**UNIVERSIDADE FEDERAL DE SANTA CATARINA**
**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS**

Igor Thiago Marques Mendonça

# ABORDAGEM SEGURA DE GERENCIAMENTO REMOTO PARA A INDÚSTRIA DE UTILIDADES USANDO SERVIÇOS WEB

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do Grau de Mestre em Engenharia de Automação e Sistemas, Área de Concentração em Controle, Automação e Sistemas.
Orientador: Prof. Dr. Joni da Silva Fraga
Co-orientador: Prof. Dr. Roberto Alexandre Dias

Florianópolis

2011

Catalogação na fonte pela Biblioteca Universitária
da
Universidade Federal de Santa Catarina

M539a Mendonça, Igor Thiago Marques
Abordagem segura de gerenciamento remoto para a indústria de utilidades usando serviços Web [dissertação] / Igor Thiago Marques Mendonça ; orientador, Joni da Silva Fraga. – Florianópolis, SC, 2011.
139 p.: il., grafs., tabs.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de sistemas. 2. Sistemas de energia elétrica - Medição. 3. Serviços da Web. 4. Segurança de sistemas. 5. Computadores - Medidas de segurança. 6. Sistemas embutidos de computador. I. Fraga, Joni da Silva. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Título.

CDU 621.3-231.2(021)

Igor Thiago Marques Mendonça

**UMA ABORDAGEM SEGURA DE GERENCIAMENTO REMOTO PARA A INDÚSTRIA DE UTILIDADES USANDO SERVIÇOS WEB**

Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia de Automação e Sistemas, Área de Concentração em Controle, Automação e Sistemas, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina.

Florianópolis, 28 de Março de 2011.

_____
Prof. Joni da Silva Fraga, Dr.
Orientador

_____
Prof. Roberto Alexandre Dias, Dr.
Co-orientador

_____
Prof. José Eduardo Ribeiro Cury, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia de Automação e Sistemas

**Banca Examinadora:**

_____
Prof. Joni da Silva Fraga, Dr.
Presidente

_____
Prof. Altair Olivo Santin, Dr.

_____
Prof. Emerson Ribeiro de Mello, Dr.

_____
Prof. Frank Augusto Siqueira, Dr.

à família ...

## AGRADECIMENTOS

Principalmente a Deus e à minha família ...

Aos meus orientadores Joni e Roberto pela atenção e paciência ... Em especial ao prof. Roberto que me acolheu em seu Laboratório no IFSC.

Aos meus amigos do NERsD: Everson, Tiago, Gregory, Priscila e Reginaldo. Que me apoiaram e ajudaram em diversas fases deste trabalho.

Aos meus amigos da República "Pé Vermeio", cujo convívio me proporcionou grande aprendizado sobre a vida.

Ao Departamento de Automação e Sistemas e à Universidade Federal de Santa Catarina por terem proporcionado um excelente ambiente de aprendizado.

A persistência é o caminho do êxito.

(Charles Chaplin)

# RESUMO

Atualmente a rede de distribuição de energia elétrica para consumidores secundários, aqueles que estão na faixa de 110 ou 220 volts, é carente de automação, sendo que as medições, desligamentos e ligamentos têm de ser realizados com o deslocamento de funcionários da concessionária até o local do consumidor. Por esses motivos, a adoção de sistemas de supervisão e controle traria inúmeras vantagens para os clientes e a concessionária. A medição remota reduziria as perdas comerciais, enquanto que para os clientes haveria a comodidade de acompanhar, em tempo real, o seu consumo.

Neste ambiente, cuja malha de distribuição de energia elétrica possui centenas de milhares de componentes numa região metropolitana e estes estão ligados por cabos de energia elétrica, o uso da tecnologia de *Power Line Communication* (PLC) se torna ideal. Para a integração destes milhares de componentes a tecnologia de Serviços *Web* aparece como solução adequada, ou seja, serve como *middleware* para integração destes componentes em ambientes abertos como a Internet.

A solução de *middleware* através de Serviços *Web* aparece através do uso de *Devices Profile for Web Service* (DPWS), pois neste cenário existem dispositivos com restrições computacionais, os quais são alvo de tal especificação. Porém, os aspectos de segurança no perfil DPWS se resumem a indicação de uso da SSL/TLS, desconsiderando, então, o roteamento em nível de mensagens.

O objetivo desta dissertação é fazer um estudo de segurança para um sistema de controle e medição de energia elétrica, considerando a infraestrutura existente hoje, incluindo o uso de PLC nos cabos de transmissão de energia elétrica. Foi assumida a escolha por padrões que adotem especificações de Serviço *Web*. Como consequência deste estudo, o DPWS foi estendido para atender os requisitos de segurança que sua especificação não trata. A extensão realizada segue os padrões definidos na WS-Security e WS-SecureConversation. Um protótipo do modelo estendido é implementado como forma de verificar a aplicabilidade do mesmo no sistema de controle e medição de dispositivos.



**Palavras-chave**: Sistemas de medição, SOA, Serviços *Web*, DPWS, Segurança, WS-Security, Sistemas Embarcados.

## ABSTRACT

Currently, the electric power distribution network for secondary consumers, those which range from 110 or 220 volts, is in need of automation. Also, the measurements and the services have to be done with the displacement of distribution companies' employees of the to the consumer's place. However, the adoption of supervision and control systems would bring numerous benefits for customers and for the distribution companies. The remote measurement would reduce the commercial losses, while for the clients there would be the convenience to follow, in real time, its consumption.

In this environment, whose electric power distribution network has hundreds of thousands of components in a metropolitan region and these are connected by power cables, the use of Power Line Communication (PLC) technology becomes ideal. For the integration of these thousands of components, the Web Services technology seems to be an adequate solution, i. e., serves as middleware to integrate these components in open environments such as the Internet.

The middleware solution through Web Services appears via the use of Devices Profile for Web Service (DPWS), because, in this scenario, there are devices with computational restrictions, which are the target of such a specification. However, the security aspects of the profile DPWS are summed up to the indication of the use of SSL/TLS, disregarding then messages routing.

The objective of this thesis is to do a study on safety for a control system and electrical energy measurement, considering the existing infrastructure today, including the use of PLC in cables of electric power transmission. It was decided to choose standards that adopt Web Services. As a result of this study, the DPWS was extended to meet the security requirements that its specification does not deal with. The extension carried out follows the standards defined in WS-Security and WS-SecureConversation. A prototype of the extended model is implemented as a way to verify its applicability on the control and devices measurement system.



**Keywords**: Measuring systems, SOA, Web Services, DPWS, Security, WS-Security, Embedded Systems.

## LISTA DE FIGURAS

**LISTA DE CÓDIGOS**

## LISTA DE TABELAS

**SUMÁRIO**

# 1 INTRODUÇÃO

## 1.1 DESCRIÇÃO DE CONTEXTO

Os Sistemas de Medição e Atuação eram instalados para coletar dados de medição em subestações de distribuição de energia elétrica normalmente usando linhas telefônicas dedicadas. Limitados pela estreita largura de banda, seus protocolos eram configurados para operar em banda estreita e, normalmente, os tempos necessários para configuração e mapeamento de dados era bastante significativos. Com o avanço dos meios de comunicação usando a Internet, muitas das funções destes sistemas vêm sendo transferidas para acessos via a rede mundial. E neste caso a questão fundamental passa a ser a segurança destas funções.

Neste, muitas experiências descritas na literatura sobre o desenvolvimento de sistemas de medição que fazem o uso da Internet para a aquisição remota do consumo de energia elétrica. Estas experiências são baseadas no protocolo *Simple Network Management Protocol* (SNMP) (ALMQVIST e WIKSTROM, 1994) ou numa integração deste com Serviço *Web* (PRAS *et al.,* 2004), criando elementos gateways que separam as duas tecnologias: o SNMP é usado nos níveis mais baixos (de dispositivos de medição) e Serviços *Web* são usados em nível de Internet permitindo ao sistema ganho em flexibilidade e em facilidade de acesso na rede mundial.

Atualmente Serviços *Web* estão sendo objeto de padrões e especificações que visam à implementação dos mesmos diretamente sobre dispositivos embarcados. Estas especificações fornecem as abstrações necessárias para tornar estes dispositivos facilmente integráveis em ambientes de larga escala e heterogêneos. A especificação *Devices Profile for Web Services* (DPWS) (OASIS, 2009d) é um destes esforços recentes cujo objetivo é descrever um conjunto mínimo de serviços e infraestrutura para levar Serviços *Web* a dispositivos embarcados. Estas especificações, ao integrarem os pequenos dispositivos à Arquitetura Orientada a Serviços (SOA), tornam possíveis aplicações como às citadas acima, sem a necessidade de recorrer à integração de Serviços *Web* com tecnologias como o SNMP. Com esta nova realidade, vários projetos como SIRENA[1],

[1] www.sirena-itea.org

SODA[2] e SOCRADES[3] têm sido desenvolvidos, levando essa tecnologia recente para segmentos antes refratários ao SOA, como a indústria automotiva, de automação residencial e ao chão de fábrica. Um fato facilmente constatado nas especificações do DPWS é que as mesmas são relativamente superficiais em relação à segurança.

Por outro lado, se verificarmos os sistemas de distribuição de energia elétrica nas cidades, chegaremos à conclusão que estes sistemas formam extensas redes, com centenas de milhares de componentes, interligados através de cabos de energia. Essa rede tornou-se um caminho natural de comunicação através do uso da tecnologia de *Power Line Communication* (PLC), proporcionando alcances significativos para comunicações no contexto das cidades atuais.

Nessas redes de distribuição existe um importante componente chamado medidor residencial responsável por calcular o consumo individual de cada cliente da concessionária. Este medidor que antes era do tipo eletromecânico, começa a ser substituído por medidores eletrônicos com arquiteturas mais adequadas ao estágio tecnológico atual. Estes medidores possibilitam obter medições envolvendo não apenas a energia elétrica consumida, mas também parâmetros que indicam a qualidade da mesma. Além disto, estes novos dispositivos possuem suporte para possíveis comunicações com a concessionária.

Esta dissertação descreve a nossa experiência no projeto de um sistema de controle e medição de dispositivos proposto para a aquisição e o controle de dispositivos via Internet. Este sistema é aplicado na aquisição de medidas de consumo de energia elétrica em medidores de residenciais. O acesso a estas medidas é feito fazendo uso das próprias linhas de distribuição de energia elétrica e de recursos de redes metropolitanas sem fio.

O uso da Internet, de redes metropolitanas sem fio e da infraestrutura de distribuição de energia elétrica como meios para interações se tornam um fator de risco para a segurança das informações e do próprio sistema de medição. Sem mecanismos que garantam as propriedades de segurança das mensagens de medição e das que objetivam o controle de dispositivos medidores a concessionária ficaria vulnerável a fraudes e a revelação de dados confidenciais de seus clientes. Poucos trabalhos que se propõem ao uso da Internet para a aquisição de medidas se preocupam com a segurança

[2] www.soda-itea.org
[3] www.socrades.eu

Segundo o Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil[4] (CERT), grupo responsável por tratar incidentes de segurança em computadores que envolvam redes conectadas à Internet brasileira, em 2009 quase 70% dos incidentes reportados estava relacionado à fraude. Considerando que a indústria de utilidades possa usar redes abertas, como a Internet e redes wireless, para tráfego de informação de seus consumidores, a atenção aos aspectos de segurança de informação se torna uma premissa. As indústrias de utilidades não estão livres de ataques cibernéticos, haja vista algumas reportagens que a mídia noticiou: Hackers atacam e deixam cidades sem luz, diz CIA[5], Ataque ao sistema de águas de Marrochy Shire[6].

Diante disto, este trabalho se propôs a um estudo sobre segurança nestes sistemas de medição de energia elétrica como forma de garantir a confidencialidade e a integridade das informações transmitidas. Com o uso do DPWS desenvolvemos um sistema de medição e controle de dispositivos remotos para o cenário de distribuição de energia elétrica. Dentre as proposições e desenvolvimentos realizados neste trabalho, nos vimos diante das limitações nas especificações do DPWS sobre segurança. Este fato foi determinante para que também fizéssemos uma proposição de extensão de segurança para o mesmo. Com o uso do DPWS estendido no sistema proposto podemos garantir a autenticidade, integridade e confidencialidade das mensagens transmitidas. Um protótipo foi por fim desenvolvido para avaliar a funcionalidade desta proposta de sistema de medição e das extensões do DPWS diante de questões de desempenho e de segurança. Vários testes foram realizados que comprovam a eficiência das escolhas feitas neste trabalho.

## 1.2 MOTIVAÇÃO

Uma grande quantidade de trabalhos vem sendo desenvolvido para o controle e monitoramento do setor de distribuição de energia elétrica, mas os aspectos de segurança geralmente são abordados de

[4] www.cert.br
[5] tecnologia.terra.com.br/interna/0,,OI2256627-EI4805,00.html
[6] www.theregister.co.uk/2001/10/31/hacker_jailed_for_revenge_sewage

maneira superficial e é desta situação que surgiu a motivação para o desenvolvimento deste trabalho.

Ao mesmo tempo em que a adoção de sistemas de supervisão e controle no setor de distribuição de energia elétrica traz inúmeras vantagens para os clientes e a concessionária, são introduzidos fatores de risco relacionados à segurança da informação, uma vez que as transmissões de dados devem ocorrer de forma íntegra e sigilosa.

Do ponto de vista da concessionária, fazer a medição remota reduz perdas monetárias pelo aumento dos níveis de leituras de consumo. No lado do cliente há o aumento da satisfação pela maior exatidão de sistemas telemétricos e o acompanhamento em tempo real, mas o cliente não pode ter suas informações de consumo reveladas a pessoas não autorizadas.

A automação nas diversas áreas das indústrias de utilidades já é uma realidade. Essas indústrias são consideradas infraestruturas críticas e de segurança nacional, e sua paralisação, interrupção significativa ou degradação dos serviços acarretam graves consequências sociais, por isso, devem possuir mecanismos para garantir sua disponibilidade.

## 1.3 OBJETIVOS DA DISSERTAÇÃO

O objetivo geral desta dissertação é fazer um estudo sobre a segurança em sistemas de medição que atuam em ambientes abertos. Sabendo que estes sistemas são em sua maioria constituídos de uma eletrônica embarcada, procuramos fazer com que estes estudos fossem focados em aspectos de *middleware*, que é a melhor forma de gerenciar os milhares de componentes que normalmente formam estes sistemas de medição. A nossa proposta de middleware foi então de usar conceitos de SOA através do DPWS. Porém a fragilidade das especificações do DPWS em relação à segurança fez com que tivéssemos que propor extensões de segurança para este perfil.

Baseados no objetivo geral são detalhados os objetivos específicos que guiaram este trabalho:

(1) Propor uma arquitetura para sistema de medição que tenha uma representatividade com a realidade dos sistemas de distribuição de energia elétrica;

(2) Desenvolver uma arquitetura de software para o sistema de medição que explicite as funcionalidades da aplicação escolhida. Esta arquitetura deve ser orientada a serviços. A

tecnologia de Serviços *Web* é a mais notória representante dos conceitos de SOA e está se tornando uma poderosa ferramenta de integração de vários níveis de aplicação via Internet;

(3) Adequar as especificações do Perfil DPWS às necessidades de segurança nestes ambientes abertos;

(4) Projetar e desenvolver um protótipo que reproduza um sistema de medição para o cenário do setor de distribuição de energia elétrica;

(5) Analisar e testar o protótipo desenvolvido com o intuito de avaliar a viabilidade da solução proposta.

Nós esperamos que estes estudos e objetivos, uma vez cumpridos, possam mostrar a viabilidade da integração de milhares de dispositivos de medição de consumo de residências sobre energia elétrica via Internet com os servidores das concessionárias, resultado num sistema bastante eficaz e automático.

## 1.4 ORGANIZAÇÃO DO TEXTO

A dissertação está dividida em sete capítulos. O capítulo 2 traz um embasamento sobre os principais conceitos de Segurança Computacional como forma de localizar o leitor sobre o tema em que este trabalho está inserido.

No capítulo 3, é feita uma revisão sobre a Arquitetura Orientada a Serviços (SOA: *Service Oriented Architecture*), e as principais especificações de Serviços *Web*. As especificações que são dirigidas para segurança e que possuem ligação com o contexto dos nossos trabalhos nesta dissertação são também examinadas neste capítulo.

O capítulo 4 trata a especificação DPWS e seus protocolos. Esta especificação visa atender um cenário de dispositivos com recursos computacionais restritos. Com o uso de DPWS é possível integrar estes dispositivos com os níveis mais altos das aplicações via Internet.

O capítulo 5 introduz o modelo de segurança proposto nesta dissertação. Este modelo tem como base um Serviço de Segurança cujo objetivo é o tratamento dos aspectos de segurança de aplicações de forma transparente e como serviço *built-in* nos dispositivos, seguindo a ótica do DPWS e seu modelo computacional. As trocas para estes ambientes dinâmicos que envolvem descoberta são adaptadas no modelo

proposto, seguindo as especificações WS-Security e WS-SecureConversation.

No capítulo 6 é definido e desenvolvido o cenário de aplicação para o sistema de Medição e Aquisição. O modelo de segurança proposto para o DPWS é então usado e leva em consideração alguns aspectos desta aplicação em ambiente totalmente aberto. Um protótipo desenvolvido é descrito nas suas particularidades de implementação. Por fim são feitas análises e testes dos resultados, levando em consideração o desempenho e a segurança das nossas propostas. As nossas propostas são também neste capítulo analisadas à luz dos principais trabalhos correlatos no sentido de evidenciar nossas contribuições. Por fim, no capítulo 7 apresentamos nossas conclusões sobre os trabalhos desenvolvidos no âmbito deste mestrado.

# 2 SEGURANÇA COMPUTACIONAL

Este capítulo faz uma revisão bibliográfica de conceitos e princípios de segurança computacional, apresentando as soluções e padrões existentes.

## 2.1 INTRODUÇÃO

A segurança em sistemas computacionais é um requisito de qualidade de serviço que se faz fundamental nos dias de hoje devido à penetração destes sistemas em nossas vidas. Hoje em dia, os sistemas informáticos estão espalhados por todas as áreas da sociedade moderna. A conectividade entre estes sistemas está também se tornando rapidamente global, através da Internet. Com isto no armazenamento controlado de informações e nas comunicações passam a ser determinantes requisitos de segurança nestes sistemas. A evolução dos computadores fez com que o significado, a granularidade e as implicações de segurança computacional (*computer security*) tenham mudado no decorrer de todos estes anos (LANDWEHR, 1981). Mas, na essência, os tipos de violações de segurança a serem evitados sobre informações e sistemas, continuam os mesmos, ou seja: o acesso não autorizado de informações, modificações não autorizadas, e a indisponibilidade de serviços e informações (negação de serviço) no sistema.

A segurança computacional está fundamentada em três propriedades básicas que devem ser mantidas para que um sistema seja considerado seguro. Estas propriedades são: a confidencialidade, a integridade e a disponibilidade. Outros autores consideram ainda, acrescidas às citadas acima, as propriedades de autenticidade e a de não repúdio (LANDWEHR, 1981).

A propriedade de confidencialidade garante que as informações não são reveladas sem autorização. A integridade é a propriedade que garante que as informações não são modificadas sem autorização. A propriedade de disponibilidade garante que as informações sempre estarão disponíveis a usuários autorizados.

A propriedade de autenticidade visa garantir a identificação de um usuário ou mensagem e a propriedade de não repúdio garante que um participante de comunicação não possa negá-la posteriormente.

Na sequência introduzimos os principais conceitos de segurança que serão usados no texto deste trabalho.

## 2.2 CONCEITOS SOBRE SEGURANÇA COMPUTACIONAL

### 2.2.1 Segurança Computacional

"A segurança corresponde a uma qualidade de serviço que é obtida através de meios, técnicas e mecanismos que visam à manutenção das propriedades de **confidencialidade**, **integridade** e **disponibilidade** de informações e recursos em um sistema computacional" (BRINKLEY e SCHELL, 1995; JOSHI *et al.*, 2001)

### 2.2.2 Política de Segurança

O termo política de segurança, dependendo do contexto, pode ter diferentes significados. Considerando a administração de uma instituição, a política de segurança pode definir um conjunto de leis e práticas que regem como a instituição gerencia, protege e distribui suas informações. Estas definições devem, ainda, se refletir nos sistemas computacionais como um conjunto de regras que especificam como um sistema provê os seus serviços mantendo as propriedades de confidencialidade, integridade e disponibilidade. (SANTIN, 2004; SHIRLEY, 2007)

As políticas de segurança em sistemas computacionais são classificadas em duas categorias: as discricionárias e as obrigatórias. Nas discricionárias é concedido ao proprietário ou responsável pela informação o livre acesso. Nas obrigatórias, também conhecidas por mandatórias ou não discricionárias, as autorizações de acesso são definidas através de um conjunto incontornável de regras globais que expressam algum tipo de organização envolvendo a segurança das informações no sistema como um todo (SANTIN, 2004).

### 2.2.3 Modelos de segurança

Segundo (GOGUEN e MESAJUER, 1982), modelos de segurança descrevem formalmente o comportamento de políticas de segurança em um sistema. Estes modelos são representados na forma de um conjunto de entidades e relacionamentos. Na literatura (SANDHU e SAMARATI, 1996), os modelos de segurança para controle de acesso

estão divididos em três tipos básicos: discricionários (DAC: *Discretionary Access Control*), baseados em regras ou obrigatórios (MAC: *Mandatory Access Control*) e os baseados em papéis (RBAC - *Role-Based Access Control*).

Não desenvolvemos mais neste documento os modelos de segurança existentes na literatura por acreditar que os mesmos não são relevantes para o presente trabalho.

### 2.2.4 Principais e Intrusos

Através das regras definidas pela política de segurança são determinadas as entidades autorizadas e responsáveis pelas ações executadas sobre informações mantidas no sistema, estas entidades são identificadas como *principal* ou sujeito. Um principal pode ser tanto um usuário, um processo ou ainda uma máquina em uma rede de computadores. Ao contrário, a entidade não autorizada pela política de segurança que obtém acesso aos recursos de um sistema computacional é denominado de *intruso* ou sujeito não autorizado.

### 2.2.5 Violações de segurança

Em sistemas computacionais ao ato ou evento que de alguma forma burle a política de segurança dá-se o nome de *violação de segurança* (SHIRLEY, 2007). A Tabela 2.1 ilustra os tipos de violação e as propriedades de segurança não verificadas. Note que a não verificação de cada uma destas propriedades dá origem a uma das violações de segurança possíveis em um sistema. Isto é, a não verificação da propriedade de confidencialidade dá origem a violação *revelação não autorizada de informação*; quando a propriedade não verificada é a de integridade temos então a violação *modificação não autorizada*; Se informações ou serviços de um sistema se tornam não acessíveis (indisponíveis) para usuários autorizados (principais) ocorre o que é identificado na literatura como à violação de *negação de serviço*.

Tabela 2.1 – Relação de violações e as propriedades de segurança

| **Tipo de violação** | | **Propriedade de segurança violada** |
|---|---|---|
| 1 | revelação não autorizada | confidencialidade |
| 2 | modificação não autorizada | integridade |
| 3 | negação de serviço | disponibilidade |

Fonte: SANTIN (2004).

### 2.2.6 Vulnerabilidades, Ameaças e Ataques

Antes de expor os principais ataques em ambiente distribuído, algumas definições devem ser introduzidas. Entende-se por *vulnerabilidade* (*vulnerability*) fraquezas ou imperfeições em procedimentos de operação, implementações de serviços e sistemas. As vulnerabilidades ligadas a implementações são oriundas de falhas de concepção ou de configuração de sistemas e serviços. Estas expõem os recursos de um sistema computacional a ameaças e ataques. Uma *ameaça* (*threat*) é caracterizada por um conjunto de ações possíveis que podem explorar vulnerabilidades (circunstâncias, condições, conhecimento sobre o sistema), colocando desta forma em risco as propriedades de segurança do sistema (LANDWEHR, 1981). Uma ameaça, quando posta em ação, é identificada como um *ataque* (*attack*) à segurança do sistema. Um ataque bem sucedido deve provocar algum tipo de dano no sistema computacional.

### 2.2.7 Vulnerabilidades, Ameaças e Ataques

## 2.3 ATAQUES EM SISTEMAS DISTRIBUÍDOS

Em (STALLINGS, 2000), os ataques à segurança de um sistema computacional através de uma rede (subsistema de comunicação) são generalizados através de informações (na forma de mensagens) que fluem de uma fonte, como um arquivo ou uma região da memória principal, para um destino através desta rede de comunicação. Um fluxo normal é exibido na Figura 2.1(a) e as ilustrações restantes categorizam em quatro os ataques de segurança possíveis:

- *Interrupção* (Figura 2.1(b)): Um nó ativo do sistema fica indisponível ou incomunicável. Este é um ataque sobre a disponibilidade de recursos ou informações que visam essencialmente à negação de serviço.

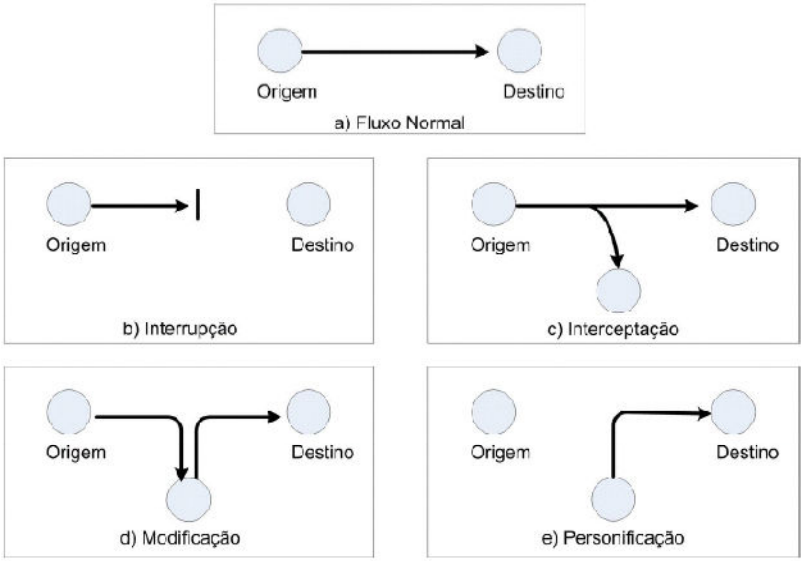


Figura 2.1 – Ameaças à Segurança
Fonte: STALLINGS (2000).

- *Interceptação* (Figura 2.1(c)): Um intruso (entidade não autorizada) tem acesso, via suporte de comunicação, a informações de uma comunicação em que não está devidamente autorizado a participar. Este é um ataque que visa violação de confidencialidade. Um exemplo deste tipo de ataque é o *sniffer*, que envolve a captura dados trafegando pela rede. É um ataque tido como passivo e, portanto, de difícil detecção.

- *Modificação* (Figura 2.1(d)): Um intruso agindo como agente intermediário em uma comunicação, não somente captura a informação (mensagem), como tenta alterar o seu conteúdo. Este é um ataque contra a propriedade de integridade do sistema. Um nó intermediário em redes *Overlay* pode produzir facilmente este tipo de ataque, modificando o conteúdo das mensagens sendo transmitidas na rede.

- *Personificação* (Figura 2.1(e)): A entidade não autorizada tenta se fazer passar por um principal do sistema. Este é um ataque à autenticidade do sistema. Como exemplo, a adição de mensagens espúrias na rede ou o ataque por mensagens antigas (*message-replay attack*) que visam forçar ao destinatário aceitar

o atacante é um das possíveis entidades autorizadas do sistema. O ataque conhecido como *man-in-the-middle* corresponde ao que é considerado aqui como Personificação.

Em (KENT, 1977) é apresentada outra classificação que é bastante usada, onde os ataques em um sistema distribuído são divididos em passivos e ativos:

### 2.2.7 Ataques passivos

Os ataques passivos são fundamentalmente baseados em escutas, monitoramento de transmissões, com o objetivo de obter informações transmitidas no sistema. Este tipo de ataque é de difícil detecção, pois não envolve qualquer alteração nos dados, então, é mais importante a prevenção do que a detecção. Em (KENT, 1977), é introduzida a subdivisão em dois tipos de ataques passivos:

- *Visualização do conteúdo da mensagem*: uma mensagem pode conter informações sensíveis ou confidenciais. A necessidade é prevenir que o atacante possa capturar o conteúdo dessas transmissões.
- *Análise de tráfego*: Mesmo quando a criptografia é usada para proteger o *payload* da mensagem, as informações de cabeçalho são deixadas em claro para permitir o roteamento. Estas informações podem ser usadas para inferir situações entre os pares comunicantes ou podem ser usadas para ataques subsequentes como a negação de serviço, visando prejudicar os pares comunicantes. É um ataque de difícil solução.

### 2.2.7 Ataques ativos

Este tipo de ataque envolve formas de modificações ou de criação de fluxos de dados no suporte de comunicação. Muitos dos ataques definidos anteriormente podem ser agrupados como ataques ativos:

- *Personificação* (*masquerade*): o atacante tenta se fazer passar por um usuário válido do sistema e assumir com isto os seus privilégios.

- *Repetição de mensagens* (*replay*): consiste na captura de mensagens (ataque passivo) e a posterior retransmissão das mesmas. O atacante espera com estas ações obter vantagens no sistema. Por exemplo, um atacante pode esperar com a *message reply* abrir uma sessão em um serviço de banco e, posteriormente, transmitir instruções de transferência de fundos para uma conta bancária sob o seu controle.
- *Modificação de mensagens* (*message modification*): o atacante modifica parte de uma mensagem legítima do sistema para produzir um efeito de autorização ou acesso.
- *Ataque de negação de serviço* (*denial of service*): o objetivo é tornar indisponível ou provocar o mau funcionamento de um determinado serviço. Por exemplo, um atacante pode inundar determinado serviço com solicitações, fazendo com que o serviço não consiga atender a solicitações legítimas do sistema.

Como dito antes, os ataques passivos são de difícil detecção, mas mesmo assim, existem meios disponíveis de prevenção (uso da criptografia, por exemplo). Entretanto, para se prevenir totalmente contra ataques ativos, seria necessária a proteção física de todo o suporte de comunicação, todo o tempo, o que é inviável. Mas, ao contrário dos passivos, a detecção é sempre possível para ataques ativos.

Muitos tipos de ataques/ameaças são identificados na literatura, mas o sucesso dos mesmos, na maioria das vezes, está condicionado à existência de vulnerabilidades de segurança em sistemas. O CERT (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil) fornece várias informações em seu site sobre vulnerabilidades e ataques correspondentes, visando prevenir e detectar tais incidentes.

## 2.4 MECANISMOS DE SEGURANÇA

Os mecanismos de segurança são responsáveis pela implementação nos sistemas das políticas de segurança, muitas vezes expressas por modelos de segurança. Para viabilizar a implantação de tais políticas, estes mecanismos são construídos a partir de controles de acesso e controles criptográficos.

### 2.4.1 Controles de Acesso

No que se referem a controle de acesso, estes mecanismos tomam o nome de monitor de referências (ANDERSON, 1972), intervindo nos acessos em um sistema. As referências a segmentos de memória são validadas nas camadas inferiores do sistema. Neste caso, o hardware pode desempenhar este papel de monitor de referências, permitindo que segmentos sejam acessados se processos tiverem os descritores de segmentos que validem os acessos desejados. No sistema operacional, por sua vez, um serviço de arquivos valida o acesso a arquivos se as permissões correspondentes de um principal (usuário) estiverem presentes numa Lista de Controle de Acesso (ACL). Ou seja, a noção de Monitor de Referências se faz presente nos vários níveis de um sistema computacional.

Como responsáveis pela intermediação de todas as requisições de acesso a objetos de um sistema, os monitores de referencia devem manter algumas propriedades:

- Ser invioláveis;
- Ser incontornáveis (em qualquer acesso, sempre sejam invocados);
- Ser pequenos o suficiente para que a verificação de suas correções sempre seja possível.

A noção de núcleo de segurança (*kernel security*) foi definida em (LANDWEHR, 1983) como o conjunto de recursos de hardware e software que permite a concretização de um monitor de referências. A implementação do monitor de referência é feita por mecanismos de controle, que podem ser discricionários (DAC – *Discretionary Access Control*), obrigatórios (MAC – *Mandatory Access Control*) ou baseados em papéis (RBAC - *Role-Based Access Control*), seguindo os modelos de controle de acesso que devem implementar.

### 2.4.1 Controles Criptográficos

Os controles criptográficos fornecem a base para a maioria dos mecanismos de segurança (STALLINGS, 2000). Cifragem de informações, autenticação de mensagens e usuários, certificação de chaves, técnicas de não repudiação são sustentadas pelo uso de controles criptográficos.

O uso de criptografia, basicamente pode ser dividido em duas partes: cifragem (*encryption*), processo em que a informação é codificada para ocultar seu significado e decifragem (*decryption*), aplicada a informação codificada para revelar o seu conteúdo. Atualmente, existem muitos algoritmos para a implementação dos chamados controles criptográficos. Em sua maioria, estes algoritmos possuem suas características de segurança baseadas no segredo de informações chamadas de chaves. Estas chaves como parâmetros de entrada nos procedimentos de encriptação (ou decriptação) definem transformações (ou anti-transformações) adequadas nos dados em claro (ou cifrados) produzindo os efeitos desejados de proteção (ou recuperação) dos mesmos.

Esses algoritmos são divididos em duas classes: simétricos e assimétricos (ou algoritmos de chave pública). Algoritmos simétricos usam uma mesma chave (chave secreta) para cifragem e decifragem dos dados. Já os algoritmos ditos assimétricos, usam duas chaves: uma chave pública para cifragem e uma chave privada para decifragem. Neste caso, as chaves públicas podem ser distribuídas livremente no sistema, enquanto que as chaves privadas devem permanecer sob sigilo com o seu proprietário. Existem diferenças de desempenho entre estes dois criptosistemas. Em relação ao tempo de processamento, os algoritmos assimétricos são de 100 a 1000 vezes mais lentos que os simétricos (COULOURIS *et al.*, 2005), porém apresentam propriedades de autenticação (assinaturas digitais) que são muito úteis em sistemas distribuídos. São usadas estas propriedades, por exemplo, na própria distribuição de chaves simétricas.

Como citado acima, através de controles criptográficos é possível garantir as propriedades de integridade, confidencialidade e autenticidade. Com o uso da cifragem a confidencialidade de informações é mantida. Para a autenticidade, são usadas técnicas de assinatura digital onde a aplicação de algoritmos específicos gera propriedades que permitem de maneira inconfundível se verificar a autenticidade de uma mensagem e identificar o seu emissor. A assinatura pode ser também base para se garantir a integridade de informações. Os dois tipos de criptosistemas possuem técnicas que podem ser usadas na autenticidade e que podemos considerar como assinaturas digitais.

Nos criptosistemas assimétricos, a distribuição indiscriminada de chaves públicas sem as devidas precauções pode levar a problemas relacionados à origem das mesmas. Isto porque, qualquer participante malicioso pode enviar sua chave pública para outros participantes se

anunciando com identidades falsas. Com isto, poderiam se fazer passar por principais de um sistema e receberem mensagens que seriam dirigidas a estes usuários autênticos do sistema. Uma das soluções para este problema é o uso de uma entidade confiável a todos os participantes do sistema, normalmente chamada de Autoridade Certificadora (AC). As relações de confiança com esta AC permitem que as assinaturas desta TTP (terceira parte confiável) sobre todas as chaves públicas de participantes no sistema, portanto, criando os certificados digitais, sejam aceitos no sistema como válidas e sem os problemas citados de falsas personificações.

Ao uso de ACs, em conjunto a outros mecanismos para estabelecer um sistema de certificação digital baseado em chave pública, dá-se o nome de Infraestrutura de Chave Pública (ICP). Tipicamente, nestas infraestruturas, existem dois tipos de certificados, um que associa uma chave pública a um nome (certificado de nomes) e outro que liga atributos à chave pública (certificado de autorização). Atualmente vários certificados estão em uso, alguns patenteados como o *Pretty Good Privacy* (PGP) (ZIMMERMAN, 1994), enquanto outros certificados populares são específicos de aplicativos, como os certificados usados no SET (*Security Electronic Transaction*) (Secure Electronic Transaction LLC, 1997) e no *Internet Protocol Security* (IPSec) (MARKHAM, 1997). O certificado mais amplamente aceito é o que segue o padrão X.509 (HOUSLEY *et al*., 2002). Há ainda propostas, como o SPKI/SDSI (ELLISON, 1999) que visam retirar a complexidade imposta pelo padrão X.509.

### 2.4.1 Controles Adicionais de Segurança

Além do controles já mencionados, outros mecanismos de segurança envolvem ações, técnicas, procedimentos ou dispositivos que têm como propósito implantar políticas de segurança. Estes controles (internos), que não atuam diretamente nas requisições de acesso, podem também estar presentes nos sistemas (AMOROSO, 1994):

A **auditoria de vestígio** está associada à geração periódica de registros de eventos relativos à segurança. Estes dados são coletados para uso potencial em detecção de intrusão e/ou auditoria de segurança.

A **auditoria de segurança** é uma inspeção independente, realizada por terceiros, nos procedimentos e registros do sistema com intuito de verificar adequação da política de segurança e as possíveis violações do sistema.

A **detecção de intrusão**, por sua vez, usa os registros das auditorias em métodos automatizados de análise em tempo real, o que envolve muitas vezes uma sequência de eventos relacionados ou não, com o intuito de identificar atividades anormais no sistema.

Para as situações cujas violações não puderem ser evitadas, mecanismos que façam uso de técnicas de backups, replicações e que permitam recuperar o sistema completam os controles adicionais.

Outros controles necessitam ser adicionados aos já comentados. É necessário, por exemplo, o conhecimento por cada usuário de suas responsabilidades e atribuições. Com isto, cada usuário saberá o que está autorizado a fazer. É importante cada usuário estar ciente e comprometido com a segurança no ambiente computacional, pois sem isto as demais medidas podem se tornar ineficazes.

## 2.5 SEGURANÇA EM SISTEMAS DISTRIBUÍDOS

Ao tratar de segurança em sistemas distribuídos a preocupação não se resume mais a uma máquina ou recursos geridos de forma centralizada. Nestes sistemas, formados por redes de computadores, é comum os nós estarem distantes uns dos outros.

Além disto, sistemas distribuídos são ambientes que apresentam uma dinâmica em suas composições, possuindo sempre números crescentes de recursos, o que se reflete em problemas de escalabilidade.

Para vencer os problemas de escalabilidade nestes ambientes, é comum reunir conjuntos de recursos computacionais destinados a um mesmo fim em domínios de segurança (que podemos entender como domínios administrativos). Cada domínio pode apresentar políticas de autenticação e de autorização específicas garantindo as propriedades de segurança dos recursos no domínio (SANTIN, 2004).

### 2.4.1 Autenticação e Autorização em sistemas distribuídos

O processo de autenticação consiste na identificação de um principal num sistema. Neste processo a individualização do usuário ou entidade é dada a partir de uma prova de identificação, que pode ser a posse de alguma informação, o conhecimento de algum segredo ou ainda, uma característica única (íris, impressão digital etc.). Cada principal recebe um identificador único que é válido em todo o sistema.

Em um sistema distribuído é comum um principal precisar acessar recursos em diferentes locais, por isso, realizar a autenticação a cada acesso a recurso torna o processo oneroso, além da necessidade de transporte da prova de identificação pela rede (com o uso de controles criptográficos). Para resolver tais problemas é necessária uma relação de confiança com uma terceira entidade (*trusted third party*) – o serviço de autenticação. Assim, para fins de identificação, este serviço é o fornecedor de informações necessárias na autenticação de dados sobre o principal. Estas informações podem ser constituídas pelo que normalmente é chamado de certificado, representando uma prova irrefutável de autenticidade, aceita pelos participantes da comunicação. O emissor e destinatário devem assumir a credibilidade do serviço de autenticação que gerou essas informações de certificação.

O serviço de autenticação de um domínio pode manter relações de confiança com homônimos de outros domínios. Neste caso, a autenticação em um domínio pode também ser válida em outro domínio que faz parte das relações de confiança do primeiro. Com isto um principal, já autenticado em um domínio e de posse de atributos de autenticação pode acessar servidores em outros domínios, caracterizando o que é chamado de identificação única (*single sign-on*). Estas relações entre domínios permitem a construção de redes de confiança que são muito úteis para prover escalabilidade nas autenticações em redes de larga escala como a Internet.

O uso de certificados evita a necessidade de mecanismos específicos para proteger informações de autenticação em trânsito pela rede. Se adicionarmos como suporte uma ICP com aceitação internacional como a X.509 e admitirmos relações de confiança entre ACs (mesmo que na forma de uma hierarquia), teremos possibilidade de uma autenticação possível em vários pontos de uma WAN (*Wide Area Network*), envolvendo vários domínios. O uso de certificados viabiliza a comprovação da autenticidade de um usuário, ligando o mesmo a sua posição na WAN.

Em sistemas distribuídos são permitidas operações remotas que envolvem trocas de mensagens cuja autenticação não se limita à identificação de principais. É necessária também a autenticação de mensagens em trânsito. Neste caso, se deseja a propriedade de autenticidade das mensagens, através da identificação incontestável da origem pelo destino das mesmas.

Devido ao fato dos usuais serviços de autenticação, como o Kerberos (chave secreta) (NEUMAN e TS'o, 1994) ou o X.509 (chave pública) (HOUSLEY *et al.*, 2002), usarem mecanismos baseados em

assinatura digital, pode-se pressupor que em sistemas distribuídos a autenticação de mensagens é suportada por mecanismos que comprovem a posse de uma chave criptográfica. O uso de certificados X.509, por exemplo, permite que o destinatário, de posse do certificado correspondente, possa verificar uma assinatura na mensagem recebida.

A autorização, ou controle de acesso, é um processo cujo objetivo é garantir que só tenham acesso aos recursos os principais que tenham legitimidade para fazê-lo. Este processo foi introduzido por Anderson (ANDERSON, 1972) e usa o conceito de monitor de referência para permitir ou negar acesso a um determinado recurso para um principal (ver seção 2.4.1).

A implementação dos mecanismos de autenticação e autorização em sistemas distribuídos é uma tarefa complexa, podendo envolver vários domínios administrativos e tecnologias distintas. Além disso, são afetados pela escalabilidade serviços como autenticação, autorização, serviços de nomes, entre outros.

Para contornar estes problemas de escalabilidade a literatura propõe diferentes abordagens para implementação de mecanismos de autenticação e autorização: abordagem centralizada, autenticação centralizada e autorização descentralizada e abordagem descentralizada.

Na abordagem centralizada, tanto o serviço de autenticação quanto o de autorização ficam numa mesma máquina. Sendo assim, traz a vantagem de possuir uma política de autorização única, mas as desvantagens de desempenho, pois esta máquina será responsável por toda autorização e autenticação de um sistema, e um ponto único de falha, apresentando alta probabilidade de ocorrência de ataques de negação de serviço. Esta concentração em somente uma máquina de importantes funções de segurança do sistema pode ser extremamente vulnerável em sistemas abertos.

Na abordagem de autenticação centralizada e autorização descentralizada, existe a caracterização de domínios de segurança. Cada domínio possui um serviço de autenticação centralizado e a autorização é controlada nos provedores de serviço, portanto é descentralizada. É necessário existir uma relação de confiança entre a entidade de autenticação e as entidades de autorização no domínio. Esta é a abordagem mais usada, muito embora se possa apontar como desvantagem a entidade de autenticação única.

Na abordagem descentralizada, várias máquinas estão distribuídas no sistema cada uma responsável pela implementação das políticas de autenticação e autorização de principais. Usualmente, a implementação desta abordagem se dá através das redes de confiança, onde existem

várias entidades confiáveis distribuídas no sistema, que se encarregam de implementar as políticas de autorização e autenticação. Embora essa abordagem retire o inconveniente da centralização de algumas atividades, o problema de gestão descentralizada normalmente compromete a coerência da política de autorização. O SPKI/SDSI (*Simple Public Key Infrastructure/Simple Distributed Security Infrastructure*) (ELLISON, 1998) e o TCSEC (*Trusted Computer System Evaluation Criteria*) (Department of Defense, 1985) são exemplos desta abordagem.

## 2.6 SEGURANÇA EM SISTEMAS DISTRIBUÍDOS

Este capítulo apresentou os conceitos e propriedades fundamentais na segurança de sistemas computacionais. As propriedades de segurança colocadas neste texto foram as de confidencialidade, integridade, disponibilidade e autenticidade. Uma vez garantida estas propriedades podemos assumir que um sistema é seguro. É evidente que não existe uma solução para se alcançar a segurança plena, mas sim, que existem lacunas de segurança que quando preenchidas de modo a manter as propriedades citadas podemos juntar qualidade às aplicações. Foram também discutidos os controles e mecanismos de segurança no contexto de sistemas distribuídos.

# 3 ARQUITETURA ORIENTADA A SERVIÇOS

Nos últimos dez anos, uma evolução da tecnologia vem abrindo possibilidades de interoperabilidade e integração de aplicações na Internet. Este caminho se iniciou com o advento dos Serviços *Web* e suas extensões, com o apoio de importantes organizações padronizadoras como W3C e OASIS.

A tecnologia de Serviços *Web* é a mais notória das concretizações da Arquitetura Orientada a Serviços (SOA – do inglês *Service-Oriented Architecture*). A SOA, por si só, é um conjunto de conceitos que enfatiza a programação orientada a serviços. Já os Serviços *Web* se caracterizam como uma tecnologia que permite a integração de uma ampla variedade de sistemas, plataformas e de programas proprietários, definindo formas de interoperabilidade própria para ambientes complexos como WANs (*Wide Area Networks*). O uso dos conceitos SOA é perfeitamente adequado para a interação de aplicações nas condições citadas.

Neste capítulo, introduzimos os principais conceitos enfatizados pelo paradigma SOA e na sequência exploramos algumas das mais relevantes especificações de Serviços *Web* que serão importantes para o trabalho desta dissertação.

## 3.1 ARQUITETURA ORIENTADA A SERVIÇO

A Arquitetura Orientada a Serviço (*Service Oriented Architecture* – SOA) pode ser entendida como uma descrição de sistemas distribuídos, na forma de serviços. Estes serviços têm suas funcionalidades expostas via descrição de uma interface, permitindo a publicação, localização e a invocação dos mesmos por meio de um formato padronizado (PAPAZOGLOU, 2003).

A SOA utiliza, portanto, o conceito de serviço como elemento fundamental na concepção de aplicações distribuídas. Portanto, é uma arquitetura que induz uma infraestrutura (de serviços) que permite que serviços de aplicações sejam invocados remotamente. Estas invocações podem envolver simplesmente a troca de dados ou ainda, a ativação de dois ou mais serviços integrados em alguma atividade específica (ZDUN *et al*., 2006).

A arquitetura define três tipos de participantes:

(1) Provedor de Serviço – entidade que cria, num formato padrão, o serviço e é responsável por publicar a interface do mesmo no registro de serviços e ainda atender as requisições originadas pelos clientes; nesta publicação, além de informações sobre o endereço do serviço, é definido também como se dão as trocas de mensagens com o serviço.

(2) Registro de Serviços – é o repositório utilizado para a publicação e localização de interfaces dos serviços; este participante tem o seu funcionamento semelhante ao das páginas amarelas das listas telefônicas, cujo objetivo é indexar os serviços disponíveis. É através deste repositório que os consumidores de serviços encontram os serviços desejados.

(3) Consumidor de Serviços – também conhecido por cliente, este participante recupera, através de consultas ao registro, as informações necessárias para acessar aos serviços. Portanto, é parte de uma aplicação (pode ser também um serviço) que envia requisições de serviços aos seus correspondentes provedores.

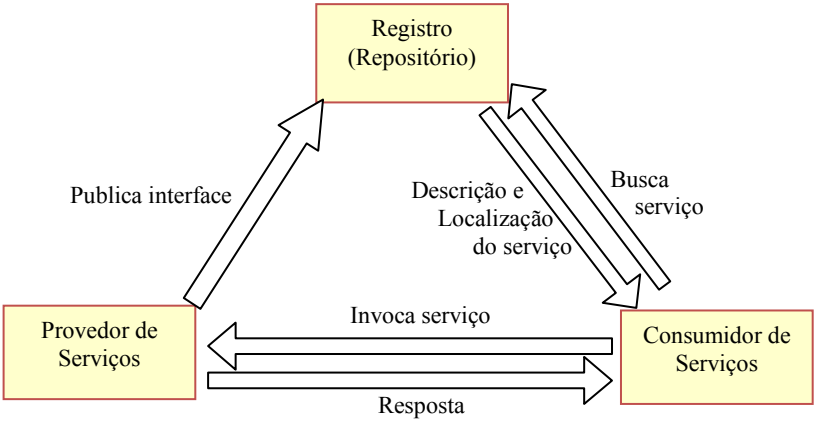


Figura 3.1 – Participantes da Arquitetura Orientada a Serviços

Cada participante da arquitetura pode ainda assumir um ou mais papéis; por exemplo, um participante pode ser provedor e ainda um cliente de outros serviços que usa para implementar o seu. A SOA

também descreve as interações entre esses papéis, através de operações como: publicar, localizar e invocar. A Figura 3.1 ilustra a colaboração entre os participantes da SOA, identificando os papéis e as operações executadas no acesso a um serviço.

No caso da Figura 3.1, fica explicitada a busca do cliente, através de suporte do registro, por informações de um serviço, especificando as características desejadas do mesmo. Quando o registro do serviço desejado é localizado, informações de interface e a localização deste serviço são retornadas ao cliente. Por fim, com o uso destas informações, o cliente faz a invocação ao serviço desejado. As interfaces de serviços são um contrato entre o provedor do serviço e o consumidor, sendo que este contrato estabelece: um conjunto de operações públicas do serviço, o protocolo que deve ser usado na comunicação com o serviço, os parâmetros e valores de retorno de uma requisição e meios para tratar possíveis exceções.

## 3.2. SERVIÇOS *WEB*

A tecnologia de Serviços *Web* é, portanto, a concretização do conceito de SOA. Proposta inicialmente pela Microsoft, adotado e regulamentado pela *The World Wide Web Consortium* (W3C), Serviços *Web* fornecem padrões abertos para a comunicação entre aplicações e serviços em diferentes plataformas.

### 3.2.1 Características de Serviços *Web*

Vários textos descrevem os Serviços *Web* e todos estes sustentam as seguintes características:

(1) Troca de mensagens: Serviços *Web* trocam mensagens através de um protocolo padrão de *request/reply* conhecido como SOAP. Na Internet, este protocolo usa os serviços de protocolos conhecidos como HTTP ou o SMTP, por exemplo, para o transporte de requisições e respostas SOAP;

(2) Descrição de serviços: Serviços *Web* fornecem uma descrição completa dos seus serviços e como os usuários podem criar aplicações para interagir com estes. Esta descrição é feita através da *Web Service Description Language* (WSDL).

(3) Auto-descoberta: os Serviços *Web* são registrados em serviços especiais para que os potenciais usuários possam achá-los. Este registro é feito na *Universal Discovery Description and Integration* (UDDI).

Através destas características indicadas, a tecnologia de Serviços *Web* torna possível as três operações fundamentais da SOA: publicar, localizar e invocar serviços.

A versão da Figura 3.1, considerando Serviços *Web*, é apresentada na Figura 3.2. Desta forma, então, um provedor de serviços (um Serviço *Web* propriamente dito) publica a descrição de seus serviços através de um arquivo WSDL em um repositório UDDI. Um consumidor, por sua vez, ao necessitar de um serviço, consulta o repositório. E quando um serviço compatível é encontrado, o WSDL deste último é retornado ao solicitante e, a partir daí, o consumidor pode interagir com o provedor de serviços através de requisições e respostas SOAP.

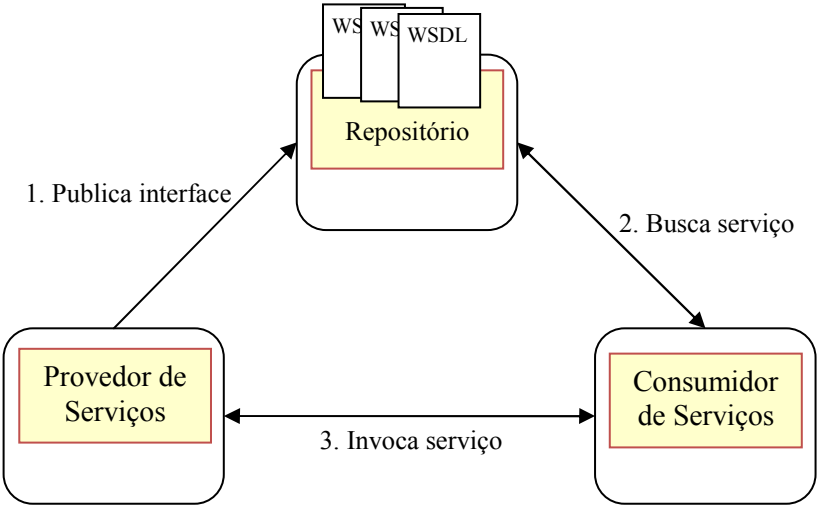


Figura 3.2 – Arquitetura de Serviços *Web*

Para a interoperabilidade almejada, os protocolos e tecnologias usadas em Serviços *Web* são baseados na *Extensible Markup Language* (XML), que por se tratarem de arquivos texto podem ser interpretados e criados por sistemas computacionais completamente independentes de

plataforma de software e hardware. A linguagem XML oferece um formato de dados flexível e extensível. A sua sintaxe permite a definição de um número ilimitado de nomeadores (chamados *tags*). Com o uso de esquemas (*schemas*) é possível a definição formal de novos documentos. Através destes esquemas, os documentos podem ser automaticamente validados, permitindo o intercâmbio de dados entre aplicações heterogêneas.

### 3.2.2 SOAP

SOAP é um protocolo de comunicação usado para permitir a troca de mensagens entre Serviços *Web* que foi concebido para ser simples e extensível. Em síntese, SOAP é uma especificação que define um formato XML para mensagens de *request* e *reply*. As especificações do SOAP citam várias possibilidades para transporte de suas mensagens. A mais notória entre estas possibilidades é o *Hypertext Transfer Protocol* (HTTP). O motivo para esta escolha é a grande aceitação deste protocolo e a existência de ferramentas para criar e analisar as mensagens SOAP sobre o HTTP.

A estrutura da mensagem SOAP envolve três elementos principais (MITRA, 2007):

(1) SOAP Envelope: toda mensagem SOAP deve contê-lo. É o elemento raiz do documento XML. O Envelope pode conter atributos adicionais como o que define o estilo de codificação (*enconding style*). Um *enconding style* define como os dados são representados no documento XML;

(2) SOAP Header: O cabeçalho é um elemento opcional que contém informações específicas da mensagem SOAP, tais como autenticação, identificação de transação etc. O Header é usado para roteamento entre serviços. Quando usado, o SOAP Header deve ser o primeiro elemento do SOAP Envelope;

(3) SOAP Body: O corpo é um elemento obrigatório de uma mensagem SOAP e contém a informação propriamente dita da mensagem. Aninhado a este elemento poderá existir, ainda, o elemento *Fault*, com informações de falha ocorridas na aplicação.

A Figura 3.3 ilustra a ordem e o aninhamento dos elementos que compõem uma mensagem SOAP.

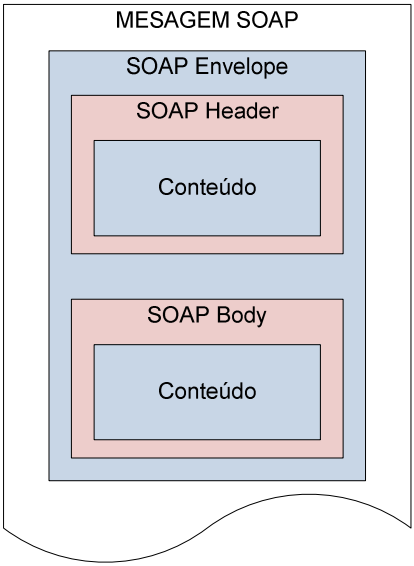


Figura 3.3 – Estrutura de uma mensagem SOAP

Os Códigos 3.1 e 3.2, a seguir, apresentam um exemplo da composição de arquivos XML nas correspondentes mensagens SOAP de *request* e *reply*. O exemplo de mensagem SOAP exibida no Código 3.1 se refere a uma solicitação de último preço negociado (*GetLastPrice*, linha 5). Nesta mensagem (linha 6), é informado o símbolo de referência para a solicitação. Note que o corpo da mensagem está aninhado ao envelope *Body* (linhas 4 e 8), assim como indicado na Figura 3.3. Esta mensagem não possui o elemento Header, que é opcional. No Código 3.2 é exibida uma mensagem de resposta SOAP ao Código 3.1. Nela é retornado, na linha 6, o preço do item correspondente ao símbolo indicado na mensagem de requisição.

O protocolo SOAP, portanto, fornece suporte para trocas de mensagens entre Serviços *Web*. Entretanto não define quais elementos deverão ser usados no corpo da mensagem. Este, entre outros, é o papel da especificação abordada a seguir, a WDSL.

Código 3.1 – Exemplo de mensagem SOAP de requisição

```
1  <SOAP-ENV:Envelope
2     xmlns:SOAP- ENV="http://schemas.xmlsoap.org/soap/envelope/"
3     SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
4     <SOAP-ENV:Body>
5        <m:GetLastTradePrice xmlns:m="Some-URI">
6           <symbol>DIS</symbol>
7        </m:GetLastTradePrice>
8     </SOAP-ENV:Body>
9  </SOAP-ENV:Envelope>
```

Código 3.2 – Exemplo de mensagem SOAP de Resposta

```
1  <SOAP-ENV:Envelope
2     xmlns:SOAP- ENV="http://schemas.xmlsoap.org/soap/envelope/"
3     SOAP-ENV:encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/"/>
4     <SOAP-ENV:Body>
5        <m:GetLastTradePriceResponse xmlns:m="Some-URI">
6           <Price>34.5</Price>
7        </m:GetLastTradePriceResponse>
8     </SOAP-ENV:Body>
9  </SOAP-ENV:Envelope>
```

### 3.2.3 WSDL

A *Web Service Description Language* (WSDL) (CHINNICI, 2007), como o nome sugere, é a linguagem de descrição de Serviços *Web*. Sua missão é proporcionar ao usuário um conjunto de informações que permitam ao mesmo construir ou adequar suas aplicações clientes para comunicações com o serviço correspondente à interface WSDL. Uma especificação WSDL é um arquivo XML descrevendo o conjunto de mensagens SOAP que um cliente pode trocar e como estas devem ser trocadas, com o Serviço *Web* correspondente. Como um arquivo XML, uma especificação WSDL pode ser lida e editada, mas comumente é gerada e interpretada por programas específicos.

Uma WSDL é uma estrutura bem definida, composta pelos seguintes itens:

(1) *types*: definem, baseados em especificações como XMLSchema, os tipos de dados usados no Serviço *Web*;

(2) *message*: define as mensagens SOAP de pedido ou resposta a serem trocadas pelo Serviço *Web*;

(3) *portType*: é elemento mais importante, pois define as operações (métodos) disponibilizadas pelo Serviço *Web* e as mensagens envolvidas;
(4) *binding*: define o protocolo usado no transporte das mensagens SOAP;
(5) *service*: define o nome e o endereço do serviço disponibilizado, através de um conjunto de elementos port.

A Figura 3.4 ilustra a estrutura de um documento WSDL, mostrando os principais elementos citados. Como pode ser observado, o elemento *definitions* representa o elemento raiz em que estão todos os outros elementos.

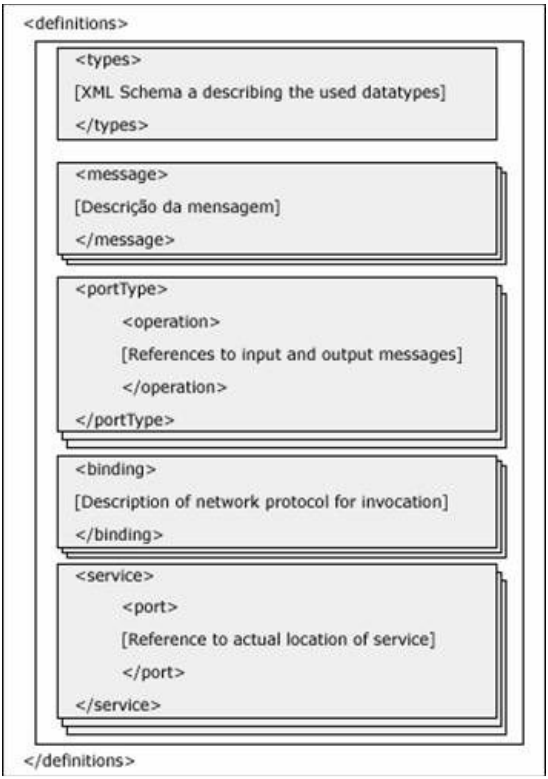


Figura 3.4 – Estrutura de um documento WSDL
Fonte: RECKZIEGEL (2006)

### 3.2.4 UDDI

Um Serviço *Web* pode ser consumido bastando serem conhecidas suas características e funcionalidades através de seu documento WSDL, desde que se saiba onde este documento esteja hospedado. Desta forma, os Serviços *Web* dependem de um repositório conhecido como UDDI para que os provedores de serviços possam publicar seu documento WSDL, possibilitando que eventuais consumidores consultem o WSDL para utilizar os serviços disponíveis.

Segundo (CLEMENT *et al.*, 2004), o foco do UDDI é a definição de um conjunto de serviços de apoio à descrição e descoberta de empresas, organizações e outros prestadores de Serviços *Web*, para disponibilizar esses Serviços *Web* e suas interfaces para acessos de consumidores. Baseado em um conjunto comum de padrões, incluindo HTTP, XML, XML Schema e SOAP, o UDDI provê suporte para a publicação e descoberta de serviços disponíveis sem restrições e também para os expostos apenas internamente dentro de uma organização.

Alguns autores fazem analogia do UDDI com uma lista telefônica (CERAMI, 2002; RECKZIEGEL, 2006; FIATKOOSKI, 2004):

(1) Páginas Brancas: contêm informações sobre nomes, endereços, números de telefone, além de outras informações sobre os fornecedores do serviço.
(2) Páginas Amarelas: contêm listagens comerciais baseadas nos tipos desses negócios, de maneira organizada por categoria específica ou regiões demográficas.
(3) Páginas Verdes: são usadas para indicar os serviços oferecidos por cada negócio, incluindo todas as informações técnicas envolvidas na interação com o serviço. Resumindo, explica como fazer a comunicação com estes.

Para melhor visualizar a estratificação de Serviços *Web*, alguns autores (CERAMI, 2002; KILGORE, 2002; FIATKOOSKI, 2004) sobrepõem as ferramentas e protocolos apresentadas até aqui em camadas, como mostrado na Figura 3.5.

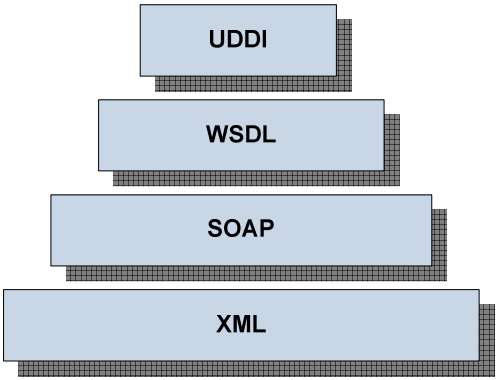


Figura 3.5 – Principais camadas que compõem Serviços *Web*

A Figura 3.5 mostra apenas as principais camadas que compõem Serviços *Web*. Entretanto, Serviços *Web* foram concebidos para serem extensíveis. Por isso, podem ser adicionadas outras camadas que incrementam mais funcionalidades, visando melhorar certas características, tais como: segurança, qualidade de serviços, políticas de uso, entre outras.

## 2.5 SEGURANÇA EM SERVIÇOS *WEB*

Como visto no capítulo de segurança, alguns autores apontam confidencialidade, integridade, disponibilidade, autenticidade e o não repúdio como as propriedades fundamentais para a garantia da segurança. O objetivo desta seção é apresentar os padrões de segurança propostos para Serviços *Web*. A tecnologia de Serviços *Web* deve incorporar técnicas e padrões no sentido de que suas aplicações possam experimentar as propriedades citadas.

As comunicações entre Serviços *Web* são feitas em nível de aplicação e podem envolver roteamento em verdadeiras redes Overlay neste nível (de aplicação). Por exemplo, um Serviço *Web* que monitore a qualidade de energia elétrica poderá enviar uma notificação ao perceber que uma medição realizada esteja fora dos padrões aceitáveis. Esta notificação poderá passar por diversos nós intermediários antes de chegar ao destino final. Ou seja, esta notificação pode ser roteada entre diversos serviços, antes de chegar ao consumidor desta notificação. O

problema está em como garantir que as informações cheguem ao destino final de forma segura, visto que as informações sensíveis são roteadas entre nós intermediários, em nível de aplicação.

O roteamento entre múltiplos Serviços *Web* é comumente utilizado para obter escalabilidade e também para agir como uma ponte entre diferentes pilhas de protocolos. Um exemplo dessa ponte é quando a comunicação entre dois nós utiliza um serviço intermediário. E as mensagens são transmitidas do remetente ao intermediário utilizando o protocolo HTTP e, na sequência, para transmitir a mensagem ao destinatário é usado o protocolo SMTP. Tecnologias como o TLS/SSL (DIERKS e ALLEN, 1999; FREIER *et al*., 1996) permitem garantir a confidencialidade em nível de transporte, porém não proporcionam segurança em nível de aplicação.

A Figura 3.6 exibe dois diferentes contextos de segurança, onde o primeiro sinaliza a segurança em nível de transporte e o segundo mostra segurança fim a fim. A lacuna de segurança mencionada deve ser preenchida por uma codificação criptográfica no nível dos protocolos de aplicação.

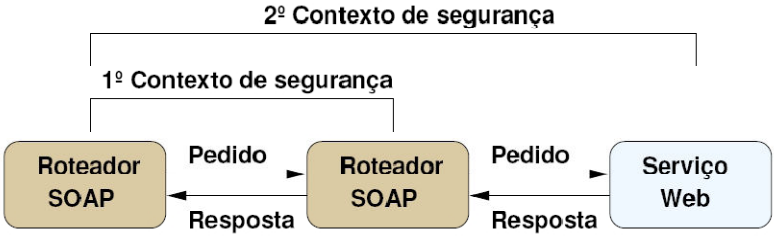


Figura 3.6 – Contextos de segurança
Fonte: (de MELLO *et al,*, 2006)

Também é importante citar que o uso de Serviços *Web* sobre o HTTP, traz o advento da transposição de firewalls, o que por outro lado abre uma lacuna de segurança, uma vez que, perde-se o filtro feito por estes. Na maioria dos casos, o protocolo HTTP e sua porta 80 são liberados em firewalls.

Os Serviços *Web* também estão suscetíveis a tipos de ataques como negação de serviço, mensagens antigas, estouro de pilhas, entre outros, conforme apresentado nos trabalhos (WESTBRIDGE, 2003; DEMCHENKO *et al*., 2005). A solução para esse novo ambiente, como ilustra a Figura 3.7, pode ser alcançada integrando mecanismos de

segurança (notados como ferramentas de segurança na Figura 3.7) nas diferentes camadas da pilha que lidam com protocolos de aplicação. (de MELLO *et al*., 2006)

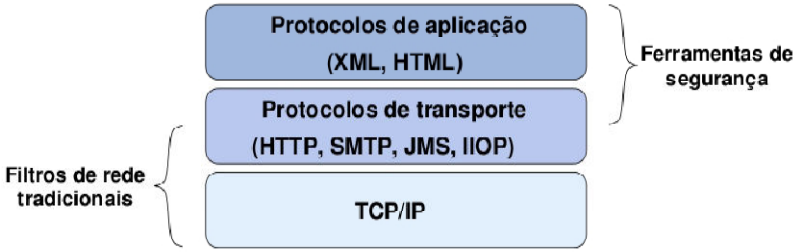


Figura 3.7 – Segurança nas diferentes camadas
Fonte: (de MELLO *et al.*, 2006)

### 3.3.1 Especificações de Segurança para Serviços *Web*

Várias organizações começaram a propor padrões de segurança para Serviços *Web*. A *World Wide Web Consortium* (W3C)[7], *Organization for the Advancement of Structured Information Standards* (OASIS)[8] e *Web Services Interoperability Organization* (WS-I)[9], lançaram propostas de segurança em Serviços *Web* no intuito de garantir sua ampla adoção. Estas visam cobrir as propriedades de segurança anteriormente mencionadas e serão discutidas na sequência.

#### 3.3.1.1 XML Signature

Assinaturas digitais são usadas como meio de garantir a autenticidade e integridade de informações digitais. A especificação XML Signature (XMLDSign) (Bartel *et al.*, 2008), define as regras para gerar e validar assinaturas digitais expressas em XML. O desafio em criar assinaturas em XML está na forma de codificação destes documentos. Isto porque, para interpretadores XML, o elemento `<Name >` e o elemento `<Name>` são tratados da mesma forma. Porém, quando aplicados a um algoritmo para assinatura digital, duas assinaturas

[7] www.w3.org
[8] www.oasis-open.org
[9] www.ws-i.org

distintas seriam geradas, pois no primeiro dos elementos há um espaço depois da string Name. (de MELLO *et al*., 2006)

Como forma de uniformizar a representação de um documento XML, foi introduzido o formato XML Canonical (BOYER, 2001) que possui uma forma dita canônica de representação. Documentos XML, que sejam sintaticamente diferentes, porém logicamente equivalentes, serão representados por uma mesma forma canônica. Assim, o uso da forma canônica possibilita que os documentos XML possam ser assinados sem que haja preocupação com a sintaxe dos mesmos.

Além de arquivos XML, com a XMLDSign é possível assinar outros tipos de documentos (arquivos binários ou textos).

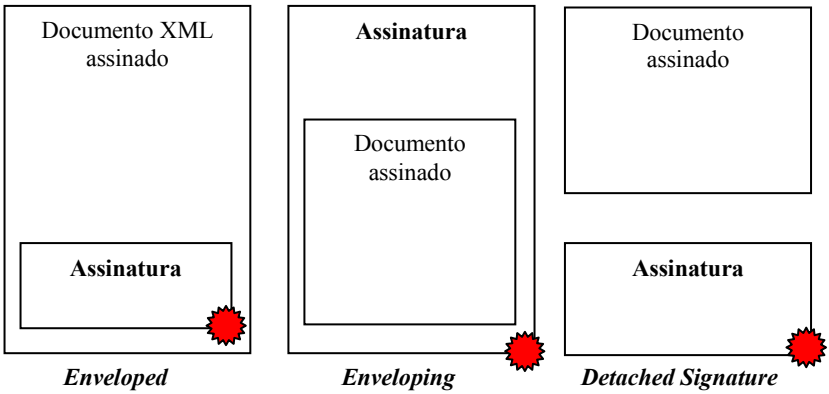


Figura 3.8 – Formas de assinaturas da XMLSign
Fonte: (de MELLO *et al.*, 2006)

Em arquivos XML é possível assinar com XMLDSign somente partes destes ou ainda, ter a parte assinada externa ao documento. Estes tipos de assinaturas são distinguidos conforme a Figura 3.8 em:

(1) *Enveloped*: a assinatura fica contida dentro do próprio documento XML a qual esta referencia. É ideal para ser utilizada, inserida em mensagens SOAP;
(2) *Enveloping*: os dados assinados, em XML ou não, ficam contidos dentro da própria estrutura do XMLDSign e
(3) *DetachedSignature*: quando os dados em XML ficam deparados de sua assinatura. Esta última é ideal para assinar documentos que não estão disponíveis localmente ou sofrem constantes modificações.

É importante salientar que esta especificação não define novos algoritmos, mas faz uso dos algoritmos existentes, como o RSA (RSA, 2002) e SHA-1 (EASTLAKE e JONES, 2001), etc.

#### 3.3.1.2 XML Encryption

A especificação XML Encryption (XMLEnc) (IMAMURA *et al*., 2002) visa prover segurança fim a fim que, como explicado antes, não é possível se obter somente com o TLS/SSL, quando aplicações necessitem fazer roteamento de mensagens de forma segura.

A XMLEnc provê possibilidades como cifrar somente partes de arquivo de dados. Ou seja, os dados cifrados são representados de uma forma estruturada e permitem que em um mesmo documento estejam presentes informações cifradas e não cifradas. Tal estrutura possibilita, ainda, o uso de diferentes chaves para cifrar as diversas partes de um documento. Com isto, um mesmo documento pode ser enviado a diversas partes receptoras, sem que ocorra a revelação de informações para participantes não autorizadas. O acesso de cada parte do documento se dá por receptores que possuam a chave correspondente da parte considerada.

A estrutura do XMLEnc, além de expressar os dados cifrados, também descreve detalhes sobre o tipo do documento cifrado (jpeg, xml, etc.); informações sobre a chave que é usada na codificação; informações sobre o algoritmo e método de cifragem utilizado, etc.

#### 3.3.1.3 XACML

O controle de autorização (controle de acesso) é um dos controles básicos para a segurança de uma aplicação. Visando garantir a interoperabilidade entre os diversos sistemas, o órgão OASIS lançou a *eXtensible Access Control Markup Language* (XACML) (OASIS, 2005a), um sistema para descrição de políticas de autorização, baseado em XML.

A especificação XACML descreve uma linguagem para políticas de controle de acesso e também um formato para mensagens de pedido e resposta para as interações necessárias na verificação de políticas. A linguagem é usada na definição dos direitos de acesso que compõem as políticas de autorização (listas de acesso). O formato de pedido e resposta descreve como as consultas (pedidos) ao sistema de políticas devem ser realizadas e como deverão ser as respostas a estas consultas.

Cada requisição está associada a um contexto de segurança envolvendo algoritmos e chaves criptográficas, controles e direitos de acesso. Para selecionar os contextos adequados de segurança a especificação XACML introduz um modelo de verificação de políticas. Este modelo é formado pelos componentes *Policy Decision Point* (PDP), que toma a decisão depois de avaliar se a requisição possui os direitos necessários para o acesso; e o *Policy Enforcement Point* (PEP) que concretiza as decisões de política, permitindo ou não o acesso desejado. Assim, uma camada de abstração entre o ambiente da aplicação e a linguagem núcleo do XACML é feita através de um Contexto XACML, que é definido através de um esquema XML, descrevendo uma representação canônica das entradas e saídas do *Policy Decision Point* (PDP) (OASIS, 2005a). O PDP é o nome dado pela OASIS em suas especificações ao monitor de referências.

A Figura 3.9, apresentada em (de MELLO *et al*., 2006; de MELLO, 2009), ilustrada o fluxo de dados entre um cliente tentando acessar um recurso, utilizando-se do XACML. No passo 1, o sujeito (cliente) cria uma requisição de acesso que é interceptada pelo PEP, que monta um pedido XACML e encaminha ao Tratador de Contextos (passo 2), que por sua vez envia o pedido para o PDP para que este decida sobre a tentativa de acesso (passo 3). O PDP pode requisitar ao Tratador de Contextos os atributos relacionados ao recurso e ao sujeito (passos 4, 5 e 6). De posse dos atributos, o PDP requisita as políticas associadas com as entidades envolvidas (passo 7) e assim gera uma resposta sobre a decisão tomada (passo 8). As políticas requisitadas devem estar codificadas segundo o XACML e dispostas em um repositório de políticas (PAP). O Tratador de Contextos gera uma resposta XACML e envia ao PEP (passo 9). E por fim, o PEP garante ou não o acesso ao recurso, seguindo as políticas correspondentes (passo 10).

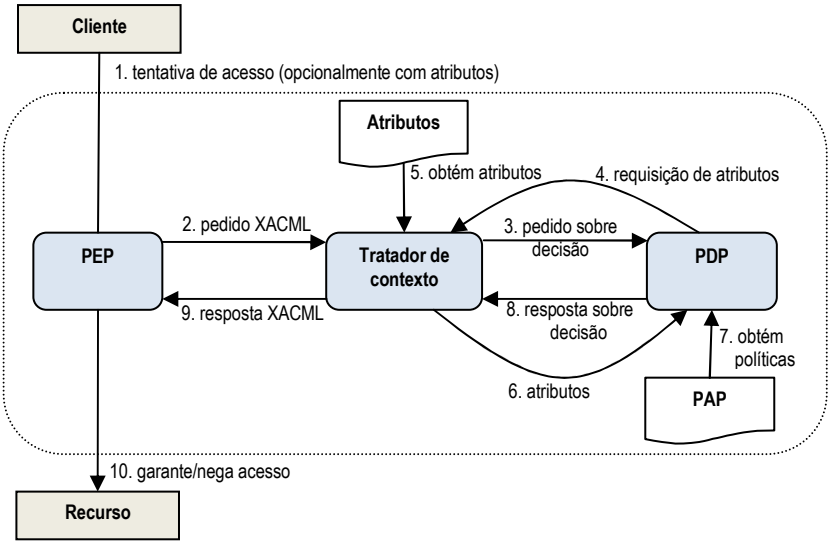


Figura 3.9 – Fluxo de dados com o XACML
Fonte: Adaptado de (de MELLO, 2009)

### 3.3.1.4 SAML

Como forma de padronizar a criação, troca e interpretação de asserções de segurança entre entidades em aplicações distribuídas, a OASIS definiu um conjunto de especificações e esquemas XML, chamado de *Security Assertion Markup Language* (SAML) (OASIS, 2005b; OASIS, 2005c). Estas especificações definem *tokens* que contém informações sobre autenticação, autorização ou atributos de um sujeito. No caso, não são definidas novas tecnologias ou formas para autenticação e autorização. São na verdade, nesta especificação, definidos formatos que visam garantir a interoperabilidade entre diferentes sistemas de autenticação e autorização.

Uma asserção de segurança é um conjunto de afirmações, concedidas por um emissor SAML, sobre um determinado principal. Nas especificações, são definidos três tipos de asserções:

(1) Asserção de autenticação: fornecida pelo emissor SAML após o ato de autenticação com sucesso do usuário. Esta asserção contém informações relacionadas ao principal autenticado, o período de validade, etc.;

(2) Asserção de atributos: que contém detalhes específicos sobre o principal em questão, por exemplo, um papel que o principal desempenha dentro do sistema;
(3) Asserção de autorização: indica os direitos que um principal possui sobre um determinado recurso, sendo que esta asserção pode levar como base as asserções de autenticação e de atributos.

O modelo de SAML prevê autoridades responsáveis pela emissão e consumo destas asserções, mas as especificações não fazem menção de como devem ser estas entidades. As especificações, como dito acima, se restringem às formas de interação dessas autoridades.

Em sua primeira versão, o principal objetivo do SAML era permitir a transferência de autenticação e autorização entre aplicações *Web*. A versão 1.1 foi lançada com o intuito de melhorar a interoperabilidade e garantir uma melhor integração com o XMLDSign. Por fim, com base nas iniciativas de projetos Liberty Alliance e Internet2 Shibboleth (CARMODY, 2001), a versão 2.0 da SAML, tem foco mais ousado, dirigido pelo conceito de identidades federadas, apresentando características como (OASIS, 2005b):

- pseudônimos: define identificadores opacos, permitindo assim que principais interajam com o sistema sem a necessidade de revelar qualquer informação que o identifique (como e-mail, nome, etc). O seu uso impede que provedores entrem em comum acordo para cruzar informações de um determinado principal e assim ferir sua privacidade;
- gerenciamento de identidades: define como dois provedores de identidades de domínios administrativos diferentes podem interoperar e gerenciar identificadores permitindo o acesso de aplicações em domínios remotos;
- metadados: definem como expressar dados de configuração e de confiança entre gerentes de identidades. O uso do padrão SAML, neste caso, fica simplificado pelas relações definidas nos metadados entre as entidades participantes. Através destes metadados, papéis, identificações, perfis, URLs e certificados podem ser aceitos em diferentes domínios devido as relações de confiança entre os respectivos gerentes de identidade;
- cifragem: possibilita que atributos, identificadores ou toda a asserção seja cifrada. Tal característica permite garantir a confidencialidade fim a fim em elementos de asserção;

- perfis de atributo: definem como os atributos poderão ser transportados nas asserções SAML. Definem um perfil básico, que utiliza os tipos primitivos do XML para expressar os atributos e também define perfis como X.500/LDAP, UUID e XACML;

A partir da versão 2.0 da SAML, a forma de uso de identidade federadas permite a "ligação entre contas". Agora, as diferentes identidades de um usuário, presentes em diferentes Provedores de Serviço, podem ser associadas de forma que possibilite o SSO, sem ferir a privacidade do usuário. Para isso, a SAML 2.0 propõe o uso de pseudônimos, o que protege a privacidade, impedindo o rastreamento da identidade do usuário.

#### 3.3.1.5 XKMS

O padrão XML *Key Management Specification* (XKMS) (HALLAM-BAKER e MYSORE, 2005) é uma especificação aberta que define interfaces, baseadas em Serviços *Web*, visando retirar dos desenvolvedores de aplicações a complexidade em se trabalhar com Infraestrutura de Chave Pública (ICP), podendo esta ser X.509, SPKI ou mesmo PGP (ZIMMERMAN, 1994). A especificação é dividida em duas subespecificações, XML *Key Information Service Specification* (XKISS) e XML *Key Registration Service Specification* (XKRSS), que juntas definem meios para gerar pares de chaves, armazenar e localizar informações sobre chaves públicas, bem como para validar assinaturas.

A especificação XKISS define os serviços que visam retirar das aplicações a complexidade em se trabalhar com assinaturas expressas em XMLDSign (Bartel *et al*., 2008). Informações como nome da chave, ou um certificado X.509, ou a própria chave, são descritas dentro do elemento XML `<ds:KeyInfo>` de uma assinatura em XMLDSign. Dois serviços são definidos para tratar com informações obtidas a partir de um documento XMLDSign: um para a localização de informações relacionadas às chaves (XKISS *Locate*), e outro para verificar se estas informações relacionadas às chaves são válidas (XKISS *Validate*).

A Figura 3.10 ilustra a utilização em conjunto desses dois serviços para localizar e validar uma assinatura. O usuário aciona o seu serviço XKISS *Validate* para encontrar, de forma confiável, a chave pública do usuário maria@empresa.com.br (passo 1). Na sequência, o XKISS *Validate*, através do XKISS *Locate*, utiliza o serviço de nomes (DNS) para localizar o serviço responsável pelo domínio

empresa.com.br (passo 2). Por fim, o serviço XKISS *Validate* aciona o serviço XKISS *Locate* do domínio empresa.com.br para obter a chave pública do usuário Maria (passo 3).

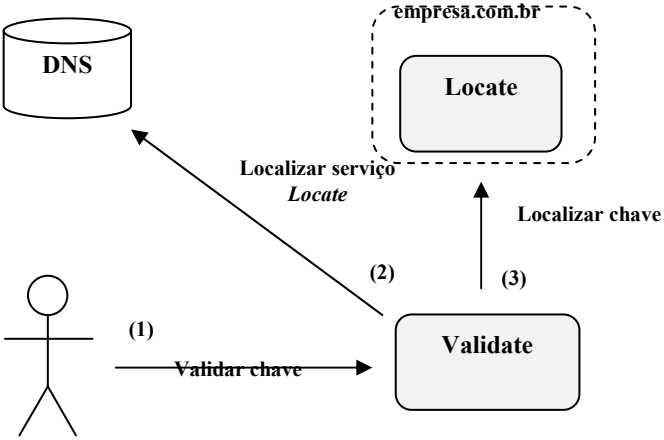


Figura 3.10 – Uso dos serviços XKISS *Locate* e *Validate*
Fonte: Adaptado de (de MELLO *et al.*, 2006)

As especificações XKRSS, definem a forma de registro e o gerenciamento de chaves públicas através dos seguintes serviços:

(1) registro de informações;
(2) reemissão das informações associadas a chaves, permitindo gerar novas credenciais na ICP subjacente, por exemplo, no caso de um certificado expirar;
(3) revogação das informações associadas; e,
(4) recuperação de uma chave privada, associada anteriormente.

Neste último caso, só é possível recuperar a chave privada, se o par de chaves em questão foi gerado pelo XKMS. Isto porque são definidas duas formas de geração de chaves. A primeira sendo o XKMS o gerador e a segunda onde cliente é o gerador. Neste último caso, junto com outras informações, irá somente à chave pública do cliente.

3.3.1.6 WS-Security

A WS-Security (OASIS, 2004a; OASIS, 2004b), apresentada inicialmente pela IBM e Microsoft, é hoje uma especificação

padronizada pela OASIS que tem como objetivo a proposição de extensões ao SOAP para permitir trocas seguras de mensagens entre Serviços *Web*. Com foco na garantia da segurança fim a fim, esta especificação possui três principais pontos (de MELLO *et al*., 2006):

(1) Credenciais de Segurança: incluir nas mensagens SOAP credenciais de segurança com informações de autenticação;
(2) Integridade da Mensagem: incluir nas mensagens SOAP informações relacionadas a assinaturas digitais de toda ou de parte da mensagem;
(3) Confidencialidade da Mensagem: mensagens SOAP podem ser cifradas, totalmente ou somente partes dela.

Através de um esquema XML, a WS-Security define formas padronizadas de incluir informações relacionadas à assinatura e à cifragem de dados em uma mensagem SOAP. Na verdade, esta especificação define como fazer uso do XMLDSign (Bartel *et al*., 2008) e do XMLEnc (IMAMURA *et al*., 2002) para assinar ou encriptar dados em mensagens SOAP.

Para tanto, as informações da WS-Security são incluídas dentro de elementos XML `<wsse:Security>`, sendo que, cada mensagem SOAP pode conter um ou mais destes elementos. Isto porque uma mensagem SOAP pode passar por diversos nós SOAP intermediários desde o remetente até o destinatário final da mensagem. Com isso, a WS-Security pode garantir que somente determinadas partes de uma mensagem possa ser lida ou modificada por determinados nós intermediários.

Cada elemento `<wsse:Security>` deverá identificar, através do atributo SOAP1.2:role, o nó a qual aquela informação está direcionada. Não sendo permitido que haja múltiplos elementos `<wsse:Security>` que tenham como alvo um mesmo nó SOAP. Porém, informações como a assinatura do emissor inicial podem interessar para todos os nós SOAP intermediários e final. Para este acesso múltiplo, é possível se definir um único elemento `<wsse:Security>` sem a necessidade de indicar o nó relacionado a este elemento e isto permite que todos os nós intermediários ou final possam tratar tal elemento (OASIS, 2004b).

Aos nós intermediários é permitido tanto a adição de sub-elementos a um elemento `<wsse:Security>` quando a inclusão de novos elementos `<wsse:Security>` (OASIS, 2004b).

O Código 3.3 apresenta um exemplo de uma mensagem SOAP com o cabeçalho da WS-Security. O exemplo consiste em enviar uma

simples credencial, no caso “Zoe” (linha 7), sem qualquer tipo de proteção. Cada elemento `<wsse:Security>` (linhas 5 a 9) pode expressar informações sobre a cifragem, a assinatura e sobre as credenciais de segurança. As linhas 6 a 8 expressam detalhes sobre uma credencial de segurança, porém os elementos `<wsse:Security>` podem conter mais de uma credencial de segurança.

Código 3.3 – SOAP com cabeçalhos de WS-Security

```
1   <soapenv:Envelope
2      xmlns:soapenv="..." xmlns:wsse="...">
3      <soapenv:Header>
4
5         <wsse:Security>
6            <wsse:UsernameTokenwsu:Id="...">
7               <wsse:Username>Zoe</wsse:Username>
8            </wsse:UsernameToken>
9         </wsse:Security>
10
11     </soapenv:Header>
12     <soapenv:Body>
13
14     </soapenv:Body>
```

Fonte – (OASIS, 2004b)

No exemplo do Código 3.3, o objetivo da mensagem é indicar ao nó SOAP final que, em um nó SOAP mais externo, a autenticação do cliente já foi realizada e esta informação está sendo repassada através do elemento <wsse:Username> (linha 7).

A especificação WS-Security provê suporte a dois tipos de credenciais de segurança: credenciais *UsernameToken* e credenciais *BinarySecurityToken*. Um uso para esta credencial *UsernameToken* está descrito no Código 3.3. A credencial *BinarySecurityToken* apresenta uma forma padrão para anexar a um pedido SOAP qualquer credencial de segurança codificada em forma binária, por exemplo, certificados X.509, tickets Kerberos, etc.

#### 3.3.1.7 WS-SecureConversation

A WS-Security, vista na seção anterior, define um esquema XML como forma de padronizar o uso das especificações XMLDSign e XMLEnc em Serviços *Web*. Entretanto, a especificação é centrada no modelo de autenticação de mensagem. Diferente disto, a WS-SecureConversation define um modelo de autenticação para uma série de mensagens SOAP.

Para isso, a WS-SecureConversation, trabalhando em conjunto com a própria WS-Security, a WS-Trust e a WS-Policy, define extensões que permitem o estabelecimento de contextos de segurança e a criação e derivação da chave de sessão. Os contextos de segurança, depois de criados, são identificados por um novo tipo de *token* definido na WS-Security. Este *token* é usado na troca de mensagens subsequentes indicando o contexto de segurança criado. (OASIS, 2009c)

Entretanto, a criação do contexto de segurança exige trocas de mensagens adicionais. Mas, como consequência, define chaves potencialmente mais eficientes ou geração de novas chaves a serem trocadas. Portanto, o uso desta especificação aumenta o desempenho e a segurança para trocas subsequentes de mensagens. (OASIS, 2009c)

A especificação WS-SecureConversation define três modos para estabelecimento de contexto de segurança:

(1) Criação de *token* de contexto de segurança por um Serviço de Emissão de *Token* de Segurança: neste caso, o contexto de segurança é criado por uma terceira parte confiável (*trusted third party*). Assim, a parte que deseja iniciar uma comunicação faz a solicitação a este serviço, que após criar o contexto de segurança encaminha para as partes que farão parte da comunicação a identificação do mesmo (*token* de segurança).

(2) Criação de *token* de segurança por uma das partes da comunicação e propagada por uma mensagem: neste modo, a criação do *token* que indica o contexto de segurança é realizada pelo próprio iniciador da comunicação e então distribuída pelo mesmo às partes que farão parte desta comunicação.

(3) Criação de *token* de segurança através de negociação entre as partes: neste modo, é adotada uma solução negociada em que parceiros trocam suas políticas e definem suas chaves e algoritmos também através de trocas entre as partes envolvidas.

Além do estabelecimento de contexto de segurança, a especificação descreve também como renovar, alterar ou cancelar os mesmos. Por fim, ainda é especificado como é realizada a derivação de chaves, bem como a mesmas devem ser trocadas pelos pares que desejam se comunicar.

Não esta no escopo desta especificação definir como a confiança entre as partes é estabelecida. Esta especificação destina-se a fornecer um conjunto flexível de mecanismos que podem ser utilizados para apoiar uma variedade de protocolos de segurança. Alguns protocolos podem necessitar de mecanismos distintos ou perfis restritos da presente especificação.

#### 3.3.1.8 Políticas para os Serviços *Web*

Como visto antes, a WSDL é a forma padronizada utilizada em Serviços *Web* para que os provedores de serviços especifiquem quais são os serviços oferecidos, quais informações são necessárias para invocar um serviço e como deverá ser o formato para troca de informações com os clientes. Porém, há ainda a necessidade de descrever as habilidades e requisitos não funcionais de um serviço, tais como, os mecanismos de segurança providos ou exigidos por um serviço, permitindo assim a um cliente decidir por um ou outro serviço baseado nessas características. Sobre tal motivação, surgiram especificações como o WS-Policy (W3C, 2007a) e o WSPolicyAttachment (W3C, 2007b). Estas especificações foram inicialmente propostas por um consórcio de empresas formado pela Microsoft, IBM, VeriSign e outras, tornando-se uma recomendação da W3C no ano de 2007.

O primeiro, WS-Policy, provê um modelo para descrever necessidades em termos de mecanismos e protocolos de segurança para acessar um serviço correspondente à WSDL considerada. O WS-Policy, através de uma gramática flexível e extensível, permite descrever uma ampla variedade de requisitos, mecanismos e habilidades para uma aplicação acessar um Serviço *Web*. Já a especificação WS-PolicyAttachment descreve como associar estas políticas WS-Policy com os recursos e a elementos XML que compõem um documento WSDL.

A especificação WS-Policy apresenta uma estrutura para descrição de políticas dividida em três principais componentes:

(1) asserção de política – expressa a habilidade específica do domínio do serviço considerado. São obrigações ou requisitos que o requisitante do serviço deve cumprir. Por exemplo, necessidade de apresentar credenciais de um esquema de autenticação (usado no domínio do serviço desejado) ou usar algoritmos ou tecnologias próprias para a troca confiável de mensagens com o servidor considerado;
(2) alternativas de políticas – descrevem as combinações aceitáveis de obrigações e requisitos (conjunto de asserções de política), para a interação entre o requisitante e o serviço;
(3) política – expressa um conjunto de alternativas de políticas válidas.

Com o intuito de facilitar a interoperabilidade entre políticas expressas pela WS-Policy foi definida uma forma normal para expressar políticas. O Código 3.4 exemplifica a estrutura à qual a política deve se adequar para estar nesta forma normal.

Código 3.4 – Exemplo de WS-Policy na forma normal

```
1  <wsp:Policy ...>
2     <wsp:ExactlyOne>
3        ( <wsp:All> ( <Assertion …> … </Assertion> )* </wsp:All>
4     </wsp:ExactlyOne>
5  </wsp:Policy>
```

Fonte – (W3C, 2007a)

Neste exemplo, expresso pelo Código 3.4, um documento XML é representado e seu elemento raiz definido é o Policy; dentro deste elemento são representadas as coleções de asserções, que quando combinadas representam um conjunto válido de alternativas de políticas. As asserções são combinadas através de dois tipos de operadores de políticas:

(1) *ExactlyOne*: indica que somente uma das asserções contidas na política poderá fazer parte de uma alternativa de política;
(2) *All*: permite a combinação de todas as asserções apresentadas como uma alternativa de política.

É através das diferentes maneiras de aninhar ou não os operadores *ExactlyOne* e *All* que são definidas as alternativas de política.

A especificação WS-Policy não define como associar políticas a Serviços *Web*, esta associação fica a cargo da especificação WS-PolicyAttachment, que descreve como associar políticas com determinados recursos e elementos XML que compõem um documento WDSL e a elementos UDDI. Estes recursos em Serviços *Web* que podem ser uma troca de mensagens, um serviço, uma coleção de serviços, etc., são representados nos documentos WSDL, pelos elementos *messages*, *portType*, *binding*, *service*, entre outros.

A WS-PolicyAttachment define duas maneiras de associar uma política. A primeira permite que estas sejam anexadas diretamente dentro de documentos XML, e a segunda permite que a associação delas não necessite a presença das políticas e dos recursos dentro de um mesmo documento XML.

Também vale citar a especificação WS-SecurityPolicy (OASIS, 2009a) que tem por objetivo descrever como deverão ser as asserções de segurança, usadas na WS-Policy, para descrever as características de segurança empregadas nas especificações WS-Security (OASIS, 2004a; OASIS, 2004b), WS-Trust (OASIS, 2009b) e WS-SecureConversation (OASIS, 2009c). A intenção da WS-SecurityPolicy é, também, permitir certa flexibilidade em termos de *token*s, criptografia e mecanismos usados na segurança em nível de transporte, definindo o suficiente para a interoperabilidade com base nas asserções de segurança.

## 3.4 CONCLUSÃO

Este capítulo revisou os principais conceitos da Arquitetura Orientada a Serviços e sua mais bem sucedida implementação que é Serviços *Web*. Foram abordadas também, as principais especificações desta tecnologia que visam a segurança de aplicações distribuídas. Estas especificações não definem uma tecnologia de segurança especifica subjacente a Serviços *Web*. Pelo contrário, estas especificações são suficientemente gerais de modo a não impor tecnologias de segurança e ainda permitem aos requisitantes se adaptarem aos requisitos de segurança do domínio do provedor de serviço.

Os conceitos abordados neste capítulo são importantes para o entendimento do próximo capítulo e formam a base do trabalho

desenvolvido nesta dissertação. A partir de uma necessidade de mercado, empresas como Microsoft, Intel, Ricoh e Lexmark apresentaram um perfil, enquadrando tecnologias de Serviços *Web* para utilização em equipamentos com restrições computacionais. É sobre isto que trataremos no próximo capítulo.

# 4 DEVICES PROFILE FOR *WEB* SERVICES (DPWS)

## 4.1 INTRODUÇÃO

Recentemente o desenvolvimento de infraestruturas de serviços próprias para dispositivos embarcados tem sido objeto de grande interesse. Estes dispositivos estão presentes em equipamentos de telecomunicação, sistemas embarcados em veículos automotivos, automação predial, instrumentação médica e automação em geral. Com o avanço da tecnologia e a grande disponibilidade de acesso através de redes (redes sem fio ou a cabo), estes dispositivos podem, por exemplo, ser facilmente integrados a sistemas de informação de corporações, ou se tornarem visíveis em níveis mais altos de aplicações distribuídas presentes na Internet. É neste contexto que surge o esforço de estender as facilidades do SOA para pequenos dispositivos.

Embora a computação ubíqua introduza novas perspectivas para a faixa de ambientes industriais a utilitários domésticos, muitas preocupações se fazem presentes com estas novas possibilidades. Estes dispositivos, ao se fazerem presentes em aplicações de larga escala, em redes abertas, envolvendo tarefas críticas, tornam difícil a manutenção das propriedades de confidencialidade, integridade e disponibilidade. Os benefícios da evolução desta tecnologia para componentes embarcados só poderão ser usufruídos em grandes sistemas distribuídos, se os aspectos de segurança forem propriamente endereçados.

Na sequência são apresentadas as principais características da especificação ou Perfil de Dispositivos para Serviços *Web* (DPWS). As recomendações de segurança nesta especificação são também objeto das seções subsequentes.

## 4.2 HISTÓRICO DO DPWS

Desde o surgimento do primeiro esboço da especificação DPWS, ele vem sendo estudado por diversos projetos de pesquisa, principalmente os de aplicações mais práticas.

Um dos trabalhos que abordam a utilização de Serviços *Web* em dispositivos embarcados veio de uma iniciativa européia, através do projeto SIRENA, conforme apresentado no artigo dos autores Jammes e

Smit (2005) e no site <http://www.sirena-itea.org/>. O objetivo do trabalho foi o de desenvolver uma infraestrutura de serviços de tempo real para aplicações embarcadas em rede. Buscou, através de SOA, interligar dispositivos embarcados entre quatro áreas distintas – a industrial, a de telecomunicações, a automotiva e a de automação residencial. Através de um estudo sobre diversas tecnologias orientadas a serviço, chegou-se à escolha da especificação DPWS e à criação de uma ferramenta de desenvolvimento que atendesse às especificações.

Outros dois projetos utilizaram como base os estudos e a ferramenta desenvolvida no projeto SIRENA. O SODA (*Service Oriented Device & Delivery Architecture*), como pode ser visto no sítio <http://www.soda-itea.org/>, com objetivo de criar uma plataforma abrangente, escalável e de fácil de implantação de sistemas baseados em SOA; o SOCRADES (<http://www.socrades.eu/>) investiga a aplicabilidade do DPWS para um grande número de dispositivos e sua integração em uma infraestrutura orientada a serviços fortemente acoplada ao chão de fábrica e às atividades da camada de negócio.

Atualmente, o DPWS é uma especificação, na versão 1.1, mantida pelo consórcio OASIS (OASIS, 2009d).

## 4.3 MODELO DPWS

O DPWS define um conjunto mínimo de funcionalidades que permitem dispositivos com limitações de recursos suportarem Serviços *Web*. Este perfil foi apresentado em maio de 2004 por um grupo de empresas, entre elas, a Microsoft, a Intel, a Ricoh e a Lexmark. Originalmente, este conjunto de protocolos para Serviços *Web* foi concebido para ser a nova versão do popular UPnP. Entretanto, por razões estratégicas de mercado e incompatibilidade entre o UPnP com as pilhas de protocolos DPWS, eles são tratados como tecnologias diferentes (MENSCH e ROUGES, 2009).

A arquitetura de Serviços *Web* especifica uma rica quantidade de protocolos, cuja combinação permite o desenvolvimento das mais variadas aplicações. O Perfil, que este capítulo introduz, identifica um conjunto de protocolos de Serviços *Web* que são propostos para preencherem as seguintes características:

(1) Segurança na troca de mensagens entre Serviços *Web*.

(2) Descoberta dinâmica de Serviços *Web*.

(3) Mecanismos de descrição de Serviços *Web*.

(4) Suporte para Serviços de Eventos.

Estas características listadas acima de certa maneira são determinantes na escolha de protocolos e serviços usados na definição do perfil. Além disto, neste perfil são estabelecidos requisitos mínimos e adaptação necessária para permitir a implementação direta em componentes embarcados destes protocolos e serviços. Estas adaptações ou extensões necessárias, todavia, não comprometem as padronizações de Serviços *Web*.

O modelo computacional enfatizado na especificação DPWS preconiza que os dispositivos podem assumir diferentes papéis. São: ou consumidores de serviços (clientes), ou serviços ou, ainda, ambos. No caso de serviços, dois tipos são distinguidos na especificação: serviços de hospedagem (*hosting services*) e serviços hospedados (*hosted services*). A Figura 4.1 ilustra como os dois tipos de serviços se enquadram no modelo.

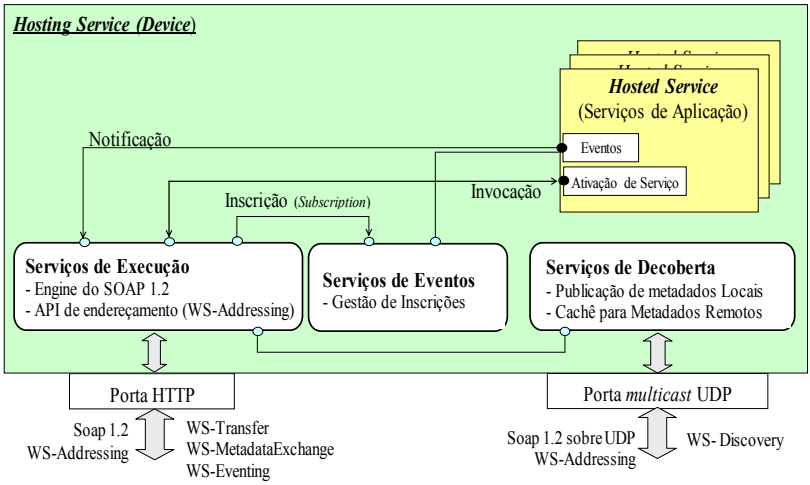


Figura 4.1 – Modelo Computacional do DPWS
Fonte: Adaptado de (MENSCH e ROUGES, 2009)

Os chamados serviços de hospedagem (ou de dispositivo) são parte importante do modelo DPWS. Vários dos aspectos não funcionais que atuam na evolução de serviços de aplicação estão concentrados no componente dispositivo, na forma de serviços embutidos. Estes serviços,

que abrangem as quatro áreas citadas pela especificação serão mais bem explicados na próxima seção que lista os protocolos usados.

## 4.4 PILHA DE PROTOCOLOS DPWS

A seguir são listados os protocolos de Serviços *Web* definidos pelo perfil que compõem a especificação. Este conjunto de protocolos, que também inclui o SOAP e a WSDL, cobre as áreas de troca de mensagens, autodescoberta, troca de metadados e eventos.

(1) WS-Addressing (W3C, 2006b): está intimamente relacionado ao SOAP e concentra toda a informação de endereçamento da mensagem nos cabeçalhos do envelope da mensagem SOAP. Isto permite que as mensagens sejam enviadas por qualquer protocolo de transporte (HTTP, SMTP, TCP, UDP etc).

(2) WS-Policy: é usado para expressar as políticas associadas ao Serviço *Web*. Estas políticas são expressas na forma de "política de asserções" (*policy assertions*) e complementam a descrição do serviço feito pela WSDL, declarando os mecanismos e algoritmos de segurança que o Serviço *Web* correspondente faz uso.

(3) WS-MetadataExchange (W3C, 2010a): permite a recuperação dinâmica de metadados associados a um Serviço *Web* (descrição, esquema e políticas), propiciando assim aos consumidores a obtenção de informações de acesso ao serviço.

E como mencionado antes, o DPWS também adicionou protocolos além daqueles de Serviços *Web*. Estes envolvem os serviços de descoberta e de eventos, e são eles:

(4) WS-Discovery (OASIS, 2009e): descreve um protocolo de descoberta que se utiliza de comunicação *multicast* para que dispositivos se divulguem na rede. Um exemplo de uso deste protocolo é quando um cliente está procurando por um ou mais serviços. Para isso, ele envia uma mensagem de reconhecimento ao endereço de grupo *multicast*, e os dispositivos que correspondam à mensagem responderão diretamente ao cliente.

(5) WS-Eventing (W3C, 2010b): é serviço que permite um Serviço *Web* (chamado "assinante") se registrar ("inscrição") para receber de outro Serviço *Web* (chamado "fonte de eventos") mensagens sobre eventos (chamadas de "notificações" ou "mensagens de eventos") produzidos por este último. O assinante pode gerenciar a inscrição através da interação com o Serviço *Web* "gerenciador de inscrições" informado pela fonte de eventos.

A pilha de protocolos DPWS é mostrada na Figura 4.2, e é composta dos protocolos citados acima. Uma das exigências feitas na especificação do perfil (OASIS, 2009d) é que o DPWS use o protocolo SOAP na versão 1.2 e a WSDL na versão 1.1.

A especificação do DPWS, em sua maior parte, é baseada em especificações padrões W3C. Entretanto, para atender às necessidades dos dispositivos embarcados, que possuem recursos limitados, são feitas algumas simplificações em alguns itens nas especificações originais de protocolos para Serviços *Web* usados no DPWS.

As próximas subseções fornecem mais detalhes das características mais específicas do Perfil de Dispositivos para Serviços *Web*.

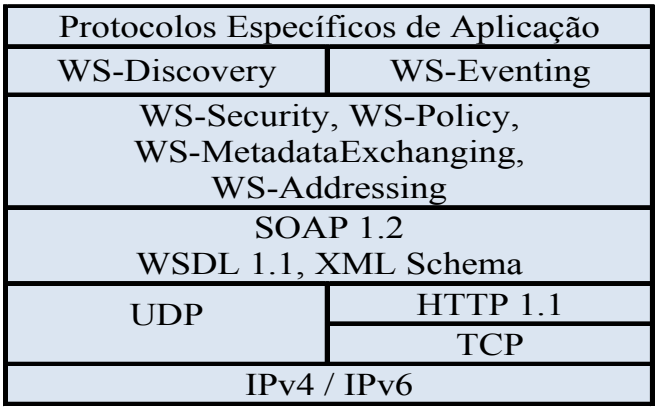


Figura 4.2 – Pilha de protocolos DPWS

Apesar da WS-Security aparecer na pilha de protocolos do DPWS, as especificações do perfil recomendam, opcionalmente, o uso da mesma para assinatura de mensagens usando a *Compact Signature*

*Format*. Neste trabalho, abordaremos de maneira mais enfática o uso da WS-Security no DPWS.

### 4.4.1 Troca de mensagens

Como mencionado antes, o DPWS introduz algumas adaptações nas especificações originais de Serviços *Web* e estabelece alguns requisitos. Um destes requisitos obrigatórios determinados pelo DPWS é o uso de SOAP na versão 1.2 como protocolo de *request/reply* entre serviços que se comunicam. Esta escolha é devido às possibilidades de extensão incluídas no SOAP na versão citada. O SOAP é um ponto chave da pilha de protocolos DPWS; a arquitetura de Serviços *Web* se torna altamente flexível com a adoção deste protocolo.

O uso dos cabeçalhos SOAP permite que os diferentes protocolos de Serviços *Web* se integrem individualmente e gradativamente, sem que interfiram no restante da pilha de protocolos, bem como permitindo melhorias e modificações isoladas sem afetar a pilha completamente.

O WS-Addressing é uma das extensões do SOAP e tem o propósito de mover todas as informações de endereçamento das mensagens para o cabeçalho SOAP, permitindo assim, padrões de trocas de mensagens mais complexos do que o modelo requisição e resposta do HTTP. E isto é possível, pois com o uso do endereçamento nos cabeçalhos SOAP, há o desacoplamento da mensagem SOAP do protocolo que serve como transporte. Para realizar isso, o WS-Addressing define um conjunto de cláusulas para as mensagens SOAP:

(1) *To*: define o destinatário da mensagem;
(2) *Action*: define a semântica da mensagem;
(3) *ReplyTo*: usada nos casos onde uma resposta é esperada. Pode ser usada também para rotear a mensagem para outro endereço, que não o remetente;
(4) *MessageId*: é usada nos casos de roteamento para que um destinatário de um *hop* possa sempre se referir a uma mensagem encaminhada pelo emissor imediato no roteamento;
(5) *RelatesTo*: A resposta a este remetente intermediário no roteamento é identificada através desta cláusula que notifica a mensagem relacionada à resposta através de seu *'MessageId'*.

Outra vantagem do uso do SOAP como protocolo padrão é o fato de poder compartilhar funcionalidades entre os protocolos de mais alto nível. Por exemplo, usar os mesmos mecanismos de segurança para

mensagens de controle, descoberta e de eventos em dispositivos (JAMMES *el al.*, 2005).

Com o uso do WS-Addressing temos um mecanismo bem definido de trocas de mensagens assíncronas com habilidade de relacionar as mensagens. Isso adiciona o conceito de *resource model* para Serviços *Web*: a organização dos recursos por trás de um Serviço *Web* é oculta, ou seja, o cliente que irá invocar um serviço não precisa saber como ele está organizado. Ele terá somente que saber o endereço de acesso ao serviço, denominado *End-Point Reference* (EPR), que é composto de um Endereço e as Propriedades da referência.

Um endereço EPR é um endereço lógico (uma URI), que identifica o endereço físico através de ligações apropriadas. Em DPWS, a parte de Endereço de qualquer EPR é construído para ser uma URI do tipo UUID, identificando assim somente um dispositivo.

Como em DPWS um dispositivo pode hospedar um serviço ou mais, a Propriedade da Referência "*ServiceId*" é usada para diferenciar um dispositivo de um dos serviços hospedados pelo dispositivo.

Para exemplificar o uso da Propriedade de Referência "*ServiceId*", no Código 4.1, é demonstrado através das linhas 31 e 43 o uso desta propriedade. A mensagem SOAP do Código 4.1 exemplifica a resposta de um dispositivo a uma requisição *Get* (W3C, 2006a), onde o mesmo dispositivo tem dois serviços hospedados: PrintService e ScanService.

Código 4.1 – Resposta a uma requisição *Get* (WS-Transfer)

```
01  <soap:Envelope
02     xmlns:gen="http://example.org/general"
03     xmlns:img="http://printer.example.org/imaging"
04     xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
05     xmlns:wsdp="http://schemas.xmlsoap.org/ws/2006/02/devprof"
06     xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
07     xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" >
08     <soap:Header>
09      <wsa:Action>
10      http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
11     </wsa:Action>
12     <wsa:RelatesTo> urn:uuid:82204a83-52f6-475c-9708-174fa27659ec
</wsa:RelatesTo>
13     <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/
anonymous</wsa:To>
14    </soap:Header>
15    <soap:Body>
```

```
16      <wsx:Metadata>
17       <wsx:MetadataSection
18        Dialect="http://schemas.xmlsoap.org/ws/2006/02/devprof/
Relationship">
19       <wsdp:Relationship
20        Type="http://schemas.xmlsoap.org/ws/2006/02/devprof/host" >
21       <wsdp:Hosted>
22        <wsa:EndpointReference>
23         <wsa:Address>http://172.30.184.244/print</wsa:Address>
24        </wsa:EndpointReference>
25        <wsa:EndpointReference>
26         <wsa:Address>http://[fdaa:23]/print1</wsa:Address>
27        </wsa:EndpointReference>
28        <wsdp:Types>
29         img:PrintBasicPortType img:PrintAdvancedPortType
30        </wsdp:Types>
31        <wsdp:ServiceId>
32         http://printer.example.org/imaging/PrintService
33        </wsdp:ServiceId>
34       </wsdp:Hosted>
35       <wsdp:Hosted>
36        <wsa:EndpointReference>
37         <wsa:Address>http://172.30.184.244/scan</wsa:Address>
38        </wsa:EndpointReference>
39        <wsa:EndpointReference>
40         <wsa:Address>http://[fdaa:24]/scan</wsa:Address>
41        </wsa:EndpointReference>
42        <wsdp:Types>img:ScanBasicPortType</wsdp:Types>
43        <wsdp:ServiceId>
44         http://printer.example.org/imaging/ScanService
45        </wsdp:ServiceId>
46       </wsdp:Hosted>
47       </wsdp:Relationship>
48      </wsx:MetadataSection>
49      <!-- Other Metadata Sections omitted for brevity. -->
50      </wsx:Metadata>
51     </soap:Body>
52   </soap:Envelope>
```

Fonte – (OASIS, 2009d)

Como em qualquer outro Serviço *Web*, serviços baseados em DPWS são descritos usando XMLSchema, WSDL e WS-Policy, como será visto na sequência.

### 4.4.2 Descoberta e Descrição

Outro requisito obrigatório da tecnologia DPWS é o uso de mecanismos de descoberta. Para isto, o protocolo WS-Discovery (OASIS, 2009d) é usado no DPWS em descobertas e publicação de dispositivos. O WS-Discovery é construído sobre um protocolo de *multicast* para a localização de recursos dispersos através da rede considerada. Usadas nas trocas durante a descoberta de dispositivos e clientes na rede, o WS-Discovery define ainda diferentes tipos de mensagens:

(1) **Hello**: mensagem enviada pelo dispositivo anunciando sua entrada na rede.
(2) **Probe**: mensagem emitida por clientes quando em busca de dispositivos de um determinado tipo.
(3) **Probe-Match**: mensagem de resposta enviada dispositivo para sinalizar que este corresponde ao tipo de dispositivo solicitado em uma mensagem Probe.
(4) **Resolve**: mensagem de clientes solicitando dispositivo com um determinado nome.
(5) **Resolve-Match**: resposta emitida por dispositivo com o nome indicado em uma mensagem Resolve.
(6) **Bye**: mensagem enviada pelo dispositivo anunciando que está saindo da rede.

No processo de descoberta, os dispositivos expõem através das mensagens do tipo 'Hello', os seguintes dados:

Seu EPR, que permitirá determinar o endereço físico do dispositivo.

- Tipos: um conjunto de mensagens que o dispositivo pode enviar e receber; podem ser funcionais (exemplo: 'ligar', 'desligar') ou tipos abstratos agrupando diversos tipos e/ou serviços hospedados (exemplo: 'impressora', 'iluminação artificial', 'gateway residencial').
- Escopo: um conjunto de atributos que podem ser usados para organizar dispositivos em grupos lógicos ou hierárquicos. Por exemplo, atributos de localização ou de direitos de acesso podem ser usados para agrupar dispositivos.

O Código 4.2 é um exemplo de mensagem do tipo ‘Hello’. As linhas 16, 17 e 18 mostram, respectivamente, a indicação do EPR, Tipos e Escopo do dispositivo.

Código 4.2 – Mensagem do tipo "Hello"

```
1   <s:Envelope ... >
2    <s:Header ... >
3     <a:Action ... >
4      http://schemas.xmlsoap.org/ws/2005/04/discovery/Hello
5     </a:Action>
6     <a:MessageID ... >xs:anyURI</a:MessageID>
7      [<a:RelatesTo RelationshipType="d:Suppression" >
8      xs:anyURI
9     </a:RelatesTo>]?
10    <a:To ... >urn:schemas-xmlsoap-org:ws:2005:04:discovery</a:To>
11    <d:AppSequence ... />
12    .......
13   </s:Header>
14   <s:Body ... >
15    <d:Hello ... >
16     <a:EndpointReference> ... </a:EndpointReference>
17     [<d:Types>list of xs:QName</d:Types>]?
18     [<d:Scopes>list of xs:anyURI</d:Scopes>]?
19     [<d:XAddrs>list of xs:anyURI</d:XAddrs>]?
20     <d:MetadataVersion>xs:unsignedInt</d:MetadataVersion>
21     .......
22    </d:Hello>
23   </s:Body>
24  </s:Envelope>
```

Fonte – (OASIS, 2009e)

Como citado antes, a descrição dos serviços em DPWS apresenta basicamente as mesmas informações das descrições usadas em Serviços *Web*. Entretanto em DPWS algumas restrições devem ser incorporadas e outras flexibilizadas. Em algumas circunstâncias, um cliente precisará saber tudo sobre um dispositivo e seus serviços hospedados para enviar e receber mensagens específicas da aplicação. Para que um cliente consiga essas informações sobre essas funcionalidades de um dispositivo e/ou de seus serviços hospedados, com modelos estendidos

ou não, ele precisará recuperar a descrição desse dispositivo e/ou de serviço hospedado (metadados).

Um cliente conseguirá obter as características de um dispositivo e seus serviços hospedados através do uso das mensagens GetMetadata definidas na WS-MetadataExchange que envolve as seguintes cláusulas para descrição de metadados:

- **ThisModel**: os metadados deste item trazem informações sobre o tipo do dispositivo. Por ex.: nome do fabricante, modelo, número do modelo etc;
- **ThisDevice**: metadados que fornecem informações sobre o dispositivo considerado. Por ex.: número serial, versão de firmware e nome.
- **Relationship**: lista dos serviços hospedados pelo dispositivo, englobando seus EPRs e tipos dos serviços hospedados.
- **WSDL**: contém a localização das descrições de operações e as estruturas de mensagens implementadas pelo *endpoint* endereçado; estas descrições podem ser exploradas por uma implementação genérica de interpretador WSDL.

A resposta de um dispositivo, com seus metadados, é exibida no Código 4.3. Os metadados sobre o dispositivos estão dentro do corpo da mensagem SOAP. Nas linhas 11-17, estão os metadados com informações sobre o tipo do dispositivo e, nas linhas 18-23, são encontrados os metadados com informações específicas sobre o dispositivo.

Código 4.3 – Metadados de dispositivos

```
<soap:Envelope
 xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:wsdp="http://schemas.xmlsoap.org/ws/2006/02/devprof"
 xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" >
 <soap:Header>
 <!-- Section omitted for brevity. -->
 </soap:Header>
 <soap:Body>
 <wsx:Metadata>
  <wsx:MetadataSection Dialect="http://schemas.xmlsoap.org/ws/2006/02/devprof/ThisModel" >
   <wsdp:ThisModel>
    <wsdp:Manufacturer>ACME Manufacturing</wsdp:Manufacturer>
    <wsdp:ModelName xml:lang="en-GB" >ColourBeam 9</wsdp:ModelName>
    <wsdp:ModelName xml:lang="en-US" >ColorBeam 9</wsdp:ModelName>
   </wsdp:ThisModel>
  </wsx:MetadataSection>
  <wsx:MetadataSection Dialect="http://schemas.xmlsoap.org/ws/2006/02/devprof/ThisDevice" >
  <wsdp:ThisDevice>
   <wsdp:FriendlyName xml:lang="en-GB" >ACME ColourBeam Printer</wsdp:FriendlyName>
   <wsdp:FriendlyName xml:lang="en-US" >ACME ColorBeam Printer</wsdp:FriendlyName>
  </wsdp:ThisDevice>
 </wsx:MetadataSection>
 <!-- Other Metadata Sections omitted for brevity. -->
 </wsx:Metadata>
 </soap:Body>
</soap:Envelope>
```

Fonte – (OASIS, 2009d)

### 4.4.3 Cenário de Busca

Um cenário comum de uso do WS-Discovery é um cliente buscando um ou mais dispositivos. Os serviços hospedados, que foram explicados anteriormente, não participam do processo de descoberta, mas são individualmente endereçados (através de suas EPRs), uma vez que os dispositivos que os hospedem sejam descobertos.

Um sistema de descoberta, como é o caso do WS-Discovery, pode interferir no comportamento da rede, se não forem tomadas certas precauções com os protocolos subjacentes. A WS-Discovery permite o uso do protocolo UDP na troca de mensagens que é feito com o uso do SOAP-over-UDP (OASIS, 2009f). Pelas características do UDP que simplesmente não oferece garantias de entrega, o SOAP-over-UDP define então alguns padrões de trocas de mensagens: *Unicast one-way, Multicast one-way, Unicast request, Unicast response* e *Multicast request, Unicast response*. Basicamente, as mensagens SOAP são

enviadas sobre UDP e utilizam a WS-Addressing para trazer ao nível da aplicação as questões de endereçamento e aspectos de reposição de mensagens.

Outro recurso, do WS-Discovery, que minimiza os efeitos de sobrecarga na rede é o uso de mensagens de *multicast* do tipo 'Hello' pelos dispositivos quando entram na rede. Sendo assim, os nós que já estão na rede, através do grupo *multicast*, são avisados de novos dispositivos entrando na rede e com isto, atualizam em *cache* as informações sobre os mesmos. Da mesma forma, quando um dispositivo deixa a rede, envia uma outra mensagem de *multicast* do tipo 'Bye'. Esta característica de mensagens por *multicast* faz com que diminuam as consultas por dispositivos ou serviços, pois os componentes que estão na rede já possuem informações sobre os serviços que entram e saem da rede, podendo enviar mensagens diretamente aos serviços alvo sem necessitar fazer uso de mensagens adicionais para encontrá-los.

A Figura 4.3 ilustra os processos de descoberta e de busca de metadados (descrição de dispositivos) que estão sempre presentes em uma rede de dispositivos.

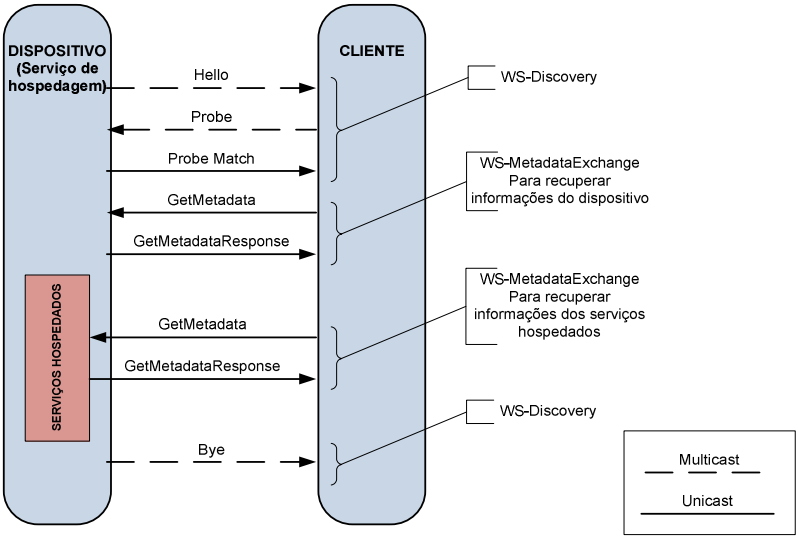


Figura 4.3 – Trocas de mensagens de descoberta e de obtenção de metadados de dispositivos e serviços
Fonte: Adaptado de (OASIS, 2009e; ARAÚJO, 2009, p. 23.)

As três primeiras mensagens da Figura 4.3 são, respectivamente, um dispositivo se anunciando na rede (*Hello*), um cliente pesquisando por um determinado serviço (mensagem *Probe*) e, por fim, o dispositivo respondendo a requisição informando possuir o serviço procurado (mensagem *Probe Match*). Estas trocas fazem parte da Descoberta Dinâmica de Serviços, enfatizada na especificação WS-Discovery. Na sequência, quatro mensagens são trocadas para recuperação, por parte do cliente, dos *metadados* do dispositivo e de serviço hospedado. Estas trocas acontecem seguindo as especificações WS-MetadataExchange. Com as informações de metadados o cliente pode então iniciar a comunicação com o serviço desejado. E por fim, quando um dispositivo deseja abandonar a rede, anuncia antes com uma mensagem de WS-Discovery do tipo *Bye*.

### 4.4.4 Notificação de Eventos

O sistema de notificação de eventos em DPWS, assim como em Serviços *Web*, envolve simplesmente trocas de mensagens SOAP. Estas mensagens fazem uso do WS-Addressing, e seguem as descrições WSDL para o serviço correspondente de eventos. Como mencionado antes, o sistema de eventos é uma exigência na tecnologia de DPWS que, para tal, usa os mecanismos de controle e notificação de eventos da especificação WS-Eventing. Assim como com as outras especificações, o DPWS impõe adaptações ao contexto de dispositivos embarcados no WS-Eventing.

No WS-Eventing, é definido um protocolo para publicação e inscrição em eventos, permitindo que um Serviço *Web* (chamado de "sorvedouro de eventos" ou "consumidor de eventos") registre interesse ("inscrição") para receber notificações de eventos produzidos por outro Serviço *Web* (o serviço "fonte de eventos").

O modo de entrega de mensagens em um sistema de eventos pode variar conforme o tipo de aplicação. Em alguns casos, o simples envio de mensagens assíncronas pode ser adequado (modo *Push*). Em outras situações pode ser melhor que o sorvedouro de eventos faça a solicitação das mensagens de evento, podendo assim controlar o fluxo e o tempo de absorção destas mensagens de notificação (modo *Pull*). Outros consumidores podem desejar que diversas mensagens de eventos venham todas empacotadas em uma única mensagem SOAP. Além disso, alguns consumidores podem desejar que haja confiabilidade na entrega destas mensagens de eventos e outros, ao contrário, aceitam somente o melhor esforço na entrega das notificações de eventos. Neste

sentido, a especificação WS-Eventing para o DPWS introduz uma abstração chamada "Modo de Entrega" (*DeliveryMode*), que é um ponto de extensão na especificação usual, permitindo que sejam criados modos de entrega e consumo de eventos conforme a necessidade das aplicações.

Apesar da extensão para os modos de entrega, a WS-Eventing mantém ainda os modos convencionais *Push* (o serviço de eventos envia as notificações) e *Pull* (o sorvedouro faz requisições por notificações existentes ao serviço).

A inscrição de um sorvedouro de eventos em uma fonte de eventos é por um tempo determinado. Mas uma fonte de eventos pode permitir que um sorvedouro de eventos renove constantemente sua inscrição. Para definir os sorvedouros de determinadas fontes, o WS-Eventing comporta as seguintes operações de configuração: Inscrição (*Subscribe*), Renovação (*Renew*) e Cancelar Inscrição (*Unsubscribe*).

O serviço de eventos, como qualquer *Serviço Web*, faz uso de mensagens SOAP. Assim sendo, o Código 4.4 exemplifica uma mensagem SOAP hipotética de inscrição em um serviço de eventos. Como exibidas no cabeçalho da mensagem SOAP, as linhas 7–9 indicam que esta é uma mensagem que requisita uma inscrição e, na linha 16, é informado que esta mensagem está sendo enviada a um serviço de eventos que trata com eventos oceânicos. Nas linhas 13–15, é informado para onde a fonte de eventos deve responder. São indicados, nas linhas 20–29, onde e como as notificações devem ser entregues. A ausência do atributo Modo ("*Mode*") na linha 20 indica que as notificações devem ser entregues no modo *Push*; ou seja, a fonte de eventos envia mensagens SOAP assíncronas (segundo a ocorrência de seus eventos) ao *endpoint* (linhas 21-28). Note nas linhas 25–27, um exemplo de padrão típico de um sorvedouro de eventos que exibe uma propriedade de referência (Linha 26), identificando a inscrição para a notificação.

Além da limitação do tempo de uma inscrição na WS-Eventing, a fonte de eventos pode fornecer filtragem de mensagens para limitar o número de notificações enviadas ao sorvedouro de eventos. Neste caso, uma requisição de inscrição define um filtro. Com isto, a fonte de eventos envia somente mensagens de acordo com o filtro introduzido. Em WS-Eventing os filtros são normalmente baseados em *XPath*, entretanto, o DPWS usa um dialeto próprio[10], que nada mais é do que

[10] http://schemas.xmlsoap.org/ws/2006/02/devprof/Action

uma lista de URIs separadas por espaço indicando as propriedades das ações desejáveis das Notificações. Este “dialeto” é permitido pela WS-Eventing. Os serviços hospedados em dispositivos devem então pelo menos suportar filtros neste dialeto próprio.

Código 4.4 – Mensagem de inscrição em eventos

```
1   <s12:Envelope
2   xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
3   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
4   xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
5   xmlns:ew="http://www.example.com/warnings" >
6    <s12:Header>
7     <wsa:Action>
8      http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe
9     </wsa:Action>
10    <wsa:MessageID>
11    uuid:d7c5726b-de29-4313-b4d4-b3425b200839
12    </wsa:MessageID>
13    <wsa:ReplyTo>
14     <wsa:Address>http://www.example.com/MyEventSink</wsa:Address>
15    </wsa:ReplyTo>
16    <wsa:To>http://www.example.org/oceanwatch/EventSource</wsa:To>
17   </s12:Header>
18   <s12:Body>
19    <wse:Subscribe>
20     <wse:Delivery>
21      <wse:NotifyTo>
22       <wsa:Address>
23        http://www.example.com/MyEventSink/OnStormWarning
24       </wsa:Address>
25       <wsa:ReferenceProperties>
26        <ew:MySubscription>2597</ew:MySubscription>
27       </wsa:ReferenceProperties>
28      </wse:NotifyTo>
29     </wse:Delivery>
30    </wse:Subscribe>
31   </s12:Body>
32  </s12:Envelope>
```

No Código 4.5, abaixo, é exibida uma mensagem de requisição de inscrição que usa o recurso de filtro de mensagem. O filtro é definido na mensagem de inscrição pelas linhas 15-18. A linha 15 indicando o

dialeto do DPWS e as linhas 16-17 indicando as URIs que definem onde as ações na filtragem das Notificações devem ser realizadas.

Código 4.5 – Requisição de Inscrição para Recebimento de Eventos

```
1   <soap:Envelope ... >
2    <soap:Header>
3     ...
4    </soap:Header>
5    <soap:Body>
6     <wse:Subscribe>
7      <wse:Delivery>
8       <wse:NotifyTo>
9        <wsa:Address>
10       urn:uuid:3726983d-02de-4d41-8207-d028ae92ce3d
11       </wsa:Address>
12      </wse:NotifyTo>
13     </wse:Delivery>
14     <wse:Expires>PT10M</wse:Expires>
15     <wse:Filter Dialect="http://schemas.xmlsoap.org/ws/2006/02/devprof/Action">
16      http://printer.example.org/imaging/PrintBasicPortType/JobEndState
17      http://printer.example.org/imaging/PrintBasicPortType/PrinterState
18     </wse:Filter>
19    </wse:Subscribe>
20   </soap:Body>
21  </soap:Envelope>
```

Fonte: Adaptado de (OASIS, 2009d)

## 2.5 SEGURANÇA NO DPWS

Os aspectos de segurança no DPWS são tratados como opcional. São feitas recomendações básicas de mecanismos para a segurança nas comunicações envolvendo dispositivos no modelo de Serviços *Web*. Estas recomendações visam essencialmente a garantia de propriedades como a integridade, confidencialidade e autenticação nas comunicações. Neste perfil, estas recomendações são divididas em duas partes:

(1) Assinatura (opcional) em nível de mensagens no tráfego UDP usado pelo *WS-Discovery*;

(2) Cifragem criptográfica em nível de transporte.

A cifragem em nível de transporte se resume à indicação do uso do TLS/SSL para criar canais seguros entre clientes e dispositivos e seus respectivos contextos.

A outra parte da recomendação tem como alvo o *WS-Discovery* que é peça fundamental para a integração dos componentes nos ambientes de rede. Os protocolos na descoberta dinâmica estão fundamentados na pilha UDP/IP e *IP Multicast* para as comunicações *Unicast* e *Multicast*. Com estas escolhas, o uso do TLS/SSL fica impossibilitado na descoberta de serviços. Para contornar o problema, em suas recomendações, o DPWS propõe o uso de mecanismos de assinaturas, utilizando certificados x.509.v3, em mensagens no nível de aplicação (faz menção ao uso de *Compact Signature Format* definido na *WS-Security*) para garantir a integridade e autenticação das mensagens em *multicast*.

Um dos problemas das recomendações de segurança citadas dá-se que as mesmas desconsideram roteamentos de mensagens SOAP ao indicar o uso do TLS/SSL para criação de canais seguros. A proposta desta dissertação, que será apresentada nos capítulos posteriores, estende o modelo funcional do DPWS, incluindo um Serviço de Segurança que age tratando os aspectos de segurança segundo os padrões definidos pela especificação *WS-Security*.

## 2.5 CONCLUSÃO

Os conceitos abordados neste capítulo revisaram a especificação de *Devices Profile for Web Services* (DPWS), cujo principal objetivo é permitir a utilização Serviços *Web* em dispositivos com restrições computacionais. Como visto, a especificação define um conjunto mínimo de serviços que contemplam a troca de mensagens, procedimento de descoberta, notificação de eventos e, ainda, a descrição de serviços.

Por fim, a especificação DPWS trata os aspectos de segurança como item opcional, sugerindo o uso de SSL na camada de transporte, quando for usado o TCP/IP, e o uso de *Compact Signature Format* para permitir assinatura diretamente nas mensagens. Baseado na condição opcional dos aspectos de segurança, esta dissertação propõe na sequência uma extensão ao DPWS para incluir aspectos de segurança seguindo as recomendações da *WS-Security*.

# 5 EXTENSÃO DO DPWS: A INCLUSÃO DO WS-SECURITY

## 5.1 INTRODUÇÃO

Este capítulo propõe uma extensão de segurança ao DPWS, baseada na especificação *WS-Security*. Esta extensão permite garantir as propriedades de segurança vistas no capítulo de segurança computacional de forma mais adequada aos conceitos de Serviços *Web*. No modelo orientado a Serviços a criação de redes *Overlay* é bastante factível. O roteamento em nível de aplicação se faz presente. Neste caso, as técnicas de segurança descritas no capítulo 3 se fazem necessárias. A segurança fim a fim e as técnicas que enfatizam o uso de controles criptográficos e de acesso em nível de aplicação são fundamentais pela garantia das propriedades de segurança.

Um dos problemas das recomendações de segurança para o DPWS, citadas no capítulo anterior, é que as mesmas desconsideram roteamentos de mensagens SOAP, muito comum através de dispositivos, no seu encaminhamento para o receptor final desejado. Este roteamento de mensagens SOAP é concretizado em nível de protocolos de aplicação. Com isto, o TLS/SSL não é suficiente para garantir a segurança fim a fim das comunicações. A Figura 5.1 ilustra a diferença entre canais de segurança criados com SSL/TLS e usando *WS-Security*. No caso da segurança em nível de transporte os dados ficam expostos em nós intermediários, o que não acontece no caso da *WS-Security*.

Como visto no capítulo 3, a *WS-Security*, principal especificação de segurança para Serviços *Web*, faz uso dos padrões XML-Signature (Bartel *et al*., 2008) e XMLEncryption[11] (IMAMURA *et al.*, 2002) para prover trocas de mensagens seguras. A especificação *WS-Security* é bastante flexível, permitindo o uso de uma grande variedade de mecanismos, algoritmos e técnicas de segurança. As possibilidades ofertadas pela especificação fornecem suporte para diferentes tipos de credenciais de segurança (*security tokens*), múltiplos formatos para assinatura e várias tecnologias de cifragem de dados. Esta multiplicidade de opções é muito importante para alcançar a interoperabilidade entre diferentes tecnologias de segurança.

[11]Como foi visto no capítulo 3, as recomendações XML-Signature XMLEncryption permitem expressar assinaturas digitais e cifragem de dados em formato XML, sendo que os dados assinados e/ou cifrados podem ser ou não documentos XML.

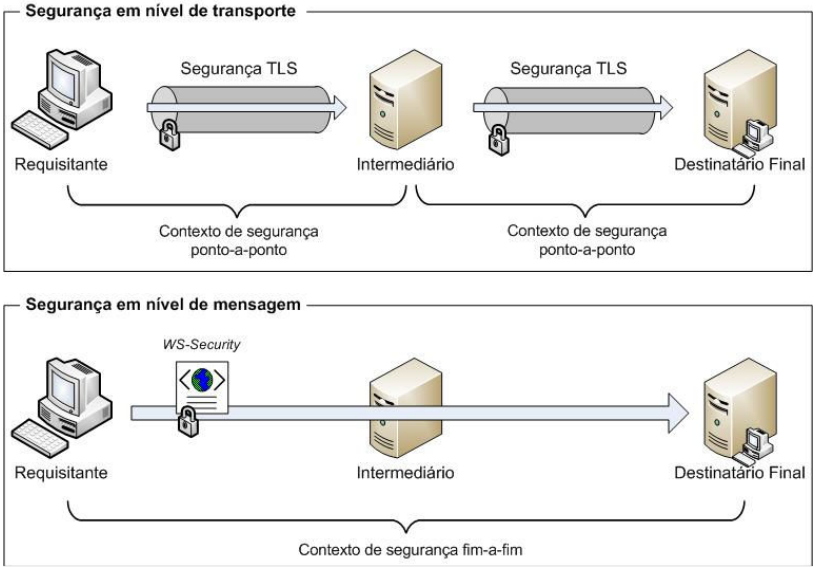


Figura 5.1 – Canais seguros com SSL/TLS e com *WS-Security*
Fonte: Adaptado de (MITCHELL, 2005)

Na literatura, alguns trabalhos defendem de maneira mais determinada o uso do *WS-Security* no modelo do DPWS (HERNÁNDEZ *et al.*, 2009; MARTÍNEZ *et al.*, 2008). Com isto, assumem então as propriedades de confiabilidade, autenticidade e integridade como garantidas em nível de protocolos de aplicação no modelo DPWS (portanto, garantias fim a fim). Este é o caminho tomado nas experiências deste trabalho com o DPWS.

Neste capítulo introduzimos o nosso modelo de segurança que traz para o ambiente de dispositivos embarcados aspectos da *WS-Security*, isto porque, especificação DPWS define somente o uso da *Compact Signature Format*. Este modelo também incorpora mecanismos próprios de dispositivos embarcados. No capítulo, procuramos colocar algumas alternativas que vão além das nossas escolhas.

## 5.2 MODELO DE SEGURANÇA PROPOSTO

A extensão proposta ao DPWS envolve um modelo de segurança que, entre outras coisas, está centrado em um Serviço de Segurança. A Figura 5.2 ilustra a extensão proposta mostrando este serviço atuando de maneira transparente para a aplicação. A atuação deste serviço é baseada na interceptação de mensagens de entrada e de saída do dispositivo, envolvendo somente o processamento dos aspectos de segurança destas mensagens, sempre seguindo os padrões especificados pela *WS-Security*. Esta atuação ocorre em duas portas de entrada do dispositivo: uma porta HTTP e uma porta UDP.

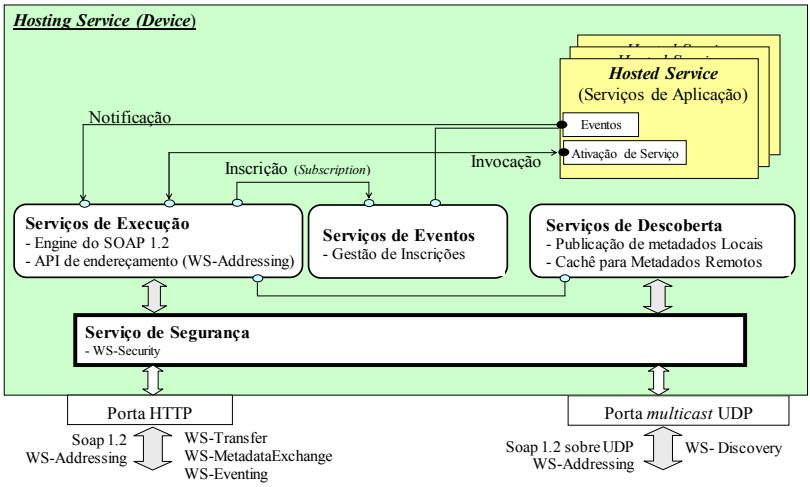


Figura 5.2 – Modelo DPWS estendido

O Serviço de Segurança proposto, portanto, atua de maneira independente dos serviços de aplicação. Este serviço deve atender os vários aspectos de confidencialidade e de integridade nas comunicações, além dos aspectos de gerenciamento destas comunicações seguras. Para tanto, o Serviço de Segurança se apresenta dividido nos seguintes módulos e funcionalidades especiais:

(1) *Cifragens e Assinaturas* (APIs): estes módulos estão disponíveis em forma de bibliotecas e são responsáveis pelas cifragens e decifragens de dados e a assinatura e a verificação das mesmas, respectivamente.

(2) *Contextos de segurança*: para cada canal seguro estabelecido com o dispositivo é criado um contexto de segurança que determina algoritmos, chaves, etc., que devem ser usados no canal correspondente.
(3) *Cache de políticas*: este módulo armazena as políticas (metadados) de interfaces de serviços que o dispositivo tem conhecimento recente. Estes metadados envolvem, entre outras informações, a política de segurança (WS-Policy) de pares comunicantes na rede.
(4) *Localizador de políticas*: este componente é usado na recuperação e armazenamento de políticas. Identifica e armazena no *cache* local as políticas compartilhadas com seus pares conhecidos.
(5) *Processador de políticas*: o módulo faz a comparação entre políticas. No processo de estabelecimento de canal seguro, será criado um contexto de segurança válido quando esta comparação for bem sucedida.
(6) *Motor de segurança*: este componente cifra, decifra, assina e verifica assinaturas, através dos módulos de Cifragens e Assinaturas, em mensagens SOAP, seguindo o WS-Security.
(7) *Gerenciador*: este elemento é o elo entre todos componentes do Serviço de Segurança. É o responsável por tomadas de decisões e respostas de controle aos dispositivos pares. É a partir deste módulo que ocorrem as interceptações necessárias para a implantação dos mecanismos e políticas que atuam sobre as mensagens.

A Figura 5.3 exibe a arquitetura desse serviço na forma de seus módulos e funcionalidades descritos acima.

Dos elementos exibidos na Figura 5.3, o componente *Cifragens e Assinaturas* é implementado para o dispositivo, conforme a necessidade das aplicações desenvolvidas. Os outros elementos não sofrem mudanças na arquitetura. O motivo para a variação das *APIs* de Cifragem e Assinaturas está na diversidade de algoritmos de criptografia existentes. Cada dispositivo poderá ter sua implementação de acordo com sua capacidade computacional e requisitos de segurança. Suas implementações, portanto, variam de acordo com as necessidades e a capacidade do dispositivo.

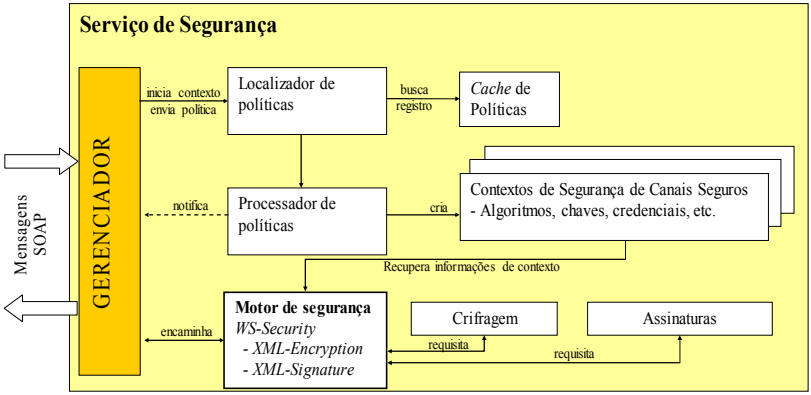


Figura 5.3 – Funcionalidades do Serviço de Segurança proposto

Na sequência descrevemos o modelo nos seus aspectos dinâmicos, tanto no gerenciamento dos canais seguros como no funcionamento dos mesmos nas comunicações das aplicações.

### 5.2.1 Contextos de Segurança no Serviço de Segurança

Como mostrado na Figura 5.2, o Serviço de Segurança está posicionado entre a atual arquitetura de DPWS e sua interface de saída. Isso porque, as mensagens de aplicação que chegam ou saem do dispositivo são interceptadas e processadas por este, segundo as políticas e mecanismos feitos disponíveis nas especificações *WS-Security*.

Mesmo sabendo que estas mensagens passam obrigatoriamente pelo Serviço de Segurança, isto não significa que serão sempre processadas no todo ou em parte por este serviço. As mensagens que estão sujeitas a intervenções deste serviço devem possuir a definição de contextos de segurança neste serviço. Sem estes contextos definidos para canais seguros, a ação do serviço de segurança não se fará presente.

A especificação DPWS define que as informações de alternativas de protocolos suportados para autenticação, confidencialidade e distribuição de chaves nos serviços devam ser anunciadas na forma de asserções de políticas de Qualidade de Proteção (QoP), seguindo a especificação *WS-Policy*. Tais políticas devem estar anexadas às interfaces WSDL correspondentes. No estabelecimento de canais seguros, clientes e dispositivos fazem uso destas políticas para se adequarem mutuamente, permitindo a definição dos contextos de segurança destes canais (escolha de algoritmos, chaves, etc.)

No modelo proposto, o estabelecimento de um contexto de segurança, além de trocas externas que são explicitadas na seção seguinte, envolve trocas internas entre os módulos do Serviço de Segurança exibidos na Figura 5.3. Durante o processo de descoberta de novos dispositivos, o módulo Gerenciador inicia um contexto de segurança encaminhando as políticas para o módulo Localizador de políticas que as armazena em *cache* e encaminha ao módulo Processador de Políticas para a criação do contexto. O processador de Políticas faz isso através da combinação das políticas do próprio dispositivo com as políticas do par com o qual será feita comunicação.

O contexto de segurança resultante do processo deve descrever as opções de algoritmos e mecanismos necessários para comunicações seguras e/ou autenticadas, adequadas às descrições da *WS-Policy* (metadados) das interfaces dos pares que desejam se comunicar. Caso não ocorra no processamento destas políticas uma definição de opções compartilhadas entre os pares considerados, a comunicação então será invalidada. Neste caso, o módulo Gerenciador será notificado desta impossibilidade que por sua vez, notifica ao requisitante da incompatibilidade de mecanismos e algoritmos entre os pares.

O conceito de contextos de segurança para Serviços *Web* é introduzido pela especificação *WS-SecureConversation*, (OASIS, 2009c) que diferentemente da *WS-Security*, não é focada na cifragem (e/ou assinatura) de uma mensagem por vez. Ou seja, a *WS-SecureConversation* define condições para cifragem de uma série de mensagens.

Quando um contexto para um canal já está criado, as mensagens referentes, ao serem interceptadas pelo módulo Gerenciador, são encaminhadas e processadas pelo módulo Motor de Segurança. O processamento do Motor de Segurança ocorre tanto para as mensagens enviadas pelo dispositivo quanto pelas mensagens que chegam a ele.

### 5.2.2 Estabelecimento de Canais Seguros

No modelo proposto de segurança neste texto para o DPWS, apresentamos como são estabelecidos e alterados os contextos de segurança. A especificação *WS-SecureConversation* define três caminhos para se estabelecer um canal seguro e o seu contexto de segurança. Cada contexto é, então, identificado por um *token de segurança* a ser embutido nas trocas de mensagens através do elemento `<wsc:SecurityContextToken>`.

No primeiro modo, o contexto de segurança é criado por um serviço de criação de contextos. Assim, a parte que deseja iniciar uma comunicação faz a solicitação a este serviço, que após criar o contexto de segurança encaminha para as partes que farão parte da comunicação a identificação do mesmo (*token de segurança*). O segundo modo, definido na especificação, indica que a criação do contexto de segurança é realizada pelo próprio iniciador da comunicação e então distribuída pelo mesmo às partes que farão parte desta comunicação.

No nosso caso, adotamos a solução negociada em que parceiros trocam suas políticas e definem suas chaves e algoritmos também através de trocas entre as partes envolvidas. Este é o último modo definido na especificação *WS-SecureConversation* para criação de contextos de segurança. Para definir a chave criptográfica compartilhada do canal seguro, foi assumido o protocolo *Diffie-Hellman*[12] (RFC 2631, 1999) para o estabelecimento de chaves simétricas. Esta escolha está baseada nas características das redes formadas com pequenos dispositivos que são de topologia normalmente variável e auto-organizável, muitas vezes sem hierarquias rígidas. Estabelecer terceiras partes confiáveis para distribuir chaves neste tipo de ambiente é sempre difícil.

A Figura 5.4 ilustra o processo de descoberta, a troca de *metadados* e o estabelecimento de um canal seguro através da criação de um contexto entre um cliente e um dispositivo seguindo as recomendações das especificações DPWS (OASIS, 2009d) e a *WS-SecureConversation*. Na figura 5.4, inicialmente, o cliente envia uma mensagem de descoberta do tipo "*Probe*" ou "*Resolve*" (passo 1). O dispositivo responde com mensagens do tipo "*Probe Match*" ou "*Resolve Match*"[13] (passo 2). Na sequência (passos 3 e 4) a requisição e resposta (*Get* e *GetResponse*), através de *WS-MetadataExchange* e *WS-Transfer,* permitem a troca de *metadados* entre cliente e dispositivo (troca de asserções de políticas *WS-Policy*). Através da mesma especificação, nos passos 5 e 6 são trocadas mensagens para recuperação de *metadados* (*WSDL, WS-Policy etc*) dos serviços hospedados.

[12] O protocolo Diffie-Hellman de troca de chaves permite que duas partes que não se conheçam possam conjuntamente estabelecer uma chave secreta compartilhada sobre um canal de comunicação inseguro.

[13] A mensagem "*Resolve Match*" indica que o dispositivo que a emitiu corresponde ao tipo de serviço solicitado pelo cliente.

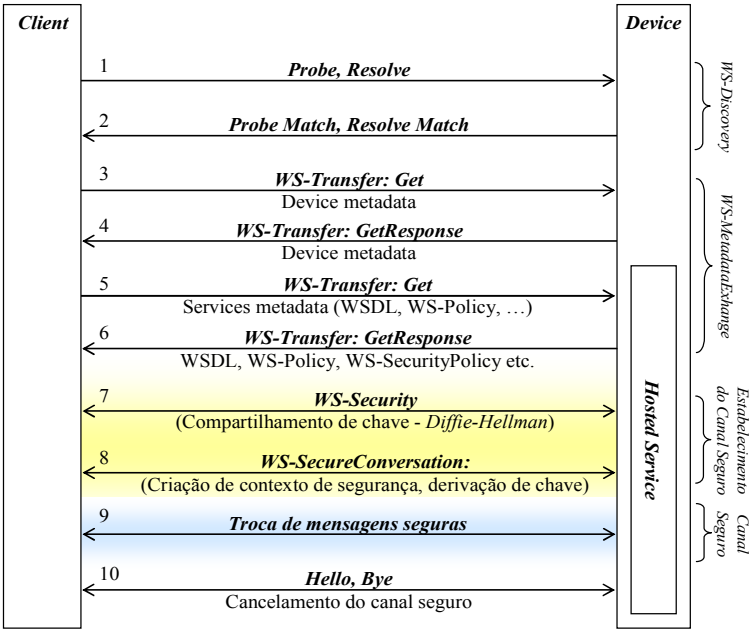


Figura 5.4 – Estabelecimento de Canais Seguros

Até este ponto, as trocas de mensagens são as previstas pelas especificações DPWS. Para estabelecer os canais seguros da nossa extensão de segurança, é necessário que se definam mais trocas. De posse dos *metadados* de um serviço hospedado, o cliente pode verificar se possui os requisitos necessários para o estabelecimento de uma conexão segura e/ou autenticada com o mesmo. Ou seja, neste ponto, localmente o serviço de segurança do iniciador (o cliente) começa a comparação das políticas através do seu Processador de Políticas para ver se atende os requisitos do serviço alvo. O resultado desta comparação deve ser o contexto de segurança para a comunicação entre os pares comunicantes.

No passo 7, já atendendo os requisitos e credenciais solicitadas, o cliente inicia o processo de estabelecimento de uma chave compartilhada com o Serviço Hospedado. Após validar as credenciais do cliente, o Dispositivo inicia as trocas do protocolo *Diffie-Hellman*, criando uma chave de sessão compartilhada com o cliente. O serviço hospedado e o cliente respondem com a criação do contexto de

segurança, próprio para o canal seguro de comunicação estabelecido entre os mesmos. A partir daí, o canal seguro entre estes pares será usado para as trocas de mensagens de aplicação entre os mesmos.

O passo 8 (Figura 5.4) sintetiza as trocas usando o *WS-SecureConversation* para a consolidação do contexto de segurança estabelecido entre os pares e para as mudanças de chaves derivadas usadas nas cifragens/decifragens do canal. A consolidação de um contexto de segurança é representada pelo envio de um *token de segurança* que representa a concretização final do canal seguro entre as partes. A partir deste ponto a troca de mensagens pode ocorrer de maneira segura.

Caso um dispositivo venha deixar a rede ou uma sessão de comunicação, este deve enviar uma mensagem do tipo "*Bye*", que irá remover o canal. Uma mensagem do tipo "*Hello*" quando enviada por um dispositivo ou cliente que já tenham um canal estabelecido também remove um canal seguro antes estabelecido. Isto ocorre caso o dispositivo tenha se desconectado sem enviar a mensagem "*Bye*". O retorno com a mensagem "*Hello*" deve iniciar um novo processo de estabelecimento de canal seguro.

Nestas trocas para o estabelecimento de canais seguro, é importante salientar que um dispositivo poderá suportar diversos protocolos e mecanismos alternativos, sendo assim, deverá listar em ordem de preferência dos mesmos, nos seus *attachments* de política. O dispositivo que responder ao cliente com mensagem de "*Probe Match*" ou "*Resolve Match*", terá suas alternativas para a autenticação e cifragem examinadas a partir da posse do cliente de seus metadados (passos 4 e 6). Após o processo de descoberta, e durante o estabelecimento do canal seguro, no passo 7 descrito acima, cliente e dispositivo invocam procedimento envolvendo as autenticações mútuas, usando protocolos, credenciais e parâmetros negociados entre os mesmos.

## 5.3 ESTRATIFICAÇÃO DE SERVIÇOS DA PROPOSTA EXTENDIDA

Com a inclusão do *WS-Security* e das trocas segundo o *WS-SecureConversation* no suporte para manter em dispositivos o conceito de serviços, fica especificado a estratificação de um conjunto de

serviços e protocolos usados nas interações entre dispositivos mostrada na Figura 5.5.

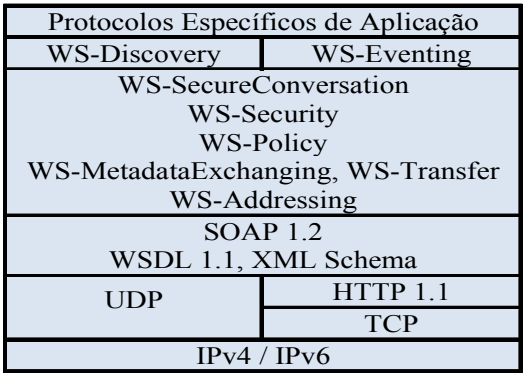


Figura 5.5 – Pilha de protocolos do DPWS extendida

## 5.4 CONCLUSÃO

Este capítulo mostrou nossos esforços em estender a segurança, seguindo os padrões definidos pelas especificações que compõem a *WS-Security*, para cobrir uma área ainda deficiente da especificação DPWS. Isto porque, no DPWS é sugerida a utilização da SSL/TLS para estabelecimento de canais seguros, mas ela não contempla o roteamento em nível de aplicação.

A *WS-Security* e os protocolos que a compõem são as principais especificações acerca de segurança para Serviços *Web*. Fundamentados nisto é que propomos esta extensão. O resultado é um Serviço de Segurança que age independentemente dos Serviços de Aplicação interceptando e processando as mensagens que chegam ou saem dos dispositivos.

Um dos maiores desafios para concretizar este Serviço de Segurança esta no fato do suporte precário de software para desenvolvimento dos sistemas embarcados, uma vez que a maioria destes vem com um conjunto pré-definido de funções ou bibliotecas. Pelo fato da especificação DPWS não enfatizar o uso da *WS-Security*, poucos trabalhos surgiram com foco a suprir essa demanda.

Outra consideração é levar em conta o poder de processamento desses equipamentos, uma vez que possuem recursos reduzidos.

# 6 UMA ABORDAGEM DE MEDIÇÃO DO CONSUMO DE ENERGIA ELÉTRICA USANDO SERVIÇOS *WEB*

## 6.1 INTRODUÇÃO

Os protocolos usados em aquisição de dados eram configurados para operar em linhas dedicadas de banda estreita e, normalmente, o tempo necessário para configuração e mapeamento de dados era bastante significativo. Com o avanço da Internet, muitas das funções destes sistemas vêm sendo transferidas para acessos via a rede mundial. E, neste caso, a questão fundamental passa a ser a segurança destas funções.

Algumas experiências são descritas na literatura sobre o desenvolvimento de sistemas de medição que fazem o uso da Internet para a aquisição remota do consumo de energia elétrica. Estas experiências são baseadas no protocolo SNMP (ALMQVIST e WIKSTROM, 1994) ou numa integração deste com Serviço *Web* (PRAS *et al*., 2004), criando elementos *gateways* que separam as duas tecnologias: o SNMP é usado nos níveis mais baixos (de dispositivos de medição) e Serviços *Web* são usados a nível de Internet permitindo ao sistema ganho em flexibilidade e em facilidade de acesso na rede mundial. As especificações DPWS, ao integrarem os pequenos dispositivos à arquitetura SOA, tornam possíveis aplicações como às citadas acima, sem a necessidade de recorrer à integração de Serviços *Web* com tecnologias como o SNMP.

Este capítulo apresenta uma abordagem de medição cuja aplicação consiste na implantação de um sistema remoto de monitoração e atuação, bem como da medição de consumo final de energia elétrica, formando o que chamamos de um sistema de medição e atuação em redes de distribuição de energia elétrica. A nossa abordagem define uma abordagem por *middleware*, usando DPWS junto com as extensões de segurança introduzidas no capítulo anterior.

A nossa experiência no projeto de um sistema de medição e atuação baseado em dispositivos trata da aquisição de valores de consumo de energia elétrica em medidores residenciais. O acesso a estas medidas é feito através das próprias linhas de distribuição de energia elétrica da concessionária e de recursos de redes metropolitanas sem fio.

O uso da Internet, de redes metropolitanas sem fio e da infraestrutura de distribuição de energia elétrica como meios para

interações se tornam um fator de risco para a segurança das informações e do próprio sistema de medição. Sem mecanismos adequados não é possível garantir a qualidade das mensagens de medição e daquelas que objetivam o controle remoto dos dispositivos. Também, a concessionária ficaria vulnerável a fraudes e a revelação de dados confidenciais de seus clientes. Poucos trabalhos que se propõem ao uso da Internet para a aquisição de medidas se preocupam com a segurança no tráfego das informações correspondentes.

Considerando as características acima, entendemos que sistemas de medição e atuação, sendo usados em ambientes abertos, constituem um excelente *testbed* para explicitar as propriedades do modelo estendido do DPWS, proposto no capítulo anterior. Diante disto, um protótipo de um sistema de medição e atuação para a distribuição de energia elétrica foi desenvolvido. O sentido deste protótipo tinha uma dupla finalidade, a primeira era avaliar as funcionalidades destas extensões do DPWS diante de questões como o desempenho e a segurança. E a segunda verificar as nossas proposições de um sistema de medição que fizesse uso de ambientes abertos na comunicação. Vários testes foram realizados para comprovar a eficiência das escolhas.

Na sequência do capítulo, descrevemos as características das redes de distribuição de energia. Introduzimos também o modelo computacional que deve executar as funcionalidades do sistema de medição. O protótipo e as análises correspondentes são então, por fim tratados neste capítulo.

## 6.2 DESCRIÇÃO DO AMBIENTE DA APLICAÇÃO

O cenário do trabalho se concentra na rede de distribuição de energia elétrica. Os consumidores, classificados pelas concessionárias pelo nome de secundários, são instalações residenciais ou comerciais, que recebem energia elétrica em uma faixa de tensão de 110 ou 220 Volts (V), dependendo da sua região.

Os componentes dessa rede secundária são:

(1) transformadores, responsáveis pela transformação de energia elétrica da tensão de 13.8KV para a tensão de 110 ou 220V, atendendo em média 90 consumidores;

(2) medidores de energia elétrica, que são equipamentos eletromecânicos responsáveis pela medição de consumo individual dos consumidores;

(3) cabeamento elétrico, que transporta a energia elétrica até os transformadores e consumidores.

Atualmente, a área de distribuição de energia elétrica é carente de automação, sendo que as medições, desligamentos e ligamentos são realizados com o deslocamento de funcionários da concessionária até o local do consumidor. Porém, não podemos ignorar as redes de distribuição de energia elétrica que são extensas, envolvendo centenas de milhares de componentes, interligados através dos cabos de energia. Estas redes formam um suporte natural de comunicação, através do uso da tecnologia de PLC (*Power Line Communication*)[14], proporcionando caminhos de alcances significativos no contexto das cidades atuais.

Os medidores residenciais são componentes importantes nestas redes de distribuição. São os responsáveis pelo cálculo do consumo individual de cada cliente da concessionária. Este medidor que antes era do tipo eletromecânico, robusto e de baixo custo, começa a ser substituído por medidores eletrônicos com arquiteturas mais adequadas ao estágio tecnológico atual. Com estas mudanças, os novos medidores possibilitam obter medições envolvendo não apenas a energia consumida, mas as potências instantâneas consumidas pela instalação, tensão e corrente, fator de deslocamento, frequência, entre outros parâmetros. Além disto, estes novos dispositivos apresentam a possibilidade de comunicação direta com a concessionária.

Baseados nas redes PLC e na evolução dos medidores, introduzimos, então, nossa aplicação de um sistema de aquisição e atuação. A idéia geral do contexto em que deve atuar este sistema é apresentada na Figura 6.1. No cenário desta aplicação, medidores de energia elétrica, disponíveis em residências, devem ficar acessíveis via Internet para a Concessionária de Energia Elétrica.

Como mencionado antes, cada transformador atende em média a 90 consumidores, mas devido ao fato desses equipamentos possuírem 3 fases (três saídas), os consumidores ficam então distribuídos em média por grupos de 30 consumidores. A partir dessa informação, na elaboração do nosso sistema de medição, os consumidores foram agrupados por fase do transformador que lhes atende. O acesso aos medidores, que estão nos consumidores, deve usar as próprias linhas de distribuição de energia elétrica que chegam a estes transformadores.

---

[14] PLC (*Power Line Communications*): Tecnologia que utiliza a rede de energia elétrica para transmitir dados e voz

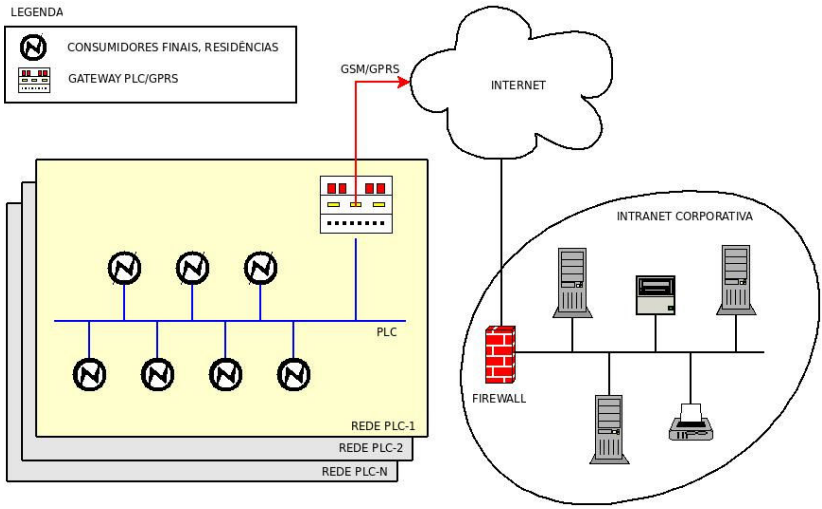


Figura 6.1 – Sistema de Medição e Atuação Remota

Com isso, as redes de medição foram segmentadas, como mostradas na Figura 6.1, e chamadas de REDES PLC. Uma rede PLC no sistema de medição proposto agrupa 30 consumidores. Todas as redes PLC de um transformador terminam em um elemento concentrador, que em nosso sistema é o responsável por agrupar e enviar as informações das redes para a concessionária através da Internet e também pelo roteamento das mensagens entre os consumidores e a concessionária. O concentrador é um PC Industrial[15] que ficará nas proximidades do transformador fixado no cabo de distribuição ou no próprio transformador.

O PLC é a tecnologia usada na comunicação entre os medidores dos consumidores e o concentrador, a partir da infraestrutura já existente de cabos de transmissão de energia elétrica. Os PCs dos concentradores devem possuir um modem para esta tecnologia PLC. A comunicação do concentrador pela Internet se dá através de um modem GSM/GPRS, com a contratação de um serviço mensal de tráfego de dados.

Os medidores, por sua vez, são sistemas microcontrolados (com processadores ARM9 de 200MHz) e estão, através de suas conexões de entrada e saída, ligados aos pontos de acesso da energia elétrica nas

[15] PC Industrial: Classe de computadores desenvolvidos para oferecer maior confiabilidade e suportar ambientes inóspitos

residências dos consumidores. Os medidores têm acesso a uma rede PLC também através de um modem apropriado a esta tecnologia.

## 6.3 ORGANIZAÇÃO FUNCIONAL DO SISTEMA DE MEDIÇÃO PROPOSTO

Tomando como base as especificações de DPWS e de Serviços *Web*, foram definidos os componentes da aplicação que através de seus serviços e métodos implementam o sistema de medição desejado. A Figura 6.2 mostra a distribuição destes componentes e serviços na topologia do sistema e a Figura 6.3 explicita a estratificação em camadas de protocolos dos vários tipos de componentes do sistema. O concentrador implementa duas pilhas: a referente a rede PLC e a da Internet através do GSM/GPRS.

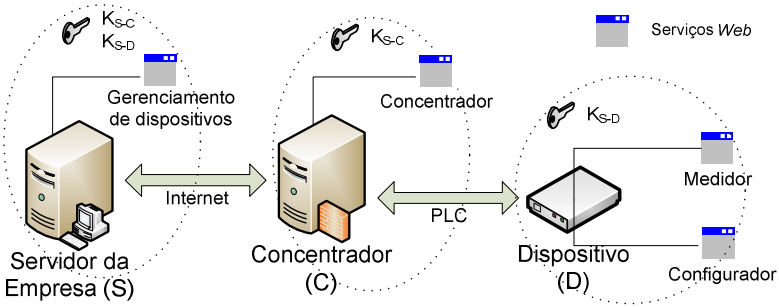


Figura 6.2 – Componentes e Serviços *Web* no Sistema de Medição

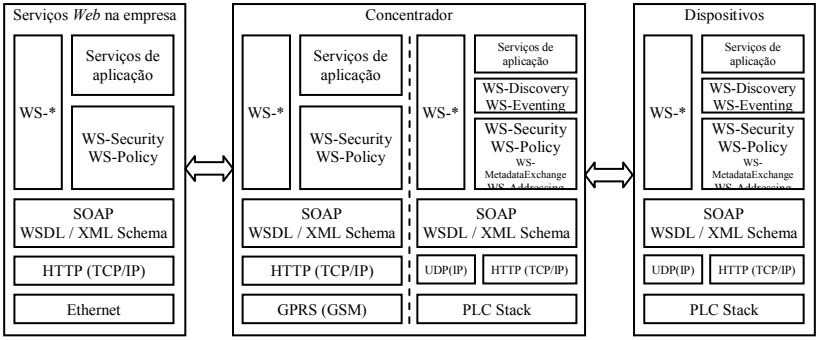


Figura 6.3 – Camadas de Protocolos dos Componentes do Sistema

Para o sistema de medição, foram identificados três tipos de componentes que se caracterizam por Serviços *Web* específicos, seguindo o papel de cada um destes: *dispositivos de medição*, *concentradores* e *servidor da empresa*. Na sequência, são descritas as funcionalidades de cada componente do sistema.

Os dispositivos de medição (ou simplesmente "dispositivos") são componentes localizados nos consumidores e são responsáveis por:

(1) coletar e enviar informações de consumo;
(2) interromper o consumo de energia elétrica;
(3) ativar o consumo de energia elétrica.

Os concentradores, por sua vez, têm acesso via rede PLC a grupos de 30 consumidores e são responsáveis por:

(1) agrupar e encaminhar para a concessionária as informações de consumo dos consumidores atendidos pelo mesmo;
(2) rotear mensagens de controle entre o servidor da empresa e os dispositivos de consumidores.

Por último, o servidor da empresa, é responsável por:

(1) gerenciar a entrada e saída de dispositivos e concentradores da rede;
(2) coletar os resumos de consumo encaminhados pelos concentradores;
(3) ativar ou interromper o consumo de energia elétrica de um consumidor (de uma residência).

A Figura 6.2 mostra o componente dispositivo possuindo dois Serviços *Web*. Um destes, o serviço *Medidor*, implementa as funcionalidades oferecidas por um medidor de energia elétrica instalado em residências. Este serviço é também responsável pelo envio dos dados de consumo do cliente. No envio destes dados, são usados recursos de um serviço de eventos, *WS-Eventing*, serviço embutido do DPWS. As operações disponíveis através do serviço medidor são descritos na Tabela 6.1.

O serviço *Configurador* implementa operações da aplicação responsáveis pela configuração do dispositivo. Através deste serviço ficam disponíveis as operações "*Ativa_Consumo*" e "*Cessa_Consumo*" que concretizam configurações no dispositivo a fim de ativar ou cessar o fornecimento de energia elétrica no consumidor atendido.

A energia elétrica para o funcionamento do dispositivo de medição (alimentação do dispositivo) é independente da energia elétrica fornecida ao consumidor. Portanto, o corte da energia elétrica do consumidor não inviabiliza o funcionamento do dispositivo de medição.

Tabela 6.1 – Operações do Serviço Medidor no Componente Dispositivo

| **Operação** | **Descrição do Serviço Medidor** |
|---|---|
| *Inscrição* | Método que recebe uma solicitação de inscrição, feita por um concentrador, ao serviço de eventos do dispositivo de medição. Como parâmetro de entrada, o sumidouro (do concentrador) precisa informar qual o tempo desejado para a inscrição no serviço de eventos. |
| *Envio_de_Eventos* | Método responsável por encaminhar aos inscritos no serviço de eventos as notificações sobre o consumo de energia elétrica da residência à qual o dispositivo é associado. Este método só é ativado se houver pelo menos um sumidouro inscrito para receber notificações de medição. A fonte de dados deste serviço de eventos é a medição do consumo de energia elétrica, e esta é realizada pelo dispositivo de medição de forma incremental, ou seja, a última mensagem contém o consumo total da residência até aquele momento. As notificações de consumo são compostas do montante de energia consumido até o momento da data e hora da leitura. Estas são remetidas aos inscritos periodicamente, por padrão a cada quinze minutos. |
| *Cancela_Inscrição* | Esta operação é responsável pelo cancelamento de inscrição no serviço de eventos. Apesar das inscrições terem um tempo pré-determinado, os sumidouros podem solicitar o cancelamento para que parem de receber as mensagens de eventos. |

O concentrador, também representado na Figura 6.2, é outro componente da aplicação definido para o sistema de medição. Para atender as especificações da aplicação, o concentrador possui um serviço chamado *Concentrador*. As operações deste serviço estão sintetizadas na Tabela 6.2.

Tabela 6.2 – Operações do Serviço Concentrador

| **Operação** | **Descrição do Serviço Concentrador** |
|---|---|
| *Roteamento* | Este método faz o roteamento (em nível da aplicação) tanto das mensagens que vão do serviço da empresa encaminhadas para os dispositivos, quanto das mensagens que saem dos dispositivos em direção ao Serviço *Web* na empresa. |
| *Inscrição_em_Medições* | Método que realiza a inscrição nos serviços de eventos dos dispositivos de medição. Este método é ativado imediatamente quando é identificado um novo dispositivo de medição na rede PLC. |
| *Recebe_Medições* | Operação que recebe os eventos de medição de consumo vindos dos dispositivos. |
| *Cancela_Inscrição* | Solicita o cancelamento da inscrição nos serviços de eventos dos dispositivos. |
| *Envia_Resumo* | Periodicamente envia dados recebidos dos dispositivos, através do agrupamento dessas informações, para um Serviço *Web* localizado no servidor da empresa concessionária. |

O componente *Servidor da Empresa,* representado na Figura 6.2, suporta o serviço *Gerenciamento de Dispositivos*. Este serviço é responsável pelo gerenciamento dos dispositivos espalhados pela rede de energia elétrica. Este componente age também como centro de distribuição de chaves (CDC) para os algoritmos criptográficos usados na segurança do sistema. As operações deste serviço envolvem aspectos de coleta de dados e de configuração do sistema elétrico (Tabela 6.3).

Tabela 6.3 – Operações do Serviço de Gerenciamento de Dispositivos da Concessionária

| **Operação** | **Descrição do Serviço de Gerenciamento de Dispositivos (no Servidor na Empresa)** |
|---|---|
| *Coleta_Consumo* | Faz a coleta das informações de consumo vindas de todos os Concentradores do sistema. Estas informações vão fazer parte da base de dados da empresa. |
| *Desliga_Dispositivo* | Envia pedido de desligamento remoto do fornecimento de energia elétrica em residências (cancela fornecimento de energia). |
| *Liga_Dispositivo* | É responsável por disparar a solicitação de ligamento remoto das residências. |
| *Registra_Dispositivo* | Inclui na base de dados da operadora um novo dispositivo. Esta operação é necessária antes que o dispositivo seja instalado na residência. A ativação da energia elétrica na residência dependerá deste passo, isto porque, durante este registro de novo dispositivo é gerada uma chave secreta que será incluída no dispositivo e mantida no servidor da concessionária para a comunicação segura entre o dispositivo e o serviço gerenciamento de dispositivos. O mesmo se aplica a concentradores. |
| *Verifica_Dispositivo* | Compara informações (de identificação) obtidas do dispositivo com aquelas inseridas na base de dados do servidor da empresa durante as operações de registro dos dispositivos. |

As operações *Registra_Dispositivo* e *Verifica_Dispositivo*do do serviço Gerenciamento de Dispositivos são importantes para o funcionamento da rede de medição. A primeira destas operações envolve o comparecimento do usuário nas dependências da concessionária para o fornecimento de informações que alimentarão a base de dados da empresa. Estabelecidas as condições de contrato é ativada a operação de *Registra_Dispositivo* que entre outras coisas cria a chave secreta (criptografia simétrica) entre o dispositivo e o Serviço de Gerenciamento de Dispositivos no servidor da empresa. O dispositivo quando sai da empresa para sua instalação na residência possui informações relevantes como seu identificador (*ID*), tipo do equipamento, etc. A operação *Verifica_Dispositivo* busca estas informações nos dispositivos instalados e confronta com suas informações sobre os mesmos na sua base de dados. A primeira "verificação do dispositivo" devidamente registrado deve liberar o seu funcionamento como emissor de medidas no sistema.

Definidos os Serviços *Web* que compõem o modelo computacional do sistema medidor, é importante ressaltar alguns aspectos do modelo de segurança proposto para o DPWS. Como dito acima, cada dispositivo que vai a campo recebe de forma estática sua chave secreta para comunicação com o serviço gerenciamento de dispositivos. No processo de identificação de novos dispositivos são distribuídas as chaves de sessão, também simétricas, para comunicação entre concentradores e dispositivos fazendo uso das chaves secretas (e estáticas) que os concentradores e dispositivos possuem individualmente com o serviço Gerenciamento de Dispositivos. O algoritmo de

criptografia simétrico utilizado é *Advanced Encryption Standard*[16] (AES), com chave e bloco de tamanho 128 *bits*. Na sequência apresentamos os protocolos definidos para a aplicação escolhida.

## 6.4 PROTOCOLOS SEGUROS NO SISTEMA DE MEDIÇÃO

Como dito anteriormente, não seguimos integralmente a proposição de interações do modelo do capítulo 5. Isto ocorre apenas na distribuição de chaves. A aplicação que usamos possui uma hierarquia funcional bem clara do Serviço de *Gerenciamento de Dispositivos*, perdendo, portanto, a característica de uma rede de iguais (pares). Neste caso, como citamos abaixo fica fácil centralizar a distribuição de chaves neste serviço centralizador. No caso do protocolo de descoberta, as outras trocas do modelo, introduzidas na Figura 5.4 (capítulo anterior), permanecem como foram definidas. As mesmas não são mostradas nos diagramas abaixo (Figura 6.4) por motivos de simplificação. Nos restringimos a salientar com mais atenção as modificações referentes a distribuição de chaves.

### 6.4.1 Protocolo de Descoberta e Inclusão de Novos Dispositivos

A primeira troca de mensagens ocorre por meio do *WS-Discovery* que notifica a entrada na rede de novos dispositivos. O diagrama de sequência exibido na Figura 6.4 ilustra o processo que leva a um *Dispositivo* se conectar à rede PLC. Estas trocas iniciam com a notificação, através do *multicast* do passo 1, de que um novo dispositivo está se apresentando na rede. Esta primeira mensagem de *Hello* leva consigo informações básicas sobre o dispositivo, assinadas com a chave estática compartilhada entre este dispositivo e o serviço *Gerenciamento de Dispositivos*. Em seguida, o *Concentrador* inicia o processo de identificação do dispositivo encaminhando uma mensagem de verificação (passo 3), com os dados do dispositivo, ao serviço Gerenciamento de Dispositivos no servidor da empresa.

[16] AES, também conhecido por Rijndael, é uma cifra de bloco adotada como padrão de criptografia pelo governo dos Estados Unidos após concurso para substituir o *Data Encryption Standard* (DES).

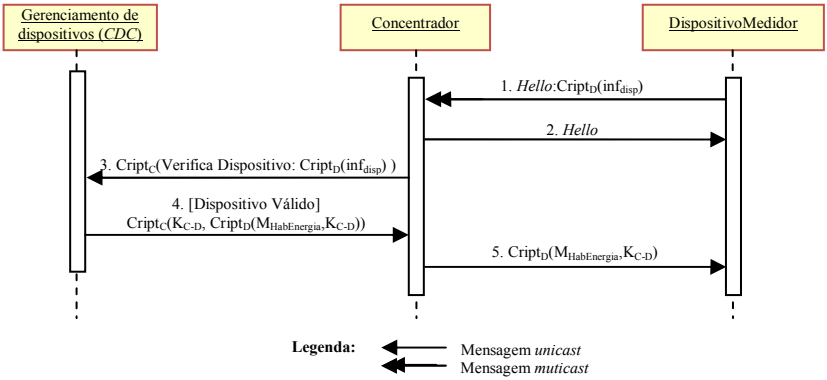


Figura 6.4 – Diagrama de sequência do processo de descoberta de um novo dispositivo

Caso o serviço Gerenciamento de Dispositivos identifique o dispositivo solicitante como válido, irá encaminhar para o concentrador (passo 4) a informação de que o mesmo é válido, uma chave de sessão para comunicação segura entre o concentrador e o dispositivo. A entrega da mensagem do passo 5, enviada pelo serviço de Gerenciamento de Dispositivos com a chave de sessão (via concentrador) ao dispositivo corresponde a liberação do mesmo para o consumo de energia elétrica. Os dados sensíveis do protocolo são cifrados usando as chaves compartilhadas com o serviço Gerenciamento de Dispositivos.

Para estabelecimento de chaves de sessão (ver Figura 5.4, no capítulo anterior) assumimos o uso do protocolo *Diffie-Hellman*. Isto era importante numa rede de pares onde não existia hierarquia. As características do sistema de medição em que estamos usando o DPWS estendido com o modelo de segurança proposto definem outra situação. Nesta aplicação do sistema de medição e atuação, é bem definida uma estrutura hierárquica entre os componentes do sistema distribuído correspondente. O serviço de *Gerenciamento de Dispositivos* é o repositório final de todas as coletas e informações de configuração. Uma vez que este serviço centraliza todas as informações nada mais simples do que dotá-lo, no uso de criptografia simétrica, das funcionalidades de um Centro de Distribuição de Chaves (CDC) para o sistema. As chaves compartilhadas com o CDC pelos Concentradores e Dispositivos de Medição são definidas estaticamente e embutidas nestes equipamentos quando estes saem da empresa. Estas chaves são usadas pelo CDC para

enviar chaves de sessão no processo de identificação dos dispositivos (passos 4 e 5 da Figura 6.4).

### 6.4.2 Protocolo de Inscrição e Notificação de Eventos

Após o processo de identificação e habilitação de consumo de energia elétrica do dispositivo, o concentrador fica responsável pelo agrupamento dos dados de consumo dos dispositivos para, posteriormente, encaminhar ao serviço *Gerenciamento de Dispositivos*, esse processo é exibido na Figura 6.5. O dispositivo fazendo uso da inscrição em seu serviço de eventos (passo 1), encaminha ao concentrador as medições obtidas localmente (passo 2). Estes envios serão periódicos, de acordo com a norma ABNT 14522 a cada 15 minutos para medidores eletrônicos, e devem durar enquanto houver inscrições em seu serviço de eventos de medição. O Concentrador que recebe dados de todos os dispositivos de consumidores, a cada 15 minutos (ou outro intervalo que pode ser configurado), repassa ao serviço Gerenciamento de Dispositivos (passo 3) um agregado destas notificações de consumo recebidas de vários dispositivos.

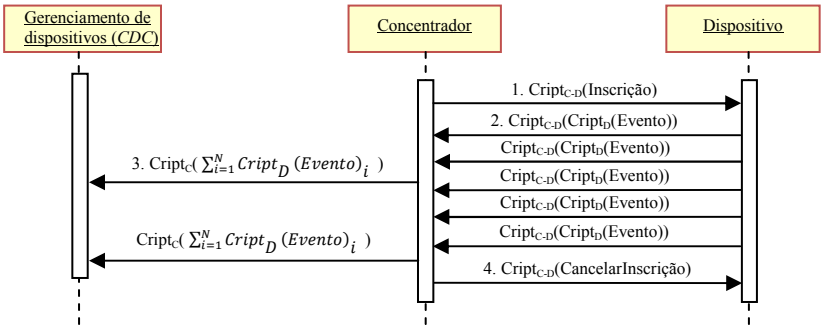


Figura 6.5 – Diagrama de sequência do serviço de eventos de medição

O dispositivo permite inscrições no seu serviço de medições a somente pares que possuem uma chave de sessão compartilhada com o mesmo, garantindo assim que somente componentes identificados via serviço Gerenciamento de Dispositivos terão acesso aos seus eventos. As informações de dados de consumo encaminhadas pelo dispositivo (passo 2, Figura 6.5) são cifradas com a chave secreta compartilhada com o serviço gerenciamento de dispositivos, garantindo assim que somente o destinatário final tenha acesso a estas mensagens. Além disso,

ainda são cifradas com a chave de sessão compartilhada entre o dispositivo e o concentrador, garantindo através de *nonces* e *hashes* a integridade e validade destas para o Concentrador evitando, por exemplo, ataques por mensagens antigas. As partes envolvendo notações e controles de *nonces* e *hashes* não são apresentadas nos protocolos por uma questão de simplicidade na apresentação.

### 6.4.3 Protocolo de Configuração de Dispositivos

Por último, a Figura 6.6 exibe as trocas de mensagens durante a configuração de um dispositivo. No passo 1, o serviço de gerenciamento de dispositivos (na empresa concessionária) encaminha uma mensagem a um dispositivo. O concentrador faz o roteamento da mensagem sem ter acesso ao conteúdo da mensagem (a mesma está cifrada com a chave compartilhada entre o dispositivo e o serviço gerenciamento de dispositivos). Os passos 3 e 4 exibem a resposta ao serviço gerenciamento de dispositivos.

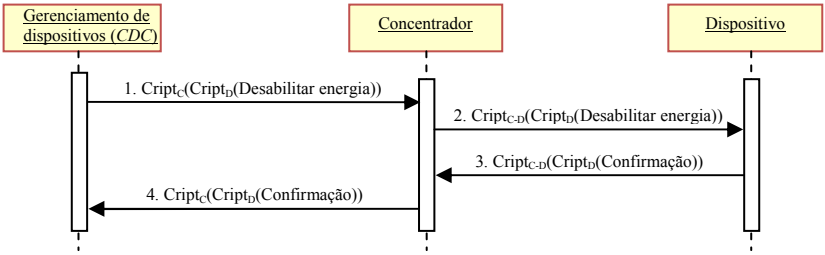


Figura 6.6 – Diagrama de sequência da configuração de um dispositivo

Os mesmos cuidados nas cifragens e verificações de autenticidade descritos para o protocolo da seção anterior são adotados nas trocas do protocolo da Figura 6.6.

## 6.5 IMPLEMENTAÇÃO DO PROTÓTIPO

Um protótipo envolvendo o modelo estendido do DPWS apresentado neste trabalho foi definido e implementado visando comprovar a efetividade do mesmo.

Para compor a camada necessária para o desenvolvimento de aplicações baseadas nas especificações de DPWS, escolheu-se o *framework* SOA4D (https://forge.soa4d.org), que por sua vez, é baseado em outra ferramenta para Serviços *Web* chamada gSOAP (http://www.cs.fsu.edu/~engelen/soap.html). A escolha se deu por dois principais motivos: (i) são ferramentas de código aberto e (ii) são baseadas em linguagem C, facilitando o porte dos desenvolvimentos para dispositivos embarcados. Para compor a camada de qualidade de proteção e prover os aspectos de segurança definidos no modelo, adotou-se como base as bibliotecas fornecidas pelo *xmlsec1* (http://www.aleksey.com/xmlsec) em conjunto a *openssl* (http://www.openssl.org).

Ainda no escopo da camada de qualidade de proteção, o conjunto de bibliotecas *xmlsec1* com *openssl* fornecem os mecanismos de implementação para *XML-Encryption* e *XML-Signature*, mas não tratam os aspectos da especificação *WS-Security* e *WS-SecureConversation*, que nesta abordagem, são implementados diretamente pelo Serviço de Segurança. O gSOAP, que fornece os motores necessários para trocas e interpretações de mensagens SOAP, permite a criação de *plugins*, com trechos de código, que agregam as novas funcionalidades ao suporte e aplicações. Fazendo uso deste recurso, o Serviço de Segurança foi desenvolvido e pode ser incluído em aplicações, independente das trocas, por *handlers* interceptando as mensagens a partir do suporte do gSOAP.

Na configuração da Figura 6.7, que ilustra o cenário utilizado nos testes, um dos computadores é assumido como o Servidor da Empresa (SE, Figura 6.7), dando suporte ao Serviço *Gerenciamento de Dispositivos* e outro caracterizado como Concentrador (*C*) de informações. A comunicação entre estes computadores se dá por intermédio de suas interfaces *Ethernet*. Entre o Concentrador *C* e os dois dispositivos de medição ($D_1$ e $D_2$) é criada uma rede PLC com *modems* Yitran (IT800D) sobre a rede elétrica em laboratório do grupo. Para os dispositivos $D_1$ e $D_2$, foram usados kits de desenvolvimento de microcontroladores modelo TS-7200 fabricados pela empresa *Technologic Systems Inc*. Estes equipamentos são dotados de processadores ARM9 de 200MHz, com memória *flash* interna do tipo STRATA de 8MB, dentre outros periféricos, possui duas portas seriais e uma porta Ethernet 10/100 Mbits. O equipamento é acompanhado de um sistema operacional Linux (Debian), cabos de conexão com PC e ferramentas de software (*Cross Toolchain ARM compiler GCC*) para compilação de programas em Linux.

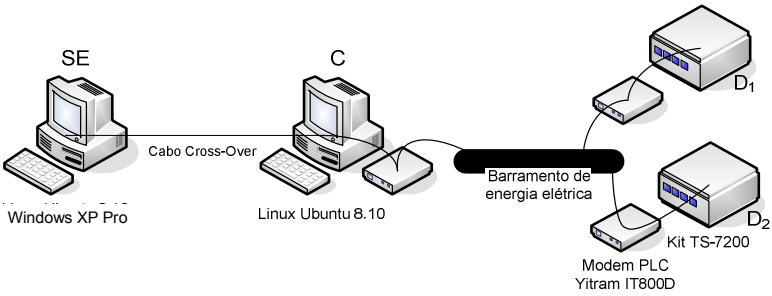


Figura 6.7 – Configuração da rede para os ensaios

O Serviço Gerenciamento de Dispositivos implementado no SE foi programado utilizando a plataforma .NET da Microsoft. Os dispositivos $D_1$ e $D_2$ e o Concentrador *C* foram programados usando o *toolkit SOA4D*, para que incluíssem o suporte completo do DPWS já estendido com o Serviço de Segurança proposto neste trabalho. No Concentrador são implementados um serviço (hospedado) consumidor de eventos, cuja função é concentrar os dados de medição de energia elétrica enviados pelos dispositivos $D_1$ e $D_2$ e um Serviço Concentrador, cujo único método faz o roteamento das mensagens entre dispositivos e o SE. Em $D_1$ e $D_2$, que são dispositivos DPWS, são implementados dois serviços hospedados: *Configurador* e *Medidor*. O primeiro, com dois métodos, simula a liberação e o corte de consumo de energia elétrica no local onde o dispositivo está instalado. O segundo, fonte de dados de consumo, notifica via serviço de eventos o concentrador *C* sobre a atual medição de consumo.

Na concretização do modelo proposto, é feito o uso dos interceptadores (*handlers*), os quais são responsáveis pela captura de mensagens SOAP e pela ativação de algum tipo de processamento ou modificação não funcional na mesma. No caso, é a partir destes *handlers* que são agregadas as propriedades básicas de segurança, tais como integridade, confidencialidade e autenticidade.

Os protocolos de Descoberta e Inclusão de Novos Dispositivos (Figura 6.4), Inscrição e Notificação de Eventos (Figura 6.5) e o de Configuração de Dispositivos (Figura 6.6) foram implementados. Através do primeiro protocolo, novos dispositivos disparam a distribuição de chaves e a criação dos contextos de segurança para as comunicações seguras entre os componentes do sistema. Através do protocolo Configuração de Dispositivos, conseguimos configurar os

dispositivos de medição $D_1$ e $D_2$ para *Ativar* ou *Cessar* a energia elétrica nas residências dos consumidores (operações *Ativa_Consumo* e *Cessa_Consumo*). Em relação ao protocolo de captura de eventos de medição, o concentrador *C* faz a inscrição nos serviços medidores dos dispositivos (concretizado a partir do serviço de eventos DPWS implementado pelo *toolkit SOA4D*). Os dispositivos $D_1$ e $D_2$, ao receber inscrição, enviam suas medições a cada 15 minutos para os inscritos. Neste protocolo o concentrador faz o agrupamento das medições e posteriormente o envio ao Servidor da Empresa, que as armazena na base de dados.

## 6.6 AVALIAÇÃO DA IMPLEMENTAÇÃO E DAS ESCOLHAS DO MODELO ESTENDIDO

### 6.6.1 Desempenho do Protótipo

Nos testes realizados foram verificados principalmente os custos da criptografia e o próprio desempenho da rede PLC, que é de banda estreita. Os testes de desempenho da rede PLC foram realizados em parceria com o INEP/UFSC e indicaram que o modem usado tem banda efetiva mínima de 2,2 Kbps, mesmo com degradação da qualidade da rede elétrica e atenuação para grandes lances de cabo.

O tamanho médio das mensagens SOAP de notificação com dados de medição, enviadas dos dispositivos $D_1$ e $D_2$ para o concentrador, é de aproximadamente 7,2 Kbits. Com o uso do Serviço de Segurança, a inclusão dos cabeçalhos *WS-Security* aumenta o tamanho das mensagens de medição para aproximadamente 24 Kbits.

A ocupação de rede por cada dispositivo pode ser dividido em duas etapas. Na primeira, durante sua entrada na rede, transfere em média 314,4 Kbits de dados. A Tabela 6.4 ilustra o consumo de rede desta etapa, segmentado por protocolos e tipos de mensagens que ocorrem. Na segunda etapa, os dispositivos enviam mensagens periódicas (15 minutos de acordo com a norma ABNT 14522) notificando o consumo de energia elétrica, cada uma com aproximadamente 24 Kbits. Está incluída nestes valores a sobrecarga dos protocolos do TCP/IP.

Os valores da Tabela 6.4 foram obtidos através de um software capturador de pacotes de rede, chamado *Wireshark* (http://www.wireshark.org), e desconsideram retransmissões. O

estabelecimento de chaves e contexto de sessão teve os cálculos baseados nos exemplos disponíveis na especificação *WS-SecureConversation*.

Tabela 6.4 – Custo médio de utilização de rede de um dispositivo

| ***Protocolo*** | ***Tipo mensagem*** | ***Tamanho*** |
|---|---|---|
| *WS-Discovery* | *Hello (~23,2 Kbits), Probe (~20 Kbits), ProbeMatch (~24,8 Kbits), Bye (~20 Kbits)* | **~88 Kbits** |
| *WS-MetadataExchange* | *Metadados do dispositivo (~28 Kbits), WSDL (~56 Kbits), Políticas (~32 Kbits)* | **~116 Kbits** |
| *Serviço de Segurança* | *WS-SecureConversation: Estabelecimento de chave de sessão* | **~56 Kbits** |
| *WS-Aplicação* | *Libera_Consumo* | **~27,2 Kbits** |
| *WS-Eventing* | *Inscrição* | **~27,2 Kbits** |
| | Total | **~314,4 Kbits** |

Considerando as medidas dos tamanhos de mensagens citadas e a taxa de transmissão obtida para o pior caso na rede PLC, a entrada de cada dispositivo na rede, com segurança, levaria 2min20seg. O envio das mensagens periódicas de notificação de consumo (~24 Kbits) leva em média 11 segundos para serem entregues. Mesmo num cenário onde existam 30 dispositivos em uma rede PLC com todos enviando notificações de medidas simultaneamente a cada 15 minutos, teríamos um total de 720 Kbits para transmitir todas estas informações. Isto ocuparia o canal de comunicação por aproximadamente 5min30seg. Estes números dão a garantia que o uso do DPWS estendido sobre uma rede PLC de banda estreita é compatível com a norma NBR14522, que regulamenta o envio eletrônico de medições de consumo.

O pior caso seria com todos os dispositivos entrando no sistema ao mesmo tempo. Se assim acontecer, a quantidade de dados a transmitir estaria em aproximadamente 9,6 Mbits, que na taxa de 2,2 Kbps levaria até 1h13min para que todos os dispositivos estejam notificando medições. Existe a possibilidade de suprimir o TCP/IP e o HTTP, utilizando o SOAP diretamente sobre PLC, com isso, a redução no tamanho das mensagens que utilizam UDP seria de aproximadamente 1,8 Kbits e as que utilizam TCP+HTTP seria de 8 Kbits. Ao invés dos 314,4 Kbits da primeira etapa teríamos em torno de 224 Kbits, o que reduzia em 38 segundos a etapa de entrada do dispositivo no sistema. Nas mensagens de notificação de consumo, para os 30 dispositivos, a redução seria de 1min54seg.

### 6.6.2 Avaliando as Escolhas do Modelo Estendido

As proposições do modelo estendido do DPWS procuram prover meios para suprir as necessidades de segurança nas trocas de dispositivos em geral que ficam disponíveis via SOA a aplicações na Internet ou em sistemas abertos como a rede PLC. As trocas de mensagens feitas pelo Serviço de Segurança foram implantadas seguindo os padrões de segurança *WS-Security* e *WS-SecureConversation*, propostos para a tecnologia Serviços *Web*. Estes padrões aportaram funcionalidades que garantem a segurança fim a fim em ambiente onde existe o roteamento em nível de aplicação.

Os componentes da arquitetura, sempre que entram na rede, estão de posse de uma chave simétrica (estática) compartilhada com o serviço de gerenciamento de dispositivos, conforme descrito na seção 6.4.1. Em seus processos de identificação na rede, estes componentes usam a chave compartilhada e o algoritmo de criptografia AES[17] para receber chaves complementares (as chaves de sessão, também simétricas) para as comunicações com seus pares. Através do uso das operações de cifragem do *XML-Encryption* e das chaves citadas, a confidencialidade das informações emitidas é garantida fim a fim. O concentrador, por exemplo, lida com mensagens de vários dispositivos sem que haja possibilidade da quebra da confidencialidade destas no mesmo.

Os aspectos de integridade e de validade são também garantidos com o uso do *XML-Encryption*, *hashes* e *nonces*. Os *hashes* usados na verdade são MACs (*Message Authentication Code*), calculados com a função HMAC-SHA1 que é disponível em biblioteca do *xmlsec1*. A validade é garantida por *nonces* e a posse das chaves simétricas somente pelos pares comunicantes. Ataques de *replay*, por exemplo, são facilmente detectados com estes mecanismos citados.

Nas proposições deste trabalho, outro aspecto a ser considerado é o uso de chaves estáticas para a comunicação com o serviço de gerenciamento de dispositivos no servidor da concessionária. Estas chaves até poderiam ser trocadas periodicamente ou ainda, usado algum esquema de chaves derivadas para diminuir a exposição das mesmas. Porém considerando a finalidade da aplicação (sistema de medição), acreditamos que os custos relacionados para diminuir a exposição destas chaves não compensam pela baixa probabilidade de revelação das mesmas. Por fim, a escolha da criptografia simétrica neste trabalho é

[17] http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf

justificável principalmente pelas restrições computacionais dos dispositivos embarcados.

Considerando o cenário de restrição computacional de dispositivos embarcados, no qual este trabalho está inserido, foram realizados testes como forma de verificar a viabilidade de uso do protótipo com e sem o uso do Serviço de Segurança. A Figura 6.8 exibe os tempos obtidos com o protótipo.

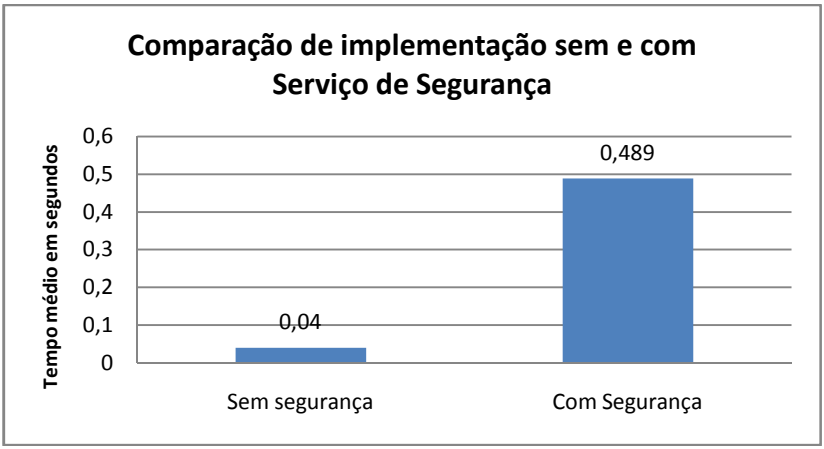


Figura 6.8 – Tempo gasto pelo protótipo no processamento com e sem segurança

Cada resultado foi obtido como uma média de 200 execuções do programa no passo de envio de uma mensagem de notificação do dispositivo para o concentrador, com kit de desenvolvimento TS-7200. Com o uso da biblioteca <*sys/resource.h*> é possível medir o tempo em segundos que um processo levou para ser executado. É possível notar um aumento de mais de 10 vezes no tempo de processamento quando se usa o Serviço de Segurança, mas deve-se considerar que o tempo sequer chegou a meio segundo. Por isso, ainda sim o uso se mostra viável.

As proposições do modelo estendido do DPWS procuram prover meios para suprir as necessidades de segurança nas trocas de dispositivos em geral que ficam disponíveis via SOA a aplicações na Internet.

Em relação ao sistema de medição, considerando a disposição dos componentes nos circuitos de distribuição de energia elétrica, a disponibilidade de seus componentes torna difícil qualquer premissa

quando o ambiente é aberto como o que assumimos para o sistema de medição. Garantir essa propriedade envolve inúmeros fatores como, por exemplo, a ocorrência de interferências externas que objetivam os ataques de negação de serviço. Nesta classe de problemas estão os roubos de componentes do sistema. O roubo de concentradores causaria um enorme impacto sobre as redes de medição. Não é possível nestes ambientes abertos, apresentar medidas que evitem ou minimizem todas as ações maliciosas possíveis no sentido de tornar o sistema indisponível.

Mas para algumas destas ações existem medidas que podem ser tomadas. Por exemplo, no caso de roubo de concentrador, este componente ao ser retirado do cabo de rede de distribuição emite uma sinalização através de mensagens SMS indicando que sua alimentação foi cortada. Isto é possível, pois o mesmo possui uma bateria interna que o mantém em funcionamento na falta de eletricidade da rede de energia elétrica. Em outras situações, valem as análises de risco que avaliam os custos das medidas em relação à probabilidade da ocorrência das ações maliciosas consideradas.

Outra consideração que podemos fazer é a possibilidade de violação dos medidores (*tampering*) pelos usuários numa tentativa de alterar o funcionamento dos instrumentos, tentando levar alguma vantagem. As medidas que podem ser tomadas para este tipo de ação maliciosa se assemelham àquelas do roubo de concentradores. Ou seja, a abertura do lacre do dispositivo pode ativar um sinal indicando a ação. Neste caso, os registros de consumo normal do consumidor desonesto podem também indicar a sua investida no instrumento.

## 6.7 TRABALHOS RELACIONADOS

Diversos trabalhos na literatura abordam a utilização de Serviços *Web* em dispositivos embarcados, poucos detalham ou falam dos aspectos de segurança.

Algumas iniciativas, como em (CASTRO *et al.*, 2001), utilizam uma infraestrutura *ad-oc* para realizar a monitoração remota de dispositivos, sem uso de padrões como, por exemplo, o DPWS. A arquitetura proposta pelos autores é composta por três principais componentes: um instrumento de monitoração de qualidade de energia elétrica, uma unidade de monitoramento e uma central de supervisão.

O instrumento de monitoração é um hardware baseado no microcontrolador 68HC11 e seu *firmware* é responsável por monitorar, em tempo real, a qualidade da energia elétrica no consumidor, notificando a unidade de monitoramento caso as medidas estejam fora de parâmetros pré-configurados. A unidade de monitoramento, por sua vez, encaminha para a central de supervisão os indicadores recebidos do instrumento de medição. O arranjo destes componentes pode ser visto na Figura 6.9.

A comunicação entre o instrumento de medição e a unidade de monitoramento é realizada via cabo serial RS-485. Entre a unidade de monitoramento e a central de supervisão a comunicação ocorre através de um modem *dial-up* sobre a linha telefônica.

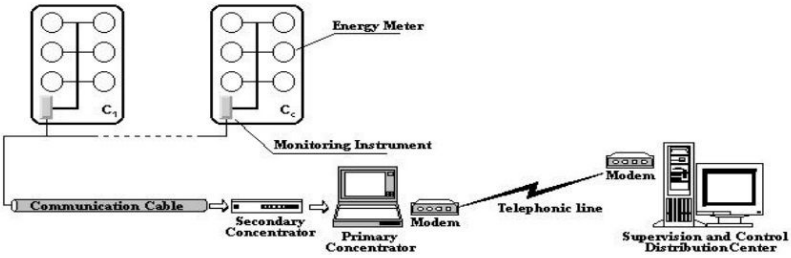


Figura 6.9 – Instrumento de medição da qualidade de energia elétrica para consumidores residenciais

Segundo os próprios autores, os dados recuperados pelo instrumento de monitoração são importantes para cômputo de indicadores de continuidade de acordo com os requisitos impostos pela agência reguladora de energia elétrica brasileira (ANEEL). Além disso, eles indicam que outros parâmetros podem ser monitorados como, por exemplo, dados para identificação de harmônicas.

Em outro trabalho que se utiliza da grande difusão do SNMP, os autores investigam e testam este protocolo no gerenciamento de dispositivos. Almqvist e Wikstrom (1994) procuram mostrar a eficiência do uso do SNMP no gerenciamento de equipamentos da área de energia elétrica. Os autores apresentam uma arquitetura do sistema de gerenciamento baseado em SNMP que é exibido na Figura 6.10. A mesma é dividida em quatro camadas:

(1) Funções de apresentação (*Presentation Functions*): fornece a interface gráfica ao usuário com janelas, mapas e os equipamentos gerenciados.
(2) Funções: operações disponíveis nos equipamentos gerenciados, tais como: gerenciamento de falha, de desempenho, de configuração e de segurança.
(3) Base de dados (MIB - *Management Information Base*): mantém a estrutura da informação do sistema. Contém informação de diferentes tipos de equipamentos bem como dados estatísticos e instantâneos.
(4) Acesso (*Mediation/Access*): representa como as informações chegam ao sistema de gerenciamento. Para equipamentos não compatíveis, funções de mediação são ativadas como agentes do SNMP que traduzem dos protocolos proprietários para o padrão SNMP.

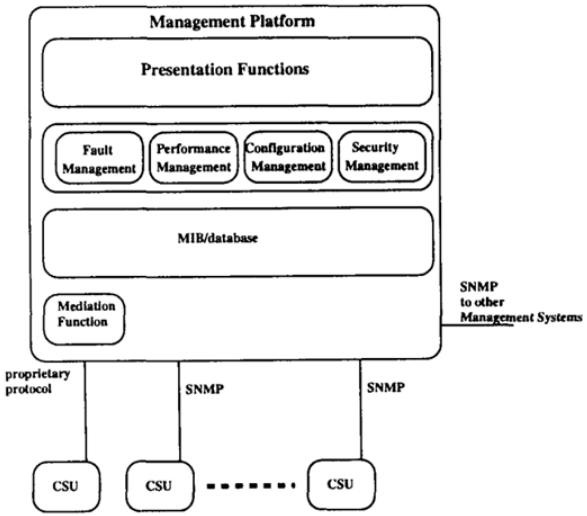


Figura 6.10 – Arquitetura do sistema de gerenciamento de energia baseado em SNMP
Fonte: (ALMQVIST e WIKSTROM, 1994)

A característica principal do protocolo é a sua simplicidade, mas no próprio artigo são feitas as ressalvas sobre a utilização deste protocolo e as mais significativas são a falta de mecanismos de segurança adequados, a entrega de notificações e alarmes não confiável

(pelo uso do UDP) e a característica estática da estrutura de armazenamento de variáveis remotas (MIB - *Management Information Base*).

Os problemas de segurança apresentados pelos autores foram sanados em especificações posteriores do SNMP (RFC 3410, 2002; RFC 3411, 2002; RFC 3414, 2002). Em sua última versão, SNMPv3 (RFC 3410, 2002), o protocolo inclui um modelo de segurança baseado em usuários que propõe mecanismos para proteção contra modificação da informação, mascaramento (*masquerade*), modificação no fluxo da mensagem e revelação (*disclosure*).

Aiko Pras *et al* (2004) fazem uma comparação de performance entre o SNMP e Serviços *Web* como protocolos de gerenciamento. Para isso, usam uma implementação de SNMP desenvolvida em linguagem C chamada Net-SNMP e desenvolveram um Serviço *Web* usando o *toolkit* gSOAP (também em linguagem C) com funcionalidades equivalentes às da implementação do Net-SNMP. São analisados aspectos de tráfego de dados, consumo de memória, tempo de processamento e *round trip delay*. Conforme exibido nos gráficos da Figura 6.10, o SNMP é mais eficiente quando a requisição é feita para poucos objetos e nos aspectos de codificação e compressão das mensagens. Quando a quantidade de objetos recuperados aumenta, Serviços *Web* se tornam mais efetivos devido às possibilidades de agregação e de roteamento em nível de aplicação deste último.

Por fim, em relação ao uso de memória, a tabela ilustrada na Figura 6.11 mostra vantagem no uso de Serviços *Web* em relação à implementação do SNMP. Entretanto, os autores sugerem que os dados desta tabela não devem ser superestimados, uma vez que refletem somente a comparação entre a implementação do Net-SNMP e de um protótipo usando o *toolkit* gSOAP.

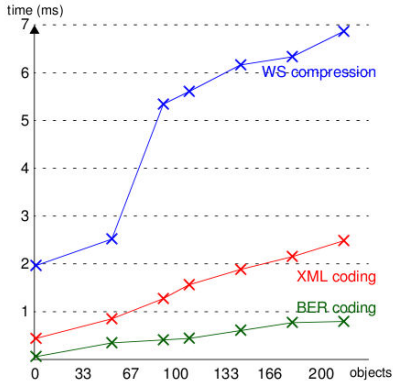


Fig. 13. CPU time for coding and compression

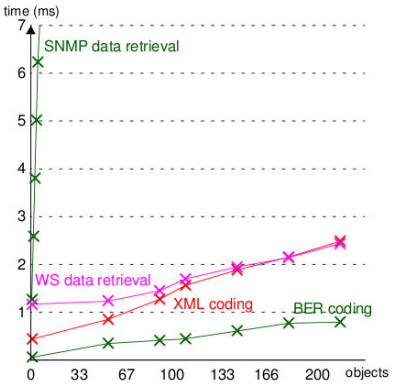


Fig. 14. CPU time for coding and data retrieval

TABLE I
MEMORY REQUIREMENTS

| | | data | |
|---|---|---|---|
| | instructions | static | dynamic |
| SNMP | 1972 kB | 129 kB | 70-160 kB |
| Web services | 580 kB | 470 B | 4 kB |

Figura 6.11 – Tabela e gráficos comparativos de desempenho entre SNMP e Serviços *Web*
Fonte: (PRAS, A. *et al.,* 2004)

Em relação ao uso de dispositivos embarcados com Serviços *Web*, desde o surgimento do primeiro *Draft* da especificação DPWS, este vem sendo estudado por diversos grupos de pesquisa, objetivando principalmente aplicações práticas do mesmo. Um dos primeiros trabalhos que abordam estas especificações em dispositivos embarcados veio de uma iniciativa européia, através do projeto SIRENA[18], que tem objetivo de desenvolver uma infraestrutura de serviços para aplicações em redes de dispositivos embarcados. Neste projeto a arquitetura SOA e as especificações DPWS são usadas para interligar dispositivos embarcados entre quatro áreas distintas de aplicações: a industrial, telecomunicações, automotiva e de automação residencial.

Outros dois projetos tomaram como base os estudos e experiências do projeto SIRENA. São neste caso o projeto SODA[19] (*Service Oriented Device & Delivery Architecture*) que tinha como objetivo a criação de uma plataforma baseada em SOA, abrangente,

[18] http://www.sirena-itea.org/
[19] http://www.soda-itea.org/

escalável e fácil de implantar para dispositivos de baixo custo e o SOCRADES[20] que investiga a aplicação do DPWS para um grande número de dispositivos objetivando a integração do chão de fábrica com uma infraestrutura orientada a serviços ligada às atividades da camada de negócios de uma organização. Dos resultados alcançados com o projeto SOCRADES, em (KARNOUSKOS e TARIQ, 2009) é demonstrado através de simulações a escalabilidade do DPWS. Com auxílio de uma plataforma de simulação de agentes, são criados milhares de dispositivos DPWS e testados os processos de descoberta dinâmica e as trocas de mensagens entre eles. Outro ponto importante ressaltado é que a adoção de SOA nos dispositivos e equipamentos que estão nos níveis de chão de fábrica permite a integração direta dos mesmos aos níveis gerenciais.

Em (MARTÍNEZ *et al.*, 2008) os autores apresentam uma proposta de extensão para a existente especificação de segurança em DPWS. Desenvolvida no âmbito do projeto SODA, essas extensões seriam disponibilizadas através de bibliotecas para os desenvolvedores de serviço e clientes no ambiente DPWS. O modelo apresentado é mais ambicioso que o apresentado por nós neste texto, pois apresenta módulos de "Autenticação Local" e "Gerenciadores de Autorização", etc. No nosso modelo, nós nos abstemos destes serviços e mecanismos porque entendemos que os controles nos níveis mais baixos devem ser mais simples. No seu subsistema de segurança os autores também estão prevendo o uso do *WS-Security* para proteções considerando as possibilidades de roteamentos em nível de aplicação e a necessidade de garantias fim a fim. Este subsistema, também como o nosso, é incluído na forma de um serviço embutido. Porém, na base necessária para estabelecer contexto de segurança e estabelecimento de chaves, não são usados protocolos padronizados como o *WS-SecureConversation*, por exemplo. A Figura 6.12 ilustra a interação entre os componentes da arquitetura para autorização e autenticação usando o DPWS.

Além de servidores de autenticação e autorização os autores incluem no subsistema de segurança um módulo de autenticação e autorização local, cujo objetivo é validar e gerar credenciais, e/ou garantir ou negar acesso baseado em informações locais. O módulo pode ser usado quando não for possível a comunicação com tais servidores.

[20] http://www.socrades.eu/

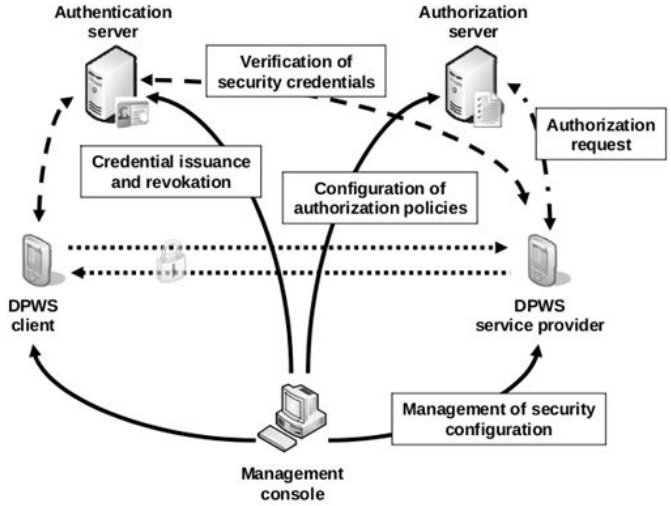


Figura 6.12 – Interação para autorização e autenticação na solução de segurança SODA
Fonte: (MARTÍNEZ *et al.*, 2008)

## 6.8 CONCLUSÃO

Neste capítulo descrevemos a maior parte de nossos esforços durante os nossos estudos de mestrado. A extensão do modelo do DPWS para a inclusão do WS-Security e de trocas baseadas na *WS-SecureConversation* é relativamente simples. Mas para mostrar a eficiência de nossas propostas seria necessário encontrar aplicações que envolvessem um contexto desafiador.

Optamos por um Sistema de Medição e Atuação para uso na distribuição de energia elétrica. A nossa escolha desta aplicação se deu porque as concessionárias começam a se preocupar com a automação na aquisição e atuação dos medidores que já começam a comportar certa evolução em suas funcionalidades. A literatura sobre estes sistemas de medição que fazem uso da Internet também está se mostrando bastante profícua.

Definimos então as funcionalidades do nosso modelo para o sistema de medição e começamos a elaborar o nosso protótipo. O capítulo então apresenta discussões sobre a concepção deste protótipo e

os testes realizados para verificar a exiguidade do mesmo. Analisamos ainda no capítulo os aspectos de segurança para o nosso sistema de medição. Por fim, apresentamos a literatura relacionada tanto para sistemas de aquisição nos mesmos termos do proposto nesta dissertação, como para algumas experiências de extensão e testes das especificações DPWS.

# 7 CONCLUSÕES

Esta dissertação apresentou os estudos sobre um sistema de controle e medição de dispositivos através da Internet, cujo cenário envolve medidores residenciais de energia elétrica e sua comunicação com um centro de controle na concessionária de energia elétrica. Encontramos na Arquitetura Orientada a Serviços, através de Serviços *Web*, o *middleware* que pôde suportar os requisitos de integração.

Este cenário, cuja existência de dispositivos com capacidade de processamento e memória limitados, nos orientou ao uso das especificações de DPWS, que como resumido neste trabalho tem por objetivo levar exatamente a estes tipos de dispositivos a Arquitetura Orientada a Serviços através de Serviços *Web*. Porém, as especificações DPWS nos aspectos de segurança se resumem ao uso de SSL/TLS para transporte seguro das mensagens. Entretanto, o cenário que este trabalho aborda caracteriza o uso de roteamento em nível de mensagem e o uso de SSL/TLS não fornece a segurança fim a fim desejada.

Por isso, nesta dissertação, definimos e implementamos uma extensão ao perfil DPWS através de um Serviço de Segurança com o intuito de prover segurança fim a fim a este sistema de controle e medição de dispositivos através da Internet. Como forma de manter a interoperabilidade a extensão segue as especificações da *WS-Security* e *WS-SecureConversation*.

Outro fator que tivemos de considerar no cenário abordado por esta dissertação foi o uso da própria infraestrutura de cabos de energia elétrica como meio de comunicação dos dispositivos medidores de energia elétrica através do uso da tecnologia *Power Line Communication* (PLC) de banda estreita. Ainda assim, foi possível provar a viabilidade de seu uso.

As experiências na bibliografia com este tipo de aplicação, envolvendo o acesso a dispositivos via Internet, normalmente envolvem a integração de ferramentas outras com Serviços *Web*. A nossa solução não envolve integração de ferramentas atuando em diferentes níveis da aplicação. Integramos dispositivos via Internet em aplicações orientadas a serviço.

## 7.1 REVISÃO DOS OBJETIVOS

Nesta seção são revisados os objetivos apresentados na seção 1.3, indicando como o modelo estendido ao DPWS satisfaz tais objetivos.

O primeiro objetivo desta dissertação consistiu no estudo de segurança para aplicação em sistemas de controle e medição em ambientes abertos como a Internet. Assumiu-se que a solução proposta deveria fazer uso de padrões amplamente aceitos como forma de garantir a integração entre aplicações. Sendo assim, nesta dissertação foi definido um modelo estendido do DPWS como forma de garantir segurança fim a fim em dispositivos com restrições computacionais.

O modelo estendido foi proposto como um Serviço de Segurança trabalhando de maneira independente dos serviços de aplicação, ou seja, o mesmo trata, através de *handlers*, as mensagens de entrada e saída nos dispositivos. Este serviço de segurança é composto de módulos que foram explicados na seção 5.2, em especial os módulos de Cifragem e Assinatura que são implementados conforme as restrições computacionais dos dispositivos.

Além do objetivo geral, os objetivos específicos desta dissertação foram compostos do desdobramento do objetivo geral e dos requisitos que motivaram o tema de dispositivos embarcados no sistema de distribuição de energia elétrica abordados nesta dissertação. A seguir os objetivos específicos são relembrados e como este trabalho os atendeu.

- **Desenvolvimento de protótipo de software para o modelo estendido.** Diante da inexistência de uma solução que englobasse os requisitos para o modelo estendido apresentado nesta dissertação, foi desenvolvido um protótipo usando como base a ferramenta de código aberto **SOA4D**, que implementa o perfil DPWS. Para os módulos de Cifragem e Assinatura foi usada a ferramenta ***xmlsec1***, que implementa o *XML-Encryption* e o *XML-Signature*. Como o xmlsec1 não trata os aspectos da especificação *WS-Security* e *WS-SecureConversation*, os mesmos são implementados diretamente pelo Serviço de Segurança. Os documentos que definem as asserções de políticas de Qualidade de Proteção para cada dispositivo (*WS-Policy*) foram escritos manualmente.
- **Software para o cenário do setor de distribuição de energia elétrica englobando a tecnologia de PLC (*Power Line Communication*) de banda estreita.** Usando como base

a implementação do modelo estendido, buscou-se a definição de protocolos que pudessem, de maneira segura, fazer a monitoração e medição da distribuição de energia elétrica. Para isso, além dos serviços e métodos relativos ao cenário de aplicação, foram definidos e implementados protocolos que puderam garantir a distribuição de chaves de sessão. Ainda neste cenário o uso de PLC de banda estreita acabou se provando viável como explicado na análise da sessão 6.6.

- **Os custos com SOA para dispositivos embarcados e o uso do modelo de segurança; e avaliação das escolhas feitas neste trabalho.** A viabilidade de uso da arquitetura orientada a serviços com dispositivos embarcados é confirmada na literatura por projetos que tomam esse rumo (gSOAP, SOA4D, WS4D, WS4J, etc). Após desenvolvimento e implementação do Serviço de Segurança e da aplicação considerando o cenário de distribuição de energia elétrica, foram realizados diversos ensaios com o kit de desenvolvimento que tínhamos disponível.

Dos nossos testes, o que ficou de positivo foi a verificação da possibilidade de usar redes PLC com a tecnologia de Serviços *Web*. Sobre o sistema proposto acreditamos que muitas questões podem ser levantadas. Como, por exemplo, por que usar PLC no lugar de colocar um GPRS em cada residência? O que diríamos em defesa da nossa solução é que além das vantagens de custo, estaríamos perdendo o uso da maior rede já instalada que são as de distribuição de energia elétrica.

A contribuição definitiva de nossas proposições é o aporte de mecanismos de segurança onde proposições da literatura pouco fazem menção sobre os aspectos envolvendo a segurança em sistemas de medição com informações transmitidas em ambientes abertos e pela Internet.

Durante o desenvolvimento deste trabalho foi produzido o seguinte documento científico:

- MENDONÇA, Igor Thiago Marques; FRAGA, Joni da Silva ; DIAS, Roberto Alexandre. Extensão de Segurança para o Perfil DPWS. In: Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais – SBSeg, 2010, Fortaleza. X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. Fortaleza: SBSEG, 2010. v. 1. p. 133-146.

Como trabalhos futuros sugerem-se:

- A instalação e testes dos protótipos em um ambiente real de medição, uma vez que os mesmos tiveram seus ensaios realizados somente em bancada de laboratório.
- A implementação de alguns componentes do modelo estendido, que nos ensaios foram simulados estaticamente, são eles: o localizador, o *cache* e o processador de política.
- Tendo em vista as restrições da rede PLC, seria interessante adequar o gSOAP para atuar diretamente sobre o PLC, o que representaria diminuir a ocupação de banda devido ao TCP.

Para concluir, diríamos que nossas expectativas sobre o trabalho foram perfeitamente preenchidas. Esperamos que estes ensaios possam indicar a viabilidade da automação do sistema de aquisição de medidas de consumo de energia elétrica a partir das linhas de distribuição. Acreditamos também que a nossa solução é perfeitamente escalável, podendo envolver milhares de medidores residenciais.

# REFERÊNCIAS

ALMQVIST, L e WIKSTROM, R. **Standardizing energy management by using simple network management protocol.** Telecommunications Energy Conference - INTELEC '94. 16th International, 1994.

AMOROSO, E. **Fundamentals of Computer Security Technology**. Prentice Hall, Englewood Cliffs, NJ, 1994.

ANDERSON, J. P. **Computer security technology planning study**. Relatório técnico, ESD-TR-73-51, 1972.

ARAÚJO, G. M. de. **Uma Infraestrutura para Integração entre Dispositivos Computacionais Heterogêneos Baseada na Especificação DPWS**. Florianópolis, SC – Brasil, 2009, 95 páginas. Dissertação (Mestrado em Ciência da Computação) – Programa de pós-graduação em Ciências da Computação. Universidade Federal de Santa Catarina.

BARTEL, M., BOYER, J., FOX, B., LaMACCHIA, B., SIMON, E. **XML-Signature Syntax and Processing (Second Edition)**. W3C, 2008. Disponível em: <http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/>

BOYER, J. **Canonical XML**. W3C, 2001. Disponível em: <http://www.w3.org/TR/xml-c14n>.

BRINKLEY, D. L.; SCHELL, R. R. **Concepts and terminology for computer security**. In M.D. Abrams, S. Jajodia, and H.J. Podell, (ed.), Information Security: An Integrated Collection of Essays, pp. 98-110. IEEE Computer Society Press, Los Alamitos, CA, 1995.

CARMODY, S. **Shibboleth Overview and Requirements**. Shibboleth Working Group, 2001.

CASTRO, A. L. S. et al. **Power Quality Monitoring Instrument for Energy Distribution Feeder**, 11th IMEKO TC-4 Symposium Trends in Electrical Measurement and Instrumentation and 6th EuroWorkshop on ADC modelling and testing, Lisbon - PORTUGAL, September 13-14, 2001.

CERAMI, E. **Web Services Essential: Distribuited Applications with XML-RPC, SOAP, UDDI & WSDL**. Ed. 1, Sebastopol: O'REILLY, 2002.

CHINNICI, R., MOREAU, J-J., RYMAN, A., WEERAWARANA, S. **Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language**, 2007. Disponível em: http://www.w3.org/TR/2007/REC-wsdl20-20070626/. Acessado em: 20 ago. 2009.

CLEMENT, L., HATELY, A., von RIEGEN, C., ROGERS. T. **UDDI Version 3.0.2** - UDDI Spec Technical Committee Draft (19 de outubro de 2004), 2004. Disponível em: <http://uddi.org/pubs/uddi-v3.0.2-20041019.htm>. Acesso em: 20 ago. 2009.

COULOURIS G., DOLLIMORE J. e KINDBERG T. **Distributed Systems: Concepts and Design**. 4th edition. Addison Wesley/Pearson Education, 2005.

de MELLO, E. R. **Um modelo para confiança dinâmica em ambientes orientados a serviço**. Tese (doutorado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2009.

de MELLO, E. R., WANGHAM, M. S., FRAGA, J. da S., CAMARGO, E. **Segurança em Serviços Web**. Minicurso SBSeg 2006. Santos, SP, 2006.

DEMCHENKO, Y., GOMMANS, L., deLAAT, C. e OUDENAARDE, B. **Web services and grid security vulnerabilities and threats analysis and model**, 2005. Disponível em: <http://www.uazone.org/demch/analytic/draft-grid-security-incident-04.pdf>. Acesso em: ago. 2009.

Department of Defense. **DoD 5200.28**-STD: Department of Defense (DoD) Trusted Computer System Evaluation Criteria (TCSEC), 1985.

DIERKS, T., ALLEN, C. **The TLS Protocol, Version 1.0**, http://www.ietf.org/rfc/rfc2246.txt, IETF RFC 2246, Janeiro 1999.

EASTLAKE, D., JONES, P. **US Secure Hash Algorithm 1 (SHA1)**. Internet Engineering Task Force RFC 3174, 2001.

ELLISON, C. M. **SPKI Certificate Documentation**, 1998. Disponível em: http://www.clark.net/pub/cme/html/spki.html

ELLISON, C. M. **RFC 2692: SPKI requirements**. The Internet Society, 1999. Disponível em: ftp://ftp.isi.edu/in-notes/rfc2692.txt

FIATKOOSKI, S. H. P. **Orquestração de Web Services**. Monografia (Especialização em Redes de Computadores e Comunicação de Dados). Universidade Estadual de Londrina. Londrina, PR - Brasil, 72 p., 2004.

FREIER, A. O., KARLTON, P., eKOCHER, P.C. **The SSL protocol - v.3**. InternetDraft, 1996.

GOGUEN, J. A. e MESAJUER, J. **Security Policies And Security Models**. Proceedings of IEEE symposium on Reseach in Security and Privacy, 1982.

HALLAM-BAKER, P., MYSORE, S. H. **XML Key Management Specification (XKMS 2.0)**. W3C – Proposed Recommendation, 2005.

HERNÁNDEZ, V., LÓPEZ, L., PRIETO, O., MARTÍNEZ, J. F., GARCÍA, A. B. e DA-SILVA., A. **Security Framework for DPWS Compliant Devices**. In: Emerging Security Information, Systems and Technologies. SECURWARE '09. Third International Conference, 2009.

HOUSLEY, R., POLK, W., FORD, W., SOLO, D. **Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile**. IETF RFC 3280, 2002.

IMAMURA, T., DILLAWAY, B., e SIMON, E. **XML Encryption Syntax and Processing**. W3C, 2002. Disponível em: <http://www.w3.org/TR/xmlenc-core/>.

JAMMES, F., MENSCH, A. e SMIT, H. **Service-Oriented Device Communications Using the Devices Profile for Web Services**. In: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, páginas: 1-8, Grenoble, France, 2005.

JOSHI, J. B. D.; AREF, W. G.; GHAFOOR, A.; SPAFFORD, E. H. **Security models for web-based applications**. Communications of the ACM, v. 44, n. 2, p. 38-44, fev. 2001.

KARNOUSKOS, S. e TARIQ, M. M. J. **Using Multi-Agent Systems to Simulate Dynamic Infrastructures Populated with Large Numbers of Web Service Enabled Devices**. In: The 9th International Symposium on Autonomous Decentralized Systems – ISADS, Athens, Greece, 2009.

KENT, S. T. **Encryption-based protection for interactive user/computer communication**. In Proceedings of the fifth symposium

on Data communications (SIGCOMM '77). ACM, New York, NY, USA, pp. 5.7-5.13, 1977.

KILGORE, R. A. **Simulation Web services with .net technologies**. Em Proceedings of the Winter Simulation Conference, v. 1, ISBN: 0-7803-7614-5, p. 841 – 846, 8 – 11 dez. 2002.

LANDWEHR, C.E. **Formal models for computer security**. ACM Computing Surveys, Vol. 13, No.3, pp. 247–278, 1981.

LANDWEHR, C. E. **Best available technologies for computer security**. IEEE Computer Vol. 16, No. 7, pág. 86-100, 1983.

MARKHAM, T. **Internet security protocol**. Dr. Dobb's Journal of Software Tools, Vol. 22, No.6, pp.70-75, 1997.

MARTÍNEZ, J. F., LÓPEZ, M., HERNÁNDEZ, V., JEAN-MARIE, K., GARCÍA, A. B., LÓPEZ, L., HERRERA, C. e SÁNCHEZ-ALARCOS, C. J. **A security architectural approach for DPWS-based devices**. In: CollECTeR Ibéroamérica 2008 conference. Madrid, Spain, 2008.

MENSCH, A., ROUGE, S. **DPWS Core version 2.1 – User Guide** (14 de abril de 2009), 2009. Disponível em: <https://forge.soa4d.org/docman/view.php/8/45/DPWSCore+User+Guide.pdf>. Acesso em: 01 ago. 2009.

MITCHELL, B. **Why WSE?**. MSDN Library, 2005. Disponível em: <http://msdn.microsoft.com/en-us/library/ms996935.aspx>.

MITRA, N., LAFON, Y. **SOAP Version 1.2 Part 0: Primer (Second Edition)** (27 de abril de 2007), 2007. Disponível em: <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/>. Acesso em: 19 ago. 2009.

NEUMAN, B. C., Ts'o, T. **Kerberos: An authentication service for computer networks**. IEEE Communications Magazine, Vol. 32, No. 9, pp. 33-38, 1994.

OASIS (2004a). **WS-Security: SOAP Message Security 1.0**. Organization for the Advancement of Structured Information Standards (OASIS), 2004.

OASIS (2004b). **WS-Security: SOAP Message Security 1.1.** Organization for the Advancement of Structured Information Standards (OASIS), 2004.

OASIS (2005a). **eXtensible Access Control Markup Language (XACML) version 2.0**. Organization for the Advancement of Structured Information Standards (OASIS), 2005. Disponível em: <http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf>.

OASIS (2005b). **SAML Executive Overview**. Organization for the Advancement of Structured Information Standards (OASIS), 2005.

OASIS (2005c). **Security Assertion Markup Language (SAML) 2.0 Technical Overview**. Organization for the Advancement of Structured Information Standards (OASIS), 2005.

OASIS (2009a). **WS-SecurityPolicy 1.3**. Organization for the Advancement of Structured Information Standards (OASIS), 2009.

OASIS (2009b). **WS-Trust 1.4**. Organization for the Advancement of Structured Information Standards (OASIS), 2009.

OASIS (2009c). **WS-SecureConversation 1.4**. Organization for the Advancement of Structured Information Standards (OASIS), 2009.

OASIS (2009d). **Devices Profile for Web Services Version 1.1 (DPWS)**. Organization for the Advancement of Structured Information Standards (OASIS) , 2009.

OASIS (2009e). **Web Services Dynamic Discovery (WS-Discovery)**. Organization for the Advancement of Structured Information Standards (OASIS) , 2009.

OASIS (2009f). **SOAP-over-UDP Version 1.1**. Organization for the Advancement of Structured Information Standards (OASIS) , 2009.

PAPAZOGLOU, M. P. **Service-oriented computing: Concepts, characteristics and directions**. In Fourth International Conference on Web Information Systems Engineering (WISE'03). Vol.0, pp. 3-12, 2003.

PRAS, A., DREVERS, T., van de MEENT, R. e QUARTEL. D. (2004) **Comparing the Performance of SNMP and Web Services-Based Management**, In: ETRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, FALL 2004. IEEE 2004.

RECKZIEGEL, M. **Descrevendo um Web Service - WSDL** (27 de julho de 2006), 2006. Disponível em: <http://imasters.uol.com.br/artigo/4422/Webservices/descrevendo_um_Web_service_-_wsdl/>. Acesso em: 20 ago. 2009.
RFC 2631 (1999). Diffie-Hellman Key Agreement Method. E. Rescorla Junho 1999.

RFC 2631. **Diffie-Hellman Key Agreement Method**. Jun, 1999.

RFC 3410. **Introduction and Applicability Statements for Internet Standard Management Framework**. Dez, 2002.

RFC 3411. **An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks**. Dez, 2002.

RFC 3414. **User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)**. Dez, 2002.

RSA. **PCKS#1 v2.1: RSA Cryptography Standard**. RSA Laboratories, 2002. Disponível em: <ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf>.

SANDHU, R., SAMARATI, P. **Authentication, Access Control, and Audit**. ACM Computing Surveys, Vol. 28, No. 1. 1996.

SANTIN, A. O. **Teias de Federações: Uma Abordagem Baseada em Cadeias de Confiança para Autenticação, Autorização e Navegação em Sistemas de Larga Escala**. Tese (doutorado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2004.

Secure Electronic Transaction LLC. **SET secure electronic transaction specification – version 1.0**, 1997.

SHIRLEY, R. **Internet Security Glossary**. IETF, RFC 4949, 2007.

STALLINGS, W. **Network Security Essentials: Applications and Standards**. Prentice-Hall, Englewood Cliffs, New Jersey, 2000.

W3C (2006a**). Web Services Transfer (WS-Transfer)**. W3C, 2006. Disponível em: <http://www.w3.org/Submission/WS-Transfer/>.

W3C (2006b). **Web Services Addressing 1.0 - Core**. W3C, 2006. Disponível em: <http://www.w3.org/TR/ws-addr-core/>.

W3C (2007a). **Web Services Policy 1.5 – Framework**. W3C, 2006. Disponível em: <http://www.w3.org/TR/ws-policy/>.

W3C (2007b). **Web Services Policy 1.5 – Attachment**. W3C, 2006. Disponível em: <http://www.w3.org/TR/ws-policy-attach/>.

W3C (2010a). **Web Services Metadata Exchange (WS-MetadataExchange)**. W3C, 2010. Disponível em: <http://www.w3.org/TR/ws-metadata-exchange/>.

W3C (2010b). **Web Services Eventing (WS-Eventing)**. W3C, 2010. Disponível em: <http://www.w3.org/TR/ws-eventing/>.

WESTBRIDGE. **Securing and Managing XML Web Services – Guide to XML Web Services Security**. Westbridge Technology Inc, 2003.

ZDUN, U., HENTRICH, C., van der AALST, W, **A survey of patterns for service-oriented architectures**, International Journal of Internet Protocol Technology, vol. 1, pp. 132–143, 2006.

ZIMMERMAN, P. **PGP User's Guide**. Massachusetts Institute of Technology, 1994.