartCard, besides presenting a user-interface to allow the user to choose which attributes she wants to release to which Service Provider. A prototype of the infrastructure was developed in order to evaluate its feasibility.*

Resumo. *Este artigo apresenta uma Infraestrutura de Autenticação e Autorização que estende os protocolos do OpenID para realizar autenticação baseada em SmartCards e controle de liberação de atributos centrado no usuário. Um componente Seletor de Identidade foi elaborado para mediar a comunicação entre o Provedor de Identidade e o SmartCard, além de apresentar uma interface para o usuário escolher quais atributos deseja liberar a cada Provedor de Serviço. Um protótipo da infraestrutura foi desenvolvido a fim de avaliar sua viabilidade.*

1. Introdução

Grande parte dos sistemas distribuídos existentes na Internet utiliza a abordagem tradicional de gerenciamento de identidade, isto é, o próprio Provedor de Serviço (*Service Provider*, ou SP) implementa todos os controles para o registro, autenticação e autorização de seus usuários [Jøsang e Pope 2005, Jøsang et al. 2005, Bhargav-Spantzel et al. 2007]. Este modelo é bastante limitado, principalmente com o crescimento em número e complexidade dos SPs, pois obriga usuários a gerenciar um número elevado de contas, ou seja, lembrar-se de todos os nomes de usuário e senhas, bem como manter as informações de identidade atualizadas em diversos serviços [Florencio e Herley 2007]. Por outro lado, SPs são encarregados de manter as bases de dados de identidade, bem como toda a infraestrutura de segurança necessária para garantir o sigilo das mesmas. Vários modelos e infraestruturas nos últimos anos têm sido introduzidos onde estes controles tomam forma a partir de um conceito mais amplo que é o de gerenciamento de identidades.

As infraestruturas de gerenciamento de identidades envolvem muitas vezes um sistema de autenticação (usualmente formado em sistemas de larga escala por uma rede de autoridades de autenticação) e um sistema de gerenciamento de atributos (serviço que fornece informações adicionais sobre clientes/usuários). Estes atributos, individual ou coletivamente, podem identificar o usuário (autenticação) e fornecer as informações necessárias (atributos do usuário) para a realização dos controles de autorização (controle de acesso baseado em atributos) que permitem assim, a execução das operações solicitadas pelo usuário. Em muitas propostas e infraestruturas, as autoridades de autenticação também concentram o gerenciamento de atributos e são identificados como provedores de identidades (*Identity Providers* - IdP). Com isto então, em vários modelos de gerenciamento de identidades (dentre os quais destacamos o centralizado, identidades federadas e o centrado no usuário [Jøsang et al. 2005]), a autenticação não é mais realizada nos SPs, mas sim em autoridades de autenticação (ou IdPs). A autorização continua sendo concretizada nos SPs porque estes conhecem seus recursos e políticas (de aplicação) e devem zelar pelos mesmos. Nestas abordagens, a partir da identificação diante de uma destas entidades confiáveis (IdPs ou autoridades de autenticação), outras facilidades ficam disponíveis aos usuários do sistema como, por exemplo, a identificação única (*Single Sign-On* – SSO) para vários acessos subsequentes (e mesmo independentes) a provedores de serviços.

Nos últimos anos houve o crescimento do modelo centralizado [Jøsang e Pope 2005], no qual as informações de identidade são mantidas em entidades centralizadas denominadas de Provedores de Identidade (*Identity Providers*, ou IdP). Provedores como Google e Facebook permitem que seus usuários utilizem suas identidades para acessar SPs distintos, atuando na mediação do processo de autenticação. Nesse modelo a confiança passa a ser centralizada no IdP, tanto para manter o sigilo das identidades dos usuários, quanto para prover informações de autenticação corretas aos SPs.

A especificação *OpenID* [OpenID 2007] se tornou um padrão de facto após a adoção por grandes provedores (p.ex. Google, Microsoft, Facebook, etc.) para construir seus IdPs centralizados. O *OpenID* é baseado nas tecnologias da Web (basicamente URL e HTTP), o que torna os protocolos leves e a implementação relativamente simples. Além disso, é possível integrar os protocolos do *OpenID* com soluções de autenticação avançadas e com modelos de gerenciamento de identidade centralizado e centrado no usuário.

A grande maioria dos IdPs atuais utiliza nome de usuário e senha como mecanismo de autenticação. Esse mecanismo apresenta diversos problemas para a segurança, pois depende da capacidade humana, em geral bastante limitada, de elaborar e lembrar-se de palavras secretas [Florencio e Herley 2007]. Por outro lado, o protocolo *OpenID* aliado ao uso de autenticação por senha, é vulnerável a ataques de *Phishing*, no qual o usuário é levado a um SP malicioso a digitar sua senha em um formulário falso que imita o IdP [Cameron 2007, Lee et al. 2008].

O trabalho que apresentamos neste texto é parte do projeto *SecFuNet*[1] que se propõe a desenvolver um framework de segurança para aplicações de *Cloud Computing*. Este

[1] Projeto aprovado em edital Europa-Brasil (Edital MCT/CNP de número 66/2010) como processo CNPq de número de 590047/2011-6.

framework introduz, entre seus vários serviços, funções de autenticação e autorização para ambientes de computação em nuvem. O objetivo deste artigo é apresentar uma abordagem de autenticação baseada em *SmartCards* e políticas de distribuição de atributos centrada no usuário. Neste sentido, apresentamos um modelo desenvolvido e seus respectivos protocolos para a autenticação de usuários na infraestrutura *SecFuNet* e a aplicação subsequente de controles de acesso baseado em atributos em provedores de serviço de *clouds*.

No gerenciamento de identidades proposto para o *SecFuNet*, as relações de confiança entre usuários, IdPs e SPs fazem uso dos protocolos do framework *OpenID* [OpenID 2007]. Para eliminar a vulnerabilidade a ataques de *phishing*, o gerenciamento de identidades é fortemente dependente de componentes de *hardware* protegido. A autenticação de usuários é feita diretamente entre *SmartCards* (de posse do usuário) e processadores seguros dispostos em um servidor de identidades (servidor de autenticação).

O restante do artigo está organizado da seguinte forma: a Seção 2 descreve os requisitos de gerenciamento de identidade no contexto do projeto *SecFuNet*; a Seção 3 apresenta os protocolos da infraestrutura proposta; a Seção 4 aborda a implementação do protótipo; a Seção 5 contém uma discussão sobre a proposta e comparação com trabalhos relacionados; a Seção 6 conclui este artigo.

2. Gerenciamento de Identidades no *SecFuNet*

Em sistemas distribuídos, o processo de autenticação de usuários envolve, em um primeiro passo, um procedimento de *login* onde um usuário, através de trocas com um provedor de identidade (usando técnicas como nome de usuário/senha, userid/certificado, protocolos de desafio/resposta, etc.), tem a sua identidade validada diante do sistema. Este procedimento, normalmente termina com o usuário recebendo do seu IdP um *token* ou asserção de autenticação que serão usados para atestar sua autenticidade diante de outras entidades do sistema em subsequentes interações. Os IdP passam a ser os guardiões de informações sigilosas como senhas, certificados, atributos de usuário, etc., deixando usuários e SPs livres da manutenção e sigilo de listas muitas vezes enormes de credenciais.

Em sistemas de computação em nuvens, serviços de gerenciamento de identidades são usualmente implementados em *clouds* privadas ou em servidores com *hardware* especial. A não colocação de um IdP em *clouds* públicas se justifica pela criticidade das informações.

O projeto *SecFuNet*, por enfatizar o uso de processadores seguros no processo de autenticação de usuários, implica em limitações de escala. Então o gerenciamento de identidades deve envolver vários IdPs, cada um com o seu próprio domínio de política. Estes provedores de identidades fornecem asserções de autenticação, seguindo suas politicas, para usuários e provedores de serviço (SPs) de seus domínios de política. Nestes domínios, portanto, o gerenciamento de identidades segue uma abordagem centralizada onde o usuário fornece informações que são verificadas em um processador seguro no IdP do seu domínio. O IdP desempenha o papel de uma terceira parte confiável nas interações entre usuário e SPs. Esta abordagem centralizada é largamente usada em sistemas corporativos.

No entanto, para sistemas complexos como *clouds*, que normalmente se estendem além destes domínios locais, este modelo de intermediação simples é limitado. É necessário

expandir esse modelo para um sistema de gerenciamento de identidade com base em redes de confiança, envolvendo vários destes provedores locais de identidade. Ou seja, é necessário adotar abordagens de gerenciamento onde ou os IdPs de diversos domínios possuem relações de confiança entre si (abordagem de "identidades federadas"), ou os SPs possuem listas de IdPs em quem confiam (abordagem centrada em SPs). Para a aplicação destas abordagens citadas, é necessário estender a escala das relações de confiança e isto é feito com o uso de Infraestruturas de Autenticação e Autorização (IAAs). Exemplos notórios destas infraestruturas são: o *Shibboleth* [Scavo e Cantor 2005] e o *Liberty Alliance* [Liberty 2003] para abordagens de identidades federadas e *OpenID* como abordagem centrada em SPs.

No projeto *SecFuNet*, seguimos a abordagem do *OpenID*, onde cada provedor de serviços (SP ou RP) mantém uma lista de IdPs em que confia e que seus usuários devem ter identidades em pelo menos um destes IdPs. Os usuários de posse de um cartão inteligente (*SmartCard*) deve fornecer ao IdP de seu domínio as informações de autenticação. No caso do projeto citado, a autenticação de usuários é baseada em certificados X509 que estão disponíveis em cartão. Na abordagem apresentada neste texto, enfatizamos a concentração de atributos de usuário também no *SmartCard* do mesmo, evitando as clássicas bases de atributos construídas com LDAP e ligadas ao IdP do usuário.

3. Infraestrutura de Autenticação e Autorização para o *SecFuNet*

Esta seção apresenta os protocolos propostos para a infraestrutura de autenticação e autorização. Antes disso, porém, a subseção 3.1 descreve o protocolo *OpenID* original. A subseção 3.2 apresenta as extensões propostas dentro da infraestrutura.

3.1 *OpenID*

OpenID é um conjunto de especificações que definem protocolos de gerenciamento de identidades na Web. Os protocolos são baseados nas tecnologias mais básicas da Web (URLs e HTTP/HTTPS) e permitem que uma entidade, denominada de *OpenID Provider* (OP) atue como IdP e emita asserções de autenticação e atributos de usuários para provedores de serviços, denominados de *Relying Parties* (RPs). Além disso, o *OpenID* suporta autenticação única (*Single Sign On* – SSO) de maneira a simplificar a experiência do usuário no acesso a múltiplos RPs em uma mesma sessão de autenticação.

O processo de autenticação no *OpenID* ocorre da seguinte forma (ilustrado na Figura 1): o usuário inicia o processo apresentando um identificador para o RP (passo 1); com base no identificador apresentado, o RP realiza a descoberta da URL do OP que autenticará o usuário (passo 2); opcionalmente, uma associação é criada entre o RP e o OP (passo 3), na qual uma chave secreta é estabelecida, usando um protocolo de Diffie-Hellman, com o objetivo de verificar mensagens trocadas futuramente; o RP redireciona o navegador do usuário para o OP, passando junto um pedido de autenticação (passo 4); por algum mecanismo não especificado (i.e. não limitado) pelo *OpenID*, o usuário prova para o OP que é um usuário registrado e dono do identificador usado (passo 5); o OP redireciona o navegador do usuário de volta ao RP, passando uma asserção indicando, ou que o usuário foi identificado (asserção positiva) ou que a autenticação falhou (asserção negativa) (passo 6); o RP verifica as

informações enviadas pelo OP, por meio da assinatura usando a chave secreta estabelecida no passo 3 (se não houver associação, o RP pode pedir a verificação diretamente no OP, porém isso não está ilustrado na Figura 1).

Após o usuário passar pelo processo de autenticação e acessar um RP, é possível acessar outro serviço eliminando alguns estágios do processo. Quando o usuário se autentica no OP, este pode criar uma sessão de autenticação com o navegador do usuário, de forma que em acessos posteriores na mesma sessão o usuário não necessita se autenticar novamente. Nesse caso, quando o usuário acessar outro RP, o mesmo processo ilustrado na Figura 1 é executado, porém o passo 5 ocorre de maneira automática e transparente para o usuário. Dessa forma o *OpenID* permite a autenticação única SSO.

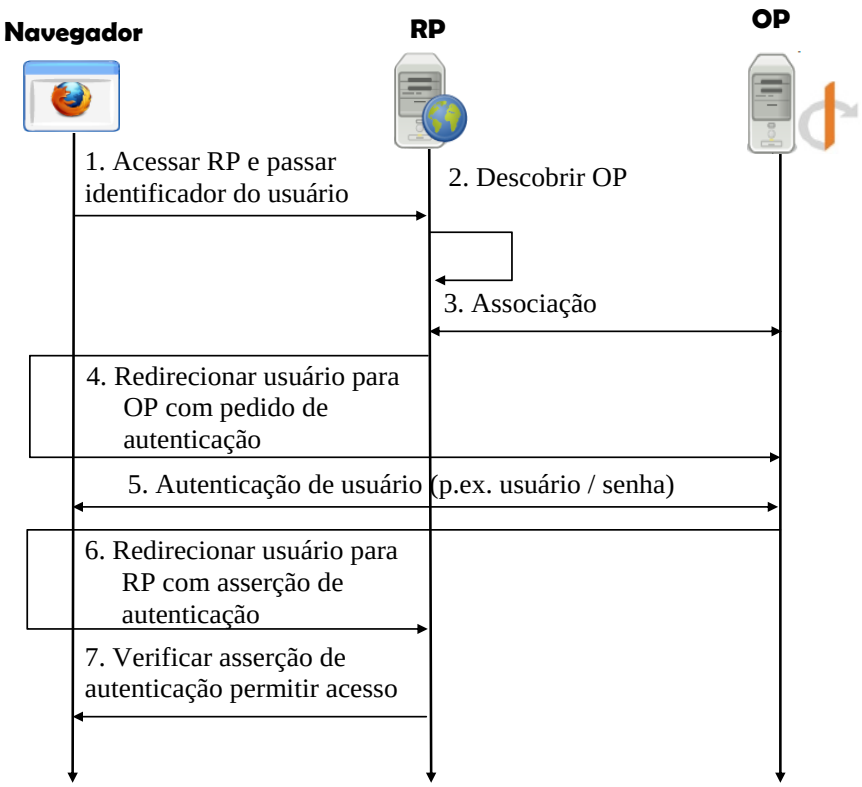

Figura 1: Interações na autenticação do *OpenID*

Durante o processo de autenticação, o RP pode listar um conjunto de atributos do usuário no pedido de autenticação (passo 4 da Figura 1). Esses atributos podem ser liberados e enviados para o RP junto com a asserção de autenticação (passo 6 da Figura 1). Dessa forma, o RP pode realizar controles de acesso adicionais, com base nos atributos do usuário, antes de liberar o acesso aos recursos. Assim como no caso do mecanismo de autenticação empregado pelo OP, a especificação do *OpenID* não dita nenhum mecanismo de armazenamento dos atributos do usuário.

Nos protocolos propostos na seção 3.2, as especificações do *OpenID* são utilizadas como protocolo de descoberta e troca de informações. Além disso, as omissões do *OpenID* são preenchidas para garantir autenticação baseada em *SmartCards* e armazenamento de atributos sob o controle do usuário.

3.2 Autenticação com *SmartCards* e uso de Infraestrutura de AAI

O protocolo de autenticação proposto integra a autenticação com *SmartCards* e o controle de atributos centrado no usuário dentro infraestrutura de autenticação do *OpenID*. Essa integração é facilitada pelo fato de o *OpenID* ser agnóstico quanto ao mecanismo de autenticação e quanto ao armazenamento dos atributos do usuário.

Na maioria dos OPs atuais se utiliza o mecanismo de usuário e senha para autenticação, no entanto o *OpenID* não depende deste mecanismo e a comunicação direta entre o usuário e o OP não é especificada. O protocolo proposto utiliza autenticação mútua baseada em certificados digitais. No lado do usuário, esses certificados devem estar armazenados em *SmartCards* que garantem a privacidade da chave privada. Toda a criptografia é executada pelo próprio *SmartCard*, sem que a chave privada seja lida por nenhum processo externo. Além disso, o *SmartCard* exige que um PIN secreto seja inserido antes de permitir o acesso às credenciais.

Outro aspecto importante do protocolo proposto é a centralização do controle dos atributos de identidade no usuário. Os valores dos atributos são armazenados no *SmartCard* juntamente com as credenciais e o acesso a estes também requer a entrada do PIN pelo usuário. O OP não armazena os valores dos atributos, devendo notificar o usuário sobre quais atributos o RP exige, para que o mesmo possa decidir se permite ou não a leitura dos valores de dentro do *SmartCard* e os repasse ao RP.

Como o OP envia ao RP os atributos liberados pelo usuário em uma asserção assinada, é necessário haver confiança de que o usuário está de fato liberando valores corretos para os atributos. Por outro lado, o OP pode exigir repassar atributos do usuário somente se estiverem assinados por uma entidade confiável. Esses atributos podem ser armazenados no *SmartCard* somente por uma entidade de registro, por exemplo mediante apresentação de documentos ou presença física do usuário. Atributos assinados não podem ser modificados pelo usuário, apesar de sua liberação ainda ser controlada pelo mesmo. Atributos com diferentes níveis de exigência podem ser armazenados independentemente no mesmo cartão e fica a cargo das políticas do domínio do OP o estabelecimento das regras específicas.

A comunicação entre o *SmartCard* e o OP é mediada pelo Seletor de Identidades (SI), componente proposto com características similares a outros projetos como *CardSpace* [Chappell 2006] e *Higgins* [EcpliseFoundation 2010]. O SI se integra à interface do OP e implementa a autenticação e a transferência de atributos do usuário. Por realizar acesso ao *SmartCard*, o SI precisa ser um aplicativo confiável, assinado pelo OP ou por outra entidade na qual o usuário confie. A tecnologia *Java Applet* foi escolhida para construir o SI por se integrar diretamente ao navegador Web, simplificando a experiência do usuário ao eliminar a necessidade de utilizar um aplicativo separado, e por suportar diretamente a assinatura digital do código do SI.

O modelo de confiança utilizado assume que o RP possui uma lista de OPs confiáveis. A confiança no OP indica que o RP aceita asserções de autenticação e atributos sobre usuários autenticados. Diferente do modelo federado, no qual provedores de serviço e identidade formam um domínio comum com limites bem definidos, a confiança nos OPs é gerenciada no RP de acordo com políticas locais. A confiança é unidirecional e o OP não se compromete

em selecionar RPs confiáveis, ficando a cargo do usuário decidir se deseja liberar suas informações de identidade.

A confiança no OP não garante, por si só, a permissão do acesso de usuários autenticados aos recursos disponibilizados pelo RP. Localmente, o RP pode possuir políticas de controle de acesso complexas, utilizando atributos disponibilizados pelo usuário, atributos do OP, etc.. Inclusive, é possível que níveis de acesso diferentes sejam aplicados a usuários dependendo da quantidade de informações disponibilizadas: se mais informações de identidade são recebidas pelo RP, maior a rastreabilidade do usuário e maior a confiança depositada pelo RP.

A Figura 2 ilustra os passos do protocolo de autenticação proposto. Os passos de comunicação envolvendo o RP (passos 1, 2, 11 e 12) ocorrem de maneira idêntica ao *OpenID* padrão, ou seja, nenhum alteração dos RPs é necessária para se adequar à infraestrutura proposta. No passo 1, o usuário tenta acessar um recurso protegido no RP, apresentando seu *OpenID*. No passo 2, o RP descobre o OP do usuário e redireciona o navegador. Nesse redirecionamento, o RP indica quais atributos de identidade são necessários para que possa realizar o controle de acesso. Nos passos 3 a 10 ocorre a autenticação do usuário e o envio de atributos, detalhados mais a seguir. No passo 11, o OP redireciona o navegador de volta ao RP, passando os atributos obtidos do usuário e a asserção de autenticação. No passo 12, o RP compara os valores de atributos com as políticas de controle de acesso e libera o recurso.

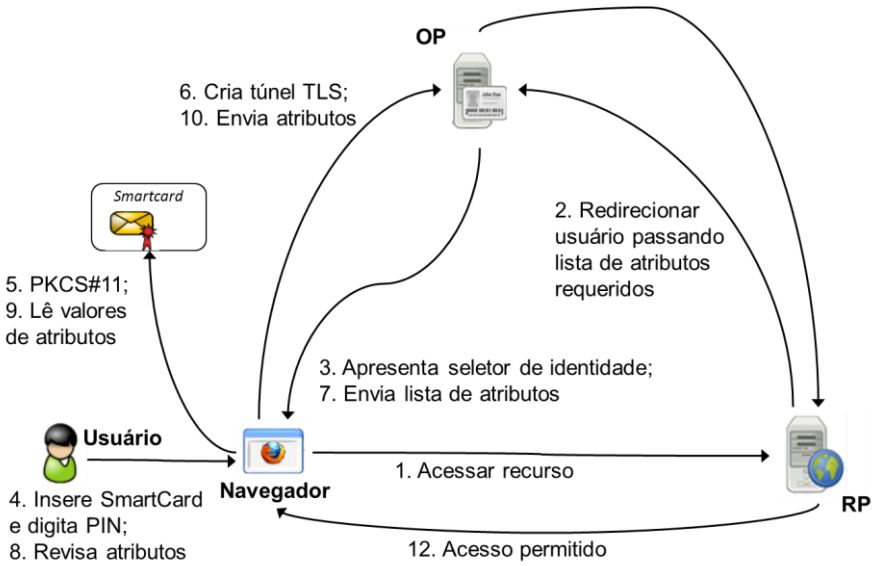

Figura 2. Comunicação no protocolo proposto

Quando o usuário é redirecionado pelo RP para o seu OP, este último apresenta ao usuário, por meio do navegador, o aplicativo Seletor de Identidades (SI) (passo 3). O SI pede para o usuário inserir o *SmartCard* e digitar o PIN, a fim de poder utilizar as credenciais e autenticar com o OP (passo 4). O SI estabelece um canal TLS mutualmente autenticado com o OP (passo 6), utilizando a biblioteca PKCS#11 para empregar a chave privada e certificado sem precisar de ter acesso a nenhum material secreto (incluindo segredos compartilhados) (passo 5). Após a autenticação, o SI recebe a lista de atributos e a identificação do RP (passo 7),

para que o usuário possa revisar (passo 8). O usuário tem total controle sobre quais atributos deseja enviar ao RP. Se o usuário aceitar liberar os atributos, os valores são lidos do *SmartCard* (também usando a API PKCS#11) (passo 9) e enviados para o OP usando o canal TLS (passo 10). Ao receber os atributos, o OP dá seguimento ao protocolo *OpenID* e encaminha o usuário de volta para o RP (passo 11). O RP utiliza os atributos do usuário para realizar o controle de acesso, liberando ou recusando o recurso (passo 12).

Se o usuário acessar um RP diferente, após ter-se autenticado, o mesmo procedimento é executado, porém o único passo visível ao usuário é a revisão dos atributos requisitados e a decisão sobre a sua liberação. Mesmo dentro de uma mesma sessão de autenticação, a liberação de atributos é individual para cada RP distinto acessado. Apesar disso, o SI permite que a decisão seja lembrada para que acessos futuros ao mesmo RP transcorram automaticamente.

4. Implementação do Protótipo

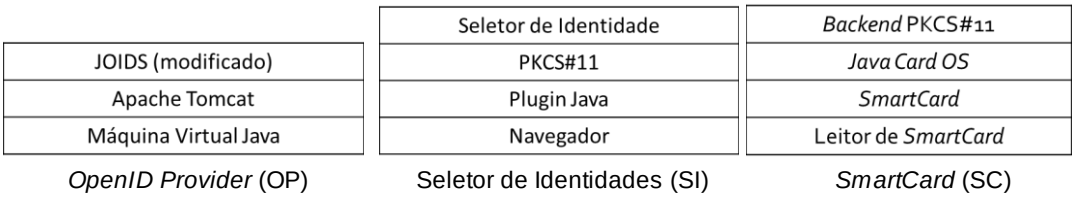

Figura 3: Componentes do protótipo

A fim de avaliar a viabilidade da infraestrutura proposta, um protótipo foi desenvolvido de acordo com os protocolos elaborados. O protótipo está dividido em três componentes principais (Figura 3): *OpenID Provider* (OP), Seletor de Identidades (SI) e *SmartCard* (SC). O OP consiste em um provedor *OpenID* modificado para realizar a autenticação de usuários por certificados digitais e para se comunicar com o SI para obter os atributos do usuário. O SI é um *applet Java* assinado que executa no navegador do usuário e apresenta duas funções básicas: (i) acessar o *SmartCard* para autenticar o usuário por meio do certificado e obter os atributos armazenados, utilizando para tal a biblioteca PKCS#11; e (ii) apresentar uma interface gráfica perguntando ao usuário se permite a liberação dos atributos pedidos para o provedor de serviço. O componente SC é, por sua vez, composto pelo leitor de cartões, pelo *SmartCard* em si e pelo software que executa no cartão. O cartão executa um *applet* de *backend* para a API PKCS#11, sendo responsável por armazenar de maneira segura o par de chaves e certificado do usuário, que serão usados na autenticação, bem como os atributos do usuário. O *applet* exige que o usuário forneça um PIN para liberar o acesso às informações.

A implementação de OP utilizada foi o JOIDS, escrito em *Java*, principalmente por se tratar de uma implementação simples e de fácil modificação. O JOIDS executa sobre um *Servlet Container*, como o *Apache Tomcat*, e está baseado na biblioteca *OpenID4Java*, que implementa os protocolos *OpenID* na linguagem *Java*. O OP foi "esvaziado", no sentido em que não armazena atributos de usuários, pois estes obtidos do *SmartCard*. Neste trabalho, o OP implementa o estabelecimento do canal TLS em software, utilizando diretamente as

funcionalidades providas pelo container. Um trabalho futuro consiste em integrar o OP com um componente seguro que ofereça proteção à chave privada do servidor, assim como o *SmartCard* do usuário.

O Seletor de Identidades foi implementado como um *applet Java*. Applets se integram facilmente com os navegadores e o código objeto pode ser assinado digitalmente para impedir a execução de códigos maliciosos no lugar do Seletor de Identidade. No procedimento de autenticação, o OP envia o *applet* do SI para o usuário, sendo que este recebe a lista de atributos e o provedor de serviços acessado. O SI exibe as informações para o usuário e aguarda a decisão sobre a liberação. Se o usuário liberar o acesso, o SI pede então para o usuário inserir o cartão e entrar com o PIN.

O SI utiliza a API *JavaCard* 2.1 e a biblioteca PKCS#11 para acessar as informações do *SmartCard* e se conecta com o OP para realizar a autenticação e a transferência de atributos. PKCS#11 é uma API geral que permite acesso a funções criptográficas providas por um hardware. Essa API fornece funções para criação de chaves ou pares de chaves, derivação de chaves, utilização de algoritmos criptográficos, entre outras. Os objetos armazenados no *SmartCard* são tratados como sensíveis, no caso de atributos, podendo ser apenas lidos de maneira segura, e não extraíveis, no caso de chaves privadas e segredos compartilhados, podendo ser apenas utilizados dentro do próprio *SmartCard* em funções criptográficas. A biblioteca SunPKCS#11, distribuída junto com o ambiente Java, utiliza as funções fornecidas pela PKCS#11 para implementar um cliente TLS baseado no *SmartCard*.

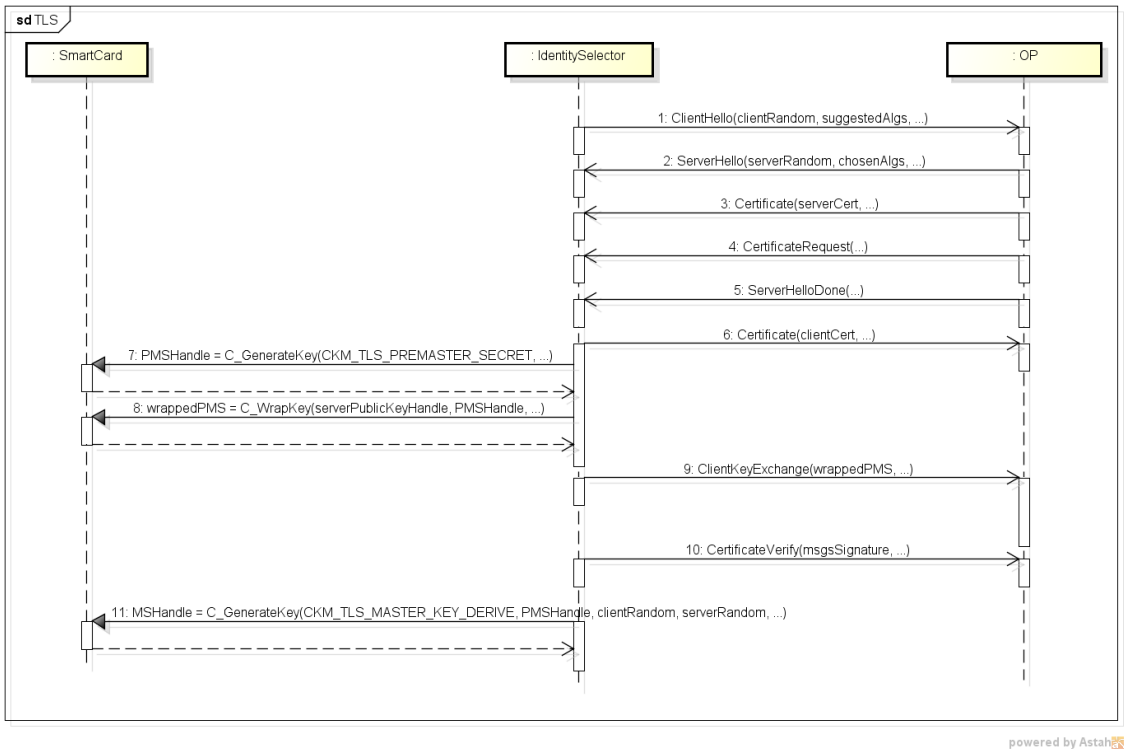

Figura 4. Estabelecimento de um canal TLS usando PKCS#11

A Figura 4 ilustra os passos principais do estabelecimento de um canal TLS usando a API PKCS#11 para acessar um *SmartCard*. No *handshake* inicial (passos 1 a 6) o SI troca o

certificado do usuário com o OP, bem como números aleatórios utilizados como material para criação de segredos. No passo 7, o SI invoca a função C_GenerateKey para criar um segredo do tipo *PreMasterSecret* (PMS), que é a base para o estabelecimento do segredo compartilhado. Essa chamada cria um objeto sensível dentro do cartão e retorna apenas um ponteiro. No passo 8, o PMS é cifrado pela chave pública do OP, usando a chamada C_WrapKey. O resultado é enviado para o OP, para que este possa derivar o mesmo segredo que o cliente (passo 9). No passo 10, o cliente deriva, usando a função o *MasterSecret* (MS), que é o segredo compartilhado utilizado para gerar as chaves simétricas que efetivam o túnel. O MS é criado como um objeto não extraível, isto é, seu valor nunca pode ser lido do cartão (do mesmo modo que a chave privada). Um ponteiro para o MS é retornado e este pode ser usado por funções criptográficas dentro do *SmartCard*.

A comunicação dos atributos entre o SI e o OP é realizada de maneira segura dentro do túnel TLS. O envio é feito por meio de mensagens HTTP ao OP, que extrai os valores e monta o redirecionamento até o RP. O SI recebe o redirecionamento e aplica ao navegador, dando seguimento ao *OpenID*. A interface do SI é implementada usando HTML e manipulada pelo *Applet* por meio da invocação de funções *JavaScript* no navegador. A Figura 5 mostra uma captura de tela do SI.

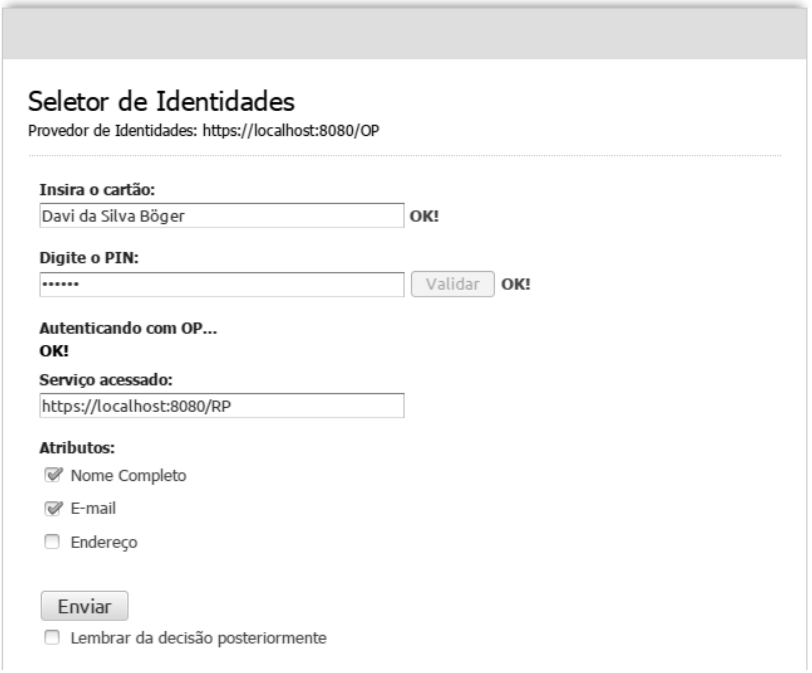

Figura 5. Interface gráfica do seletor de identidades

As versões de software utilizadas foram as seguintes: plugin *Java* e *Java Runtime Environment* 7u21[2]; *Apache Tomcat* 7.0.39[3]; JOIDS svn r577[4] (a última versão estável apresenta problemas para lidar com atributos de usuário, resolvidos na versão em

[2] http://java.oracle.com

[3] http://tomcat.apache.org

[4] http://code.google.com/p/openid-server

desenvolvimento); biblioteca PKCS#11 e *applet* de *backend SafeSign* 3.0.45; *JavaCard* 2.1[5].

5. Discussão

A substituição do mecanismo de autenticação para o uso de *SmartCards* aumenta o nível de segurança na medida em que, para se autenticar, não basta conhecer o segredo do usuário, mas também ter posse do dispositivo físico. Cada *SmartCard* contém um par de chaves e um certificado associado, que somente podem ser utilizados mediante apresentação do PIN secreto. Toda a criptografia é executada pelo processador do próprio *SmartCard*, de forma que a chave privada nunca fica exposta. Apesar do PIN se assemelhar a uma senha, o valor secreto é verificado localmente na máquina do usuário e somente tem algum valor se aliado à posse do cartão físico.

Com o uso de *SmartCards*, os atributos de identidade passam a ser armazenados em segurança no dispositivo de posse do usuário. Esses atributos são protegidos de acessos não autorizados pelo PIN. Dessa forma o provedor de identidade fica aliviado do gerenciamento de atributos e também ocorre aumento do controle do usuário sobre sua identidade.

Os protocolos elaborados não alteram nenhum aspecto do *OpenID* original relacionado com a comunicação com os RPs. Isso implica na compatibilidade do protocolo com provedores de serviço já habilitados para aceitar autenticação via *OpenID*. Além disso, as relações de confiança do RP continuam as mesmas, isto é, confiança nas asserções de autenticação e atributos emitidos pelo OP.

O uso de atributos assinados permite ao OP certificar-se da correção das informações liberadas pelo usuário, fato que pode ser explorado para aumentar a confiança de RPs caso necessitem desses atributos específicos. Uma extensão simples dos protocolos pode incluir a publicação pelo OP de uma lista de atributos que são disponibilizados e quais são assinados, permitindo ao RP decidir o acesso de usuários com base nessa lista, juntamente com suas políticas locais. Nesse caso, um usuário autenticado em um OP que usa um atributo assinado pode receber acesso, enquanto um usuário que apresenta um atributo sem garantia não.

5.1 Trabalhos relacionados

Em [Ahn et al. 2009] é apresentada uma abordagem de controle de atributos centrada no usuário que visa facilitar ao usuário a escolha de quais atributos devem ser liberados. A proposta assume um modelo de credenciais, como em *CardSpace*, e categoriza as credenciais de acordo com o domínio de aplicação (p.ex. credencias de banco e de operadoras de crédito são associadas à categoria "finanças") e atribuindo níveis de sigilo a essas categorias. Dessa forma o seletor de identidade apresenta informações mais completas a fim de permitir ao usuário julgar melhor as políticas de atributos e proteger sua privacidade. Neste trabalho, o objetivo principal é aumentar a privacidade pelo uso de componentes de hardware protegidos (*SmartCards*), no entanto a abordagem de categorização pode ser incorporada ao SI proposto neste trabalho sem alteração dos protocolos.

[5] http://download.oracle.com/otndocs/jcp/7234-javacard-2.1-spec-oth-JSpec/

Em [Urien 2010] é demonstrada uma implementação de autenticação para o protocolo OpenID baseada em *SmartCards*. A proposta é similar a deste trabalho, exceto que os atributos do usuário são tratados da maneira usual, isto é, armazenados no OP. Aqui, propomos que o controle do usuário sobre seus atributos seja aumentado, incluindo armazenamento local e liberação seletiva.

Em [Vossaert et al. 2011] Os autores apresentam uma abordagem que utiliza um elemento seguro (*SmartCard*, SIM *Card*, celular, etc.) que faz o intermédio no processo de autenticação de um usuário e um provedor de serviços. Nesta abordagem os provedores de serviço não se associam com os provedores de identidades quando da necessidade de solicitação de atributos. Este processo é realizado diretamente entre o elemento seguro de posse do usuário e o provedor de serviço. O elemento seguro é o provedor de atributos locais e também a entidade responsável pela recuperação dos atributos disponíveis nos provedores de identidades. Na abordagem proposta pelos autores, não existe qualquer comunicação entre os provedores de serviços e provedores de identidades, e toda troca de atributos entre as partes é realizada com o intermédio do elemento seguro, o que mantém sob o controle do usuário a liberação ou não de seus atributos. Essa abordagem exige que os provedores de serviço se comuniquem diretamente com o dispositivo do usuário, o que pode tornar a implementação de SPs custosa. A nossa proposta utiliza o *OpenID* a fim de simplificar o desenvolvimento e a integração de SPs.

Em [Leicher et al. 2012] é apresentado uma abordagem que utiliza dispositivos móveis (celular) como forma de aumentar a segurança no processo de autenticação de usuários utilizando o protocolo *OpenID*. O protocolo proposto pelos autores descreve como principal componente da abordagem uma entidade denominada "*local OP*". Este "*local OP*" é um processo executado diretamente no dispositivo móvel do usuário e mantém as informações sigilosas deste, assim como os certificados necessários para operações criptográficas envolvidas no protocolo *OpenID*. A principal função deste "*local OP*" é a criação da asserção que será enviada para o RP ao qual o usuário deseja se autenticar. Um detalhe a ser observado é que as credenciais do usuário no modelo proposto são gerenciadas pela operadora de telefonia, dessa forma caso o usuário queria ter várias credenciais este deve adquiri-las diretamente com sua operadora para estarem dispostas no "*local OP*".

6. Conclusão

O protocolo proposto e implementado neste trabalho apresenta uma solução para o gerenciamento de identidades no contexto do projeto *SecFuNet*, especificamente para o acesso a serviços em *Cloud* e redes virtuais. A infraestrutura desenvolvida está baseada nos protocolos da especificação *OpenID*, integrada com autenticação por meio de *SmartCards* e controle de atributos centrado no usuário. Essa integração provê uma solução com alto nível de segurança, evitando os problemas da autenticação por nome de usuário e senha e garantindo o sigilo dos atributos dos usuários.

Como trabalhos futuros, tem-se como meta estender os protocolos para possibilitar o gerenciamento federado de identidades, isto é, o estabelecimento de relações de confiança entre OPs de domínios distintos e a transposição de asserções emitidas em um domínio para acessar serviços em outro. Outra perspectiva é a integração do protótipo com outras partes

do projeto *SecFuNet* que envolvem uso de componentes de hardware seguro no OP e implementação de mecanismos de tolerância a faltas e a intrusões.

7. Agradecimentos

Este trabalho foi financiado pelo CNPq através dos processos 590047/2011-6 (Projeto SecFuNet) e também pelos processos 307588/2010-6 e 384858/2012-0. Agradecemos ainda a CAPES pelo apoio financeiro com bolsa de doutorado.

Referências

Ahn, G.-J., Ko, M. and Shehab, M. (2009) Privacy-enhanced User-Centric Identity Management, In *Communications, 2009. ICC '09. IEEE International Conference on*. IEEE.

Bhargav-Spantzel, A., Camenisch, J., Gross, T., and Sommer, D. (2007). User centricity: a taxonomy and open issues. *Journal of Computer Security*, 15(5):493-527.

Cameron, K. (2007). *Integrating OpenID and Infocard - Part 1*. http://www.identityblog .com/?p=659.

Chappell, D. (2006). *Introducing windows cardspace*. Msnd technical articles, Microsoft Corporation. http://msdn.microsoft.com/en-us/library/aa480189.aspx.

Florencio, D. and Herley, C. (2007). A large-scale study of web password habits. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pp 657–666, New York, NY, USA. ACM.

EclipseFoundation (2010). *Higgins open source identity framework*. http://www.eclipse.org/higgins/.

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

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

Lee, H., Jeun, I., Chun, K. and Song, J. (2008). A New Anti-Phishing Method in OpenID. In: *SECURWARE '08. Second International Conference on*, pp.243,247, IEEE.

Leicher, A., Schmidt, A. U. and Shah, Y. (2012) Smart OpenID: A Smart Card Based OpenID Protocol, In *IFIP Advances in Information and Communication Technology* v. 376, pp 75-86, Springer.

Liberty (2003). Introduction to the Liberty Alliance Identity Architecture. Liberty Alliance.

OpenID (2007). Openid authentication 2.0. OPENID. http://openid.net/specs/openid-authentication-2_0.html.

RSA Laboratories (2009). PKCS #11: Cryptographic Token Interface Standard. http://www.rsa.com/rsalabs/node.asp?id=2133

Scavo, T. e Cantor, S. (2005). Shibboleth Architecture. http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf.

Urien, P. (2010). An OpenID Provider based on SSL Smart Cards. In *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*. http://dx.doi.org/10.1109/CCNC.2010.5421756.

Vossaert, J., Lapon, J., Decker, B. and Naessens, V. (2011) User-centric identity management using trusted modules. In *Lecture Notes in Computer Science*, v. 6711, pp 155-170, Springer.

