

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE COMPUTAÇÃO

**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA
COMPUTAÇÃO**

FLÁVIO DE OLIVEIRA SILVA

**CONTROLE DE ACESSO A *WEB SERVICES* BASEADO EM UM
PROTOCOLO DE AUTENTICAÇÃO SEGURA**

UBERLÂNDIA, MG

2004

FICHA CATALOGRÁFICA

Elaborado pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

S586c Silva, Flávio de Oliveira, 1970-

Controle de acesso a Web services baseado em um protocolo de autenticação segura / Flávio de Oliveira Silva. - Uberlândia, 2004.

93f. : il.

Orientador: Pedro Frosi Rosa.

Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.

Inclui bibliografia.

1. Serviços na Web - Teses. 2. Computadores - Controle de acesso - Teses. 3. Redes de computação - Medidas de segurança - Teses. I. Rosa, Pedro Frosi. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3.02 (043.3)

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE COMPUTAÇÃO

**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA
COMPUTAÇÃO**

FLÁVIO DE OLIVEIRA SILVA

**CONTROLE DE ACESSO A *WEB SERVICES* BASEADO EM UM
PROTOCOLO DE AUTENTICAÇÃO SEGURA**

Dissertação apresentada ao programa de Pós Graduação da Faculdade de Computação da Universidade Federal de Uberlândia como exigência parcial para obtenção do título de Mestre, sob a orientação do Prof. Dr. Pedro Frosi Rosa.

UBERLÂNDIA, MG

2004

FLÁVIO DE OLIVEIRA SILVA

**CONTROLE DE ACESSO A *WEB SERVICES* BASEADO EM UM
PROTOCOLO DE AUTENTICAÇÃO SEGURA**

Dissertação apresentada ao programa de Pós Graduação da Faculdade de Computação da Universidade Federal de Uberlândia como exigência parcial para obtenção do título de Mestre, sob a orientação do Prof. Dr. Pedro Frosi Rosa.

BANCA EXAMINADORA

PROF. DR. PEDRO FROSI ROSA
Universidade Federal de Uberlândia

PROF. DR. LUÍS FERNANDO FAINA
Universidade Federal de Uberlândia

PROF. DR. JOSÉ GONÇALVES PEREIRA FILHO
Universidade Federal do Espírito Santo

UBERLÂNDIA, 15 DE DEZEMBRO DE 2004.

Dedico este trabalho à memória de João
Roberto Horta, antes de tudo, um grande
amigo e incentivador.

A meu Pai, de quem herdei a garra e a
disposição para trabalhar.

E a Elyse uma legítima representante de
nosso futuro...

AGRADECIMENTOS

Agradeço a Deus, pelos dons da vida e da inteligência... E por ser o meu auxílio em todos os momentos de minha vida... E pela possibilidade de realizar mais este projeto.

À minha esposa Andréa, pela compreensão, por seu carinho e apoio e por esta pessoa tão bonita, tão especial...

Ao meu Pai, Geraldo. Que nunca se cansa de me apoiar e auxiliar, sempre com a maior boa vontade, um exemplo de doação...

À minha mãe, Célia. De quem herdei muitas qualidades. Agradeço sua vida de dedicação, suas orações e por sua presença neste momento tão especial...

À Mariana, minha querida sogra. Um exemplo de bondade e de dedicação e um contra-exemplo do estigma da sogra...

Ao meu cunhado, Roberto Horta. Uma pessoa íntegra, honesta e companheiro, o irmão que não tive...

Durante a realização deste trabalho, tive a felicidade de fazer parte do corpo docente da Faculdade de Computação da Universidade Federal de Uberlândia, durante este período pude manter um ótimo relacionamento com todos: professores, funcionários e alunos. Com certeza fui agraciado com novas amizades.

Agradeço ao Professor Sérgio de Mello Schneider. Um dos primeiros a me receber na graduação e o primeiro a me receber para a realização deste trabalho, me incentivando e guiando durante a volta ao meio acadêmico.

Ao professor Stéphane Júlia, pelo choque de realidade... E pela reciclagem. *Merci!*

Ao Professor Luís Fernando Faina, um de exemplo de dedicação ao seu trabalho e à sua profissão. Com certeza, aquele que me inspirou na escolha da linha de pesquisa...

Ao Professor João Nunes, pela oportunidade de trabalho em conjunto, ainda que por pouco tempo e por seus ensinamentos sempre mais amplos que a ementa.

Ao Professor Autran Macêdo, professor durante a graduação, coordenador de Estágio Supervisionado e um amigo nos dias atuais.

Ao Professor Anilton Joaquim da Silva, um companheiro, amigo e apoiador nas atividades de docência.

Ao Professor Ilmério Reis da Silva, pela oportunidade de trabalho em conjunto e pela confiança recebida.

Ao Professor Carlos Roberto Lopes, companheiro na docência e de trabalho nos finais de semana...

Ao Professor Jamil Salem Barbar, pelo apoio operacional, pela disponibilidade e pelo incentivo à paternidade...

À Professora Rita Maria da Silva Júlia, pelo apoio e pelas informações administrativas, sempre que necessário.

Ao Professor Mauro Hemerly Gazzani, que acreditou na realização deste trabalho. Obrigado pelo apoio e incentivo!

Ao Professor João Augusto, cujas aulas durante a graduação foram muito importantes para a minha inserção no mundo dos negócios... Durante o mestrado, um colega de trabalho, um companheiro de sala... Agradeço sua amizade e sua presença.

Agradeço ao Professor, orientador e amigo Pedro Frosi Rosa. Sua presença, seu apoio, nossas trocas de idéias durante o café... Agradeço sua disponibilidade, sua integridade como pessoa e pela inspiração durante toda a execução deste trabalho. Obrigado pela oportunidade de trabalho em conjunto.

Finalmente gostaria de agradecer a todas as pessoas, com as quais pude conviver: Neusa, André, Mello, Marcelo, Ademir, Dante, Cynthia, Célia, Libéria, Denise, Sandra, Márcia, Cláudio, Izabela, Marcelo Rodrigues, Maria Amélia, Bacalá, Márcio e tantas outras mais, tanto no plano pessoal como profissional.

A realização pessoal nunca é solitária... O coração é bem mais amplo que o espaço para a escrita... Fazemos parte de um todo e, portanto, agradeço a todos sem distinção.

Muito obrigado!

**“Confia a Deus os teus projetos e eles se
realizarão”**

Pr 16,3

RESUMO

A computação vem revolucionando a vida de pessoas e de empresas. A Internet é seguramente um fator que tem propiciado esta presença ubíqua da computação em todos os processos do dia-a-dia. Os *Web Services* oferecem uma nova forma para o uso da Internet, onde aplicações remotas podem se comunicar utilizando padrões que garantam a interoperabilidade destes sistemas. Através desta abordagem, a *Web* pode oferecer não somente informações, mas uma nova maneira de se obter a computação de forma distribuída. Neste ambiente, segurança é um aspecto crítico e o modelo de segurança para os *Web Services* ainda não está completamente definido. O controle de acesso é uma forma de garantir que apenas usuários, ou sistemas, devidamente reconhecidos possam ter acesso a serviços e informações. Um meio amplamente utilizado para efetuar este controle é o uso de senhas. Entretanto, na maioria dos sistemas utilizados atualmente, este controle apresenta deficiências e um intruso devidamente preparado e mal intencionado pode quebrar este controle. Este trabalho propõe e implementa uma solução para o controle de autenticação em *Web Services*. Esta solução é baseada no protocolo SRP (*Secure Remote Password*), um protocolo de autenticação seguro através do uso de senhas, que é estendido a fim de oferecer o controle de acesso como uma camada adicional. Esta camada pode ser implementada em qualquer plataforma disponível para a construção de *Web Services*. Neste trabalho foi utilizado o Apache AXIS.

Palavras-chave: Segurança; Autenticação; *Web Services*; XML; Protocolos.

ABSTRACT

Computing is causing a revolution in peoples and business life. Internet is for sure the reason why this ubiquitous computing is present in all almost all daily processes. Web Services offers a new way to use Internet, where remote applications can communicate each other using interoperable standards. Through this approach the Web can be the source not of only information, but distributed computing. In this environment security is a critical issue and the security model to Web Services is not fully defined. Access control is a way to guarantee that only the users, or systems, that are fully recognized can access services and information. Passwords are a spread way used to accomplish this. Nevertheless, most systems currently used to perform this control shows deficiencies and an intruder can break it down. This work presents and deploys a solution for the Web Services authentication control. This solution is based on SRP (Secure Remote Password), a strong-based password authentication protocol. To accomplish this, the protocol has to be extended in order to offer authentication control as an independent layer. This additional independent layer can be built on any available Web Service platform. In this work we have used Apache AXIS for this end.

Keywords: Security; Authentication; Web Services; XML; Protocols.

SUMÁRIO

Resumo	viii
Abstract	ix
Sumário	x
Lista de Figuras	xii
Lista de Tabelas	xiii
Lista de Acrônimos.....	xiv
1. Introdução	1
2. <i>Web Services</i>	4
2.1 Exemplos de Utilização	6
2.2 Arquitetura Orientada a Serviços (SOA).....	10
2.3 A Arquitetura dos <i>Web Services</i>	11
2.3.1 Camada de Transporte	14
2.3.2 Camada XML	14
2.3.3 Camada de Mensagens	15
2.3.3 Camada de Segurança.....	15
2.3.4 Camada de Mensagens Confiáveis	16
2.3.5 Camada de Transações e Coordenação	16
2.3.6 Camada de Coreografia e Orquestração	17
2.3.7 Camada de Domínios Específicos	18
2.3.8 Camada de Metadados	18
2.4 A Linguagem XML	18
2.4.1 Namespaces	19
2.4.2 Esquemas XML	20
2.5 O Protocolo SOAP	21
2.5.1 Estrutura da mensagem SOAP	21
2.5.2 SOAP <i>Envelope</i>	23
2.5.3 SOAP <i>Header</i>	23
2.5.4 SOAP <i>Body</i>	25
2.5.5 SOAP <i>Fault</i>	25
2.6 A Linguagem WSDL	27
2.6.1 Definição dos Tipos.....	28
2.6.2 Definição das Mensagens	28
2.6.3 Definição das interfaces	29
2.6.4 Definição das ligações	31
2.6.4 Definição do Serviço	32
3. Segurança em <i>Web Services</i>	33
3.1 Integridade	34
3.2 Confidencialidade	35
3.3 Autenticação	36
3.4 Autorização	38
3.5 Não-Repúdio.....	38

3.6 <i>Web Services Security (WS-Security)</i>	39
3.7 A Linguagem SAML	41
3.8 O Protocolo <i>Secure Remote Password</i> (SRP)	42
3.9 O Funcionamento do protocolo <i>Secure Remote Password</i> (SRP)	44
4. Autenticação de Acesso em <i>Web Services</i>	47
4.1 O Apache AXIS.....	49
4.2 A Solução Proposta.....	51
5. Implementação e Análise dos Resultados Obtidos	59
5.1. Implementação e Implantação da solução.....	59
5.2. Testes da solução proposta.....	61
5.3. Tempo de Latência	63
6. Conclusão	67
7. Referências Bibliográficas	70
Apêndice a - Padrões utilizados em <i>Web Services</i>	77

LISTA DE FIGURAS

Figura 1 - Aplicação utilizando <i>Web Services</i> .	7
Figura 2 - <i>Web Services</i> em um ambiente B2B.	8
Figura 3 - <i>Web Services</i> em transações complexas.	9
Figura 4 - Arquitetura SOA.	10
Figura 5 - Arquitetura dos <i>Web Services</i> .	13
Figura 6 - Exemplo de documento XML.	19
Figura 7 - Documento XML utilizando <i>namespaces</i> .	20
Figura 8 - Visão parcial do esquema XML de um documento.	21
Figura 9 - Estrutura de uma mensagem SOAP.	22
Figura 10 - Exemplo de uma mensagem SOAP.	22
Figura 11 - Exemplos de valores para o atributo <i>encodingStyle</i> .	23
Figura 12 - Valores padrão para o atributo <i>role</i> .	24
Figura 13 - Exemplo de Mensagem SOAP com elemento Fault.	26
Figura 14 - Estrutura de um documento WSDL.	27
Figura 15 - Exemplo de tipo complexo utilizando XML esquema.	28
Figura 16 - Exemplo de mensagens definidas em um documento WSDL.	29
Figura 17 - Exemplo de interface definida em um documento WSDL.	30
Figura 18 - Elemento de ligação sobre SOAP em um documento WSDL.	31
Figura 19 - Definição do Serviço descrito em um documento WSDL.	32
Figura 20 - Comunicação fim a fim versus comunicação ponto a ponto.	33
Figura 21 - Exemplo de uso de SSL / TLS em uma comunicação fim a fim.	35
Figura 22 - Confidencialidade parcial da mensagem.	36
Figura 23 - Autenticação única em vários <i>Web Services</i> .	37
Figura 24 - Exemplo de mensagem SOAP utilizando <i>WS-Security</i> .	40
Figura 25 - Cálculo do verificador no protocolo SRP.	44
Figura 26 - Troca de mensagens do protocolo SRP.	45
Figura 27 - Arquitetura do Apache AXIS.	50
Figura 28 - Solução proposta para autenticação segura.	51
Figura 29 - SOAP <i>header</i> adicionado à mensagem.	51
Figura 30 - <i>SrpHandler</i> adicionado a cadeia global do AXIS.	52
Figura 31 - Troca de mensagens do protocolo SRP como chamadas a <i>Web Services</i> .	54
Figura 32 - WSDL com a definição do serviço de autenticação utilizando SRP (a).	55
Figura 33 - WSDL com a definição do serviço de autenticação utilizando SRP (b).	56
Figura 34 - WSDL com a definição do serviço de autenticação utilizando SRP (c).	57
Figura 35 - Arquitetura utilizada na implantação.	60
Figura 36 - Mensagem GetSalt (pedido).	62
Figura 37 - Mensagem GetSalt (resposta).	62
Figura 38 - Mensagem setPublicKeyA para envio da chave pública temporária.	62
Figura 39 - Resposta da mensagem getPublicKeyB.	63
Figura 40 - Resposta da mensagem getRandomU.	63
Figura 41 - Envio do valor de comparação através da mensagem setMatch1.	63
Figura 42 - Tempo de latência em milissegundos.	64
Figura 43 - Tempo de latência em milissegundos com outras autenticações.	65

LISTA DE TABELAS

Tabela 1 - Códigos de erro SOAP.	25
Tabela 2 - Tipos de operação que podem ser definidos em um documento WSDL 1.1.	30
Tabela 3 - Comparação entre os protocolos de autenticação.....	42
Tabela 4 - Tecnologias de autenticação utilizadas por alguns processadores SOAP.....	48
Tabela 5 - Informação armazenada sobre uma senha.....	61

LISTA DE ACRÔNIMOS

API	<i>Application Programming Interface</i>
AXIS	<i>Apache Extensible Interaction System</i>
B2B	<i>Business-to-Business</i>
<i>E-Commerce</i>	<i>Electronic Commerce</i>
<i>E-Business</i>	<i>Electronic Business</i>
<i>E-Government</i>	<i>Electronic Government</i>
<i>E-Procurement</i>	<i>Electronic Procurement</i>
<i>e-WSTRAVEL</i>	Agência de turismo virtual baseada em <i>Web Services</i>
FTP	<i>File Transfer Protocol</i>
HTTP	<i>HyperText Transfer Protocol</i>
HTTPS	<i>Secure HyperText Transfer Protocol</i>
J2EE	<i>Java 2 Platform, Enterprise Edition</i>
JWS DP	<i>Java Web Services Developer Pack</i>
MIME	<i>Multipurpose Internet Mail Extensions</i>
P2P	<i>Peer-to-Peer</i>
OASIS	<i>Organization for the Advancement of Structured Information Standards</i>
OSI	<i>Open Systems Interconnection</i>
QoS	<i>Quality of Service</i>
RPC	<i>Remote Procedure Call</i>
SAML	<i>Security Assertion Markup Language</i>
SMTP	<i>Simple Mail Transfer Protocol</i>
SOA	<i>Service Oriented Architecture</i>
SOAP MTOM	<i>SOAP Message Transmission Optimization Mechanism</i>
SRP	<i>Secure Remote Password</i>
SSL	<i>Secure Sockets Layer</i>
TCP	<i>Transmission Control Protocol</i>
TLS	<i>Transport Layer Security</i>
UDDI	<i>Universal Description Discovery & Integration</i>
UDP	<i>User Datagram Protocol</i>
URI	<i>Uniform Resource Identifier</i>

XML	<i>Extensible Markup Language</i>
W3C	<i>World Wide Web Consortium</i>
WSCI	<i>Web Services Choreography Description Language</i>
WSDL	<i>Web Services Description Language</i>
<i>WS-Authorization</i>	<i>Web Services Authorization</i>
WS-CAF	<i>Web Services Composite Application Framework</i>
<i>WS-BaseNotification</i>	<i>Web Services Base Notification</i>
WS-BPEL	<i>Web Services Business Process Execution Language</i>
WS-I	<i>Web Services Interoperability Organization</i>
<i>WS-Policy</i>	<i>Web Services Policy</i>
<i>WS-Security</i>	<i>Web Services Security</i>
<i>WS-Reliability</i>	<i>Web Services Reliability</i>

Capítulo 1

Introdução

A computação vem revolucionando a vida de pessoas, de empresas e cada vez mais sua presença está se infiltrando nas mais variadas áreas. Desde a abertura para o mundo corporativo, e usuários em geral, pelo governo Americano em 1992, a Internet é seguramente um fator que tem propiciado esta presença ubíqua da computação em todos os processos do dia-a-dia. À medida que ocorre esta ubiquidade, os requisitos necessários à comunicação de variadas naturezas incluem a necessidade de se considerar a existência de sistemas, ambientes e plataformas heterogêneas.

Em relação à comunicação entre pessoas e entre pessoas e sistemas, a Internet, ou melhor, a *Web*, tem desempenhado um importante papel na satisfação destes requisitos.

Os *Web Services* [1] oferecem uma nova forma para o uso da Internet, na qual aplicações remotas podem se comunicar através de padrões que garantam a interoperabilidade destes sistemas. Com esta abordagem, a *Web* pode oferecer não somente informações, mas uma nova maneira de se obter a computação de forma distribuída.

Neste contexto a Internet pode oferecer a capacidade de conectar sistemas autônomos, heterogêneos e distribuídos geograficamente, a fim de que possam trabalhar de forma cooperativa. Os *Web Services* surgem como a tecnologia natural que será capaz de transformar a *Web*, hoje rica em informações, em um ambiente rico em funcionalidades e serviços que poderão ser utilizados de forma ubíqua.

Neste ambiente, segurança é um aspecto crítico, visto que a comunicação de forma ampla tem os seus benefícios, porém a acompanham seus riscos inerentes. Uma falha na segurança pode acarretar grandes prejuízos.

A autenticação é uma forma de garantir que apenas usuários, ou sistemas, devidamente reconhecidos possam ter acesso a serviços e informações. Um meio amplamente utilizado para efetuar este controle é através do uso de senhas.

Entretanto na maioria dos sistemas utilizados atualmente este controle apresenta deficiências, e um intruso, devidamente preparado e mal intencionado, pode quebrar este controle.

Este controle é de fundamental importância para que os *Web Services* sejam efetivamente adotados como a arquitetura para o oferecimento de computação distribuída através da *Web*.

Considerando que os *Web Services* utilizam os protocolos da internet como HTTP (*HyperText Transfer Protocol*) [2], por exemplo, o controle de autenticação a estes serviços pode ser feito utilizando tecnologias de autenticação já estabelecidas na *Web* tais como autenticação básica via HTTP ou autenticação via HTTPS (*Secure HyperText Transfer Protocol*) [2]. O problema destas tecnologias é que a maioria delas apresenta deficiências em relação à segurança [3].

Outro aspecto importante é que muitos servidores utilizados na *Web* não utilizam o protocolo HTTPS. Este protocolo é utilizado, por exemplo, na maioria dos sites de aplicações bancárias através da Internet e oferece um maior grau de segurança em relação ao protocolo HTTP.

Segundo o site *SecuritySpace.com*, que realiza pesquisas e auditorias on-line sobre a Internet, se considerarmos o número de servidores que possuem suporte para o uso de HTTPS [4] em relação ao número total de servidores web ativos [5] em outubro de 2004, chega-se a conclusão de que apenas pouco mais de 1% (1,12%) de todos os servidores utilizam o HTTP “seguro”.

Segundo a Netcraft [6], outra empresa especializada em pesquisas e análises de dados sobre aspectos da Internet, em Janeiro de 2004, apenas 0,68% dos servidores brasileiros (.br) utilizavam suporte a HTTPS.

Uma outra abordagem possível, para o controle de autenticação a *Web Services*, é o uso tecnologias que estão em uma camada superior e que são baseadas na linguagem XML (*Extensible Markup Language*) [7], a linguagem padrão de todos os protocolos utilizados em *Web Services*.

A especificação *WS-Security* (*Web Services Security*) [8], por exemplo, possui mecanismos para a construção de protocolos para autenticação, porém não trata diretamente de como a autenticação deve ser realizada, mas sim da segurança entre as mensagens trocadas entre *Web Services*.

Adicionalmente existe ainda o fato de que o modelo de segurança para os *Web Services* ainda não está completamente definido, havendo várias propostas, algumas já adotadas e outras em fase de definição.

Este trabalho propõe e implementa uma solução para o controle de autenticação seguro a *Web Services*. Esta solução será baseada em um protocolo de autenticação seguro através de senhas.

Para a realização desta proposta serão consideradas as tecnologias atuais disponíveis para a implementação dos requisitos de segurança e também os principais protocolos para autenticação existentes atualmente.

Este trabalho está organizado em cinco capítulos e, além disso, conta com um apêndice onde é apresentado um resumo dos vários padrões utilizados em *Web Services*.

No Capítulo 2 são introduzidos os principais conceitos relativos aos *Web Services* e, além disso, são apresentadas as várias camadas existentes na arquitetura desta tecnologia, bem como os protocolos para comunicação, definição e descrição de *Web Services*.

O Capítulo 3 aborda, sob a óptica dos *Web Services*, os requisitos necessários ao estabelecimento da segurança em sistemas distribuídos baseados nesta tecnologia além de apresentar os protocolos já padronizados para tal fim.

No Capítulo 4 é apresentada a solução para o controle de autenticação a *Web Services*. Este capítulo define a solução do ponto de vista teórico, porém com informações que permitam sua implementação.

O Capítulo 5 mostra os resultados obtidos em relação à implementação da solução proposta. Finalmente, no capítulo 6 serão apresentadas algumas importantes conclusões obtidas durante a realização deste trabalho e sugestões para potenciais trabalhos futuros, ligados ao tema.

Capítulo 2

Web Services

Web Services [1] são programas (software) cujo objetivo é permitir a comunicação entre diferentes aplicações através de uma rede, utilizando para isto mensagens e protocolos baseados na linguagem XML [7] e em protocolos da *Web* tais como o protocolo HTTP [2].

Uma importante característica dos *Web Services* é que eles permitem um fraco acoplamento entre sistemas, ou seja, existe um baixo grau de interdependência entre as partes que se comunicam e as modificações internas em cada parte não são percebidas externamente.

Sendo assim, é possível a comunicação entre diferentes plataformas e ambientes, desde que estas aplicações adotem o conceito representado pelos *Web Services*. Para tanto, é necessária a criação de uma interface que disponibilize a aplicação como *Web Services*.

Outra importante característica dos *Web Services* é que juntamente com os mesmos existem informações sobre o formato das mensagens que serão trocadas entre as aplicações; a forma como estas mensagens serão transportadas, a localização do serviço, entre outras informações. Todas estas informações são baseadas na linguagem XML, permitindo desta forma uma descrição completa do serviço e possibilitando que outras aplicações possam usar este serviço da maneira adequada.

Em essência os *Web Services* não representam uma nova idéia para a comunicação entre sistemas, mas representam uma importante evolução, pois a comunicação se baseia em padrões de comunicação amplamente aceitos e utilizados em um âmbito global que são os padrões utilizados pela Internet.

Os *Web Services* fazem parte de uma nova onda que está movimentando e transformando a *Web*. Hoje em dia, a *Web* é rica em informações e a interação com a *Web* é feita basicamente através do binômio homem-máquina.

Em sua evolução a *Web* caminha para oferecer uma série de funcionalidades que permitam a interação máquina-máquina, através de seus padrões de comunicação. Desta forma o uso da infra-estrutura de comunicação hoje disponibilizada pela Internet será ampliado, permitindo novas aplicações e novos contextos para o uso da Internet.

A *Web* é uma realidade e sua penetração é cada vez maior em todas as áreas. Pode-se afirmar sem sombra de dúvidas que o sucesso da *Web* está ligado aos padrões que a mesma

utiliza. Para acessar uma informação na Internet, basta conhecer a URI (*Uniform Resource Identifier*) [9], não importando a plataforma e o ambiente onde este recurso está localizado. Da mesma forma os *Web Services* compartilham este princípio e isto se dá graças aos padrões definidos para os *Web Services*.

Existem atualmente algumas organizações envolvidas na criação e na especificação de padrões para os *Web Services*. Normalmente nestas organizações existem grupos de trabalho envolvidos com os vários aspectos da tecnologia e uma importante característica destes grupos é a presença de representantes de universidades, pesquisadores e principalmente de empresas, normalmente concorrentes entre si. Isto é de fundamental importância para o sucesso e a adoção dos *Web Services*.

O W3C (*World Wide Web Consortium*) [10] é responsável por uma série de padrões diretamente ligados aos *Web Services*, como por exemplo: a linguagem XML; o protocolo SOAP [11] que define o formato das mensagens utilizadas; a linguagem para descrição de *Web Services*, WSDL (*Web Services Description Language*) [12]; entre outras. Em linhas gerais pode-se dizer que as tecnologias básicas ligadas aos *Web Services* são tratadas pelo W3C.

Outra importante organização é a OASIS (*Organization for the Advancement of Structured Information Standards*) [13]. Entre os vários padrões desenvolvidos por esta organização pode-se citar: UDDI (*Universal Description Discovery & Integration*) [14] que consiste em uma série de métodos para a descoberta e chamada de forma dinâmica de *Web Services*; WS-Security (*Web Services Security*) [8] que contém especificações cujo objetivo é garantir a integridade e a confidencialidade na troca de mensagens SOAP entre *Web Services*; a linguagem SAML (*Security Assertion Markup Language*) [15] cujo objetivo é criar e trocar informações sobre segurança, como uma autenticação bem sucedida, por exemplo, entre *Web Services*. Comparado com o W3C, o trabalho da OASIS está ligado a questões de mais alto nível nos *Web Services*, envolvendo áreas tais como *E-Commerce*; *Supply Chain*; setor público e outras áreas específicas.

Uma outra organização importante no contexto dos *Web Services* é a WS-I (*Web Services Interoperability Organization*) [16]. Esta organização tem por foco promover a interoperabilidade entre *Web Services* nas diversas plataformas, sistemas operacionais e linguagens de programação, contribuindo desta forma, para que este objetivo seja atingido através dos *Web Services*. Uma contribuição importante desta organização é a publicação de perfis (*profiles*).

Estes perfis contêm um conjunto de padrões ligados aos *Web Services*, suas respectivas versões e uma série de recomendações sobre como estes padrões podem ser utilizados em conjunto a fim de evitar problemas de interoperabilidade entre sistemas. Além disso, esta organização produz casos de uso e também ferramentas cujo objetivo é verificar se implementações de *Web Services* estão em conformidade com as especificações.

2.1 Exemplos de Utilização

A fim de facilitar e deixar mais claro alguns dos possíveis usos para os *Web Services*, apresentaremos alguns exemplos de utilização.

Suponha que um investidor deseje obter informações sobre a cotação de ações na bolsa de valores e sobre opções de investimento. Esta tarefa poder ser facilmente realizada utilizando a Internet, basta que o investidor acesse o *site* de uma corretora de sua confiança. Caso ele não encontre neste site todas as informações para a tomada de decisão, ele pode procurar outros *sites* e repetir o processo.

O uso dos *Web Services* estende este conceito da seguinte forma: agora este investidor possui instalado em seu computador um aplicativo que lhe fornece as melhores opções para investimento em um dado momento a partir de uma série de informações de entrada e, além disso, a partir das opções de investimento com informações fornecidas por algumas corretoras.

Ao iniciar este aplicativo, que está conectado à Internet, o investidor informará alguns dados referentes ao investimento desejado e em seguida realizará uma consulta. O aplicativo então enviará através da Internet mensagens SOAP a todas as corretoras envolvidas na pesquisa, solicitando os dados referentes às melhores posições sobre investimentos em ações.

Ao receber a solicitação, o serviço de informação de cada corretora, disponibilizado como *Web Services* receberá a mensagem, processá-la-á e responderá com uma outra mensagem SOAP que então será retornada para o aplicativo do cliente.

Deve-se notar que mesmo que cada corretora utilize diferentes formatos para representar internamente seus dados ainda assim esta aplicação poderia enviar mensagens SOAP para as corretoras, a fim de obter os dados e tratá-los de acordo com as suas especificações próprias. Isto é possível, pois juntamente com os *Web Services*, estaria presente sua descrição que inclui os serviços disponibilizados e o formato das mensagens utilizadas por cada corretora. A Figura 1 mostra este processo.

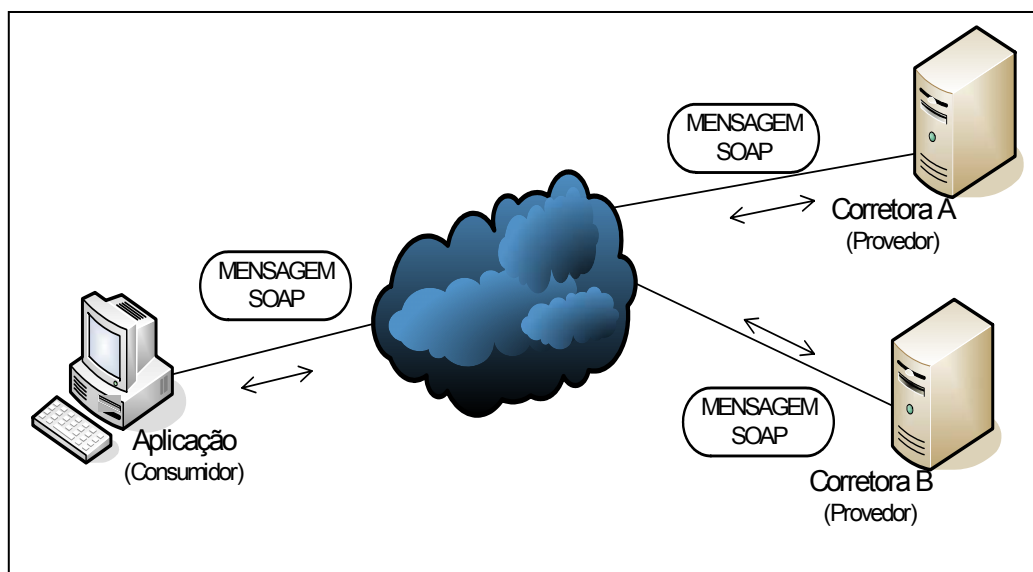


Figura 1 - Aplicação utilizando *Web Services*.

Neste tipo de aplicação, o uso dos *Web Services*, do ponto de vista de execução, é semelhante a uma chamada *RPC (Remote Procedure Call)*, dentro da aplicação cliente feita ao servidor de cada corretora. O importante aqui é que mesmo que a corretora A utilize, por exemplo, a plataforma .Net [17] e caso a corretora B utilize a plataforma J2EE para implementar os seus serviços, a operação acima será perfeitamente factível, ainda que o equipamento do cliente seja um equipamento *Macintosh*.

Um segundo exemplo de uso dos *Web Services* é o seguinte: Em uma relação B2B (*Business-to-Business*) a empresa XYZ deseja adquirir produtos de seu fornecedor, a empresa ZYZ, porém a transação será efetuada entre o sistema de *E-Procurement (Electronic Procurement)* da empresa XYZ e o sistema de vendas da empresa ZYZ.

Para isto o sistema de vendas da empresa ZYZ é disponibilizado através de *Web Services*, sendo assim o sistema da empresa XYZ solicitará os produtos necessários e enviará este pedido para o sistema de vendas da empresa ZYZ através de mensagens SOAP.

Ao receber o pedido, a empresa ZYZ responderá apenas indicando o recebimento com sucesso deste pedido. Este pedido será então processado pela empresa ZYZ e assim que estiver pronto para ser enviado à empresa XYZ, uma cópia da nota fiscal de venda é enviada, concretizando toda a negociação.

Todas as trocas de mensagens serão feitas utilizando o protocolo SOAP. Neste exemplo a interação entre os *Web Services* funciona de maneira assíncrona e as mensagens podem conter documentos que são trocados entre as empresas a fim de confirmar a transação, conforme mostrado na Figura 2.

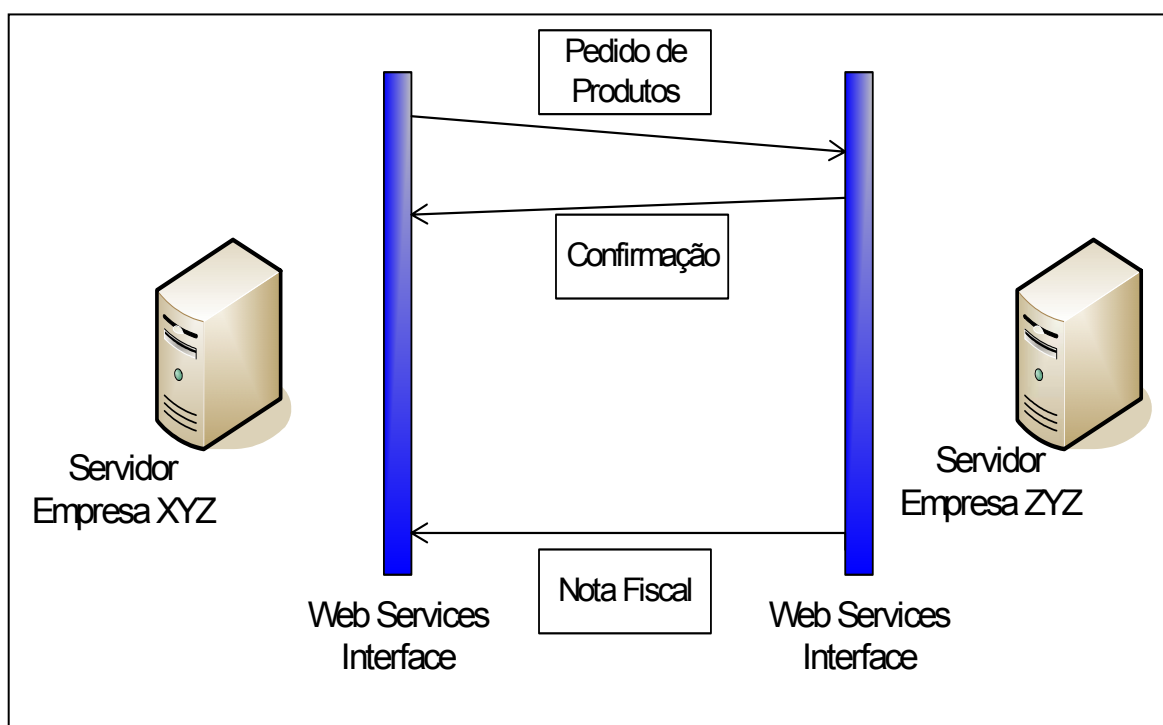


Figura 2 - Web Services em um ambiente B2B.

Finalmente, um terceiro exemplo: Uma agência de turismo oferece para os seus clientes a possibilidade de contratação de um pacote completo de serviços incluindo: reservas em hotéis; aluguel de veículo; passagens aéreas; reservas em restaurantes; e, eventos culturais.

A agência de turismo possui uma plataforma disponível na Internet (*e-WSTRAVEL*) e dessa forma o cliente pode entrar em contato com esta agência, a partir de seu URI e então utilizar a sua plataforma de serviços que basicamente é um software que se comunica com outras empresas através do uso dos *Web Services*.

A partir dos dados informados pelo cliente sobre a viagem e suas necessidades, o sistema de atendimento da agência poderá se comunicar com diferentes empresas, via *Web Services*, a fim de solicitar os serviços que se encaixam na preferência do cliente.

A partir do resultado destas pesquisas, o pacote que atende os requisitos poderá ser montado. Este pacote poderá conter as passagens aéreas, o aluguel de um veículo, o hotel, e para um dia específico, a reserva de um restaurante e a compra de ingressos para um show de teatro. A Figura 3 mostra um diagrama de como esta agência de turismo eletrônica poderia funcionar.

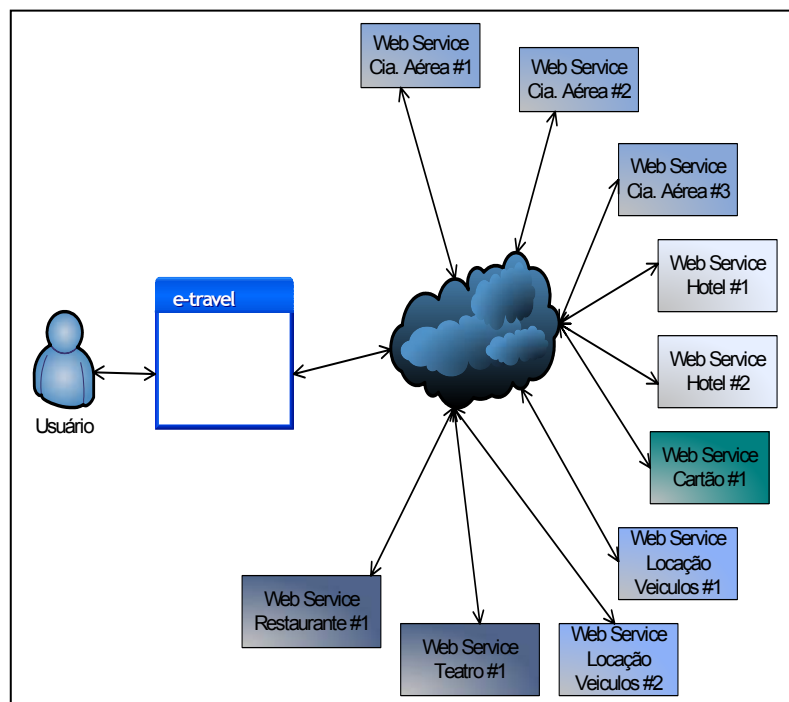


Figura 3 - Web Services em transações complexas.

Considerando que o cliente concorde com a compra do pacote, uma série de passos deverá ser executada a fim de confirmar a compra de todos estes serviços.

Neste caso o cliente deverá informar os dados de seu cartão de crédito, na plataforma da agência de turismo a fim de confirmar a compra de todo o pacote.

Aqui surgem algumas questões interessantes, como por exemplo: é necessária que a comunicação entre a plataforma da agência de turismo e os *Web Services* da administradora de cartão de crédito seja feita de forma segura, bem como entre a agência de turismo e seus fornecedores a fim de se garantir a integridade e confidencialidade das mensagens.

Outro fato a ser considerado; supondo que a plataforma realize as reservas das passagens e do hotel e, em seguida, ao tentar realizar a compra dos ingressos para o teatro esta compra não possa ser realizada, devido ao fim dos ingressos, por exemplo, neste caso a solicitação do cliente não poderá ser completada, porém foi parcialmente realizada.

Neste caso, a plataforma deverá possuir um mecanismo semelhante a uma transação, a fim de que seja possível desfazer as operações realizadas, visto que é necessário que todas sejam concluídas com sucesso e não apenas algumas.

Finalmente a sequência de todo o processo de compra dos vários serviços, comunicação com a empresa de cartões a fim de realizar cada transação precisa ser organizada e gerenciada de alguma forma, ou seja, é necessária uma orquestração dos vários *Web Services* a fim de o resultado esperado aconteça. Neste exemplo isto é importante visto que o cliente poderá

adquirir um ou mais serviços e em cada situação a ordem e a sequência das operações será diferente uma das outras.

Considerando que a comunicação se dá através de *Web Services*, é necessária a presença de mecanismos que garantam todos os comportamentos especificados.

Outrossim, pode-se pensar em um ambiente dinâmico onde os fornecedores para a agência de turismo seriam alterados conforme a necessidade, ou seja, em uma determinada consulta um conjunto de *Web Services* seria utilizado em outra, um outro conjunto poderia ser a base para a comunicação com outros fornecedores.

Neste caso, seria necessária a utilização do UDDI para a pesquisa, descoberta e utilização dos *Web Services* em modo de execução da plataforma da agência de turismo.

2.2 Arquitetura Orientada a Serviços (SOA)

O modelo de funcionamento dos *Web Services* é baseado na arquitetura orientada a serviços (*Service Oriented Architecture* – SOA) [18]. A SOA é uma definição abstrata que descreve como entidades computacionais podem interagir a fim de se obter os resultados desejados. Uma entidade é um elemento ativo que pode se comunicar com outro utilizando um protocolo de comunicação.

Neste contexto, um serviço é uma unidade de trabalho que é executada por uma entidade em benefício de outra. O diagrama da Figura 4 mostra as entidades presentes nesta arquitetura e os relacionamentos entre as mesmas:

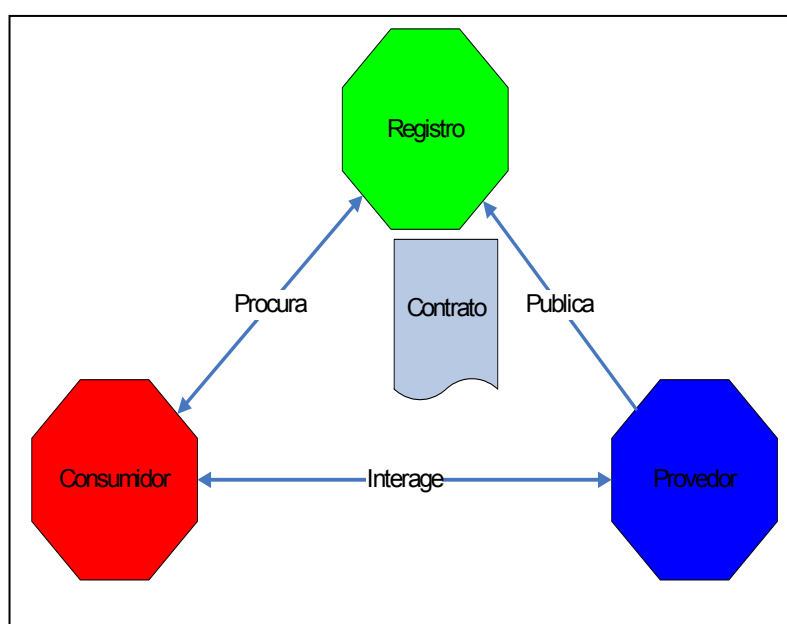


Figura 4 - Arquitetura SOA.

O consumidor é uma aplicação, um módulo de software, ou mesmo um outro serviço que requisita um serviço. Para tanto esta entidade realiza uma procura em um registro; após encontrá-lo, o consumidor obtém o contrato deste serviço e se liga ao mesmo utilizando um meio de transporte, iniciando então uma interação com o provedor do serviço.

O consumidor utiliza o serviço através da troca de mensagens com o provedor. Esta mensagem é formatada de acordo com o contrato deste serviço.

O provedor do serviço é uma entidade que pode ser endereçada em uma rede e que aceita e executa alguma unidade de trabalho, a partir das mensagens recebidas do consumidor. O provedor poder ser um *mainframe*, um servidor de aplicações, um componente de software, ou qualquer outro tipo de entidade que execute um serviço. Uma das responsabilidades do provedor é publicar o seu contrato de uso no registro a fim ele seja acessado pelos consumidores.

O registro é uma entidade, também endereçável em uma rede, e que pode armazenar os contratos recebidos dos provedores e enviá-los no caso de uma busca feita pelo consumidor.

O contrato de serviço é uma especificação contendo a forma que o consumidor utilizará o serviço fornecido pelo provedor. Para tanto, o mesmo especifica as mensagens trocadas entre as entidades e seus respectivos formatos. Um contrato pode conter pré-condições e pós-condições para a execução do serviço, como por exemplo, um determinado nível de segurança que deve ser satisfeito. Além disso, o contrato pode especificar níveis de qualidade de serviço (QoS - *Quality of Service*), como por exemplo, o tempo máximo de resposta do serviço.

Um importante aspecto da SOA é que existe uma separação entre a implementação do serviço e a sua interface, ou seja, o consumidor não precisa se preocupar com os detalhes de implementação do serviço ou como o mesmo é executado, após receber o pedido de uso pelo consumidor.

2.3 A Arquitetura dos *Web Services*

O conhecimento da arquitetura dos *Web Services* é importante para o entendimento de seus objetivos e da visão que os envolve. A arquitetura especifica as camadas e os componentes funcionais, definindo-se relações a fim de obter as propriedades desejadas desta arquitetura.

A fim de estabelecer a arquitetura, o W3C, inclusive, criou um grupo de trabalho [19] que funcionou de janeiro de 2002 a janeiro de 2004. Porém neste período, devido a fatores como os inúmeros padrões e especificações que surgiram e a divergência de opiniões sobre como esta arquitetura seria definida por parte dos participantes do grupo não produziram os

resultados desejados. Sendo, assim ainda não há uma especificação definitiva das organizações e indústria de software sobre como é a arquitetura dos *Web Services* [20].

Por um lado isto é perfeitamente aceitável, visto que a tecnologia é recente e está em franco desenvolvimento. Por outro lado, a inexistência de uma arquitetura comum pode levar as situações da falta de interoperabilidade entre sistemas, mutilando, desta forma, uma das principais características do *Web Services*.

Neste momento a arquitetura para os *Web Services*, em algumas camadas ainda é algo incerto. Baseado em algumas propostas [21] [1], e após um estudo dos vários protocolos existentes, pode-se dizer que a Figura 5 representa uma visão bastante aproximada da arquitetura dos *Web Services*.

Esta arquitetura contempla os principais aspectos relacionados com os *Web Services*, considerando o seu estágio atual de desenvolvimento e os vários protocolos propostos até o presente momento.

A Figura 5 relaciona cinquenta diferentes protocolos, cada um com o seu objetivo. Deve ser ressaltado que muitos protocolos são redundantes, ou seja, oferecem a mesma solução com pequenas variações. Na figura eles são circuncidados por uma linha tracejada. Há casos em que um novo protocolo substituiu um protocolo anterior, neste caso ele está inserindo dentro de caixa que representa o protocolo mais atual.

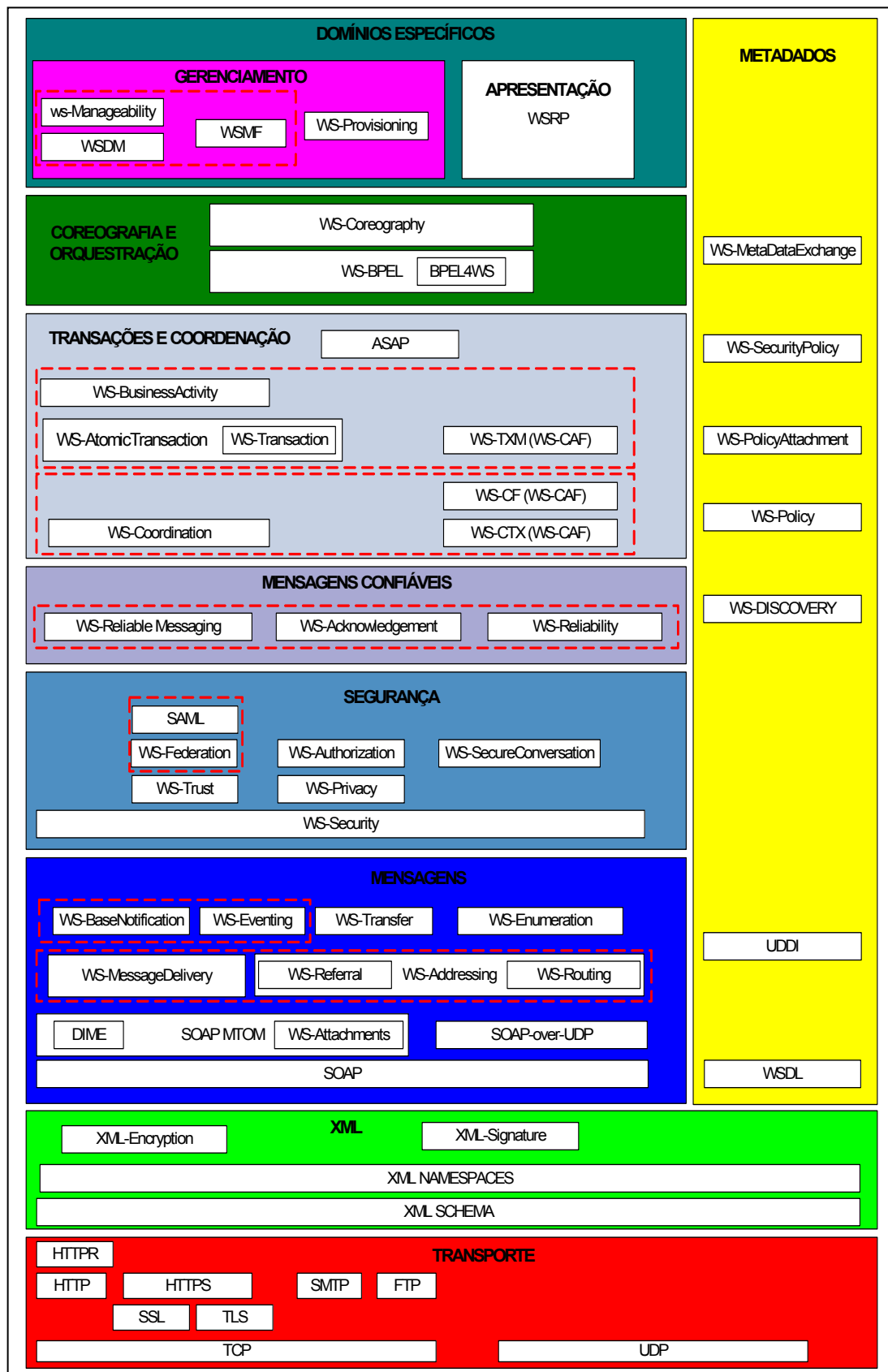


Figura 5 - Arquitetura dos Web Services.

Os protocolos presentes em cada camada representam as abordagens existentes ligadas ao domínio da camada. Deve ser ressaltado que muitos protocolos ainda não foram analisados e publicados como padrão pelas organizações de padronização, e existem protocolos, como por exemplo, *WS-Authorization* [22], que ainda nem foram publicados, mas que estão previstos

sê-lo em um curto período de tempo. O apêndice A traz informações sobre todos os protocolos existentes na Figura 5. A arquitetura é composta de várias camadas, cada uma delas com objetivos específicos e fornecendo serviços às camadas superiores.

2.3.1 Camada de Transporte

A Camada de Transporte da arquitetura dos *Web Services* é responsável pela troca de mensagens entre as partes. Esta camada se baseia nos protocolos de transporte e de aplicação da arquitetura Internet [2]; principalmente o protocolo HTTP, que inclusive é o protocolo de transporte definido no *Basic Profile* [23], publicado pela WS-I.

O protocolo HTTP especifica a troca de mensagens entre o cliente e o servidor e não oferece recursos para lidar com situações como: perda de mensagens; envio de mensagens duplicadas; transmissão de longas mensagens; e, envio de confirmações.

Face a esta característica, a IBM, por exemplo, especificou o protocolo HTTPR (*Hypertext Transfer Protocol Reliable*) [24] que utiliza o HTTP e permite uma entrega confiável de mensagens HTTP entre o servidor e o cliente, abrindo caminho para o serviço confiável.

Esta é uma camada bem definida e existem ligações do protocolo SOAP com os protocolos HTTP, SMTP (*Simple Mail Transfer Protocol*) [2] e FTP (*File Transfer Protocol*) [2].

Cabe aqui um importante registro, pois o termo “camada de transporte” originalmente se refere a uma das camadas do modelo de referência [2] OSI (*Open Systems Interconnection*) e também a uma das camadas da arquitetura Internet.

Os protocolos TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*) são protocolos da camada de transporte nesta arquitetura, sendo que os protocolos HTTP, SMTP e FTP são protocolos da camada de aplicação.

Apesar disto, foi utilizado o termo “camada de transporte”, visto que o mesmo é utilizado amplamente para se referir aos protocolos básicos da Internet sobre os quais são construídos os *Web Services*.

2.3.2 Camada XML

A Camada XML contém os protocolos que especificam a linguagem XML e suas extensões, como por exemplo, o protocolo *XML-Encryption* [25]. Todos os outros protocolos são construídos a partir da linguagem que representa o fundamento da próxima geração da

Web. Esta camada está padronizada e completamente aceita na arquitetura. A seção 2.4 mostra as principais características desta linguagem.

2.3.3 Camada de Mensagens

A Camada de Mensagens define a formatação básica da mensagem e as opções básicas para a entrega, independentemente de linguagem de programação, sistema operacional ou plataforma. O protocolo SOAP é a base desta camada, sendo que uma série de outros protocolos relacionados à entrega de mensagens está presente nesta camada.

Uma iniciativa interessante é o SOAP MTOM (*Message Transmission Optimization Mechanism*) [26], que basicamente descreve um mecanismo para otimizar a transmissão e o formato de uma mensagem SOAP através de uma codificação em certas partes da mensagem, especificamente aqueles elementos cujo conteúdo é do tipo **xs:base64Binary** [27].

Desta forma é possível otimizar a transmissão da mensagem através da compactação, utilizando seqüências de caracteres compactas que podem ser reconstruídas no lado receptor.

2.3.3 Camada de Segurança

Em sistemas fracamente acoplados segurança é um importante aspecto a ser considerado. No exemplo de uso da agência de turismo (*e-WSTRAVEL*) para efetivar a compra, o cliente deve informar seus dados de cartão de crédito e estes dados serão repassados para diferentes empresas a fim de autorizar a compra de cada item do pacote de turismo.

Neste ambiente, é necessário garantir a segurança fim a fim, visto que existem intermediários na comunicação. A segurança disponível na camada de transporte como *Secure Sockets Layer* (SSL) [2] **Erro! Fonte de referência não encontrada.**, ou *Transport Layer Security* (TLS)[2], está voltada para uma comunicação ponto a ponto, sendo assim, existem requisitos específicos de segurança em relação aos *Web Services*.

A camada de segurança define os protocolos a fim de garantir uma série de propriedades como, por exemplo, a integridade da mensagem, sua confidencialidade e serviços como autenticação e autorização.

A especificação *WS-Security*, cujo objetivo, é possibilitar a segurança das mensagens SOAP com o objetivo de garantir sua integridade e confidencialidade, é um padrão OASIS para segurança de mensagens SOAP. Seu funcionamento basicamente envolve o uso de assinaturas digitais e de criptografia.

2.3.4 Camada de Mensagens Confiáveis

O uso do protocolo SOAP sobre HTTP, por exemplo, não garante a entrega das mensagens. A Camada de Mensagens Confiáveis possui mecanismos para garantir a entrega da mensagem. A entrega confiável de mensagens envolve aspectos como a garantia de entrega; a ausência de duplicações; e, ainda a entrega ordenada destas mensagens.

Para tanto, os protocolos desta camada adicionam informações à mensagem como, por exemplo: um identificador a fim de garantir sua unicidade; número de seqüência; tempo de duração da mensagem; e, ainda o uso de um método para confirmar a entrega com sucesso ou não da mensagem, semelhante a uma confirmação positiva ou negativa.

A especificação *WS-Reliability* [28] é um padrão OASIS recém aprovado e que aborda a questão do envio de mensagens confiáveis usando o protocolo SOAP e como transporte o protocolo HTTP.

2.3.5 Camada de Transações e Coordenação

No terceiro exemplo de uso dos *Web Services*, após a escolha dos itens do pacote de turismo, várias mensagens devem ser trocadas a fim de efetuar a compra de cada item deste pacote. Caso a compra de um dos itens não seja possível, o pacote não pode ser confirmado de maneira integral e é necessário que as compras já efetuadas sejam canceladas.

Isto ilustra claramente a necessidade do conceito de transação que representa uma série de atividades que devem ser executadas de maneira integral e que podem ser vistas externamente como uma única operação. Caso ocorra uma falha em uma destas atividades, o processamento da transação deve ser cancelado ou outra ação pode ser tomada. A transação pode ainda, ser de curta ou longa duração, conforme o caso.

Para a implementação de requisitos como a transação, em linhas gerais esta camada define um contexto de operação que pode ser compartilhado entre serviços. Este contexto define condições de operação dos vários *Web Services* e permite a troca de informações sobre as atividades que estão sendo executadas.

Neste ambiente é necessária ainda, uma coordenação a fim de alterar o estado deste contexto e propagá-lo para outros *Web Services* que estão trabalhando em conjunto a fim de executar um objetivo coletivo comum.

O *Web Services WS-CAF (Web Services Composite Application Framework)* [29], que é tema de um comitê técnico da OASIS, representa uma proposta para especificar a camada de transações para os *Web Services*.

2.3.6 Camada de Coreografia e Orquestração

As características dos *Web Services*, como o fraco acoplamento entre as partes que se comunicam, permitem que esta tecnologia possa ser utilizada de várias maneiras nos processos de *E-Business*.

Estes processos normalmente envolvem vários sistemas entre seus requisitos, podendo-se citar: a existência de uma sequência de operações, que muitas das vezes, pode ser variável; a realização de transações de maneira síncrona ou assíncrona; a necessidade do monitoramento de todo o estado do processo; entre outras.

Esta camada trata dos protocolos de alto nível necessários para a implementação dos requisitos comuns do *E-Business*. Nesta camada podem ser destacadas duas subcamadas, uma que trata da orquestração e a outra da trata da coreografia dos vários *Web Services*.

O conceito de orquestração está ligado a um grupo de *Web Services* que interagem a fim de realizar as regras de negócios, e, neste grupo existem um supervisor e um maestro, que é responsável pela condução de todo o processo a fim de alcançar os resultados desejados. A especificação *WS-BPEL* [30], que é objeto de especificação por parte de um comitê técnico do OASIS, representa uma proposta para a orquestração de *Web Services*.

Acima da subcamada de orquestração se situa a subcamada de coreografia. O conceito de coreografia é mais amplo e está ligado a várias partes que interagem, normalmente de forma pública, a fim de realizar um objetivo comum, seu conceito está próximo e é semelhante ao P2P onde cada parte possui sua visão ou seu papel nesta coreografia e o executa. A execução de uma parte desta coreografia pode ser feita, por exemplo, através de uma orquestração.

Para melhor ilustrar, no terceiro exemplo a plataforma da agência de turismo (*e-WSTRAVEL*) e os *Web Services* da companhia aérea são exemplos de partes envolvidas em uma coreografia. A sequência de operações que deve ser executada pela plataforma é diferente daquela que será executada pela companhia.

Para confirmar uma reserva e realizar a venda de uma passagem, o *Web Service* da companhia deverá orquestrar uma série de operações a fim de garantir que a venda foi realizada com sucesso e este estado será passado para a outra parte, que executará outros processos a fim de completar a compra de todo o pacote de turismo.

Um grupo de trabalho do W3C está trabalhando na especificação da *Web Services Choreography Description Language* (WSCI) [31] cujo objetivo é permitir a coreografia entre vários *Web Services*.

2.3.7 Camada de Domínios Específicos

A perspectiva de uso dos *Web Services* no *E-Business* vai muito além dos exemplos aqui citados. No topo das camadas apresentadas existe a camada de Domínios Específicos, que contém os protocolos de uso particular para certas áreas de negócio como, por exemplo: saúde [32]; comércio eletrônico [33]; *E-Procurement* [34]; gerenciamento de recursos [35]; interfaces com usuário [36]; *supply chain* [37]; *E-Government* [38]; voz sobre XML (VoiceXML) [39]; entre outras.

As especificações desta camada podem variar conforme a necessidade particular de cada um destes domínios e o uso destes protocolos se restringem a estas aplicações, atendendo a requisitos específicos. A abordagem destes protocolos foge ao escopo deste trabalho.

2.3.8 Camada de Metadados

A Camada de Metadados é utilizada tanto em modo de execução, quanto em modo de criação dos *Web Services* e trata da definição, descoberta e políticas de uso para *Web Services*. A base desta camada é a linguagem WSDL, um padrão de fato para definição e descrição de *Web Services*.

Esta camada contém o protocolo UDDI, utilizado para a descoberta de *Web Services*, conforme pode ser visto na descrição da arquitetura orientada à serviços (SOA), mostrada na Figura 4, ou seja, o registro utiliza o protocolo UDDI para permitir a busca de serviços pelos consumidores e o envio da descrição destes serviços, aos consumidores.

Os protocolos ligados à política de uso de *Web Services*, também estão presentes nesta camada, como o protocolo *WS-Policy* [40] que expressa políticas de uso, ou requisitos que os *Web Services* devem respeitar para serem utilizados. Como exemplo, pode-se citar: esquemas de autenticação; protocolo de transporte a ser utilizado; características de QoS; políticas de segurança; entre outros.

2.4 A Linguagem XML

A linguagem XML é uma linguagem utilizada para a representação de documentos de forma estruturada, independentemente da plataforma onde é utilizada.

Estes documentos, chamados documentos XML, são escritos conforme as regras desta linguagem. Entre estas regras pode-se citar: os documentos devem ser simples de serem utilizados através da Internet; estes documentos podem ser processados por computadores e legíveis a seres humanos, entre outras.

Um documento XML é composto de marcas (*tags*) e dentro destas *tags* é possível a existência de dados e atributos. A linguagem contém regras de sintaxe que determinam o formato que estas *tags* devem possuir, porém, a especificação da linguagem não contém a semântica e a sintaxe das *tags* que são utilizadas. O conjunto de *tags* é criado pelos usuários da XML conforme suas necessidades e sua semântica advém de um acordo entre as partes que utilizarão estes dados.

Devido a estas características, a linguagem XML é um padrão para troca de dados através da Internet. A Figura 6 mostra um exemplo de um documento XML válido, ou seja, aquele que está de acordo com as regras da linguagem XML.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<priceList>
  <coffee> id="10003"
    <name>Mocha Java</name>
    <price>11.95</price>
  </coffee>
  <coffee> id="10004"
    <name>Sumatra</name>
    <price>12.50</price>
  </coffee>
</priceList>
```

Figura 6 - Exemplo de documento XML.

Como pode-se ver no exemplo acima, em um documento XML cada *tag* inicial (**<priceList>**) precisa de uma *tag* final (**</priceList>**) correspondente. A *tag* pode possuir atributos, no exemplo acima **id** é um atributo para a *tag* **<coffee>**. Na linguagem XML um par constituído pela *tag* inicial e final, pode ser chamado de elemento. O elemento pode conter outros elementos filhos e desta forma o documento XML é estruturado.

2.4.1 Namespaces

Como o conjunto de *tags* pode ser criado livremente, nada impede que duas *tags* iguais sejam utilizadas para representar diferentes dados. Na Figura 6, a *tag* **<priceList>** é utilizada para representar uma lista de preços de diferentes tipos de café. Nada impede, porém, que este mesma *tag* venha a ser utilizada para representar uma lista de preços de veículos, por exemplo.

A fim de resolver este conflito, a linguagem contém o conceito de *namespace* [41]. O *namespace* é um prefixo colocado em cada *tag* para diferenciá-la de outras *tags* que possam haver em um mesmo documento. Este prefixo deve ser único, visto que prefixos iguais

levariam ao mesmo problema inicial, logo um bom método para nomear *namespaces* é utilizar um URI acrescido de identificadores, desta forma este conceito garante a unicidade das *tags*.

O uso de um URI não significa que o *namespace* esteja relacionado a um endereço válido na web. A idéia é possuir um nome único que posteriormente será substituído pelo programa que processará o documento XML.

Para acrescentar um *namespace* a uma *tag* é necessário utilizar o atributo **xmlns** seguido do prefixo que o identifica. O exemplo abaixo mostra o código da Figura 6, utilizando o conceito de *namespaces*.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<cfpl:priceList xmlns:cfpl="www.larc.facom.ufu.br/tags/cfpl"
  <coffee id="10003"
    <name>Mocha Java</name>
    <price>11.95</price>
  </coffee>
  <coffee id="10004"
    <name>Sumatra</name>
    <price>12.50</price>
  </coffee>
</cfpl:priceList>
<vepl:priceList xmlns:cfpl="www.larc.facom.ufu.br/tags/vepl"
  <vehicle id="10004"
    <maker>GM</maker>
    <name>Astra</name>
    <price>9222.50</price>
  </vehicle>
</vepl:priceList>
```

Figura 7 - Documento XML utilizando *namespaces*.

2.4.2 Esquemas XML

Um documento XML contém elementos, que por sua vez, podem possuir elementos filhos. Considerando esta estrutura é necessário um meio para verificar se um documento é válido, ou seja, além de seguir as regras da linguagem XML o documento deve estar estruturado conforme acordado entre as partes.

No exemplo acima, pode-se perceber que o elemento **<cfpl:priceList>** pode conter, e deve conter, pelo menos um elemento do tipo **<coffee>**.

O Esquema XML [42] consiste de um documento XML que descreve a estrutura que deve ser seguida pelos documentos escritos segundo este esquema. Este mecanismo garante então que duas partes distintas possam trocar um documento XML e cada uma delas possa verificar se o mesmo é válido, considerando que a atividade primária destas partes é processar um documento XML, a validação de um esquema XML é algo natural para estas partes.

O elemento **<priceList>** mostrado anteriormente pode ser definido utilizando-se o esquema mostrado na Figura 7.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<xsd:schema> xmlns:xsd="http://w3.org/2001/XMLSchema"
               xmlns:cfpl="http://larc.facom.ufu.br/tags/cfpl"
               targetNamespace="http://larc.facom.ufu.br/tags/cfpl"
  <xsd:element name="priceList" minOccurs="1"
               type="cfpl:priceListType"/>
  <xsd:complexType name="priceListType">
    <xsd:sequence>
      <xsd:element name="coffee" type="cfpl:coffeType" />
    <xsd:sequence>
    </xsd:complexType>
  </xsd:schema>
```

Figura 8 - Visão parcial do esquema XML de um documento.

2.5 O Protocolo SOAP

O protocolo SOAP especifica a troca de informações entre pontos descentralizados em um ambiente distribuído. A troca de informação é feita utilizando-se documentos XML, que seguem um padrão (esquema) específico.

O protocolo SOAP é uma especificação e dessa forma não está vinculado a um único fornecedor. Dois nós SOAP, implementados por diferentes fornecedores, podem interagir entre si, trocando informações. Uma mensagem SOAP pode ser transportada de várias formas como, por exemplo: HTTP, SMTP e FTP entre outras.

Desta forma o protocolo SOAP utiliza a infra-estrutura existente de Internet para a troca de informações, podendo trafegar livremente entre *firewalls*, simplificando desta forma, seu uso por diferentes pontos distribuídos sobre esta rede. O protocolo SOAP é a base para a criação dos *Web Services*.

2.5.1 Estrutura da mensagem SOAP

Uma mensagem SOAP basicamente é um documento XML que contém um único elemento, o elemento chamado *Envelope*. Este elemento, por sua vez, pode conter um elemento filho chamado *Header* e obrigatoriamente contém um elemento filho chamado *Body*, conforme pode ser visto na Figura 9.

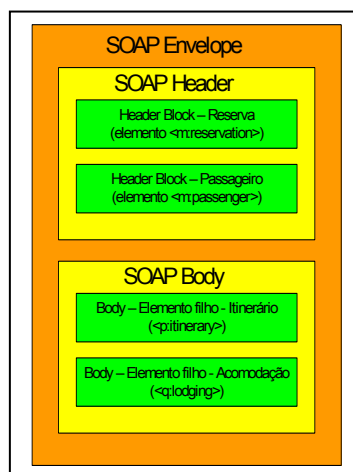


Figura 9 - Estrutura de uma mensagem SOAP.

A mensagem, mostrada na Figura 10, é um exemplo de uma mensagem SOAP escrita em XML.

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation
      xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true"
      <m:reference>uuid:093a2da1-q345-739r-ba5d</m:reference>
      <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <n:name>Åke Jógvan Øyvind</n:name>
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itinerary
      xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2001-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        <p:departing>Los Angeles</p:departing>
        <p:arriving>New York</p:arriving>
        <p:departureDate>2001-12-20</p:departureDate>
        <p:departureTime>mid-morning</p:departureTime>
        <p:seatPreference/>
      </p:return>
    </p:itinerary>
    <q:lodging
      xmlns:q="http://travelcompany.example.org/reservation/hotels">
      <q:preference>none</q:preference>
    </q:lodging>
  </env:Body>
</env:Envelope>
```

Figura 10 - Exemplo de uma mensagem SOAP.

2.5.2 SOAP Envelope

O envelope SOAP consiste de um elemento XML que contém o nome Envelope. Este elemento está definido no *namespace* `http://www.w3.org/2003/05/soap-envelope`. Além disso, o elemento pode conter a definição de outros *namespaces* cujo escopo será todo o envelope.

Dentro do elemento Envelope deve existir um elemento filho *Body* e opcionalmente pode ocorrer a presença de um elemento filho *Header*.

2.5.3 SOAP Header

O elemento *Header* consiste de um mecanismo para estender a funcionalidade de uma mensagem SOAP. Este elemento está definido no *namespace* `http://www.w3.org/2003/05/soap-envelope` e pode conter outros *namespaces* que qualificarão seus atributos ou então seus elementos filhos. Estes elementos filhos são chamados *Header Blocks*. Na Figura 10, o elemento *Header* possui dois filhos.

O elemento *Header* é opcional e, quando presente, deve ser o primeiro elemento filho presente no Envelope. Entre as funcionalidades que podem estar presentes utilizando este elemento pode-se citar: Identificação de Transações; Identificadores da sequência do Pacote; Informações para Autenticação; Identificação de uma Sessão; Informações para roteamento da mensagem e outras informações que podem ser utilizadas pelos processadores SOAP.

Muitos dos protocolos mostrados na Figura 5 utilizam este conceito para implementar suas funcionalidades.

O *Header block* deve estar definido em um *namespace* próprio e pode conter dentro de si qualquer número de elementos filhos. Cada *Header Block* pode conter os seguintes atributos: *encodingStyle*; *role*; *mustUnderstand*; *relay*. Estes atributos têm um significado especial em relação ao processamento da mensagem SOAP.

O atributo *encodingStyle* indica quais regras serão utilizadas para a serialização de partes de uma mensagem SOAP. Este atributo está definido no *namespace* `http://www.w3.org/2003/05/soap-envelope` e seu valor é do tipo **xs:anyURI**, indicando este conjunto de regras. A Figura 11 mostra exemplos de valores para este atributo:

```
http://www.w3.org/2003/05/soap-encoding
http://larc.facom.ufu.br/encoding/
http://www.w3.org/2003/05/soap-envelope/encoding/none
```

Figura 11 - Exemplos de valores para o atributo *encodingStyle*.

O escopo deste atributo é o elemento que o possui e todos os seus elementos filhos, que não tenham nenhum atributo *encodingStyle* diferente do elemento pai. Caso este atributo não esteja presente então o valor padrão `http://www.w3.org/2003/05/soap-envelope/encoding/none` é utilizado indicando que nenhuma declaração é feita sobre as regras de serialização para este elemento e seus filhos.

O atributo *role* é utilizado para indicar o papel de um nó SOAP para o qual um determinado *Header Block* é endereçado. A Figura 12 mostra valores padrão para este atributo:

```
http://www.w3.org/2003/05/soap-envelope/role/next
http://www.w3.org/2003/05/soap-envelope/role/none
http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver
```

Figura 12 - Valores padrão para o atributo *role*.

Um nó SOAP indica um ponto onde a mensagem será processada. Este nó pode ser o nó inicial, o nó final, ou então algum nó intermediário que receberá a mensagem. Este atributo é do tipo **xs:anyURI** indicando o endereço do nó que deverá processar o *Header Block*. Quando este atributo é omitido o valor `http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver` é utilizado, indicando que apenas o último nó deverá tratar o respectivo *Header Block*.

O atributo *mustUnderstand* é utilizado para indicar se o processamento de um *Header Block* é obrigatório ou opcional. Este atributo está definido no *namespace* `http://www.w3.org/2003/05/soap-envelope` e seu tipo é **xs:boolean** podendo ter o valor *true* ou *false*. Caso o atributo seja omitido seu valor padrão é entendido como *false*.

O atributo *relay* é utilizado para indicar se um *Header Block*, direcionado para um nó SOAP receptor, deve ser transmitido ou não caso não seja processado por um nó intermediário. Este atributo é do tipo **xs:boolean**, podendo possuir o valor *true* ou *false*, e pertence ao *namespace* `http://www.w3.org/2003/05/soap-envelope`. A omissão deste atributo é semanticamente equivalente a utilizá-lo com o valor *false*.

Os atributos *role*, *mustUnderstand* e *relay* devem ser utilizados somente nos *Header Blocks* e um nó SOAP receptor deve ignorá-los caso apareça em seus descendentes ou então apareça no elemento *Body* e seus descendentes.

2.5.4 SOAP *Body*

O elemento *Body* contém a informação que será transmitida para o último nó SOAP presente na cadeia, ou seja, o nó receptor. Este elemento está definido no *namespace* `http://www.w3.org/2003/05/soap-envelope` e pode conter outros *namespaces* que qualificarão seus atributos ou então seus elementos filhos.

O *Body* pode conter qualquer número de elementos filhos e todos estes elementos devem ter o atributo *namespaces* definido, isto produz mensagens em que sua interpretação é menos ambígua o que pode ocorrer para elementos não qualificados pelo *namespace*.

O *Body* pode conter o atributo *encodingStyle*, visto anteriormente, que será utilizado para o processamento da informação.

Um filho particular definido para o elemento *Body* é o elemento SOAP *Fault*, que é utilizado para a indicação de erros encontrados em uma mensagem SOAP.

2.5.5 SOAP *Fault*

Um elemento SOAP *Fault* é utilizado para descrever um erro encontrado em uma mensagem SOAP. Este elemento está definido no *namespace* "http://www.w3.org/2003/05/soap-envelope" e contém obrigatoriamente os elementos *Code* e *Reason*. Opcionalmente este elemento pode conter os elementos-filhos *Node*, *Role* e *Detail*.

Para uma mensagem SOAP ser reconhecida como uma mensagem de erro a mesma deve conter um único elemento *Fault* presente no *Body* da mensagem.

O elemento *Code* contém um ou dois filhos que são: elemento obrigatório chamado *Value* e um elemento opcional chamado *SubCode*. O elemento *Value* é do tipo `env:faultCodeEnum` e define um pequeno conjunto de erros de alto nível, conforme mostrado na Tabela 1 a seguir:

Tabela 1 - Códigos de erro SOAP.

NOME	SIGNIFICADO
<i>VersionMismatch</i>	Um elemento inválido foi encontrado no Envelope. O <i>namespace</i> , o nome local ou ambos não estão de acordo com o recomendado pela versão de SOAP utilizada.
<i>MustUnderstand</i>	Um elemento filho de um <i>Header block</i> que possui um atributo <i>mustUnderstand</i> não foi entendido por um nó.
<i>DataEncodingUnknown</i>	Não foi possível a decodificação de um <i>Header block</i> ou elemento <i>Body</i> em um determinado nó
<i>Sender</i>	A mensagem foi incorretamente formada ou não contém

	a informação na ordem especificada
<i>Receiver</i>	A mensagem não pôde ser utilizada devido a problemas de processamento e não na mensagem propriamente dita.

O elemento *SubCode* contém uma estrutura semelhante ao elemento *Code* e contém um ou dois filhos que são: elemento obrigatório chamado *Value* e um elemento opcional chamado *SubCode*. O elemento *Value* é do tipo *xs:Qname* e é um valor definido pela aplicação.

Deve ser notado que a definição é recursiva e que um elemento *Fault* contém obrigatoriamente um elemento *Code* e uma lista de elementos *subCode* que pode conter erros particulares de uma aplicação ou seja, os valores indicados pelo elemento *Value* do *SubCode* podem ser criados de acordo com a necessidade da aplicação.

A Figura 13 mostra uma mensagem SOAP que contém um elemento *Fault*.

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:rpc='http://www.w3.org/2003/05/soap-rpc'>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>rpc:BadArguments</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">Processing error</env:Text>
        <env:Text xml:lang="cs">Chyba zpracování</env:Text>
      </env:Reason>
      <env:Detail>
        <e:myFaultDetails
          xmlns:e="http://travelcompany.example.org/faults">
          <e:message>Name does not match card number</e:message>
          <e:errorCode>999</e:errorCode>
        </e:myFaultDetails>
      </env:Detail>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Figura 13 - Exemplo de Mensagem SOAP com elemento Fault.

O elemento *Reason*, também obrigatório, contém uma explicação do erro em linguagem natural. Este elemento pode conter como filhos, um ou mais elementos *Text*, que havendo mais de um, devem possuir valores diferenciados para o atributo *xml:lang*, obrigatório para este elemento.

O elemento *Node* contém informações a respeito do nó SOAP, existente no caminho da mensagem onde o problema ocorreu. Este elemento é do tipo *xs:anyURI*. Caso um nó não

seja o último no caminho da mensagem o mesmo deve incluir este elemento, na ocorrência de algum erro.

O elemento *Role* identifica o papel do nó em que a falta ocorreu e deve ser um valor válido assumido pelo nó durante o processamento da mensagem. Este elemento é do tipo **xs:anyURI**.

O elemento *Detail* contém informações específicas de erro relacionadas ao elemento SOAP *Body*. Este elemento pode conter zero ou mais elementos filhos chamados Detail Entry.

2.6 A Linguagem WSDL

Na arquitetura orientada a serviços (SOA), mostrada na Figura 4, existe um elemento central e de fundamental importância que é o contrato. O contrato deve conter informações sobre quais são os serviços oferecidos e como eles podem ser utilizados.

A linguagem WSDL consiste em um documento XML que permite a descrição dos *Web Services*. A gramática da linguagem define os *Web Services* de duas formas, uma abstrata e outra concreta. A definição abstrata descreve os *Web Services* considerando as mensagens enviadas e recebidas, independentemente de como serão transportadas. A definição concreta realiza uma ligação entre a descrição abstrata e um meio de transporte específico para as mensagens. A Figura 14 mostra a estrutura geral de um documento WSDL.

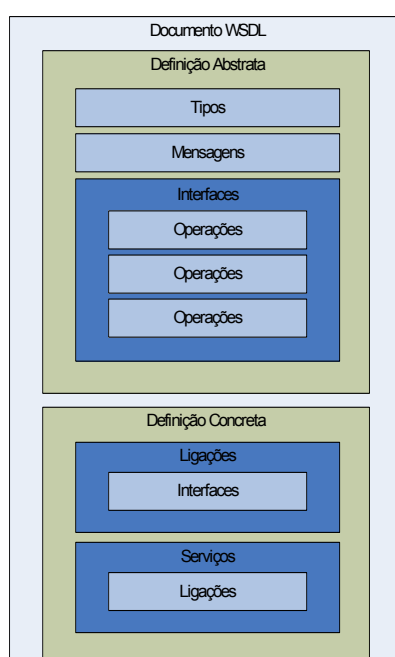


Figura 14 - Estrutura de um documento WSDL.

2.6.1 Definição dos Tipos

Nesta seção do documento WSDL são definidos os tipos de dados utilizados pelo serviço. Isto se faz necessário visto que um sistema de tipos qualquer não pode suportar todos os tipos possíveis, sendo assim este é um mecanismo para estender a capacidade da linguagem em relação a tipos de dados.

O sistema de tipos padrão utilizado no documento WSDL é aquele definido pelo esquema XML [27]. Por exemplo, se um determinado método necessitar de receber tipos inteiros ou mesmo um tipo *string*, não é necessário criar um novo elemento nesta seção. Basta utilizar os tipos **xsd:int** e **xsd:string**, respectivamente, considerando que foi utilizado o *namespace* (`xmlns:xsd = "http://www.w3.org/2001/XMLSchema"`).

Caso o método necessite de um tipo complexo, como por exemplo, os dados de um assinante de telefone, é necessário fornecer a definição deste novo tipo. Esta definição é feita utilizando um XML esquema [42] para definir um novo tipo complexo, conforme mostrado no exemplo da Figura 15.

```
<types>
  <xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <xsd:complexType name="assinante">
      <xsd:all>
        <xsd:element name="sNome" type="xsd:string"/>
        <xsd:element name="sFone" type="xsd:string"/>
        <xsd:element name="sEndereco" type="xsd:string"/>
        <xsd:element name="sBairro" type="xsd:string"/>
        <xsd:element name="sCidade" type="xsd:string"/>
        <xsd:element name="sRamoAtividade" type="xsd:string"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:schema>
</types>
```

Figura 15 - Exemplo de tipo complexo utilizando XML esquema.

Tantos os tipos padrão como os tipos definidos podem ser mapeados diretamente para linguagens de programação, como a linguagem Java, por exemplo. Desta forma, é possível a conversão XML para um tipo padrão da linguagem e vice-versa.

2.6.2 Definição das Mensagens

Os *Web Services* trocam mensagens entre si. Cada mensagem contém uma série de tipos de dados que são enviados ou recebidos. Este elemento da linguagem permite definir quais são as mensagens utilizadas por estes *Web Services*.

Um documento WSDL pode conter tantos elementos quanto necessários; um elemento deste tipo contém um atributo **name**, que deve ser único em todo o documento WSDL, e um

conjunto de elementos lógicos chamados **part**. O elemento **part**, por sua vez, contém um atributo **name** que deve ser único em uma mensagem e um atributo **type** que indica o tipo de dado utilizado neste elemento. A Figura 16 mostra a definição de algumas mensagens utilizadas em um documento WSDL.

```
<!-- message declns -->
<message name="empty"/>
<message name="GetAssinantesListRequest">
  <part name="cidade" type="xsd:string"/>
  <part name="ramoAtividade" type="xsd:string"/>
</message>
<message name="GetAssinantesListResponse">
  <part name="assinantes" type="s:ArrayOfAssinante"/>
</message>
<message name="GetRamosAtividadesListResponse">
  <part name="ramosAtividades" type="s:ArrayOfString"/>
</message>
<message name="GetCidadesListResponse">
  <part name="cidades" type="s:ArrayOfString"/>
</message>
```

Figura 16 - Exemplo de mensagens definidas em um documento WSDL.

No exemplo mostrado na Figura 16, um elemento **<message>** não contém elementos filho do tipo **<part>**, isto é perfeitamente possível e indica uma mensagem que não recebe ou que não retorna valores.

2.6.3 Definição das interfaces

Os *Web Services* podem ser vistos como uma interface, que oferece uma série de operações. Estas operações podem conter mensagens de entrada e/ou saída. Neste ponto pode-se comparar a interface como um conjunto de métodos da programação orientada a objetos e cada método pode receber e/ou enviar mensagens.

A versão 1.1 da WSDL utiliza o elemento **<portType>** para indicar uma interface. Este elemento possui um atributo **name**, que deve ser único para esta interface, e possui um conjunto de operações.

Uma operação, indicada pela tag **<operation>**, contém um atributo **name** que representa o nome desta operação, e, além disso, este elemento pode conter os elementos filhos **<input>**, **<output>** e/ou **<fault>**. Estes elementos indicam as mensagens utilizadas pela operação para a entrada de parâmetros, o retorno da operação e um retorno em caso de falha, respectivamente. Cada um destes elementos possuem o atributo **message** que contém o nome de uma mensagem definida anteriormente. A Figura 17 mostra a definição de uma interface em um documento WSDL.

```

<portType name="PhoneListInterface">
  <operation name="getAssinantesList">
    <input message="tns:GetAssinantesListRequest"/>
    <output message="tns:GetAssinantesListResponse"/>
  </operation>
  <operation name="getRamosAtividadesList">
    <input message="tns:empty"/>
    <output message="tns:GetRamosAtividadesListResponse"/>
  </operation>
  <operation name="getCidadesList">
    <input message="tns:empty"/>
    <output message="tns:GetCidadesListResponse"/>
  </operation>
</portType>

```

Figura 17 - Exemplo de interface definida em um documento WSDL.

Na especificação da linguagem WSDL [43], versão 1.1, as combinações de uso dos elementos `<input>` e `<output>` podem levar aos tipos de operação mostrados na Tabela 2.

Tabela 2 - Tipos de operação que podem ser definidos em um documento WSDL 1.1.

Tipo de Operação	Ordem dos Elementos	Significado
Pedido-Resposta	<code><input></code> <code><output></code>	O consumidor do serviço envia uma mensagem, que produzirá uma resposta ou então caso ocorra alguma falta a resposta indica esta situação.
Solicitação-Resposta	<code><output></code> <code><input></code>	Aqui ocorre o inverso do tipo de operação Pedido-Resposta. O provedor do serviço envia uma mensagem para o consumidor, solicitando uma resposta por parte do mesmo.
Ativação	<code><input></code>	A operação consiste em enviar um valor para o serviço, sem a necessidade de obter uma resposta. Este tipo de operação é útil para uma serviço do tipo assíncrono.
Notificação	<code><output></code>	Neste tipo de operação o serviço notifica o consumidor sobre algum evento ocorrido

Apesar da especificação citar os tipos acima, os tipos de operações Solicitação-Resposta e Notificação não são completamente definidos [44], visto que seriam necessários outros mecanismos para especificar complementemente tais operações. Outras especificações se propõem a resolver tais problemas, como por exemplo, a especificação *WS-BaseNotification* [45].

2.6.4 Definição das ligações

Até este ponto, o documento WSDL contém as definições das operações de forma abstrata sem nenhuma relação com a forma através da qual elas serão transmitidas. Cada interface (elemento **<portType>**) pode possuir vários tipos de ligação associados à mesma, desta forma, é possível reutilizar estes elementos.

Um elemento **<binding>** possui um atributo **name**, que deve ser único no documento WSDL, e um atributo **type**, que contém o nome de uma interface definida anteriormente. O elemento **<binding>** contém elementos filhos do tipo **<operation>**, porém neste caso, é possível associar em cada parte da operação detalhes específicos de cada protocolo, ou seja, a entrada (**<input>**), a saída (**<output>**) e o retorno em caso de um erro (**<fault>**).

A linguagem define ligações para os protocolos SOAP sobre HTTP, HTTP GET/POST e MIME. Em cada uma destas ligações, elementos específicos são inseridos no elemento nos vários elementos filhos de um **<binding>**, a fim de caracterizar o transporte. A Figura 18 mostra um exemplo de elemento **<binding>** utilizando o protocolo SOAP sobre HTTP.

```
<binding name="PhoneListSOAPBinding" type="tns: PhoneListInterface">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getAssinantesList">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <soap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>
  <operation name="getRamosAtividadesList">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <soap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>
</binding>
```

Figura 18 - Elemento de ligação sobre SOAP em um documento WSDL.

Na ligação sobre o protocolo SOAP, mostrada na Figura 18, elementos específicos deste tipo de ligação são inseridos dentro dos elementos **<operation>**, **<input>**, **<output>** e **<fault>**. Estes elementos são definidos no *namespace* SOAP

(<http://schemas.xmlsoap.org/wsdl/soap/>) e basicamente indicam o estilo de envio das mensagens ("**rpc**" ou "**document**"), uma ação que pode ser disparada antes do processamento de cada operação (**soapAction**) e um esquema de codificação que é usado para o transporte dos dados.

2.6.4 Definição do Serviço

O elemento serviço descreve o serviço de forma pública. O atributo **name**, indica o nome do serviço que será visível a seus usuários. O serviço é caracterizado como um conjunto de elementos **<port>**.

Este elemento possui como atributos um nome (**name**), uma ligação (**<binding>**) e um elemento que indicará a localização desta porta, ou deste serviço. A Figura 19 mostra um elemento **<service>** para um serviço que utiliza a ligação SOAP sobre HTTP.

```
<service name="PhoneListService">
  <port name="PhoneListServices" binding="tns: PhoneListSOAPBinding ">
    <soap:address
      location="http://facom.ufu.br:0/axis/services/PhoneListServices"/>
    </port>
  </service>
```

Figura 19 - Definição do Serviço descrito em um documento WSDL.

Capítulo 3

Segurança em *Web Services*

Segurança é um aspecto crítico em muitos sistemas e, considerando as características dos *Web Services* este aspecto torna-se ainda mais relevante visto que os *Web Services* utilizam como transporte os protocolos da Internet.

Neste ambiente as comunicações são feitas pelas mais diversas plataformas e sobre diferentes contextos de segurança. Além disso, em certas aplicações é necessário que a segurança seja estabelecida não para comunicação ponto a ponto, mas uma comunicação fim a fim, em que as mensagens podem passar por vários nós (pontos) intermediários até chegar a seu destino.

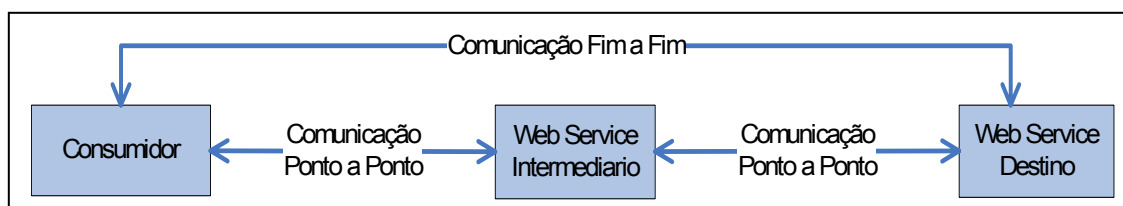


Figura 20 - Comunicação fim a fim versus comunicação ponto a ponto.

Neste capítulo, serão abordados os vários aspectos de segurança relativos aos *Web Services*, e nesta abordagem serão utilizados alguns exemplos a fim de contextualizar os requisitos de segurança.

Os *Web Services* disponibilizam uma parte de uma aplicação da empresa para outras aplicações remotas através da *Web*. O que ocorre aqui é o fato de que através dos *Web Services* é possível chegar até às camadas mais internas de software de uma empresa e seu uso indevido pode causar grandes danos.

Como mostrado na Figura 5, na arquitetura dos *Web Services* existe uma camada de segurança que contém protocolos que atendem aos requisitos de segurança próprios dos *Web Services*. Apesar dos requisitos básicos de segurança dos *Web Services* não serem diferentes dos requisitos gerais necessários para a comunicação entre sistemas, os mesmos possuem características próprias, visto que a segurança deve ser vista de forma distribuída, envolvendo as mais diferentes tecnologias.

Os requisitos básicos de segurança consistem em: integridade; confidencialidade; autenticação; autorização; e, não-repúdio.

3.1 Integridade

A garantia da integridade é um aspecto fundamental na segurança e consiste na capacidade de garantir que documentos ou mensagens estão intactos e não foram alterados de maneira deliberada, ou acidental, devido a erros durante o trânsito destas mensagens, por exemplo. O uso de assinaturas digitais é o meio mais comum para garantir a integridade.

A assinatura digital consiste na criação de um sumário (*hash*) a partir dos dados da mensagem, que é então criptografado utilizando uma chave privada e o resultado desta criptografia é então anexado à mensagem.

Para criar este sumário utiliza-se normalmente uma função *hash* não-inversível, que produz um resultado de tamanho fixo e cuja operação não pode ser desfeita, ou seja, a partir do sumário não é possível voltar aos dados.

Para verificar a assinatura é necessário utilizar uma chave pública, fornecida pelo assinante a fim de descriptografar o sumário anexado. Este sumário é então comparado com o outro sumário gerado pelo lado receptor. O receptor deve utilizar a mesma função *hash* utilizada inicialmente pelo assinante. No caso de modificação da mensagem, os sumários serão diferentes, indicando que a mensagem foi violada.

A especificação *XML-Signature* [46] define os mecanismos para suporte a criação e verificação de assinaturas digitais.

O uso do SSL seria suficiente para uma interação simples e síncrona entre um consumidor e um provedor de *Web Services*, pois neste caso o contexto de segurança envolve apenas duas partes e a duração desta interação equivale ao tempo de troca das mensagens de pedido-resposta.

Porém em uma situação mais complexa, como uma cadeia de *Web Services*, a integridade deve ser garantida em todo o percurso e para isto é necessária uma abordagem fim a fim e não ponto a ponto, como é o caso do SSL, conforme mostrado na Figura 20.

Além disso, em um nó, durante o processamento do pacote SOAP seu conteúdo pode ser alterado, neste caso o requisito é a garantia da integridade de certas partes da mensagem e não da mensagem como um todo.

A especificação *WS-Security*, é um padrão OASIS, que possui mecanismos para garantir a integridade de mensagens SOAP.

3.2 Confidencialidade

A confidencialidade é a capacidade de garantir que um conteúdo de uma mensagem não seja observado por partes alheias à comunicação. Para isto é necessário que as mensagens trafeguem entre os vários sistemas de forma sigilosa.

É importante fazer uma distinção, pois, basicamente, podem ocorrer duas situações: uma em que é necessário garantir a confidencialidade de mensagens e informações que estão em um meio ou ambiente público, neste caso o meio usualmente utilizado para garantir a confidencialidade é a criptografia; a outra situação é que a informação pode ser mantida de forma confidencial através de outros meios, como a autorização, por exemplo, ou seja, neste caso é possível manter as informações em um ambiente privado cujo acesso é controlado, garantindo desta forma a confidencialidade.

A especificação *XML-Encryption* [25] contém regras sobre como devem ser os processos de criptografar e descriptografar mensagens XML.

Em relação aos *Web Services*, é necessária a garantia da confidencialidade de uma maneira fim a fim, visto que uma mesma mensagem pode trafegar por muitos sistemas até chegar a um destino final. Desta forma o uso de SSL ou TLS não garante a confidencialidade, visto que nestes sistemas intermediários a mesma pode ser perdida, conforme ilustrado na Figura 21, supondo-se que nos *Web Services* intermediários exista alguém observando a comunicação.

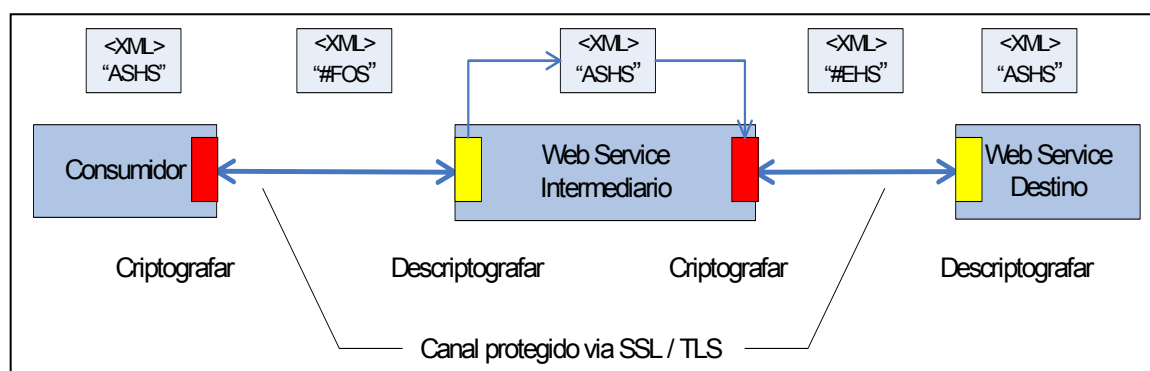


Figura 21 - Exemplo de uso de SSL / TLS em uma comunicação fim a fim.

Outro ponto interessante é que muitas das vezes uma mensagem não pode ser criptografada em sua totalidade, mas somente em algumas partes e isto inviabiliza o uso dos protocolos SSL / TLS para garantir o requisito da confidencialidade. A Figura 22 ilustra este fato.

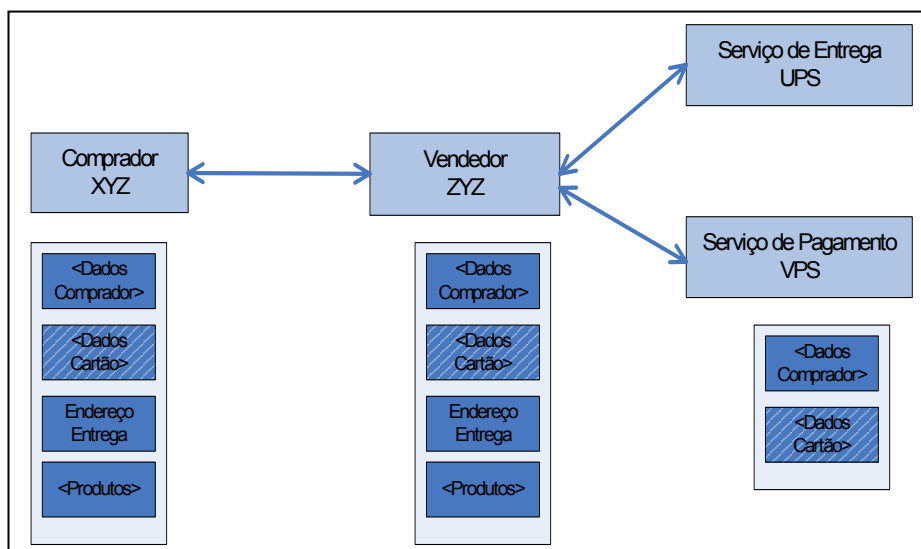


Figura 22 - Confidencialidade parcial da mensagem.

Este exemplo estende a situação mostrada na Figura 2. Aqui o sistema de *E-Procurement* (comprador) realiza uma compra de produtos, utilizando os *Web Services* do vendedor. A mensagem enviada do comprador para o vendedor possui seus dados de identificação, os dados do cartão de crédito para pagamento, o endereço de entrega e os produtos requisitados.

Os dados do cartão de crédito são enviados de forma confidencial e não é necessário que o vendedor tenha acesso a tais dados, visto que o único interessado nesta cadeia em visualizar e processar tais dados é o serviço de pagamento. Para o vendedor a única informação relevante é se a compra foi aprovada ou não e, em caso de aprovação, apenas os dados pertinentes à entrega devem ser repassados ao serviço de entrega de produtos.

Os mecanismos existentes na especificação *WS-Security*, atendem aos requisitos para garantir a confidencialidade de mensagens SOAP.

3.3 Autenticação

A autenticação envolve a capacidade de identificar e garantir identidade de uma parte envolvida na comunicação. Normalmente o consumidor do serviço precisa provar que ele realmente é quem afirma ser. Do mesmo modo, é importante que o comprador tenha certeza que o vendedor é quem afirma ser.

A autenticação é crítica, pois normalmente é a porta de entrada para a maioria dos sistemas, desta forma, um intruso, mal intencionado, poderia acessar os mais variados recursos e produzir grandes prejuízos.

O método tradicional para a autenticação de uma parte é através do uso de senhas, ou seja, a parte que deseja se autenticar mantém consigo um código, muitas das vezes de fácil

descoberta, e mediante uma identificação pessoal como seu nome, ou os dados de seu cartão de crédito informa a sua senha a fim de se autenticar.

No contexto dos *Web Services* e de sistemas distribuídos surgem várias questões relativas ao uso de senhas. Neste contexto, existem duas, ou mais, aplicações se comunicando remotamente através de *Web Services*. Para que uma aplicação possa se autenticar através de uma senha é necessário que a mesma seja enviada via uma conexão de rede, através da Internet. Caso esta conexão seja alvo de um monitoramento é possível obter a senha. Isto pode ser combatido através do uso de algum meio que garanta a confidencialidade.

Entretanto, surge aqui um outro problema. Normalmente as senhas são armazenadas em algum repositório de dados na autoridade responsável pela autenticação. Caso este repositório seja vítima de um ataque é possível obter as senhas e neste caso todo o esforço utilizado em segurança é perdido.

Além dos vários problemas de segurança ligados à autenticação existem ainda requisitos particulares dos *Web Services* em relação à autenticação, como por exemplo, a necessidade de que uma única autenticação (*Single Sign On*) seja feita com uma parte e que as informações sobre esta autenticação sejam transferidas para outras partes presentes na transação, conforme mostrado na Figura 23.

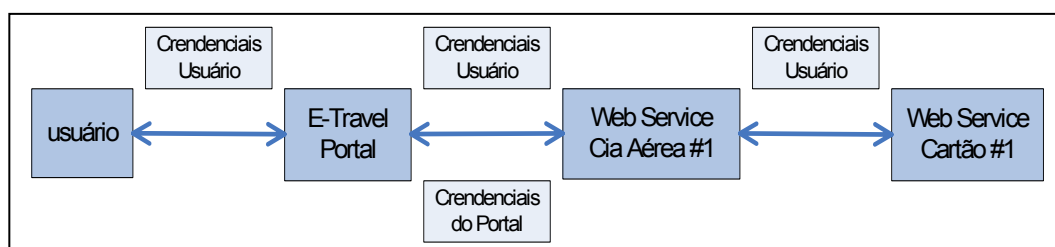


Figura 23 - Autenticação única em vários *Web Services*.

No exemplo acima, o usuário da plataforma de turismo (*e-WSTRAVEL*) precisa se autenticar para efetuar suas reservas e compras. Assim que um pacote for escolhido é necessário então realizar a compra da passagem aérea, por exemplo.

Neste caso as informações sobre a autenticação do usuário deverão ser encaminhadas à companhia aérea bem como as informações de autenticação da plataforma visto que a companhia aérea realizará duas operações: o encaminhamento da autenticação do usuário para sua operadora de cartão de crédito, a fim de autorizar o pagamento; e, o armazenamento das informações sobre o portal para que o mesmo seja comissionado em um momento posterior.

Outro aspecto importante em relação aos *Web Services* é a necessidade, em certos casos, de que a autenticação tenha uma longa duração e seja persistente. Isto é particularmente

válido para *Web Services* que funcionam de maneira assíncrona, conforme mostrado no segundo exemplo sobre *Web Services* na seção 2.1.

Neste caso, antes do envio da nota fiscal o servidor da empresa ZYZ, precisa se certificar que o sistema da empresa XYZ realizou sua autenticação de maneira correta, sendo que este sistema pode neste momento, inclusive estar desligado. É necessário que o servidor ZYZ armazene as informações sobre autenticação, para que possa verificá-la, antes do envio da nota fiscal.

A linguagem SAML, que é um padrão OASIS, especifica o uso de documentos XML para a troca de informações sobre autenticação, autorização e outros atributos relativos a um usuário de *Web Services* entre várias autoridades.

3.4 Autorização

Após autenticar uma entidade é necessário determinar quais operações esta entidade está autorizada a realizar. O processo de autorização passa então por estabelecer quais os privilégios que a entidade autenticada pode usufruir naquele domínio.

Juntamente com o conceito de autorização, existe o conceito das políticas disponíveis para usuários e também as políticas relativas à segurança. Todos estes componentes trabalham em conjunto a fim de que os recursos disponíveis sejam usados apenas pelas entidades que estejam habilitadas.

Outro aspecto importante em relação à autorização é o nível de serviço contratado. Por exemplo, a plataforma de turismo (*e-WSTRAVEL*), mostrada no terceiro exemplo da seção 2.1, pode oferecer três diferentes tipos de serviços a seus clientes: no primeiro nível o associado ao *e-WSTRAVEL* não paga pelo serviço e pode realizar a compra de seus pacotes.

No segundo nível, o usuário paga pelo serviço recebe em cada transação um desconto de cinco por cento que é revertido para compra de novos pacotes; no terceiro nível de serviço, para aqueles usuários que realizam uma grande movimentação a plataforma oferece além do desconto, uma passagem de ida gratuitamente em cada pacote negociado ao cliente. Neste caso existem três níveis de serviço que são oferecidos a cada cliente, conforme o seu perfil.

3.5 Não-Repúdio

O não-repúdio é a capacidade que uma entidade possui de garantir que determinada parte, realizou alguma operação e, portanto não pode negá-la em um momento posterior. Desta forma é possível afirmar que um determinado consumidor realizou as operações e deve se responsabilizar pelos resultados da mesma.

A autenticação tem um papel importante no não-repúdio, pois a garantia de que o cliente é ele mesmo é uma condição fundamental para o não repúdio. A outra é a capacidade de armazenar registros de segurança como autenticação, assinaturas, a fim de realizar uma auditoria futura no sistema.

3.6 *Web Services Security (WS-Security)*

A especificação *WS-Security*, proposta inicialmente pela Microsoft e IBM, foi adotada como padrão OASIS em março de 2004 [8]. Esta especificação trata basicamente da segurança das mensagens SOAP de uma maneira fim a fim, garantindo a integridade e a confidencialidade destas mensagens.

Seu objetivo é oferecer uma série de mecanismos a fim de construir protocolos de segurança, não tratando, por exemplo, de mecanismos para autenticação e da garantia de não-repúdio. As especificações *XML-Encryption* e *XML-Signature* são utilizadas maciçamente para atender a esses objetivos e podem ser usadas como protocolos básicos para segurança, como pode ser visto na Figura 5.

A técnica usada pela *WS-Security* consiste em adicionar informações sobre assinaturas digitais, criptografia, *timestamps* e *tokens* (fichas) de segurança dentro de um elemento, chamado **<wsse:Security>**, que está presente no cabeçalho da mensagem SOAP (**soap:header**).

Este elemento pode conter os seguintes elementos filhos: **UsernameToken**; **BinarySecurityToken**; **SecurityTokenReference**; **KeyInfo**; **Signature**; **ReferenceList**; e, **EncryptedData**. Estes elementos são opcionais e seu uso decorre da forma com a especificação é utilizada. A Figura 24 mostra a estrutura básica de uma mensagem SOAP que utiliza a especificação *WS-Security*.

O elemento **<UsernameToken>** é opcional e representa uma maneira de incluir o nome do usuário e uma senha. Este elemento pode ser utilizado para carregar informações de autenticação de um usuário.

O elemento **<BinarySecurityToken>** contém informações que não encaixam no formato XML e necessitam de um tipo especial de codificação, como por exemplo, certificados X.509 [47] e tíquetes Kerberos [48].

Um outra maneira de utilizar *tokens* de segurança é através do elemento **<SecurityTokenReference>**. Este elemento pode ser utilizado em certas ocasiões

onde é necessário que as informações sobre certificados sejam obtidas pela aplicação e estão localizadas em outro local.

```
<? Xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope"
  xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
  xmlns:ds="http://www.w3.org/2002/02/xmldsig#"
  xmlns:enc="http://www.w3.org/2002/03/xmlenc#">
  <env:Header>
    <wsse:Security
      env:role="http://www.w3.org/2002/06/soapenvelope/role/next"
      env:mustUnderstand="true">
        <wsse:BinarySecurityToken
          ...
        </wsse:BinarySecurityToken>
        <ds:Signature>
          <ds:SignedInfo>
            <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
            <ds:Reference URI="#MsgBody">
              <ds:DigestMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
              <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
            </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
          <ds:KeyInfo>
            <wsse:SecurityTokenReference>
              <wsse:Reference URI="#MyID" />
            </wsse:SecurityTokenReference>
          </ds:KeyInfo>
        </ds:Signature>
        <enc:EncryptedKey>
          ...
        </enc:EncryptedKey>
      </wsse:Security>
    </env:Header>
    <env:Body>
      <enc:EncryptedData>
        ...
      </enc:EncryptedData>
    </env:Body>
  </env:Envelope>
```

Figura 24 - Exemplo de mensagem SOAP utilizando *WS-Security*.

O elemento **<Signature>** contém informações sobre uma assinatura digital conforme definido na especificação *XML-Signature*.

O elemento **<ReferenceList>** é definido na especificação *XML-Encryption* e pode ser utilizado para identificar na mensagem as partes da mesma que se encontram criptografadas. Estas partes que se encontram criptografadas são expressas por elementos **<EncryptedData>**.

A fundação Apache possui o projeto WSS4J [49] que tem como objetivo a implementação da especificação *WS-Security*. Como um de seus resultados este projeto criará uma API (*Application Programming Interface*) para o desta especificação em aplicações baseadas na linguagem Java. Atualmente uma versão beta desta especificação já está disponível.

3.7 A Linguagem SAML

A linguagem *Security Assertions Markup Language* (SAML) especifica uma maneira para a troca de informações como autenticação, autorização e outros atributos referentes a um usuário entre diferentes infra-estruturas de segurança e aplicações.

A linguagem trata de questões como a autenticação, porém não estabelece meios para realizá-la, apenas especifica o tipo de autenticação realizada.

Uma asserção (**<saml:Assertion>**) é um conjunto de informações que contém uma ou mais expressões feitas por uma autoridade SAML.

A linguagem define três tipos de assertivas. Assertivas de autenticação (**<AuthenticationStatement>**) que descrevem como o usuário foi autenticado, em que hora, por exemplo. Assertivas de atributos (**<AttributeStatement>**) que especificam detalhes sobre o usuário e finalmente assertivas sobre autorização (**<AuthorizationStatement>**) que identificam quais ações o usuário pode de executar em um sistema específico.

Uma autoridade SAML é uma entidade que emite asserções. A entidade que recebe estas asserções (*relying part*) normalmente possui uma relação de confiança com a autoridade SAML e decide o que fazer baseado nas asserções recebidas.

A linguagem define um protocolo do tipo pedido/resposta para obter asserções. Um pedido SAML pode conter uma solicitação de uma asserção conhecida ou então realizar consultas sobre asserções de autenticação, atributos ou autorização, sendo que a resposta devolve as asserções solicitadas. A linguagem define uma ligação com o protocolo HTTP.

Uma importante característica da linguagem SAML é que as asserções podem ser trocadas entre diferentes empresas, sistemas e plataformas. Desta forma, uma asserção sobre autenticação, por exemplo, pode ser enviada a outros *Web Services*. Esta propriedade é um importante requisito para resolver o problema da autenticação única (*Single Sign On*).

3.8 O Protocolo *Secure Remote Password* (SRP)

O protocolo SRP é o mais recente protocolo [50] para a autenticação baseada no uso de senhas. Este protocolo oferece uma maneira segura para autenticar um cliente sem os riscos de um ataque de dicionário, presentes nos sistemas tradicionais baseados em senhas [51].

De uma maneira geral, o ataque ao dicionário, consiste em monitorar a comunicação e obter a senha, mesmo que criptografada, ou outra informação relativa à mesma e a partir desta informação realizar um ataque sistemático, testando todas as várias possibilidades de senha até que finalmente o intruso possa se autenticar. Normalmente este ataque é feito através de software.

Outra importante característica deste esquema de autenticação é que a senha nunca é enviada através da rede, evitando desta forma que um intruso, através do monitoramento da rede obtenha a senha ou alguma informação que permita a sua reconstrução.

Neste protocolo a senha não é armazenada nem como texto plano, ou seja, sem criptografia, e também não armazenada de forma criptografada. Nesta abordagem o que é armazenado no servidor é um verificador (✓) que é obtido a partir do uso de uma função *hash* não-inversível que tem como um dos parâmetros a senha original. A seção seguinte aborda os detalhes de funcionamento do protocolo SRP.

A Tabela 3 mostra uma comparação entre os vários protocolos de autenticação baseados em senha.

Tabela 3 - Comparação entre os protocolos de autenticação.

CATEGORIA	PROTOCOLO	PUBLICAÇÃO	VULNERABILIDADES CONHECIDAS
Autenticação forte utilizando senhas	Secure Remote Password (SRP-6) [52]	2002	NA
	Secure Remote Password (SRP-3) [50]	1997	[52]
	Extended Strong Password Exponential Key Exchange (B-SPEKE) [53]	1997	[50]
	Strong Password Exponential Key Exchange (SPEKE) [54]	1996	[50]
	Augmented Encrypted Key Exchange (A-EKE) [55]	1994	[50]
	Encrypted Key Exchange (EKE) [56]	1992	[50]
Falsa autenticação forte utilizando senhas	SSH Public Key Authentication [57]	1998	[3]
	SSH Password Authentication [57]	1998	[3]
	Password-protected Client-side SSL/TLS Certificates [58]	1999	[3]
	Pré-autenticação no Kerberos V5 [48]	1994	[62]
Autenticação fraca	Cleartext passwords	NA	NA
	HTTP Basic Authentication [59]	1997	[63]
	HTTP Digest Authentication [60]	1997	[64]
	Kerberos V4 [61]	1988	[65]

De maneira geral, os protocolos da categoria “Autenticação forte utilizando senhas”, em maior ou menor grau, protegem senhas de fácil suposição, como por exemplo, a sequência “123”, de ataques tanto “*on-line*”, quanto “*off-line*”, mesmo em sistemas em que o invasor possui um acesso total à rede.

No ataque “*off-line*”, também chamado “ataque de dicionário”, o invasor possui algumas informações e dispõe de tempo para tentar acessar o sistema, este ataque é feito não no momento em que um usuário autêntico está tentando se conectar, mas, em um momento posterior, e normalmente o invasor utiliza informações que podem ter sido capturadas anteriormente, como uma senha criptografada, por exemplo.

No ataque “*on-line*” a tentativa de obter a senha se faz de várias formas, como por exemplo, o intruso pode fingir ser o sistema de destino e receber informações sobre a autenticação no momento em que um usuário real tenta se autenticar. Uma outra forma de ataque “on line” é conhecida como “*man-in-the-middle*”. Neste caso, no momento de uma autenticação o invasor está entre o usuário e o sistema de destino, capturando as informações relativas à autenticação.

A categoria chamada “Falsa autenticação forte” é uma categoria intermediária, onde os protocolos possuem alguns mecanismos mais elaborados para autenticação, porém apresentam problemas de segurança em relação a ataques conhecidos, como por exemplo: a vulnerabilidade a ataques “*off-line*” e “*on-line*”; ataque “*man-in-the-middle*”; entre outros, em maior ou menor grau.

Esta categoria apresenta um grande perigo, pois o usuário possui a falsa impressão de que está protegido, quando isto na realidade pode não acontecer. Os métodos usados nesta categoria estão disponíveis em muitas aplicações comerciais, mesmo em *Web Services*.

Finalmente na última categoria, chamada “autenticação fraca”, existem protocolos que são muito utilizados, muitas das vezes, devido à sua simplicidade e compatibilidade com sistemas legados. Os métodos de autenticação desta categoria não oferecem praticamente nenhuma proteção em caso de ataques, ou seja, neste caso o invasor facilmente obterá informações de autenticação de um usuário.

Este trabalho não tem por objetivo detalhar as características dos protocolos acima e realizar uma análise profunda sobre cada um deles. O importante a ser notado é que o protocolo SRP é um dos mais recentes e apresenta uma série de propriedades que lhe permite resistir a vários tipos conhecidos de ataques. As referências mostradas na Tabela 3 podem ser consultadas para maiores informações sobre estes protocolos.

3.9 O Funcionamento do protocolo *Secure Remote Password* (SRP)

O protocolo SRP baseia-se em autenticação forte, utilizando senhas. Uma característica deste protocolo é que a autenticação de um usuário é feita através de uma troca de mensagens entre o usuário e a entidade que o autenticará.

Outra característica é que a senha não é trocada, em nenhum momento entre as partes, e, além disso, a entidade que é responsável pela autenticação não mantém armazenada esta senha. Esta característica é importante para a garantia da eficiência deste protocolo, visto que a rede por onde as mensagens trafegam pode ser totalmente insegura, o que é desejável em se tratando de *Web Services* e a Internet.

A entidade que autenticará o cliente, aqui chamada servidor, possui internamente um valor, chamado verificador (**v**) que é obtido a partir da senha (**p**), porém, não é possível reconstruir a senha a partir deste valor.

Para o cálculo do verificador (**v**) é utilizado valor temporário (**x**) e um gerador (**g**). O valor temporário (**x**) é resultante da aplicação da função *hash* (**H**) que recebe como parâmetros: o nome do usuário (**U**) e um valor aleatório, chamado salto (**s**), que é obtido antes do cálculo do verificador e a senha (**p**). A Figura 25, mostra como é feito o cálculo do verificador. Todas as operações são executadas em módulo N, sendo este um número primo de alto valor.

$$\begin{aligned} \mathbf{x} &= \mathbf{H}(\mathbf{s}, \mathbf{U}, \mathbf{p}) \bmod \mathbf{N} \\ \mathbf{v} &= \mathbf{g}^{\mathbf{x}} \bmod \mathbf{N} \end{aligned}$$

Figura 25 - Cálculo do verificador no protocolo SRP.

Em sua construção o protocolo considera que mesmo que o usuário escolha uma senha fraca, ou seja, de fácil descoberta, isto não afetará o desempenho do protocolo.

Durante o processo de autenticação, chaves públicas temporárias são trocadas entre o servidor e o cliente. Estas chaves são diferentes a cada tentativa de autenticação, a Figura 26, mostra o processo de autenticação deste protocolo.

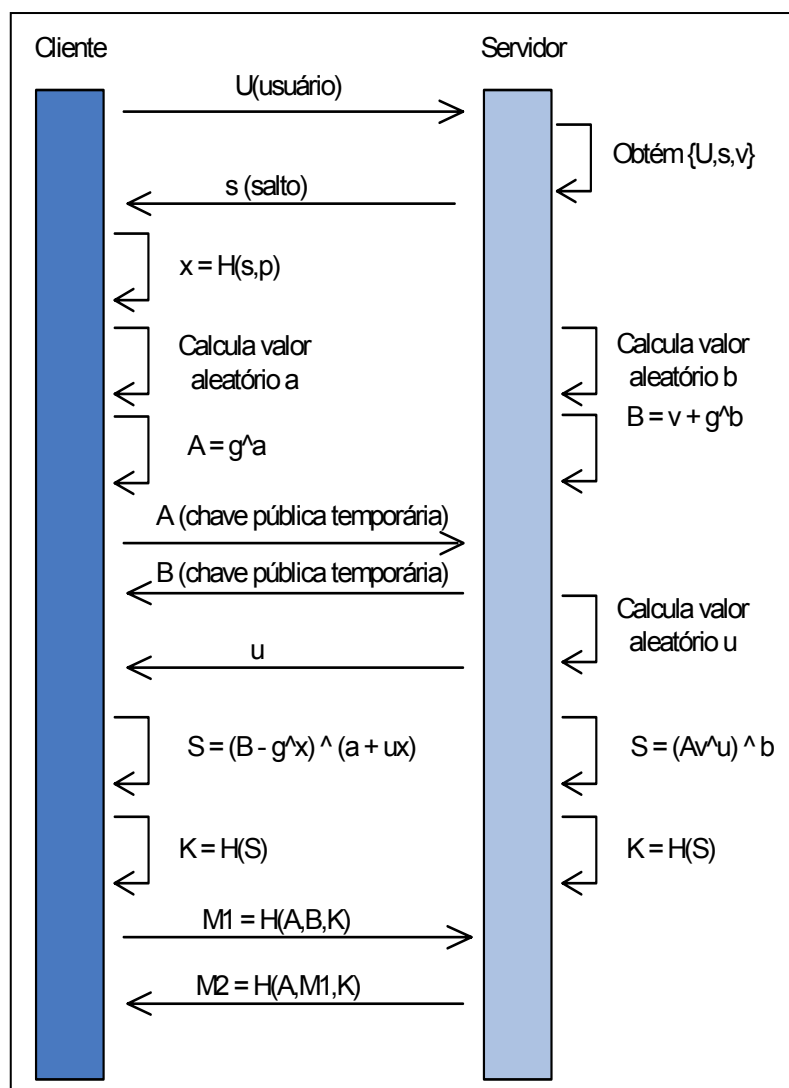


Figura 26 - Troca de mensagens do protocolo SRP.

Inicialmente o cliente envia para o servidor o seu nome de usuário (U). A partir deste nome o servidor realiza uma consulta em seu banco de dados e obtém o verificador (v) e o salto (s). O verificador (v) será utilizado no lado servidor para realizar uma série de operações e o salto (s) é enviado para o cliente.

Após receber o salto (s) o cliente calcula a sua chave privada (x). Esta chave privada é calculada utilizando uma função *hash* não-inversível (H) que recebe como parâmetros o salto (s) e a senha (p), informada pelo cliente. Em seguida o cliente gera uma chave privada temporária (a). Usando o gerador (g) e sua chave privada (a) o cliente calcula uma chave pública (A). Enquanto isto o servidor calcula a sua chave privada temporária (b). As chaves privadas temporárias criadas tanto pelo cliente quanto pelo servidor são aleatórias e não são publicamente reveladas.

O servidor calcula a sua chave pública temporária (**B**). Este cálculo é feito usando o verificador (**v**), obtido no banco de dados do servidor, o gerador (**g**) e a sua chave privada temporária (**b**).

Neste ponto o cliente e o servidor trocam suas chaves públicas temporárias. Usando diferentes métodos e as chaves públicas trocadas entre si, cliente e servidor, calculam uma chave de sessão (**S**), se a senha estiver correta estas chaves serão idênticas. A chave de sessão é então transformada em uma chave forte através do uso de uma função *hash* sobre a mesma.

Para mostrar uma evidência de que contém a chave de sessão correta, o cliente calcula um valor de comparação (**M1**) e o envia para o servidor. Em seguida o servidor, calcula este valor de comparação e verifica se o mesmo é idêntico àquele enviado pelo cliente.

Caso os valores sejam idênticos o servidor, calcula um segundo valor de comparação (**M2**) e o envia ao cliente, como uma evidência de que ele também possui a chave de sessão correta. Ao receber este valor, o cliente também calcula este segundo valor de comparação (**M2**) e caso ambos seja iguais o cliente será autenticado com sucesso. Assim que o cliente estiver autenticado, é possível iniciar uma sessão de comunicação.

Capítulo 4

Autenticação de Acesso em *Web Services*

No estágio atual de desenvolvimento dos *Web Services*, o transporte de mensagens através do protocolo SOAP e a descrição através da linguagem WSDL são padrões de fato de toda a indústria de software. Entretanto, à medida que avançamos nas camadas superiores da arquitetura, muitos padrões ainda não estão aceitos de fato.

O controle de autenticação é requisito imprescindível para a segurança dos *Web Services*, e como visto anteriormente, este controle faz uso do conceito de garantia de integridade e confidencialidade.

Isto se reflete perfeitamente na arquitetura de camadas dos *Web Services*, mostrada na Figura 5, visto que a especificação *WS-Security* garante a integridade e a confidencialidade baseando-se nas especificações *XML-Encryption* e *XML-Signature*. A especificação SAML, que está em uma camada superior, utiliza como base a especificação *WS-Security*. Esta especificação pode garantir se uma entidade está autenticada ou não, porém não contém o mecanismo que deve ser utilizado para autenticar o usuário de um serviço.

Outro ponto importante a ser considerando é que os processadores SOAP possuem como função básica o transporte de mensagens SOAP, o processamento de cabeçalhos e a implementação de padrões de troca de mensagens, porém não estabelecem meios para se garantir a confidencialidade, a integridade e a autenticidade da outra parte.

Em relação à segurança e aos requisitos exigidos pelos *Web Services*, a autenticação, por exemplo, é um problema, visto que não existe um mecanismo padrão para ser utilizado a fim de garantir uma autenticação segura para *Web Services*.

A Tabela 4 mostra alguns processadores SOAP e os recursos que os mesmos disponibilizam para autenticar a outra parte.

Tabela 4 - Tecnologias de autenticação utilizadas por alguns processadores SOAP.

NOME	RESPONSÁVEL	SOAP	LINGUAGEM	VERSÃO ATUAL	OPEN SOURCE	TECNOLOGIAS PARA AUTENTICAÇÃO
Apache AXIS [66]	Apache	1.1 1.2	Java C++	1.2 (16/11/2004)	Sim	HTTP Basic authentication Client certificates over HTTPS
SOAP::Lite for Perl [67]	Pavel Kulchenko	1.1 1.2	Perl	0.65 Beta 2	Sim	HTTP Basic authentication Digest HTTP authentication
gSOAP [68]	Robert A. van Engelen	1.1 1.2	C C++	2.7 (10/10/2004)	Sim	SSL (HTTPS) HTTP/1.1 keep-alive Chunking Authentication
IBM WebSphere SDK for Web Services (WSDK) Version 5.1 [69]	IBM	1.1 1.2	Java	5.1	Não	HTTP Basic authentication Digest HTTP Authentication IDAssertion Signature LTPA LDAP Kerberos
Java Web Services Developer Pack (JWSDP) [70]	Sun	1.1 1.2	Java	1.5 (10/11/2004)	Não	None HTTP Basic authentication Client-certificate authentication Digest authentication Form-based authentication
Microsoft .NET Framework [71]	Microsoft	1.1 1.2	C++ C# ASP VB	1.1	Não	Basic Basic over SSL Digest Integrated Windows(NTLM or Kerberos) Client Certificates SOAP headers
BEA WebLogic Workshop [72]	Bea	1.1 1.2	Java	8.1	Não	HTTP Basic authentication Basic HTTP over SSL SOAP Message Security
webMethods GLUE [73]	webMethods	1.1 1.2	Java	4.0.1 (05/02/2003)	Não	Basic HTTP Authentication Digest HTTP Authentication LDAP SOAP Message Security
Systinet Server for Java, 5.0 (WASP Server for Java) [74]	Systinet	1.1 1.2	Java	5.0	Não	HTTP Basic Authentication HTTP Digest Authentication Kerberos SOAP Message Security

Ao se observar a Tabela 3 e a Tabela 4 é possível constatar que os métodos para autenticação utilizados pelos processadores SOAP, mostrados na Tabela 4, se enquadram, na maioria dos casos, nas categorias “Autenticação fraca” ou “Falsa autenticação forte utilizando senhas”. A Tabela 4 contém os processadores SOAP mais significativos existentes atualmente no mercado.

Considerando estes fatos, propõe-se uma solução para autenticar o acesso a *Web Services*, utilizando um protocolo de autenticação forte baseado em senhas. A solução se baseia nas seguintes premissas: primeira, o protocolo SRP será utilizado para autenticação visto suas características técnicas; segunda, este protocolo deverá ser estendido e convertido a fim de que possa funcionar segundo a filosofia de trabalho dos *Web Services*; e, terceira, a solução deve ser simples, de fácil implantação e deve ser utilizável com qualquer processador SOAP. Para efeito de implementação neste trabalho foi utilizado o Apache AXIS.

4.1 O Apache AXIS

O apache AXIS (*Apache Extensible Interaction System*) é um processador SOAP, de código fonte aberto, mantido pela fundação Apache. Este sistema oferece a infra-estrutura necessária para a construção de aplicações que manipulam mensagens SOAP, ou seja, *Web Services*.

O AXIS é construído sobre o conceito de *handlers*. Essencialmente um *handler* é um objeto que executa uma tarefa específica. Outro conceito importante na arquitetura do AXIS são as *handler chains*. Uma *handler chain* é um tipo especial de *handler* que pode conter outros *handlers* e outras *handler chains*.

A arquitetura do AXIS consiste de três *handler chains* - chamadas *Transport*, *Global* e *Service* – que processam as mensagens. Quando o AXIS está processando uma mensagem, uma série de *handlers* são executados em uma sequência pré-definida. Esta sequência é determinada por dois fatores: a configuração utilizada na implantação do serviço; e, se o AXIS está sendo executado como cliente, ou como servidor. A Figura 27 mostra a arquitetura do AXIS.

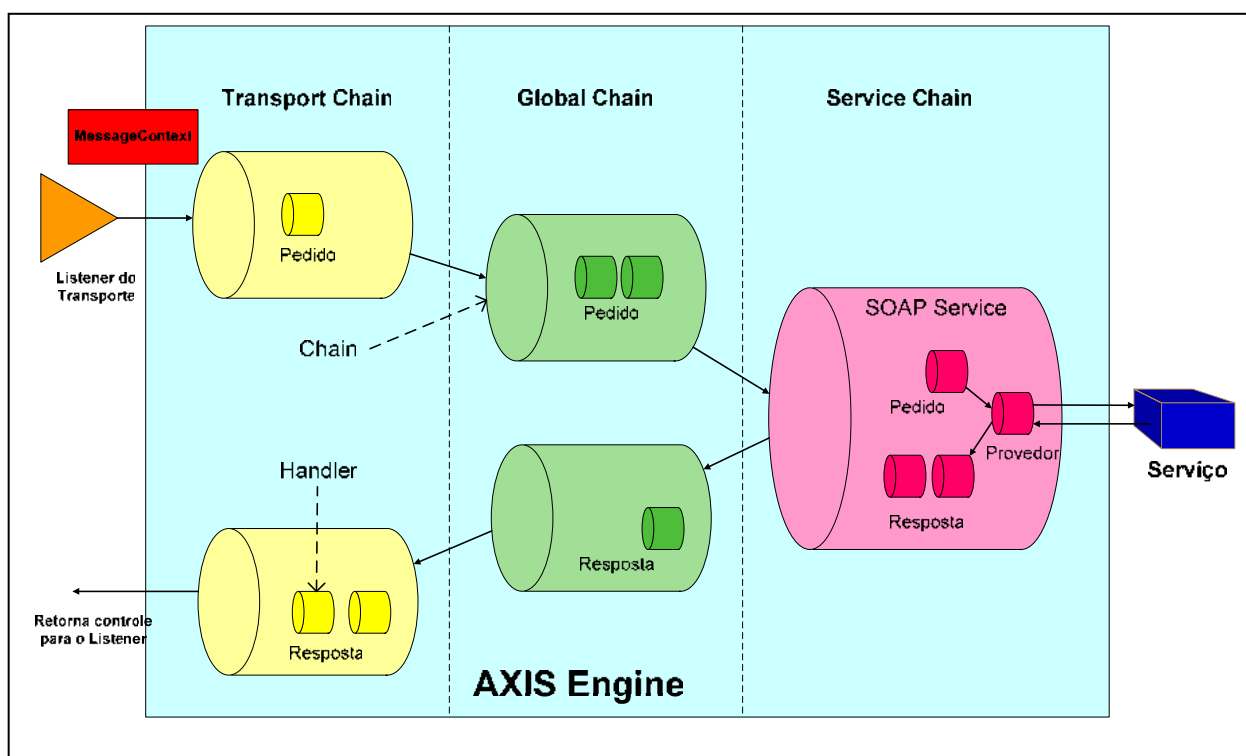


Figura 27 - Arquitetura do Apache AXIS.

Quando uma mensagem é recebida, o ouvinte (*Listener*) do transporte criará um objeto *MessageContext* [75]. Este objeto contém todas as propriedades referentes ao processamento do serviço, incluindo toda a mensagem SOAP, ou seja, seu cabeçalho (*Header*) e o seu corpo (*Body*).

Este objeto é então transferido para o AXIS (*AXIS Engine*) que o enviará para a cadeia de *handlers* do transporte (*Transport Chain*). A sequência de *handlers* existentes nesta cadeia será invocada, conforme a ordem configurada e cada *handler* receberá e possivelmente transformará este objeto durante seu processamento.

Este processo se repetirá com a cadeia global (*Global Chain*), caso a mesma esteja configurada, e finalmente com a cadeia de *handlers* existentes para o serviço (*Service Chain*) onde finalmente este objeto será recebido pelo *handler* provedor, que, finalmente executará a lógica existente no serviço.

Uma importante característica do AXIS é que esta sequência de *handlers* pode ser modificada e novos *handlers* podem ser adicionados, modificando o comportamento do AXIS, desta forma a existência ou não destas cadeias e os *handlers* nelas contidos dependem exclusivamente da configuração realizada em uma determinada implantação do AXIS.

4.2 A Solução Proposta

Considerando os objetivos estabelecidos anteriormente e as características do protocolo SOAP, do Apache AXIS e do protocolo SRP, foi concebida a proposta, mostrada na Figura 28, para a autenticação segura de *Web Services* [76].

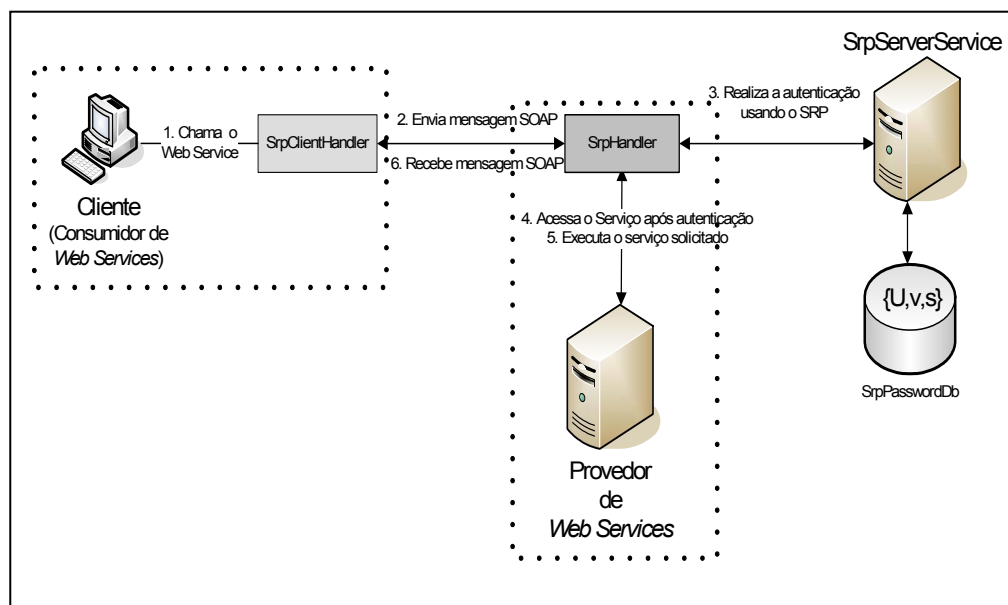


Figura 28 - Solução proposta para autenticação segura.

Na solução proposta, um *handler* do AXIS (**SrpClientHandler**) é adicionado ao fluxo de mensagens SOAP e será executado pelo cliente. Este *handler* será parte da *chain* global do cliente utilizada durante o pedido, ou seja, este *handler* será executado sempre antes do envio da mensagem SOAP que solicita algum serviço por parte do provedor.

Ao ser chamado, este *handler* adicionará à mensagem SOAP original um conjunto específico de elementos XML. Este elemento será adicionado ao cabeçalho da mensagem SOAP e conterá o nome do usuário e a senha que será utilizada no processo de autenticação.

A Figura 29 mostra o fragmento de uma mensagem SOAP após ser processada pelo *handler* **SrpClientHandler**.

```

<soapenv:Header>
  <srpAuth:SrpAuthentication>
    <SprAuthenticationRequest>
      <user>flavio</user>
      <password>123</password>
      <sessionId><sessionId>
    </SprAuthenticationRequest>
  </srpAuth:SrpAuthentication>
</soapenv:Header>
  
```

Figura 29 - SOAP header adicionado à mensagem.

Uma consideração feita nesta solução é que a mensagem SOAP será transportada no primeiro trecho em um canal “seguro”. Isto pode ser conseguido de duas formas: através do uso da especificação *WS-Security* a fim de garantir a confidencialidade e a integridade da mensagem, ou, dependendo da situação é possível utilizar o protocolo HTTP juntamente com o protocolo SSL. Esta última suposição é válida, pois a comunicação entre o cliente (**SrpClientHandler**) e o *handler* do servidor de *Web Services* (**SrpHandler**) é ponto a ponto.

Da mesma forma um *handler* será adicionado à *chain global* do provedor de *Web Services*. Deste modo sempre que uma mensagem for recebida por este servidor, este *handler*, chamado **SrpHandler**, será executado antes do processamento da mensagem SOAP. A Figura 30 mostra o *handler* SrpHandler adicionado cadeia global do AXIS.

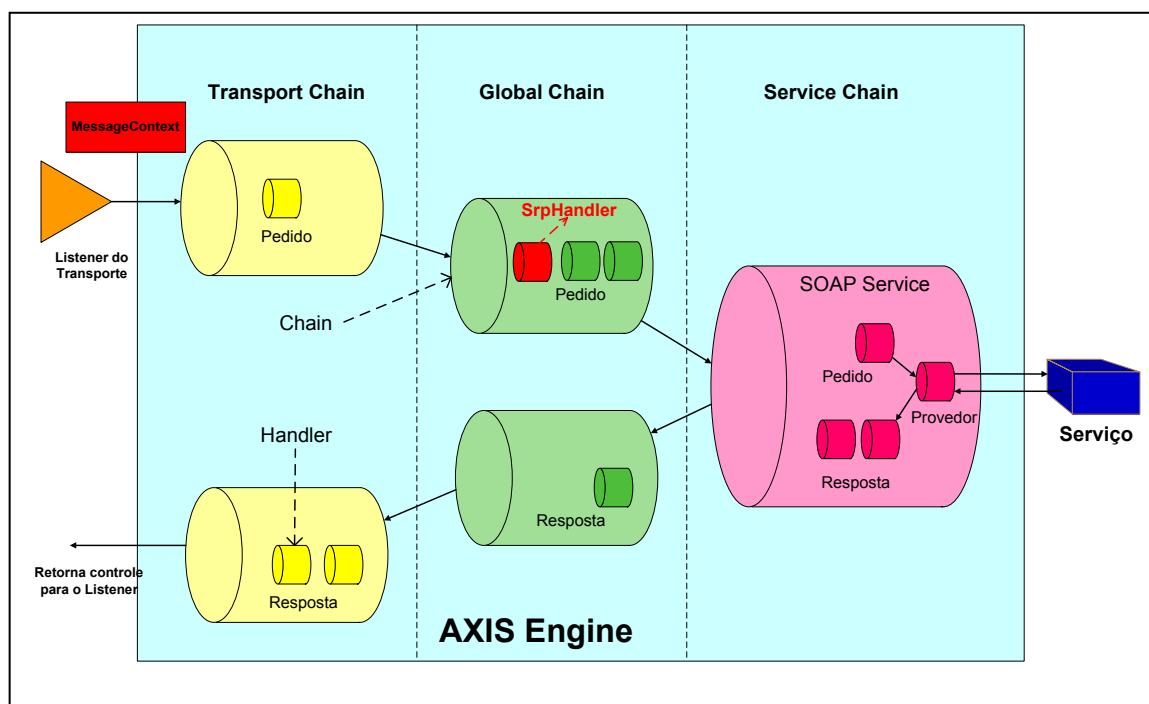


Figura 30 - SrpHandler adicionado a cadeia global do AXIS.

Este *handler* será o processo par (*peer*) do *handler* **SrpClientHandler**, ou seja, este *handler* capturará todas as mensagens recebidas e recuperará o *header block* desta mensagem, adicionado à mesma, inicialmente, por seu par.

Assim que for executado o *handler* **SrpHandler** recebe um objeto *MessageContext*. Este objeto contém a mensagem SOAP que foi modificada inicialmente. A partir deste objeto é possível recuperar o *header block* que será analisado a fim de verificar se o mesmo é um elemento XML válido e que contenha as informações necessárias para a autenticação do cliente.

Em caso de erro durante este processamento, devido, por exemplo, a não existência do nome do usuário, este *handler* disparará um mensagem SOAP que indicará uma falta (**<soap:Fault>**) e o processamento será abortado.

É importante ressaltar que o código que tem acesso à senha e ao nome de usuário é o *handler* **SrpHandler**, e considerando que a mensagem chegou até este ponto de forma segura, esta informação está até aqui protegida, e deverá ser removida do pedido SOAP, assim que o *handler* terminar o seu processamento. Para garantir a integridade do *handler* é possível criar um *hash* a partir do seu código e armazenar esta informação juntamente com o mesmo.

Considerando que o *header block* é válido, o *handler* **SrpHandler** iniciará a troca de mensagens necessárias para a execução do protocolo SRP a fim de verificar se o acesso aos *Web Services* solicitados pode ou não ser efetivado.

Na solução proposta, o outro lado, ou seja, a entidade par para este *handler*, que executará o lado servidor do protocolo SRP, será um servidor de *Web Services*. Desta forma a troca de mensagens previstas pelo protocolo SRP será executada como uma troca de mensagens SOAP que se realizará utilizando o padrão pedido-resposta. A Figura 31, mostra as mensagens trocadas entre o **SrpHandler** e os *Web Services* do lado servidor, aqui chamados **SrpServerService**.

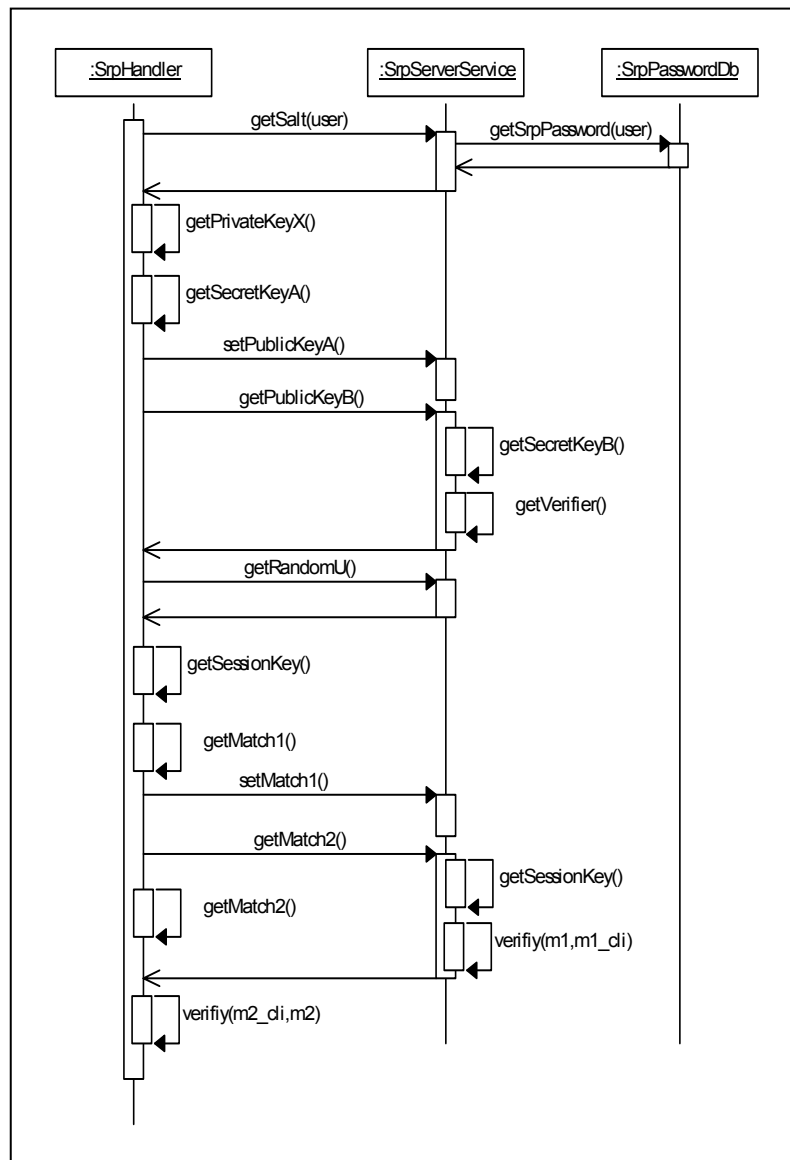


Figura 31 - Troca de mensagens do protocolo SRP como chamadas a *Web Services*.

A troca de mensagens é a base do protocolo SRP. Em duas seqüências de autenticação de um mesmo usuário, os valores utilizados nestas seqüências não se repetirão, ou seja, caso esta troca seja forjada em outro momento os valores serão diferentes, garantindo a validade do protocolo SRP.

Outro ponto interessante é que esta troca pode ser feita em um canal inseguro e não é necessária a utilização de outros elementos a fim de proteger este troca de mensagens, sendo possível a utilização de mensagens SOAP sem nenhuma proteção das mesmas. No caso de uma autenticação com sucesso, a última operação – **verify(m2,m2_cli)** – retornará um valor verdadeiro indicando que a autenticação ocorreu com sucesso.

Neste ponto o *handler* **SrpHandler** removerá da mensagem SOAP recebida inicialmente o *header block* que contém as informações de autenticação e inserirá na mensagem uma chave de sessão, que foi criada pelo protocolo SRP. Esta chave é a garantia da autenticação do usuário que terá uma duração pré-definida a fim de garantir que o usuário realize a autenticação sempre que necessário.

Para realizar as operações acima, o protocolo SRP foi estendido, e a sua troca de mensagens realizada como chamadas de *Web Services*, ou seja, mensagens SOAP. A fim de criar estes serviços foi necessário definir as primitivas do protocolo utilizando a linguagem WSDL. Esta definição descreve completamente as primitivas do protocolo e as chamadas de *Web Services*. A Figura 32, Figura 33 e a Figura 34, mostram este código.

```
<?xml version="1.0" ?>
<definitions name="urn:srpserver"
    targetNamespace="urn:srpserver"
    xmlns:tns="urn:srpserver"
    xmlns:typens="urn:srpserver"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:s="http://soapinterop.org/xsd"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <!-- type defs -->
    <!-- message declns -->
    <message name="empty"/>
    <message name="GetSaltRequest">
        <part name="user" type="xsd:string"/>
    </message>
    <message name="GetSaltResponse">
        <part name="salt" type="xsd:string"/>
    </message>
    <message name="SetPublicKeyARequest">
        <part name="A" type="xsd:string"/>
    </message>
    <message name="GetPublicKeyBResponse">
        <part name="B" type="xsd:string"/>
    </message>
    <message name="GetRandomUResponse">
        <part name="u" type="xsd:string"/>
    </message>
    <message name="SetMatch1Request">
        <part name="m1" type="xsd:string"/>
    </message>
    <message name="GetMatch2Response">
        <part name="m2" type="xsd:string"/>
    </message>
    <!-- continua na próxima figura -->
```

Figura 32 - WSDL com a definição do serviço de autenticação utilizando SRP (a).

```

<!-- port type declns -->
<portType name="SrpServerServices">
  <operation name="getSalt">
    <input message="tns:GetSaltRequest"/>
    <output message="tns:GetSaltResponse"/>
  </operation>
  <operation name="setPublicKeyA">
    <input message="tns:SetPublicKeyARequest"/>
    <output message="tns:empty"/>
  </operation>
  <operation name="getPublicKeyB">
    <input message="tns:empty"/>
    <output message="tns:GetPublicKeyBResponse"/>
  </operation>
  <operation name="getRandomU">
    <input message="tns:empty"/>
    <output message="tns:GetRandomUResponse"/>
  </operation>
  <operation name="setMatch1">
    <input message="tns:SetMatch1Request"/>
    <output message="tns:empty"/>
  </operation>
  <operation name="getMatch2">
    <input message="tns:empty"/>
    <output message="tns:GetMatch2Response"/>
  </operation>
</portType>
<!-- binding declns -->
<binding name="SrpServerServicesSOAPBinding"
type="tns:SrpServerServices">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getSalt">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="encoded" namespace="urn:srpserver"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:srpserver"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>
  <operation name="setPublicKeyA">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="encoded" namespace="urn:srpserver"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:srpserver"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>
<!-- continua na próxima figura -->

```

Figura 33 - WSDL com a definição do serviço de autenticação utilizando SRP (b).

```

</operation>
<operation name="getPublicKeyB"> <soap:operation soapAction=""/>
  <input>
    <soap:body use="encoded" namespace="urn:srpserver"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
    <soap:body use="encoded" namespace="urn:srpserver"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
</operation>
<operation name="getRandomU">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="encoded" namespace="urn:srpserver"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
    <soap:body use="encoded" namespace="urn:srpserver"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
</operation>
<operation name="setMatch1">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="encoded" namespace="urn:srpserver"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
    <soap:body use="encoded" namespace="urn:srpserver"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
</operation>
<operation name="getMatch2">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="encoded" namespace="urn:srpserver"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
    <soap:body use="encoded" namespace="urn:srpserver"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
</operation>
</binding>
<!-- service decln -->
<service name="SrpServerService">
  <port name="SrpServerServices"
    binding="tns:SrpServerServicesSOAPBinding">
    <soap:address
location="http://labmda.larc.facom.ufu.br:8888/axis/services/SrpServerSe
rvices"/>
  </port>
</service>
</definitions>

```

Figura 34 - WSDL com a definição do serviço de autenticação utilizando SRP (c).

Apesar da solução proposta utilizar a arquitetura do Apache AXIS para implementar todos os serviços, os *Web Services* de destino bem como o cliente podem estar em qualquer outra plataforma, e os *Web Services* utilizados no servidor de autenticação são baseados no Apache AXIS.

Capítulo 5

Implementação e Análise dos Resultados Obtidos

A solução foi implementada, implantada e então foram realizados testes a fim de validar a abordagem proposta, verificar sua viabilidade técnica e observar como estes *handlers* poderiam ser adicionados a quaisquer *Web Services*.

5.1. Implementação e Implantação da solução

O entendimento da arquitetura e do funcionamento do AXIS, bem como de outros processadores SOAP foi fundamental para a proposição da solução.

A partir da solução proposta, o desafio basicamente seria implementar e implantar esta solução a fim de realizar os testes necessários para a validação da mesma.

Para a implementação foi necessário um estudo da API de programação do AXIS [77] disponível para a linguagem Java a fim conhecer as classes e objetos envolvidos durante o processamento de mensagens SOAP pelo AXIS. Foi utilizada a versão 1.1 do Apache AXIS, visto ser esta última versão estável disponível no momento da implementação.

Em relação ao protocolo SRP foi utilizada como base uma implementação [78] do mesmo disponibilizada pelo autor do protocolo.

Esta implementação apenas mostra o algoritmo SRP para ser utilizado em um ambiente de testes onde basicamente a troca de informações entre o cliente e o servidor é feita através do recurso de copiar a informação em uma janela console e colar em outra a fim de verificar o funcionamento do protocolo, bem como as trocas de mensagens entre o cliente e servidor.

Desta forma, o código exemplo disponível, foi utilizado como ponto de partida e foi totalmente reescrito a fim de atender as necessidades da implementação. Sendo assim, o lado cliente, bem como o lado servidor, foram preparados para serem implementados e disponibilizados como *Web Services*.

Para a implantação de *Web Services* é necessário algum servidor de aplicações ou então um contêiner de *Servlets*, como o Apache Tomcat [79], que é a implementação de referência de *Servlets* utilizada pela SUN.

Servlets são certos tipos de aplicações Java que podem ser disponibilizadas na *Web* através dos *Servlets* contêineres e desta forma representam uma forma para estender servidores *Web* e disponibilizar aplicações através da *Web*, sendo possível, por exemplo, acessar o conteúdo de uma tabela existente em um banco de dados e mostrar este conteúdo em um cliente HTTP.

Desta forma foi necessário um aprofundamento no modo de funcionamento do Apache Tomcat, bem como de sua arquitetura, visto que este servidor foi utilizado para a implantação dos *Web Services*. Além disso, seus recursos de autenticação via HTTP e HTTPS também foram utilizados nos testes comparativos com solução proposta. Na implantação, foi utilizada a versão 5.0.19 do Apache Tomcat.

Para a implantação dos *Web Services* o Apache Tomcat foi instalado em todos os servidores, bem como o Apache AXIS. Foi criada uma infra-estrutura conforme mostrada na Figura 35.

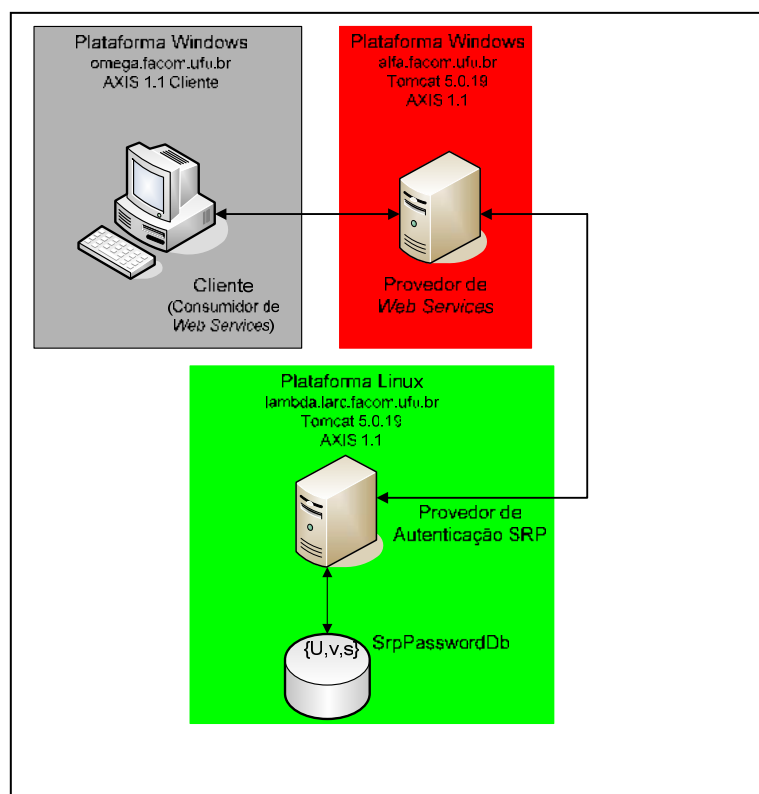


Figura 35 - Arquitetura utilizada na implantação.

Além disso, para trabalhar com o protocolo SRP foi necessária a criação de um banco de dados (**SrpPasswordDb**) a fim de armazenar as informações de autenticação que são utilizadas pelo protocolo. Neste banco de dados cada objeto contém o nome do usuário (U), o salto (s) e o verificador da senha, que é calculado, conforme mostrado na Figura 25. Uma aplicação simples foi criada para gerenciar este banco e acrescentar novos usuários de acordo com as regras estabelecidas pelo protocolo SRP, sempre que necessário. A Tabela 5 mostra como uma senha “fraca” como “123” foi armazenada neste banco de dados.

Tabela 5 - Informação armazenada sobre uma senha.

Usuário (U)	flavio
Salto (s)	1495651668
Verificador (v)	794475833261338707663717 023610959323625378500793 796898490895902743448916 587325152144362360149735 177660234988614249774914 394619824624673472788571 207151515345373469522257 461663172628582595309974 034591083070888771145141 425854549397348237012950 496970919285196756046873 246483705961449828584583 13579650792550023389

O ponto de partida para a implantação da solução proposta foi a criação do documento WSDL que descreve completamente a troca de mensagens entre o lado cliente e o lado servidor do protocolo SRP. Esta definição pode ser vista na Figura 32, Figura 33 e Figura 34. Este documento contém todas as operações necessárias para o funcionamento do protocolo SRP.

A partir do documento WSDL foi utilizada a ferramenta WSDLtoJava [80] que permite gerar o código Java necessário para a implantação e consumo dos *Web Services*.

5.2. Testes da solução proposta

A fim de observar as mensagens SOAP trocadas entre o cliente (consumidor), o provedor de *Web Services* (que também é o cliente SRP) e o provedor de autenticação SRP (servidor SRP), foi utilizada a ferramenta TCP Monitor (tcpmon) [80]. Esta ferramenta, disponível juntamente com o Apache AXIS, permite capturar as mensagens SOAP trocadas entre os vários *Web Services*, facilitando desta forma os testes e a visualização das mesmas.

Quando uma mensagem chega ao *handler* **SrpHandler**, mostrado na Figura 28, inicia-se a troca de mensagens requerida pelo protocolo SRP. A seguir são mostradas as mensagens necessárias para que uma autenticação com sucesso seja efetuada.

Inicialmente é necessário recuperar o salto, que está armazenado no banco de dados. A Figura 36 e a Figura 37 mostram respectivamente, de forma simplificada, o pedido e a resposta desta chamada.

```
<soapenv:Body>
  <ns1:getSalt>
    <user xsi:type="xsd:string">flavio</user>
  </ns1:getSalt>
</soapenv:Body>
```

Figura 36 - Mensagem GetSalt (pedido).

```
<soapenv:Body>
  <ns1:getSaltResponse>
    <salt xsi:type="xsd:string">1495651668</salt>
  </ns1:getSaltResponse>
</soapenv:Body>
```

Figura 37 - Mensagem GetSalt (resposta).

Assim que recebe o valor do salto, o *handler* **SrpHandler** calcula a chave pública temporária A e em seguida a envia para o servidor de autenticação (**SrpServerService**). A Figura 38 mostra a mensagem enviada.

```
<soapenv:Body>
  <ns1:setPublicKeyA>
    <A xsi:type="xsd:string">70527425795...</A>
  </ns1:setPublicKeyA>
</soapenv:Body>
```

Figura 38 - Mensagem setPublicKeyA para envio da chave pública temporária.

Em seguida *handler* **SrpHandler** envia um pedido para recuperar a chave pública temporária, que neste momento já foi calculada pelo servidor. O código XML mostrado na Figura 39, mostra a mensagem SOAP devolvida pelo servidor após a execução da mensagem **GetPublicKeyB**.

```

<soapenv:Body>
  <ns1:getPublicKeyBResponse>
    <B xsi:type="xsd:string">11518396...</B>
  </ns1:getPublicKeyBResponse>
</soapenv:Body>

```

Figura 39 - Resposta da mensagem getPublicKeyB.

Para calcular a chave de sessão (**S**) e a chave de sessão segura (**K**), que basicamente é o resultado de um *hash* feito sobre esta chave **S**, o *handler* **SrpHandler** necessita do valor chamado **u**. Este valor é um parâmetro que tem por objetivo embaralhar os resultados das funções *hash* executadas no lado cliente. A Figura 40 mostra o resultado deste pedido.

```

<soapenv:Body>
  <ns1:getRandomUResponse>
    <u xsi:type="xsd:string">413525534</u>
  </ns1:getRandomUResponse>
</soapenv:Body>

```

Figura 40 - Resposta da mensagem getRandomU.

Após calcular as chaves de sessão, o *handler* **SrpHandler** calculará o valor de comparação (**M1**) e este valor será enviado ao servidor de autenticação. A Figura 41 mostra o valor enviado ao servidor.

```

<soapenv:Body>
  <ns1:setMatch1>
    <m1
xsi:type="xsd:string">132065658659...</m1>
  </ns1:setMatch1>
</soapenv:Body>

```

Figura 41 - Envio do valor de comparação através da mensagem setMatch1.

Neste exemplo, o valor de comparação (**M1**), calculado pelo *handler* **SrpHandler** e o valor de comparação (**M2**), calculado pelos *Web Services* de autenticação (**SrpServeService**) serão iguais. Então, finalmente, o servidor enviará o valor de comparação **M2** calculado e o mesmo será comparado com um segundo valor de comparação calculado pelo cliente, neste exemplo, ambos serão iguais, indicando uma autenticação segura com sucesso, realizada através do protocolo SRP implementado através de *Web Services*.

5.3. Tempo de Latência

A fim de avaliar o tempo de latência da solução proposta realizou-se uma comparação entre o sistema sem autenticação e o sistema com a autenticação SRP.

Nos testes foi considerado somente o tempo utilizado pelo protocolo e desta forma o tempo de resposta (*round-trip*) das mensagens através da Internet foi desprezado. Para isto todos os componentes da solução, mostrados na Figura 28, foram implantados em um mesmo computador, ou seja, a aplicação cliente, o servidor de autenticação e os *Web Services* de destino. A Figura 42 mostra os tempos obtidos.

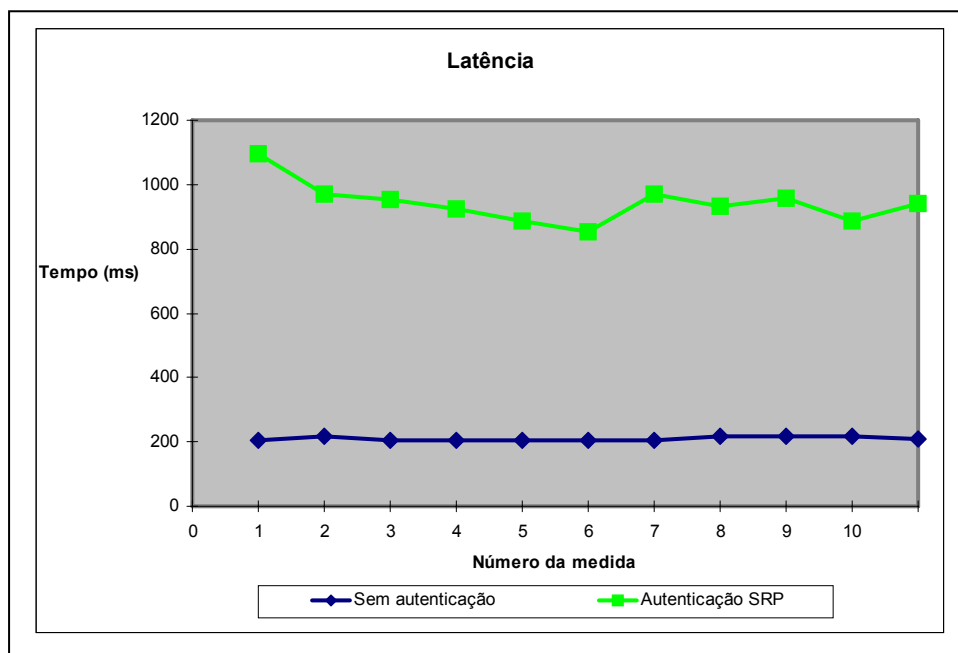


Figura 42 - Tempo de latência em milissegundos.

As medições em cada situação, com autenticação SRP e sem a autenticação, foram feitas em momentos distintos. Foram realizadas dez medições e em cada uma delas foi obtido o tempo necessário para o processamento.

Como era esperado, o protocolo introduziu um tempo extra necessário ao processamento. Este tempo foi em torno de 4.49 vezes maior que o tempo sem o uso do protocolo de autenticação.

A fim de comparar a solução proposta com outras abordagens foi verificado o tempo de latência de outras duas formas de autenticação. Autenticação básica via HTTP e também a autenticação básica via HTTPS.

A autenticação básica via HTTP representa o há de mais fraco em relação à autenticação de recursos na *Web*.

Já a autenticação básica via HTTPS utiliza o mesmo princípio, porém a transmissão dos dados é feita através de um canal “seguro” com os dados criptografados. Deve ser ressaltado que ambas as abordagens apresentam uma série de deficiências conforme pode ser notado pela Tabela 3.

Cabe aqui também a ressalva de que estas técnicas são feitas basicamente utilizando recursos disponíveis na camada de transporte dos *Web Services*, conforme mostrado na Figura 5. Devido a este fato as técnicas não atendem aos requisitos de segurança necessários aos *Web Services* e, além disso, estas técnicas dependem do tipo de servidor *Web* e/ou servidor de aplicações utilizado, o que afeta a interoperabilidade dos *Web Services*.

Para a realização dos testes foi necessária a configuração do Apache Tomcat a fim de que o mesmo realizasse a autenticação ao serviço. A Figura 43 abaixo mostra os resultados obtidos.

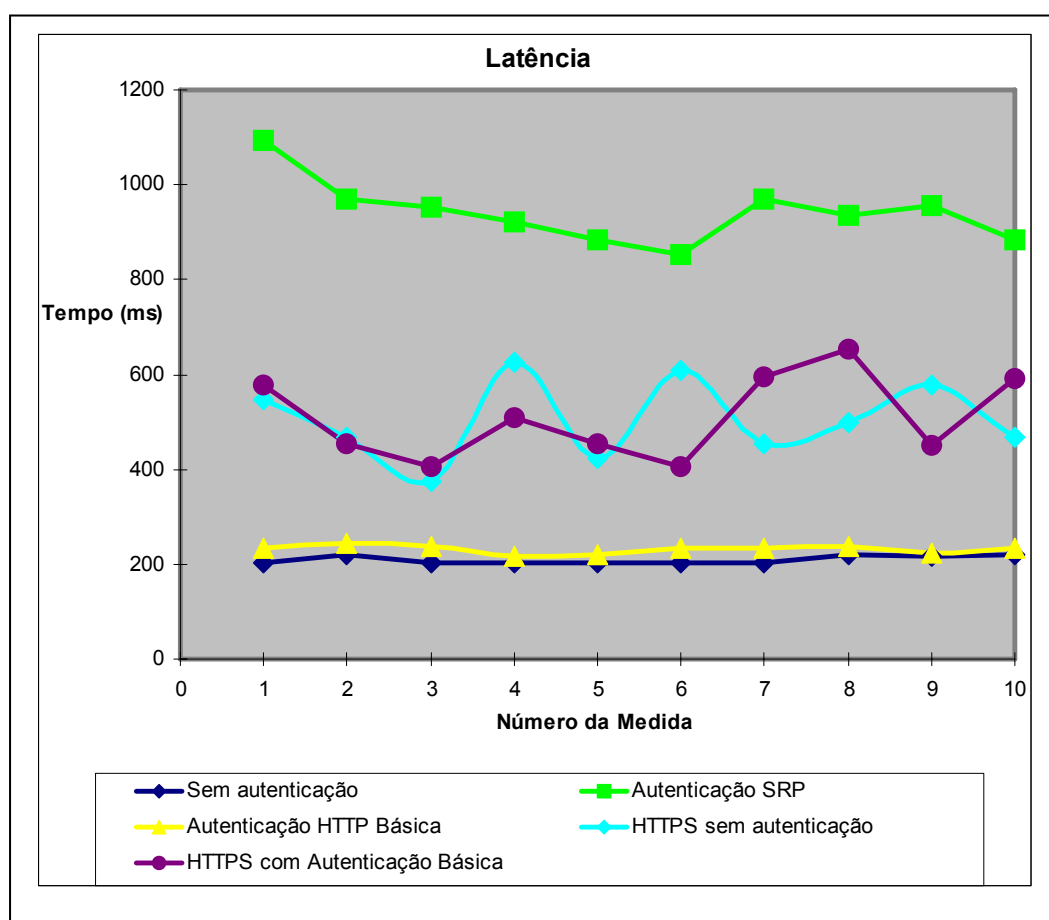


Figura 43 - Tempo de latência em milissegundos com outras autenticações.

Pela análise dos dados acima percebe-se que o uso de autenticação básica via HTTP impõe um tempo de latência, devido à autenticação, em torno de 10% maior que o tempo de latência para uso do serviço sem autenticação. Este resultado por um lado é animador, porém a deficiências deste tipo de autenticação inviabilizam seu uso.

Foi realizado também um teste a fim de verificar o tempo de latência ao se utilizar apenas o protocolo HTTPS, sem autenticação. Neste caso o foi introduzido um tempo de latência que

foi 2.41 vezes maior que o tempo para acesso ao serviço sem autenticação sobre o protocolo HTTP.

Finalmente foi feita a comparação do acesso ao serviço com autenticação básica via HTTPS. Neste caso o tempo de latência foi 2.43 vezes maior que o tempo de latência sem autenticação.

Os resultados mostram e comprovam que à medida que se incrementa a forma de autenticação os tempos crescem.

Outro aspecto interessante é que o tempo de latência da solução proposta é 84,92% maior que o tempo de latência apresentado pelo uso de autenticação básica via HTTPS, ou seja, o tempo que inicialmente representava um aumento de 450% na realidade é de aproximadamente apenas 85% frente a um método de autenticação que apresenta deficiências. Apesar dos testes não contemplarem este aspecto, o uso de outro método, como troca de certificados via HTTPS, provavelmente consumiria mais tempo e desta forma o aumento acarretado pela solução proposta tende a ser menor se comparado a outras formas de autenticação.

Capítulo 6

Conclusão

A solução proposta e implementada mostrou-se válida e de grande utilidade para a execução do controle de autenticação baseado em um protocolo de autenticação segura através de senhas, que é o protocolo SRP.

A estrutura do protocolo SOAP permite que a solução seja implementada em qualquer plataforma, visto que todos os processadores SOAP oferecem recursos para a inserção e remoção de *header blocks* nestas mensagens.

Outra característica da solução proposta é que o controle de autenticação é implementado por *Web Services* e desta forma, os desenvolvedores de outros *Web Services*, podem se concentrar na lógica própria de seu negócio e utilizar o serviço de autenticação, não sendo necessária a utilização de recursos e esforços para implementar este controle em cada um dos *Web Services* desenvolvidos.

Apesar desta propriedade não ter sido verificada, pode-se afirmar, que troca de mensagens proposta para a implementação do protocolo SRP é de fácil implantação nos mais variados processadores SOAP disponíveis. Todas as mensagens necessárias para a execução do protocolo utilizam um elemento XML do tipo **xsd:string**, amplamente suportado, garantindo um alto nível de interoperabilidade à solução proposta.

Considerando as propriedades e a segurança dos protocolos envolvidos em autenticação segura através de senhas, pode-se afirmar que a chave de sessão (**K**), criada pelo protocolo SRP, pode ser utilizada, com segurança, como um identificador de sessão de um usuário autenticado. Esta informação pode, ainda, ser utilizada em relações de confiança mútua entre o cliente e outros servidores.

Neste trabalho foi utilizado o protocolo SRP-3 [50] para implementar a troca de mensagens necessárias à autenticação. O aumento do tempo de latência, necessário à autenticação, foi da ordem de quatro vezes e meia (4.5) maior que o tempo para o acesso a um serviço sem a autenticação, porém foi apenas 85% maior que a autenticação básica via HTTPS. Este resultado pode ser melhorado através do uso do protocolo SRP-6 [52] que contém um refinamento do protocolo SRP, proposto inicialmente. Neste protocolo são

necessárias apenas quatro mensagens ao invés de sete utilizadas no protocolo SRP-3, e, além disso, este protocolo garante uma maior segurança face a certos tipos de ataque.

Outra possibilidade de trabalho futuro é a extensão da solução proposta, através do uso da linguagem SAML. Neste caso o servidor responsável pela autenticação pode também fornecer assertivas SAML a respeito de um usuário autenticado para outros *Web Services*. Desta forma abre-se caminho para a implementação da autenticação única (*Single Sign On*) de forma segura entre vários *Web Services*, um importante requisito nesta área.

Para a comunicação ponto a ponto entre o cliente e provedor de *Web Services* na solução proposta foi utilizado o protocolo HTTP juntamente como protocolo SSL. Uma possível modificação que pode ser implementada é o uso de *WS-Security* a fim de garantir a confidencialidade e integridade da mensagem que transporta a senha do cliente até o *handler* localizado no provedor de *Web Services*.

Os protocolos para a autenticação segura baseados em senha estão em constante evolução; esta proposta pode ser estendida para outros protocolos que venham a surgir. Como o serviço de autenticação é descentralizado e possui um fraco acoplamento, os *Web Services* que o utilizarem estarão sempre contanto com a última palavra em termos de segurança.

O desenvolvimento deste trabalho permitiu um amplo estudo sobre os protocolos e as tecnologias envolvidas nos *Web Services*, através deste estudo foi possível conhecer as várias camadas envolvidas, as tecnologias que estão padronizadas de fato e aquelas que ainda estão em fase de adoção. Esta percepção é importante para desenvolvedores e outros profissionais da área que necessitam utilizar a tecnologia dos *Web Services*.

Uma importante contribuição deste trabalho é o estabelecimento de uma visão da arquitetura dos *Web Services* em seu estágio atual. Esta arquitetura, mostrada na Figura 5 agrega uma série de informações e pode ser usada como uma importante referência sobre os *Web Services*, as tecnologias atuais e seu futuro.

Através dos comparativos entre os vários processadores SOAP existentes foi possível perceber que ferramentas como o Apache AXIS, o Java *Web Services Developer Pack* (JWSDP) e o Microsoft .Net oferecem vários recursos, porém é necessário que o desenvolvedor implemente sua solução muitas vezes partindo do zero. Já outras ferramentas como o *BEA WebLogic Workshop* e o *Systinet Server for Java*, estão em um interessante estágio de evolução e são alternativas interessantes para a implantação rápida de *Web Services*.

Os protocolos de autenticação segura através de senha necessitam de uma troca de mensagens sempre em número maior que o protocolo padrão onde o usuário informa a senha

e espera a resposta. Esta característica favorece o uso dos *Web Services*, visto que os mesmos se baseiam na troca de mensagens.

O conhecimento envolvido nos protocolos de autenticação segura é um mundo à parte. Durante a realização deste trabalho foi possível inteirar-se destes conceitos e isto provocou um grande amadurecimento pessoal em relação ao entendimento dos requisitos de segurança e aos métodos necessários para garantir a mesma em sistemas distribuídos como os *Web Services*.

Outro ganho pessoal importante foi o entendimento da arquitetura do Apache AXIS. À medida que este entendimento se aprofundava, mais vez mais, se verificava que a solução proposta estava baseada em premissas corretas.

Os *Web Services* representam a tecnologia do presente e do futuro para computação distribuída. Apesar de estarem em evolução, logo esta tecnologia se estabelecerá e ocupará um lugar de destaque, assim como acontece hoje com o protocolo HTTP para o transporte de informações.

A autenticação de um usuário é a porta de entrada para uma série de outros serviços. O uso de protocolos de autenticação seguros baseados em senha, e oferecidos como *Web Services*, representa um meio eficaz para o controle de autenticação de acesso a serviços e cada vez mais esta tecnologia estará presente na Internet, participando e contribuindo com a sua evolução.

Capítulo 7

Referências Bibliográficas

- [1] Booth, David et al. *Web Services Architecture*. W3C Working Group Note 11 February 2004. Disponível em: <<http://www.w3.org/TR/ws-arch/>>. Acesso em: 10 de nov. de 2004.
- [2] Tanenbaum, Andrew S. *Computer Networks*, Fourth Edition. New Jersey: Prentice Hall, 2003
- [3] Wu, T. Competitive Analysis of SRP. The Stanford SRP Authentication Project. Disponível em: <<http://srp.stanford.edu/analysis.html>>. Acesso em: 10 de nov. de 2004.
- [4] SecuritySpace.com. Secure Server Survey. October 1st, 2004. Disponível em: <http://www.securityspace.com/s_survey/sdata/200409/index.html>. Acesso em: 10 de nov. de 2004
- [5] SecuritySpace.com. Web Server Survey. October 1st, 2004. Disponível em: <http://www.securityspace.com/s_survey/data/200409/index.html>. Acesso em: 10 de nov. de 2004
- [6] Netcraft. SSL Server Survey January 2004. Disponível em: <<http://survey.netcraft.com/surveys/analysis/https/2004/Jan/CMatch/index.html>>. Acesso em: 10 de nov. de 2004
- [7] Newcomer, Eric. *Understanding Web Services: XML, WSDL, SOAP and UDDI*. Boston: Addison-Wesley, 2002
- [8] Nadalin, Anthony et al. *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)*. OASIS Standard 200401, March 2004. Disponível em: <<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>>. Acesso em: 10 de nov. de 2004.
- [9] Berners-Lee, Tim. Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396. Disponível em: <<http://ietf.org/rfc/rfc2396.txt>>. Acesso em: 10 de nov. de 2004.
- [10] World Wide Web Consortium. Disponível em: <<http://www.w3.org/>>. Acesso em: 10 de nov. de 2004.
- [11] Mitra, Nilo. SOAP Version 1.2 Part 0: Primer. W3C Recommendation 24 June 2003. Disponível em: <<http://www.w3.org/TR/2003/REC-soap12-part0-20030624/>>. Acesso em: 10 de nov. de 2004.
- [12] Chinnici, Roberto et al. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. W3C Working Draft 3 August 2004. Disponível em: <<http://www.w3.org/TR/2004/WD-wsdl20-20040803/>>. Acesso em: 10 de nov. de 2004.

- [13] Organization for the Advancement of Structured Information Standards (OASIS). Disponível em: <<http://www.oasis-open.org/home/index.php>>. Acesso em: 10 de nov. de 2004.
- [14] Clément, Luc et al. UDDI Version 3.0.2 specification. UDDI Spec Technical Committee Draft. Disponível em: <<http://www.oasis-open.org/committees/uddi-spec/doc/spec/uddi-v3.0.2-20041019.htm>>. Acesso em: 10 de nov. de 2004.
- [15] Cantor, Scott et al. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Disponível em: <<http://www.oasis-open.org/committees/download.php/9455/sstc-saml-core-2.0-cd-02.pdf>>. Acesso em: 10 de nov. de 2004.
- [16] *Web Services* Interoperability Organization(WS-I). Disponível em: <<http://ws-i.org/>>. Acesso em: 10 de nov. de 2004.
- [17] Scribner, Kenn; Stiver, Mark C. *Applied SOAP: Implementing .NET XML Web Services*. Sams Publishing, 432p, 2002
- [18] Chappell, David A.; Jewell, Tyler. *Java Web Services*. San Francisco: O'Reilly, 2002
- [19] World Wide Web Consortium. *Web Services Architecture Working Group*. Disponível em: <<http://www.w3.org/2002/ws/arch/>>. Acesso em: 10 de nov. de 2004.
- [20] Vinoski, Steve. "WS-Nonexistent Standards". *IEEE Internet Computing*, vol. 8, no. 6, 2004, pp. 94-96. Disponível em: <<http://dsonline.computer.org/0412/d/w6towp.htm>>. Acesso em: 10 de nov. de 2004.
- [21] Cabrera, Luis F.; Kurt, Christopher; Box, Don. *An Introduction to the Web Services Architecture and Its Specifications*, version 2.0. Disponível em: <<http://msdn.microsoft.com/webservices/understanding/advancedwebservices/default.aspx?pull=/library/en-us/dnwebsrv/html/introwsa.asp>>. Acesso em: 10 de nov. de 2004.
- [22] Microsoft; IBM. *Security in a Web Services World: A Proposed Architecture and Roadmap*. Disponível em: <<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp>>. Acesso em: 10 de nov. de 2004.
- [23] Ballinger, Keith et al. *Basic Profile Version 1.1*. The Web Services-Interoperability Organization. Disponível em: <<http://www.ws-i.org/Profiles/BasicProfile-1.1.html>>. Acesso em: 10 de nov. de 2004.
- [24] Banks, Andrew et al. *HTTPR Specification*. IBM Software Group. Disponível em: <<http://www.ibm.com/developerworks/library/ws-httpspec/>>. Acesso em: 10 de nov. de 2004.
- [25] Eastlake, Donald et al. *XML Encryption Syntax and Processing*. W3C Recommendation 10 December 2002. Disponível em: <<http://www.w3.org/TR/xmlenc-core/>>. Acesso em: 10 de nov. de 2004.

- [26] Gudgin, Martin et al. SOAP Message Transmission Optimization Mechanism. W3C Proposed Recommendation 16 November 2004. Disponível em: <<http://www.w3.org/TR/soap12-mtom/>>. Acesso em: 10 de nov. de 2004.
- [27] Malhotra, Ashok; Biron, Paul V. XML Schema Part 2: Datatypes Second Edition. W3C Recommendation 28 October 2004. Disponível em: <<http://www.w3.org/TR/xmlschema-2/>>. Acesso em: 10 de nov. de 2004.
- [28] Iwasa, Kazunori et al. WS-Reliability 1.1. Committee Draft 1.086, 24 August 2004. Disponível em: <<http://www.oasis-open.org/committees/download.php/9330/WS-Reliability-CD1.086.zip>>. Acesso em: 18 de nov. de 2004.
- [29] Bunting, Doug et al. Web Services Composite Application Framework (WS-CAF) Ver1.0. Disponível em: <<http://www.oasis-open.org/committees/download.php/4343/WS-CAF%20Primer.pdf>>. Acesso em: 10 de nov. de 2004.
- [30] Askary, Sid et al. Web Services Business Process Execution Language. Working Draft 01, 08 September 2004. Disponível em: <<http://www.oasis-open.org/committees/download.php/9094/wsbpel-specification-draft-Sept-08-2004.html>>. Acesso em: 10 de nov. de 2004.
- [31] Kavantzaz, Nickolas et al. Web Services Choreography Description Language Version 1.0. W3C Working Draft 12 October 2004. Disponível em: <<http://www.w3.org/TR/ws-cdl-10/>>. Acesso em: 10 de nov. de 2004.
- [32] OASIS. OASIS International Health Continuum (IHC) TC. Disponível em: <<http://www.oasis-open.org/committees/ihc/charter.php>>. Acesso em: 10 de nov. de 2004.
- [33] OASIS. OASIS Universal Business Language (UBL) TC. Disponível em: <<http://www.oasis-open.org/committees/ubl/charter.php>>. Acesso em: 10 de nov. de 2004.
- [34] OASIS. OASIS Electronic Procurement Standardization (EPS) TC. Disponível em: <<http://www.oasis-open.org/committees/eps/charter.php>>. Acesso em: 10 de nov. de 2004.
- [35] OASIS. OASIS Web Services Distributed Management (WSDM) TC. Disponível em: <<http://www.oasis-open.org/committees/wsdm/charter.php>>. Acesso em: 10 de nov. de 2004.
- [36] Kropp, Alan et al. Web Services for Remote Portlets Specification. Approved as an OASIS Standard August 2003. Disponível em: <<http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>>. Acesso em: 10 de nov. de 2004.
- [37] OASIS. OASIS Product Life Cycle Support (PLCS) TC. Disponível em: <<http://www.oasis-open.org/committees/wsrp/charter.php>>. Acesso em: 10 de
- [38] OASIS. OASIS E-Government TC. Disponível em: <<http://www.oasis-open.org/committees/egov/charter.php>>. Acesso em: 10 de nov. de 2004.

- [39] McGlashan, Scott et al. Voice Extensible Markup Language (VoiceXML) Version 2.0. W3C Recommendation 16 March 2004. Disponível em: <<http://www.w3.org/TR/voicexml20/>>. Acesso em: 10 de nov. de 2004.
- [40] Bajaj, Siddharth et al. Web Services Policy Framework (WSPolicy). Disponível em: <<http://ftpna2.bea.com/pub/downloads/WS-Policy.pdf>>. Acesso em: 10 de nov. de 2004.
- [41] Bray, Tim; Hollander, Dave; Layman, Andrew. Namespaces in XML. Disponível em: <<http://www.w3.org/TR/REC-xml-names>>. Acesso em: 10 de nov. de 2004.
- [42] Thompson, Henry S. XML Schema Part 1: Structures Second Edition. W3C Recommendation 28 October 2004. Disponível em: <<http://www.w3.org/TR/xmlschema-1/>>. Acesso em: 10 de nov. de 2004.
- [43] Christensen, Erik. Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001. Disponível em: <<http://www.w3.org/TR/wsdl>>. Acesso em: 10 de nov. de 2004.
- [44] Chinnici, Roberto. Web Services Description Language (WSDL) Version 1.2. W3C Working Draft 9 July 2002. Disponível em: <<http://www.w3.org/TR/2002/WD-wsdl12-20020709/#portType-desc-component>>. Acesso em: 10 de nov. de 2004.
- [45] Graham, Steve; Murray, Bryan. Web Services Base Notification 1.2 (WS-BaseNotification). Working Draft 03, 21 June 2004. Disponível em: <<http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-03.pdf>>. Acesso em: 10 de nov. de 2004.
- [46] Eastlake, Donald et al. XML-Signature Syntax and Processing. W3C Recommendation 12 February 2002. Disponível em: <<http://www.w3.org/TR/xmlsig-core/>>. Acesso em: 10 de nov. de 2004.
- [47] Housley, R. et al. Internet X.509 Public Key Infrastructure. Certificate and Certificate Revocation List (CRL) Profile. RFC 3280. Disponível em: <<http://www.faqs.org/rfcs/rfc3280.html>>. Acesso em: 10 de nov. de 2004.
- [48] Neuman, C. Ts'o, T. Kerberos: An Authentication Service for Computer Networks, IEEE Communications, 32(9):33-38. September 1994. Disponível em: <<http://www.isi.edu/gost/publications/kerberos-neuman-tso.html>>. Acesso em: 10 de nov. de 2004.
- [49] Apache Web Services Project. Project WSS4J. Disponível em: <<http://ws.apache.org/ws-fx/wss4j/index.html>>. Acesso em: 10 de nov. de 2004.
- [50] Wu. T. The Secure Remote Password Protocol. Proceedings of the Internet Society Network and Distributed System Symposium, pages 97-111, March 1998.
- [51] Coulson, Brenda. The Future of Secure Remote Password (SRP). Disponível em: <http://www.technology-evaluation.com/Research/ResearchHighlights/Security/2003/03/research_notes/MI_ST_XBC_03_03_03_1.asp>. Acesso em: 10 de nov. de 2004.

- [52] Wu, T. SRP-6: Improvements and Refinements to the Secure Remote Password Protocol, Submission to the IEEE P1363 Working Group, Oct 2002. Disponível em: <<http://srp.stanford.edu/srp6.ps>>. Acesso em: 10 de nov. de 2004.
- [53] D. Jablon. Extended Password Key Exchange Protocols Immune to Dictionary Attacks. Proceedings of the Sixth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE '97), IEEE Computer Society, June 18-20, 1997, Cambridge, MA, pp. 248-255. Disponível em: <<http://grouper.ieee.org/groups/1363/passwdPK/contributions/p1363-pw.pdf>>. Acesso em: 10 de nov. de 2004.
- [54] D. Jablon, Strong Password-Only Authenticated Key Exchange. Computer Communication Review, ACM SIGCOMM, vol. 26, no. 5, pp. 5-26, October 1996. Disponível em: <<http://grouper.ieee.org/groups/1363/passwdPK/contributions/jablon.pdf>>. Acesso em: 10 de nov. de 2004
- [55] Bellovin, S. M.; Merrit, M. Augmented Encrypted Key Exchange: A Password-Based Protocol Secure Against Dictionary Attacks and Password File Compromise. Proceedings of the 1st ACM Conference on Computer and Communications Security, pages 244-250, ACM Press, November 1993. Disponível em: <<http://www.research.att.com/~smb/papers/aeke.pdf>>. Acesso em: 10 de nov. de 2004
- [56] Bellovin, S. M.; Merrit, M. Encrypted Key Exchange: Password- Based Protocols Secure Against Dictionary Attacks. Proceedings of the I.E.E.E. Symposium on Research in Security and Privacy, Oakland, May 1992. Disponível em: <<http://downloads.securityfocus.com/library/neke.ps>>. Acesso em: 10 de nov. de 2004
- [57] Ylonen, T. et al. SSH Authentication Protocol. Network Working Group, INTERNET-DRAFT, 6 August 1998. Disponível em: <<http://www.ietf.org/proceedings/98aug/I-D/draft-ietf-secsh-userauth-04.txt>>. Acesso em: 10 de nov. de 2004
- [58] Dierks, T.; Allen, C. The TLS Protocol. Version 1.0. RFC 2246. Disponível em: <<http://www.faqs.org/rfcs/rfc2246.html>>. Acesso em: 10 de nov. de 2004.
- [59] Fielding, R. et al. Hypertext Transfer Protocol -- HTTP/1.1. RFC 2068. Disponível em: <<http://www.faqs.org/rfcs/rfc2068.html>>. Acesso em: 10 de nov. de 2004.
- [60] Franks, J. et al. An Extension to HTTP : Digest Access Authentication. RFC 2069. Disponível em: <<http://www.ietf.org/rfc/rfc2069.txt>>. Acesso em: 10 de nov. de 2004.
- [61] Steiner, J.; Neuman, C.; Schiller, J. Kerberos: An Authentication Service for Open Network Systems, pp. 191-202 in Usenix Conference Proceedings, Dallas, Texas, February, 1988. Disponível em: <[http://www.cybersafe.ltd.uk/docs_other/Kerberos%20-%20An%20Authentication%20Service%20for%20Open%20Network%20Systems%20\(P%20roject%20Athena\).pdf](http://www.cybersafe.ltd.uk/docs_other/Kerberos%20-%20An%20Authentication%20Service%20for%20Open%20Network%20Systems%20(P%20roject%20Athena).pdf)>. Acesso em: 10 de nov. de 2004.
- [62] Wu, T. A real world analysis of kerberos password security. In Proceedings of the 1999 Internet Society Network and Distributed System Security Symposium, 1999. Disponível em: <http://www.nata2.info/misc/Wireless/A_Real-World_Analysis_of_Kerberos_Password_Security-MIRROR.html>. Acesso em: 10 de nov. de 2004.

- [63] Franks, J. et al. HTTP Authentication: Basic and Digest Access Authentication. RFC 2617. Disponível em: <<http://www.ietf.org/rfc/rfc2617.txt>>. Acesso em: 10 de nov. de 2004
- [64] Niemi, A.; Arkko, J.; Torvinen, V. Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA). RFC 3310. Disponível em: <<http://www.faqs.org/rfcs/rfc3310.html>>. Acesso em: 10 de nov. de 2004
- [65] Bellovin, Steve M., Merrit, M. Limitations of the Kerberos Authentication System. ACM SIGCOMM Computer Communication Review archive Volume 20 , Issue 5, October 1990. Disponível em: <<http://www.research.att.com/~smb/papers/kerblimit.usenix.pdf>>. Acesso em: 10 de nov. de 2004
- [66] Apache Axis. Disponível em: <<http://ws.apache.org/axis/>>. Acesso em: 10 de nov. de 2004
- [67] SOAP::Lite for Perl. Disponível em: <<http://www.soaplite.com/>>. Acesso em: 10 de nov. de 2004.
- [68] gSOAP. Disponível em: <<http://www.cs.fsu.edu/%7Eengelen/soap.html>>. Acesso em: 10 de nov. de 2004.
- [69] IBM WebSphere SDK for Web Services (WSDK) Version 5.1. Disponível em: <http://www-106.ibm.com/developerworks/webservices/wsdk/?S_TACT=103AMW10&S_CMP=WSDD>. Acesso em: 10 de nov. de 2004.
- [70] Java Web Services Developer Pack (Java WSDP). Disponível em: <<http://java.sun.com/webservices/jwsdp/index.jsp>>. Acesso em: 10 de nov. de 2004.
- [71] Microsoft .NET Framework. Disponível em: <<http://msdn.microsoft.com/webservices/downloads/default.aspx>>. Acesso em: 10 de nov. de 2004.
- [72] BEA WebLogic Workshop. Disponível em: <<http://dev2dev.bea.com/technologies/webservices/index.jsp>>. Acesso em: 10 de nov. de 2004.
- [73] webMethods GLUE. Disponível em: <<http://www.webmethods.com/meta/default/folder/0000007014>>. Acesso em: 10 de nov. de 2004.
- [74] Systinet Server for Java. Disponível em: <http://www.systinet.com/products/wasp_jserver/overview>. Acesso em: 10 de nov. de 2004.
- [75] Apache AXIS. Axis Architecture Guide. Disponível em: <<http://ws.apache.org/axis/java/architecture-guide.html>>. Acesso em: 10 de nov. de 2004.
- [76] Silva, Flávio O.; Pacheco, João A. A.; Rosa, Pedro F. A SRP Based Handler for Web Service Access Control. In Proceedings of the 2004 IEEE International Conference on

- Services Computing (SCC'04). IEEE Computer Society, 2004, Shanghai, pp. 57-62. Disponível em: <http://www.facom.ufu.br/~flavio/webservices/srp/ws_srp_flaviosilva.pdf>. Acesso em: 10 de nov. de 2004
- [77] Apache AXIS. API Documentation. Disponível em: <<http://ws.apache.org/axis/java/apiDocs/index.html>>. Acesso em: 10 de nov. de 2004.
- [78] Wu, T. The Stanford SRP Authentication Project. Binary Distributions. Disponível em: <<http://srp.stanford.edu/binaries.html>>. Acesso em: 10 de nov. de 2004.
- [79] Vivek Chopra et al. Professional Apache Tomcat 5. Indianapolis: Wiley, 2004.
- [80] Apache AXIS. Axis User's Guide. Disponível em: <<http://ws.apache.org/axis/java/user-guide.html>>. Acesso em: 10 de nov. de 2004.

Apêndice a

Padrões utilizados em *Web Services*

A seguir é apresentado um breve resumo sobre as várias especificações existentes relativas aos Web Services. Além de uma descrição básica da mesma, existem informações sobre os autores das especificações, a data em que foi inicialmente publicada e a sua situação atual frente às organizações de padronização.

Existem nesta lista especificações que foram substituídas por outras antes mesmo de chegar aos organismos de padronização, as mesmas são citadas a fim de esclarecer e também mostrar a evolução de determinado padrão.

Certas especificações, propostas por diferentes empresas, visam o mesmo objetivo e neste caso a alternativa a especificação também é citada.

Esta lista fornece subsídios para uma visão geral da arquitetura dos *Web Services*, dos padrões já estabelecidos, daqueles que ainda virão e também permite uma comparação entre padrões semelhantes. O conhecimento dos autores e das empresas que participam de determinado grupo de trabalho é importante para verificar o comprometimento do mesmo. Via de regra, quando grandes empresas que normalmente concorrentes estão do mesmo lado defendendo um padrão o mesmo pode ser considerado como um forte candidato a ser um padrão de fato para toda a indústria de software.

Nome: ASAP

Domínio: TRANSAÇÕES

Data Criação: 1/6/2004 **Autores:** Cisco, Fujitsu

Organização: OASIS

Status: TC em trabalho

Grupo de Trabalho: OASIS Asynchronous Service Access Protocol (ASAP) TC

Data de Aprovação:

Alternativas:

Descrição:

Esta especificação estabelece um padrão para a integração de serviços assíncronos utilizando Web Services. A integração e interações consistem de um controle e do monitoramento destes serviços. O controle envolve criar em criar, configurar, iniciar e finalizar o serviço, sendo informado das exceções, da conclusão e obtendo seus resultados. O monitoramento consiste em verificar o estado atual do serviço e obter sua história.

A ASAP (Asynchronous Service Access Protocol) é uma proposta para resolver o problema acima utilizando o protocolo SOAP, para tanto uma série de métodos são definidos, bem como as informações envolvidas a fim de garantir os objetivos descritos acima.

Nome: DIME **Domínio:** MENSAGEM

Data Criação: **Autores:**

Organização: **Status:** SUBSTITUÍDO

Grupo de Trabalho:

Data de Aprovação: **Alternativas:**

Descrição:
SUBSTITUÍDO POR SOAP MTOM

Nome: HTTPR **Domínio:** TRANSPORTE

Data Criação: 1/4/2002 **Autores:** IBM

Organização: **Status:** Em desenvolvimento

Grupo de Trabalho:

Data de Aprovação: **Alternativas:**

Descrição:
Entrega confiável de mensagens http entre o servidor e o cliente. Abre caminho para o serviço com conexão confiável

Nome: NAMESPACES IN XML **Domínio:** XML

Data Criação: **Autores:**

Organização: W3C **Status:** Recomendação

Grupo de Trabalho: XML Core Working Group

Data de Aprovação: 1/1/1999 **Alternativas:**

Descrição:
Contém um método para qualificar os elementos e atributos utilizados pela linguagem XML, através da associação de namespaces identificados por URI.

Nome: SAML **Domínio:** SEGURANÇA

Data Criação: 1/11/2002 **Autores:** Internet2, Nokia, RSA Security, Sun, AOL, BEA, Boeing, Hewlett-Packard, IBM, Netegrity, Sigaba

Organização: OASIS **Status:** Padrão OASIS

Grupo de Trabalho: OASIS Security Services (SAML) TC

Data de Aprovação: **Alternativas:**

Descrição:
Define uma sintaxe e uma semântica para o processamento de asserções feitas sobre um assunto por um sistema. Através desta linguagem é possível a troca de informações sobre autenticação, autorização e outros atributos entre diferentes infra-estruturas de segurança e entre aplicações.

Nome: SOAP **Domínio:** MENSAGEM

Data Criação: **Autores:**

Organização: W3C **Status:** Recomendação

Grupo de Trabalho: XML Protocol Working Group

Data de Aprovação: 1/5/2002 **Alternativas:**

Descrição:

O protocolo SOAP é a base para a troca de informações entre Web Services. Este protocolo utiliza os meios de transportes hoje amplamente utilizados na Internet para a troca de informações entre diferentes pontos. O protocolo especifica o formato da mensagem e o seu processamento

Nome: SOAP MTOM **Domínio:** MENSAGEM

Data Criação: **Autores:**

Organização: W3C **Status:** Candidato a Recomendação

Grupo de Trabalho: XML Protocol Working Group

Data de Aprovação: 1/8/2004 **Alternativas:**

Descrição:

Descreve um mecanismo a fim de otimizar a transmissão e/ou formato de uma mensagem SOAP através de uma codificação em certas partes da mensagem. Desta forma é possível otimizar a transmissão da mensagem através da compactação, utilizando seqüências de caracteres compactas, que podem ser reconstruídas, no lado receptor.

Nome: SOAP-OVER-UDP **Domínio:** MENSAGEM

Data Criação: 1/9/2004 **Autores:** Microsoft

Organização: Ainda não **Status:** Especificação publicada

Grupo de Trabalho:

Data de Aprovação: **Alternativas:**

Descrição:

O comportamento do User Datagram Protocol (UDP), RFC 768, pode ser utilizado em muitas aplicações em WebServices. Algumas não necessitam de garantia de entrega, e outras podem utilizar a multidifusão (multicast). A fim de permitir que os Web Services utilizem estes padrões de funcionamento é necessário mapear mensagens SOAP em datagramas.

Em Aplicações, como por exemplo, WS-Discovery, tais padrões são de muita utilidade.

Nome: UDDI **Domínio:** METADADOS

Data Criação: **Autores:**

Organização: OASIS **Status:** Padrão OASIS

Grupo de Trabalho: OASIS UDDI Specification TC

Data de Aprovação: 1/6/2002 **Alternativas:**

Descrição:

Define um conjunto de serviços que permite a descrição e descoberta de negócios, organizações e outros provedores de Web Services, os serviços que os mesmos oferecem e as interfaces que podem ser utilizadas para acessar estes serviços. Baseado em padrões da indústria como HTTP, XML, XML Schema e SOAP o UDDI fornece a infra-estrutura necessária para que sistemas baseados em Web Service sejam compartilhados internamente através de uma organização ou de forma pública através da Internet.

Nome: WS-Acknowledgement**Domínio:** MENSAGEM CONFIÁVEL**Data Criação:** 1/2/2003 **Autores:** BEA**Organização:** Ainda não**Status:** Especificação publicada**Grupo de Trabalho:****Data de Aprovação:****Alternativas:** WS-Reliability**Descrição:**

Destinado a suportar a troca de mensagens confiáveis entre Web Services através da garantia de transferências de mensagens "pelo menos uma vez" e "exatamente uma vez". Baseada no conceito de envio de uma confirmação indicando que a mensagem foi corretamente recebida.

Nome: WS-Addressing**Domínio:** MENSAGEM**Data Criação:****Autores:** BEA, IBM, Microsoft, TIBCO; Sun**Organização:** W3C**Status:** Submetido**Grupo de Trabalho:** Web Services Addressing Working Group**Data de Aprovação:****Alternativas:** WS-MessageDelivery**Descrição:**

Estabelece um mecanismo independente do transporte para endereçar Web Services e mensagens. Isto é feito através da definição de elementos XML a fim de identificar precisamente origem e destino em um processamento fim a fim. A especificação permite, por exemplo, identificar um endereço para resposta de uma mensagem.

Nome: WS-AtomicTransactions**Domínio:** TRANSAÇÕES**Data Criação:** 1/11/2004 **Autores:** BEA, IBM, Microsoft**Organização:** Ainda não**Status:** Especificação publicada**Grupo de Trabalho:****Data de Aprovação:****Alternativas:****Descrição:**

Esta especificação fornece uma definição para um tipo de coordenação chamado transação atômica, baseado na especificação WS-Coordination. Para tanto são definidos três tipos de coordenação: completa (completion), volátil com execução em duas fases (volatile two-phase commit) e durável com execução em duas fases (durable two-phase commit). Qualquer um destes protocolos pode ser utilizado em atividades de curta duração.

Nome: WS-Attachments **Domínio:** MENSAGEM
Data Criação: **Autores:**
Organização: **Status:** SUBSTITUÍDO
Grupo de Trabalho:
Data de Aprovação: **Alternativas:**
Descrição:
 SUBSTITUÍDO POR SOAP MTOM

Nome: WS-Authorization **Domínio:** SEGURANÇA
Data Criação: **Autores:**
Organização: **Status:** Ainda não publicado!
Grupo de Trabalho:
Data de Aprovação: **Alternativas:**
Descrição:
 Esta especificação definirá um formato e uma linguagem para autorizações

Nome: WS-BPEL **Domínio:** COREOGRAFIA
Data Criação: **Autores:** BEA, Hewlett-Packard, IBM, IONA, Microsoft, NEC, Novell,
 Oracle, SAP, SeeBeyond, Sterling, Sun, Tibco,
 webMethods
Organização: OASIS **Status:** TC em trabalho
Grupo de Trabalho: OASIS Web Services Business Process Execution Language TC
Data de Aprovação: **Alternativas:** WS-Coreography
Descrição:
 Define um modelo e uma gramática para descrever o comportamento de processos de negócio baseados em interações entre o processo e seus parceiros. As entre parceiros são feitas através de Web Services. Um processo define como múltiplas interações entre serviços entre estes parceiros são coordenadas a fim de atingir a meta do negócio. Além disso, a linguagem possui mecanismos para tratar de exceções ou erros de processamento.

Nome: WS-BusinessActivity **Domínio:** TRANSAÇÕES
Data Criação: 1/11/2004 **Autores:** BEA, IBM, Microsoft

Organização: Ainda não **Status:** Especificação publicada

Grupo de Trabalho:

Data de Aprovação: **Alternativas:**

Descrição:

Esta especificação fornece a definição de um contexto de coordenação chamado atividade de negócio (Business Activity), que deve ser utilizado com o framework WS-Coordination. A especificação define dois protocolos diferentes que podem ser utilizados (BusinessAgreementWithParticipantCompletion e BusinessAgreementWithCoordinatorCompletion) que podem ser utilizados em atividades de longa duração.

Nome: WS-CF (WS-CAF) **Domínio:** TRANSAÇÕES
Data Criação: 1/7/2003 **Autores:** Arjuna Technologies, Fujitsu, IONA, Oracle, Sun

Organização: OASIS **Status:** TC em trabalho

Grupo de Trabalho: OASIS Web Services Composite Application Framework (WS-CAF) TC

Data de Aprovação: **Alternativas:** WS-Coordination

Descrição:

Coordenação é um ato de um agente (o coordenador) disseminar informações para um número qualquer de participantes a fim de garantir que todos os participantes obtenham uma mensagem específica. Um coordenador pode aceitar a responsabilidade, por exemplo, de notificar todos os participantes de uma atividade de um resultado comum (por exemplo, a aprovação de um pagamento por pelo WS da operadora de cartão de crédito poderia ser informado a todas as outras Web Services participantes na venda do pacote de turismo para o cliente).

Coordenação é um requisito fundamental em sistemas distribuídos, visto que muitas aplicações podem utilizá-la de formas como, por exemplo, em um fluxo de trabalho (Workflow) uma transação atômica, chaching e replicação, segurança e outras atividades entre negócios. O coordenador também é responsável pela propagação do contexto de coordenação (WS-CTX)

Como não existe um tipo único de coordenação para todos os tipos de problemas esta especificação é extensível e flexível a fim de se adaptar as várias situações possíveis.

Nome: WS-Coordination **Domínio:** TRANSAÇÕES
Data Criação: 1/11/2004 **Autores:** BEA, IBM, Microsoft

Organização: Ainda não **Status:** Especificação publicada

Grupo de Trabalho:

Data de Aprovação: **Alternativas:** WS-CTX (WS-CAF)

Descrição:

Esta especificação descreve um protocolo que permite a coordenação de atividades entre aplicações distribuídas.

Para isto o protocolo permite que uma aplicação crie um contexto de coordenação e se registre neste contexto a fim de propagar suas atividades para outros serviços. O protocolo permite a realização de transações, controle do fluxo de trabalho de forma que a aplicação utilize um protocolo próprio para um ambiente heterogêneo. O protocolo permite a realização de atividades complexas incluindo complexas relações entre os participantes, incluindo atividades de longa duração devido às latências próprias de negócios e das iterações com usuários.

Nome: WS-Coreography **Domínio:** COREOGRAFIA
Data Criação: **Autores:** Various inc., EDS, HP, Oracle, Sun, Tibco

Organização: W3C **Status:**

Grupo de Trabalho: Web Services Choreography Working Group

Data de Aprovação: **Alternativas:** WS-BPEL

Descrição:

Consiste de uma linguagem baseada em XML que descreve colaborações entre partes que realizam uma troca ordenada de mensagens cujo objetivo é uma meta comum de negócio entre estas partes. Desta forma é possível descrever precisamente as colaborações peer-to-peer entre partes independente de sua plataforma ou modelo de programação utilizado neste ambiente.

Nome: WS-CTX (WS-CAF) **Domínio:** TRANSAÇÕES
Data Criação: 1/7/2003 **Autores:** Arjuna Technologies, Fujitsu, IONA, Oracle, Sun

Organização: OASIS **Status:** TC em trabalho

Grupo de Trabalho: OASIS Web Services Composite Application Framework (WS-CAF) TC

Data de Aprovação: **Alternativas:** WS-Coordination

Descrição:

A especificação (WS-CTX) define um escopo de contexto compartilhado para Web Services ou outras especificações. O contexto é compartilhado quando os Web Services realizam atividades relacionadas executando múltiplas interações, por exemplo, um recurso para gerenciamento de dados ou um processo de negócios automatizado. Esta especificação (WS-CTX) é parte da especificação Web Services Composite Application Framework (WS-CAF).

Nome: WS-Discovery **Domínio:** METADADOS
Data Criação: 1/2/2004 **Autores:** BEA, Canon, Intel, Microsoft

Organização: Ainda não **Status:** Especificação publicada

Grupo de Trabalho:

Data de Aprovação:

Alternativas:

Descrição:

Esta especificação define um protocolo a fim de descobrir serviços utilizando a multidifusão (multicasting).

Basicamente sua operação consiste no envio de um pacote de investigação através de processo de multidifusão, e então os serviços encontrados e que atendam aos requisitos da busca irão responder para o serviço que originou o pacote de investigação. Para atender um grande número de serviços a multidifusão pode ser suprimida caso um proxy de descoberto esteja disponível na rede. Para minimizar o número de consultas, os serviços que desejam ser descobertos podem enviar um aviso assim que entrarem ou deixarem a rede.

Nome: WSDL **Domínio:** METADADOS
Data Criação: **Autores:**

Organização: W3C **Status:** Recomendação

Grupo de Trabalho: Web Services Description Working Group

Data de Aprovação: 1/3/2001 **Alternativas:**

Descrição:

Esta especificação define uma linguagem para a descrição abstrata da funcionalidade de um Web Service, bem como um framework para a descrição concreta destes serviços. A descrição abstrata trata dos tipos de dados utilizados nas operações, as mensagens que são transmitidas e as operações disponíveis em um Web Service. A descrição concreta define como a mensagem será transmitida e outros aspectos do serviço, como a sua localização e protocolos de transporte utilizados.

Nome: WSDM **Domínio:** GERENCIAMENTO
Data Criação: **Autores:** Various inc.,BMC, CA, Cisco, IBM, HP, Novell, Tibco

Organização: OASIS **Status:** TC em trabalho

Grupo de Trabalho: OASIS Web Services Distributed Management (WSDM) TC

Data de Aprovação:

Alternativas:

Descrição:

Existem duas partes para o gerenciamento distribuído de Web Services: O gerenciamento utilizando Web Services e o gerenciamento dos próprios Web Services. Esta especificação aborda o primeiro caso. O gerenciamento utilizando Web Services (MUWS) estende a tecnologia de Web Services, facilitando e possibilitando um uso mais eficiente de sistemas de TI de gerenciamento através do uso de um framework comum para as interfaces de gerenciamento, permitindo a interoperabilidade entre vários sistemas distribuídos.

Nome: WS-Enumeration**Domínio:** MENSAGEM**Data Criação:** 1/9/2004 **Autores:** BEA, CA, Microsoft, Sonic Software, Systinet**Organização:** Ainda não**Status:** Especificação publicada**Grupo de Trabalho:****Data de Aprovação:****Alternativas:****Descrição:**

Descreve um protocolo a fim de enumerar uma sequência de elementos XML a fim de atravessar, por exemplo, um log, filas de mensagens ou outros modelos lineares. O protocolo permite que uma aplicação solicite itens de uma lista de dados que é mantida por um Web Service. Desta forma, o protocolo é útil para a leitura de logs de eventos, filas de mensagens, streaming e outras aplicações onde o modelo pedido-simples/resposta-simples é insuficiente para a transferência de grandes conjuntos de dados.

Nome: WS-Eventing**Domínio:** MENSAGEM**Data Criação:** 1/8/2004 **Autores:** Microsoft, BEA, TIBCO, CA, IBM, Sun**Organização:** Ainda não**Status:** Especificação publicada**Grupo de Trabalho:****Data de Aprovação:****Alternativas:** WS-Notification**Descrição:**

Define um protocolo para que um Web Service (subscriber) possa registrar seu interesse em receber mensagens sobre eventos (notificações) ocorridos em outro Web Service (origem). O subscriber pode interagir com um terceiro Web Service (chamado subscription manager) designado pelo Web Service origem do evento. O registro pode ter uma duração limitada no tempo, e seu cancelamento ou renovação pode ser feito pelo "subscription manager".

Nome: WS-Federation**Domínio:** SEGURANÇA**Data Criação:** 1/7/2003 **Autores:** IBM, Microsoft, BEA, RSA Security, Verisign**Organização:** Ainda não**Status:** Especificação publicada**Grupo de Trabalho:****Data de Aprovação:****Alternativas:** Liberty Alliance**Descrição:**

Define mecanismos que são utilizados a possibilitar a federação de identidades, contas, atributos, autenticação e autorização entre Web Services existentes em diferentes domínios de confiança. Entre os principais mecanismos contidos nesta especificação pode ser destacado: Permitir o compartilhamento de identidade, autenticação e autorização; Troca de tokens de segurança e gerenciamento de confiança (Brokering of trust) e a mostra ou ocultação de informações de identidade e outros atributos.

Nome: WS-Manageability**Domínio:** GERENCIAMENTO**Data Criação:****Autores:** CA, IBM, Talking Blocks**Organização:** OASIS (WSDM) **Status:** TC em trabalho**Grupo de Trabalho:** OASIS Web Services Distributed Management (WSDM) TC**Data de Aprovação:****Alternativas:****Descrição:**

Esta especificação introduz conceitos gerais para um modelo de gerenciamento relacionado a tópicos de gerência (identificação, configuração, estado, métricas e relacionamentos) e os aspectos (propriedades, operações e evento) usados para definir estes tópicos. Apesar destes conceitos serem aplicáveis a qualquer recurso de TI, os mesmos são utilizados nesta especificação para mostrar o gerenciamento de Web Services.

Nome: WS-MessageDelivery**Domínio:** MENSAGEM**Data Criação:****Autores:**

Oracle, Arjuna, Cyclone Commerce, Enigmatec, IONA, Nokia, SeeBeyond, Sun

Organização: W3C**Status:** Submetido**Grupo de Trabalho:****Data de Aprovação:**1/4/2004 **Alternativas:** WS-Addressing**Descrição:**

Especifica um conjunto de propriedades de entrega de mensagens que permitem o uso de padrões de trocas de mensagens associados com documentos WSDL.

Nome: WS-MetaDataExchange**Domínio:** METADADOS**Data Criação:**

1/9/2004

Autores:

BEA, IBM, Microsoft, SAP, CA, Sun e webMethods

Organização: Ainda não**Status:** Especificação publicada**Grupo de Trabalho:****Data de Aprovação:****Alternativas:****Descrição:**

Esta especificação define três serviços do tipo pedido-resposta para obter os seguintes metadados de um Web Service: a política (WS-Policy) associada com o mesmo, a WSDL associada com o serviço e o namespace associado e o XML Schema associado a este namespace.

Nome: WSMF**Domínio:** GERENCIAMENTO**Data Criação:****Autores:** HP**Organização:** OASIS (WSDM) **Status:** TC em trabalho**Grupo de Trabalho:** OASIS Web Services Distributed Management (WSDM) TC**Data de Aprovação:****Alternativas:****Descrição:**

Arquitetura lógica para o gerenciamento de recursos, incluindo os próprios Web Services, através de Web Services.

Baseado na noção de objetos gerenciáveis e suas relações. Um objeto gerenciável essencialmente representa um recurso e suas interfaces de gerenciamento.

Nome: WS-Notification**Domínio:** MENSAGEM**Data Criação:**

1/8/2004

Autores: IBM, Akamai, HP, SAP, Sonic Software, The Globus Alliance,

TIBCO

Organização: OASIS**Status:** Especificação publicada**Grupo de Trabalho:** OASIS Web Services Notification (WSN) TC**Data de Aprovação:****Alternativas:** WS-Eventing**Descrição:**

Especifica o padrão de notificação em Web Services, onde um provedor de serviço ou outra entidade envia mensagens baseadas em uma inscrição (subscription) ou registro. Define como o padrão publica/inscreve, comumente utilizado em produtos de middleware pode ser conduzido através do uso de Web Services.

Nome: WS-Policy**Domínio:** METADADOS**Data Criação:**

1/9/2004

Autores: BEA, IBM, Microsoft, SAP**Organização:** Ainda não**Status:** Especificação publicada**Grupo de Trabalho:****Data de Aprovação:****Alternativas:****Descrição:**

Esta especificação contém uma gramática a fim de expressar as capacidades, requisitos e características gerais de entidades presentes em Web Services, como políticas. Como exemplos de políticas, podemos citar, esquemas de autenticação; protocolo de transporte a ser utilizado; características de QoS; políticas de segurança. A especificação não trata da maneira como estas políticas são encontradas e anexadas a um Web Service.

Nome: WS-PolicyAttachment **Domínio:** METADADOS
Data Criação: 1/9/2004 **Autores:** BEA, IBM, Microsoft, SAP

Organização: Ainda não **Status:** Especificação publicada

Grupo de Trabalho:

Data de Aprovação: **Alternativas:**

Descrição:

Esta especificação define dois mecanismos gerais para associar políticas com os sujeitos aos quais as mesmas se relacionam. Desta forma é mostrado como associar políticas em definições WSDL, em Web Services já implantados e em entidades UDDI.

Nome: WS-Privacy **Domínio:** SEGURANÇA
Data Criação: **Autores:**

Organização: **Status:** Ainda não publicado!

Grupo de Trabalho:

Data de Aprovação: **Alternativas:**

Descrição:

Esta especificação descreverá um modelo em que Web Services e consumidores irão estabelecer uma política de privacidade.

Nome: WS-Provisioning **Domínio:** GERENCIAMENTO
Data Criação: **Autores:** IBM

Organização: OASIS **Status:** Padrão OASIS

Grupo de Trabalho: OASIS Provisioning Services TC

Data de Aprovação: 1/11/2003 **Alternativas:**

Descrição:

Descreve uma API e os esquemas necessários para facilitar a interoperabilidade entre sistemas de provisionamento (provisioning) e permitir que fornecedores de software possam oferecer sistemas de provisionamento consistentes abertos.

A especificação foca muitos dos problemas encontrados por fornecedores de sistemas de provisionamento no uso de protocolos, que normalmente são baseados em conceitos de diretório e oferece uma nova perspectiva através da proposta de Web Services de provisionamento descritos usando WSDL e XML.

Para tanto é definido um modelo que contém as entidades primárias e as operações comuns a sistemas incluindo o provisionamento e a liberação de recursos e ainda fornece um mecanismo para controlar o ciclo de vida deste recursos.

Nome: WS-Referral **Domínio:** MENSAGEM
Data Criação: **Autores:**

Organização: **Status:** SUBSTITUÍDO

Grupo de Trabalho:
Data de Aprovação: **Alternativas:**
Descrição:
 SUBSTITUÍDO POR WS-Addressing

Nome: WS-Reliability **Domínio:** MENSAGEM CONFIÁVEL
Data Criação: 1/8/2004 **Autores:** Various inc., Fujitsu, Hitachi, IONA, NEC, Nokia, Oracle, SAP, Sonic, Sun

Organização: OASIS **Status:** TC em trabalho

Grupo de Trabalho: OASIS Web Services Reliable Messaging (WSRM) TC
Data de Aprovação: **Alternativas:** WS-Reliable Messaging

Descrição:
 Estabelece um padrão a fim de garantir a entrega ponto a ponto de mensagens. A especificação define uma série de padrões para trocas de mensagens a fim de implementar os seguintes serviços: garantia de entrega; entrega ordenada; remoção de duplicatas; padrão para envio de reposta; tempo de inatividade e tempo de duração da mensagem.

Nome: WS-Reliable Messaging **Domínio:** MENSAGEM CONFIÁVEL
Data Criação: **Autores:** BEA, IBM, Microsoft, TIBCO

Organização: Ainda não **Status:** Especificação publicada

Grupo de Trabalho:
Data de Aprovação: **Alternativas:** WS-Reliability

Descrição:
 Descreve um protocolo que permite a entrega confiável de mensagens entre aplicações distribuídas na presença de falhas de software, do sistema ou da rede. Para tanto o protocolo define elementos a fim de ordenar as mensagens, solicitar confirmações positivas (acknowledgment) e negativas, entre outros.

Nome: WS-Routing **Domínio:** MENSAGEM
Data Criação: **Autores:**

Organização: **Status:** SUBSTITUÍDO

Grupo de Trabalho:
Data de Aprovação: **Alternativas:**
Descrição:
 SUBSTITUÍDO POR WS-Addressing

Nome: WSRP **Domínio:** PROCESSOS DE NEGÓCIO
Data Criação: **Autores:** Various inc., BEA, Bowstreet, IBM, Novell, Oracle, Plumtree,
 SAP, Sun

Organização: OASIS **Status:** Padrão OASIS
Grupo de Trabalho: OASIS Web Services for Remote Portlets (WSRP) TC
Data de Aprovação: **Alternativas:**
Descrição:

Portais e outras aplicações da Web obtêm informações de diferentes fontes e agregam as mesmas de forma que a informação possa ser facilmente consumida por um usuário final. Esta especificação trata deste problema através da introdução de uma interface orientada à apresentação que permite a inclusão e a interação com o conteúdo através de um Web Service. Este Web Service fornece tanto a lógica da aplicação quanto a sua apresentação.

Nome: WS-SecureConversation **Domínio:** SEGURANÇA
Data Criação: 1/5/2004 **Autores:** IBM, Microsoft, RSA, Verisign

Organização: Ainda não **Status:** Especificação publicada

Grupo de Trabalho:
Data de Aprovação: **Alternativas:**
Descrição:

Define extensões a fim de permitir o estabelecimento e o compartilhamento de um contexto de segurança e a criação de uma chave de sessão. O uso de contextos permite incrementar a performance geral do sistema e a segurança das futuras trocas de mensagens. O modelo de autenticação para o contexto requer uma comunicação adicional caso a autenticação aconteça antes da troca de mensagens requeridas pela aplicação.

Nome: WS-Security **Domínio:** SEGURANÇA
Data Criação: **Autores:** IBM, Microsoft, Verisign, Sun

Organização: OASIS **Status:** Padrão OASIS
Grupo de Trabalho: OASIS Web Services Security (WSS) TC
Data de Aprovação: 1/3/2004 **Alternativas:**

Descrição:

O objetivo desta especificação é possibilitar a troca segura de mensagens SOAP com o objetivo de garantir sua integridade e confidencialidade. Para isto a especificação utiliza assinaturas e a criptografia do cabeçalho de mensagens SOAP. Além disso, especifica como anexar certificados x.509 e tiquetes Kerberos às mensagens.

Nome: WS-SecurityPolicy **Domínio:** METADADOS
Data Criação: 1/12/2002 **Autores:** IBM, Microsoft, RSA, Verisign

Organização: Ainda não **Status:** Especificação publicada

Grupo de Trabalho:
Data de Aprovação: **Alternativas:**

Descrição:

Várias especificações de padrões em Web Services indicam as declarações de políticas para uso com WS-Policy.

Como a como WS-Security foi publicado em uma data anterior que WS-Policy, esta especificação trata destas declarações. Basicamente é definido um elemento <wsp:Policy> que contém as propriedades de segurança relativas à Web Services.

Nome: WS-Transaction **Domínio:** TRANSAÇÕES
Data Criação: **Autores:**

Organização: **Status:** SUBSTITUÍDO

Grupo de Trabalho:
Data de Aprovação: **Alternativas:** WS-TXM (WS-CAF)

Descrição:

SUBSTITUÍDO POR WS-AtomicTransactions

Nome: WS-Transfer **Domínio:** MENSAGEM
Data Criação: 1/9/2004 **Autores:** BEA, CA, Microsoft, Sonic Software, Systinet

Organização: Ainda não **Status:** Especificação publicada

Grupo de Trabalho:

Data de Aprovação: **Alternativas:**

Descrição:

Define um protocolo geral para acessar representações de recursos baseados em Web Service. Através deste protocolo é possível utilizar um conjunto de ações (GET, POST, PUT, DELETE) utilizando SOAP. Basicamente estas operações são ações definidas pelo protocolo WS-Addressing

Nome: WS-Trust **Domínio:** SEGURANÇA
Data Criação: 1/5/2004 **Autores:** IBM, Microsoft, RSA, Verisign

Organização: Ainda não **Status:** Especificação publicada

Grupo de Trabalho:

Data de Aprovação: **Alternativas:**

Descrição:

Esta especificação usa os mecanismos básicos definidos em WS-Security e define primitivas adicionais para a troca segura de tokens a fim de emitir e disseminar credenciais entre diferentes domínios de confiança.

A fim de permitir uma comunicação segura entre duas partes, ambas devem trocar credenciais de segurança e para isto é necessário que cada parte determine se elas realmente podem confiar uma na outra.

Para conseguir este objetivo, esta especificação introduz elementos que permitem requisitar credenciais de segurança, estabelecer e quebrar relações de segurança

Nome: WS-TXM (WS-CAF) **Domínio:** TRANSAÇÕES
Data Criação: 1/7/2003 **Autores:** Arjuna Technologies, Fujitsu, IONA, Oracle, Sun

Organização: OASIS **Status:** TC em trabalho

Grupo de Trabalho: OASIS Web Services Composite Application Framework (WS-CAF) TC

Data de Aprovação: **Alternativas:** WS-AtomicTransactions

Descrição:

A técnica de Transações atômicas é bem conhecida para garantir a consistência na presença de falhas. As propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) garantem que mesmo em complexas aplicações de negócios a consistência é preservada, apesar de acessos concorrentes e falhos. Este recurso é de extrema importância, principalmente quando recursos remotos estiverem envolvidos.

Esta especificação tem por objetivos, prover as definições básicas de um serviço de transação para o ambiente de Web Services, através da definição de mapeamentos no ambiente de Web Services como mensagens SOAP, definições de cabeçalho, entre outros.

Nome: XML ENCRYPTION

Domínio: XML

Data Criação:

Autores:

Organização: W3C

Status: Recomendação

Grupo de Trabalho: XML Encryption WG

Data de Aprovação: 1/12/2002 **Alternativas:**

Descrição:

Contém regras sobre como deve ser o processo para criptografar e decriptografar mensagens XML.

Nome: XML SCHEMA

Domínio: XML

Data Criação:

Autores:

Organização: W3C

Status: Recomendação

Grupo de Trabalho: XML Schema Working Group

Data de Aprovação: 1/5/2001 **Alternativas:**

Descrição:

Define a estrutura de um documento XML, ou seja, o XML Schema define as tags que serão utilizadas e em qual ordem.

Nome: XML SIGNATURE

Domínio: XML

Data Criação:

Autores:

Organização: W3C / IETF

Status: Recomendação

Grupo de Trabalho: XML Signature WG

Data de Aprovação: 1/2/2002 **Alternativas:**

Descrição:

Especifica as regras de processamento e sintaxe para criação de assinaturas digitais em sobre documentos XML com o objetivo de prover integridade da mensagem, verificação de sua autenticidade e do assinante.