RICHARD FLÁVIO DA SILVA

ATESTAMENTO EM ARQUITETURA ABERTA DE SERVIÇOS (SOA): UM ARCABOUÇO PARA VALIDAÇÃO DE LEGITIMIDADE DOS CONSUMIDORES DE SERVIÇOS E SEUS DISPOSITIVOS DE ACESSO

São Paulo
2010

RICHARD FLÁVIO DA SILVA

ATESTAMENTO EM ARQUITETURA ABERTA DE SERVIÇOS (SOA): UM ARCABOUÇO PARA VALIDAÇÃO DE LEGITIMIDADE DOS CONSUMIDORES DE SERVIÇOS E SEUS DISPOSITIVOS DE ACESSO

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia

São Paulo
2010

RICHARD FLÁVIO DA SILVA

ATESTAMENTO EM ARQUITETURA ABERTA DE SERVIÇOS (SOA): UM ARCABOUÇO PARA VALIDAÇÃO DE LEGITIMIDADE DOS CONSUMIDORES DE SERVIÇOS E SEUS DISPOSITIVOS DE ACESSO

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia

Área de Concentração:
Sistemas Digitais

Orientador:
Prof. Dr. Wilson Vicente Ruggiero

São Paulo
2010

AGRADECIMENTOS

Ao meu orientador, Wilson Ruggiero, por seu papel como um verdadeiro mentor apaixonado pela pesquisa e evolução do conhecimento. Um profissional que depositou uma confiança muito motivadora a este trabalho.

Também sou especialmente grato à minha esposa com toda sua compreensão e apoio que necessitei para conciliar os compromissos profissionais e acadêmicos com a dedicação à família.

Agradeço também às minhas filhas que, embora ainda tão pequenas, se comportaram muito carinhosamente enquanto assistiam aos meus trabalhos de pesquisa.

RESUMO

Em Arquiteturas Abertas e Orientadas a Serviços (*Service Oriented Architectures* - SOA) a preocupação com a área de segurança tem recebido importante atenção no desenho das aplicações em função das vulnerabilidades intrínsecas associadas a este novo paradigma. Este trabalho tem por objetivo propor um arcabouço para o desenvolvimento seguro de aplicações em SOA, com foco nos aspectos de segurança através da validação dos consumidores de serviços e seus dispositivos de acesso. Para este objetivo, foi conduzido um estudo sobre as abordagens tradicionais para a segurança em soluções *Web* uma vez que esta é a plataforma predominante na implementação de soluções SOA. Neste estudo, foi identificado que uma área fértil para contribuição à segurança em soluções SOA é o atestamento dos componentes ou programas consumidores de serviços. Por atestamento, deve-se entender um processo de verificação da legitimidade dos participantes (indivíduos, *hardware* e *software*) de uma cadeia de acesso. Como resultado deste trabalho, foi proposto um arcabouço, denominado ASACS (Arcabouço para Segurança por Atestamento dos Consumidores de Serviços), para controle de acesso aos serviços baseado na validação de etapas de atestamento dos consumidores. Estas etapas vão desde o fornecimento de informações sobre a plataforma de execução dos consumidores até a análise comportamental para definir o grau de confiança de cada consumidor em uma rede distribuída de serviços. A utilização do arcabouço traz um importante reforço à segurança ao buscar a negação de acesso a consumidores mal intencionados, não legítimos ou que tenham sido alvo de ataque. Como principal contribuição, este arcabouço orienta uma abordagem estruturada para a validação de legitimidade dos consumidores e de seus dispositivos e programas de acesso, resultando na necessidade de um aumento dos esforços requeridos para um ataque na tentativa de violar a segurança dos serviços oferecidos.

Palavras-chave: Segurança. SOA. Atestamento. Autenticação. *Web*.

ABSTRACT

In Service Oriented Architectures (SOA) the concern with security has received important focus on solution design as a consequence of intrinsic vulnerabilities at the basis of this new paradigm. This work proposes a framework to secure development of SOA applications, with a special attention to security matters regarding validation of service consumers and its access means. In order to accomplish this goal, it was conducted a research over traditional approaches for security in *Web* applications, considering that *Web* platform is definitely dominant for SOA implementation. In this research, validation of service consumers was figured out as a promising area to security enforcement. Validation of service consumers states for a process to verify legitimacy of participants (individuals, hardware and software) in an access chain. As result of this work, it was proposed the framework ASACS designed to control accesses to service providers based on consumer's legitimacy validation stages. Such validation stages cover since a check of client station's boot stack layers until a behavioral monitoring to graduate trust levels for each consumer in a network of distributed services. The framework adoption promotes security enforcement while avoiding access from malicious consumers, non legitimate ones or genuine consumers compromised by an attacker. As principal contribution, this framework guides to a structured approach to validate legitimacy of consumers and its programs and access means, requiring higher levels of efforts to an attack attempting violate the security of published services.

Keywords: Security. SOA. Legitimacy. Authentication. Web.

LISTA DE FIGURAS

Figura 2.1 - Casos de uso UML para SOA 17
Figura 2.2 - Operações e componentes básicos de SOA 19
Figura 2.3 - Princípios de segurança 22
Figura 2.4 - Passos da assinatura digital 27
Figura 2.5 - Comunicações entre receptor, emissor e autoridade certificadora 28
Figura 2.6 - Modelo conceitual de uma PKI 30
Figura 2.7 - Configuração do modo de segurança nó-a-nó 32
Figura 2.8 - Configuração do modo de segurança fim-a-fim 32
Figura 2.9 - Cabeçalho de um envelope SOAP com as *tags* do WS-Security 34
Figura 2.10 - Tipos de atestamento 36
Figura 3.1 - Representação simplificada de uma arquitetura *Web* 38
Figura 3.2 - Uma arquitetura *Web* com elementos de segurança 39
Figura 3.3 - Arquitetura de uma aplicação *Web* orientada a serviços 43
Figura 3.4 - *Stack trace* com vazamento de informações 47
Figura 4.1 - Etapas do arcabouço de atestamento ASACS 53
Figura 4.2 - Diagrama de componentes do ASACS 56
Figura 4.3 - Papéis dos programas tratadores no ASACS 60
Figura 4.4 - Registro dos consumidores e verificação da integridade das camadas de inicialização .. 65
Figura 4.5 - Cadastramento do consumidor 65
Figura 4.6 - Personalização e envio do *token* 67
Figura 4.7 - Ativação do *token* 67
Figura 4.8 - Registro das seqüências de inicialização das estações dos consumidores 67
Figura 4.9 - Diagrama de seqüência para a primeira invocação de um serviço do provedor 69
Figura 4.10 - Geração de chaves simétricas de sessão sincronizadas 73
Figura 4.11 - Diagrama de seqüência representando a utilização da chave simétrica de sessão 74
Figura 4.12 - Certificação e assinatura digital bilaterais 76
Figura 4.13 - Diagrama de seqüência da certificação e assinatura digital do consumidor 77
Figura 4.14 - Regulação do grau de confiança dos consumidores 81
Figura 4.15 - Diagrama de seqüência representando a regulação do grau de confiança 82
Figura 4.16 - Modelo de alto desempenho e disponibilidade do agente regulador 90
Figura 4.17 - Técnica para evitar o uso do *token* por usuários não legítimos 93
Figura 4.18 - Instalação de programas tratadores nas estações dos consumidores 97
Figura 4.19 - Agentes participantes da aplicação exemplo para o mercado de capitais (MCSEG) ... 100

LISTA DE TABELAS

Tabela 2.1 - Comparação entre arquiteturas tradicionais e SOA 18
Tabela 2.2 - Padrões/especificações de segurança para uma aplicação *Web* SOA 23
Tabela 2.3 - Diferenças entre sistemas simétricos e assimétricos 25
Tabela 2.4 - Comparação dos modos de segurança nó-a-nó e fim-a-fim 32
Tabela 3.1 - Dez vulnerabilidades de segurança mais freqüentes em aplicações *Web* 41
Tabela 4.1 - Serviços da MCSEG 100
Tabela A.1 - Técnicas de proteção a vulnerabilidades de aplicações *Web* tradicionais 119

SUMÁRIO

1 INTRODUÇÃO 12

1.1 MOTIVAÇÕES 12

1.2 OBJETIVOS 14

1.3 ORGANIZAÇÃO DO TRABALHO 15

2 REVISÃO DOS PRINCIPAIS CONCEITOS 17

2.1 FUNDAMENTOS DE SOA 17

2.2 PRINCÍPIOS FUNDAMENTAIS DE SEGURANÇA 21

2.3 CRIPTOGRAFIA: ALGORITMOS SIMÉTRICOS E ASSIMÉTRICOS 24

2.4 ASSINATURAS DIGITAIS 26

2.5 CERTIFICADOS DIGITAIS 27

2.6 INFRAESTRUTURA DE CHAVE PÚBLICA 29

2.7 CHAVES DE SESSÃO 30

2.8 MODOS DE SEGURANÇA 31

2.8.1 Segurança nó-a-nó *versus* segurança *fim-a-fim* 31

2.8.2 WS-Security e a segurança fim-a-fim 33

2.9 ATESTAMENTO 35

3 SEGURANÇA EM APLICAÇÕES *WEB* 37

3.1. INTRODUÇÃO 37

3.2. ARQUITETURA DE UMA APLICAÇÃO *WEB* TRADICIONAL 37

3.3. TIPOS DE ATAQUE E TÉCNICAS DE PROTEÇÃO EM APLICAÇÕES *WEB* TRADICIONAIS 39

3.4. ARQUITETURA DE UMA APLICAÇÃO *WEB* SOA 42

3.5. TIPOS DE ATAQUE E TÉCNICAS DE PROTEÇÃO EM APLICAÇÕES *WEB* SOA..... 44

3.6. ATESTAMENTO COMO UM DOS FUNDAMENTOS À SEGURANÇA EM *WEB* SOA 51

4 ARCABOUÇO DE ATESTAMENTO PARA APLICAÇÕES *WEB* SOA 52

4.1. INTRODUÇÃO 52

4.2. REQUISITOS GERAIS DO ARCABOUÇO DE ATESTAMENTO 54

4.3. DIAGRAMA DE COMPONENTES PARA O ARCABOUÇO DE ATESTAMENTO 55

4.4. PROGRAMAS TRATADORES PARA INTERMEDIAÇÃO DAS COMUNICAÇÕES NO ARCABOUÇO 59

4.5. ETAPAS DE ATESTAMENTO 61

4.5.1. Verificação da integridade das camadas nas estações dos consumidores 62

4.5.1.1. PROCESSO TÍPICO DE INICIALIZAÇÃO DE UM COMPUTADOR.....63

4.5.1.2. VALIDAÇÃO DAS CAMADAS DE INICIALIZAÇÃO.....63

4.5.1.3. DESAFIO À VALIDAÇÃO DAS CAMADAS DE INICIALIZAÇÃO.....70

4.5.2. Geração de chaves simétricas de sessão sincronizadas 71

4.5.3. Autenticação via certificado e assinatura digital dos consumidores.....75

4.5.4. Regulação do grau de confiança sobre os consumidores 77

4.5.4.1. CONFIANÇA.....77

4.5.4.2. MOTIVAÇÃO PARA A MONITORAÇÃO DO GRAU DE CONFIANÇA.....79

4.5.4.3. MONITORAÇÃO DO GRAU DE CONFIANÇA COMO ETAPA DE ATESTAMENTO.....80

4.5.4.4. REQUISITOS DO ALGORITMO DE REGULAÇÃO DO GRAU DE CONFIANÇA.....83

4.5.4.5. SEGURANÇA DO AGENTE REGULADOR.....86

4.5.4.6. DESEMPENHO E DISPONIBILIDADE DO AGENTE REGULADOR 87

4.6. TÉCNICA PARA EVITAR USO DO *TOKEN* POR USUÁRIOS NÃO AUTORIZADOS 92

4.7. REQUISITOS DO *TOKEN* DE SEGURANÇA 93

4.8. REQUISITOS DA BIBLIOTECA DE SUPORTE 94

4.9. REQUISITOS DAS ESTAÇÕES DOS CONSUMIDORES 96

4.9.1 Instalação de programas na estação do consumidor 96

4.9.2 Privilégios no ambiente das estações dos consumidores 98

4.9.3 Habilitação de um canal de comunicação com o *token* de segurança 98

4.10. CONSIDERAÇÕES SOBRE A IMPLANTAÇÃO DO ARCABOUÇO 99

4.10.1 Características das aplicações candidatas à implantação do arcabouço 99

4.10.2 Discussão de um caso prático para uso do arcabouço 100

5 TRABALHOS RELACIONADOS E CONTRIBUIÇÕES DESTE TRABALHO 105

6 CONCLUSÕES 109

7 DESAFIOS PARA TRABALHOS FUTUROS 112

REFERÊNCIAS 114

REFERÊNCIAS COMPLEMENTARES 118

APÊNDICE A - PROTEÇÃO EM APLICAÇÕES *WEB* TRADICIONAIS 119

1 INTRODUÇÃO

1.1 Motivações

Atualmente, a arquitetura SOA (*Service Oriented Architecture*) já se consolidou com uma participação bastante significativa no mercado e ainda conta com um grande potencial de crescimento para soluções comerciais, incluindo as fortemente dependentes de segurança da informação. O Gartner Research Group sugere que as organizações deverão adotar as tecnologias para a implantação de SOA para evitar o risco de ficarem fora do mercado (SHAH, 2002). Pesquisas recentes indicam que os negócios com SOA atingem o volume atual aproximado de 35 bilhões de dólares ao ano no mercado mundial.

Esse cenário é a conseqüência lógica das vantagens deste novo paradigma, destacadas por Gutiérrez; Medina e Piattini (2004):

- Tecnologia de *middleware* padronizada;
- Alto nível de reusabilidade dos serviços de negócios;
- Reutilização facilitada de sistemas legados;
- Integração entre plataformas de sistemas heterogêneas.

Em função desses benefícios, muitos departamentos de TI (Tecnologia da Informação) das organizações estão implementando SOA com o objetivo prioritário de rapidamente operacionalizar a nova solução, nem sempre dedicando a devida atenção aos problemas relacionados à segurança.

Neste contexto, agravado pelas vulnerabilidades características das implementações sob o paradigma orientado a serviços, a preocupação com a área de segurança tem recebido importante peso no desenho das soluções SOA.

Como exemplo de vulnerabilidade característica do paradigma, *Web services*, que representam a tecnologia predominante para a implementação de serviços em arquiteturas SOA, podem atuar como portais de entrada às redes das corporações, permitindo que usuários externos mal intencionados acessem bases de dados e aplicações.

Entre as questões que revelam a importância dos aspectos de segurança em SOA, pode-se citar:

- Anteriormente a SOA, a quantidade de consumidores de um serviço podia ser rápida e facilmente potencializada pela arquitetura?
- Ao aumentar a população de consumidores, qual a probabilidade de trazer consigo um maior número de ofensores?
- Em arquiteturas tradicionais, anteriores a SOA, seria simples a descoberta dos contratos de serviços providos por um determinado servidor?

Considerando a condição das redes distribuídas de serviços, nas quais consumidores são programas e não pessoas, surgem novas vulnerabilidades que necessitam ser endereçadas. O volume de consumidores, na forma de componentes de *software*, tem potencial para ser muito maior que nas arquiteturas anteriores a SOA, o que aumenta a população a interagir com os provedores de serviços. Essa população maior de consumidores traz consigo maiores perspectivas de ataque, muitas vezes tomando por base a publicação das características de cada serviço, como suas interfaces, tipos de dados e localização.

Para responder a essa condição de vulnerabilidade, faz-se necessária a aplicação de conceitos de segurança que não podem se limitar às abordagens tradicionais tipicamente implementadas através de soluções simplificadas de autenticação, confidencialidade, integridade e demais princípios de segurança.

Neste trabalho, o foco é dado sobre o aspecto da oferta de serviços a um grande número de consumidores, o que intensifica a importância do atestamento dos programas ou componentes que se utilizam dos serviços de um servidor. Por atestamento deve-se entender um processo de verificação da legitimidade dos elementos participantes de uma cadeia de acesso (*hardware*, *software* e indivíduos), representando um conceito estendido à autenticação tradicional. Como definido no arcabouço especificado neste trabalho, trata-se de validações que se complementam para demonstrar a legitimidade dos consumidores perante os provedores de serviços.

Como resultado deste trabalho, é definido um arcabouço composto por quatro etapas de atestamento dos consumidores perante os provedores de serviços. A utilização desse arcabouço traz a contribuição de uma abordagem estruturada para

a validação de legitimidade dos consumidores de serviços, resultando na necessidade de um aumento dos esforços de ataque para violar a segurança dos provedores.

Como desafios e sugestão de trabalhos futuros são apresentados aspectos a serem detalhados para a implantação do arcabouço. Entre estes desafios, pode-se citar a implementação de um dispositivo de *hardware* (*token* de segurança) que suporta a operação do arcabouço e a codificação de um mecanismo eficiente no lado do provedor para manter as validações de integridade das estações dos consumidores diante das freqüentes atualizações nas camadas de inicialização destas estações.

1.2 Objetivos

Primeiramente, é importante definir que, para os objetivos deste trabalho, foi assumida a plataforma *Web* como ambiente de implementação de soluções SOA. A escolha dessa plataforma se justifica por ser a predominante nos sistemas orientados a serviços, delimitando o escopo da pesquisa realizada neste trabalho.

A partir de um estudo sobre os aspectos de segurança em aplicações *Web*, identifica-se que as implementações tradicionais aplicam o princípio de autenticidade para assegurar ao lado cliente que o servidor invocado é legitimo. Como exemplo, pode-se citar o protocolo SSL (*Secure Sockets Layer*) que visa prover os princípios de confidencialidade, integridade e autenticidade das mensagens trafegadas em uma comunicação entre cliente e servidor. No que se referem à autenticidade, as implementações típicas de SSL recorrem a uma solução de *PKI* (*Public Key Infrastructure*) que certifica digitalmente o servidor como supostamente legítimo perante uma autoridade certificadora confiada pelo cliente. Conforme apresentado por Harris (2002), embora tecnicamente esse protocolo possa ser utilizado para também certificar o cliente, esta não é a implementação usual.

Neste cenário em que as soluções típicas de *Web commerce* zelam essencialmente pela legitimidade do provedor perante o consumidor de serviços, o

objetivo deste trabalho é definir um arcabouço para validar a legitimidade dos consumidores perante o provedor. Através deste arcabouço, denominado ASACS (Arcabouço para Segurança por Atestamento dos Consumidores de Serviços), soluções de *Web commerce* que tipicamente validam a autenticidade dos provedores através de certificados digitais poderão também implementar um processo de atestamento dos consumidores. Trata-se de validar se um determinado consumidor de serviços é legítimo, resultando em negação de acessos ao provedor quando o consumidor não é capaz de prover as informações de atestamento requeridas.

O arcabouço definido neste trabalho tem como principal contribuição a proposta de uma abordagem estruturada para a implementação de etapas de atestamento dos clientes em soluções *Web* SOA, resultando na necessidade de um aumento do esforço para os ataques aos provedores de serviços.

Quanto à sua aplicabilidade, a implementação do ASACS é recomendada a grandes aplicações e que efetivamente se caracterizam como de segurança crítica. Considerando-se as variáveis de custo e complexidade associadas à aplicação do arcabouço, os consumidores de serviços em uma implantação prática do ASACS devem ser tipicamente empresas. Nas empresas, ao contrário dos ambientes de computadores das pessoas físicas, é possível uma administração mais eficiente para execução de procedimentos e cumprimento das políticas de segurança requeridas à operação do arcabouço.

1.3 Organização do trabalho

A Seção 2 apresenta uma revisão dos principais conceitos utilizados no desenvolvimento deste trabalho. Nessa seção são apresentados os fundamentos de SOA, princípios tradicionais de segurança, tipos de algoritmos de criptografia, recursos de segurança como assinaturas e certificados digitais, a infraestrutura de chave pública, o conceito de chaves de sessão, modos de segurança nó-a-nó *versus* fim-a-fim, uma introdução ao padrão WS-Security e a definição de atestamento.

Na Seção 3 é apresentado inicialmente um estudo das características de segurança em implementações tradicionais na plataforma *Web*, incluindo uma breve descrição dos tipos mais comuns de ataques e de técnicas de proteção. Também na Seção 3 são descritos os ataques e técnicas de proteção para soluções *Web* SOA que, além da suscetibilidade aos riscos de segurança das implementações *Web* tradicionais, são também alvo de abordagens de ataque particulares das soluções orientadas a serviços. Por fim, é apresentada a importância do atestamento dos consumidores de serviços como uma solução eficiente para a segurança em aplicações *Web* SOA.

A Seção 4 apresenta a proposta central deste trabalho, que é a definição de um arcabouço para atestamento em soluções *Web* SOA, fundamentado em quatro etapas para validação dos clientes consumidores em uma rede distribuída de serviços. Nessa seção, as quatro etapas de atestamento são classificadas em três etapas de validação de legitimidade e uma etapa de validação de comportamento. Também nessa seção é apresentado o conjunto de requisitos gerais a serem cumpridos pelo arcabouço, seguindo-se uma descrição detalhada de cada uma de suas etapas. São também apresentados os requerimentos para a operação do arcabouço, o que inclui os requisitos de um dispositivo de *hardware* (*token* de segurança), os requisitos de uma biblioteca de suporte e as características necessárias às estações dos consumidores. Ainda na Seção 4 é feita uma discussão sobre um caso prático sugerido para uso do arcabouço.

A Seção 5 é reservada à análise de trabalhos relacionados e também para apresentação das contribuições deste trabalho.

A Seção 6 apresenta as conclusões, com um resumo das características do arcabouço como uma abordagem eficiente para a segurança em uma arquitetura orientada a serviços.

A Seção 7 descreve os principais desafios para trabalhos futuros, seguida pelas referências bibliográficas e um apêndice com exemplos das vulnerabilidades e técnicas de proteção mais comuns em aplicações *Web* tradicionais.

2 REVISÃO DOS PRINCIPAIS CONCEITOS

2.1 Fundamentos de SOA

De acordo com a *World Wide Web Consortium* (W3C), a Arquitetura Orientada a Serviços é uma metodologia para alcançar interoperabilidade de aplicações e reutilização de ativos de TI que caracterizam um forte foco arquitetural, um nível ideal de abstração, uma infraestrutura de implantação e uma biblioteca de serviços reutilizáveis (BOOTH et al., 2003, apud ABUOSBA; EL-SHEIKH, 2008)[1].

A Organização para o Avanço de Padrões de Informação Estruturados (OASIS), por sua vez, define SOA como "um paradigma para organização e utilização de capacidades distribuídas que podem estar sob o controle de diferentes domínios de propriedade" (OASIS, 2006, p. 8, tradução nossa). De acordo com Abuosba e El-sheikh (2008), esta definição da OASIS fundamenta-se em três dimensões: *visibilidade*, *interação* e *efeito*, representados na Figura 2.1, onde o consumidor obtém efeitos do provedor de acordo com as capacidades que atendem às suas necessidades.

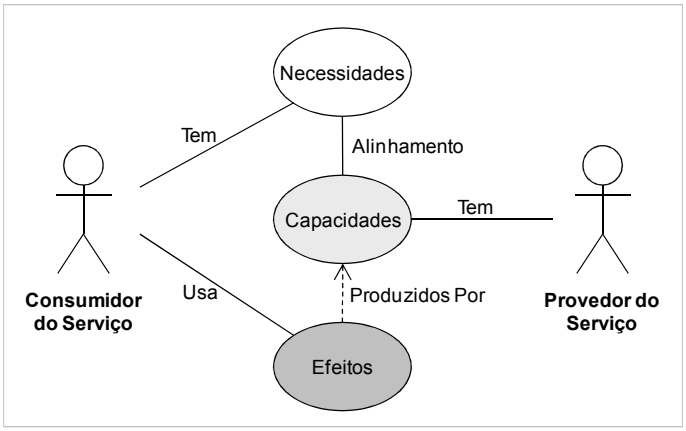

Figura 2.1 - Casos de uso UML para SOA (ABUOSBA; EL-SHEIKH, 2008)

Visibilidade refere-se à habilidade do consumidor e provedor para interagir e realizar um determinado serviço. Esta dimensão define o casamento entre as necessidades do consumidor e as capacidades do provedor. *Interação*, por sua vez,

[1] BOOTH, D.; CHAMPION, M.; FERRIS, C.; MCCABE, F.; NEWCOMER, E.; ORCHARD, D. Web Services Architecture, W3C working draft, 8 Aug. 2003. Disponível em: <http://www.w3.org/TR/2003/WD-ws-arch-20030808>. Acesso em: 12 out. 2009.

refere-se ao uso pelo consumidor de uma das capacidades do provedor, o que é intermediado através de uma troca de mensagens, produzindo um determinado *efeito* (ou conjunto de efeitos).

Segundo uma perspectiva de negócios, Ganci et. al. (2006) definem SOA como um conjunto de serviços compostos para atender o modelo de negócios que a organização deseja expor internamente e também externamente a seus clientes e parceiros.

O fato é que SOA pode ter diferentes definições sob múltiplos pontos de vista que podem ser encontrados a partir de várias fontes, sejam acadêmicas ou não. Diante desta diversidade de definições, um modo efetivo para se entender SOA é através de uma comparação com arquiteturas tradicionais. A Tabela 2.1 representa uma comparação simples entre uma arquitetura tradicional e uma arquitetura SOA.

Tabela 2.1 - Comparação entre arquiteturas tradicionais e SOA

Arquitetura tradicional	**Arquitetura SOA**
Implementações sem foco ao suporte a mudanças	Implementações com forte suporte a mudanças
Aplicações em silos	Soluções orquestradas que trabalham juntas
Alto acoplamento	Aplicações fracamente acopladas
Aplicações estruturadas pelo uso de componentes e objetos	Aplicações estruturadas pelo uso de serviços reutilizáveis
Tecnologia de implementação previamente conhecida	Abstração e independência das tecnologias de implementação
Conjunto de usuários das aplicações tipicamente conhecido, com crescimento e volume mais controlados	Conjunto de consumidores dos serviços com forte potencial de crescimento, o que é facilitado pela publicação de interfaces

Para o objetivo do trabalho aqui proposto, as características de SOA mais relevantes são seus fundamentos de reutilização e abertura do acesso aos serviços para um número de consumidores maior que nas arquiteturas tradicionais, que tipicamente restringiam as possibilidades de interação a um número limitado e mais controlado de usuários.

SOA também pode ser compreendida em seu nível mais básico como uma arquitetura composta pelos seguintes elementos:

- Provedor do serviço
- Consumidor do serviço
- Registro do serviço

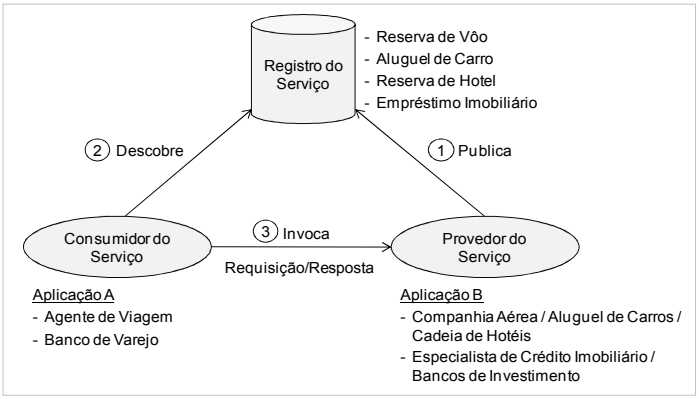

Figura 2.2 - Operações e componentes básicos de SOA (GANCI et al., 2006)

Quando presente, o *registro de serviços* é responsável por disponibilizar aos consumidores a interface e informações de acesso aos serviços.

Os *consumidores*, por sua vez, localizam (descobrem) as entradas de um registro de serviço em um repositório e então se conectam ao provedor para invocar as operações publicadas por esse serviço.

O *provedor de serviços* é o responsável pela criação de um serviço, podendo publicar sua interface e informações de acesso em um repositório (*registro de serviços*). É o provedor de serviços que deve definir as características de um serviço como segurança, disponibilidade, precificação do uso e outros atributos funcionais ou técnicos.

Para os objetivos deste trabalho, é considerada a plataforma *Web* para a discussão dos conceitos de SOA. Consequentemente, os provedores de serviços serão, a partir deste ponto, denominados como *Web services*.

Os *Web services* são os provedores de serviço para soluções SOA em plataforma *Web* e, para seu funcionamento, requerem um conjunto de componentes e tecnologias conforme definido por Sidharth e Liu (2007):

- *Universal Discovery Description and Integration* (UDDI): especificação para definir os registros de serviços para descoberta pelos consumidores. Através dos *Universal Business Registries* (UBRs), o UDDI define um meio de

publicação e descoberta de informações sobre *Web services*, o que inclui a disponibilização de arquivos em *Web Services Description Language* (WSDL);

- *WSDL*: um formato XML para descrever *Web services* e disponibilizado nos registros de serviço para consulta via UDDI. Descreve a estrutura de um *Web service* através de dados formatados em XML, incluindo:
  - Nome e localização do serviço;
  - Nomes dos métodos (operações) e os tipos de seus argumentos;
  - Tipo do valor de retorno;
  - Documentação sobre o serviço.
- *Simple Object Access Protocol* (SOAP): protocolo baseado em XML para a troca de informações na *Web*, tipicamente utilizado para invocar *Web services*. É um mecanismo utilizado para troca de mensagem entre aplicações e que implementa chamadas remotas de procedimentos. SOAP é independente da plataforma e da linguagem das aplicações e é usualmente utilizado em conjunção com o protocolo HTTP.

Ainda como parte de soluções SOA completas, recentemente tem se destacado o papel do *Enterprise Service Bus* (ESB). O ESB é um barramento para integração de diferentes tipos de serviços através de mensageria, tratamento de eventos e gestão do desempenho de negócio. GANCI et al. (2006) definem o ESB como um barramento que provê uma infraestrutura para remover a dependência de uma conexão direta entre consumidores e provedores. Os consumidores se conectam ao ESB e não ao provedor que efetivamente implementa o serviço, favorecendo o desacoplamento. Através dos ESBs podem ainda ser implementadas as capacidades de segurança e garantia de entrega, bem como funções de *log*, monitoramento e gerenciamento de sistemas que tipicamente devem ser implementadas centralizadamente, ao invés de uma abordagem com implementações espalhadas pelas aplicações.

2.2 Princípios fundamentais de segurança

Segurança em computadores pode ser definida como “evitar que os atacantes alcancem seus objetivos através de acesso ou uso não autorizado de computadores e redes” (HOWARD, 1997, p. 43, tradução nossa). Em (CHESWICK; BOLLOVIN, 1994, tradução nossa), a segurança é definida como “evitar que qualquer um faça coisas que você não quer que façam para, com, sobre ou a partir de seus computadores ou quaisquer periféricos”, o que permite a interpretação de que a segurança de computadores não é um objetivo em si, mas sim um meio para se atingir um objetivo que é a segurança da informação.

Alternativamente, a segurança de computadores pode ser compreendida a partir de princípios que foram definidos em resposta aos problemas de segurança identificados ao longo do tempo e que foram intensificados pela computação em rede e especialmente potencializados com a expansão das aplicações *Web*. Trata-se de um conjunto de sete princípios que podem ser entendidos como um ponto de partida para o desenvolvimento de soluções seguras (HARRIS, 2002):

- Irretratabilidade: refere-se à comprovação das transações do emissor em uma comunicação, garantindo que o emissor não possa negar ter enviado uma determinada mensagem.
- Identificação: descreve um método para checar se um usuário, programa ou processo é realmente a entidade que declara ser. A identificação pode ser provida, por exemplo, com a utilização de nome do usuário ou número de conta.
- Autenticação: complementa a identificação ao requerer uma segunda informação de credencial como, por exemplo, uma senha, um número de identificação pessoal ou o uso de um dispositivo de *hardware.* O conjunto completo das credenciais é então comparado com as informações previamente armazenadas para cada usuário, que passa então ao estado de autenticado perante o sistema.

- Autorização: trata da atribuição dos direitos e privilégios a cada usuário autenticado. A autorização valida se um dado usuário pode acessar um recurso ou executar as ações solicitadas em suas interações com o sistema.
- Confidencialidade: corresponde à proteção contra a exposição de informações a indivíduos, programas ou processos não autorizados. Entre os mecanismos que provêem confidencialidade, pode-se citar a criptografia, controles de acesso lógicos e físicos, segregação de perfis para visão de uma base de dados, entre outros.
- Integridade: refere-se à certificação de que as informações não são alteradas de modo não autorizado. A integridade tem especial importância sobre as mensagens em trânsito, buscando certificar que seu conteúdo não foi adulterado, o que resultaria na interpretação e processamento incorretos pelo receptor.
- Administração das políticas: refere-se à administração das políticas de segurança definidas para uma dada organização. As políticas de segurança podem ser entendidas como um conjunto de regras e práticas que determinam como as informações e recursos devem ser gerenciados, protegidos e distribuídos. Através das políticas de segurança são definidos os fundamentos para especificação dos sistemas e processos em uma organização, definindo, por exemplo, qual o nível de segurança requerido para um sistema e que ações devem ser tomadas quando estes requisitos não são cumpridos. Para estruturar as políticas de segurança, é necessária a especificação de padrões, guias de referência, procedimentos, papéis e responsabilidades, bem como a definição da abordagem de testes para validar que uma dada implementação está aderente às políticas definidas.

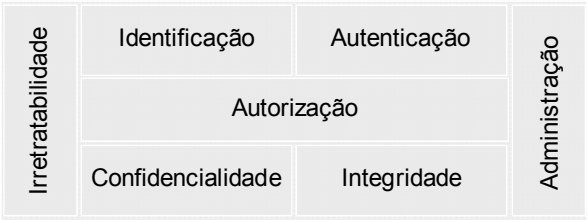

Figura 2.3 - Princípios de segurança

Em um primeiro momento de aplicação destes princípios de segurança sobre soluções SOA, as organizações recorreram a implementações específicas que

resultaram em abordagens proprietárias e de alto acoplamento. Como exemplo, pode-se citar a disponibilização pela Google no início da década de uma versão *Web service* de seu motor de buscas para um pequeno grupo de parceiros de negócios. Para o uso deste serviço, os consumidores tinham que se registrar previamente através de um formulário HTML. Como parte deste processo de registro, a Google enviava ao solicitante um *e-mail* com uma cadeia de caracteres secreta de segurança. Quando os consumidores invocavam o serviço, tinham que informar essa cadeia de caracteres secreta como parte da mensagem SOAP para verificação se eram usuários autorizados e registrados para o *Web service*.

Como parte do amadurecimento do uso dos princípios de segurança em SOA, foram definidos padrões que evitam a implementação de soluções altamente acopladas e que sejam específicas a uma dada organização.

Estes padrões definem meios e regras para o desenvolvimento de *Web services* seguros e foram elaborados por instituições de pesquisa e, em alguns casos, inclusive com a colaboração de empresas do setor privado. A Tabela 2.2 apresenta um resumo deste conjunto de padrões e os princípios de segurança que cada padrão visa atender.

Tabela 2.2 - Padrões/especificações de segurança para uma aplicação *Web* SOA

Padrão / Especificação	**Princípios de Segurança**	**Descrição**
SSL	Confidencialidade, Integridade, Autenticação, Irretratabilidade	A especificação SSL define um protocolo de segurança TCP/IP usado para comunicação *Web* segura na camada de transporte e é tipicamente conjugado com HTTP, resultando no HTTPS.
XML-Encryption	Confidencialidade, Integridade	Padrão para criptografia de mensagens XML SOAP.
XML-Signature	Integridade, Autenticação, Irretratabilidade	Padrão para geração de *hash* e assinatura de mensagens XML SOAP.
XKMS	Administração de políticas	*XML Key Management Standard* é uma especificação que permite aos *Web services* SOAP o registro e gestão de chaves criptográficas utilizadas para assinaturas digitais e criptografia.

(continua)

(continuação)

Padrão / Especificação	Princípios de Segurança	Descrição
SAML	Autenticação, Autorização	*Security Assertion Markup Language* é um padrão que permite a troca de informações de autorização e autenticação. Define asserções que são validadas pelo provedor para autorizar ações sobre um determinado serviço.
XACML	Autorização	*Access Control Markup Language* é um padrão em desenvolvimento para definição de políticas de autorização para solicitações de *Web services* SOAP. Define extensões ao SAML que permitem regras de autorização complexas.
WS-Security	Confidencialidade, Integridade, Autenticação, Irretratabilidade	*Web Services Security* é um padrão em expansão desenvolvido pelos principais membros da indústria (IBM, Microsoft, e VeriSign) que define principalmente como usar padrões de assinatura e criptografia de XML em mensagens de *Web services* SOAP.

2.3 Criptografia: algoritmos simétricos e assimétricos

Segundo Harris (2002), de uma maneira simplificada, pode-se definir que o processo de criptografia é composto por duas peças principais: o algoritmo e a chave. Os algoritmos correspondem a fórmulas matemáticas complexas que definem as regras de como um texto puro se transforma em um texto cifrado. As chaves, por sua vez, são seqüências de caracteres aleatórias que são utilizadas pelos algoritmos, proporcionando que os textos cifrados sejam efetivamente randômicos.

Os algoritmos que utilizam uma única chave entre emissor e receptor para as funções criptográficas são denominados simétricos, enquanto que aqueles que utilizam chaves distintas, embora relacionadas, para criptografia e decriptografia são denominados assimétricos.

Como pontos fortes da criptografia simétrica, pode-se destacar a velocidade dos algoritmos em relação aos sistemas assimétricos e sua força contra violações quando utilizadas chaves de tamanho grande. Como pontos fracos, destacam-se a não cobertura do princípio de irretratabilidade, a necessidade de um mecanismo seguro para distribuição das chaves e o requisito de uma chave específica para

cada par de usuários, dificultando a administração em um ambiente com alto volume de participantes. Exemplos de algoritmos simétricos são o *Advanced Encryption Standard* (AES) e o *Triple Data Encryption Standard* (3DES).

Por outro lado, na criptografia assimétrica cada participante conta com um par de chaves matematicamente relacionadas, denominadas chave pública e chave privada. Quando uma mensagem é criptografada com uma das chaves, a outra chave é requerida para decriptografá-la. Enquanto a chave privada deve ser de conhecimento exclusivo de uma das partes, a chave pública é revelada para conhecimento de qualquer participante que queira criptografar e decriptografar as mensagens trocadas com o participante emissor desta chave pública. Como pontos fortes da criptografia assimétrica, destaca-se o mecanismo de distribuição de chave, melhor escalabilidade e atendimento ao princípio de irretratabilidade (a decriptografia com a chave pública do emissor comprova que a mensagem foi criptografada com a chave privada associada, a qual tem o pressuposto de ser de conhecimento exclusivo daquele emissor). Como ponto fraco, destaca-se o processamento mais lento que os algoritmos simétricos. Exemplos de algoritmos assimétricos são o *RSA, Digital Signature Algorithm (DSA)* e o *Elliptic Curve Cryptosystem* (ECC).

A Tabela 2.3 resume as diferenças entre sistemas de criptografia simétricos e assimétricos.

Tabela 2.3 - Diferenças entre sistemas simétricos e assimétricos (HARRIS, 2002)

Atributo	Sistema Simétrico	Sistema Assimétrico
Distribuição de chaves	Dependência de mecanismos externos seguros	A chave pública é disponibilizada a qualquer participante e a chave privada é mantida em segredo pelo seu detentor
Chaves	Uma chave é compartilhada entre duas ou mais entidades	Uma entidade tem a chave privada, enquanto a outra tem a correspondente chave pública
Velocidade	Algoritmos menos complexos e mais rápidos	Algoritmos mais complexos e mais lentos
Uso	Criptografia de grandes volumes de dados	Distribuição de chaves e assinatura digital
Princípios de segurança	Confidencialidade	Confidencialidade e irretratabilidade

A assinatura digital referida na Tabela 2.3 corresponde a uma solução para implementação do princípio de irretratabilidade e seu funcionamento está descrito na próxima seção.

2.4 Assinaturas digitais

As assinaturas digitais correspondem à abordagem mais comumente utilizada para a implementação conjugada dos princípios de integridade, autenticidade e irretratabilidade das mensagens trafegadas entre emissor e receptor.

O esquema padrão de assinatura digital se baseia na aplicação de criptografia assimétrica, com a utilização de chaves privadas e públicas do emissor para, respectivamente, criptografar e decriptografar um valor *hash*. Um valor *hash*, também conhecido como *message digest*, corresponde ao resultado da transformação processada sobre uma mensagem através de uma função *hash*. Uma função *hash* recebe como entrada uma cadeia de caracteres de comprimento variável e produz um valor de comprimento fixo denominado valor *hash*. Sua utilização para implementar o princípio de integridade pressupõe que ambas as partes da comunicação utilizem o mesmo algoritmo de *hash,* habilitando o receptor a aplicar a mesma transformação sobre a mensagem e comparar seu resultado com o valor *hash* recebido.

Segundo Harris (2002), tecnicamente, as assinaturas digitais podem ser definidas como valores *hash* criptografados utilizados para validar a integridade da mensagem trafegada (sem alteração enquanto esteve em trânsito), bem como para assegurar a origem (o emissor que detém a chave privada). Para o funcionamento das assinaturas digitais, o algoritmo de criptografia assimétrica é utilizado para prover autenticidade e irretratabilidade, enquanto o algoritmo de *hash* é aplicado para validar a integridade da mensagem. A Figura 2.4 representa os passos típicos de uma assinatura digital. Nestes passos não estão representados os processos de criptografia da mensagem propriamente dita e sim apenas os passos da assinatura digital.

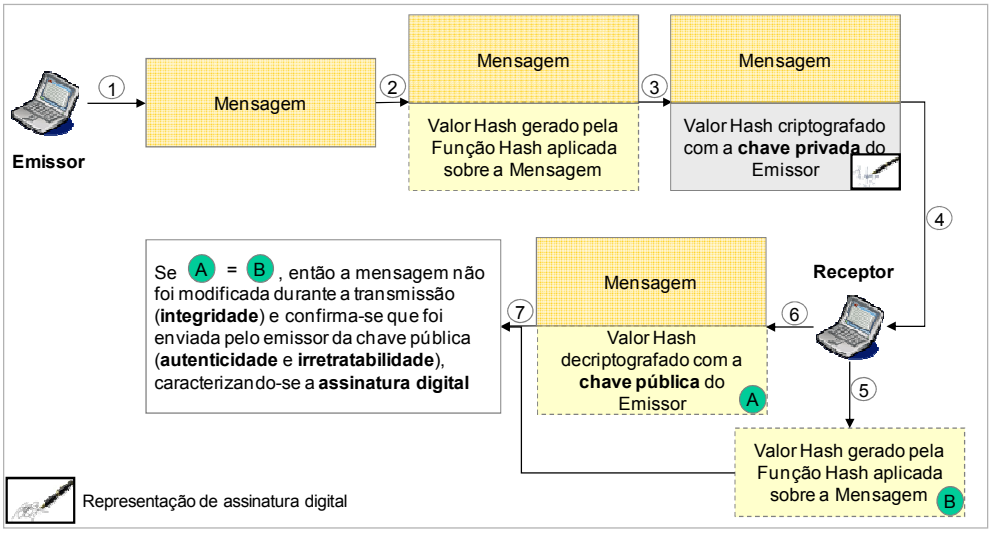

Figura 2.4 - Passos da assinatura digital

Importa referir que dada a importância das assinaturas digitais para a aplicação dos princípios de integridade, autenticidade e irretratabilidade, foi definido um padrão internacional para a solução, denominado *Digital Signature Standard* (DSS), proposto pelo NIST[1] em 1991. O padrão determina os seguintes algoritmos para prover a correta combinação de serviços de segurança: o DSA (*Digital Signature Algorithm*), RSA ou ECDSA (algoritmo de assinatura digital por curva elíptica) e SHA (*Secure Hash Algorithm*). O DSA, RSA ou ECDSA cumprem o papel de algoritmo de criptografia assimétrica, provendo a validação de autenticidade e o princípio de irretratabilidade. O SHA, por sua vez, é aplicado para certificar a integridade através da comparação entre o valor *hash* enviado pelo emissor e o gerado pelo receptor a partir da aplicação da mesma função *hash* sobre a mensagem comunicada.

2.5 Certificados digitais

Os certificados digitais podem ser definidos como uma solução para delegação de confiança a uma terceira entidade agregada em uma comunicação entre emissores e receptores. Nas aplicações que utilizam certificados digitais, cada uma das partes de uma comunicação tradicional (o emissor ou o receptor) pode recorrer a uma entidade denominada autoridade certificadora. A autoridade certificadora, por

[1] National Institute of Standards and Technology, fundado em 1901, como uma agência federal não regulatória dentro do Departamento de Comércio dos Estados Unidos.

sua vez, mantém uma base de certificados que são então validados a pedido do emissor ou receptor.

Assumindo o exemplo de comprovação do certificado de um servidor perante o cliente, o servidor envia, no início da comunicação, as informações de seu certificado ao cliente. O cliente recorre à autoridade certificadora (entidade em quem confia) que então valida o certificado apresentado pelo servidor. Em caso de validação positiva do certificado, o cliente então assume que o servidor é legítimo, o que é inferido por dispor de um certificado digital emitido por autoridade certificadora em quem confia.

Harris (2002) define certificado digital como o mecanismo para associar uma chave pública a uma coleção de componentes suficientes para identificar unicamente o dono do certificado. Para criar um certificado, as autoridades certificadoras utilizam um padrão denominado X.509, o qual determina os campos e valores válidos para seu preenchimento em um certificado digital. Os campos de um certificado digital incluem informações de identificação do emissor e do dono do certificado, um identificador único junto ao emissor (autoridade certificadora), versão do certificado, prazo de validade, algoritmo utilizado para assinatura digital, a própria assinatura digital do certificado pelo emissor e a chave pública do dono do certificado.

A Figura 2.5 representa o fluxo de comunicação entre emissor e receptor, recorrendo-se à autoridade certificadora para validação da autenticidade de cada uma da partes.

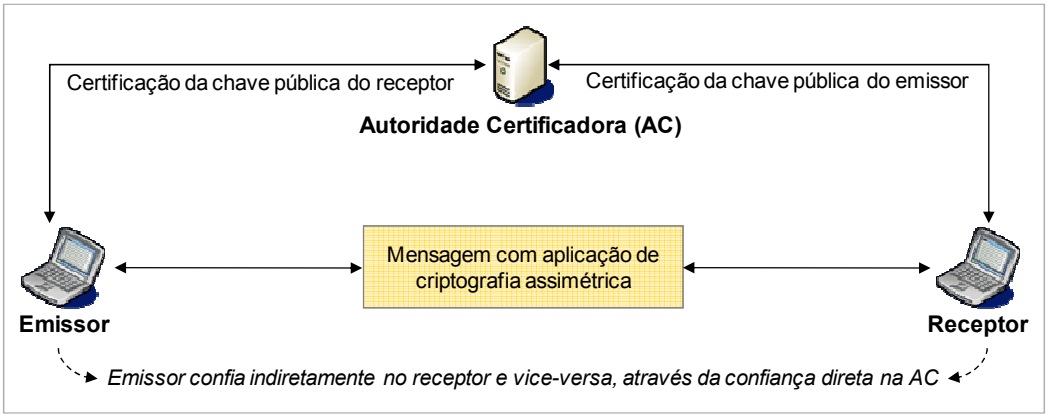

Figura 2.5 - Comunicações entre receptor, emissor e autoridade certificadora

Como referido na Seção 1.2, a utilização de certificados digitais é parte do protocolo SSL e embora tecnicamente possam ser aplicados para autenticação do lado cliente, esta não é a implementação usual, o que resulta em risco à segurança para o lado servidor de uma comunicação.

Para a simplificação dos principais conceitos requeridos ao entendimento deste trabalho, não foi representada na Figura 2.5 a Autoridade de Registro (AR) que tem a função de realizar os procedimentos necessários para registrar um novo certificado. É papel das ARs verificar a identidade do solicitante e então iniciar o processo de certificação junto à Autoridade Certificadora (AC). As ARs não emitem os certificados, o que é de responsabilidade das ACs, mas atuam como intermediárias entre os solicitantes e as ACs, requerendo todas as informações necessárias antes de encaminhar às ACs as solicitações de emissão de certificados.

2.6 Infraestrutura de chave pública

Uma maneira de compreender a infraestrutura de chave pública, conhecida como PKI (*Public Key Infrastructure*), é diferenciando-a da criptografia de chave pública. Enquanto a criptografia de chave pública corresponde ao uso de um algoritmo assimétrico, a PKI não é um algoritmo e sim uma infraestrutura baseada na criptografia de chave pública. A PKI consiste de programas, formatos de dados, procedimentos, protocolos de comunicação, políticas de segurança e mecanismos de criptografia de chave pública que são coordenados para operarem conjuntamente, possibilitando que um conjunto amplo e disperso de entidades se comunique de uma maneira segura (HARRIS, 2002). Portanto, a criptografia de chave pública dever ser entendida como uma das peças da PKI que é ainda composta por autoridades certificadoras, autoridades de registro, certificados, chaves e usuários.

A infraestrutura PKI assume que a identidade de cada participante pode ser positivamente confirmada através de certificados digitais e que uma implementação de algoritmo assimétrico tratará da distribuição das chaves. É uma infraestrutura que contém as peças que identificam os usuários, gerenciam a distribuição e manutenção das chaves de criptografia e ainda criam, mantém, distribuem e

revogam certificados, disponibilizando todas as tecnologias requeridas para autenticação e comunicação segura.

A PKI provê confidencialidade, integridade, irretratabilidade e autenticação via autoridade certificadora, constituindo-se em um sistema híbrido de métodos e algoritmos simétricos e assimétricos.

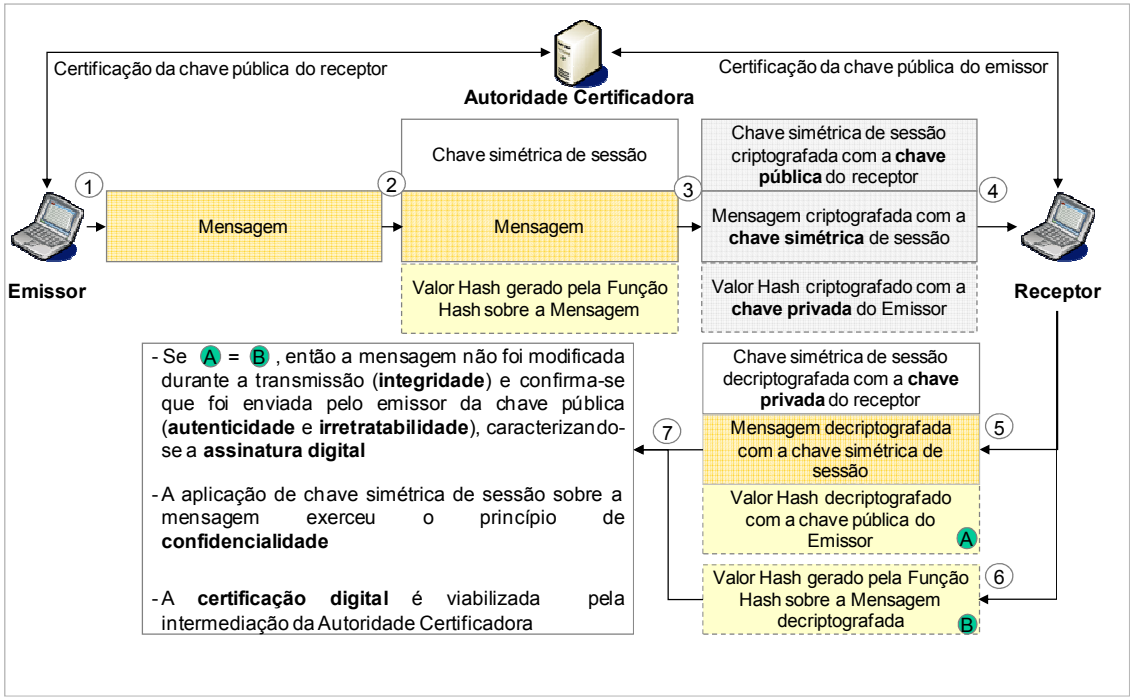

Figura 2.6 - Modelo conceitual de uma PKI

2.7 Chaves de sessão

As chaves de sessão correspondem a uma variação das chaves simétricas já definidas na Seção 2.3. A diferença consiste no período de validade da chave, sendo que as chaves de sessão devem ser válidas por apenas uma sessão de comunicação.

Ao requerer a renovação da chave a cada nova sessão, a abordagem das chaves simétricas de sessão promove um fator adicional de segurança ao reduzir o risco de comprometimento da confidencialidade pela captura indevida da chave.

2.8 Modos de segurança

2.8.1 Segurança nó-a-nó *versus* segurança *fim-a-fim*

Nesta seção são apresentadas as diferenças entre os modos de segurança que podem ser adotados como base para uma implementação do arcabouço proposto neste trabalho. Com este objetivo, são definidos os modos de segurança denominados nó-a-nó e fim-a-fim, este último também conhecido como segurança no nível da mensagem.

Harris (2002) define o modo nó-a-nó como a alternativa para promover segurança enquanto a mensagem trafega entre dois nós consecutivos de processamento SOAP. No modo nó-a-nó, quando a mensagem é recebida e reencaminhada por um nó intermediário de processamento SOAP (exemplo: uma instância de IBM Websphere ou SAP XI), que está acima da camada de transporte, a confidencialidade, integridade e qualquer outro princípio de segurança aplicado à mensagem podem ser perdidos. Neste modo de segurança não há implementação seletiva do que deve ser criptografado ou submetido a qualquer outro princípio de segurança. Assim, por exemplo, são submetidos indistintamente à criptografia todos os dados que são parte dos pacotes de comunicação, incluindo o conteúdo da mensagem propriamente dita e informações de protocolo como cabeçalhos, trailers, endereços e informações de roteamento.

O modo fim-a-fim, por sua vez, também conhecido como segurança no nível de mensagem (RAHAMAN; SCHAAD; RITS, 2006), define um contexto de segurança que não se limita aos instantes em que a mensagem está em trânsito entre nós de processamento SOAP. Neste modo de segurança, o conteúdo da mensagem permanece protegido (sem decriptografia intermediária) até o destino final. Outra diferença importante é que permite a definição seletiva das partes da mensagem que de fato precisam ser criptografadas ou submetidas a outros princípios de segurança.

As Figuras 2.7 e 2.8 representam os dois modos seguindo a taxonomia utilizada por Rahaman; Schaad e Rits (2006).

Figura 2.7 - Configuração do modo de segurança nó-a-nó

Figura 2.8 - Configuração do modo de segurança fim-a-fim

No modo de segurança nó-a-nó, representado na Figura 2.7, como as informações de protocolo, incluindo roteamento, estão criptografadas, os agentes intermediários decriptografam toda a mensagem para interpretação e prosseguimento da comunicação. Na Figura 2.8, representando o modo de segurança fim-a-fim, as informações de protocolo (como, por exemplo, informações de roteamento) estão decriptografadas, mantendo a confidencialidade apenas do conteúdo em si gerado pelo emissor e eliminando a necessidade de decriptografia nos agentes intermediários.

Uma análise comparativa entre os dois modos de segurança é resumida na Tabela 2.4 que consolida as principais vantagens de cada modo e exemplos de formas de implementação.

Tabela 2.4 - Comparação dos modos de segurança nó-a-nó e fim-a-fim

Modo de segurança	Principais vantagens	Exemplo de implementação
Nó-a-nó	• Todos os dados são criptografados, protegendo contra ações de *sniffing* sobre informações de protocolo como cabeçalhos, *trailers*, endereços e roteamento • Maturidade dos protocolos de implementação (exemplo, SSL)	SSL
Fim-a-fim	• Mais flexibilidade, ao permitir, por exemplo, a definição do que e como criptografar, possibilitando a criptografia apenas de partes críticas da mensagem • Cada intermediário na rede não precisa ter uma chave para decriptografar as mensagens em trânsito • Menos vulnerabilidade das mensagens trafegadas, uma vez que não são decriptografadas nos agentes intermediários	WS-Security

2.8.2 WS-Security e a segurança fim-a-fim

Neste trabalho, o WS-Security é referido como o padrão para implementar a segurança no nível da mensagem, considerando-se a recomendação de implantação do arcabouço com segurança fim-a-fim, independente da camada de transporte.

Nesta seção são apresentadas definições e alguns aspectos de especificação do WS-Security com o objetivo de prover os conceitos necessários ao entendimento das referências feitas a esse padrão ao longo deste trabalho.

A especificação formal da OASIS (OASIS, 2004) para o WS-Security propõe um conjunto padrão de extensões a SOAP que podem ser utilizadas durante a construção de *Web services* seguros para implementar integridade do conteúdo das mensagens e confidencialidade.

Seely (2002) define WS-Security como um padrão criado sobre um conjunto de outros padrões e especificações já existentes. WS-Security, portanto, não se propõe a definir soluções novas para a implantação dos princípios de segurança, mas sim a oferecer uma maneira de se conjugar as soluções já existentes, não se restringindo a tecnologias específicas. Assim, por exemplo, pode ser implementado utilizando Kerberos e X.509 para autenticação, XML Encryption e XML Signature para criptografia e assinatura digital e ainda a especificação XML Canonicalization para a função de preparar e padronizar os documentos XML a serem processados pela aplicação. O padrão WS-Security pode então ser entendido como um *container* de metadados dado que, em sua essência, serve para definir como devem ser inseridas em uma mensagem SOAP as descrições XML de outros padrões já existentes.

Segundo Periorellis (2008), WS-Security tem o suporte de uma massa crítica de fornecedores de soluções para SOA e será um padrão chave para a segurança de *Web services*.

Quanto à sua implementação sobre o protocolo SOAP, o WS-Security utiliza-se da possibilidade de extensão suportada pelo SOAP que permite a criação livre de itens de cabeçalho em uma dada mensagem. WS-Security especifica então um item de cabeçalho em um envelope SOAP, sob o qual são inseridos os *tags* definidos

pelos diversos padrões existentes para a implementação de segurança no nível da mensagem.

A Figura 2.9 representa um fragmento de cabeçalho de uma mensagem SOAP com as *tags* dos padrões que são conjugados pelo WS-Security para a implementação dos princípios fundamentais de segurança.


```
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://w
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsc
  - <soapenv:Header>
    - <wsse:Security soapenv:actor="http://www.jStartcustomer.com/actors#verifier" soapenv:mustUnd
        xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
      - <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        - <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          - <Reference URI="#sign_content_1043176028580">
            - <Transforms>
                <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
              </Transforms>
              <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
              <DigestValue>FLuQTa/LqDIZ5F2JSaMRHSRuaiQ=</DigestValue>
            </Reference>
          </SignedInfo>
          <SignatureValue>kGlrrXjKku/WXKxID+JJkEXY+aGNYHc5dy8GwbLFtB5Msll2/MhwdnO9wastJ0g
            7A8ggkMkjgAqnLg6PTzM7MdKoIAhe+xRHdOysamGucFJQRMrU+JQ4WATJt0bpdClwJy6me
        - <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
          - <KeyValue>
            - <RSAKeyValue>
                <Modulus>2sW+eBjx5D2QMyr8ocZIZWNYHGf9zYhB4XWILPCTvhNV7dIe3l8ARepOA1AB
                  pzb+Rb+nWQeo//yFz/28PmL63kdLiE72qmmQuzuPa5NXaV9pJ4JKw86QdLhGGpFIR
                  18Iugf3xLFwQEZqKYnblTUs7ftnTgW5r4HH492k=</Modulus>
                <Exponent>AQAB</Exponent>
              </RSAKeyValue>
            </KeyValue>
          - <X509Data>
            - <X509IssuerSerial>
                <X509IssuerName>OU=Java,O=IBM,L=Unknown,ST=Oklahoma,C=US</X509IssuerName>
                <X509SerialNumber>0</X509SerialNumber>
              </X509IssuerSerial>
              <X509SubjectName>CN=John Doe</X509SubjectName>
              <X509Certificate>MIIB0TCCAToCAQAwDQYJKoZIhvcNAQEEBQAwTzELMAkGA1UEBhMCVVM
```


Figura 2.9 - Cabeçalho de um envelope SOAP com as *tags* do WS-Security (THOMPSON, 2003)

No exemplo da Figura 2.9, a *tag* de cabeçalho “wsse:Security” representa o elemento raiz para as demais *tags* da mensagem necessárias para transformar o XML em forma canônica, processar a criptografia e assinatura (via os algoritmos RSA e SHA1) e informar o certificado digital (via o padrão X.509). Como pode ser observado neste fragmento, é no nível da mensagem que são aplicados os princípios de autenticidade, confidencialidade e integridade. Deverá ser papel do receptor final o processamento da decriptografia e validações de assinatura e integridade da mensagem. Mesmo que o circuito de rede requeira que a mensagem trafegue por nós SOA intermediários, a mensagem estará segura durante toda

comunicação, pois, ao contrário do SSL, WS-Security é um padrão para implementar a segurança no nível da mensagem, de modo independente da camada de transporte.

2.9 Atestamento

De uma maneira geral, em uma comunicação entre consumidores e provedores de serviços, o atestamento pode ser definido como uma prova de legitimidade de cada uma das partes perante a outra. Neste trabalho, o foco é dado ao atestamento dos consumidores perante os provedores de serviços e é definido como um processo de verificação da legitimidade dos elementos participantes (*hardware*, *software* e indivíduos) de uma cadeia de acesso. Trata-se de implementação de segurança via contestação de autenticidade dos componentes de *software* dos consumidores de serviços de um determinado provedor.

Segundo Mittelsdorf (2004), o atestamento pode ser classificado em tipos definidos pela localidade e instante em que está sendo realizado. Quanto à localidade, existem o atestamento local e o remoto. No atestamento local, o resultado é gerado e utilizado no mesmo equipamento, como no exemplo do atestamento resultante de uma varredura por vírus. O atestamento remoto, por sua vez, tem como objetivo garantir que um sistema remoto ou um programa em execução neste sistema é autêntico. Quanto ao instante, o atestamento pode ser classificado em estático e dinâmico. O atestamento estático é realizado sobre sistemas ou programas, antes que sejam executados, enquanto que o atestamento dinâmico é realizado com o sistema em atividade, a qualquer instante. O atestamento estático resulta em uma limitação, no sentido em que se o sistema ou programa atestado for adulterado durante sua execução, isto não será detectado enquanto não for iniciada uma nova comunicação para validação de atestamento antes da execução do sistema ou programa.

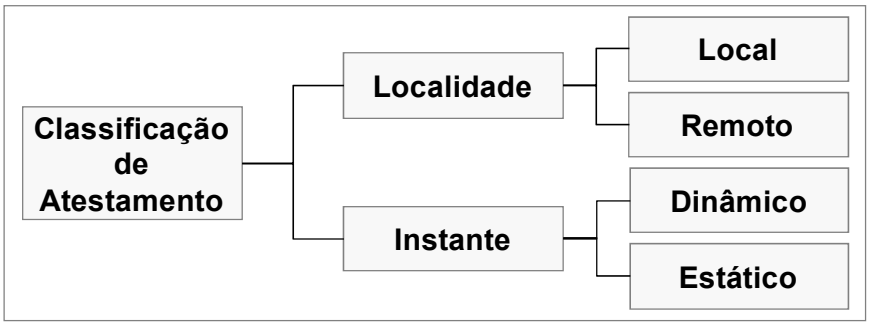

Figura 2.10 - Tipos de atestamento (MITTELSDORF, 2004)

Neste trabalho, considerando o contexto de aplicações SOA e o objetivo de atestar os consumidores dos serviços de um determinado provedor, são explorados atestamentos remotos, pelo critério de localidade, e dinâmicos pelo critério de instante de execução. A classificação como atestamentos remotos é compreendida pela própria natureza distribuída de SOA com o consumo a distância dos serviços disponibilizados em rede. Quanto à classificação como atestamentos dinâmicos, esta é discutida ao longo da Seção 4.5 quando são detalhadas as quatro etapas de atestamento definidas pelo arcabouço.

3 SEGURANÇA EM APLICAÇÕES *WEB*

3.1. Introdução

Nesta seção são apresentados inicialmente os atores de uma arquitetura *Web* tradicional, tipos de ataque e técnicas de proteção. A partir dessas definições são também apresentados aspectos de arquitetura, ataques comuns e técnicas de proteção para arquiteturas *Web* SOA.

As aplicações *Web* SOA somam às vulnerabilidades típicas das aplicações *Web* outras possibilidades de ataques derivadas de características intrínsecas das arquiteturas SOA.

Nesse cenário de riscos à segurança de provedores de serviços SOA, surge a necessidade de uma abordagem estruturada e efetiva para proteção contra usuários mal intencionados. No final desta seção, é então identificada a utilização de atestamento como um dos fundamentos para a segurança dos provedores perante os consumidores de serviços.

3.2. Arquitetura de uma aplicação *Web* tradicional

Para a análise das características de uma solução *Web* de interesse a este trabalho, é utilizada uma definição simplificada através da interação entre atores classificados em três grandes grupos:

- Atores do lado cliente
- Atores intermediários
- Atores do lado servidor

Pelo lado cliente, destaca-se o usuário (tipicamente um elemento humano) e o navegador (*Web browser*). Como atores intermediários, têm papel especialmente importante os elementos de rede como, por exemplo, *hubs*, roteadores, *switches* e servidores de nomes de domínio (*DNS: Domain Name Servers*). Os atores pelo lado

servidor, por sua vez, referem-se a servidores *Web*, servidores de aplicação e as aplicações propriamente ditas, incluindo bancos de dados e sistemas legados. A Figura 3.1 representa esta simplificação de uma arquitetura *Web*.

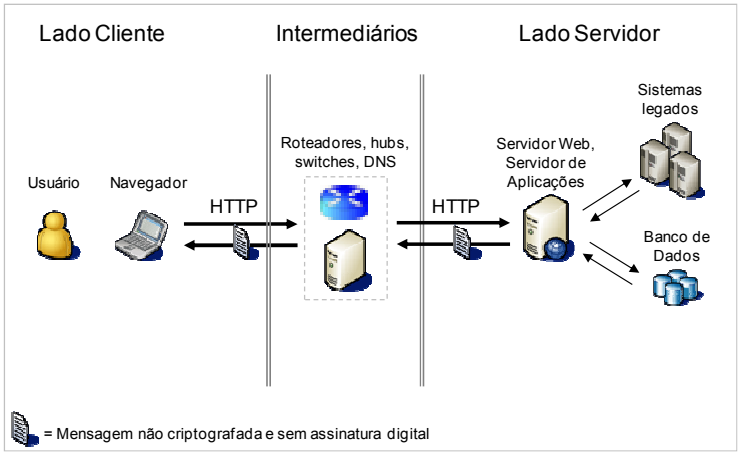

Figura 3.1 - Representação simplificada de uma arquitetura *Web*

Considerando-se os aspectos de segurança em uma aplicação *Web* tradicional, os seguintes elementos de segurança são tipicamente adicionados:

- Autenticação: tipicamente implementada através de uma abordagem de fornecimento, via tela da aplicação no navegador, dos dados de identificação do usuário e senha para execução do processo de *login*;
- Autorização: associação das funções autorizadas na aplicação para cada perfil de usuário que se autentique pelo processo de *login*;
- *Firewall* e *Intrusion Detection System* (IDS): impedimento do acesso a redes privadas (*firewall*) e detecção de comportamentos suspeitos de ataque (*IDS*);
- Protocolo SSL: implementação de PKI (*Public Key Infrastructure*) provendo destacadamente os princípios de confidencialidade e integridade dos dados trafegados, bem como a certificação digital (tipicamente apenas do servidor) através de autoridade certificadora.

Os quatro elementos de segurança acima não são exaustivos e tem por objetivo apenas representar simplificadamente os aspectos de interesse especial a este trabalho. A Figura 3.2 representa uma arquitetura *Web* tradicional, agregando-se os aspectos de segurança mencionados.

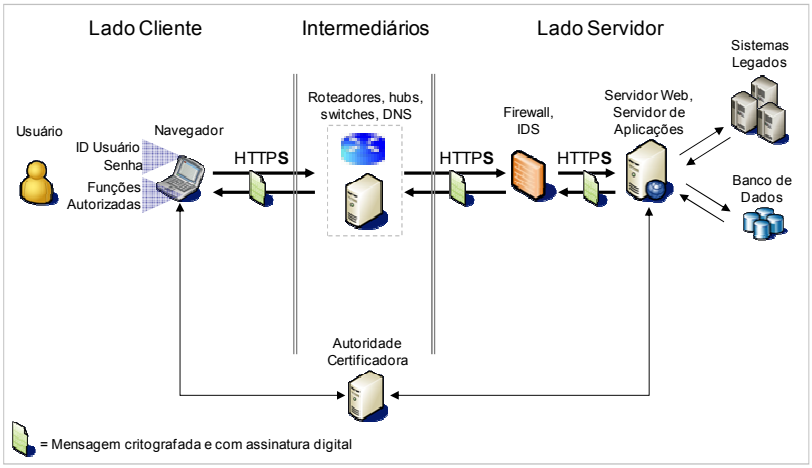

Figura 3.2 - Uma arquitetura *Web* com elementos de segurança

3.3. Tipos de ataque e técnicas de proteção em aplicações *Web* tradicionais

Primeiramente, é importante referir que o conjunto de tipos de ataque a soluções *Web* é demasiadamente extenso para ser explorado em sua totalidade neste trabalho. Assim, para os propósitos desta pesquisa é apresentado um subconjunto dos tipos de ataque e respectivas técnicas de proteção mais comumente utilizadas.

Segundo Wongtschowski (2005), os tipos de ataque em aplicações *Web* podem ser classificados em:

- <u>Roubo de dados</u>

  Ataque tipicamente aplicado para roubar dados críticos a partir do conhecimento ilícito das informações de autenticação. Também conhecido como ataque de personificação, é realizado a partir de mecanismos como a captura da digitação no teclado e o fornecimento das informações de autenticação através de telas e *e-mails* falsos (*phishings*[1]). Este tipo de ataque teve presença especialmente importante sobre as primeiras aplicações *Web* que faziam uso indiscriminado do recurso de *browser cookies.* O uso inadequado de *cookies* resultava no armazenamento de

[1] *Phishing* corresponde a um tipo de ataque baseado em engenharia social para obtenção de informações críticas como credenciais, números de cartão de crédito e informações de contas bancárias através de *e-mails* e *links* a endereços falsos das organizações de confiança dos clientes.

informações de autenticação em arquivos não protegidos na estação dos usuários.

Para a proteção contra o roubo de dados, os métodos mais elementares consistem na própria proteção das informações de identificação e senha, sem as quais o atacante não poderá reproduzir o processo de autenticação com o objetivo de se passar como usuário legítimo com acesso aos dados críticos da aplicação. Como exemplo destes métodos, pode-se citar o armazenamento seguro e criptografado das informações de autenticação e a utilização de OTP (*One Time Password*), que consiste na modificação da senha a cada operação de autenticação.

- Roubo de sessão

Em resposta à proteção de sistemas que requerem senhas diferentes a cada autenticação, surgiu uma técnica de ataque que se baseia no roubo de uma sessão ativa após o usuário ter realizado sua autenticação. De posse da sessão, o atacante é capaz de se passar pelo usuário legítimo e realizar transações *online*. Como técnica de proteção, pode-se destacar a tecnologia de assinatura digital que, ao requerer uma chave privada de conhecimento exclusivo do usuário legítimo, evita que um usuário ilícito execute transações sobre a sessão que tenha roubado.

- *Man-in-the-middle*

Os ataques *man-in-the-middle* se baseiam na interceptação dos dados durante seu tráfego entre cliente e servidor, podendo o atacante modificá-los, impactando o resultado das transações. Obviamente, este tipo de ataque pode se tornar mais grave quando realizado sobre comunicações não criptografadas, podendo o atacante manipular as informações em benefício próprio através, por exemplo, da modificação da conta destino para crédito em uma dada transação. Considerando comunicações criptografadas, o atacante ainda assim pode intervir na integridade do sistema, através da modificação cega dos dados ou simplesmente através da repetição das transações enviadas com o objetivo de provocar um estado de negação de serviços por parte do servidor.

Entre as técnicas para proteção contra ataques de *man-in-the-middle* pode-se citar novamente a assinatura digital, especialmente em função da validação de integridade por meio da checagem dos resultados de uma função *hash* sobre os dados a serem protegidos contra alterações em trânsito.

Sob uma outra perspectiva de estudo, a OWASP (*Open Web Application Security Project*) (OWASP, 2007) destaca um conjunto de dez vulnerabilidades mais comuns em aplicações *Web*, descritas na Tabela 3.1. Nessa tabela, na primeira coluna são apresentados os nomes das vulnerabilidades ou tipos de ataque e na segunda coluna a sua respectiva definição. Em especial, as vulnerabilidades enumeradas de um a oito merecem destacada atenção uma vez que não podem ser evitadas por uma camada de segurança periférica que implemente os princípios tradicionais como identificação, autenticação, autorização, entre outros. Trata-se de vulnerabilidades que são alvos de ataque que atravessam esta segurança ao redor das aplicações e exploram o desenho e programação inadequados realizados pela equipe de desenvolvimento.

Tabela 3.1 - Dez vulnerabilidades de segurança mais freqüentes em aplicações *Web* (OWASP, 2007)

Vulnerabilidade	Definição
1. Cross Site Scripting (XSS)	As falhas XSS permitem aos atacantes executarem *scripts* no navegador da vítima, os quais podem roubar sessões de usuário, "pichar" sites *Web*, introduzir *worms*, etc.
2. Falhas de Injeção	A injeção (com destaque a *SQL Injection*) ocorre quando os dados fornecidos pelo usuário são enviados a um interpretador como parte de um comando, visando a execução de instruções que revelem dados críticos ou simplesmente interfiram na integridade das informações e transações
3. Vazamento de Informações e Tratamento de Erros Inapropriado	Exposição das configurações, processos internos ou qualquer violação de privacidade por meio da exposição indevida de informações a partir de situações de erro de execução das aplicações. Os atacantes podem usar essa fragilidade para roubar informações consideradas sensíveis
4. Execução Maliciosa de Arquivo	As aplicações suscetíveis a esta vulnerabilidade permitem ao atacante incluir código e dados maliciosos nos servidores. Pode afetar qualquer aplicação que aceite o *upload* de arquivos pelos usuários ou simplesmente a entrada de nomes de arquivos a serem executados nos servidores

(continua)

(continuação)

Vulnerabilidade	Definição
5. Referência Insegura Direta a Objeto	Exposição da referência a um objeto interno, como arquivos, diretórios, registros ou chaves da base de dados, na forma de uma URL ou parâmetro de formulário. Os atacantes manipulam estas referências para acessar outros objetos sem autorização
6. Falha ao Restringir Acesso a URLs	Frequentemente, uma aplicação protege suas funcionalidades críticas somente pela supressão de informações como *links* ou URLs aos usuários não autorizados. Os atacantes podem fazer uso desta fragilidade para realizar operações não autorizadas por meio do acesso direto a estas URLs
7. Cross Site Request Forgery (CSRF)	Para seu funcionamento, a vítima deve acessar uma aplicação *Web* sob domínio do atacante e também estar autenticada na aplicação *Web* alvo do ataque. As páginas da aplicação sob domínio do atacante tentarão forçar a invocação de URLs da aplicação alvo com o objetivo de executar comandos determinados pelo atacante
8. Falhas de Autenticação e Gerência de Sessão	Falhas resultantes da proteção inadequada de credenciais de acesso e *tokens* de sessão. Nestes casos, os atacantes comprometem senhas, chaves ou *tokens* de autenticação de forma a assumir a identidade de outros usuários
9. Armazenamento Criptográfico Inseguro	Utilização de funções criptográficas de forma inadequada para proteção de informações e credenciais. Os atacantes se aproveitam de informações mal protegidas para realizar roubo de identidade e outros crimes
10. Comunicações Inseguras	Falhas por não criptografar tráfego de rede quando se faz necessário proteger comunicações confidenciais

Em seu estudo, a OWASP apresenta também um conjunto de técnicas de proteção contra os tipos de ataques apresentados na Tabela 3.1. Estas técnicas de proteção estão descritas no Apêndice A.

3.4. Arquitetura de uma aplicação *Web* SOA

Os fundamentos de uma arquitetura SOA em plataforma *Web* inclui todos os elementos de uma aplicação *Web* tradicional. As diferenças são definidas por elementos adicionais de protocolo (exemplo SOAP – *Simple Object Access Protocol*), diretórios UDDI (*Universal Description, Discovery and Integration*), barramentos ESB (*Enterprise Service Bus*) e provedores de serviços de distribuídos

(*Web services*). Com o objetivo de simplificação dos fundamentos de interesse a este trabalho, não são apresentados nesta seção os elementos UDDI e ESB.

A Figura 3.3 representa uma arquitetura *Web* estendida com os elementos adicionais de SOA.

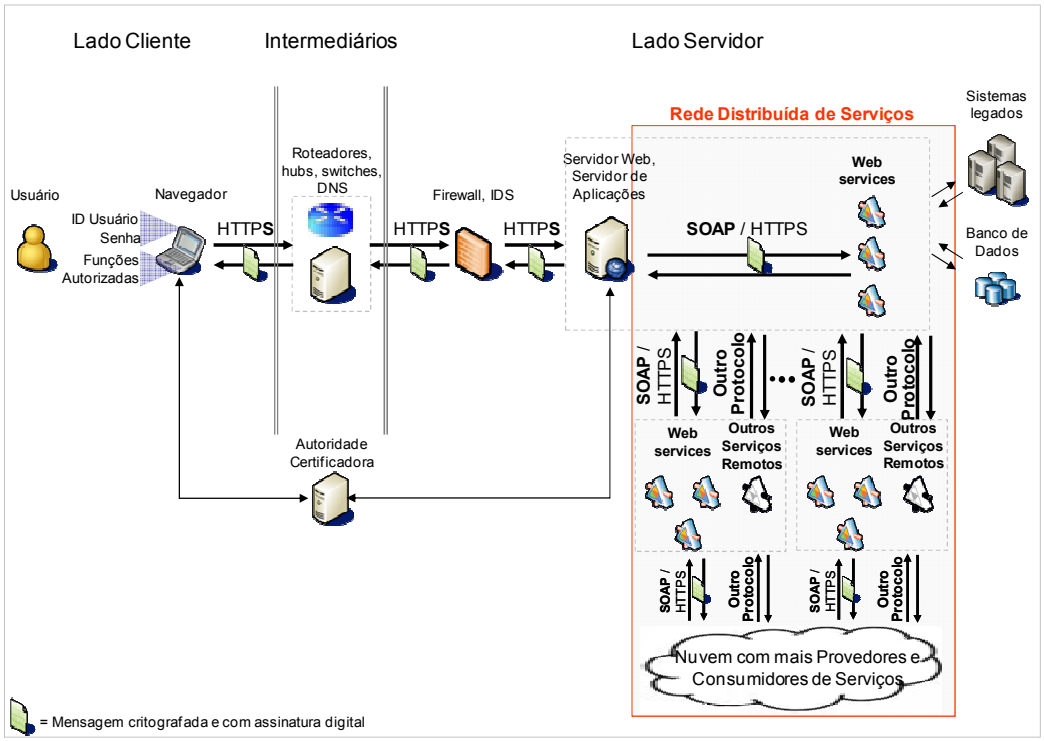

Figura 3.3 - Arquitetura de uma aplicação *Web* orientada a serviços

Como representado na Figura 3.3, em soluções SOA é comum a utilização de uma rede distribuída de serviços no lado servidor, conectando serviços da própria organização ou serviços externos, como seria o caso em B2B (*Business to Business*). Neste cenário, é importante considerar que um provedor de serviço pode também atuar como consumidor de outro serviço gerando conexões de dependência para a entrega da resposta à solicitação original.

Importa observar, embora não representado na Figura 3.3, que é possível o cenário onde *Web services* também compõem a aplicação no lado cliente, resultando em uma rede distribuída de serviços também pelo lado cliente.

3.5. Tipos de ataque e técnicas de proteção em aplicações *Web* SOA

Deve-se observar que todos os tipos de ataque e técnicas de proteção apresentados na Seção 3.3 para aplicações *Web tradicionais* são também aplicáveis a soluções *Web* SOA. Adicionalmente, há tipos ataques aplicáveis especificamente a soluções SOA, sendo que neste trabalho é estudado um conjunto dos tipos mais comuns, sem a pretensão de descrever de modo exaustivo todos os ataques e técnicas de proteção existentes.

A seguir, um resumo dos tipos mais conhecidos de ataques e técnicas de proteção em soluções *Web* SOA:

- Ataques sobre UDDI

  Segundo Sidharth e Liu (2007), consistem basicamente na exploração de informações disponibilizadas nos repositórios de registro de serviços ou UBR (*Universal Business Registry*), incluindo informações como nome do serviço, operações oferecidas, detalhes do WSDL e localização do serviço.

  Shah (2002) descreve um método para obter informações de um *Web service* através de técnicas de investigação (*footprinting*) sobre UBRs. O método recorre às APIs padrão de pesquisa do protocolo UDDI e que operam sobre HTTP/HTTPS com SOAP. Estas APIs de pesquisa permitem a consulta a detalhes de um *Web service*, como é o caso da operação *get_serviceDetail* que revela, por exemplo, a URL de um *Web service*. A partir da URL, tem-se então um ponto de acesso para investigação do WSDL, o que pode ser utilizado para fins não legítimos de exploração e ataques sobre o serviço.

  Shah et al. (2009) apresentam a possibilidade de invasão em um repositório UDDI através de ataques que resultem no aumento dos privilégios de acesso pelo usuário fraudador. Através de acesso privilegiado, o fraudador pode executar atualizações maliciosas como alterações das políticas de segurança de um serviço ou mesmo o redirecionamento das requisições para *Web services* forjados.

  Quanto às técnicas de proteção, as principais estão fundamentadas no controle da visibilidade dos registros em um repositório UBR. Este controle

pode ser implementado tanto no nível individual de cada registro, definindo quais usuários terão visibilidade sobre o mesmo, bem como para o repositório como um todo.

A versão 3.0.2 do protocolo UDDI provê proteção contra ataques de investigação das informações do repositório (*footprinting*) ao requerer identificação, autenticação, autorização e até mesmo assinatura digital dos usuários antes da liberação de acesso aos registros.

- Ataques sobre WSDL

Sidharth e Liu (2007) discutem um tipo de ataque sobre WSDL em que, a partir da descrição das operações e seus parâmetros, são realizadas invocações com falhas intencionais que, ao provocarem erros de execução, podem revelar informações sobre a base de dados da aplicação.

Como outro exemplo de ataque, a partir do conhecimento do WSDL, este pode ser copiado para criação de um falso *Web service*. Esse *Web service* pode então ser publicado em algum UBR e ser invocado equivocadamente pelos consumidores que poderão passar informações (em forma de parâmetros das operações) de interesse a um destinatário fraudador.

Segundo Sidharth e Liu (2007) e Singhal; Winograd e Scarfone (2007), um outro tipo de ataque sobre WSDL pode ser originado a partir de comentários ou mesmo de métodos de depuração que podem ser deixados inadvertidamente em arquivos WSDL. Essas informações podem ser utilizadas para estreitar a busca de vulnerabilidades na implementação do *Web service*.

Uma técnica importante para dificultar essas tentativas de ataque por exploração é garantir a remoção de quaisquer informações desnecessárias dos arquivos WSDL. Comentários internos e úteis apenas à equipe de desenvolvimento devem ser removidos. Analogamente, deve-se evitar a publicação desnecessária de métodos de depuração, os quais podem contribuir para a revelação de informações sobre a tecnologia de implementação, plataformas de banco de dados acessadas e outras informações de interesse que podem ser conjugadas para estruturar um ataque.

Ainda como técnica de proteção, o próprio controle de acesso aos registros UDDI indiretamente já protege contra ataques de exploração das informações contidas nos arquivos WSDL. Alternativamente, pode-se também recorrer até mesmo à criptografia de elementos privados de um arquivo de definições WSDL.

- Ataques sobre mensagens SOAP

Segundo Sidharth e Liu (2007) e Singhal; Winograd e Scarfone (2007), os ataques sobre mensagens SOAP consistem em:

- *Alteração de parâmetros*

Este ataque pode provocar falhas que revelem o tipo de base de dados acessada pelo serviço (como, por exemplo, uma base relacional ou um servidor LDAP). Também através de alteração maliciosa de parâmetros é possível muitas vezes provocar erros em transações SQL para tentar revelar informações da estrutura do banco e do dicionário de dados. Todas estas informações podem então ser utilizadas estrategicamente para formular ataques com base no conhecimento das estruturas acessadas pelos serviços.

Como referido na Seção 3.3, este tipo de ataque explora a vulnerabilidade denominada falha de injeção. A Figura 3.4 ilustra um exemplo de exposição indevida de informações em conseqüência de uma falha de injeção SQL executada por um usuário mal intencionado. Nesse exemplo, informações como nomes de tabelas (“p_version e p_namespace”) e colunas (“id, namespaceId”) são indevidamente expostas aos usuários finais da aplicação através da publicação não tratada de um *stack trace*[1] (vide trechos indicados em vermelho na Figura 3.4). A partir dessas informações, um usuário mal intencionado é capaz de começar a entender a estrutura do banco de dados acessado pela aplicação e então utilizar este conhecimento na elaboração de estratégias de ataque (como tentativas de consulta, alteração ou mesmo inserção sobre registros deste banco de dados).

[1] *Stack trace* é um relatório tipicamente utilizado para depuração de erros de execução em um programa de computador, incluindo a indicação das partes do programa onde ocorreu o erro (é um recurso suportado nativamente por muitas linguagens como, por exemplo, Java e C#).

```
org.apache.ojb.broker.PersistenceBrokerSQLException: * Can't prepare statement: * sql
statement was 'SELECT id FROM p_version A0 INNER JOIN p_namespace A1 ON
A0.namespaceId=A1.id WHERE (id >= ?) AND namespace = ?'
* Exception message is [Column name 'ID' is in more than one table in the FROM list.]
* Vendor error code [30000]
* SQL state code [42X03]
        at
org.apache.ojb.broker.util.ExceptionHelper.generateException(ExceptionHelper.java:256
```

Figura 3.4 - *Stack trace* com vazamento de informações

Uma variação do ataque de alteração de parâmetros é conhecida como ataque de reescrita de XML e foi definida por Rahaman; Schaad e Rits (2006). O ataque de reescrita de XML pressupõe, ao contrário da técnica de simples alteração de parâmetros, que a mensagem SOAP seja interceptada para que então seja manipulada e retransmitida. Trata-se, portanto, de uma técnica conjugada com uma intervenção do tipo *man-in-the-middle*. Como resultado, um ataque de reescrita pode provocar falhas na execução de uma solicitação originalmente legítima e até mesmo a modificação de valores informados como entrada às operações publicadas por um serviço, com prejuízo ao consumidor e ou provedor do serviço.

- *Repetição de requisição (replay attack)*

Os ataques de repetição consistem em provocar repetidas mensagens SOAP com o objetivo de gerar a sobrecarga de processamento de um serviço. Este tipo de ataque muitas vezes não é detectado por *firewalls* e sistemas de detecção de intrusão uma vez que o endereço IP de origem é valido, o comportamento sob o ponto de vista do pacote de rede também é válido e a solicitação HTTP é bem formada. Muitas vezes são ataques provocados através de intervenções de *man-in-the-middle* e, no limite, podem resultar à negação de serviço (*denial of service*). Como definido por Cabrera e Kurt (2005), a negação de serviços consiste no comprometimento da capacidade de resposta de um servidor em conseqüência da sobrecarga de processamentos intensivos disparados pela repetição maliciosa de requisições.

- *Tentativas forçadas de descoberta de Web services não publicados*

  Estes ataques consistem basicamente em repetidas requisições com pequenas alterações no padrão de composição das URLs. Alternativamente, este tipo de ataque pode buscar por arquivos WSDL que serão então indicativos da existência de *Web services* não publicados.

- *Exploração de overflow*

  A exploração de *overflow* é tipicamente aplicada para implementações de *Web services* em linguagens como C e C++ que confiam ao programador o controle da alocação dinâmica de memória. Através de *overflow*, o ataque pode provocar até mesmo a execução de programas maliciosos que passem a ocupar espaços de memória reservados para alocar códigos executáveis. Há também outros ataques mais simples que consistem basicamente em provocar panes na execução dos serviços, resultando potencialmente em situações de negação de serviço.

- *Envenenamento de XML Schema (XML Schema Poisoning)*

  O ataque consiste em tentar modificar os XML *schemas* que provêem instruções de pré-processamento aos interpretadores de documentos XML. Através de modificações nos *schemas*, o atacante pode provocar a rejeição de documentos XML válidos ou até mesmo causar que documentos XML maliciosos sejam aceitos pela aplicação.

- *Aumento malicioso de arquivos XML*

  Este tipo de ataque consiste em gerar documentos XML excessivamente grandes para o processamento pelo interpretador (*parser*) no provedor de serviço. Este ataque pode provocar situações de negação de serviço (*denial of service*), especialmente nas implementações em que o interpretador requer a carga em memória de todo o arquivo XML a ser processado.

Uma técnica importante para proteger contra os ataques sobre mensagens SOAP consiste em implementar mecanismos que evitem a

intrusão de parâmetros maliciosos no corpo das mensagens SOA a serem processadas pelos provedores de serviços. O objetivo é evitar o processamento de entradas que provoquem as chamadas falhas de injeção, como injeção de SQL e de instruções LDAP. Singhal; Winograd e Scarfone (2007) referem um mecanismo de proteção a partir da delimitação de tipos e formatos dos parâmetros nos arquivos WSDL. A cada requisição recebida pelo *Web service* é verificado se o tipo e formato corretos foram enviados pelo consumidor, evitando-se submeter ao processamento do serviço parâmetros não aderentes a estas especificações.

Como exemplos de implementação de mercado para técnicas de proteção contra a intrusão de parâmetros maliciosos, pode-se citar interfaces do arcabouço .NET da Microsoft e também funcionalidades nativas do servidor *Web* Apache. No arcabouço .NET as interfaces *IHTTPHandler* e *IHTTPModule* provêem uma funcionalidade de checagem das requisições HTTP antes que os parâmetros de entrada sejam repassados ao código da aplicação. Analogamente, nos servidores *Web* Apache é possível configurar expressões regulares que definem padrões de texto que deverão ser rejeitados (como, por exemplo, "*%*" e "*like*" que podem representar ataques de injeção de SQL), evitando o seu processamento pelos *Web services* (SIDHARTH; LIU, 2007).

Para a proteção contra ataques de reescrita de XML, Rahaman; Schaad e Rits (2006) propõem uma técnica baseada na inserção de uma nova *tag* de cabeçalho em um envelope SOAP, denominada SOAPAccount. Esta nova *tag* tem a função de descrever informações a respeito da estrutura global do envelope SOAP como, por exemplo, número total de elementos filho do envelope e número de elementos de cabeçalho. A proteção é viabilizada através de assinatura digital dessa *tag* SOAPAccount, visando garantir integridade de seu conteúdo contra alterações em trânsito e evitando que a própria *tag* seja alvo do ataque de reescrita de XML. Pelo lado do provedor de serviços, uma vez recebida a mensagem, este dispara o processo de decriptografia e validação de integridade sobre a *tag* SOAPAccount. Uma vez identificado que a *tag* SOAPAccount está íntegra, o servidor checa então se o envelope SOAP está preservado tal como descrito nas informações

estruturais (número de elementos filho, número de elementos de cabeçalho, etc.) contidas na *tag* SOAPAccount.

Quanto à proteção contra os ataques de repetição de requisições, Faw (2004) apresenta a técnica de implantação de intervalos curtos de autorização pelo provedor para o processamento de requisições dos consumidores. Por esta técnica, as autorizações deferidas, por exemplo, através de mensagens SAML (*Security Assertions Markup Language*), deveriam expirar no menor intervalo de tempo possível, o qual é especificado como parâmetro na própria mensagem de retorno SAML. Sob estas condições, um ataque por interceptação da mensagem com o objetivo de forçar repetição de requisições fracassará tão logo expire o intervalo de autorização concedido pelo provedor de serviços. Alternativamente, outra técnica simples contra os ataques de repetição consiste em implementar uma lógica de seqüenciadores comandada pelo provedor. Por esta técnica, a cada interação o consumidor deverá prover um seqüenciador de conhecimento exclusivo entre o provedor (responsável por gerar o seqüenciador) e consumidor. Sob este controle, requisições com seqüenciador repetido não serão processadas, sendo também importante definir que em casos de perda da sincronização, a reinicialização deverá sempre ocorrer a partir de valor determinado pelo provedor.

Para proteção contra ataques de exploração de *overflow*, pode-se citar como técnica importante o uso da programação segura contra falhas na codificação de alocação de memória (SINGHAL; WINOGRAD; SCARFONE, 2007). Esta programação segura pode ser viabilizada através de linguagens que automaticamente evitam o estouro na alocação dinâmica de memória ou ainda através do uso de compiladores e bibliotecas que acusam e impedem erros de codificação na alocação de memória em linguagens suscetíveis a esse tipo de falha, como C e C++.

Shah e Patel (2009) e Sidharth e Liu (2007) descrevem a implementação de *firewalls* de *Web services* como uma técnica para proteção contra diversos tipos de ataques e que é independente da plataforma e dos *softwares* básicos utilizados para os servidores *Web* e servidores de aplicação. Os *firewalls* de *Web services* consistem em programas tratadores que interceptam as

requisições SOAP e atuam nos processamentos de segurança antes da entrega das mensagens aos provedores e consumidores de serviços. Os programas tratadores podem então ser compreendidos como intermediários entre consumidores e provedores de serviços, podendo processar validações dos parâmetros das operações, checagem de aderência a *XML schemas* e às políticas de segurança, bem como checagens de autenticidade, integridade e outras validações contra diversos tipos de ataques. A proposição de uso de programas tratadores é retomada na Seção 4.4, na qual é descrito seu papel na implementação de funções de segurança definidas pelo arcabouço e na intermediação das chamadas às operações de negócio.

3.6. Atestamento como um dos fundamentos à segurança em *Web* SOA

Identifica-se, pelo exposto na Seção 3.5, que é extenso o conjunto de possíveis ataques a aplicações *Web* SOA, uma vez que se somam aos ataques tradicionais a aplicações *Web* outros tipos de intervenções maliciosas baseadas em vulnerabilidades de segurança intrínsecas à arquitetura SOA. Nesse contexto de vulnerabilidades e em resposta à característica das soluções SOA de disponibilizar os serviços a uma quantidade de consumidores muito maior que nas arquiteturas tradicionais, a abordagem de atestamento surge como uma proposta promissora para a segurança dos provedores perante os consumidores de serviços.

Considerando-se as aplicações *Web SOA* de segurança crítica, é recomendação deste trabalho a utilização de atestamento para comprovação da legitimidade dos vários componentes de *software* dos consumidores de serviços. O atestamento realiza o papel essencial de validar os clientes como consumidores legítimos, reduzindo a probabilidade de incidentes provocados pela extensa lista de técnicas de ataque discutidas na Seção 3.5.

Através do arcabouço definido, apenas consumidores atestados tem suas requisições processadas e, mesmo que haja falhas por programação inadequada dos serviços, o atestamento reduz a probabilidade de invasão e aumenta o esforço dos ataques.

4 ARCABOUÇO DE ATESTAMENTO PARA APLICAÇÕES *WEB* SOA

4.1. Introdução

Para aumentar a eficácia da segurança em uma rede de provedores de serviços se faz necessário validar a legitimidade de todos os elementos humanos, físicos e lógicos (usuários, *hardware* e *software*) que participam da cadeia de acesso desde o provedor até o consumidor final dos serviços oferecidos. Especial atenção deve ser dada ao lado consumidor. Este é o lado mais vulnerável da cadeia e com menor controle do provedor de serviço sobre os recursos críticos envolvidos nas interações entre consumidores e provedores. O controle do lado consumidor, em geral, fica sob responsabilidade do próprio usuário consumidor que nem sempre está habilitado a exercer tão importante função de garantia de segurança nas interações com os provedores de serviços. Na maioria das vezes os usuários deste tipo de sistema estão sujeitos a ataques de engenharia social, sejam eles explícitos ou dissimulados, podendo expor inadvertidamente informações sensíveis que podem levar a situações de fraude nos serviços.

Neste cenário de vulnerabilidade e de múltiplos tipos de ataques, conforme descritos na Seção 3.5, o atestamento dos consumidores se estabelece como uma abordagem promissora para a segurança em aplicações *Web* SOA.

Este trabalho se propõe a especificar um arcabouço, que define um conjunto de etapas de atestamento dos consumidores para a implementação de soluções *Web* SOA seguras.

A principal contribuição do arcabouço proposto, denominado ASACS (Arcabouço para Segurança por Atestamento dos Consumidores de Serviços), é a definição de uma abordagem estruturada para a validação de legitimidade dos consumidores, resultando na necessidade de um maior esforço para os ataques aos provedores de serviços.

As quatro etapas de atestamento definidas para o ASACS estão detalhadas na Seção 4.5 e podem ser agrupadas em:

- Etapas de validação de legitimidade: etapas responsáveis pela verificação de legitimidade e autenticidade dos envolvidos na cadeia de acesso a uma rede de serviços;
- Etapa de validação de comportamento: etapa de atestamento baseada na análise de comportamento dos consumidores através de um mecanismo de avaliação e métricas para graduar a confiança atribuída a cada consumidor em um dado instante.

Para apoiar a implementação destas etapas de atestamento do ASACS, é também especificada neste trabalho uma biblioteca de suporte. Esta biblioteca provê a implementação de classes de objetos para atender a um conjunto de funções de suporte que incluem interfaces com um dispositivo de *hardware* no lado consumidor, interfaces com um agente de regulação do grau de confiança dos consumidores, validações de sincronização de chave simétrica de sessão, processamentos de criptografia e outras funcionalidades requeridas à operação do ASACS. A implementação desta biblioteca, bem como de uma instância do ASACS, não é escopo do presente trabalho e deverá ser objeto de trabalhos futuros.

A Figura 4.1 representa o arcabouço proposto, composto de quatro etapas para atestamento dos consumidores de serviços e uma biblioteca de suporte.

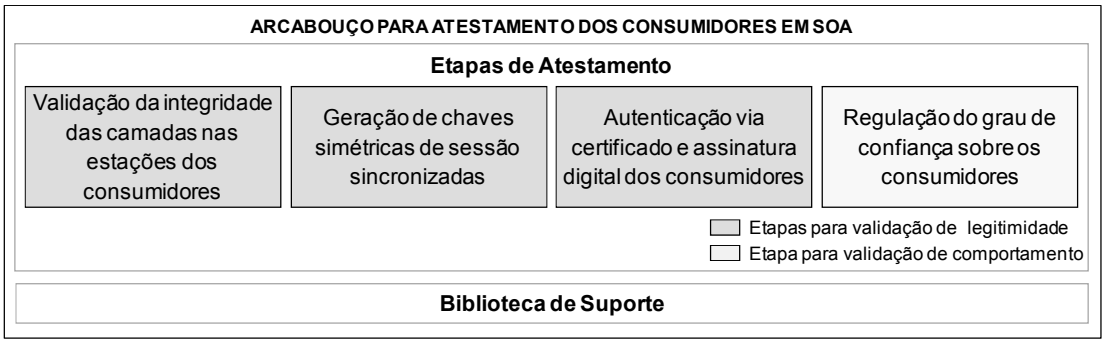

Figura 4.1 - Etapas do arcabouço de atestamento ASACS

Quanto ao modo de segurança definido para o ASACS, a opção recomendada é o modo de segurança fim-a-fim, também denominado segurança no nível da mensagem. A opção pela segurança no nível da mensagem é sustentada pela motivação de propor um arcabouço com foco na maior eficiência possível dos princípios de segurança em aplicações de segurança crítica. Utilizando segurança no nível da mensagem, as etapas de atestamento do arcabouço se beneficiam de um grau maior de segurança uma vez que as mensagens não são decriptografadas

nos agentes intermediários SOA (como, por exemplo, em servidores de aplicação no caminho da comunicação entre consumidor e provedor final). Adicionalmente, a segurança no nível da mensagem garante mais flexibilidade, ao permitir, por exemplo, a definição do que e como criptografar em cada mensagem trafegada.

O padrão WS-Security é o recomendado neste trabalho para a implementação da segurança no nível das mensagens SOAP em função de sua maturidade, grau de cobertura dos princípios de segurança e boa aceitação no mercado quando comparado a outros padrões para implementação de segurança em SOA.

4.2. Requisitos gerais do arcabouço de atestamento

A especificação do ASACS é detalhada ao longo deste trabalho e busca cumprir um conjunto de requisitos, a saber:

- Estruturar implementações de segurança com base em atestamento: foco na aplicação de atestamento dos consumidores como proposta central e fundamental para proteção contra consumidores não legítimos ou mal intencionados;
- Garantir possibilidade de atestamento também do ambiente de execução dos consumidores de serviços: trata-se da validação de legitimidade e integridade também das camadas de inicialização (da BIOS à camada de aplicação) nos computadores dos consumidores;
- Permitir flexibilidade no nível de segurança com implementação de apenas um subconjunto das etapas de atestamento: diante de restrições de custo e complexidade para implantação indiscriminada do arcabouço para a segurança de uma aplicação, a especificação do ASACS deve permitir a implantação seletiva de apenas parte do conjunto de etapas de atestamento do arcabouço;
- Possibilitar monitoração do comportamento de consumidores validados como legítimos: como um passo complementar à segurança da aplicação, o ASACS deve especificar uma etapa voltada à avaliação do comportamento de consumidores validados como legítimos, permitindo que cada serviço da rede

ou mesmo cada operação de um serviço decida pelo processamento ou negação das requisições destes consumidores;

- Viabilizar segregação entre as funcionalidades de segurança e regras de negócio: considerando-se a definição para uso do modo de segurança fim-a-fim (no nível da mensagem), a especificação do ASACS deve viabilizar a implementação estruturada e padronizada das funcionalidades de segurança na camada de aplicação e independente das funções de negócio;
- Prover uma biblioteca de suporte à implementação: a especificação do ASACS deve contemplar a definição de uma biblioteca com implementação de objetos que disponibilizem diversas funções requeridas à operação do arcabouço, detalhadas na Seção 4.8.

4.3. Diagrama de componentes para o arcabouço de atestamento

Para representar os principais componentes em uma instância de implantação do ASACS, nesta seção é definido e discutido um diagrama de componentes em linguagem de modelagem UML.

A Figura 4.2 representa esse diagrama de componentes onde se identifica três ambientes distintos que interagem dentro do arcabouço: ambiente do dispositivo de *hardware* independente, ambiente da estação do consumidor e ambiente da estação do provedor.

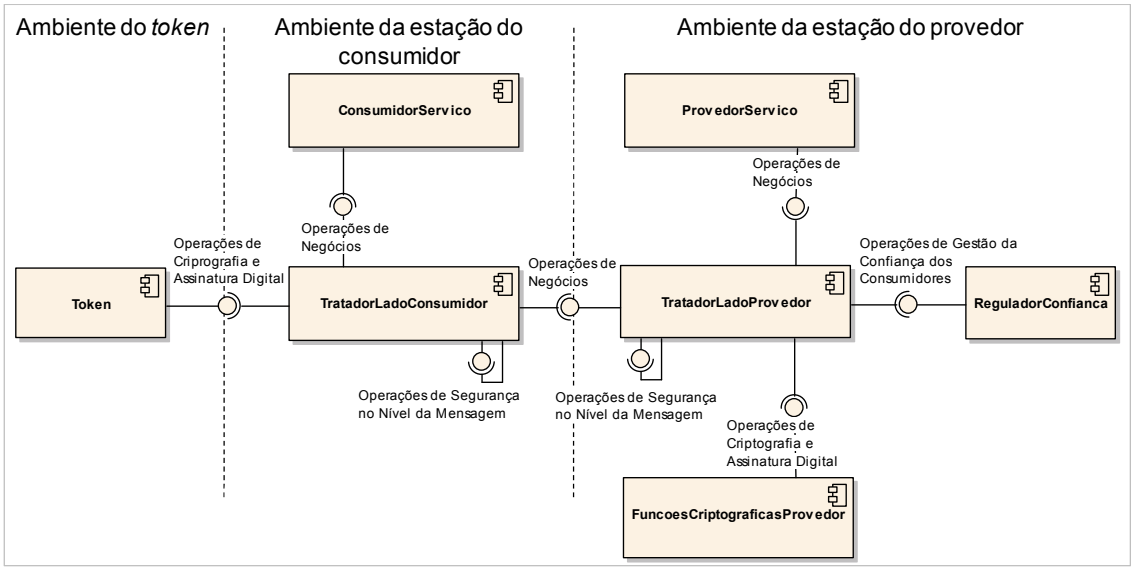

Figura 4.2 - Diagrama de componentes do ASACS

Os ambientes representados no diagrama de componentes da Figura 4.2 podem ser assim descritos:

- Ambiente do dispositivo de *hardware* independente: refere-se a um dispositivo de *hardware* acoplado à estação do consumidor e denominado simplesmente como *token* no contexto deste trabalho. Este *token* deverá conter um processador próprio e oferecer funções criptográficas invocadas nas estações dos consumidores. Os papéis definidos para o *token* são explorados ao longo da Seção 4.5 que detalha as etapas de atestamento definidas pelo arcabouço;
- Ambiente da estação do consumidor: plataforma de execução do computador do consumidor, onde são executadas as aplicações de consumo dos serviços e também os chamados programas tratadores. Os programas tratadores são detalhados na Seção 4.4 e podem ser definidos como intermediários SOAP com o papel de intermediar as comunicações entre consumidores e provedores e prover a implementação de funções requeridas pelas etapas de atestamento do ASACS;
- Ambiente da estação do provedor: plataforma de execução do computador do provedor, onde são disponibilizados os *Web services* da rede distribuída de serviços e também os programas tratadores no lado provedor. Pelo lado do provedor, os programas tratadores também exercem o papel de

intermediários SOAP, recepcionando e respondendo às mensagens enviadas pelos programas tratadores do lado consumidor.

Enquanto para os ambientes do *token* e da estação do provedor é perfeitamente possível garantir a integridade de seus componentes, o que facilita uma operação segura dos mesmos, para o ambiente da estação do consumidor se faz necessário um processo de atestamento mais elaborado. Afinal, o ambiente da estação do consumidor está sujeito a uma maior exposição aos riscos de fraude por depender fortemente do próprio usuário consumidor que nem sempre está habilitado a exercer a proteção requerida contras os mais diversos tipos de ataques. As etapas do ASACS visam exatamente propor esse processo de atestamento mais elaborado sobre as aplicações consumidoras através de validações da legitimidade e do comportamento destas aplicações na rede distribuída de serviços.

No diagrama de componentes da Figura 4.2, é importante apresentar a definição de cada componente representado em linguagem UML. A implementação destes componentes poderá ser viabilizada por uma ou mais classes e são assim definidos:

- <u>Ambiente do dispositivo de *hardware* independente</u>
  - *Token*: componente que oferece as APIs de interface com o dispositivo de *hardware* de segurança definido como parte da solução proposta pelo ASACS. Como será explorado na Seção 4.5, esse dispositivo deverá prover APIs para a execução de funções criptográficas (criptografia, decriptografia e assinatura digital) e também para a geração de chaves simétricas de sessão utilizadas para a criptografia das mensagens trocadas entre consumidores e provedores de serviços;
- <u>Ambiente da estação do consumidor</u>
  - *ConsumidorServico*: componente consumidor nas estações clientes que consome as operações publicadas por um provedor de serviços;
  - *ComponenteTratadorLadoConsumidor*: representa o conjunto de classes ou programas tratadores definidos na Seção 4.4 e que tem como principal objetivo expor ao componente consumidor as operações publicadas pelo provedor e também implementar as funções de segurança de modo segregado e independente das funcionalidades de negócios. Trata-se de um componente fundamental para a implementação da segurança no nível

da mensagem nas implantações do ASACS. É através do *ComponenteTratadorLadoConsumidor* que são invocadas as operações de negócio do provedor e também implementadas funções de segurança requeridas pelo ASACS, incluindo interfaces com o *token* de segurança e implementação da segurança no nível da mensagem com o padrão *WS-Security*.

- <u>Ambiente da estação do provedor</u>
  - *ProvedorServico*: componente provedor que publica as operações para uso pelos consumidores na rede distribuída de serviços;
  - *TratadorLadoProvedor*: representa o conjunto de classes ou programas tratadores pelo lado do provedor. Este componente tem o papel de intermediar as comunicações com o consumidor e prover implementações de segurança do ASACS como, por exemplo, validação da chave simétrica de sessão enviada pelo consumidor, decriptografia das mensagens e validação dos parâmetros de autenticação recebidos. É por meio deste componente que são publicadas as interfaces das operações de negócios a serem invocadas pelo componente de programas tratadores do lado consumidor (*TratadorLadoConsumidor*), o qual, por sua vez, repassa os retornos das operações ao componente que representa o consumidor de serviços (*ConsumidorServico*);
  - *FuncoesCriptograficasProvedor*: refere-se às classes que implementam pelo lado provedor as funções criptográficas que, diferentemente do lado consumidor, não são disponibilizadas através do *token* de segurança;
  - *ReguladorConfianca*: componente que representa as classes para atendimento ao requisito do ASACS de regulação do grau de confiança sobre cada consumidor na rede distribuída de serviços. Por meio das classes que implementam este componente, os provedores de serviço devem ser capazes de delatar comportamentos suspeitos dos consumidores e consultar o grau de confiança de um dado consumidor em um determinado instante.

Quanto às interfaces entre os componentes na Figura 4.2, foi utilizada a notação do conector “assembly” ( ) da especificação UML 2.0, visando representar a ponte

entre a interface requerida por um componente com a que é provida pelo outro componente com o qual interage.

4.4. Programas tratadores para intermediação das comunicações no arcabouço

A implementação do ASACS sob o modo de segurança fim-a-fim (segurança no nível de mensagem) resulta no desafio da implementação de segurança dentro das aplicações dado que, neste modo de segurança, a confidencialidade e outros princípios não são delegados à camada de transporte. Segundo Shah e Patel (2009), a abordagem de confiar aos códigos dos provedores (*Web services*) e consumidores a implementação da segurança no nível de mensagem introduz uma complexidade que pode tomar grandes proporções conforme a aplicação cresce em escala e torna-se altamente distribuída.

Para solucionar esse problema, é então proposta a abordagem de utilização de programas tratadores que podem ser definidos como componentes intermediários entre os consumidores e provedores de serviços.

Também conhecidos como interceptores de SOAP, os programas tratadores provêem um mecanismo para modificação das mensagens de requisição e resposta SOAP. Um exemplo é o uso de programas tratadores para a implementação de criptografia e decriptografia de dados no corpo de mensagens SOAP, utilizando-se o padrão WS-Security. Neste exemplo, o consumidor não invoca diretamente uma dada operação de negócio disponibilizada pelo provedor de serviço. Para invocar a operação do provedor de serviço, o consumidor utiliza-se de uma instância local de programa tratador que criptografa a mensagem antes do seu envio ao provedor. O provedor, por sua vez, confiará à sua respectiva instância local de programa tratador a responsabilidade de decriptografar a mensagem recebida para que só então seja processada pela lógica de negócio da aplicação. Trata-se de uma abordagem que visa segregar as funções de segurança, tornando-as independentes das funcionalidades de negócios.

A Figura 4.3 representa os principais papéis dos programas tratadores na comunicação entre consumidor e provedor de serviços em uma implementação do ASACS.

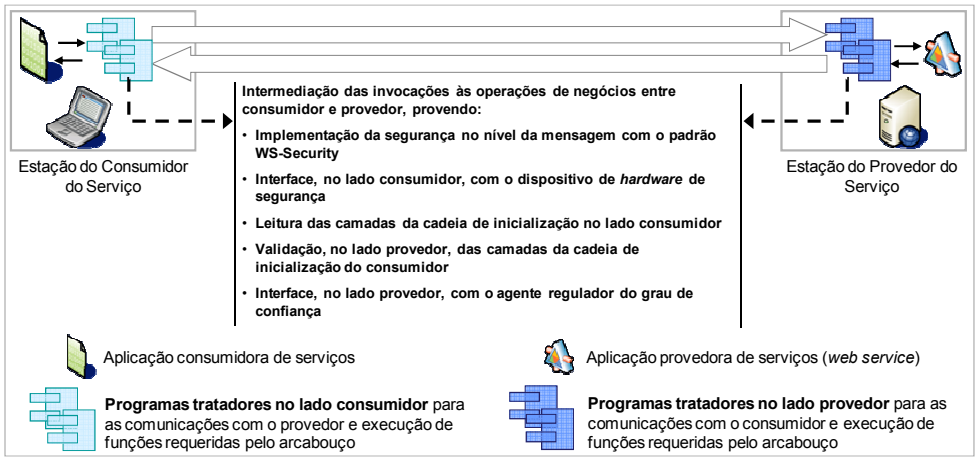

Figura 4.3 - Papéis dos programas tratadores no ASACS

Conforme representado na Figura 4.3, adicionalmente à intermediação das chamadas às operações dos serviços e modificação das mensagens para aplicar e interpretar a segurança no nível da mensagem com o padrão WS-Security, os programas tratadores requeridos à implementação do ASACS acumulam outras funções, a saber:

- Interface com um *token* de segurança no lado do consumidor. As interfaces com esse dispositivo são viabilizadas por APIs publicadas pelo *token* para a execução de funções criptográficas por um processador próprio deste dispositivo. Como será apresentado na Seção 4.5, as interfaces com esse *token* são requeridas às etapas de validação de legitimidade dos consumidores definidas pelo ASACS;
- Leitura das camadas da cadeia de inicialização no ambiente da estação do consumidor (da BIOS à camada de aplicações). A leitura e assinatura digital das camadas da cadeia de inicialização são partes da solução para a etapa de atestamento do ASACS dedicada à validação da integridade das camadas nas estações dos consumidores. O detalhamento desta etapa do ASACS para a validação da integridade e legitimidade das estações dos consumidores é apresentado na Seção 4.5.1;

- Validação, pelo lado provedor, da cadeia de camadas de inicialização informada pelo consumidor;
- Interface, no lado provedor, com um agente regulador do grau de confiança de cada consumidor em um dado momento na rede distribuída de serviços. A etapa do ASACS que descreve esta validação do comportamento dos consumidores com o objetivo de lhes atribuir um determinado grau de confiança está detalhada Seção 4.5.4.

Para a implementação das quatro funções descritas acima, os programas tratadores contam com uma biblioteca de suporte ao ASACS. Os principais requisitos e APIs desta biblioteca são apresentados na Seção 4.8.

4.5. Etapas de atestamento

As quatro etapas de atestamento do ASACS podem ser agrupadas em etapas de validação de legitimidade e de validação de comportamento. A seguir são apresentadas as descrições resumidas destas quatro etapas que serão detalhadas nas próximas seções:

- <u>Etapas de Validação de Legitimidade</u>
  - *Verificação da integridade das camadas nas estações dos consumidores*: refere-se à validação de integridade das camadas de *software* básico carregadas na estação cliente até o nível da aplicação consumidora de serviços;
  - *Geração de chaves simétricas de sessão sincronizadas*: etapa de atestamento que limita o conjunto de consumidores àqueles detentores de um dispositivo de *hardware* (*token* de segurança) para geração sincronizada de chaves simétricas válidas para as interações com um provedor de serviços;
  - *Autenticação via certificado e assinatura digital dos consumidores*: os certificados digitais possibilitam ao provedor de serviços a validação da autenticidade dos consumidores através da consulta a autoridades

certificadoras. As assinaturas, por sua vez, possibilitam validar que a mensagem foi enviada por um consumidor que de fato detém uma chave privada correspondente à chave pública informada (autenticidade e irretratabilidade) e também validar que a mensagem não foi adulterada em trânsito (integridade).

- Etapa de Validação de Comportamento
  - *Regulação do grau de confiança sobre os consumidores*: etapa de atestamento que visa aplicar uma graduação dinâmica de confiança aos consumidores a partir da análise de seu comportamento nas comunicações com os diversos provedores de uma rede distribuída de serviços.

4.5.1. Verificação da integridade das camadas nas estações dos consumidores

Nesta seção é apresentado um mecanismo que busca validar a integridade do consumidor de serviços através da validação de integridade não só do código do programa consumidor como também das camadas de *software* básico necessárias à execução deste programa. Para este objetivo, são discutidos:

- Conceitos da cadeia típica de inicialização de um computador;
- Um processo para assinatura das camadas dessa cadeia de inicialização;
- A utilização de um dispositivo de *hardware* (*token* de segurança) para as operações criptográficas e outras funções de segurança;
- Um protocolo de “desafio-resposta” para certificação dinâmica da integridade do código do programa consumidor e também das camadas de *software* básico carregadas para a execução desse programa.

Quanto à sua classificação pelo critério de instante de execução, trata-se de uma etapa de atestamento dinâmico. Esta classificação é devida à proposta de certificações periódicas de integridade como definido pelo protocolo de “desafio-resposta” detalhado na Seção 4.5.1.2.

Em decorrência das características de implementação apresentadas, esta etapa de atestamento pode ser classificada como uma das mais intrusivas entre todas as etapas definidas pelo arcabouço. Para o seu funcionamento são requeridas

intervenções importantes nas estações dos consumidores, o que inclui a utilização de um dispositivo de *hardware* e a atribuição de privilégios para o acesso a áreas de memória tipicamente protegidas, reservadas às camadas de inicialização das estações.

4.5.1.1. PROCESSO TÍPICO DE INICIALIZAÇÃO DE UM COMPUTADOR

Conforme apresentado por Mittelsdorf (2004), a seqüência típica de inicialização de um computador é composta pelas seguintes camadas:

1. BIOS
2. *Boot loader*
3. Sistema operacional
4. Aplicações

A BIOS é um *firmware* (programa embutido no *hardware*) que disponibiliza um conjunto de funções básicas que assumem o controle de execução no computador assim que o processador é inicializado. O *boot loader*, por sua vez, é invocado pela BIOS como seqüência da cadeia de inicialização e tem o papel de carregar em memória o sistema operacional. A partir deste ponto, o controle é passado para o sistema operacional, quando então é habilitada a camada de execução das aplicações.

É na camada de aplicações que residem os consumidores de serviços que, em conjunto com as camadas antecessoras da cadeia de inicialização, devem ser objeto de atestamento de integridade como parte do arcabouço proposto neste trabalho.

4.5.1.2. VALIDAÇÃO DAS CAMADAS DE INICIALIZAÇÃO

Mittelsdorf (2004) define que o atestamento remoto estático baseado em inicialização segura (*boot seguro*) tem como objetivo garantir ao provedor de serviços que a estação do consumidor foi inicializada de forma íntegra. Por inicialização íntegra, deve-se entender que as camadas da seqüência de *boot* não

foram alteradas, garantindo a cada *boot* que estas camadas estão preservadas tal como a sua instalação original.

O ASACS poderia incluir suporte para a realização do *boot* seguro inicial, porém esta característica foi deixada para trabalhos futuros por não introduzir novidade na implementação. É assumida a premissa que as estações dos consumidores sejam inicializadas de forma íntegra, não sendo explorado aqui o processo de *boot seguro* que visa a inicialização da estação apenas sob as verificações de integridade de cada camada. Um estudo sobre as abordagens para o *boot seguro* pode ser consultado em (MITTELSDORF, 2004).

Por outro lado, como parte do presente trabalho, é especificada uma solução que verifica se a integridade da estação do consumidor continua preservada após o *boot seguro* inicial. As camadas da cadeia de inicialização são verificadas periodicamente pelo arcabouço, garantindo que continuam íntegras conforme foram carregadas na ocasião do *boot* seguro.

Para implantar este atestamento da integridade da estação de um consumidor, é proposta a validação remota de uma cadeia de assinaturas sobre valores *hash* gerados para cada uma das camadas da seqüência de inicialização. As assinaturas destes valores *hash* devem ser realizadas com uma chave privada armazenada em um dispositivo de *hardware* acoplado às estações dos consumidores. Este mesmo dispositivo, denominado neste trabalho como *token*, deverá ser também utilizado no ASACS para o papel de geração de chaves simétricas de sessão para a criptografia das mensagens trafegadas entre consumidores e provedores de serviços. Os passos para o processo de validação dessa cadeia de assinaturas estão representados na Figura 4.4.

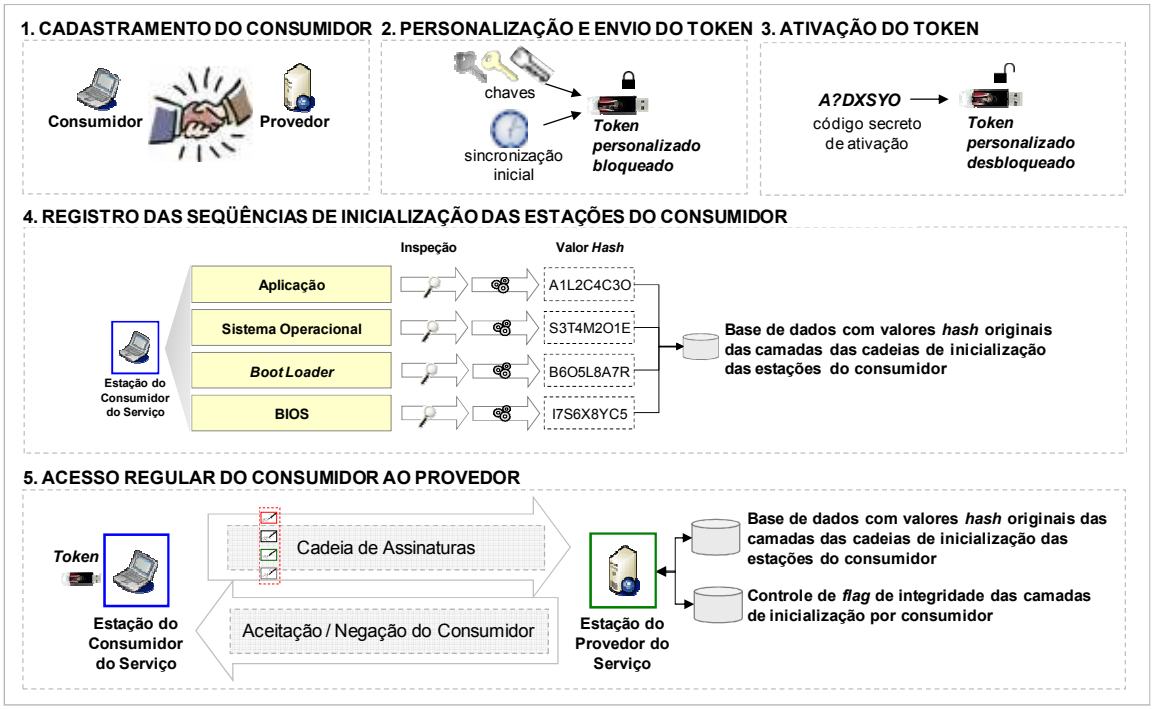

Figura 4.4 - Registro dos consumidores e verificação da integridade das camadas de inicialização

Os cinco passos representados na Figura 4.4 podem ser assim descritos:

- Passo 1: Cadastramento do consumidor

  Este passo consiste no fornecimento dos dados cadastrais iniciais suficientes para a identificação e cadastro do consumidor. O provedor valida os dados cadastrais do consumidor, bem como seu certificado digital para a troca de chaves públicas entre consumidor e provedor.

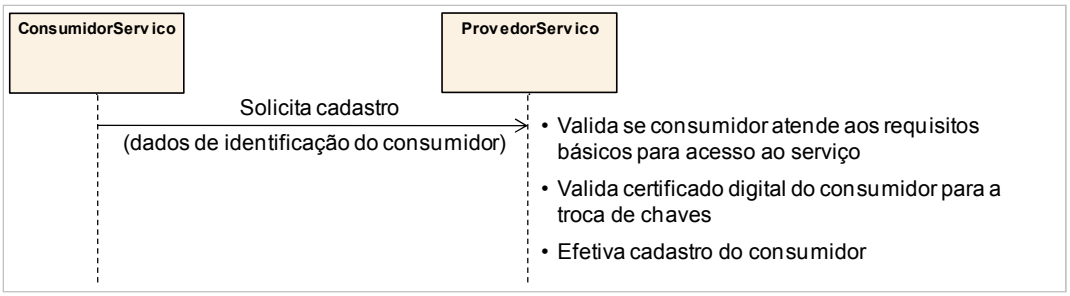

Figura 4.5 - Cadastramento do consumidor

- Passo 2: Personalização e envio do *token*

A personalização consiste em armazenamento no *token* de chaves privada e pública necessárias à operação do ASACS e uma inicialização de seu processo de geração de chaves simétricas de sessão.

Para o armazenamento das chaves privada e pública no *token*, há a necessidade de geração de três chaves, integrantes de três pares de chaves assimétricas, a saber:

- Par 1: chave privada (armazenada no *token*) e chave pública do consumidor a serem utilizadas nas interações entre consumidor e provedores;
- Par 2: chave privada e chave pública (armazenada no *token*) do provedor a serem utilizadas nas interações entre provedor e consumidores;
- Par 3: chave privada (atribuída ao usuário consumidor registrado) e pública (armazenada no *token*) geradas especificamente para o propósito de codificação dos comandos enviados ao *token*, como detalhado na Seção 4.6.

Quanto à inicialização do processo de geração de chaves simétricas de sessão pelo *token*, refere-se à sincronização inicial com um mecanismo em execução no servidor para a geração de chaves simétricas temporárias válidas por um dado intervalo de tempo.

Para o envio ao consumidor do *token* personalizado recomenda-se a utilização de um canal seguro *off-line* como, por exemplo, a utilização de um serviço de entrega contratado especificamente para este fim.

Neste passo, o provedor também disponibiliza ao consumidor um código secreto para ativação do *token*. O envio deste código secreto deverá ocorrer em momento e ou meio distinto do envio do *token*, ou seja, o *token* e respectivo código de ativação não devem ser enviados por um mesmo canal em um mesmo instante.

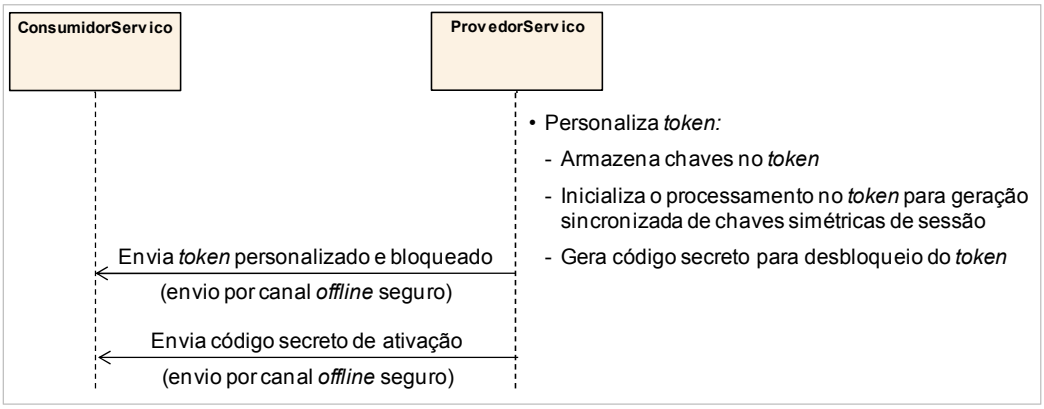

Figura 4.6 - Personalização e envio do *token*

- Passo 3: Ativação do *token* recebido pelo consumidor

A partir do recebimento do *token*, o consumidor pode então contatar o provedor para a ativação (desbloqueio) do dispositivo.

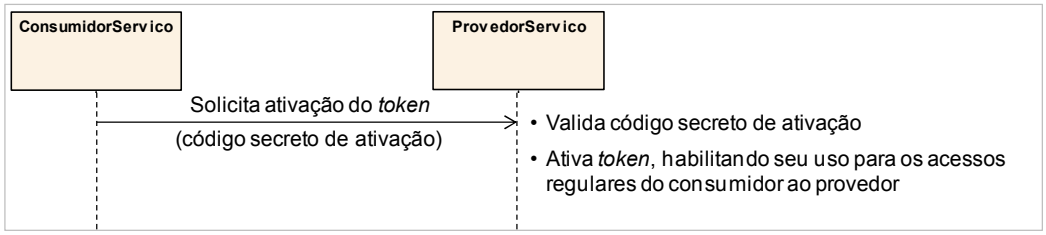

Figura 4.7 - Ativação do *token*

- Passo 4: Registro das seqüências de inicialização das estações do consumidor

Neste passo o consumidor deverá definir quais estações utilizará para as interações com o provedor de serviços. Para cada uma dessas estações deverá ser realizado um registro dos valores *hash* de cada camada da cadeia de inicialização. Este registro é feito em base de dados do provedor que será consultada no início de toda nova comunicação do consumidor com o provedor.

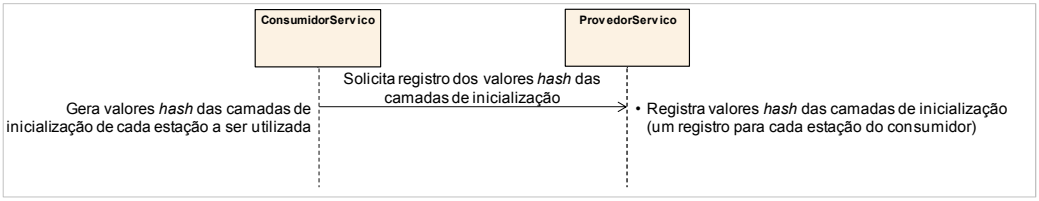

Figura 4.8 - Registro das seqüências de inicialização das estações dos consumidores

- Passo 5: Acesso regular do consumidor ao provedor

A partir deste passo, o consumidor está habilitado para o consumo dos serviços oferecidos a partir da validação da cadeia de inicialização em atividade na estação em uso pelo consumidor.

Para o primeiro acesso aos serviços do provedor, o consumidor deverá prover na mensagem os valores *hash* das camadas de inicialização assinados digitalmente pela chave privada do *token*.

O processo de geração da cadeia de assinaturas é viabilizado por uma operação denominada "gerarCadeiaDeAssinaturas" disponibilizada pelo programa tratador instalado no lado consumidor para as comunicações com o provedor. A operação "gerarCadeiaDeAssinaturas" é responsável por calcular os valores de *hash*, bem como assiná-los, para cada camada de inicialização (da BIOS à camada de aplicação). Para os cálculos de *hash* e assinaturas, essa operação utiliza APIs de comunicação com o *token* que então executa as funções criptográficas. O *token* executa estas funções através de algoritmos de *hash* (exemplo: SHA) e de criptografia e assinatura digital (exemplo: RSA). Para estas funções de criptografia e assinatura digital, o *token* utiliza a chave privada que está nele armazenada e que é de seu conhecimento exclusivo.

A cadeia de valores *hash* assinados é enviada sob criptografia com a chave pública do provedor. Após decriptografá-la, o provedor é então capaz de comprovar os princípios de autenticidade e irretratabilidade dado que os valores *hash* de cada camada estão assinados com a chave privada de conhecimento exclusivo do *token*. Uma vez decriptografada a seqüência de assinaturas, os valores *hash* de cada camada são então comparados com valores registrados previamente em base de dados do provedor. Estes valores registrados referem-se àqueles armazenados no passo de registro das seqüências de inicialização das estações do consumidor. Ao verificar que os valores *hash* são coincidentes, o provedor terá validado a integridade de cada camada da seqüência de inicialização remota no consumidor.

A Figura 4.9 representa em um diagrama de seqüência UML as principais interações entre os componentes do ASACS para a primeira invocação a uma operação de negócio do provedor. Nesta condição de primeira troca de

mensagens entre consumidor e provedor de serviços, é requerida a geração e validação da cadeia das camadas de inicialização do consumidor.

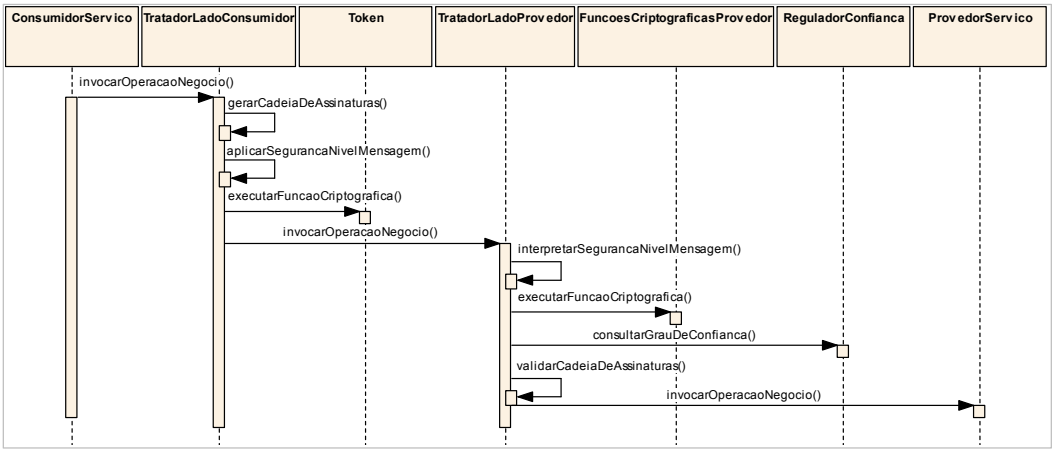

Figura 4.9 - Diagrama de seqüência para a primeira invocação de um serviço do provedor

Considerando a possibilidade de adulterações nas estações dos consumidores depois que as camadas de inicialização já foram carregadas, o ASACS especifica uma solução para validação periódica da cadeia de assinaturas sobre estas camadas.

Esta validação periódica proporciona um processo de atestamento dinâmico e está baseada na implementação de um mecanismo de desafio-resposta que é lançado pelo provedor ao consumidor. O funcionamento deste mecanismo está fundamentado na implementação de um controle no provedor que define, para cada consumidor, um *flag* que indica se está com uma cadeia de assinaturas válida ou inválida. Este *flag* tem uma validade definida por intervalos de tempo configurados no provedor. O *status* deste *flag* é verificado a cada mensagem recebida do consumidor e sempre que for identificado que o *flag* expirou é disparado pelo provedor um desafio-resposta ao consumidor. O consumidor, por sua vez, sempre que desafiado deverá responder com a cadeia de assinaturas atual que será então validada pelo provedor para atualização do *flag* de indicação da integridade deste consumidor.

Como parte dessa solução proposta para a validação remota da integridade dos consumidores é importante também considerar uma funcionalidade de revogação de chaves. Nos casos de perda da integridade de algum componente da cadeia de acesso como, por exemplo, o desaparecimento do *token*, o consumidor deve ser

capaz de solicitar a revogação das chaves, resultando no bloqueio pelo provedor de novas comunicações que se utilizem de chaves revogadas.

4.5.1.3. DESAFIO À VALIDAÇÃO DAS CAMADAS DE INICIALIZAÇÃO

Supondo a implementação deste atestamento das camadas de inicialização para ambientes das estações dos consumidores com sistemas operacionais de mercado, surge a necessidade de uma função de recadastramento ou atualização na base de dados do provedor com os registros das seqüências de inicialização aceitas. Esta necessidade é gerada, por exemplo, em função de alguma atualização em um dos componentes do *software* básico da estação em uso pelo consumidor.

Para resolver esse problema, o consumidor deve ser capaz de solicitar, por iniciativa própria, um recadastramento dos valores *hash* gerados para a cadeia de inicialização segura. Em resposta às solicitações de recadastramento, o provedor deve disponibilizar uma função de versionamento em sua base de modo a manter um histórico de todas as combinações aceitas. Através deste histórico, é possível otimizar a entrada de novas estações do consumidor nas comunicações com o provedor caso suas camadas de inicialização repitam a composição de uma cadeia já registrada na base histórica mantida pelo provedor.

Todavia, mesmo com o suporte dessa funcionalidade de recadastramento, pode ser muito difícil manter uma lista de todas as combinações possíveis de assinaturas das camadas em aplicações complexas com alto volume de consumidores. Como mencionado por Mittelsdorf (2004), esta dificuldade é devida ao fato que cada tipo de sistema operacional suporta diferentes e freqüentes conjuntos de atualizações (também conhecidas como *service packs*), o que resulta em muitas variações de valores *hash*. Neste cenário, a implantação deste atestamento das camadas de inicialização torna-se mais viável em aplicações com clientes que apresentem poucas versões de camadas. Alternativamente, esse processo de validação das camadas de inicialização poderia se ater a certas partes do *software* básico, visando diminuir o efeito das múltiplas atualizações.

4.5.2. Geração de chaves simétricas de sessão sincronizadas

Como uma técnica adicional para comprovação de autenticidade dos consumidores de serviços, é parte do ASACS a utilização do *token* definido na Seção 4.5.1 também para a geração de chaves sincronizadas com o servidor.

As chaves geradas por esse *token* são utilizadas para o papel de chaves simétricas de sessão na criptografia das mensagens trafegadas entre consumidor e provedor de serviços. A utilização de chaves simétricas de sessão contribui para:

- Em caso de roubo de uma chave trafegada, impedir reutilização em ataques futuros dado que é de uso temporário para uma dada sessão de comunicação entre provedor e consumidor de serviços;
- Aumentar o grau de dificuldade de ataque, ao gerar a necessidade de descoberta de ao menos três códigos secretos no lado do consumidor de serviços: a chave simétrica sincronizada e a chave privada armazenadas no *token* e ainda uma terceira chave privada, descrita na Seção 4.6, reservada para a codificação dos comandos de interface com o *token*.

Alternativamente à técnica de utilização de *tokens*, poderia ser considerada a distribuição de programas a serem instalados nas estações dos consumidores de serviços para o papel de geração sincronizada de chaves simétricas. Todavia, esta técnica resulta em uma desvantagem do ponto de vista de segurança que é a maior facilidade de cópia destes programas *versus* a dificuldade de acesso físico e clonagem dos *tokens*.

Quanto à classificação pelo critério de instante de execução, assumindo a recomendação deste trabalho para implementação do arcabouço com o modo de segurança no nível da mensagem, a sincronização de chaves simétricas poderá ocorrer, no limite, a cada mensagem trocada entre consumidor e provedor de serviços, caracterizando um atestamento dinâmico. Na prática, como a geração de cada nova chave simétrica ocorre no *token*, a chave é trocada apenas a cada mudança comandada pelo processador deste dispositivo de *hardware*. Assim, poderá ocorrer que um conjunto de mensagens sejam criptografadas pela mesma chave simétrica caso sejam trafegadas em um intervalo de tempo em que o processador do *token* ainda não tenha comandado a troca de chaves. Em função

desta possibilidade de reutilização de um valor da chave sincronizada, é possível incorporar à implementação do arcabouço um tratamento especial para determinadas operações que sejam especialmente críticas do ponto de vista de segurança. Para o caso em que uma operação disponibilizada pelo provedor demande uma proteção adicional contra o risco de ataques de repetição de requisição (*replay attacks*), é possível considerar um mecanismo em que a cada execução desta operação seja feito um controle no lado servidor que impeça novas execuções utilizando o mesmo valor de chave sincronizada.

Embora a implementação do arcabouço com segurança no nível de mensagem utilizando WS-Security caracterize tipicamente uma forma de atestamento dinâmico, é também possível enquadrá-la como uma forma de atestamento estático a depender da utilização conjugada com o padrão WS-SecureConversation (OASIS, 2007). Como definido por Gudgin (2004), WS-Security prove mecanismos para a segurança de cada mensagem trafegada, significando que as múltiplas mensagens trocadas entre consumidor e provedor podem gerar algum nível de degradação de desempenho ao requerer, por exemplo, a repetição dos processamentos de criptografia assimétrica necessários à troca das chaves simétricas. Para endereçar esta questão foi proposto o padrão WS-SecureConversation que define uma sessão segura, à semelhança do processo de *handshake* que caracteriza o estabelecimento do protocolo SSL entre um cliente e servidor. Sob esta sessão segura, uma mesma chave simétrica é reutilizada para otimização de desempenho das comunicações entre consumidor e provedor. Na condição de utilização de WS-SecureConversation conjugado com WS-Security, a etapa de atestamento “geração de chaves simétricas de sessão sincronizadas” caracteriza um atestamento estático, uma vez que a chave simétrica seria validada no estabelecimento da sessão e depois reutilizada sem revalidações enquanto a sessão permanecer ativa.

Para ilustrar a alternativa de implementação do arcabouço com o modo de segurança nó-a-nó, assumindo-se a segurança delegada à camada de transporte através do protocolo SSL, a chave simétrica de sessão seria definida no momento de *handshake* deste protocolo e então utilizada para todas as comunicações até o encerramento da sessão. Portanto, para o caso de uso de SSL em uma implementação do arcabouço sobre HTTPS, a geração de chaves simétricas de sessão sincronizadas caracteriza um atestamento estático. Importante também

registrar que, como a sincronização temporal de chaves não é nativa do protocolo SSL, deverá ser implementado um mecanismo de validação, pelo lado provedor, para assegurar que a chave simétrica gerada pelo *token* está sincronizada, no momento do estabelecimento da sessão, com um valor de chave válido gerado através de um processamento interno no provedor.

A Figura 4.10 representa a etapa de atestamento "geração de chaves simétricas de sessão sincronizadas". Para simplificar a representação desta etapa de atestamento do ASACS, não são apresentados na Figura 4.10 os programas tratadores intermediários das comunicações SOAP entre consumidor e provedor.

Figura 4.10 - Geração de chaves simétricas de sessão sincronizadas

A Figura 4.11, por sua vez, visa representar esta etapa de atestamento na forma de um diagrama de seqüência. Na representação das Figuras 4.10 e 4.11, é assumida uma comunicação seguinte ao primeiro acesso ao provedor e que dispensa a necessidade de apresentar a cadeia de assinaturas sobre valores *hash* da seqüência de inicialização. Também é assumido nesta representação que o consumidor está com o *flag* válido, controlado pelo provedor, que indica sua cadeia de inicialização como íntegra, conforme definido na Seção 4.5.1.2.

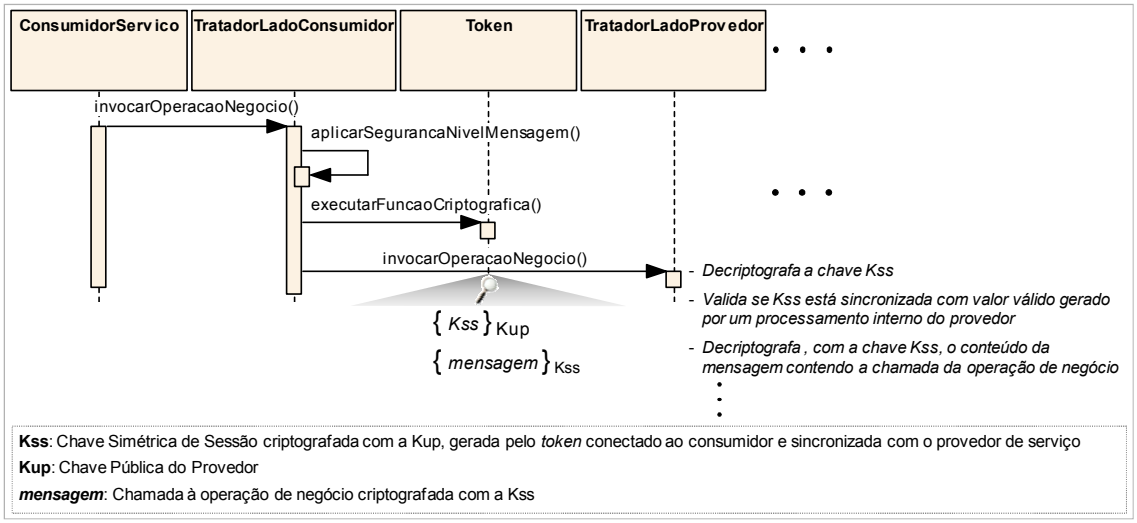

Figura 4.11 - Diagrama de seqüência representando a utilização da chave simétrica de sessão

Nas Figuras 4.10 e 4.11, nota-se a combinação de algoritmos de criptografia simétricos e assimétricos. A conjugação destes dois algoritmos no ASACS visa obter o que há de melhor em cada algoritmo:

- Uso do algoritmo simétrico para a criptografia da mensagem propriamente dita, uma vez que é mais rápido que o assimétrico. Este aspecto é especialmente importante considerando que aplicações SOA são suscetíveis a questões de desempenho, especialmente em função da dependência da interpretação de arquivos XML e da necessidade de uma alta capacidade de vazão (*throughput)* às requisições dos múltiplos consumidores;
- Uso do algoritmo assimétrico para a troca das chaves simétricas, através da implementação, por exemplo, do algoritmo RSA, resolvendo o problema de escalabilidade inerente ao modelo de distribuição de chaves requerido pelas soluções baseadas puramente em algoritmos simétricos.

A criptografia representada nas Figuras 4.10 e 4.11 é aplicada sobre as partes da mensagem que requerem confidencialidade e não necessariamente sobre a mensagem toda. Esta flexibilidade é possibilitada pelo modo de segurança no nível da mensagem com o uso do padrão WS-Security. Como exemplo, apenas a parte da mensagem SOAP referente à invocação da operação de negócio e seus argumentos de entrada poderia ser submetida à criptografia, evitando-se processamentos desnecessários com a aplicação de confidencialidade sobre todas as partes da mensagem.

4.5.3. Autenticação via certificado e assinatura digital dos consumidores

A finalidade de um certificado digital é carregar dados cadastrais sobre um indivíduo, sistema ou entidade e principalmente associar a este a sua chave pública de maneira verificável por meio de um processo de confiança estabelecido com uma terceira parte, denominada autoridade certificadora.

Como referido na Seção 1.2, a implementação típica de SSL provê a validação de certificados digitais apenas dos provedores de serviços (servidores). Todavia, considerando o cenário de soluções SOA, com potencial de um grande número de consumidores em uma rede distribuída de serviços, torna-se fundamental também a validação dos certificados dos consumidores de serviços.

No contexto do ASACS, a contribuição do uso de certificados digitais dos consumidores está na implementação do princípio de autenticidade dos consumidores perante os provedores. Ao apresentarem certificados emitidos por autoridades certificadoras de confiança dos provedores, os consumidores contribuem para a demonstração de sua legitimidade perante os provedores.

Quanto às assinaturas digitais, conforme referido na Seção 2.4, podem ser definidas como valores *hash* criptografados utilizados para validar a integridade da mensagem trafegada e assegurar que foi enviada pelo detentor da chave privada.

A contribuição da assinatura digital sobre as mensagens enviadas pelo consumidor está em aumentar a dificuldade de ataque ao implementar um mecanismo de validação de legitimidade através dos princípios de autenticidade e irretratabilidade perante os provedores de serviços, além de promover a validação de integridade contra alterações em trânsito.

Assumindo-se a recomendação deste trabalho para implementação do arcabouço com o modo de segurança no nível da mensagem, o envio do certificado e a assinatura digital poderão ocorrer a cada mensagem trafegada, caracterizando um atestamento dinâmico. Por outro lado, na hipótese de implementação com o modo de segurança fim-a-fim, com base em SSL, o certificado e assinatura digital ocorreriam apenas no momento de *handshake*, caracterizando um atestamento estático.

A Figura 4.12 representa a adição da certificação e assinatura digital ao arcabouço, com a participação da autoridade certificadora entre consumidor e provedor de serviços. Novamente com o objetivo de simplificar a representação desta etapa de atestamento do ASACS, não são apresentados na Figura 4.12 os programas tratadores intermediários.

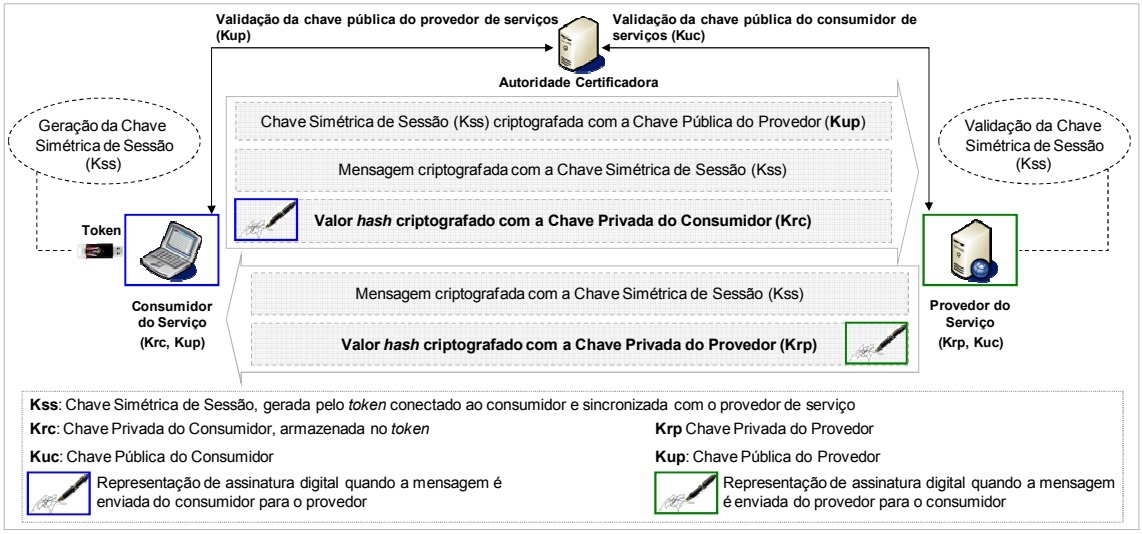

Figura 4.12 - Certificação e assinatura digital bilaterais

Uma representação em diagrama de seqüência UML é apresentada na Figura 4.13, incluindo a participação do componente AutoridadeCertificadora para verificação das chaves públicas informadas por ambas as partes (consumidor e provedor). Na representação das Figuras 4.12 e 4.13 é assumido um momento da comunicação em que o consumidor é dispensado da necessidade de envio da cadeia de assinaturas sobre valores *hash* das camadas de sua seqüência de inicialização.

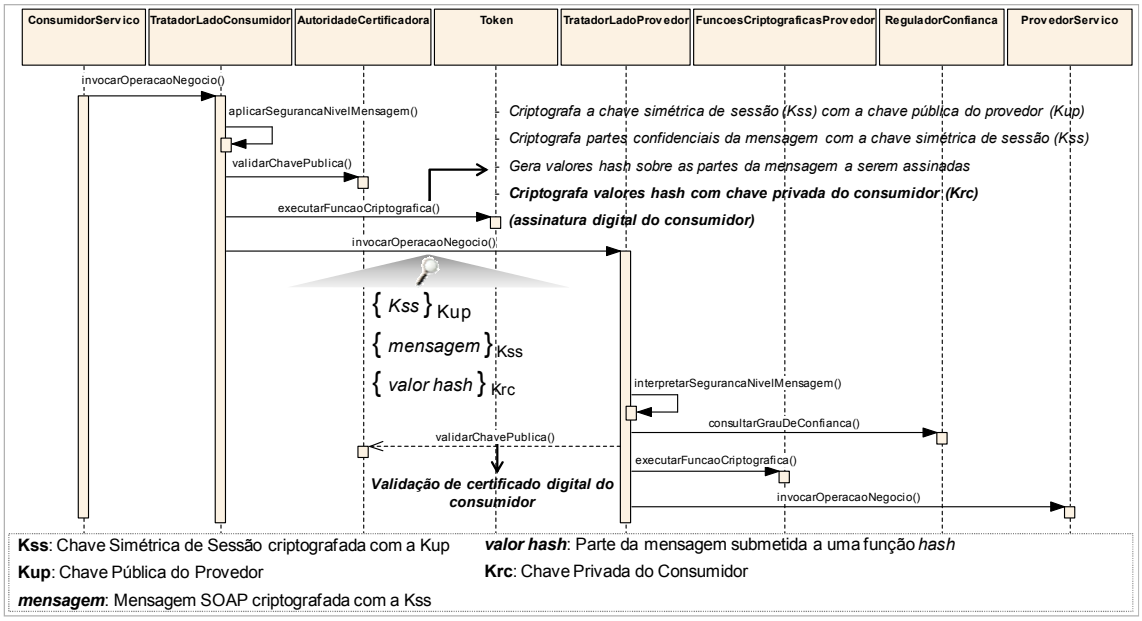

Figura 4.13 - Diagrama de seqüência da certificação e assinatura digital do consumidor

Com base no processo de assinatura digital e visando evitar os ataques de reescrita de XML, a implementação do ASACS deve contar com a aplicação da técnica proposta por Rahaman; Schaad e Rits (2006). Por esta técnica, é aplicada a assinatura digital sobre uma nova *tag* adicionada ao envelope SOAP denominada SOAAccount. Esta nova *tag* tem a função de comportar informações estruturais como, por exemplo, o número total de elementos filho do envelope e número de elementos de cabeçalho. A *tag* SOAPAccount é assinada digitalmente, possibilitando a validação de integridade no lado provedor contra alterações sobre a própria *tag*. Uma vez validada a integridade da *tag*, o provedor é então capaz de verificar se o envelope SOAP está preservado tal como descrito nas informações estruturais contidas na *tag* SOAPAccount.

4.5.4. Regulação do grau de confiança sobre os consumidores

4.5.4.1. CONFIANÇA

Josang (1999) define a confiança como uma opinião (crença) subjetiva resultante da avaliação de um sistema e seu ambiente. Em várias literaturas de

segurança da informação é apresentada uma relação entre confiança e segurança, definindo que se um sistema é considerado seguro, então ele é confiável. Por outro lado, há também definições baseadas na condição oposta e que consideram um sistema como não confiável se o mesmo for classificado como inseguro.

Mittelsdorf (2004) refere que dada a impossibilidade de determinar perfeitamente a segurança de sistemas, a confiança vem sendo utilizada como uma medida subjetiva da segurança real destes sistemas.

Um atributo importante da confiança é o grau de incerteza que caracteriza qualquer modelo aplicado para sua medição. Tipicamente a confiança é medida em um intervalo de zero a um, onde zero representa nenhuma confiança e um a máxima confiança.

Um modelo de confiança bastante aceito é apresentado por Mitteldorf (2004), onde as relações são definidas aos pares, definindo que uma entidade “A” confia na entidade “B” com certo valor que exprime a grandeza desta confiança. Em termos formais, este modelo define a medida de confiança expressa por uma dupla (minC, minD), onde minC e minD podem assumir valores no intervalo [0, 1] e representam, respectivamente, a mínima confiança e mínima desconfiança de uma entidade em relação à outra.

Para simplificação do modelo de confiança a ser considerado para o ASACS, é assumida apenas a grandeza de confiança, não sendo utilizadas referências expressas aos valores de desconfiança. Como será apresentado na Seção 4.5.4.4 a confiança inicial será considerada em seu valor máximo (valor um) para cada consumidor que é aceito e registrado na rede distribuída de serviços. Os requisitos ao algoritmo para a regulação do grau de confiança definem uma lógica que resulta no decréscimo do grau de confiança a partir de eventos negativos ou suspeitos provocados pelos consumidores.

No ASACS, a regulação do grau de confiança sobre os consumidores é viabilizada por um mecanismo colaborativo onde todos os provedores compartilham as informações sobre eventos negativos ou suspeitos dos consumidores. Para o funcionamento deste mecanismo, é também assumido um grau de confiança máximo (valor um) entre os provedores e entre os provedores e a entidade denominada agente regulador, responsável pelos registros dos graus de confiança

dos consumidores. Ou seja, para a operação desta etapa de atestamento, é fundamental que os provedores e agente regulador confiem nas informações e serviços prestados entre si. Por este modelo, a redução do grau de confiança deve ser aplicável estritamente sobre os consumidores da rede distribuída de serviços. A confiança entre provedores e entre provedores e o agente regulador deve ser mantida como uma constante que representa a máxima confiança entre essas entidades.

4.5.4.2. MOTIVAÇÃO PARA A MONITORAÇÃO DO GRAU DE CONFIANÇA

A importância da regulação do grau de confiança no ASACS se deve ao fato que os consumidores podem ser alvo de ataques explícitos ou dissimulados, podendo revelar inadvertidamente informações sensíveis à segurança da aplicação. Estas informações, na posse de usuários não legítimos, podem ser utilizadas para tentar fraudar o sistema.

Trata-se da situação em que esses usuários não legítimos conseguem ser atestados nas três etapas de validação de legitimidade definidas pelo ASACS, utilizando-se das informações privilegiadas obtidas através de ataque aos consumidores. Estes usuários podem então alcançar então um *status* na rede distribuída de serviços que os classificam como válidos pelas três etapas de validação de legitimidade do ASACS, a saber: validação da integridade das camadas de inicialização, geração de chaves simétricas de sessão sincronizadas e autenticação via certificado e assinatura digital.

Por outro lado, é provável que esses usuários não legítimos passem a incorrer em comportamentos negativos ou suspeitos que podem ser monitorados pelo arcabouço através de uma quarta etapa de atestamento. Definida como uma etapa de validação de comportamento, a regulação do grau de confiança dos consumidores pode revelar a ação de usuários não legítimos e então limitar o potencial de fraude.

Desta forma, através da regulação do grau de confiança, mesmo que um consumidor seja capaz de apresentar uma cadeia válida de camadas de inicialização, bem como chaves simétricas de sessão reconhecidas pelo provedor e

ainda certificados e assinaturas digitais válidos, ainda assim poderá ser delatado na rede se realizar comportamentos negativos ou suspeitos nas interações com os provedores de serviços. Trata-se de uma solução suportada por uma rede colaborativa, onde todos os provedores contribuem entre si para proteção da rede como um todo.

4.5.4.3. MONITORAÇÃO DO GRAU DE CONFIANÇA COMO ETAPA DE ATESTAMENTO

A regulação do grau de confiança, no contexto do ASACS, tem por objetivo monitorar o comportamento dos consumidores nas interações com os provedores em uma rede distribuída de serviços.

Na rede distribuída de serviços, identifica-se três papéis para a regulação da confiança: serviço de registro, entidade opinante e entidade avaliada (usuária da rede). O serviço de registro é identificado no ASACS como o agente regulador. As entidades opinantes, por sua vez, correspondem aos provedores de serviços, enquanto as entidades avaliadas são os consumidores. Adicionalmente, é importante também definir a base de dados mantida pelo agente regulador para o armazenamento das informações de grau de confiança de cada consumidor em um dado instante. Portanto, para a implementação da regulação do grau de confiança, além das entidades tradicionais de rede de serviços (consumidores e provedores), deve-se definir os seguintes participantes:

- Agente regulador: papel de centralizar a recepção das delações de cada comportamento negativo ou suspeito e também as notificações de eventos regulares e positivos dos consumidores como será descrito na Seção 4.5.4.4. É responsabilidade do agente regulador a implementação do algoritmo de graduação da confiança dos consumidores na rede distribuída de serviços. Através do agente regulador são disponibilizadas aos provedores as consultas dos graus de confiança dos consumidores;
- Base de dados dos graus de confiança: repositório de informações do grau de confiança atribuído em um determinado momento a cada consumidor da rede distribuída de serviços.

Em uma rede distribuída é importante considerar que mesmo que um determinado provedor receba acessos não suspeitos de um consumidor em particular, este consumidor pode estar atacando outros provedores da rede. Neste cenário, é importante a implementação do conceito de segurança distribuída conforme definido por Mittelsdorf (2004). Através da segurança distribuída, a vigilância sobre o comportamento dos consumidores é compartilhada entre todos os provedores que então interagem com o agente regulador para alimentação de uma base comum com a graduação de confiança de cada consumidor.

Quanto à classificação pelo critério de instante de execução, o atestamento via validação do grau de confiança pode ser definido como um atestamento dinâmico. Afinal, a cada interação com um provedor de serviços, o consumidor é submetido a uma consulta de seu grau de confiança na rede, sendo arbitrada ao provedor a decisão por negar ou prosseguir com a operação solicitada pelo consumidor.

A Figura 4.14 representa esta quarta etapa do ASACS, com foco na validação do comportamento dos consumidores de serviços. Nesta figura, destacam-se as entidades adicionais “agente regulador” e “base com os registros dos graus de confiança” e, para simplificar apresentação do conceito fundamental desta etapa, está omitida a representação dos programas tratadores intermediários.

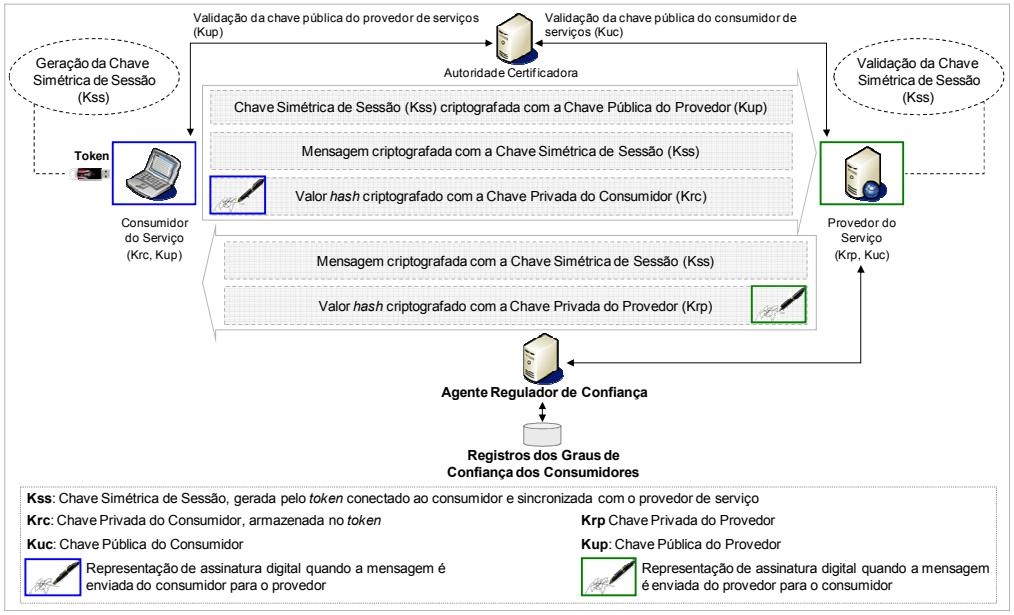

Figura 4.14 - Regulação do grau de confiança dos consumidores

Uma representação da regulação do grau de confiança através de diagrama de seqüência UML é apresentada na Figura 4.15, incluindo dois exemplos de interação:

um para registro de comportamentos negativos ou suspeitos e outro para consulta do grau de confiança de um dado consumidor em um determinado instante.

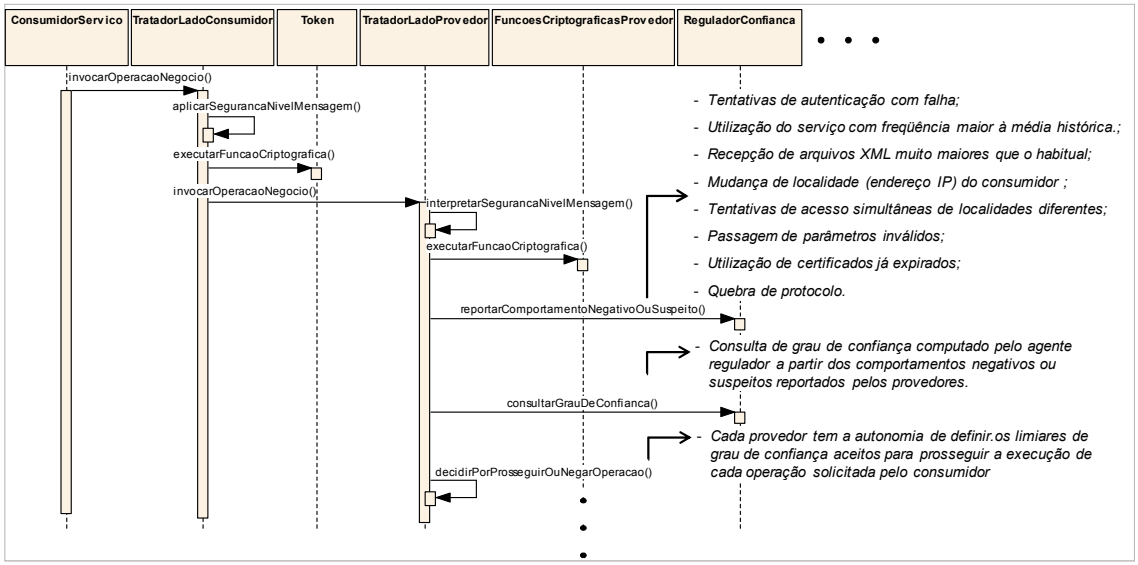

Figura 4.15 - Diagrama de seqüência representando a regulação do grau de confiança

Para o funcionamento da regulação do grau de confiança, cada provedor de serviço é responsável por implementar métodos para identificar e reportar os comportamentos dos consumidores. Para cumprir esta responsabilidade, os provedores devem recorrer aos chamados programas tratadores já definidos neste trabalho. Em seu papel de intermediários das comunicações entre consumidor e provedor, os programas tratadores contribuem para garantir a segregação entre as funções de segurança e as funções de negócio das aplicações. Através dos programas tratadores com o apoio de classes base da biblioteca de suporte ao arcabouço, são disponibilizados métodos padrão para identificar os comportamentos negativos ou suspeitos, comunicá-los e registrá-los junto ao agente regulador e também para consultar o grau de confiança de um dado consumidor em um dado instante. Como será apresentado na Seção 4.5.4.4, uma implementação mais completa do algoritmo de regulação do grau de confiança requer que os programas tratadores reportem também comportamentos regulares ou positivos, alimentando um processo de compensação de eventos com interferências de aumento e diminuição dos graus de confiança atribuídos.

Em conseqüência das funções críticas acumuladas pelo agente regulador, o funcionamento eficiente de uma instância do ASACS requer as devidas proteções de segurança, tempos de resposta adequados e alta taxa de disponibilidade deste

agente. As seções 4.5.4.5 e 4.5.4.6 apresentam técnicas para tratar os requisitos de segurança, desempenho e disponibilidade do agente regulador.

4.5.4.4. REQUISITOS DO ALGORITMO DE REGULAÇÃO DO GRAU DE CONFIANÇA

O algoritmo a ser implementado pelo agente regulador de confiança deverá ter a capacidade de submeter os eventos reportados pelos provedores a um processamento de atualização do grau de confiança de um dado consumidor.

Cada consumidor deve receber um grau de confiança inicial máximo (valor um) que poderá ser reduzido ao longo das interações do consumidor com os provedores da rede distribuída de serviços. Comportamentos negativos ou suspeitos dos consumidores são reportados pelos provedores ao agente regulador, onde é implementado um algoritmo responsável por converter estes eventos em variações aplicadas sobre o grau de confiança dos consumidores.

O algoritmo a ser implementado como parte da solução proposta pelo ASACS deverá ser capaz de tratar o seguinte conjunto mínimo de comportamentos negativos ou suspeitos (MITTELSDORF, 2004) dos consumidores em uma solução *Web* SOA:

- Tentativas sucessivas de autenticação com falha;
- Utilização de um serviço de modo diferente ao habitual como, por exemplo, com freqüência muito maior à média histórica. Em situações extremas, este comportamento pode caracterizar uma tentativa de ataque de negação de serviços (*denial of service*);
- Recepção de arquivos XML muito maiores que o habitual, o que pode levar a situações de falha de processamento no provedor e até mesmo negação de serviços, especialmente quando o interpretador requer a carga em memória de todo o arquivo XML a ser processado;
- Mudança de localidade (endereço IP) do consumidor ou tentativas de acesso simultâneas de localidades diferentes;
- Erros de invocação como, por exemplo, passagem de parâmetros inválidos;

- Utilização de certificados já expirados, podendo indicar que foram roubados;
- Quebra de protocolo, identificada quando a interação com o serviço não segue o protocolo estabelecido, podendo indicar uma tentativa de explorar alguma falha.

Após sucessivas reduções do grau de confiança de um dado consumidor, seus acessos aos serviços da rede ficam cada vez mais restritos. Supondo a situação de falso positivo, em que o consumidor é de fato legítimo e que eventos suspeitos não configurem tentativas reais de ataque, faz-se então necessária uma intervenção para restaurar o grau de confiança e reabilitar o consumidor como usuário da rede.

Com o objetivo de garantir uma operação do arcabouço mais autônoma no que se refere à regulação do grau de confiança, faz-se necessária uma solução que reduza a necessidade de intervenções de um administrador da rede de serviços para restabelecer o grau de confiança de um dado consumidor. Para este objetivo de operação mais autônoma do ASACS, deve-se considerar também o registro dos eventos regulares ou positivos dos consumidores. Cada serviço especifica quais eventos são suspeitos ou negativos e quais são regulares ou positivos, de acordo com suas características e necessidades de segurança. Esta classificação dos eventos é específica por serviço e pode ser melhorada com a experiência do uso continuado do serviço pelo conjunto de consumidores da rede. Através do registro dos eventos regulares ou positivos, o algoritmo de regulação deve resultar em aumento dos graus de confiança, gerando um processo de compensação que reduz a freqüência das situações que requeiram a intervenção do administrador.

Por esse mecanismo de compensação, o agente regulador passa então a processar o conjunto total de comportamentos de um consumidor na rede distribuída de serviços, o que requer a disponibilização de interfaces entre os provedores e o agente regulador também para o registro dos comportamentos regulares ou positivos. A graduação de confiança atribuída passa então a ser impactada positivamente para os casos em que o consumidor compense comportamentos suspeitos ou negativos com uma contrapartida de comportamentos regulares ou positivos.

A implementação do algoritmo com estas características de compensação pode, em alguns casos, até mesmo recuperar a graduação de confiança de um

consumidor bloqueado, habilitando-o automaticamente a novas requisições aos provedores de serviços da rede.

Esta implementação não é trivial e traz consigo riscos colaterais ao desempenho e também à segurança da rede de serviços. Os riscos ao desempenho são devidos a uma possível sobrecarga das comunicações dos provedores com o agente regulador, o que pode ser mitigado através de uma solução de comunicações em lote. Os provedores acumulariam lotes de informações sobre os eventos de comportamento regular ou positivo para que sejam comunicados ao agente regulador em intervalos de tempo parametrizados. Quanto ao risco de segurança, por sua vez, é devido à possibilidade de consumidores mal intencionados manipularem o balanceamento entre comportamentos regulares ou positivos e suspeitos ou negativos, a partir do conhecimento do modo de operação do algoritmo. Através desta manipulação, buscariam perpetuar as tentativas de ataque através das devidas compensações de seus comportamentos na rede. Uma alternativa a considerar para evitar tal tipo de manipulação é considerar uma redução do grau de confiança de um consumidor quando o mesmo demonstra alternância entre comportamentos regulares ou positivos e suspeitos ou negativos.

Outro aspecto importante a considerar na solução é que deve ser de responsabilidade de cada provedor definir os limiares de grau de confiança aceitos para prosseguir a execução de cada operação publicada pelo provedor. Assim, enquanto para determinada operação publicada por um provedor é requerido um grau mínimo de confiança do consumidor no valor de, por exemplo, 0.9 em um intervalo [0, 1], para outras operações poderá ser permitido executá-las para consumidores com valores inferiores do grau de confiança como, por exemplo, 0.5.

Por fim, para ilustrar soluções de implementação do algoritmo pelo agente regulador do grau de confiança dos consumidores, são apresentadas a seguir abordagens propostas por Shah et al. (2009) e que monitoram o comportamento dos consumidores com o objetivo de evitar dois tipos de ataques de negação de serviço em soluções *Web* SOA. Dado que não é possível evitar estes tipos de ataque através de *firewalls* ou sistemas de IDS (*Intrusion Detection Systems*), as abordagens de Shah et al. (2009) são baseadas na implementação, no nível da aplicação, de métodos para monitoração do comportamento dos consumidores. A primeira abordagem endereça os ataques de negação de serviços a partir da

repetição de requisições (*replay attacks*) e é denominada pelos autores como "prevenção contra a inundação de *Web services*". Para sua implementação é definido um algoritmo que compara, para cada nova invocação, se esta ocorreu em um intervalo de tempo mínimo em relação à invocação anterior. Invocações que ocorram em um intervalo menor que o mínimo definido pelo algoritmo disparam um mecanismo de bloqueio do consumidor de serviços. A segunda abordagem, por sua vez, endereça a negação de serviços a partir do aumento malicioso de arquivos XML e é denominada pelos autores como "prevenção à negação de serviço a partir de mensagens SOAP grandes". A implementação desta segunda abordagem é também baseada em uma tabela para comparação das características das invocações correntes contra valores limitantes calculados pelo algoritmo. Neste caso, para cada *Web servic*e é calculado um tamanho máximo das mensagens SOAP a serem recebidas e então monitorado o tamanho de cada mensagem enviada pelos consumidores. Quando um consumidor envia uma mensagem que excede o tamanho máximo, esta mensagem é bloqueada e o consumidor de serviços é registrado em um *log* que o classifica como um risco ou ameaça potencial. Em caso de reincidência, o consumidor é então bloqueado permanentemente e passa a depender de uma intervenção externa para comprovar sua legitimidade e ser reativado perante os provedores de serviços da rede.

4.5.4.5. SEGURANÇA DO AGENTE REGULADOR

Um aspecto importante a considerar sobre o agente regulador é a sua própria segurança contra ataques de falsos provedores de serviços ou de provedores legítimos sob controle de usuários mal intencionados. Embora delimitado em uma rede fechada (não publicado a consumidores externos ao *firewall)*, o agente regulador também deve ser compreendido como um provedor de serviços e, como tal, suscetível em certo grau aos ataques discutidos neste trabalho.

Para tratar essa condição, a proposta é recorrer ao uso da técnica de lista branca (*white list*). A utilização de lista branca é baseada no pressuposto de um volume controlado de provedores de serviços. Afinal, ao contrário das operações publicadas pelos provedores de serviços, as operações do agente regulador são

destinadas ao uso por um conjunto supostamente conhecido de participantes da rede (os próprios provedores de serviços).

A cada nova inclusão de um provedor no mesmo domínio de rede de um agente regulador, o mesmo deverá ser submetido a um processo de cadastro na lista branca que está sob controle deste agente regulador.

4.5.4.6. DESEMPENHO E DISPONIBILIDADE DO AGENTE REGULADOR

A depender da quantidade de provedores de serviços com que interage, bem como do volume de acessos dos consumidores a estes provedores, o agente regulador poderá representar um ponto de gargalo ao desempenho da solução.

Quanto à disponibilidade, é também um requisito não funcional muito importante na implementação do agente regulador. Em caso de indisponibilidade por falha de *software* ou mesmo de *hardware*, o funcionamento seguro da rede distribuída de serviços fica comprometido. Neste contexto, devem ser evitadas implantações do ASACS que utilizem uma ou poucas instâncias do agente regulador e também da base com registro dos graus de confiança dos consumidores.

Para responder a estas questões de desempenho e disponibilidade, considerando a criticidade da operação eficiente do agente regulador, a solução proposta neste trabalho define a utilização conjugada de três técnicas a serem aplicadas na implementação de uma instância do ASACS:

- *Pooling* e balanceamento de carga

  Os papéis requeridos aos agentes reguladores no ASACS possibilitam sua qualificação como objetos com modo de operação *stateless*. O modo de operação *stateless* significa que não se baseiam no armazenamento de qualquer informação de estado das interações anteriores com um dado provedor de serviços. Ou seja, não são utilizados atributos nas instâncias das classes que implementam os agentes reguladores. Todas as informações necessárias às funcionalidades do agente regulador são disponibilizadas como parte das mensagens enviadas pelos provedores ou estão disponíveis em uma base de dados acessada pelos métodos do agente regulador.

Sem o armazenamento de atributos nas instâncias dos agentes reguladores é possível então recorrer a soluções de otimização de desempenho como a utilização da técnica de *pooling* que é tipicamente disponibilizada pelos servidores de aplicação do mercado. Um *pool* de instâncias de agentes reguladores representa um conjunto de objetos pré-instanciados e que podem ser reutilizados entre as várias requisições dos provedores da rede. O fato de já estarem pré-instanciados contribui para a redução do tempo de processamento por eliminar a necessidade de execução da carga do objeto a cada invocação feita por um provedor de serviços.

Adicionalmente à técnica de *pooling*, pode-se também configurar *proxies* (MILANOVIC, 2006) que, através de um algoritmo de balanceamento de carga, executam o mapeamento de qual servidor de aplicação deve ser acionado a cada requisição de consulta ou atualização dos graus de confiança dos consumidores. Mais do que apenas mapear o servidor de aplicação a ser acionado, o uso de *proxies* pode também apoiar a identificação, dentro do servidor selecionado, de qual instância do agente regulador deve ser utilizada para o processamento de uma dada requisição. Como pode ser observado, os algoritmos de balanceamento dos *proxies* atendem tanto aos requisitos de disponibilidade quanto aos de desempenho ao suportarem a distribuição de instâncias do agente regulador entre diversos servidores de aplicação.

Considerando as interações dos agentes reguladores com um banco de dados relacional (onde são registradas as graduações de confiança dos consumidores), é possível também assumir a implementação de um *pool* de conexões com este banco de dados. As conexões pré-estabelecidas com o banco de dados também contribuirão para a redução dos tempos de resposta do agente regulador aos provedores de serviços.

- <u>*Cache* das graduações de confiança dos consumidores</u>

As transações de consulta invocadas pelos provedores de serviços para identificar o grau de confiança de um dado consumidor tendem a ser as de maior freqüência nas interações com o agente regulador. Sem considerar qualquer tratamento de otimização, a toda nova requisição de um consumidor,

o provedor invocará o agente regulador para uma operação de consulta do grau de confiança deste consumidor.

Neste cenário, a implantação do arcabouço poderia resultar em uma sobrecarga da rede, o que gera a necessidade de uma proposta de otimização das consultas ao grau de confiança dos consumidores.

Para solucionar esse problema é então proposta uma solução de *cache* local em memória das graduações de confiança a serem então acessadas diretamente pelos provedores de serviços. Como a criticidade da segurança não necessariamente é a mesma para todos os serviços disponibilizados na rede, a solução deve suportar diferentes prazos de validade dessa informação em *cache.* Assim, para diferentes provedores ou mesmo para diferentes operações disponibilizadas por um mesmo provedor deve ser possível configurar diferentes durações para a validade deste *cache* das graduações de confiança dos consumidores.

- Distribuição do armazenamento dos graus de confiança dos consumidores

Analogamente à importância de desempenho e disponibilidade do programa que implementa o agente regulador, é também essencial considerar estes mesmos requisitos não funcionais para a base de dados que armazena as informações de grau de confiança dos consumidores.

No ASACS, essa base de dados é acessada via o agente regulador para as operações de atualização e consulta do grau de confiança corrente de um dado consumidor. Assumindo o pressuposto de utilização de um banco de dados relacional, a depender dos volumes de acesso, é possível que os servidores de base de dados demandem *upgrades* de *hardware* e ajustes de configuração comandados por DBAs (profissionais de administração de base de dados) para atender aos requisitos de desempenho.

Considerando-se também os requisitos de disponibilidade, uma técnica sugerida é a utilização de *Distributed Hash Table (DHT*) que, como o próprio nome sugere, corresponde a uma estrutura de tabela *hash* distribuída entre um conjunto de computadores que cooperam entre si, denominados nós e organizados em partições (GHODSI, 2006). Exatamente como uma *hash table*, a função principal de uma DHT é oferecer operações de busca, retornando os

itens (valores) associados com uma determinada chave que consiste em um valor *hash* gerado através de um algoritmo de *hash* como, por exemplo, o SHA1. Considerando que cada nó (computador) de uma estrutura DHT não armazena todos os itens, as requisições a um determinado nó são roteadas para outros nós sempre que o valor solicitado não estiver sob responsabilidade do nó inicial da busca. Para o funcionamento deste mecanismo, cada nó tem uma tabela de roteamento com ponteiros a outros nós, conhecidos como nós vizinhos. A requisição é roteada entre as diversas partições em que estão organizados os nós até alcançar a partição e nó que contenha o item associado à chave (valor *hash*) de pesquisa. Para oferecer desempenho, as soluções DHT são tipicamente suportadas por algoritmos de pesquisa que buscam a otimização do processo.

A Figura 4.16 representa a utilização de múltiplas de instâncias do agente regulador em um modelo de balanceamento de carga e *pooling* e também a distribuição da base de dados dos graus de confiança em um modelo DHT.

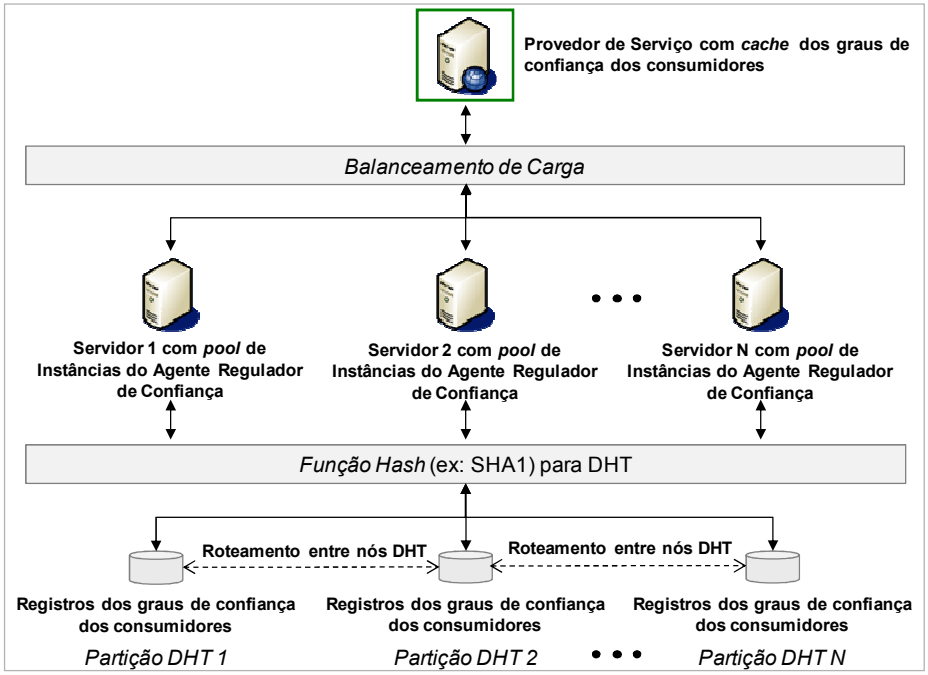

Figura 4.16 - Modelo de alto desempenho e disponibilidade do agente regulador

Como exemplo de interação entre os elementos representados na Figura 4.16, os seguintes passos seriam executados na situação em que um dado consumidor de serviço solicite uma operação ao provedor:

- O provedor verifica se a informação do grau de confiança daquele consumidor está disponível em *cache*;
- Se a informação estiver disponível e não for considerada expirada conforme parametrização do prazo de validade do *cache* para a operação em questão, o provedor avalia se o grau de confiança atinge o limiar mínimo requerido para prosseguir a operação, decidindo por executá-la ou rejeitá-la;
- Caso o provedor identifique que a informação não está em *cache* ou encontra-se expirada, então requisita ao agente regulador a consulta do grau de confiança deste consumidor de serviço;
- Através de uma solução de balanceamento de carga, esta requisição de consulta é então direcionada a um dado servidor com múltiplas instâncias do agente regulador;
- No servidor selecionado é então acionada a primeira instância disponível do agente regulador;
- A instância do agente regulador acionada submete então o identificador do consumidor de serviço a uma função *hash* como, por exemplo, SHA1 que é comumente utilizada para DHT;
- Como resultado da função *hash*, é gerado um índice utilizado para identificar a partição onde se encontram as informações sobre o grau de confiança deste consumidor;
- Uma vez encontrado o registro com as informações sobre o grau de confiança, o fluxo de volta se inicia, terminando com a entrega destas informações ao provedor de serviços;
- De posse das informações do grau de confiança do consumidor, o provedor então decide se prosseguirá ou rejeitará a execução da operação.

4.6. Técnica para evitar uso do *token* por usuários não autorizados

Em função do risco de utilização indevida do *token* por algum usuário mal intencionado que tenha se apropriado deste dispositivo, há a necessidade de se considerar um mecanismo que permita a sua utilização apenas por um grupo controlado de usuários.

Conforme já mencionado na Seção 4.1, as interações com o *token* ocorrem através de interfaces encapsuladas em classes da biblioteca para suporte ao arcabouço. Com o objetivo de controlar o uso do *token*, a solução recomendada é a utilização de um protocolo baseado em criptografia assimétrica aplicada sobre os comandos de interação com o *token*.

Para implementar essa solução de controle sobre o uso do *token* é necessário disponibilizar um par de chaves assimétricas geradas especificamente para este fim. A chave pública deve ser armazenada no *token* e a correspondente chave privada deve ser de conhecimento exclusivo do consumidor registrado como usuário da rede distribuída de serviços. A geração e distribuição destas chaves fazem parte do passo descrito na Seção 4.5.1.2 como "personalização e envio do *token"*.

O mecanismo proposto define que os comandos enviados via APIs ao *token* sejam criptografados com a chave privada de conhecimento exclusivo do usuário consumidor. Ao receber os comandos criptografados, o *token* deverá decriptografá-los com a chave pública correspondente para então interpretá-los e efetivamente executá-los. Com o objetivo de ofuscar este mecanismo de utilização de chave privada para as comunicações com o *token*, é proposto que a mesma seja inoculada nos códigos dos programas consumidores.

A Figura 4.17 representa esse mecanismo para disponibilizar o funcionamento do *token* apenas a usuários legítimos, limitando o processamento das chamadas às suas funções criptográficas apenas aos detentores da chave privada de conhecimento controlado.

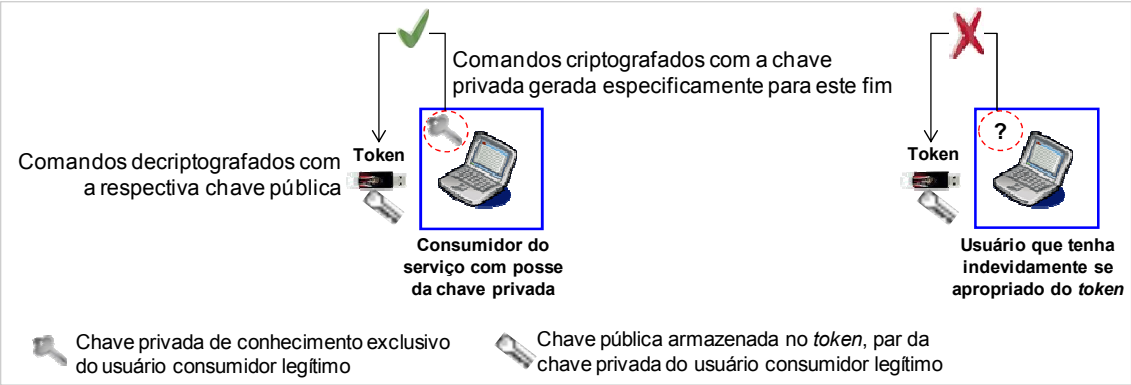

Figura 4.17 - Técnica para evitar o uso do *token* por usuários não legítimos

4.7. Requisitos do *token* de segurança

A definição do ASACS pressupõe a disponibilidade de *tokens* de segurança que deverão atender aos seguintes requisitos mínimos:

- Conter um processador interno para execução de funções criptográficas;
- Armazenar chaves privadas de um modo seguro;
- Disponibilizar funcionalidade de geração de chaves simétricas de sessão sincronizadas com um processo de geração de chaves em execução nos servidores;
- Disponibilizar interface de comunicação via APIs para:
  - Processar funções de criptografia simétrica e assimétrica;
  - Processar cálculo de *hash*, criptografia e assinatura com sua chave privada;
  - Reiniciar a geração de chaves simétricas de sessão em caso de perda da sincronização.
- Executar somente comandos (APIs) recebidos criptografados e com autenticação de origem;
- Possibilitar uma linha de produção a um custo não impeditivo à sua implementação em redes com grandes volumes de consumidores.

4.8. Requisitos da biblioteca de suporte

Para os objetivos deste trabalho, é proposto que a biblioteca de suporte ao ASACS seja baseada no paradigma de orientação a objetos e composta essencialmente por classes e interfaces que implementem os componentes definidos na Seção 4.3.

Assumindo a implantação do arcabouço com o modo de segurança fim-a-fim (segurança no nível da mensagem), um requisito básico da biblioteca é a implementação do padrão WS-Security. Através da disponibilização de funções que implementem WS-Security, a biblioteca assume um papel fundamental no desenvolvimento das classes dos programas tratadores no lado cliente e no lado servidor.

Embora não seja escopo do presente trabalho a implementação desta biblioteca de suporte ao arcabouço, é apresentado aqui o seguinte conjunto mínimo de funções a serem providas:

- Interação com o *token* para a geração dos valores *hash* e assinaturas para cada camada da seqüência de inicialização das estações do consumidor;
- Interação com o *token* para criptografar e assinar as mensagens enviadas ao provedor, utilizando-se o processamento de funções criptográficas executadas pelo *token* com base nas chaves privada do consumidor, pública do provedor e também a simétrica de sessão, todas armazenadas no *token*;
- Analogamente, interação com o *token* para invocação de suas funções criptográficas para decriptografar e validar assinaturas das mensagens recebidas dos provedores;
- Validação dos certificados digitais dos provedores e consumidores;
- Validação, no lado servidor, da cadeia de assinaturas das camadas da seqüência de inicialização das estações dos consumidores;
- Validação da chave simétrica de sessão enviada pelo consumidor, certificando que está sincronizada com um processamento interno no provedor;

- Implementação do processamento interno no provedor para a geração de chaves simétricas temporárias válidas por um dado intervalo de tempo;
- Decriptografia das mensagens enviadas pelos consumidores, bem como validação dos parâmetros de autenticação informados;
- Identificação e comunicação ao agente regulador de comportamentos suspeitos ou negativos dos consumidores de serviços;
- Analogamente, comunicação ao agente regulador dos comportamentos regulares ou positivos dos consumidores de serviços, assumindo a operação do algoritmo do agente regulador de confiança com compensação de comportamentos no cálculo do grau de confiança dos consumidores;
- Consulta ao agente regulador de confiança para obter o grau atual de confiança de um dado consumidor;
- Funções criptográficas e de assinatura digital utilizadas na solução pelo lado do provedor.

É importante mencionar que todas as especificações da biblioteca para o arcabouço devem possibilitar a implementação dos provedores de serviço em modo *stateless*. Conforme definição já apresentada na Seção 4.5.4.6, a classificação como *stateless* significa que o funcionamento de um objeto não depende do armazenamento de qualquer estado gerado em interações anteriores com este objeto. A implementação dos provedores de serviço em modo *stateless* torna possível a otimização de desempenho através dos conceitos de *pooling e proxies* para seleção da instância de *Web service* a ser utilizada a cada requisição (conceitos também já definidos na Seção 4.5.4.6). A pré-instanciação dos *Web services* (*pooling*) e a distribuição das cargas de processamento através de *proxies* são especialmente importantes para as soluções *Web* SOA uma vez que são tipicamente suscetíveis a problemas de desempenho.

4.9. Requisitos das estações dos consumidores

Para a implementação das funcionalidades especificadas para o ASACS faz-se necessário o atendimento a um conjunto de requisitos nas estações dos consumidores.

O cumprimento parcial destes requisitos reflete em implantações parciais das etapas de atestamento dos consumidores, com relação direta na redução do nível de segurança proposto pelo ASACS.

Os principais requisitos no lado consumidor para a operação plena do ASACS são:

- Possibilidade de instalação local dos programas tratadores que publicam as operações dos serviços de um provedor e também implementam as funções de segurança requeridas pelo arcabouço;
- Liberação de privilégios a esses programas tratadores instalados localmente para acesso a áreas de memória tipicamente protegidas como a BIOS, *boot loader* e sistema operacional;
- Habilitação de um canal para comunicação (exemplo: USB, Bluetooth etc.), com o dispositivo móvel de *hardware*, necessário ao funcionamento das etapas de validação de legitimidade definidas pelo arcabouço.

4.9.1 Instalação de programas na estação do consumidor

Para a implementação das funcionalidades do ASACS são requeridos programas clientes, para cada provedor de serviços, a serem instalados localmente nas estações dos consumidores.

Estes programas clientes representam os programas tratadores com o papel de publicação das operações dos provedores de serviços e também das funções de segurança requeridas pelo arcabouço.

Como referido na Seção 4.4, é necessário que programas tratadores sejam também implementados e disponibilizados no lado provedor para intermediação das

mensagens enviadas pelo consumidor. No papel de execução das funções requeridas pelo arcabouço, tanto no lado consumidor como no lado provedor, os programas tratadores utilizam a biblioteca de suporte. O biblioteca de suporte, por sua vez, provê as APIs para implementação do padrão WS-Security e de outras funções requeridas pelo arcabouço como o acesso de leitura às camadas de inicialização típicas de um computador (da BIOS à camada de aplicação).

Considerando-se como exemplo a linguagem de programação Java, a implantação de programas tratadores no lado cliente pode ser viabilizada através da instalação de arquivos com extensão ".jar". Os arquivos ".jar" conteriam os *bytecodes* com a implementação das chamadas às operações dos provedores de serviços, bem como a codificação das funções de segurança requeridas pelo arcabouço.

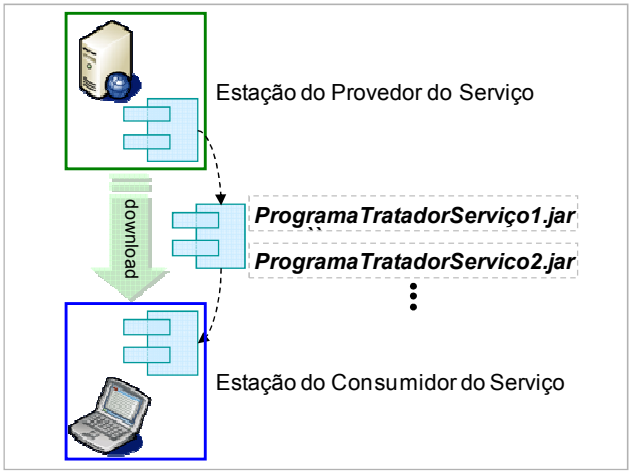

Figura 4.18 - Instalação de programas tratadores nas estações dos consumidores

Na Figura 4.18, a solução de *download* é ilustrativa e representa apenas uma das alternativas para instalação dos programas tratadores nas estações dos consumidores. Outras formas de distribuição podem ser utilizadas como, por exemplo, a disponibilização destes programas em mídias de armazenamento como CDs, DVDs e *pen-drives*.

Considerando-se a alternativa de *download* pela Internet, uma atenção especial deve ser dada para mitigar os riscos de segurança referentes à:

- Instalação dos programas clientes por consumidores não legítimos;

- Possibilidade de ataques às estações dos consumidores através da disponibilização de falsos programas tratadores.

Para mitigação destes riscos, recomenda-se a autenticação e autorização dos consumidores para execução dos *downloads* e a utilização de certificados digitais dos provedores para validação de sua autenticidade perante os consumidores.

4.9.2 Privilégios no ambiente das estações dos consumidores

Determinadas funções definidas pelo ASACS para os programas tratadores clientes instalados nas estações dos consumidores requerem acessos privilegiados ao ambiente destas estações. Como exemplo, na Seção 4.5.1.2 é descrita uma função para cálculo de valores *hash* das camadas típicas de inicialização de um computador (da BIOS à camada de aplicação). Para esta função, é pressuposto que esses programas tratadores clientes tenham acesso a áreas de memória tipicamente protegidas como a BIOS, *boot loader* e sistema operacional. É assumido neste trabalho que esses programas tratadores terão todos os privilégios requeridos nas estações dos consumidores de serviços.

4.9.3 Habilitação de um canal de comunicação com o *token* de segurança

Para a implementação das etapas do ASACS para validação de legitimidade dos consumidores é necessário o acoplamento de um dispositivo de *hardware* na estação dos consumidores, responsável por prover funções de criptografia e assinatura digital, bem como gerar chaves simétricas de sessão.

Não é escopo deste trabalho a definição da tecnologia para essa comunicação com o *token* segurança, reservando-se a um outro trabalho a pesquisa sobre alternativas como USB, Bluetooth, entre outras.

4.10. Considerações sobre a implantação do arcabouço

4.10.1 Características das aplicações candidatas à implantação do arcabouço

Como referido na Seção 4.1, o arcabouço proposto neste trabalho está fundamentado na implementação de etapas de atestamento que resultam na necessidade de um aumento do esforço para o sucesso dos ataques aos provedores de serviços.

Todavia, considerando as variáveis de custo e de complexidade associadas à aplicação dessa proposta de maneira indiscriminada nas relações entre clientes e provedores em uma aplicação *Web* SOA, é necessário avaliar a sua aplicação seletiva para grandes aplicações e que efetivamente se caracterizam como de segurança crítica. Esta condição é especialmente importante ao se verificar que a implantação do ASACS requer um conjunto de procedimentos e de investimentos que podem ser excessivos para aplicações pequenas e não caracterizadas como de segurança crítica.

Nesse contexto, deve-se considerar a possibilidade de variáveis restritivas à implantação do ASACS, podendo-se citar os custos de emissão de certificados digitais perante as autoridades certificadoras, custos de fabricação, distribuição e manutenção dos *tokens* de segurança ou ainda a simples condição dos computadores em que, por normas de segurança da organização, é desabilitada qualquer interface de comunicação com dispositivo externo, impossibilitando a utilização dos *tokens* de segurança.

Para viabilizar uma implantação prática do ASACS, deve-se considerar que os consumidores de serviços devem ser empresas, caracterizando um ambiente com administração mais eficiente para execução de procedimentos e cumprimento das políticas de segurança requeridas pelo arcabouço.

Outro aspecto importante das implantações do ASACS é que não necessariamente necessitam ocorrer de maneira indiscriminada e homogeneizada sobre todos os serviços publicados pelas aplicações. É possível separar conjuntos de serviços que requerem a implantação plena das etapas do ASACS enquanto

outros podem ser tratados com uma implantação parcial destas etapas (com redução do nível de segurança) ou simplesmente não requerem mecanismos de segurança. Esta segregação pode ocorrer tanto no nível dos *Web services* quanto no nível de cada operação disponibilizada por um *Web service*.

4.10.2 Discussão de um caso prático para uso do arcabouço

Com o objetivo consolidar uma visão prática de uso do ASACS, esta seção apresenta uma descrição dos passos para a configuração de um caso de uso com implantação das quatro etapas definidas pelo arcabouço: três para validação de legitimidade e uma para validação do comportamento dos consumidores.

Para o papel de uma aplicação prática é considerada uma solução para o mercado de capitais, denominada MCSEG (Mercado de Capitais Seguro). Essa aplicação exemplo tem como principal objetivo viabilizar a comunicação segura entre a bolsa de valores, corretoras e clientes (investidores) das corretoras. A Figura 4.19 representa os principais agentes participantes da MCSEG.

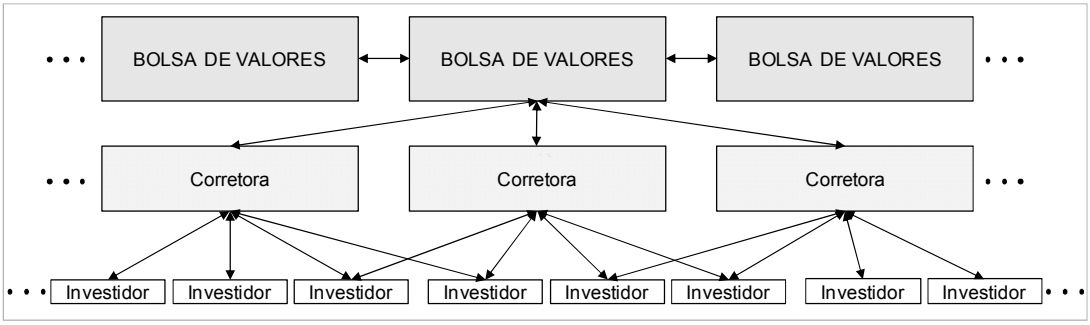

Figura 4.19 - Agentes participantes da aplicação exemplo para o mercado de capitais (MCSEG)

Quanto aos serviços disponibilizados pela rede distribuída que compõe a MCSEG, os mesmos podem ser agrupados pelas relações entre consumidores e provedores. A lista de alguns destes serviços é representada na Tabela 4.1.

Tabela 4.1 - Serviços da MCSEG

Consumidor	Provedor	Exemplos de Serviço
Investidor	Corretora	Consulta de cotações Execução de ordem de compra Execução de ordem de venda

(continua)

(continuação)		
Consumidor	**Provedor**	**Exemplos de Serviço**
Corretora	Bolsa	Consulta de cotações Execução de ordem de compra Execução de ordem de venda
Bolsa	Bolsa	Consulta de índices

Considerando os tipos de serviços disponibilizados nessa rede, é possível organizá-los em dois grupos diferenciados pelo nível de sensibilidade à segurança. No grupo de serviços mais sensíveis enquadram-se as execuções de ordem de compra e venda, enquanto os serviços de consulta podem ser classificados em um nível menor de segurança requerida.

Para implantação do ASACS nessa aplicação exemplo, vamos considerar especificamente as relações entre as corretoras e as bolsas de valores. Essas relações são tipicamente implementadas por conexões de rede (*links*) dedicadas e a utilização do ASACS traz a contribuição de aumentar o nível de segurança ao implantar suas quatro etapas de atestamento das corretoras perante as bolsas de valores. Do ponto de vista de segurança, as transações entre as corretoras e bolsas representam um campo de especial interesse. Trata-se de transações entre bolsas nacionais e estrangeiras e um alto volume de corretoras. Uma determinada bolsa de valores estará exposta a um volume significativamente alto de consumidores (corretoras) em um ambiente de operações financeiras sensível à segurança.

Nessas relações entre corretoras e bolsas de valores, os seguintes passos devem ser considerados visando a implantação das quatro etapas de atestamento definidas pelo ASACS:

- <u>Cadastro das corretoras junto às bolsas de valores</u>

  Neste passo, as bolsas de valores deverão estender seus procedimentos padrão para atender também a validações e registros de aspectos técnicos requeridos pelo arcabouço. Assim, além das verificações dos documentos de identificação, registros legais e outras consultas requeridas por normas do mercado de capitais, também serão requeridos:

  - Inspeções nos computadores das corretoras;

- Registros dos valores *hash* das camadas de inicialização dos computadores que serão utilizados pelas corretoras para comunicação com as bolsas.

- Personalização dos *tokens*

Configuração dos *tokens* de segurança a serem instalados nos computadores das corretoras, o que inclui as gerações de chaves, armazenamento de chaves privadas e públicas e sincronização inicial da geração de chaves simétricas de sessão. Para essa sincronização é pré-requisito a implantação de um sistema, nos servidores das bolsas de valores, para a geração de chaves simétricas válidas a cada intervalo de tempo.

A partir da personalização e da ativação dos *tokens* é possível operar as duas primeiras etapas de atestamento do ASACS ("verificação da integridade das camadas nas estações dos consumidores" e "geração de chaves simétricas de sessão sincronizadas").

- Estruturação de uma infraestrutura de autoridade certificadora

Para atender à demanda requerida para implantação da etapa do ASACS denominada "autenticação via certificado e assinatura digital dos consumidores", é necessária a inserção de uma entidade de confiança dos consumidores (corretoras) e provedores (bolsas de valores). Trata-se da inserção de autoridades certificadoras que deverão ser responsáveis por oferecer uma maneira de verificar as chaves públicas apresentadas por cada uma das partes nas comunicações (corretoras e bolsas).

Alternativamente ao uso de autoridades certificadoras padrão e já reconhecidas nas redes abertas, é possível considerar a instauração de autoridades certificadoras dedicadas e estruturadas especificamente para atender às verificações de chaves públicas entre consumidores e provedores na rede do ambiente do mercado de capitais. A instauração deste tipo de solução para a certificação digital bilateral requerida pelo ASACS pode contribuir para a redução dos custos totais de manutenção do ambiente seguro.

Por fim, cabe ainda referir a implementação das assinaturas digitais bilaterais, garantindo os princípios de irretratabilidade e integridade, ambos

críticos no ambiente da MCSEG, caracterizado pelo alto volume de transações e que totalizam grandes valores financeiros. Pelo lado dos consumidores (corretoras), as assinaturas são processadas com chamadas às funções criptográficas dos *tokens,* enquanto pelo lado dos provedores (bolsas) são executadas através de processamentos criptográficos publicados por componentes implementados com a biblioteca de suporte ao arcabouço.

- Validações de comportamento dos consumidores que se apresentam como corretoras legítimas

  Como etapa final e complementar do ASACS, a implementação do algoritmo para graduação dinâmica da confiança sobre os consumidores (corretoras) contribui para tentar interceptar possibilidades de fraude quando usuários consumidores não legítimos conseguem aprovação pelas etapas anteriores do arcabouço destinadas à validação de legitimidade. Trata-se da possibilidade de ataques em que o invasor tenha conseguido se infiltrar na rede com posse de credenciais que tenha obtido ilicitamente de algum usuário (corretora) legítimo.

  Dados os requisitos de alta velocidade e agilidade requeridas pelas transações na MCSEG, torna-se especialmente importante a implementação do algoritmo com compensação entre eventos suspeitos ou negativos e regulares ou positivos. Através deste tipo de algoritmo é reduzida a necessidade de intervenções externas por administradores de rede para restabelecer uma corretora ao estado de ativa, o que tipicamente provoca grande volume de perdas de negócio.

  Ainda em função da necessidade de velocidade e disponibilidade neste tipo de mercado, não deverão ser poupados investimentos nas técnicas definidas pelo ASACS para segurança e disponibilidade dos agentes de regulação dos graus de confiança, bem como da base com os registros dos graus de confiança dos consumidores (corretoras). Neste ambiente com alta demanda por velocidade das transações, a implementação do recurso de *cache* local nos provedores é também recomendada para reduzir os tempos de consulta dos graus de confiança de um consumidor em um dado instante.

A expansão do uso pleno do ASACS para outras relações representadas na Figura 4.19 como, por exemplo, entre investidores e corretoras pode ser de execução mais difícil pela razão da vasta diversidade das plataformas de execução dos computadores dos investidores. Neste cenário, um desafio especialmente importante diz respeito às atualizações das camadas de inicialização desses computadores. Nesta relação entre investidores e corretoras, considerando a realidade do mercado atual, o controle por atestamento se configura mais viável nas relações de investidores que são representados por empresas, excluindo as pessoas físicas.

Alternativamente, é possível sempre recorrer a implantações parciais do ASACS com a implementação de algumas etapas, embora sob prejuízo do nível de segurança oferecido. Assim, para o universo das relações entre investidores pessoas físicas e corretoras, pode-se considerar simplificações como, por exemplo, a implementação apenas da etapa do ASACS denominada “geração de chaves simétricas de sessão sincronizadas”. Através da distribuição de *tokens* que geram as chaves simétricas de sessão sincronizadamente com os servidores das corretoras já é possível a entrega de um fator adicional de segurança por atestamento. O atestamento neste caso é provido pelo fato de que apenas os detentores do *token* e também da chave privada necessária para as interações com este dispositivo estarão habilitados a transacionarem com as corretoras.

5 TRABALHOS RELACIONADOS E CONTRIBUIÇÕES DESTE TRABALHO

Durante os estudos realizados para o presente trabalho foi identificado um conjunto de textos relacionados à segurança em aplicações SOA e que foram então utilizados como referências para a estruturação das contribuições aqui propostas. Nesta seção são avaliadas estas referências e apresentadas as principais contribuições do presente trabalho.

No artigo “IAPF: A Framework for Enhancing Web Services Security”, Sidharth e Liu (2007) propõem um arcabouço que busca mitigar os riscos à segurança não tratados por padrões como WS-Security, WS-Trust, WS-SecureConversation, WS-Federation, WS-Authorization, and WS-SecurityPolicy. Como exemplo de um ponto comum com o presente trabalho, é apresentado nesse artigo que as aplicações SOA introduzem preocupações à segurança que não eram características das arquiteturas tradicionais. Como exemplo, cita-se o fato que mensagens XML baseadas em SOAP não são bloqueadas por *firewalls*, o que pode levar intrusos a ganhar acessos a sistemas sensíveis utilizando-se das interfaces providas por arquivos WSDL. Denominado *Integrated Application and Protocol Framework* (IAPF), o arcabouço proposto por Sidharth e Liu (2007) é fundamentado em quatro passos. O primeiro passo é orientado a técnicas de proteção contra ataques sobre UDDI, passando para um segundo e terceiro passos focados em proteções contra ataques sobre WSDL e mensagens SOAP. O quarto e ultimo passo endereça técnicas para tratar ataques contra *Web services* disponíveis a um alto volume de consumidores.

No artigo "Architecture framework proposal for dynamic and ubiquitous security in global SOA", Shah e Patel (2009) propõem uma arquitetura para a segurança em aplicações SOA. Em seu trabalho, os autores se fundamentam no uso de interceptores SOA, também denominados programas tratadores, os quais foram considerados como parte da proposta de implementação do ASACS, como descrito na Seção 4.4. Shah e Patel definem a prerrogativa que quando SOA migra para o que o denominam SOA global (onde não é possível ter um conjunto predefinido, pequeno e controlado de consumidores), torna-se necessário implementar um mecanismo de segurança entre consumidores e provedores que opere de modo

dinâmico e ubíquo. Para este objetivo de execução dinâmica dos processamentos de segurança, a proposta dos autores se baseia na instanciação sob demanda dos interceptores (programas tratadores) SOA, conforme parametrização declarativa em arquivos WSDL.

Em “Towards Secure SOAP Message Exchange in a SOA”, Rahaman; Schaad e Rits (2006) apresentam uma solução focada no problema específico de ataques de reescrita de XML em mensagens SOAP. Para resposta a este problema, propõem a utilização de uma estrutura denominada SOAPAccount, descrita na Seção 3.5 deste presente trabalho, a ser incorporada como extensão aos *tags* de cabeçalho de um envelope SOAP. Através deste *tag* adicional, os autores propõem a declaração de informações estruturais da mensagem XML em trânsito como, por exemplo, o número total de elementos de cabeçalho e de corpo da mensagem. Intervenções sobre a estrutura da mensagem resultantes de ataques de reescrita seriam então reveladas por diferenças na estrutura do XML comparativamente às descrições contidas na *tag* SOAPAccount.

Uma diferença essencial entre os trabalhos relacionados e o ASACS é o foco e orientação à aplicação de atestamento dos consumidores como proposta central e fundamental para proteção contra consumidores mal intencionados.

Sobre a utilização da *tag* SOAPAccount proposta por Rahaman; Schaad e Rits (2006), esta técnica foi incorporada à especificação do ASACS, definindo-se sua aplicação na implementação da etapa de atestamento denominada neste trabalho como “autenticação via certificado e assinatura digital dos consumidores”.

Quanto à incorporação das contribuições de Shah e Patel (2009), a utilização de programas tratadores é apresentada neste trabalho como base para a implementação de instâncias do ASACS. Uma diferença importante em relação ao trabalho de Shah e Patel é que os programas tratadores aqui propostos já implementam também as chamadas às interfaces de negócio das operações de um serviço. Ou seja, o lado consumidor simplesmente invoca, pelo programa tratador, a operação que deseja consumir e então o programa tratador coordena a execução das funções de segurança requeridas à mensagem (ex.: criptografia) e invoca a operação ao provedor com a segurança no nível da mensagem de acordo com o padrão WS-Security.

Quanto à possibilidade de incorporação ao ASACS das técnicas definidas nos quatro passos do arcabouço IAPF, definido por Sidharth e Liu (2007), deve-se ponderar o esforço e complexidade de implementação *versus* os níveis de segurança requeridos de cada aplicação. Enquanto o IAPF é baseado em uma abordagem reativa para tentar evitar o êxito de um conjunto conhecido de tipos de ataque, o ASACS busca a segurança preventiva ao limitar o processamento das requisições apenas a consumidores atestados. A implementação do ASACS complementada com a implementação dos passos do IAPF definitivamente leva a um cenário com níveis de segurança mais altos, mas, o custo total requerido deve ser considerado nesta decisão. Trata-se da ponderação entre o esforço para a entrega de segurança e a real necessidade definida pelos requisitos de segurança de uma aplicação em particular.

Como principal contribuição deste trabalho, destaca-se a especificação de uma abordagem estruturada para a segurança em aplicações *Web* SOA, baseada na utilização de atestamento e que resulta na necessidade de maiores esforços para que sejam efetuados ataques aos provedores em uma rede distribuída de serviços. O arcabouço definido para suportar a implementação desta abordagem tem foco na validação de legitimidade dos consumidores, incorporando algumas técnicas de proteção definidas nos trabalhos similares aqui relacionados. Ainda como parte das contribuições do presente trabalho, pode-se citar:

- A exploração das principais características de vulnerabilidade em uma aplicação *Web* SOA, destacando-se o aspecto de exposição dos provedores de serviços a um universo muito maior de consumidores (majoritariamente programas de *software*) quando comparado às arquiteturas tradicionais;
- A apresentação do conceito de atestamento como fator fundamental ao desenvolvimento de aplicações SOA seguras, tendo como foco a proteção dos provedores de serviços através da validação remota da legitimidade dos consumidores;
- A introdução de aspectos para implantação do arcabouço, descrevendo atributos da implementação das quatro etapas de atestamento definidas, além da apresentação de um conjunto inicial de requisitos ao desenvolvimento de uma biblioteca de suporte à implementação destas etapas;

- Incorporação da análise de comportamento dos consumidores como base para o cálculo do grau de confiança sobre estes consumidores, interferindo na decisão entre processar ou recusar suas solicitações perante uma comunidade de provedores de serviço.

6 CONCLUSÕES

Em um cenário de constantes e crescentes ameaças à segurança das aplicações de *software*, as aplicações SOA são especialmente vulneráveis em função de um conjunto de características do modelo de soluções orientadas a processos e serviços. Entre estas características, destaca-se a publicação das interfaces de serviços a consumidores que tipicamente não são humanos e sim programas com capacidade de computação das requisições em quantidades e freqüências substancialmente maiores que nas arquiteturas tradicionais.

Mesmo com o suporte de um extenso conjunto de técnicas de proteção às vulnerabilidades conhecidas, a segurança das aplicações ainda deve ser objeto de intensa atenção no desenvolvimento de *software*. Trata-se de uma área de pesquisa muito fértil, com grande potencial de crescimento ao passo em que novas tecnologias inserem novas vulnerabilidades, as quais requerem novos métodos de proteção.

Especialmente a partir da revolução provocada pela *Internet* e mais recentemente com SOA, é também verdade que os métodos de proteção têm provado muitas vezes um curto ciclo de vida em função de novas técnicas criminosas que os tornam não mais efetivos. Trata-se de uma batalha de longa perspectiva nos mercados de segurança crítica e que dependem de sistemas e aplicações computacionais distribuídas. Nesta batalha pela proteção das aplicações, considerando-se o cenário tecnológico atual e possivelmente dos próximos anos, a expectativa é que o avanço da tecnologia de computação ocorra de um modo ainda muito fortemente estabelecido sobre o paradigma de distribuição e compartilhamento de serviços.

Sob estas condições, uma abordagem que tem demonstrado importante potencial ao fortalecimento da segurança na *Web* e, mais notadamente sobre as aplicações *Web* SOA, é fundamentada na implantação de atestamento dos consumidores de serviços. Enquanto as aplicações usualmente implementam a validação da autenticidade dos servidores, como é o caso típico das soluções de *e-commerce* sobre o protocolo SSL, resta um vasto de campo de investimento à segurança através de mecanismos que validem também a legitimidade dos clientes

consumidores e seus dispositivos de acesso. Por validação de legitimidade dos consumidores deve-se entender um conceito estendido que incorpora diversos elementos de certificação e graduação da confiança destes consumidores perante os provedores de serviços. Conforme definido no decorrer deste trabalho, trata-se do conceito de atestamento que foi então utilizado como base para a proposta de um arcabouço para desenvolvimento de aplicações *Web* SOA seguras.

O arcabouço proposto (ASACS) traz a contribuição de organizar a aplicação de importantes etapas de atestamento dos consumidores em uma rede distribuída de serviços e deverá ser objeto de trabalhos futuros para sua implementação prática.

As etapas de atestamento selecionadas para o ASACS caracterizam-se por validações do ambiente de execução nas estações dos clientes, checagem de chaves simétricas geradas por dispositivos de *hardware* dedicados, assinatura e certificação digital dos consumidores e ainda uma monitoração do comportamento dos consumidores na rede com atribuição de uma graduação de confiança computada como parte do atestamento de cada consumidor.

Uma discussão do uso do ASACS em uma aplicação prática foi apresentada neste trabalho, contribuindo para confrontar a viabilidade de sua aplicação e a conveniência de sua implementação. Nessa discussão foram mapeadas necessidades reais de adaptação e simplificações para os diferentes tipos de interação entre consumidores e provedores de uma rede para operações no mercado de capitais. Também durante a exploração dos aspectos de implementação do ASACS para essa aplicação exemplo foram identificados fatores que reforçaram a necessidade da implementação integral de algumas soluções definidas neste trabalho em favor da disponibilidade e velocidade das transações. Como exemplos dessas soluções cujo investimento não deve ser poupado para o caso da aplicação exemplo, citam-se as técnicas para a segurança e alta disponibilidade do agente de regulação dos graus de confiança, bem como da base com os registros das informações de confiança sobre os consumidores.

O ASACS representa uma abordagem para controle e filtro dos acessos aos serviços de uma rede distribuída, demonstrando-se como uma importante ferramenta para proteção de aplicações *Web* SOA. A utilização do arcabouço traz um importante reforço à segurança ao buscar a negação de acesso a consumidores mal intencionados, não legítimos ou que tenham sido alvo de ataque. Através do

arcabouço, mesmo que haja falhas de segurança em função de programação inadequada dos serviços nos provedores, ainda assim apenas os consumidores atestados terão suas requisições processadas, reduzindo a probabilidade das invasões e aumentando o esforço dos ataques. É ainda importante reiterar que a implantação do arcabouço não deve implicar na desativação de técnicas de segurança complementares, devendo-se, todavia, sempre ponderar os investimentos e esforços para proteção em face aos requisitos de segurança de cada aplicação em particular.

7 DESAFIOS PARA TRABALHOS FUTUROS

Para a extensão do presente trabalho é possível identificar um conjunto de desafios que vão desde a implementação do *token* de segurança até a codificação de um mecanismo ágil para manter as validações de integridade das estações dos consumidores diante das freqüentes atualizações nas camadas de inicialização destas estações.

Como um próximo passo é sugerida a implementação de uma instância do ASACS e sua aplicação prática para uma aplicação *Web* SOA, com foco à solução de importantes desafios:

- Adequada ponderação entre os requisitos de segurança da aplicação *Web* SOA e os elementos de segurança e atestamento a serem providos pela instância do ASACS;
- Solução para operacionalizar a autenticação digital pelo lado cliente, uma vez que o volume de consumidores pode ser alto o suficiente para dificultar a aplicação de um modelo padrão de emissão de certificado (variáveis de custo financeiro e de gerenciamento destes certificados). Este desafio consiste em encontrar uma solução que não comprometa a escalabilidade, suporte operacional e desempenho (FAW; 2004);
- Resposta ao risco de sobrecarga de processamento em conseqüência das validações bilaterais de assinaturas e certificados digitais e também em função das validações dinâmicas, pelo lado do provedor, dos graus de confiança dos consumidores;
- Desenvolvimento do dispositivo de *hardware* (*token* de segurança) a ser acoplado às estações dos consumidores e que é parte da solução para implementar o ASACS;
- Resposta ao problema da natureza dinâmica das camadas de inicialização nas estações dos consumidores, que podem ser alteradas por uma simples atualização de sistema operacional, gerando um obstáculo à validação de seqüência de assinaturas sobre estas camadas;

- Definição e implementação de um algoritmo eficiente para o gerenciamento, atualizações e validações dinâmicas das graduações de confiança de cada consumidor da rede da aplicação *Web* SOA a ser desenvolvida com suporte do ASACS.

REFERÊNCIAS

ABUOSBA, K. A.; EL-SHEIKH, A. A. Formalizing Service-Oriented Architectures. IEEE Computer Society, IT Professional jul./ago. 2008. v. 10, issue 4, p. 34-38.

CABRERA, L. F.; KURT, C. Web Services Architecture and Its Specifications: Essentials for Understanding WS-*.1ª ed. Redmond: Microsoft Press, 2005. 174 p. Apêndice C. ISBN: 9780735621626.

CHESWICK, W., BOLLOVIN, S. Firewalls and Internet Security: Repelling the Wily Hacker. 2ª ed. Estados Unidos: Addison-Wesley, 1994. ISBN: 020163466X.

FAW, K. Securing Web Services, Part I. In: BORLAND CONFERENCE, 2004. Disponível em: <http://conferences.embarcadero.com/article/32136>. Acesso em: 17 set. 2009.

GANCI, J.; ACHARYA, A.; ADAMS, J.; EUSEBIO, P. D.; RAHI, G.; STRACHAN, D.; UTSUMI, K.; NORITOSHI, W. Patterns: SOA Foundation Service Creation Scenario, IBM Redbook SG24-7240-00. IBM Redbook, set. 2006. Disponível em: <http://www.redbooks.ibm.com/abstracts/sg247240.html?Open>. Acesso em: 25 set. 2009.

GHODSI, A. Distributed k-ary System: Algorithms for Distributed Hash Tables. 2006. 193 p. Tese (Doutorado) - The Royal Institute of Technology (KTH), School of Information and Communication Technology, Department of Electronic, Computer, and Software Systems, Estocolmo, 2006. Capítulo 1.

GUDGIN, M. Web Services Technical Articles: Using WS-Trust and WS-SecureConversation, 2004. Microsoft Corporation, maio 2004. Disponível em: <http://msdn.microsoft.com/en-us/library/ms996521.aspx>. Acesso em: 26 set. 2009.

GUTIÉRREZ, C.; MEDINA, E. F.; PIATTINI, M. A Survey of Web Services Security. In: COMPUTATIONAL SCIENCE AND ITS APPLICATIONS - ICCSA, 2004, Assisi - Italy. Proceedings, v. 3043/2004, p. 968-977.

GUTIÉRREZ, C.; MEDINA, E. F.; PIATTINI, M. Web Services Enterprise Security Architecture: A Case Study. In: WORKSHOP ON SECURE WEB SERVICES (SWS'05), nov. 2005, Fairfax - Virginia - USA. Proceedings, Nova Iorque: ACM, 2005, p. 10-19.

HARRIS, S. All in One, CISSP Exam Guide. 4ª ed. Estados Unidos: McGraw-Hill, 2008. 1145 p. Capítulos 4 e 8. ISBN: 9780071497879.

HOWARD, J. D. An Analysis of Security Incidents on the Internet 1989-1995. 1997. 322 p. Tese (Doutorado) - Carnegie Mellon University, Pittsburg, 1997. p. 43-52.

JOSANG, A. Trust-Based Decision Making For Electronic Transactions. In: NORDIC WORKSHOP ON SECURE COMPUTER SYSTEMS (NORDSEC'99), 4., 1999, Sweden. Proceedings, Stockholm University, report 99-005, p. 496-502.

MILANOVIC, N. Asynchronicity and Loose Coupling in Service Oriented Architectures. In: WORKSHOP OF THE HPI RESEARCH SCHOOL ON SERVICE-ORIENTED SYSTEMS ENGINEERING, 2006. Proceedings, Seção 5.

MITTELSDORF, A. W. Uma plataforma para computação com confiança baseada em monitor de máquinas virtuais e atestamento dinâmico. 2004. 128 p. Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2004.

OASIS. Reference Model for Service Oriented Architecture 1.0, public review draft 2, 2006. Disponível em: <http://www.oasis-open.org/committees/download.php/18486/pr-2changes.pdf>. Acesso em: 04 jan. 2010.

_____. WS-SecureConversation 1.3, OASIS Standard, March 7th, 2007. Disponível em: <http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html> . Acesso em: 26 set. 2009.

_____. WSS: SOAP Message Security (WS-Security 2004), OASIS Standard 200401. Disponível em: <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>. Acesso em: 15 ago. 2009.

OWASP. Open Web Application Security Project, 2007. Disponível em: <http://www.owasp.org/index.php/Main_Page>. Acesso em: 17 set. 2009.

PERIORELLIS, P. Securing Web Services: Practical Usage of Standards and Specifications.1ª ed. Estados Unidos: Information Science Reference, 2008. 420 p. Capítulo 1. ISBN: 9781599046396.

RAHAMAN, M. A.; SCHAAD, A.; RITS, M. Towards Secure SOAP Message Exchange in a SOA. In: ACM CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY: WORKSHOP ON SECURE WEB SERVICES, 3., 2006, Alexandria. Proceedings, Nova Iorque: ACM, 2006, p. 77-84.

SEELY, S. Seely. XML and Web Services Security: Understanding WS-Security. Microsoft Corporation, oct. 2002. Disponível em: <http://msdn.microsoft.com/en-us/library/ms977327.aspx>. Acesso em: 25 jun. 2009.

SHAH, D.; MANGAL, A.; AGARWAL, M.; MEHRA, M; DAVE, T.; PATEL, D. Protecting Global SOA from DoS and Other Security Threats. In: INTERNATIONAL CONFERENCE AND WORKSHOPS - ISA, 3., 2009, Seoul - Korea. Proceedings, v. 5576/2009, p. 652-661.

SHAH, D.; PATEL, D. Architecture framework proposal for dynamic and ubiquitous security in global SOA. International Journal of Computer Science and Applications, v. 6, n. 1, p. 40- 52, 2009. Technomathematics Research Foundation.

SHAH, S. Web Services - Attacks and Defense, Information Gathering Methods: Footprints, Discovery & Fingerprints. Net Square Corporation, 2002. Disponível em: <http://www.net-square.com/whitepapers/WebServices_Info_Gathering.pdf>. Acesso em: 28 jan. 2010.

SIDHARTH, N.; LIU, J. IAPF: A Framework for Enhancing Web Services Security. In: ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE (COMPSAC), 31., 2007, Beijin. Proceedings, Washington: IEEE Computer Society, 2007, v. 1, p. 23-30.

SINGHAL, A.; WINOGRAD, T.; SCARFONE, K. Guide to Secure Web Services: Recommendations of the National Institute of Standards and Technology. NIST Special Publication 800-95. Gaithersburg: Computer Security Division, 2007. 128 p.

THOMPSON, S. Implementing WS-Security. IBM Corporation, abr. 2003. Disponível em: <http://www.ibm.com/developerworks/webservices/library/ws-security.html>. Acesso em: 01 fev. 2010.

WONGTSCHOWSKI, A. Segurança em Aplicações Transacionais na Internet: O Elo Mais Fraco. 2005. 108 p. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2005.

REFERÊNCIAS COMPLEMENTARES

EPSTEIN, J.; SCOTT, M.; MACGRAW, G. Software Security and SOA: Danger, Will Robinson!. IEEE Computer Society, IEEE Security and Privacy, jan./fev. 2006, v. 4, n. 1, p. 80-83. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01588834>. Acesso em: 15 jun. 2009.

LUCCA, G. A. di; FASOLINO, A. R.; MASTROIANNI, M.; TRAMONTANA, P. Identifying Cross Site Scripting Vulnerabilities in Web Applications. In: IEEE INTERNATIONAL WORKSHOP ON WEB SITE EVOLUTION (WSE'04), 6., 2004, Chicago. Proceedings, Washington: IEEE Computer Society, 2004, p. 71-80.

O' BRIEN, L.; MERSON, P.; BASS, L. Quality Attributes for Service-Oriented Architectures. In: INTERNATIONAL WORKSHOP ON SYSTEMS DEVELOPMENT IN SOA ENVIRONMENTS (SDSOA'07), ICSE Workshops, 2007, Minneapolis. Proceedings, Washington: IEEE Computer Society, 2007.

O'NEILL; M. Web Services Security. 1ª ed. Estados Unidos: McGraw-Hill, 2003. 312 p. ISBN: 9780072224719.

SINGHAL, A. Web Services Security: Challenges and Techniques. In: IEEE INTERNATIONAL WORKSHOP ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS (POLICY'07), 8., 2007, Bologna. Proceedings, Washington: IEEE Computer Society, 2007.

APÊNDICE A - Proteção em aplicações *Web* tradicionais

Em resposta às vulnerabilidades apresentadas na Seção 3.3, há um conjunto de técnicas e recomendações de proteção da OWASP (OWASP, 2007) que vão desde orientações conceituais a práticas de programação e alguns exemplos específicos na linguagem Java.

A Tabela A.1 resume estas técnicas e recomendações que devem ser objeto de atenção em todo o ciclo de engenharia de *software*, contribuindo para a aplicação das boas práticas de segurança. Como pode ser observado na Tabela A.1, grande parte das técnicas de proteção dependem de uma abordagem baseada na implementação da segurança de "dentro para fora". Também conhecida como "*built in security*", refere-se à atenção ao desenvolvimento de aplicações seguras durante todas as fases do ciclo de engenharia de *software*, desde as primeiras atividades de planejamento e análise. Esta abordagem visa evitar as vulnerabilidades que não podem ser tratadas por uma camada de segurança periférica a ser considerada nas fases finais do desenvolvimento e que implemente os princípios tradicionais como identificação, autenticação, autorização, entre outros. Como referido na Seção 3.3, trata-se de vulnerabilidades que são alvo de ataques que atravessam esta segurança ao redor das aplicações, explorando falhas das atividades de desenho e programação. Um processo estruturado para integrar os aspectos de segurança a todas as etapas do ciclo de vida de desenvolvimento de *softwares* baseados em *Web services* é proposto por Gutiérrez; Medina e Piattini (2005), cujo trabalho define um guia para executar as fases de mapeamento de requisitos, desenho de arquitetura e seleção das tecnologias e padrões de segurança.

Tabela A.1 - Técnicas de proteção a vulnerabilidades de aplicações *Web* tradicionais (OWASP, 2007)

Vulnerabilidade	Técnicas de Proteção
1. Cross Site Scripting (XSS)	▪ Mecanismo padrão de validação de entrada (tamanho, tipo, sintaxe e regras de negócio) antes que seja mostrada ou armazenada ▪ Forte codificação de saída, garantindo que qualquer dado de entrada seja codificado antes da "renderização" nas telas da aplicação ▪ Não utilização de "listas negras" (procura e troca de poucos caracteres de risco como "<" e ">"), pois os ofensores têm frequentemente explorado com sucesso esta restrição parcial ▪ Em J2EE, o mecanismo de saída <bean: write...> contribui para esta proteção

(continua)

(continuação)

Vulnerabilidade	Técnicas de Proteção
2. Falhas de Injeção	▪ Mecanismo padrão de validação de entrada (tamanho, tipo, sintaxe e regras de negócio) antes que seja mostrada ou armazenada ▪ Não utilização de abordagem de sanitização de dados potencialmente hostis ▪ Utilização de APIs de *query* fortemente "tipadas" com marcadores substituição ▪ Atenção ao utilizar *stored procedures* uma vez que, embora sejam geralmente seguras contra falhas de injeção, são suscetíveis a ataques por meio de comandos como *exec*() ou concatenação de argumentos ▪ Em J2EE, o objeto *PreparedStatement* fortemente "tipado" contribui para esta proteção
3. Vazamento de Informações e Tratamento de Erros Inapropriado	▪ Inibição do detalhamento na exibição de erros (não exibição de informações de debug, pilha ou caminhos para usuários finais) ▪ Checagem e configuração prévia de erros de todas as camadas (Web Server, Aplication Server, Banco de Dados, etc.) para prevenir que mensagens de erro sejam exploradas ▪ Criação de um manipulador de erros padrão que retorne uma mensagem de erro já "sanitizada" para a maioria dos usuários
4. Execução Maliciosa de Arquivo	▪ Mapeamento indireto a objetos de referência por meio de apresentação de identificadores "*hashed*" ao invés de seus valores puros ▪ Regras de *firewall* para evitar conexões indevidas a Web sites externos ou sistemas internos ▪ Mecanismos de *sandbox* como virtualização para isolar aplicações umas das outras ▪ Em J2EE, certificar que o gerenciador de segurança está habilitado e configurado adequadamente, conferindo as permissões mínimas requeridas à aplicação
5. Referência Insegura Direta a Objeto	▪ Não exposição (sempre que possível) de referências de objetos privados como chaves primárias e nomes de arquivos ▪ Verificação da autorização de acesso a todos os objetos referenciados
6. Falha ao Restringir Acesso a URLs	▪ Implementação de controle de acesso efetivo que garanta que todas *URLs* são protegidas por um mecanismo que verifique não apenas o passo inicial de um processo (como a mera inibição de opções de menu, o que pode ser facilmente descoberto pelo invasor) ▪ Bloqueio do acesso a arquivos de extensões cuja execução não é requisito da aplicação
7. Cross Site Request Forgery (CSRF)	▪ Eliminação prévia de vulnerabilidades XSS (que podem ser utilizadas como um passo anterior para acesso não autorizado a credenciais) ▪ Uso de *tokens* randômicos em todos os formulários e URLs (exemplo: <form action="/transfer.do" method="post"> <input type="hidden" name="8438927730" value="43847384383">…). O *token* submetido deve então ser verificado para o usuário corrente ▪ Re-autenticação, assinatura de transação, tokens por telefone e e-mail para confirmação ▪ Não utilização de requisições GET para dados sensíveis

(continua)

(conclusão)

Vulnerabilidade	Técnicas de Proteção
8. Falhas de Autenticação e Gerência de Sessão	▪ Uso apenas de mecanismos padrão de gerenciamento de sessão ▪ Limitação ao uso de *cookies* ▪ A função de *logout* deve destruir todas as sessões e a aplicação não deve perguntar por confirmações de *logout* quando o usuário o solicita explicitamente ou via fechamento da janela ou aba ativa
9. Armazenamento Criptográfico Inseguro	▪ Utilização apenas de algoritmos de criptografia aprovados publicamente ▪ Armazenamento de chaves privadas e credenciais com extremo cuidado ▪ Garantia que dados armazenados criptografados no disco não são fáceis de decriptografar (por exemplo, criptografia de banco de dados tem pouco valor se a conexão de banco de dados permite acessos não criptografados)
10. Comunicações Inseguras	▪ Uso de SSL para todas as conexões ▪ Comunicações entre os elementos da infraestrutura, como servidores *Web* e sistemas de banco de dados devidamente seguras