UNIVERSIDADE DE BRASÍLIA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

ARQUITETURA DE REFERÊNCIA ORIENTADA A SERVIÇOS COM FOCO EM SEGURANÇA

ROGÉRIO ALVES DA CONCEIÇÃO

Dissertação apresentada para obtenção do título de Mestre em Computação Aplicada do Curso de Pós-Graduação em Computação Aplicada em Engenharia de Software da Universidade de Brasília.

Orientador: Dr. Rodrigo Bonifácio de Almeida

Co-orientadora: Dra. Edna Dias Canedo

BRASÍLIA

2014

RESUMO

A DITEC, Divisão de Tecnologia da Polícia Civil do Distrito Federal, tem como responsabilidade estratégica o desenvolvimento dos softwares da instituição, muitas vezes apresentando necessidades de integração e compartilhamento de informações sensíveis com órgãos conveniados. Dada a criticidade desses sistemas e informações compartilhadas, preocupações relacionadas a segurança devem ser tratadas sob uma perspectiva arquitetural dentro da instituição, que atualmente adota diferentes alternativas de integração, desde *Web Services* até a replicação das bases de dados para instituições parceiras. O objetivo desse trabalho é propor uma arquitetura orientada a serviços para ser adotada como alternativa única de integração, balanceando os requisitos de segurança com outros atributos de qualidade; em particular o tempo de processamento das requisições.

Palavras-chave: Interoperabilidade, Segurança em arquiteturas orientadas a serviço, Web Services, Arquitetura de referência em SOA

ABSTRACT

DITEC, Technology Division Civil Police of the Federal District, is responsible for the software's institution strategic development, often presenting needs of integration and sharing of sensitive information with government insured. Given the criticality of these systems and shared information, security concerns should be treated under an architectural perspective within the institution, which currently adopts different integration alternatives, from emph Web Services to the replication of databases for partner institutions. The aim of this paper is to propose a service-oriented architecture to be adopted as an alternative single integration, balancing security requirements with other quality attributes, in particular the processing time of the requests.

Keywords: Interoperability, Security in service-oriented architectures, Web Services, SOA Reference Architecture

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

SUMÁRIO

1 INTRODUÇÃO

As atribuições da Polícia Civil do Distrito Federal, no que diz respeito à sua competência de Polícia Judiciária, tangenciam em vários pontos as atribuições do Ministério Público do Distrito Federal e Territórios, do Tribunal de Justiça do Distrito Federal e Territórios e da Defensoria Pública do Distrito Federal. De forma que a competência de cada um desses Órgãos, por apresentarem pontos que se complementam, demanda intensa troca de informações.

A Polícia Civil do Distrito Federal, por meio de sua Divisão de Tecnologia, tem como propósito desenvolver seus próprios softwares. Esta atividade permite uma vantagem estratégica para a instituição, uma vez que a torna detentora dos softwares desenvolvidos evitando dessa forma a dependência tecnológica e administrativa de empresas privadas.

Nesse sentido, tem-se buscado estudar técnicas de desenvolvimento de software que promovam de forma efetiva a integração dos sistemas internos com os sistemas de órgãos parceiros e que necessitem consumir de forma segura os dados e informações oriundos dos sistemas legados da Polícia Civil do Distrito Federal.

1.1 PROBLEMA DA PESQUISA

Nesse contexto, um dos principais desafios encontrados na DITEC refere-se à necessidade de integração e compartilhamento de informações de maneira segura, observando que as aplicações foram desenvolvidas em diferentes linguagens de programação e a integração ocorre com diferentes órgãos conveniados, tais como: Tribunal de Justiça do Distrito Federal e Território (TJDFT), Ministério Público da União (MPU), Departamento de Trânsito do DF (DETRAN-DF), Secretária de Segurança Pública do Distrito Federal (SSP-DF), Secretárias de Justiça do DF e estados.

A ocorrência de uma vulnerabilidade de confidencialidade, por exemplo, ocorrendo o vazamento de informações sensíveis, criminosos poderiam utilizar essas informações e comprometer de forma significativa uma investigação policial.

Outra preocupação está relacionada à autenticidade, uma vez que todos os acessos a informações no âmbito da Polícia Civil o Distrito Federal devem ser realizados somente por pessoal autorizado. Caso isso não seja observado, pessoas podem se valer do anonimato e divulgar dados sigilosos de forma criminosa, o que também acarretaria inúmeros problemas de ordem jurídica para a instituição.

Dessa forma, devido à importância dessas informações, elas devem ter um tratamento diferenciado com relação a segurança nos aspectos de confidencialidade, autenticidade, integralidade e disponibilidade.

Por outro lado, na maioria das vezes, são disponibilizadas técnicas não seguras de integração, como a replicação ou o acesso direto a base de dados, apesar de existirem algumas iniciativas de integração baseadas em *Web Services*.

No intuito de possibilitar que os sistemas possam ser integrados de forma eficiente e principalmente segura com outros sistemas, a Divisão de Tecnologia busca desenvolver uma metodologia própria que possa melhorar o processo integração de software no âmbito da Polícia Civil do Distrito Federal. Para isso, optou-se pela utilização da Arquitetura Orientada a Serviços (SOA), que é um modelo arquitetural que propõem o uso de um conjunto de padrões para disponibilizar, descrever, publicar e invocar serviços. Neste cenário, este trabalho propõe-se a investigar as seguintes questões de pesquisa:

1. Quais são os principais problemas de segurança encontrados na adoção da Arquitetura Orientada a Serviços - (SOA)?
2. Quais padrões para construção de software seguro em arquiteturas SOA podem ser empregados pela Divisão de Tecnologia da Polícia Civil do Distrito Federal para realizar efetivamente a integração de seus sistemas com os sistemas dos órgãos parceiros?

1.2 JUSTIFICATIVA

Uma vez que a Polícia Civil do Distrito Federal desenvolva produtos de software mais seguros, que auxiliem no trabalho investigativo, ela realizará seu trabalho de uma forma mais efetiva, influenciando diretamente no combate da criminalidade e beneficiando a comunidade em geral e todos os órgãos distritais e federais tais como: Secretaria de Segurança Pública do Distrito Federal, Tribunal de Justiça do Distrito Federal, Ministério da Justiça, Secretarias de Governo Distritais, dentre outros órgãos, que necessitem das informações da instituição para realizar qualquer tipo de integração de software.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

Avaliar e aplicar o uso de técnicas, ferramentas e procedimentos que garantam os requisitos de segurança em uma arquitetura orientada a serviços a ser usada para integrar os sistemas e automatizar os processos entre órgão parceiros (TJDFT, MPU, DETRAN, SSP).

1.3.2 OBJETIVOS ESPECÍFICOS

a) Realizar um mapeamento sistemático da literatura para compreender o estado da arte e da prática de segurança em SOA;

b) Identificar e avaliar quais são os principais problemas de segurança encontrados na adoção da Arquitetura Orientada a Serviços (SOA);

c) Estudar as especificações de Web Services relacionados a segurança e selecionar padrões e ferramentas para garantir confidencialidade, autenticidade e integridade nas integrações da arquitetura orientada a serviços. Essa seleção deve considerar o impacto na disponibilidade e no tempo de resposta dos serviços;

d) Estabelecer uma arquitetura de referência na construção de software seguro em SOA que possam ser empregados pela Divisão de Tecnologia da Polícia Civil do Distrito Federal para realizar efetivamente a integração de seus sistemas com os sistemas dos órgãos parceiros.

1.4 ORGANIZAÇÃO DO TRABALHO

Revisar Este trabalho está organizado em cinco capítulos. No capítulo 2 é realizada uma revisão da literatura onde são abordados os conceitos gerais sobre da Arquitetura Orientada a Serviços (SOA), Web Services, segurança e vulnerabilidades em SOA. No capítulo 3 consta a metodologia que será empregada para a realização da dissertação. No capítulo 4 são apresentados os resultados preliminares, envolvendo um mapeamento sistemático e os resultados obtidos com a sua realização. Finalmente, no capítulo 5 é descrito um cronograma de execução de tarefas.

2 MAPEAMENTO SISTEMÁTICO

2.1 INTRODUÇÃO

A utilização da SOA tornou-se uma solução para a problemática da interoperabilidade entre sistemas, uma vez que permitiu a integração automatizada de negócios entre aplicações. Contudo, apesar dos benefícios, existem vários desafios que devem ser considerados quando da implantação de uma arquitetura orientada a serviços (MARKS; BELL, 2006). Dentre eles destaca-se o requisito não funcional de segurança, que deve ser muito bem planejado de forma que o serviço oferecido não seja vulnerável a ameaças que comprometam sua integridade, confidencialidade, disponibilidade e autenticidade. Dessa forma, é de suma importância conhecer e aplicar os mecanismos de segurança em uma Arquitetura Orientada a Serviços.

Com a intenção de identificar trabalhos primários relevantes e reconhecidos, com vistas a aumentar o conhecimento da aplicabilidade de mecanismos de segurança a arquitetura orientada a serviços, fez-se necessário realizar um mapeamento sistemático que identificasse as principais pesquisas que envolvessem o tema segurança em SOA.

2.2 MAPEAMENTO SISTEMÁTICO

Essa seção descreve os resultados do mapeamento sistemático conduzido, sendo estruturada conforme as diretrizes discutidas em (PETERSEN et al., 2008).

2.3 PROTOCOLO DE ESTUDO

Para a realização do mapeamento sistemático foi definido um protocolo de estudo que contemplasse as questões de pesquisa, a string de busca e os critérios de exclusão e inclusão de artigos. A finalidade deste processo é a de documentar as etapas do mapeamento além de permitir a sua replicação por outros pesquisadores.

2.4 QUESTÕES DE PESQUISA (QP)

Com a finalidade de identificar as principais contribuições e estudos sobre "Segurança e SOA " a pesquisa toma a vertente mais específica com as questões abaixo:

QP1) Quais são os principais veículos que publicam artigos nessa área? busca-se verificar quais são os principais veículos que publicam informações sobre segurança em SOA, de forma que seja possível categorizar e verificar onde foram publicadas as contribuições mais significativas.

QP2) Qual o tipo de contribuição é a mais proposta nas pesquisas realizadas? Neste caso, procura-se verificar quais são as principais contribuições de pesquisa, identificando se o tipo da contribuição é uma proposta, solução, avaliação, validação de algum estudo ou um artigo de opinião.

QP3) Quais atributos de segurança são mais abordados nos estudos? Objetiva-se identificar dentre os atributos privacidade, confidencialidade, autenticidade e disponibilidade, quais são os mais abordados nos estudos categorizados e classificados como solução. Além disso, quando uma solução não for classificada em nenhum desses atributos ele deverá ser classificado como qualidade de serviço. Neste caso, serão classificados os artigos que não se enquadrarem em nenhum dos atributos anteriores, mas que mesmo assim, vise garantir a qualidade de outros atributos que no contexto de segurança em SOA seja relevante. Por exemplo, desempenho e escalabilidade.

QP4) Quais contribuições classificadas como solução e categorizadas como: método, técnica ou ferramenta/arquitetura, foram os mais propostos nas pesquisas? Objetiva-se identificar dentre os artigos selecionados e categorizados como Solução quais delas podem ser classificados e quantificados como método, técnica ou ferramenta/arquitetura.

2.5 ESTRATÉGIA DE BUSCA

É oportuno definir uma estratégia de busca para a pesquisa dos estudos primários, sendo necessário determinar as palavras chave a serem pesquisadas e também onde as buscas serão realizadas (KITCHENHAM; CHARTERS, 2007). A estratégia de busca adotada consistiu objetivamente na busca eletrônica das seguintes bibliotecas digitais:

a) ACM Digital Library referente aos seguintes periódicos:

(a) ACM Transactions on Information and System Security (TISSEC);

(b) ACM Computing Surveys (CSUR).

b) IEEE Xplore, apenas por periódicos e

c) DBLP Computer Science Bibliography nas seguintes conferências:

(a) ICWS International Conference on Web Services;

(b) SERVICES;

(c) ARES.

A opção por essas fontes foi motivada por uma pesquisa inicial realizada na base de dados da DBLP, que retornou 77 publicações. Sendo que após uma análise preliminar, observou-se que as as conferências mais relevantes quantitativamente foram as citadas no item c desta Seção. Seguiu-se o mesmo procedimento para as bibliotecas citadas nos ítens a e b.

No contexto da elaboração dos termos de busca, para a base de dados eletrônica, usou-se a abordagem descrita em (KITCHENHAM; CHARTERS, 2007), que consiste em criar os termos de busca a partir de questões de pesquisa, usando a composição de operadores AND e OR. Dessa forma, a *string* de busca usada na realização desse estudo foi ***SOA and SECURITY*** não sendo usados sinônimos ou outras variações.

2.6 CRITÉRIOS DE INCLUSÃO E EXCLUSÃO

Após serem realizadas as pesquisas, de acordo com a *string* de busca, vários estudos primários foram recuperados e avaliados de forma que fossem excluídos aqueles que não satisfizessem os objetivos do estudo. A investigação dos estudos consistiu inicialmente em analisar o título, o resumo, a introdução e a conclusão.

As conclusões foram analisadas para entender melhor as contribuições do trabalho. Os critérios de seleção dos estudos foram baseados nas questões de pesquisa e todos os estudos recuperados foram armazenados. Logo para este mapeamento foram definidos os seguintes critérios de inclusão e exclusão:

No que se refere aos critérios de inclusão, foram incluídos todos os artigos que faziam referência a SOA Security e que foram publicados nos periódicos e conferências publicados no período compreendido do ano de 2000 até 2013 e que satisfizessem a *string* de busca anteriormente definida.

Já no que tange os critérios de exclusão, foram excluídos os artigos e resumos com menos de quatro páginas, artigos que não tratavam diretamente do tema SOA SECURITY, contribuições que, apesar de versar sobre SOA, não faziam referência à segurança e vice-versa, as que tratavam de segurança, mais não tratavam de arquitetura orientada a serviços. Além disso, foram também excluídos os estudos irrelevantes para a pesquisa e aqueles que não puderam ser obtidos gratuitamente.

2.7 COLETA, ARMAZENAMENTO DOS DADOS E ANÁLISE

Após a seleção preliminar e seguindo o que foi preconizado na estratégia de busca, foram realizadas buscas automáticas a cada uma das bibliotecas digitais citadas na Seção 2.5, sendo recuperadas ao todo 54 publicações. A última etapa da seleção consistiu em aplicar os critérios de exclusão e inclusão aos artigos selecionados, de forma que se obteve um total de 25 publicações, que foram analisadas e classificadas de acordo com a seguinte classificação:

1. Veículos: Neste tópico buscou-se categorizar quais veículos que mais apresentaram publicações;

2. Pesquisa: Este tópico foi utilizado para definir quais tipos de pesquisa foram propostas no estudo de forma que as publicações foram classificadas como:

   a) Solução: Representa as publicações que propõem uma nova técnica, método ou ferramenta/arquitetura e que a partir dela possa ser realizado algum tipo de verificação de viabilidade;

   b) Validação: Neste caso são proposições de validação de algum estudo que apresente rigor cientifico, tais como estudos empíricos ou provas da aplicação de alguma técnica, podendo ser uma validação formal ou experimental;

   c) Avaliação: Publicações que apresentaram avaliações comparativas entre técnicas propostas de algum estudo relacionado a Segurança em SOA, sendo classificado como:

      - Avaliação Formal, que é aquele que possui detalhes do estudo, sendo possível, caso necessário, realizar uma reprodução do trabalho;
      - Avaliação Informal, que é aquela em que os detalhes do estudo são poucos o que torna difícil sua reprodução;
      - Avaliação Preliminar, neste caso serão considerados os estudos cujos resultados podem ser questionados e não apresentam detalhes para reprodução

d) Artigos de Opinião: que são estudos informais que fazem uma abordagem geral dos aspectos do tema Segurança em SOA.

3. Contexto: Busca-se neste tópico classificar as publicações que sejam classificados como uma Solução e possuam atributos relacionados a segurança em SOA que abordem: privacidade, confidencialidade, autenticidade e disponibilidade.

2.8 RESULTADOS

Nesta seção são descritos os resultados obtidos após o estudo e o mapeamento das principais publicações coletadas na seção anterior e que serão utilizados para responder as questões de pesquisas anteriormente definidas.

2.8.1 QP 1 QUESTÃO DE PESQUISA 1

Quais são os principais veículos que publicam artigos nessa área? Para responder essa pergunta foram analisados os 25 artigos selecionados após a aplicação dos critérios de exclusão e inclusão. Eles foram classificados de acordo com o veículo de publicação do artigo. Essa classificação é descrita na figura 1.

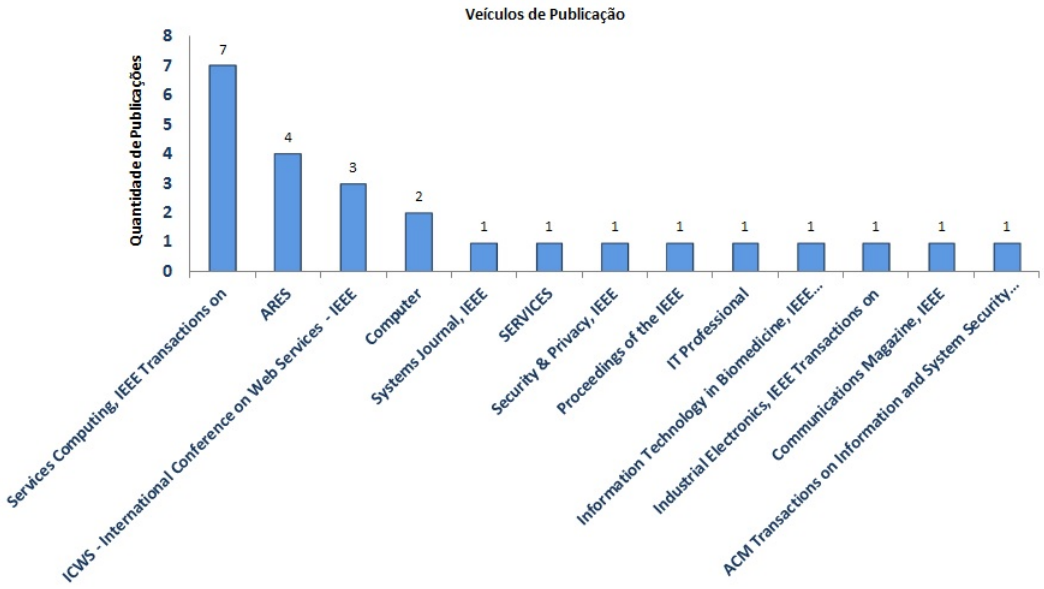

Figura 1: Gráfico representativo da quantidade de contribuições por veículo de publicação.

Verifica-se que os veículos que mais publicaram artigos relacionados a segurança em SOA foram Services Computing, ARES, ICWS e Computer com 28%, 16%, 12% e 8% dos artigos publicados, respectivamente. Eles reponderam juntos por aproximadamente 64% das publicações.

2.8.2 QP 2 QUESTÃO DE PESQUISA 2

Qual o tipo de contribuição é a mais proposta nas pesquisas realizadas? Após a realização do mapeamento foi possível verificar que dentre os artigos mapeados houve artigos que fizeram referência a mais de um tipo de contribuição. Isso pode ser verificado no artigo proposto por (ZHANG; CHEN, 2011) que foi classificado como sendo uma Solução e uma Validação. Um outro exemplo pode ser observado no artigo (LOWIS; ACCORSI, 2011) que foi classificado como sendo uma Solução e uma Avaliação. Dessa forma, a contribuição mais proposta foi a Solução com 42% do total de artigos selecionados, seguidos por Avaliações e Artigos de Opinião ambos com 24% e finalmente a Validação com 10%. Dentre essas soluções, cabe ressaltar a que é proposta por (LOWIS; ACCORSI, 2011) que propõem um novo método denominado ATLIST, que realiza análise de vulnerabilidades em processos de negócios e serviços baseados em SOA. Tal trabalho pode ser útil para a arquitetura de referência que será proposta como resultado deste trabalho de mestrado.

Além desta, outra que também deve ser citada é a que é idealizada por (DAGOSTINI et al., 2012), neste artigo é proposta uma Ontologia, ASSERT4SOA, que foca na interoperabilidade e comparação de certificados heterogêneos e possibilitando a verificação em tempo de execução da conformidade dos serviços com os requisitos de segurança.

Outro artigo relevante para a problemática discutida nesta dissertação é o artigo proposto por (WEBER et al., 2007). Nesta solução, é proposta uma ferramenta que tem por objetivo identificar possíveis violações de segurança em ambientes SOA. Nela são descritas formas para inspecionar os arquivos de configuração da plataforma SOA, sendo possível detectar possíveis violações de segurança. Sendo também realizada uma avaliação informal que procura analisar as melhores práticas de segurança em SOA.

Já quanto à contribuição que se refere à avaliação pode-se verificar 4 (quatro) artigos classificados neste tipo, são de avaliações formais, ou seja, estudos que traziam resultados detalhados que podem ser reproduzidos por outros pesquisadores. Um exemplo de avaliação formal pode ser identificado no artigo (COETZEE, 2012), este artigo analisa os desafios de segurança enfrentados em arquiteturas orientadas a serviço. Nele é proposta uma estrutura de segurança aplicada a SOA baseado em componentes, que consistem em uma variedade de controles que podem minimizar os desafios referentes à aplicação dos mecanismos de segurança em SOA. Os outros artigos classificados como avaliação foram classificados como avaliações informais 4 (quatro) artigos e experimentais 1 (um) artigo totalizando 5 (cinco) artigos.

No que se refere ao tipo de contribuição Validação, foram identificados 4(quatro)

artigos sendo 3(três) validações formais e 1(uma) experimental. Neste tipo de contribuição e relevante citar o trabalho realizado no artigo (XU et al., 2012). Neste artigo, é proposta uma ferramenta, que também é uma solução técnica, de um novo protocolo de autenticação para interações de serviços dinâmicos, com base na noção de sessões de negócios multipartidárias orientadas a serviços. Esse protocolo não requer nem conversão de credencial nem estabelecimento de qualquer caminho de autenticação entre os serviços que participam de uma sessão de negócios. Nele são realizadas provas e experimentos para verificar a viabilidade da nova técnica. Este trabalho também pode ser útil para a arquitetura de referência que será proposta, uma vez que traz uma nova técnica que pode ser utilizada como referência nos processos de autenticação dos serviços oferecidos.

E por fim, no caso dos Artigos de Opinião, forma verificados 9(nove) artigos que faziam uma abordagem geral dos panoramas e desafios de segurança em arquitetura orientada a serviços, nesses artigos não foi proposto nenhuma contribuição importante. Porém como eles se enquadraram nos critérios de busca estabelecidos, foram considerados.

A figura 2 representa os resultados categorizados pelo tipo da contribuição nele é apresentado o quantitativo de publicações.

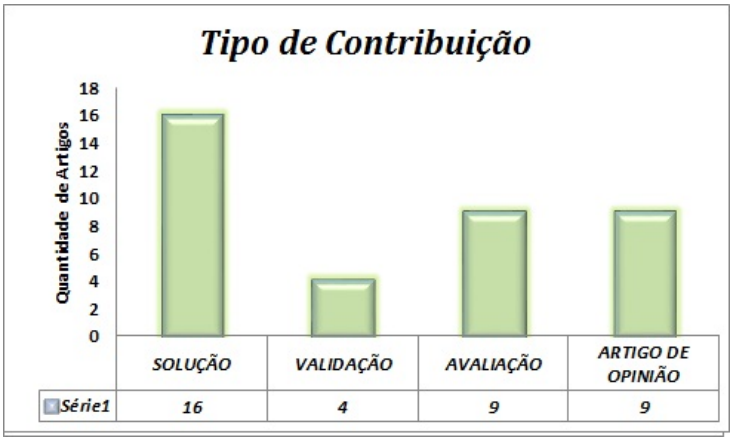

Figura 2: Gráfico representativo dos tipos de contribuições

2.8.3 QP 3 QUESTÃO DE PESQUISA 3

Quais atributos de segurança são os mais abordados nos estudos? Para responder essa pergunta inicialmente realizou-se uma análise nos artigos classificados com Solução. Em seguida foram categorizados de acordo com os atributos relativos à segurança em ambiente SOA. Os atributos analisados foram: integridade, autenticidade, disponibilidade e confidencialidade. Foi possível verificar que dentre os artigos mapeados houve artigos que faziam referência a mais de um atributo. Um exemplo desse fato é descrito na publicação

de (DELESSY; FERNANDEZ, 2008) onde são abordados todos os atributos de segurança: integridade, autenticidade, disponibilidade e confidencialidade. Dessa forma, o número de artigos mapeados com os atributos descritos não será equivalente com o número de artigos totalizados no mapeamento.

Sendo assim, o mapeamento identificou que os conceitos mais abordados referem-se aos atributos de autenticidade com 32% ou 13 artigos e Integridade com 28% ou 11 artigos. Já o atributo confidencialidade foi verificado em 20% das publicações, com 8 artigos, e o atributo disponibilidade foi verificado em 15% das publicações, sendo identificado em 6 artigos. Finalmente, para os artigos que foram classificados como uma solução e que não se enquadraram em nenhum dos atributos, sendo classificados como atributos de qualidade de serviços, foram identificados em apenas 5% das publicações ou 2 artigos. A figura 3 representa graficamente essa categorização.

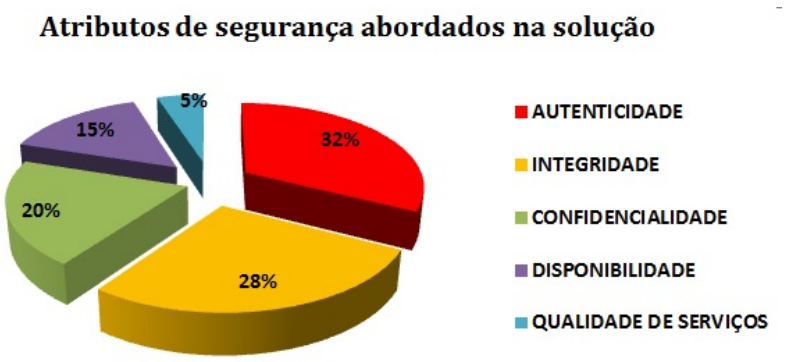

Figura 3: Gráfico dos atributos de segurança abordados na solução

2.8.4 QP 4 QUESTÃO DE PESQUISA 4

Quais contribuições classificadas como solução e categorizadas como: método, técnica ou ferramenta/arquitetura, foram os mais propostos nas pesquisas? Para responder essa pergunta, foi realizada uma análise nos artigos classificados com solução. Em seguida foram categorizados de acordo com os tipos: técnica, método ou ferramenta/arquitetura. Verificou-se que dentre os artigos mapeados que um artigo que fez referência a mais de um tipo de solução. Esse artigo é descrito na publicação de (XU et al., 2012) onde são abordados os tipos de solução técnica e ferramenta/arquitetura. Dessa forma, o número de artigos mapeados como sendo uma solução do tipo: método, técnica ou ferramenta/arquitetura não será equivalente com o número de artigos totalizados no mapeamento.

Sendo assim, o mapeamento identificou dentre as contribuições classificadas como solução, que o tipo de solução mais proposta é a de solução técnica com 44% das contribuições

ou 7 artigos, já as classificadas como ferramenta/arquitetura, foi observado em 31% das contribuições, ou 5 artigos. Finalmente, o tipo de solução classificado como método, foi verificado em 25% das contribuições selecionadas, ou 4 artigos. A figura3 representa graficamente essa categorização.

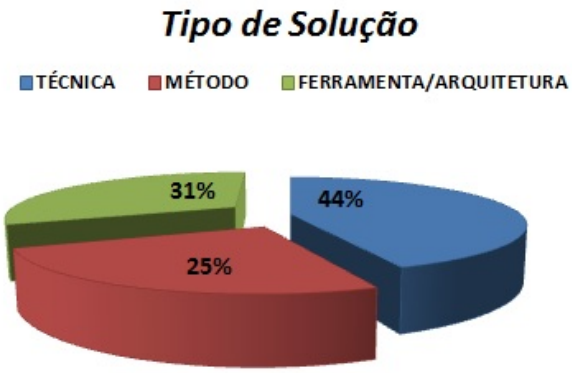

Figura 4: Gráfico dos tipos de Solução mais propostos

2.8.5 DISCUSSÃO SOBRE OS RESULTADOS

De acordo com os resultados obtidos por este mapeamento sistemático foi possível observar que a maioria doa artigos se concentram nos veículos: Services Computing, ARES, ICWS e Computer com 28%, 16%, 12% e 8%, respectivamente. Eles reponderam juntos por aproximadamente 64% das publicações.

No que tange os resultados obtidos a respeito do tipo de contribuições, verificou-se que a maior parte de contribuições foram de soluções com 42% dos artigos e que destas destacaram-se as soluções técnicas com 44% dos artigos classificados nesta faceta de pesquisa. Este resultado denota que os pesquisadores tem buscado estabelecer padrões e procedimentos com objetivos específicos de resolver problemas ou melhorar as técnicas existentes que estejam relacionados a segurança em SOA.

No contexto das contribuições referentes aos atributos de segurança que foram abordados e classificados com solução, verificou-se que as maiores preocupações estavam associadas aos atributos de autenticidade, integridade e confidencialidade com 33%, 27% e 20% das contribuições respectivamente. Dessa forma, verifica-se que dentre as soluções anteriormente citadas a maior preocupação é em estabelecer técnicas eficientes para autenticar os serviços e garantir que o conteúdo das mensagens não seja modificado. Isso pode denotar uma preocupação com o desempenho dos serviços no momento da aplicação dos mecanismos de segurança.

2.8.6 CONCLUSÃO

Esse mapeamento sistemático possibilitou aprofundar os conhecimentos referentes à segurança em SOA. Com ele foi possível verificar quais são os principais veículos que publicam artigos nesta área o que ajuda a direcionar os estudos. Também foi possível identificar e analisar quais são os tipos de contribuições que mais são propostas nas pesquisas envolvendo esta temática. Além disso, verificou-se dentre as contribuições propostas como solução, quais delas envolviam atributos de segurança e que tratavam de integridade, autenticidade, disponibilidade e confidencialidade. E por fim, nas contribuições classificadas como solução, foi possível identificar e quantificar quais tipos de solução (método, técnica e ferramentas/arquitetura) foram as mais propostas.

Com este mapeamento sistemático foi possível identificar alguns artigos que serão fundamentais para nortear a pesquisa desenvolvida nesta dissertação. Eles são descritos a seguir:

No artigo (WEBER et al., 2007), são descritas formas para inspecionar os arquivos de configuração da plataforma SOA, sendo possível detectar possíveis violações de segurança. Um dos focos do trabalho está relacionado com as melhores práticas para a implantação de segurança em SOA, o que pode enriquecer a arquitetura de referência proposta nesta dissertação.

Outro trabalho que também deve ser citado, e o descrito na publicação de (DELESSY; FERNANDEZ, 2008) onde são abordados todos os atributos de segurança: integridade, autenticidade, disponibilidade e confidencialidade. Nele é proposta uma nova abordagem para proteger aplicativos de SOA. Outro ponto importante, e que a segurança não é considerada como um aspecto isolado, mas como um aspecto presente em todas as fases de um desenvolvimento do sistema.

Já o artigo descrito em (COETZEE, 2012), analisa os desafios de segurança enfrentados em arquiteturas orientadas a serviço. Sendo proposta uma estrutura de segurança aplicada a SOA, baseado em componentes, que consiste em uma variedade de controles que podem minimizar os desafios referentes à aplicação dos mecanismos de segurança em SOA. Esse artigo realça os desafios de segurança em SOA e pode servir como base para a proposta de criação da arquitetura de referência.

3 REVISÃO DA LITERATURA

Este capítulo apresenta uma revisão dos principais conceitos relacionados ao tema deste trabalho, envolvendo Arquitetura Orientada a Serviço, da Tecnologia Web Service e da Segurança Aplicada a SOA.

3.1 ARQUITETURA ORIENTADA A SERVIÇOS

A Arquitetura Orientada a Serviços (SOA) , consiste em uma coleção de componentes distribuídos que fornecem e ou consomem serviços (CLEMENTS et al., 2010), tem sido amplamente utilizada por um grande número de empresas.

Serviços correspondem a recursos de software bem definidos através de uma linguagem padrão, são auto-contidos, proveem funcionalidades padrões do negócio, independentes do estado ou contexto de outros serviços (FURTADO et al., 2009).

SOA pode ser caracterizada como uma arquitetura corporativa onde serviços podem ser criados, reutilizados e facilmente compartilhados entre aplicações. Neste caso as funcionalidades de um sistema são decompostas em serviços interoperáveis o que permite a integração entre aplicações. O objetivo da SOA é estruturar sistemas distribuídos com base nas abstrações de regras e funções de negócio (JOSUTTIS, 2007).

Existem muitas definições para SOA, no entanto elas possuem pontos em comuns pois em todos os conceitos são abordados temas que remetem ao compartilhamento do serviços, da independência da plataforma e linguagem de programação, da possibilidade de flexibilidade e agilidade no desenvolvimento de uma aplicação para gerenciar um negócio (ERL, 2009). Segundo (JOSUTTIS, 2007), o funcionamento de SOA baseia-se em três conceitos: serviços, interoperabilidade e baixo acoplamento.

Por serviços entende-se SOA como uma arquitetura neutra, que objetiva abstrair a realidade, concentra-se nos aspectos do negócio, possibilitando que sistemas sejam construídos em plataformas diferentes, tendo como finalidade a redução de problemas de integração,

uma vez que isso pode ser feito de forma flexível por meio dos serviços disponibilizados na arquitetura. Portanto, serviço pode ser visto como um conjunto de funções, abstrações de funcionalidades de negócios de um sistema com uma interface bem definida.

A interoperabilidade visa à integração entre esses sistemas e representa um objetivo fundamental da orientação a serviços, pois estabelece uma base para a realização de outros objetivos e benefícios estratégicos. Isso é possível, pois uma das características de SOA é que seus serviços são reutilizáveis, possuem baixo acoplamento, tem contratos formais e são independentes. Logo, uma vez que esses serviços estejam disponíveis aos clientes eles não precisam conhecer a lógica ou os processos de negócio para consumir e integrar serviços a suas aplicações.

O baixo acoplamento ou acoplamento fraco é um conceito vital para o funcionamento de um sistema distribuído, uma vez que ele determina que diferentes partes e funcionalidades de um sistema sejam independentes umas das outras, dessa maneira, alterações ou problemas em uma determinada parte do sistema não trará consequências para o resto do sistema, trazendo benefícios como escalabilidade, flexibilidade e tolerância a falhas. Acoplamento fraco refere-se a uma abordagem em que as interfaces podem ser desenvolvidas com o mínimo de suposições mútuas entre o emissor e os destinatários, reduzindo assim o risco de que uma mudança em um aplicativo ou módulo force uma mudança em outra aplicação ou módulo.

Segundo (BIANCO et al., 2007), SOA é um estilo arquitetural e para implementá-lo podem ser usadas diversas tecnologias tais como RMI (*Remote Method Invocation*), CORBA (*Common Object Request Broker Architecture*), DCOM (*Distributed Component Object Model*), REST (*Representational State Transfer*) e Web Services, que é uma das principais tecnologias para implementação desses serviços.

A abordagem de arquiteturas orientadas a serviços (SOA) e de Web Services estão centradas no conceito de serviço, tanto no nível de negócios quanto no nível tecnológico, e compartilham os mesmos princípios.

Apesar de SOA e Web Service compartilharem os mesmos princípios, SOA representa propositadamente uma tecnologia neutra, a definição de Web Service destaca o papel central das tecnologias da Web específicas.

3.2 WEB SERVICES

Web Service pode ser definido como um sistema de software projetado para suportar interações interoperáveis máquina-a-máquina sobre uma rede (BOOTH et al., 2004).

Um dos fatores da aceitação de Web Services está no fato dele usar protocolos abertos de comunicação na Internet e XML para transacionar o seu negócio. Um WS é, portanto, um sistema de software que pode agir a pedido de qualquer computador conectado à rede e que se comunica usando padrões XML (PULIER; TAYLOR, 2005).

Por meio desta tecnologia é possível promover a interoperabilidade entre aplicações e que tenham sido desenvolvidos em plataformas diferentes tornando-as compatíveis permitindo que as aplicações enviem e recebam dados em formatos variados. Cada aplicação pode ter a sua própria linguagem, que é traduzida para uma linguagem universal, como é o caso do formato XML.

A abordagem de arquiteturas orientadas a serviço e Web Services estão centradas no conceito de serviço, tanto a nível de negócios quanto a nível tecnológico, e compartilham os mesmos princípios (BERTINO et al., 2010). Dentre os princípios que mais se destacam podem ser citados:

- Autonomia de serviço: Para os serviços realizarem suas capacidades de modo consistente e confiante, sua lógica precisa ter um grau significativo de controle sobre seu ambiente e recursos (ERL, 2009).
- Baixo acoplamento: Acoplamento refere-se a uma conexão do relacionamento entre dois elementos. Uma medida de acoplamento se compara a um nível de dependência (ERL, 2009). De outra forma pode dizer que o baixo acoplamento refere-se a uma abordagem em que as interfaces podem ser desenvolvidas com a mínima dependência uma das outras o que reduz o risco de uma mudança e qualquer uma das partes forçar a mudança na outra parte não modificada.
- Contrato formal: O contrato informa o que o Serviço faz e como ele se comunica (o que deve receber e o que deve entregar). Em outras palavras Contratos são documentos textuais que descrevem o que o serviço faz e eles são o foco do design de serviço, porque regem praticamente tudo que é feito pelos serviços (ERL, 2009). Logo, todo serviço possui um contrato entre o requisitante e o provedor deste serviço.

Segundo (BERTINO et al., 2010), cada organização tem que ter autonomia para exercer um controle independente sobre os seus serviços. Para isso a autonomia do negócio tem que ser correspondente a do Web Service no momento do oferecimento e execução do serviço.

A plataforma de Web Services é definida por vários padrões da indústria suportados por todas as comunidades de fornecedores. Essa plataforma está associada à coleção de padrões e especificações de tecnologias abertas tais como Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) e Universal Description and Discovery Information (UDDI). Nas seções a seguir será feito uma descrição de cada uma dessas tecnologias.

3.2.1 SIMPLE OBJECT ACCESSS PROTOCOL (SOAP)

SOAP é um protocolo de transporte que é responsável pela troca de mensagens entre aplicações em ambientes distribuídos e descentralizados, ele segue um padrão que foi especificado pelas normas da W3C, sendo baseado em XML o que o torna totalmente compatível com qualquer plataforma e com linguagens que tenham suporte para a manipulação de arquivos XML. Seu conteúdo é composto por informações e estruturas de dados (COYLE, 2002).

A estrutura de uma mensagem SOAP é definida em um documento XML, sua estrutura possui os seguintes elementos: *envelope, body* que são obrigatórios e *header e fault* que são opcionais. As seguir uma breve descrição desses elementos:

- *Envelope*: Este é o elemento raiz da mensagem SOAP sendo responsável por identificar o documento XML com uma mensagem SOAP e por definir o conteúdo da mensagem;
- *Header* (opcional): Contêm os dados do cabeçalho, este elemento possui informações especificas do aplicativo da mensagem SOAP;
- *Body*: Contém as informações de chamada e resposta ao servidor, ele contém a mensagem SOAP pretendida que o usuário espera;
- *Fault*: Este elemento possui as informações dos erros ocorridos no envio da mensagem. Esse elemento só aparece nas mensagens de resposta do servidor.

3.2.2 WEB SERVICES DESCRIPTION LANGUAGE (WSDL)

Web Services Description Language (WSDL) é uma linguagem para descrição de serviços escrita em XML. Nela são descritos os serviços externos, ou interfaces que são oferecidos por uma determinada aplicação, independente de sua plataforma ou linguagem de programação. Além disso, ela contém as especificações de localização das operações (métodos ou serviços) que fazem parte dos Web Services. Atualmente ela encontra-se na versão 2.0.

WSDL podem ser mapeados para qualquer linguagem de implementação, plataforma, modelo de objeto e ou sistema de mensagens (BERTINO et al., 2010). Ele é caracterizado por uma parte abstrata, onde é descrita a interface do serviço e outra concreta local onde são definidos os protocolos de conexão e outras informações, conforme Figura 5.

A parte abstrata é constituída pelos seguintes elementos: Tipos, que são elementos que definem os tipos de dados usados pelos Web Services, neles são especificados os tipos que serão trocados nas mensagens de entrada e saída do serviço; Mensagem, que é um elemento que permite descrever de forma abstrata os dados que serão transmitidos entre o serviço e o consumidor do serviço; Operações, que é um elemento que é semelhante à definição de um método, no entanto, ele só permite que você defina a entrada, saída e mensagens de erro que estão associados com uma operação e PortType, que são conjuntos de operações abstratas, que são suportadas por um serviço, cada um contendo mensagens de entrada e saída.

Já a parte concreta do WSDL, local de definição do protocolo e do endereço onde o serviço estará disponibilizado, compõe-se pelos seguintes elementos: O binding (ligação), que é o elemento que é responsável por ligar os elementos abstratos e concretos em um documento WSDL e de fornecer detalhes de como as mensagens serão transmitidas. E os Serviços e Portas, que são elementos que especificam a localização (endereço URL ou e-mail), neste caso, o elemento de serviço atribui um nome para o serviço e o associa a uma interface abstrata e descrevendo o local onde o serviço será acessado.

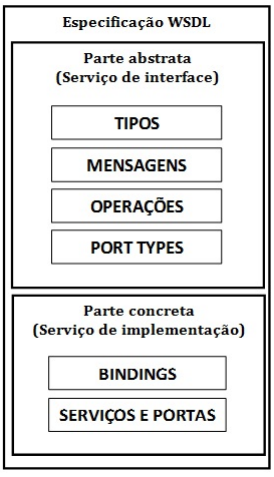

Figura 5: Especificação de serviços WSDL, adaptado de (BERTINO et al., 2010)

3.2.3 UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION (UDDI)

UDDI é um componente importante da arquitetura de Web Services, sendo formado por um serviço de diretório que armazena descrições de serviço. Esse serviço obedece ao padrão

integração, descoberta e descrição universal. Além disso, ele prescreve o layout de um banco de dados que contém descrições de serviços que permitirão a clientes de serviços web procurar serviços relevantes (TANENBAUM, 2007).

O UDDI provê um método padronizado para a publicação e descoberta de informações, permitindo que as empresas tanto publiquem como encontrem Web Services. Segundo (CERAMI, 2002), os dados capturados dentro de UDDI são divididos em três categorias principais:

a) Páginas brancas: Esta categoria inclui informações gerais tais como nome, descrição e endereço dentre outras informações sobre o fornecedor do serviço;

b) Páginas amarelas: Esta categoria inclui dados de classificação geral para qualquer empresa ou serviço oferecido. Por exemplo, esses dados podem incluir a indústria, o produto, ou códigos geográficos baseados sobre taxionomias padronizadas;

c) Páginas verdes: Esta categoria inclui informações técnicas sobre um Web Service. Geralmente, essa informação inclui um apontador (ponteiro) para uma especificação externa e um endereço para invocar o serviço.

3.3 *REPRESENTATIONAL STATE TRANSFER*

A arquitetura *Representational State Transfer (REST)*, foi proposta por Roy Fielding em 2000 em sua tese de doutorado e pode ser descrita como um conjunto de princípios arquiteturais que podem ser utilizados para o desenvolvimento de serviços web e que utilizam o protocolo HTTP para realizar as trocas de mensagens (FIELDING, 2000).

Dessa forma, para que os princípios arquiteturais que permeiam *REST* sejam seguidos, um conjunto de restrições deve ser implementado (FIELDING, 2000). Um aplicação que esteja em conformidade com essas restrições, a seguir descritas, são classificadas com RESTfull. (RICHARDSON; RUBY, 2007)

Cliente-Servidor: Essa restrição está associada a separação de interesses, que é o princípio por trás das restrições da arquitetura cliente-servidor. Nela procura-se separar as preocupações relacionadas à interface do usuário das preocupações de armazenamento de dados. Isso permite que os componentes possam evoluir de forma independente melhorando a portabilidade e a escalabilidade das aplicações.

***Stateless*:** Diz respeito à interação entre cliente e servidor. Nesse caso, a comunicação deve ser realizada sem que haja o armazenamento de qualquer tipo de estado no servidor. Sendo

assim, toda informação de estado deve ser conhecida somente pelo cliente. Esta característica permite a escalabilidade do servidor, uma vez que pode liberar recursos no final de cada pedido. Contudo, uma desvantagem associado a essa característica está relacionada à performance da rede, pois em decorrência das constantes requisições com dados repetidos ela é reduzida.

***Cache*:** A utilização do cachê, tem a finalidade de diminuir o impacto da desvantagem ocasionada pela redução de performance. Uma vez que exige que os dados de uma resposta vinda de uma requisição ao servidor, sejam marcados como *cacheable* (sujeito à utilização do cachê) ou *noncacheable* (não sujeito à utilização do *cache*). Se uma resposta for marcada como cacheable, então ela será reutilizada como resposta em futuras requisições equivalentes, permitindo que o servidor fique mais livre, e portanto, mais escalável, haja vista que algumas interações poderão ser eliminadas por completo o que melhora a eficiência e performance de acesso a recursos percebido pelo usuário.

Sistema em camadas: Essa restrição caracteriza-se pela divisão do sistema em camadas hierárquicas, restringindo a visualização dos componentes participantes de forma que cada componente só possa ver a camada com a qual esteja interagindo diretamente. Ao restringir a visibilidade de um sistema a uma única camada, torna-se possível delimitar a complexidade do sistema e promover a independência de cada uma das camadas. Essa separação permite que o sistema seja mais robusto e resistente a erros.

Code-On-Demand: Dentre o conjunto restrições propostas pelo estilo REST, esta é aquela que permite a opção de baixar e executar diretamente códigos no lado cliente, sendo opcional. Com isso, busca-se obter extensibilidade e simplificar o cliente. No entanto, isso também reduz a visibilidade.

Interface Uniforme: A principal característica que diferencia o estilo arquitetural REST de outros utilizados em rede é a ênfase quanto ao uso de uma interface uniforme entre os componentes. Aplicando o princípio de generalização de engenharia de software à interface dos componentes, a arquitetura é simplificada e a visibilidade da interações é melhorada. Contudo, esta generalização pode diminuir a eficiência do sistema, devido à aplicação não poder transmitir a informação num formato especifico de acordo com a sua necessidade. Com o objetivo de obter uma interface uniforme, REST define quatro requisitos de interface:

- **Identificação dos recursos:** Na arquitetura REST, cada recurso deve possuir um identificador universal denominado Uniform Resource Identi?er (URI). Que é definido como uma sequência de caracteres que identi?cam um recurso físico ou abstrato [RFC 3986]. E são utilizados para descoberta de recursos e serviços.

- **Representação de recursos:** Os recursos devem ser manipulados a partir de suas representações, uma vez que elas podem estar representadas em formatos diferentes formatos, tais como: JSON, XML,PDF, texto puro, etc. É importante frisar que uma aplicação REST não transmite o recurso efetivamente, mas sim a sua uma representação, em um formato pré-acordado entre o cliente e o servidor;

- **Mensagem auto descritivas:** Os recursos são dissociados da sua representação, haja vista que o seu conteúdo pode que ser acessado em em formatos diferentes. Dessa forma, as mensagens devem conter metadados que indicam como o conteúdo transmitido deve ser tratado. Os metadados são utilizados para controlar cache, detectar erros de envio, negociar formatos de uma representação adequada, realizar controle de autenticação e acesso, etc;

- **Utilização de hipermídia para estado da aplicação:** Neste caso, as representações de recursos obtidas em uma aplicação REST devem possuir *hiperlinks* que permitam a navegação do cliente pelos recursos. Uma vez que o servidor não pode armazenamento de qualquer tipo de estado. Dessa forma, o Cliente pode interagir com outros recursos existentes sem a necessidade de que ele conheça a relação completa destes recursos pois poderá seguir estas ligações para se deslocar de um recurso para outro.

O protocolo HTTP é o padrão utilizado na arquitetura REST para promover a comunicação entre o cliente e o servidor. Isso se dá pela manipulação dos recursos utilizando os métodos HTTP: GET, POST, PUT ou DELETE.

3.4 SEGURANÇA EM SOA

Segurança da informação pode ser definida como um conjunto de ações que são executadas com a finalidade de prover segurança às informações de indivíduos e organizações. Atualmente a segurança é um requisito importante para qualquer aplicação distribuída, tais como aplicações governamentais, aplicações de segurança pública e de defesa dentre outros (BERTINO et al., 2010).

A Segurança aplicada a SOA requer o estabelecimento de propriedades básicas, uma vez que os aspectos funcionais aplicados a esta arquitetura são iguais aos de aplicações tradicionais. Logo, segundo (VERISSIMO; RODRIGUES, 2001), a segurança está fundamentada nos seguintes atributos básicos: autenticidade, integridade, confidencialidade e disponibilidade.

No que se refere à autenticidade, este é um atributo que visa estabelecer a origem da informação, buscando verificar a identidade de um usuário. Assim, objetiva-se garantir que o usuário ou serviço é realmente quem diz ser e que tem os privilégios necessários para acessar e ou enviar uma determinada informação.

A integridade é aquela que se preocupa em evitar ou em detectar a modificação não autorizada de informações ou mensagens. Esse atributo busca proteger a mensagem de modificações não permitidas.

A confidencialidade preocupa-se com a proteção contra acessos não autorizados de dados e informações. Segundo (BERTINO et al., 2010), esse atributo procura proteger o conteúdo de uma mensagem ou informação para que ele não possa ser visualizado no momento da transmissão, exceto por serviços autorizados a visualizá-los por terem a necessidade de ver o conteúdo da mensagem, a fim de realizar o seu encaminhamento.

Finalmente, a disponibilidade está preocupada com a garantia de que os serviços de informação permaneçam acessíveis somente a usuários autorizados. De forma que uma mensagem ou informação uma vez solicitada possa ser prontamente entregue ao destinatário, garantindo assim que os usuários legítimos recebam os serviços a que têm direito. E outra palavras esse atributo busca garantir que a informação estará disponível quando solicitada.

Apesar de compartilhar estes conceitos, a segurança em SOA exige uma abordagem diferenciada e outros aspectos devem ser verificados. Um exemplo disto é a proposta de segurança em nível de mensagem.

Segundo (KANNEGANTI; CHODAVARAPU, 2008), a segurança em nível de mensagem busca sanar problemas e complementar a segurança que é oferecida na camada de transporte, que é realizada por meio dos protocolos SSL (*Security Socker Layer*) e TLS (*Transport Layer Security*). Neste caso, os dados são cifrados na camada de transporte sendo estabelecido um canal seguro de comunicação entre dois serviços. Dessa forma, a comunicação ponto a ponto é garantida e segura. Porém, uma vez existindo interfaces intermediárias entre provedor do serviço e o consumidor, o processo de cifrar e decifrar os dados, que ocorre na camada de transporte, irá ocorrer toda vez que os dados trafegarem por um serviço intermediário, isso faz com que o sigilo dos dados e a segurança sejam quebrados em cada serviço intermediário. Para resolver este problema, propõem-se a segurança em nível de mensagem que consiste em utilizar mecanismos de segurança como cifrar partes da mensagem, o que resolve este problema, pois a segurança é fim a fim o que significa dizer que os dados estarão seguros mesmo quando a transmissão envolver um ou mais intermediários.

3.5 VULNERABILIDADES EM SOA

Segundo (VERISSIMO; RODRIGUES, 2001), hackers buscam por falhas e vulnerabilidades dos sistemas operacionais, aplicativos, software de rede e assim por diante. Usuários imprudentes ou administradores podem apresentar outras falhas, por causa da maneira como eles configuram ou executam os sistemas que operam. Esses problemas podem ocorrer em SOA. Para que se possa entender melhor esse contexto, são descritos a seguir conceitos que auxiliam o entendimento de vulnerabilidades.

Em relação a vulnerabilidades, pode-se afirmar que elas são falhas introduzidas, acidentalmente ou intencionalmente, durante o processo de desenvolvimento, configuração, operação e manutenção do sistema. Já ataques, são ações que exploram vulnerabilidades, podendo comprometer ou não as propriedades de segurança do sistema. E intrusão no sistema, é o resultado de um ataque bem sucedido no sistema.

A segurança aplicada a SOA é algo que deve ser continuamente buscado. Porém, essa tarefa é complexa e muitas vezes difícil. Isso decorre do fato de existirem muitas vulnerabilidades nos componentes de software. Com isso, para evitar que problemas de segurança ocorram, é necessário estar sempre revisando os métodos, técnicas e ferramentas que possibilitem verificar vulnerabilidades e propiciar segurança aos sistemas computacionais. A seguir serão descritos algumas vulnerabilidades que afetam a arquitetura orientada a serviços e que são citados no trabalho (JENSEN et al., 2007).

3.5.1 ALTERAÇÃO DAS MENSAGENS

Neste tipo de ataque, as mensagens são interceptadas por um atacante que realiza alterações na mensagem toda ou em parte dela, afetando sua integridade. *SQL Injection*, *XML Injection* e *XPath Injection* são exemplos de ataques que utilizam essa estratégia.

O *SQL Injection*, é o ataque em que o atacante envia determinados parâmetros como argumentos das funções que, após processados pelo Web Service, originam procedimentos SQL definidos pelo atacante (PINHO, 2008). Esses procedimentos podem retornar informações não previstas, alterar dados constantes nas bases de dados ou provocar indisponibilidade do próprio Web Service. Neste tipo de ataque podem ser executados comandos SQL inválidos.

No caso do *XML Injection*, que é o ataque onde se procura modificar a estrutura XML de uma mensagem SOAP (ou qualquer outro documento XML), através da inserção de elementos XML (JENSEN et al., 2007). Isto pode ser usado para substituir os dados inseridos

por usuário no mesmo documento. Nele, são aproveitadas as situações em que o processo de validação do XML não é efetuado corretamente, são inseridas tags num documento XML. As referidas tags XML podem alterar a estrutura do documento XML de tal forma que o comportamento da aplicação seja comprometido.

O *XPath Injection*, é o ataque onde são inseridos parâmetros maliciosos no XPath, com o objetivo de realizar consultas a informações cujo acesso não foi autorizado. Segundo a W3C O XPath é uma linguagem utilizada para realizar consultas em documentos XML, e assim como SQL, também é suscetível a injeção de parâmetros (BHALLA; KAZEROONI, 2007).

3.5.2 NEGAÇÃO DE SERVIÇOS

Ataques de negação de serviço são utilizados para impedir que o serviço funcione conforme o esperado, resultando em perda de disponibilidade (SIDDAVATAM; GADGE, 2008). Esse ataque é focado em tornar indisponível (site, aplicativo de servidor, serviço). Se um serviço recebe um número muito grande de pedidos, ele pode parar de fornecer o serviço aos usuários legítimos. Por exemplo, um ataque de negação de serviço pode ser realizado enviando uma grande quantidade de informações a um servidor com pedidos para consumir todos os recursos disponíveis no sistema, ou passando os dados de entrada mal formatados ao servidor de forma que ele pare de funcionar.

Ao contrário da maior parte dos ataques, esse não tem a intenção de invadir um computador para roubar informações confidenciais, seu objetivo é o de tornar inacessíveis os serviços providos pela vítima a usuários legítimos.

3.5.3 ATAQUES DE REFERÊNCIAS EXTERNAS

Neste tipo de ataque, o atacante consegue burlar as proteções criadas, como por exemplo, no caso de validadores de XML, e realiza a inclusão de referências externas que só serão consultadas após a validação do XML, mas antes da aplicação iniciar o seu processamento.

3.5.4 INTERCEPTAÇÃO DAS MENSAGENS

Neste ataque, as mensagens são interceptadas e alteradas sem que qualquer das partes, emissor ou consumidor de serviço saiba que houve a interceptação. São exemplos deste ataque: *Replay Attacks* e *Man-in-the-middle*.

No ataque *Replay Attacks*, o atacante intercepta uma mensagem e se faz passar pelo emissor. Dessa forma, ele pode reenviar mensagens que já tinham sido previamente enviadas, ou incluir partes de uma ou mais mensagens previamente enviadas numa nova mensagem (*replay* de partes da mensagem). Já no caso do ataque *Man-in-the-middle*, os dados trocados entre duas partes são de alguma forma interceptados, registrados e possivelmente alterados pelo atacante sem que as vitimas percebam as modificações.

3.5.5 DESCOBERTA DE INFORMAÇÃO

Neste tipo de ataque, as informações sobre os sistemas são descobertas e utilizadas para realizar um ataque, de acordo com o tipo de vulnerabilidade, que mais se adeque aos dados dos sistemas obtidos, tais como: tipo e versões do sistema operacional, localização de cópias de segurança, arquivos temporários, informações sobre os serviços dentre outras. Os principais ataques são:*WSDL Scanning* e Ataques aos UDDI.

No ataque *WSDL Scanning*, realiza-se uma varredura no documento WSDL, com a finalidade de se obter informações sobre os métodos, parâmetros e operações constantes no WSDL. Dessa forma, o atacante busca revelar informações sensíveis e possíveis falhas em um Web Service, o que lhe permite realizar um ataque bem sucedido (MORADIAN; HåKANSSON, 2006);

No caso do Ataques aos UDDI, o atacante analisa dados desprotegidos dos registros UDDI, e obtém detalhes relativos às funções disponibilizadas pelos Web Services e como acessá-los. Como, muitas vezes, os WSDL são gerados automaticamente por utilitários criados para exporem toda a informação disponível, relativa a um determinado método, é necessário ter alguma atenção na escolha/utilização desses utilitários de forma a ser possível proteger os Web Services desse tipo de ataque. Através da utilização de UDDI v.3.0.2 já é possível implementar alguma proteção relativa a este tipo de ataque. Essa versão já permite solicitar a identificação, autenticação e autorização das entidades antes de lhes dar acesso aos registos UDDI (PINHO, 2008).

3.6 PROTOCOLOS DE AUTENTICAÇÃO E AUTORIZAÇÃO

A comunidade web tem desenvolvido uma série de protocolos que abordam questões como identidade e confiança (WEBBER et al., 2010). Estes protocolos visam garantir que os sistemas possam interagir de forma segura. O principal benefício de se criar um serviço HTTP é a acessibilidade. Uma vez que uma ampla gama de clientes em plataformas diferentes podem

consumir os serviços HTTP (LAKSHMIRAGHAVAN, 2013).

Para aplicar segurança em aplicações, geralmente e necessário fornecer mecanismos que permitam o cliente se identificar. Para isso, realiza-se o gerenciamento de identidade, que tem por finalidade permitir que os sistemas possam interoperar com segurança,divide-se basicamente em: Autenticação, que é o processo de descobrir a identidade de uma entidade por meio de um identificador e verificar a identidade através da validação de credenciais fornecidas pela entidade (LAKSHMIRAGHAVAN, 2013).E autorização que é o processo que analisa se um usuário após ser autenticado tem permissão para executar uma determinada ação, como por exemplo, acessar um recurso (BRISIS; MANSFIELD; RUNDLE, 2009)(monográfia de gerenciamento de identidade).

3.6.1 *OPENID*

O OpenID é um protocolo SSO(Single Sign-On)que permite a autenticação em diversos websites através de um Uniform Resource Identifier(URI). Ele foi desenvolvido em 2005 pela comunidade open source (RECORDON; REED, 2006). Dentre as caracteríticas inerentes a ele podem ser destacadas:a descentralização e a identidade única, compartilhada com consumidores diferentes. Atualmente, encontra-se na versão 2.0.

OpenID é atraente por causa de sua simplicidade. Com apenas algumas interações, o cliente consegue solicitar e validar uma autenticação um servidor OpenID e interagir com um serviço usando a alegação fundamentada. O fluxo do protocolo *end-to-end* é descrito na figura 6.

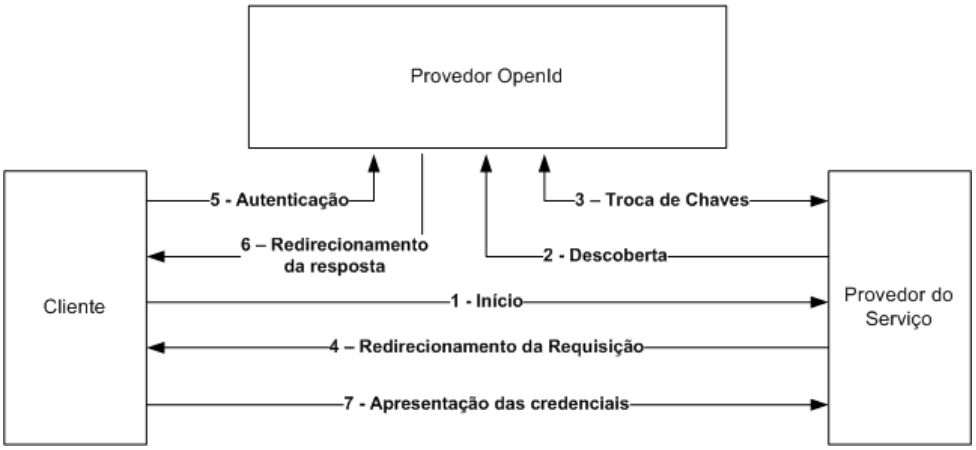

Figura 6: Fluxo do protocolo OpenId, adaptado dede (HARDT, 2012)

1) Início, Um cliente envia um indentificador OpenID que alega possuir.

2) Descoberta, o Provedor do Serviço descobre o provedor OpenID correspondente ao identificador OpenID apresentado pelo cliente.

3) Troca de chaves, segredos são trocadas entre o Provedor do Serviço e o Provedor OpenID.

4) O Provedor do Serviço redireciona o cliente para provedor de OpenID, para que ele possa se autenticar.

5) Autenticação, o cliente se autentica no provedor OpenID. (A maneira em que a autenticação ocorre está fora do escopo OpenID.)

6) O Provedor de OpenID redireciona novamente o Cliente para o Provedor do Serviço.

7) Apresentação das credenciais, finalmente, a carga OpenID contendo a declaração de identidade validada é enviada para O Provedor do Serviço.

3.6.2 *SAML*

O *Security Assertion Markup Language - SAML* é uma especificação padrão para troca de credenciais,*tokens* de segurança, que contém informações de autenticação e autorização, baseadas em *XML*, que visam garantir a interoperabilidade entre diferentes sistemas. Foi desenvolvido em 2005 pela OASIS (OASIS,2005b; OASIS, 2005c). Ele está dividido no seguintes componentes: asserções, protocolos, ligações e perfis, que são descritos a seguir (MADSEN et al., 2005)

Uma Asserção é um conjunto de informações que fornece uma ou mais informações feitas por uma autoridade *SAML*. Ela é dividida em três tipos diferentes: autenticação, atributo e decisão de autorização (MADSEN et al., 2005; NORDBOTTEN, 2009; BERTINO et al., 2010). Uma asserção de autenticação, é aquela que afirma que determinada informação foi autenticada por um meio qualquer em determinado momento. Este tipo de asserção é geralmente emitido por uma autoridade *SAML* denominada de fornecedor de entidades. A sua responsabilidade é a de autenticar consumidor do serviço e manter registros dos dados relacionados com a sua sessão, enquanto esta for válida. Já o atributo é a asserção que contém informações específicas de um determinado cliente. E finalmente a decisão de autorização, que são as informação sobre a decisão de permitir, ou não, o acesso a um determinado recurso por parte do consumidor do serviço.

Os Protocolos, nesse contexto são uma série de mensagens protocolares do tipo pedido/resposta que permitem um provedor de serviços, dentre outras coisas, solicitar a uma

autoridade *SAML*, uma ou mais asserções, pedir a autenticação de um usuário a um provedor de identidades e obter a asserções correspondentes.

Já a ligações, que são aquelas que realizam o mapeamento entre mensagens protocolares e as normas de comunicação entre sistemas, em linhas gerais, define como mensagens *SAML* serão transportadas em cima dos protocolos padrão de mensagens e transporte.

Por fim, os perfis, que são aqueles que define um conjunto de restrições e ou extensões para o suporte da utilização de *SAML* numa determinada aplicação.

Na versão 2.0 do *SAML*, foram adicionadas uma série de novos recursos tais como os pseudónimos, gerenciadores de identidades e de sessões, metadados, encriptação, perfis de atributos, suporte a dispositivos móveis, mecanismos que permitem a aplicação de políticas de privacidade e métodos de pesquisa de provedores de identidades.

3.6.3 *OAUTH*

O *Open Authorization Protocol - OAuth* é um protocolo desenvolvido com o objetivo de solucionar os problemas relacionados com a gestão de identidades entre provedores de serviços. A primeira versão 1.0 foi lançada em 2007, e sua última revisão foi publicada em 2010, sendo especificada no Request For Comments (RFC)5849 (HAMMER-LAHAV, 2010). Em 2012, a versão 2.0 do protocolo, *OAuth 2.0*, foi lançada com objetivo de resolver problemas encontrados na versão 1.0, tais como escalabilidade e complexidade (HARDT, 2012).

Na última versão, 2.0, quatro papéis básicos são definidos e necessários para compreensão do fluxo de execução do protocolo, são eles: proprietário do recurso, que é a entidade que tem o poder de conceder a permissão de acesso, aos seus recursos, às outras entidades; O servidor de recursos, que é o responsável por hospedar e responder às solicitações de acesso a recursos protegidos, usando *tokens* de acesso; Cliente, que é uma aplicação, que realiza solicitações de acesso de recursos protegidos, ao servidor de recursos, em nome do proprietário, dono do recurso, após a obtenção de sua autorização; E o servidor de autorização, que é responsável por emitir *tokens* de acesso aos clientes, após autenticar e obter autorização do proprietário de recursos (HARDT, 2012).

Na maioria dos casos, o papel do servidor de autorização e o servidor de recursos podem ser representados por uma única entidade. A figura 7 representa de forma abstrata o fluxo do protocolo OAuth 2.0 e descreve a interação entre os quatro papéis descritos acima.

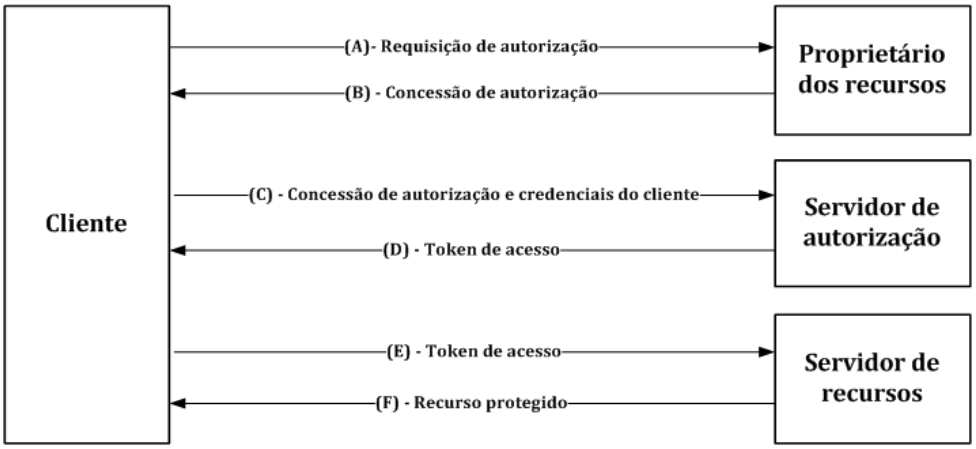

Figura 7: Fluxo do protocolo OAuth 2.0 adaptado de (HARDT, 2012)

a) O cliente solicita a autorização do proprietário do recurso.

b) O cliente recebe uma concessão de autorização, que representa a autorização fornecida pelo proprietário do recurso.

c) O cliente solicita ao servidor de autorização um *token* de acesso que pode ser usado para acessar os recursos protegidos. Durante este processo, o cliente fornece suas credenciais e a concessão de autorização para autenticar-se com o servidor de autorização.

d) O servidor de autorização confirma a validade das credenciais do cliente e da concessão de autorização, se elas forem validas, ele então fornece ao cliente um *token* de acesso.

e) O cliente solicita os recursos protegidos, hospedados no servidor de recursos, apresentando o *token* de acesso.

f) O proprietário do recurso verifica a validade do *token* de acesso e, se válido, ele atende ao pedido.

3.6.4 *TRAUSTH*

3.6.5 *EXTENSIBLE ACCESS CONTROL MARKUP LANGUAGE*

A eXtensible Access Control Markup Language (XACML) foi desenvolvida pelo OASIS Security Services Technical Commitee (SSTC). O padrão XACML define linguagens de marcação que permitem especificar políticas de segurança, requisições e respostas para decisões de controle de acesso, permitindo a organizações utilizarem essas políticas para controlar acesso a conteúdos e informações protegidas.

A linguagem para definição de políticas XACML é utilizada para descrever requisitos gerais de controle de acesso, e possui pontos de extensão para definição de novas funções, tipos de dados e combinações lógicas. As políticas de segurança XACML podem controlar o acesso a informação utilizando identidade de clientes, o método de autenticação de clientes ou ainda uma porta pela qual um cliente se comunica.

O XACML difere de outras linguagens e padrões proprietários primeiramente pelo fato de ser um padrão aberto (Open-Standard). Segundo, por ser genérico, permite que possa ser usado para prover controle de acesso para sistemas completos bem como a um recurso específico. Finalmente, por ser aplicável em conjunto com outros padrões, como o SAML, podendo formar a base para tomada de decisões.

O uso do XACML pode ser feito tanto em ambientes e aplicações proprietárias quanto públicas, podendo facilitar o processo de tomada de decisões e proporcionar interoperabilidade entre diferentes plataformas e domínios.

3.7 CONSIDERAÇÕES FINAIS

Este capítulo apresentou conceitos básicos da arquitetura orientada a serviços. Foram abordados conceitos fundamentais da tecnologia Web Services, REST, sendo apresentadas as definições de sua estrutura. Além disso foram descritos alguns protocolos de autenticação e autorização. No que tange a segurança em SOA, foram apresentados os conceitos básicos de segurança e elencadas algumas das principais vulnerabilidades que afetam a arquitetura orientada a serviços. No próximo capítulo, será descrito o protocolo de autenticação e autorização proposto e sua análise formal, utilizando para isso a lógica BAN.

4 PROTOCOLO DE AUTENTICAÇÃO E AUTORIZAÇÃO PROPOSTO

4.1 *INTRODUÇÃO*

A Polícia Civil do Distrito Federal, diante da necessidade de compartilhar suas informações com órgão parceiros, no intuito de possibilitar que os sistemas possam ser integrados de forma eficiente e principalmente segura busca estabelecer uma arquitetura de referência para a adoção de uma arquitetura orientada a serviços. Essa arquitetura deve primar pela segurança, haja vista a criticidade e sensibilidade das informações que são tratadas no âmbito da PCDF.

Dessa forma, optou-se por adotar a tecnologia *REST* para implementar SOA na instituição. Neste caso, estudos específicos foram realizados com vistas a estabelecer uma política de segurança eficiente que possibilite o fornecimento dos serviços e promova a integração com os órgãos parceiros. Para que isso ocorra, surgiu a necessidade da criação de um protocolo seguro e personalizado de autenticação e autorização que atenda às necessidades da PCDF.

4.2 *REQUISITOS DO PROTOCOLO*

O protocolo de autenticação e autorização proposto será utilizado juntamente com o protocolo HTTPS, aderente a arquitetura REST, de forma que possa permitir que os serviços ofertados pela Divisão de Tecnologia possam ser acessados por um número relativamente grande de clientes. Os requisitos inerentes ao protocolo são descritos nesta seção.

RQ1 A autenticação e autorização será baseada em desafios e resposta, que serão elaborados a partir da apresentação de declarações de identidade (Claims). Tal requisito torna mais flexível o gerenciamento da identidade do usuário, uma vez que possibilita ao administrador desabilitar credenciais que tenham sido comprometidas de forma transparente ao usuário.

RQ2 A política de autenticação e autorização proposta no protocolo será estabelecia por meio de contrato onde serão definidos todas as regras que deverão ser atendidas pelos usuário se pelo fornecedor do serviço. Justificativa ...

Como o padrão para fornecimento dos serviços e o REST, as chamadas ao protocolo deverão ser executadas sobre HTTPS, garantido por um certificado digital que deverá ser assinado e emitido por uma autoridade certificadora confiável. Todos os clientes devem validar o certificado antes de interagir com o servidor.

O protocolo deverá permitir acesso aos serviços apenas ao pessoal autorizado, de forma que a autenticação e autorização, siga padrões definidos na política de segurança. Sendo que para ser autenticado e autorizado o usuário deverá apresentar credenciais válidas. Essas credenciais deverão ser criptografadas, assinadas e enviadas no cabeçalho do protocolo HTTP. Devendo ser escalável em termos de sobrecarga, tamanho do domínio de proteção e de manutenção. Além disso, ele deverá permitir a preservação de privacidade, uma vez que para proteger os clientes e fornecedores de recursos de entidades maliciosas, suas interações deverão revelar o mínimo de informações possíveis.

4.3 *ARQUITETURA DO PROTOCOLO*

Nesta seção, será apresentado uma visão geral do protocolo de comunicação proposto e que será usado pela arquitetura de referência, objeto dessa dissertação. Dessa forma, são detalhados os componentes da arquitetura.

4.3.1 *SEGURANÇA DE SESSÃO*

Toda comunicação entre o cliente e o servidor será realizada utilizando HTTPS *(Hypertext Transfer Protocol over Secure Sockets Layer)*, usando o SSL/TLS para garantir a confidencialidade e integridade para a sessão. Para isso será usado o certificado digital X.509, emitido por uma autoridade de certificação, para encriptar as comunicações e garantir a autenticidade do servidor e do cliente. Devendo os clientes realizar a validação do certificado antes de interagir com o servidor.

4.3.2 *DEFINIÇÃO DO CONTRATO*

Para que qualquer usuário possa ter acesso aos serviços ofertados pela Divisão de Tecnologia da PCDF ele deverá concordar com um contrato prévio de acesso. Devendo

primeiramente ser cadastrado e ter definido quais são seus privilégios de acesso/autorização. Uma vez cadastrado o usuário deverá informar os dados que possam comprovar sua identidade no momento da autenticação de forma que ele possa ser autorizado de acordo com o seus privilégios.

No momento do credenciamento será gerado para o cliente múltiplas credenciais, que serão utilizados no processo de autenticação e autorização, essas informações serão compartilhados entre o **Cliente** o **Servidor de Autenticação e Autorização** e o **Servidor REST**. Além disso, o Contrato poderá ter acesso a multiplos serviços. A a figura 8 representa o relacionamento entre o contrato, credenciais e serviços.

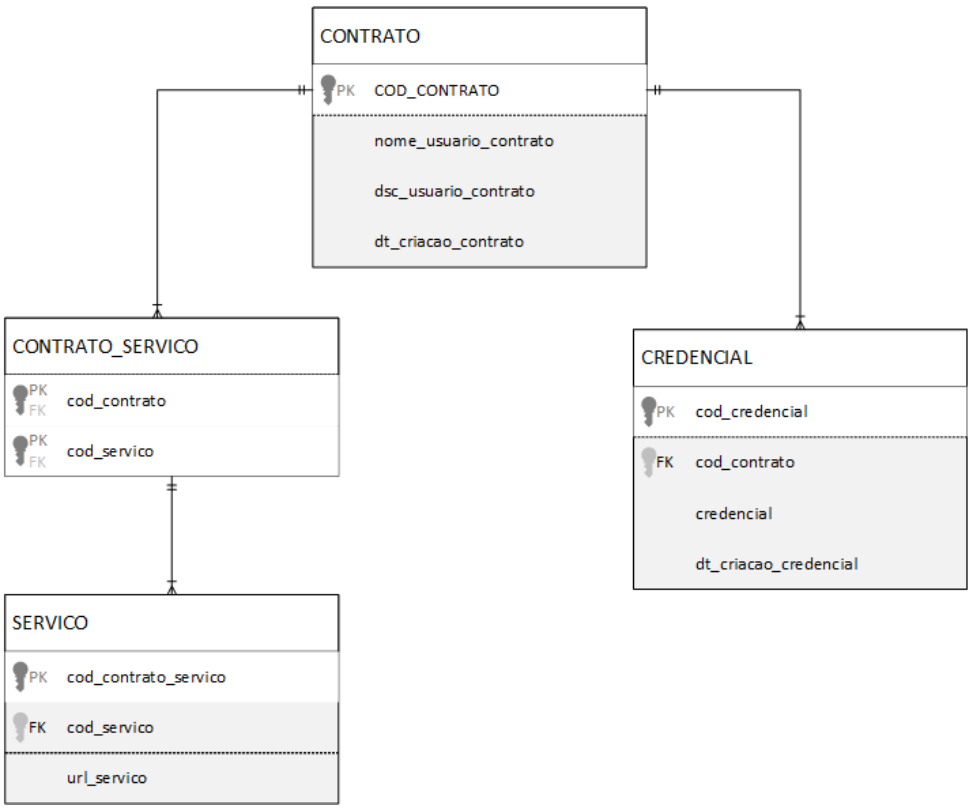

Figura 8: Diagrama de relacionamento entre contrato, credenciais e usuários

4.3.3 *AUTENTICAÇÃO/AUTORIZAÇÃO*

Para ter acesso a API *REST*, referente aos serviços ofertados, o cliente deverá ser autenticado e autorizado a acessar o serviço. Para isso, será usado a autenticação baseada em *tokens* de segurança, que são recipientes de reivindicações da autoridade emissora. Os *tokens* de segurança utilizados serão os *Web Tokens* no formato *JSON*. Esse, ao contrário dos tokens *SAML*, que são baseados em *XML*, são mais compactos e, portanto mais adequados para serem usados em um cabeçalho *HTTP*. Além disso, todas as mensagens deverão ser assinadas

e criptografadas de forma assimétrica. O processo de autenticação e autorização é descrito em dois cenários distintos. No primeiro cenário, representado na figura 9, o Cliente, não está autenticado. Já no segundo ele esta autenticado e possui uma credencial de autorização.

Logo, no primeiro cenário, o cliente não está autenticado, e irá solicitar a autenticação pela primeira vez, conforme descrito a seguir:

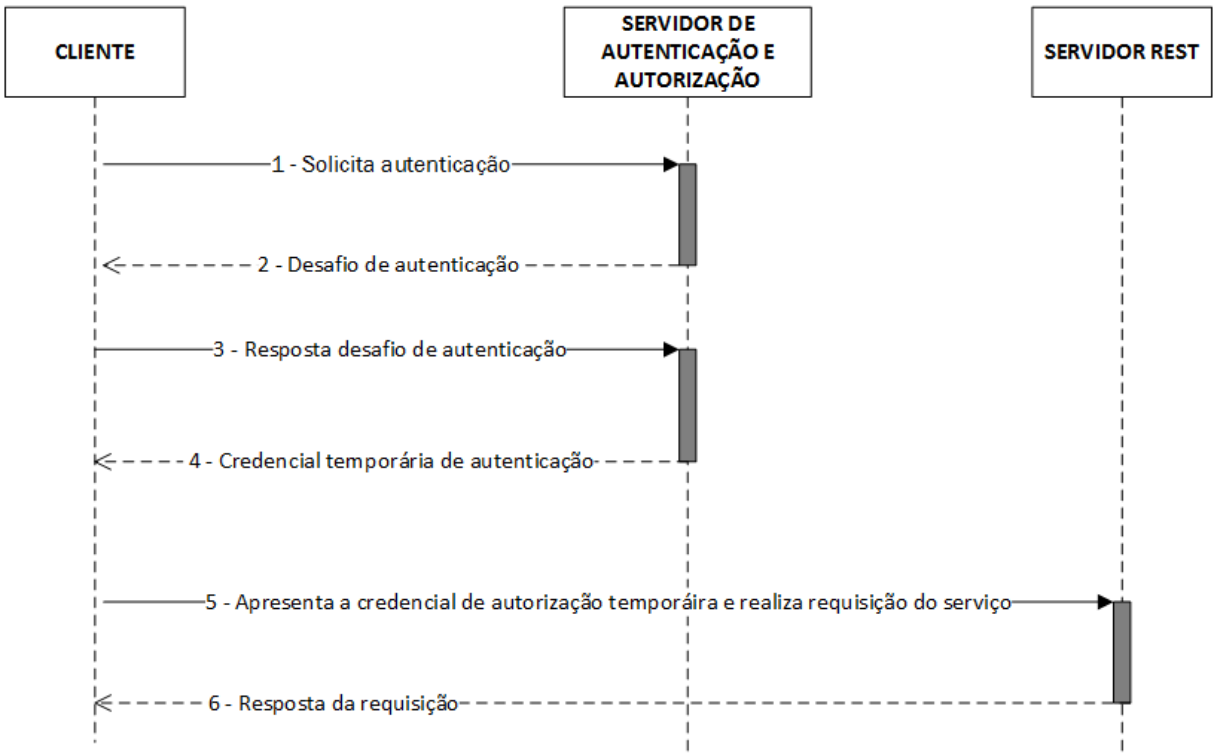

Figura 9: Fluxo do protocolo de autenticação/autorização proposto, 1º cenário.

O protocolo tem início quando o Cliente envia uma solicitação de autenticação ao servidor de Autenticação e Autorização. Esse pedido é realizado por meio de um mensagem (mensagem 1 na Figura 9) que contém um *token* JSON, enviado no cabeçalho HTTP da requisição REST. O *token* contém uma credencial, extraída de forma aleatória da tabela de credenciais do Cliente, e o código do serviço a ser requisitado. O token é assinado digitalmente pelo Cliente e criptografado com a chave pública do servidor de Autenticação e Autorização. Detalhes sobre o conteudo das mensagens trocadas são apresentados na Seção **??**. É importante frisar que tanto o **Cliente** quanto o **servidor de Autenticação e Autorização** possuem as mesmas tabelas de credenciais e de serviços, pois elas são geradas no momento de assinatura do contrato de prestação do serviço.

Ao receber uma solicitação de autenticação, o servidor de Autenticação e Autorização

extrai o Token cifrado com sua chave privada e verifica a autenticidade e integridade da requisição por meio da verificação da assinatura digital do Cliente. Se houver qualquer problema, uma mensagem de erro *HTTP* (código 400) é retornada ao **Cliente**. Caso não haja problemas, procede-se com o processo de validação da credencial informada, que consiste em consultar a credencial em uma base de dados e se a credencial for válida e estiver associada ao **Cliente** o **Servidor de Autenticação e Autorização** gera um desafio de autenticação. Tal desafio consiste em fazer uma busca aleatória à tabela de credenciais e selecionar um código de credencial que esteja associado ao **Cliente**. Em seguida grava-se o desafio, a data e hora de geração do desafio, a resposta e o código do serviço que o **Cliente** está querendo consumir em uma base de dados. O desafio é enviado ao **Cliente** em um token JSON, assinado e criptografado contendo o código do desafio, o código da credencial e um *Timestamp* representando a data e hora de criação.

1) O ***Cliente*** ao receber o desafio, descriptografa o token e verifica a autenticidade e integridade da mensagem, através da verificação da assinatura digital do **Servidor de Autenticação e Autorização**, se houver qualquer problema, o processo de autenticação atual é descartado e inicia-se um novo processo de autenticação. Caso não haja problemas, o **Cliente** verifica e responde o desafio solicitado, enviando-o para o **Servidor de Autenticação e Autorização** por meio de um token JSON. Antes de enviá-lo, o **Cliente** assina e criptografa o referido token.

2) O **servidor de Autenticação e Autorização** recebe a resposta do desafio de autenticação, descriptografa o token e verifica a autenticidade e integridade da requisição por meio da verificação da assinatura digital do **Cliente**. Não ocorrendo problemas, inicia-se o processo de verificação da resposta. A primeira verificação que é realizada refere-se ao tempo de geração do desafio, se a resposta tiver sido enviada em um período de tempo superior a 10 segundos, do tempo que foi gravado na base de dados, ele envia uma mensagem de erro HTTP de usuário não autenticado. Caso o intervalo de tempo seja menor ou igual há 10 segundos, ele procede com a verificação do desafio que consiste em realizar uma consulta na tabela de desafios verificando se a resposta dada é a mesma que a esperada, e se o **Cliente** tem privilégios necessários para consumir o serviço requisitado.

   Se a resposta for positiva, o **servidor de Autenticação e Autorização** gera uma credencial de autorização temporária para o serviço solicitado. Ela é gravada em uma tabela de credencias de autorização temporária juntamente com a data e hora de criação, data de expiração e o código do cliente. A tabela de credencias de autorização temporária será acessada pelo **Servidor REST** para verificar quais privilégios a entidade requisitante do

serviço tem acesso e se ela está autenticada. O token de autorização é enviado, assinado e criptografado ao **Cliente**. Já, com a resposta do desafio negativa, o cliente recebe uma mensagem de erro HTTP de usuário não autenticado.

É importante frisar que a credencial de autorização temporária será gerada apenas para o serviço que o ***Cliente*** tenha solicitado e possua o privilégio de acesso para utilizá-la. Ela será válida por um período de 20 (vinte) minutos.

3) O **Cliente** recebe o token JSON com a credencial de autorização temporária, descriptografa e verifica a sua autenticidade e integridade, por meio da verificação da assinatura digital do **Servidor de Autenticação e Autorização**. Ocorrendo problemas, o processo de autenticação atual é descartado, iniciando-se um novo processo. Caso não haja problemas, o **Cliente** verifica a data e hora de validade da credencial de autorização temporária para saber se ela é válida. Confirmada sua validade, ele envia ao ***Servidor REST*** a requisição do serviço que deseja consumir juntamente com a credencial de autorização temporária. O token de autenticação e autorização é assinado e criptografado, sendo enviado no cabeçalho da requisição.

4) O **Servidor REST** recebe o token de autenticação e autorização, faz a verificação na tabela de credenciais temporárias para saber se o **Cliente** está autenticado e se tem privilégios de autorização para consumir o serviço. Caso não haja problemas, o ***Cliente*** recebe os dados referentes à sua requisição. Havendo qualquer problema ele recebe uma mensagem de erro HTTP de usuário não autorizado.

Já no segundo cenário,O ***Cliente***, já possui uma credencial de autorização temporária, neste caso ele deverá apresentá-la sempre que desejar consumir algum serviço que ele tenha acesso conforme descrito a seguir:

1) O **Cliente** envia um token contendo uma credencial temporária no cabeçalho da requisição do serviço que deseja consumir, ao ***Servidor de REST***.

2) O **Servidor de REST**, recebe o token de autenticação e autorização, faz a verificação na tabela de credenciais temporárias para saber se o Cliente possui uma credencial válida, em caso positivo ele verifica quais são os privilégios de autorização da credencial, se ele tiver permissão para acessar o serviço, sua requisição é atendida.

Caso a credencial não seja válida ou tenha expirado o **Cliente** é redirecionado para o **servidor de Autenticação e Autorização** para que possa se autenticar novamente e obter uma nova credencial conforme descrito no primeiro cenário.

4.4 FORMALIZAÇÃO DO PROTOCOLO

Os protocolos estão sujeitos a ocorrência de erros em qualquer fase de seu projeto (especificação, construção e verificação). Por isso, com certa frequência são descobertas falhas em protocolos publicados e utilizados por vários anos. Como exemplo, o protocolo Needham & Schroeder () foi utilizado durante 4 anos até que Denning e Sacco demonstraram que ele estava sujeito ao tipo de ataque intitulado *homem-do-meio* () e propuseram um protocolo alternativo (**??**). Porém, em 1994 Abadi e Needham demonstraram que este protocolo também possuía falhas ().

O emprego de avaliações mais formais na área de criptografia não é recente. Grande parte dos trabalhos nesta área foram desenvolvidos na década de 90 (MEADOWS, 1995). Estes métodos permitem fazer uma análise completa do protocolo criptográfico e sua função principal é especificar se os objetivos propostos pelos autores são alcançados. Muitas vezes um protocolo é construído para realizar a distribuição segura de uma chave de sessão e quando analisado verifica-se que essa meta não é atingida. *Neste trabalho, o protocolo proposto foi descrito formalmente, utilizando a lógica BAN, com o intuito de favorecer a comunicação e o entendimento utilizando uma linguagem mais precisa. Além disso, a propriedade de terminação com a geração da credencial temporária de autorização foi verificada com um programa escrito em Prolog (ver Apêndice XYZ).*

4.5 LÓGICA BAN

A lógica BAN foi desenvolvida por Burrows, Abadi eNeedham em 1989. É a primeira lógica a analisar formalmente os protocolos criptográficos,principalmente os de autenticação e distribuição de chaves (BURROWS et al., 1990).

É a lógica mais popular na literatura para a análise de crenças e de conhecimento entre os participantes dos protocolos criptográficos.

4.5.1 NOTAÇÃO BÁSICA

Na lógica BAN, existem vários tipos distintos de objetos tais como entidades ou partes que se comunicam, chaves de criptografia e fórmulas lógicas. Uma fórmula lógica é uma versão idealizada da mensagem original, podendo ser referenciada como uma declaração lógica. Normalmente, os símbolos A, B e S denotam entidades (*principles*); Kab, Kas e Kbs denotam chaves compartilhadas; Ka, Kb e Ks denotam chaves públicas e $Ka^{-1}, Kb^{-1} e Ks^{-1}$ denotam

as chaves privadas dos participantes. Já Na, Nb e Ns são os identificadores gerados pelos participantes. As construções mais frequentemente utilizadas são apresentadas na tabela abaixo:

Expressão	***Leitura/Significado***
$P \mid\equiv X$	P **acredita** X: P crê em X, ou P acredita que X é verdadeiro
$P \triangleleft X$	P **recebeu** X:Alguém enviou uma mensagem para P contendo X ou de outra forma P recebeu X
$P \mid\sim X$	P **disse** X:P disse uma vez X. A entidade P em algum momento enviou uma mensagem incluíndo a declaração X.
$P \Rightarrow X$	P **controla** X:P tem jurisdição sobre X. Onde P é uma autoridade sobre X é deve ser confiável
#(X)	novo(X): a fórmula novo X, ela não foi usada numa mensagem anterior à execução protocolo atual.
$P \overset{k}{\longleftrightarrow} Q$	(lê-se "k é uma chave satisfatória para P e Q"). A chave k nunca será descoberta por qualquer participante, exceto por P, Q ou por alguém em quem eles confiam
$\{X\}_K$	fórmula X foi cifrada com a chave K. As mensagens cifradas somente são legíveis e verificáveis pelo possuidor da chave
$P \overset{k}{\Longleftrightarrow} Q$	k é o segredo compartilhado entre P e Q e possivelmente também com as entidades de confiança deles. Somente P e Q podem usar k para provar suas identidades
$\overset{K}{\mapsto} P$:	k é a chave pública de P

Tabela 1: Notação básica Lógica BAN, adaptação de (BURROWS et al., 1990).

4.5.2 *POSTULADOS LÓGICOS*

No estudo de protocolos de segurança é importante diferenciar o tempo das demonstrações ou eventos. Haja vista que se isso não for observado, problemas , como a não detecção do reenvio de mensagens, poderá acontecer. Segundo (BURROWS et al., 1990), a lógica BAN trata dessa distinção dividindo-a em duas épocas: presente, que é o tempo durante a execução do protocolo, e o passado, que refere-se às mensagens enviadas antes da execução do protocolo, o que faz com que elas sejam rejeitadas, uma vez que não são confiáveis . Essa divisão de tempo é suficiente para facilitar o entendimento de como a lógica pode ser manipulada. Para realizar a análise dos protocolos de segurança a lógica BAN possui uma serie de postulados lógicos, regras (BURROWS et al., 1990). Alguns deles são descritos a seguir:

a) Regra de significado da mensagem. Esta regra faz parte da interpretação das mensagens

$$\frac{P \textbf{ acredita } P \overset{k}{\longleftrightarrow} Q, \quad P \textbf{ recebeu } \{X\}_k}{P \textbf{ acredita } Q \textbf{ disse } X}$$

Se P acredita que k é uma chave satisfatória para se comunicar com Q e se P recebeu a mensagem X cifrada com a chave k, então P acredita que Q uma vez disse X

b) Regra de verificação do identificador. Essa regra verifica se a mensagem é recente, se foi enviada durante a execução atual do protocolo e conseqüentemente, se o emissor acredita nela.

$$\frac{P \textbf{ acredita } \textit{novo}(\mathrm{X}), \quad P \textbf{ acredita } Q \textbf{ disse } X}{P \textbf{ acredita } Q \textbf{ acredita } X}$$

Se P acredita que X é novo e P acredita que em algum momento Q disse X,então P também acredita que Q acredita em X.

c) Regra da jurisdição. Esta regra representa a confiança e a autoridade de uma entidade sobre as declarações.

$$\frac{P \textbf{ acredita } Q \textbf{ controla } X, \quad P \textbf{ acredita } Q \textbf{ acredita } X}{P \textbf{ acredita } X}$$

Se P acredita que Q tem jurisdição sobre a declaração X e P acredita que Q acredita em X, então P confia na declaração X.

Estes são alguns dos principais postulados utilizados na construção da análise formal de protocolos criptográficos. A utilização destas regras juntamente com as notações descritas na sessão anterior possibilita que a crença dos participantes de um protocolo possa ser declarada.

4.6 *ANÁLISE FORMAL DO PROTOCOLO PROPOSTO*

Nesta sessão será realizado a análise formal do protocolo de autenticação e autorização proposto. Para isso será utilizada a lógica BAN.

Na lógica BAN a análise de um protocolo é dividido em quatro etapas: A idealização do protocolo, que é originada a partir do protocolo original. As Hipóteses, que são a suposições a respeito do estado inicial, nessa etapa são definidas as fórmulas lógicas estão ligados às declarações do protocolo, as declarações que são afirmações sobre o estado do sistema. As regras de inferência ou postulados lógicos que são aplicadas para os pressupostos e as afirmações a fim de descobrir as crenças detidas pelas partes no protocolo.

4.6.1 *IDEALIZAÇÃO DO PROTOCOLO*

A seguir será descrito a idealização do protocolo proposto. A figura 9. representa o fluxo de troca de mensagens executado pelo protocolo.

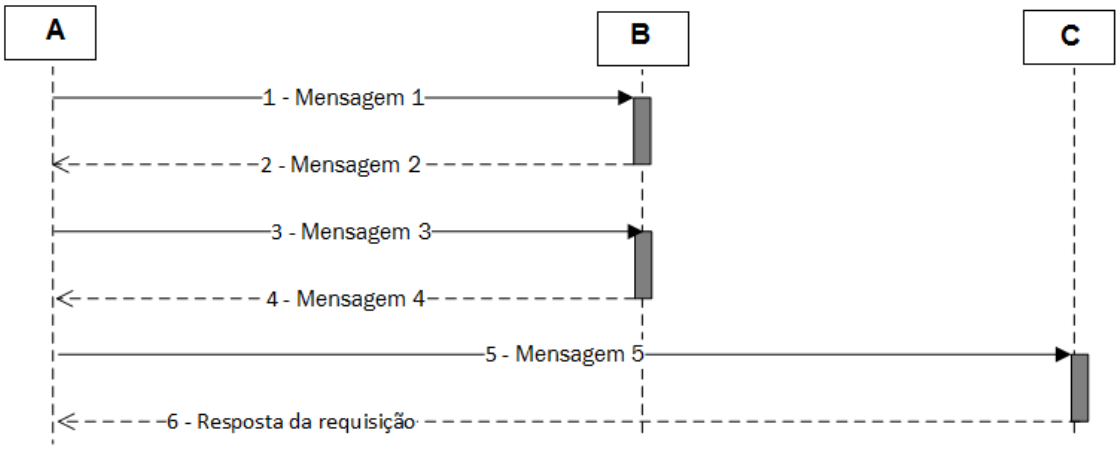

Figura 10: Diagrama de idealização do protocolo de autenticação/autorização proposto

Dessa forma, para modelar o protocolo foram utilizadas as seguintes entidades participantes, que são descritas a seguir:

- A = CLIENTE
- B= Servidor REST
- C = Servidor de Autenticação e Autorização
- $Urlrequest_A$ = Url que contém a requisição de serviço de A (Get, Put, Post ou Delete)
- Ka = Chave pública da entidade A;
- Kb = Chave pública da entidade B;
- Kc = Chave pública da entidade C;
- Ka^{-1} = Chave privada da entidade A;
- Kb^{-1} = Chave privada da entidade B;

- Kc^{-1} = Chave privada da entidade C;
- ta = Timestamp emitido pela entidade C;
- tc = Timestamp emitido pela entidade A;
- $Cred_A$ = Credencial utilizada pela entidade A;
- $Cod_{Contrato_A}$ = Código do contrato que identifica a entidade A;
- Cod_{Srv_A} = Código que identifica o serviço que a entidade A está requerendo;
- $Cod_{Desafio_C}$ = Código único que identifica o desafio gerado pela entidade C para a entidade A;
- $Credencial_{Alea}$ = Código da Credencial da entidade A selecionada de forma aleatória pela entidade C e enviada como desafio à entidade A;
- $Credencial_{Solicitada}$ = Credencial da entidade A enviada como resposta à solicitação da entidade C;
- H_{SHA3} = Representa o Hashing de uma mensagem utilizando o algoritmo SHA3;
- Msg_{AC} = Representa o resumo da mensagem enviada pela entidade A a entidade C;
- Msg_{CA} = Representa o resumo da mensagem enviada pela entidade C a entidade A;
- Msg_{AB} = Representa o resumo da mensagem enviada pela entidade A a entidade B;
- Exp_{credA} = Data/hora de expiração da Credencial Temporária de autorização e autenticação;
- $CredAutA$ = Credencial Temporária de autorização e autenticação gerada para a entidade A

Idealização do protocolo proposto:

1. Mensagem 1: $A \longrightarrow C : \{ta, Cred_A, Cod_{Contrato_A}, Cod_{Srv_A}, H_{SHA3}\{Msg_{AC}\}_{Ka^{-1}}\}_{Kc}$.
2. Mensagem 2: $C \longrightarrow A : \{tc, \#Cod_{Desafio_C}, Credencial_{Alea}, H_{SHA3}\{Msg_{CA}\}_{Kc^{-1}}\}_{Ka}$.
3. Mensagem 3: $A \longrightarrow C : \{ta, Cod_{Desafio_C}, Credencial_{Solicitada}, H_{SHA3}\{Msg_{AC}\}_{Ka^{-1}}\}_{Kc}$.
4. Mensagem 4: $C \longrightarrow A : \{tc, Exp_{credA}, \#(A \overset{CredAutA}{\Longleftrightarrow} C), H_{SHA3}\{Msg_{CA}\}_{Kc^{-1}}\}_{Ka}$.
5. Mensagem 5: $A \longrightarrow B : \{ta, (A \overset{CredAutA}{\Longleftrightarrow} C), H_{SHA3}\{Msg_{AB}\}_{Ka^{-1}}\}_{Ka}, Urlrequest_A$.

4.6.2 *SUPOSIÇÕES*

Todas as suposições a seguir são baseadas em um canal seguro de comunicação SSL/TSL, onde tanto o receptor quanto o emissor do serviço são conhecidos e autenticados usando-se certificados digitais X509.

1. A acredita $\overset{Kc}{\mapsto} C$.
2. B acredita $\overset{Ka}{\mapsto} A$.
3. C acredita $\overset{Ka}{\mapsto} A$.
4. A acredita $\#tc$.
5. B acredita $\#ta$.
6. C acredita $\#ta$.
7. A acredita C Controla $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$.
8. B acredita C Controla $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$.
9. A acredita C Acredita $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$.
10. B acredita C Acredita $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$.
11. A acredita $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$.
12. B acredita $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$.

representação por simbologia

1. $A \mid\equiv \overset{Kc}{\mapsto} C$.
2. $B \mid\equiv \overset{Ka}{\mapsto} A$.
3. $C \mid\equiv \overset{Ka}{\mapsto} A$.
4. $A \mid\equiv \#tc$.
5. $B \mid\equiv \#ta$.
6. $C \mid\equiv \#ta$.
7. $A \mid\equiv \#Exp_{credA}$.

8. $A \mid\equiv C \Rightarrow \#(A \overset{CredAutA}{\Longleftrightarrow} C)$.
9. $B \mid\equiv C \Rightarrow \#(A \overset{CredAutA}{\Longleftrightarrow} C)$.
10. $A \mid\equiv C \mid\equiv \#(A \overset{CredAutA}{\Longleftrightarrow} C)$.
11. $B \mid\equiv C \mid\equiv \#(A \overset{CredAutA}{\Longleftrightarrow} C)$.
12. $A \mid\equiv \#(A \overset{CredAutA}{\Longleftrightarrow} C)$.
13. $B \mid\equiv \#(A \overset{CredAutA}{\Longleftrightarrow} C)$.

4.6.3 *PROVAS*

1. **Mensagem 1**: $A \longrightarrow C : \{ta, Cred_A, Cod_{Contrato_A}, Cod_{Srv_A}, H_{SHA3}\{Msg_{AC}\}_{Ka^{-1}}\}_{Kc}$.

   C recebeu $\{ta, Cred_A, Cod_{Contrato_A}, Cod_{Srv_A}, H_{SHA3}\{Msg_{AC}\}_{Ka^{-1}}\}_{Kc}$

   C acredita **A** disse $H_{SHA3}\{Msg_{AC}\}$

   C acredita **A** disse ta

   C acredita $Cred_A, Cod_{Contrato_A}, Cod_{Srv_A}$

   representação por símbolos

   $C \triangleleft \{ta, Cred_A, Cod_{Contrato_A}, Cod_{Srv_A}, H_{SHA3}\{Msg_{AC}\}_{Ka^{-1}}\}_{Kc}$

   $C \mid\equiv A \mid\sim H_{SHA3}\{Msg_{AC}\}$

   $C \mid\equiv A \mid\sim \#ta$

   $C \mid\equiv Cred_A, Cod_{Contrato_A}, Cod_{Srv_A}$

   C Recebe a fórmula $\{ta, Cred_A, Cod_{Contrato_A}, Cod_{Srv_A}, H_{SHA3}\{Msg_{AC}\}_{Ka^{-1}}\}_{Kc}$, e a decifra usando sua chave privada, em seguida aplicando a regra do significado da mensagem na suposição 3 usando a função $H_{SHA3}\{Msg_{AC}\}$ confirma a autenticidade e integridade da mensagem. Por fim, aplica a regra de verificação do identificador na suposição 6 usando a fórmula ta para obter os dados: $Cred_A, Cod_{Contrato_A}, Cod_{Srv_A}$ da entidade **A**.

   Resultado: C obtém os dados da entidade A $Cred_A, Cod_{Contrato_A}, Cod_{Srv_A}$

2. **Mensagem 2**: $C \longrightarrow A : \{tc, \#Cod_{Desafio_C}, Credencial_{Alea}, H_{SHA3}\{Msg_{CA}\}_{Kc^{-1}}\}_{Ka}$.

   A recebeu $\{tc, \#Cod_{Desafio_C}, Credencial_{Alea}, H_{SHA3}\{Msg_{CA}\}_{Kc^{-1}}\}_{Ka}$

   A acredita **A** disse $H_{SHA3}\{Msg_{CA}\}$

A acredita **A** disse tc

A acredita $\#Cod_{Desafio_C}, Credencial_{Alea}$

representação simbólica:

$\mathbf{A} \triangleleft \{tc, \#Cod_{Desafio_C}, Credencial_{Alea}, H_{SHA3}\{Msg_{CA}\}_{Kc^{-1}}\}_{Ka}$

$\mathbf{A} \mid\equiv \mathbf{C} \mid\sim H_{SHA3}\{Msg_{CA}\}$

$\mathbf{A} \mid\equiv \mathbf{C} \mid\sim tc$

$\mathbf{A} \mid\equiv \#Cod_{Desafio_C}, Credencial_{Alea}$

A Recebe a fórmula $\{tc, \#Cod_{Desafio_C}, Credencial_{Alea}, H_{SHA3}\{Msg_{CA}\}_{Kc^{-1}}\}_{Ka}$, e a decifra usando sua chave privada, em seguida aplicando a regra do significado da mensagem na suposição 1 usando a função $H_{SHA3}\{Msg_{CA}\}$ confirma a autenticidade e integridade da mensagem. Por fim, aplica a regra de verificação do identificador na suposição 4 e usando a fórmula tc para obter os dados: $\#Cod_{Desafio_C}, Credencial_{Alea}$, referentes ao desafio de autenticação gerado pela entidade **C**.

Resultado: **A** obtém o desafio de autenticação gerado pela entidade **C** $\#Cod_{Desafio_C}, Credencial_{Alea}$

3. **Mensagem 3**: $A \longrightarrow C : \{ta, Cod_{Desafio_C}, Credencial_{Solicitada}, H_{SHA3}\{Msg_{AC}\}_{Ka^{-1}}\}_{Kc}$.

   C recebeu $\{ta, Cod_{Desafio_C}, Credencial_{Solicitada}, H_{SHA3}\{Msg_{AC}\}_{Ka^{-1}}\}_{Kc}$

   C acredita **A** disse $H_{SHA3}\{Msg_{AC}\}$

   C acredita **A** disse ta

   C acredita $Cod_{Desafio_C}, Credencial_{Solicitada}$

   representação por símbolos

   $C \triangleleft \{ta, Cod_{Desafio_C}, Credencial_{Solicitada}, H_{SHA3}\{Msg_{AC}\}_{Ka^{-1}}\}_{Kc}$

   $C \mid\equiv A \mid\sim H_{SHA3}\{Msg_{AC}\}$

   $C \mid\equiv A \mid\sim \#ta$

   $C \mid\equiv Cod_{Desafio_C}, Credencial_{Solicitada}$

   C Recebe a fórmula $\{ta, Cod_{Desafio_C}, Credencial_{Solicitada}, H_{SHA3}\{Msg_{AC}\}_{Ka^{-1}}\}_{Kc}$, e a decifra usando sua chave privada, em seguida aplicando a regra do significado da mensagem na suposição 3 usando a função $H_{SHA3}\{Msg_{AC}\}$ confirma a autenticidade e integridade da mensagem. Por fim, aplica a regra de verificação do identificador na suposição 6 usando a fórmula ta para obter os dados: $Cred_A, Cod_{Contrato_A}, Cod_{Srv_A}$ e validar a resposta enviada e autenticar a entidade **A**.

Resultado: **C** autentica a entidade **A**.

4. **Mensagem 4**: $C \longrightarrow A : \{tc, Exp_{credA}, \#(A \overset{CredAutA}{\Longleftrightarrow} C), H_{SHA3}\{Msg_{CA}\}_{Kc^{-1}}\}_{Ka}$.

   A recebeu $\{tc, Exp_{credA}, \#(A \overset{CredAutA}{\Longleftrightarrow} C), H_{SHA3}\{Msg_{CA}\}_{Kc^{-1}}\}_{Ka}$

   A acredita **C** disse $H_{SHA3}\{Msg_{CA}\}$

   A acredita **C** disse tc

   A acredita **C** disse Exp_{credA}

   A acredita **C** disse $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$

   A acredita **C** controla $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$

   A acredita $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$

   representação simbólica:

   $\mathbf{A} \triangleleft \{tc, Exp_{credA}, \#(A \overset{CredAutA}{\Longleftrightarrow} B), H_{SHA3}\{Msg_{CA}\}_{Kc^{-1}}\}_{Ka}$

   $\mathbf{A} \mid\equiv \mathbf{C} \mid\sim H_{SHA3}\{Msg_{CA}\}$

   $\mathbf{A} \mid\equiv \mathbf{C} \mid\sim tc$

   $\mathbf{A} \mid\equiv \mathbf{C} \mid\sim Exp_{credA}$

   $\mathbf{A} \mid\equiv \mathbf{C} \Rightarrow \#(A \overset{CredAutA}{\Longleftrightarrow} C)$

   $\mathbf{A} \mid\equiv \mathbf{C} \mid\equiv \#(A \overset{CredAutA}{\Longleftrightarrow} C)$

   $\mathbf{A} \mid\equiv \#(A \overset{CredAutA}{\Longleftrightarrow} C)$

   A Recebe a fórmula $\{tc, Exp_{credA}, \#(A \overset{CredAutA}{\Longleftrightarrow} B), H_{SHA3}\{Msg_{CA}\}_{Kc^{-1}}\}_{Ka}$, e a decifra usando sua chave privada, em seguida aplicando a regra do significado da mensagem na suposição 1 usando a função $H_{SHA3}\{Msg_{CA}\}$ confirma a autenticidade e integridade da mensagem. Ela aplica a regra de verificação do identificador nas suposições 4 e 7 usando as fórmulas tc e Exp_{credA}. E, finalmente, aplicando a regra da jurisdição, nas suposições 7 e 9, obtém a credencial de autorização temporária $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$.

   Resultado: **A** obtém a credencial de autorização temporária $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$.

5. **Mensagem 5**: $A \longrightarrow B : \{ta, (A \overset{CredAutA}{\Longleftrightarrow} C), H_{SHA3}\{Msg_{AB}\}_{Ka^{-1}}\}_{Ka}, Urlrequest_A$.

   B recebeu $\{ta, (A \overset{CredAutA}{\Longleftrightarrow} C), H_{SHA3}\{Msg_{AB}\}_{Ka^{-1}}\}_{Ka}, Urlrequest_A$

   B acredita **A** disse $H_{SHA3}\{Msg_{AB}\}$

   B acredita **A** disse ta

   B acredita **A** disse $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$

B acredita **A** controla $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$

B acredita $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$

representação simbólica:

$\mathbf{B} \triangleleft \{ta, (A \overset{CredAutA}{\Longleftrightarrow} C), H_{SHA3}\{Msg_{AB}\}_{Ka^{-1}}\}_{Ka}, Urlrequest_A$

$\mathbf{B} \mid\equiv \mathbf{A} \mid\sim H_{SHA3}\{Msg_{AB}\}$

$\mathbf{B} \mid\equiv \mathbf{A} \mid\sim ta$

$\mathbf{B} \mid\equiv \mathbf{A} \Rightarrow \#(A \overset{CredAutA}{\Longleftrightarrow} C)$

$\mathbf{B} \mid\equiv \mathbf{A} \mid\equiv \#(A \overset{CredAutA}{\Longleftrightarrow} C)$

$\mathbf{B} \mid\equiv \#(A \overset{CredAutA}{\Longleftrightarrow} C)$

B Recebe a fórmula $\{ta, (A \overset{CredAutA}{\Longleftrightarrow} C), H_{SHA3}\{Msg_{AB}\}_{Ka^{-1}}\}_{Ka}, Urlrequest_A$, e a decifra, usando sua chave privada. Em seguida aplicando a regra do significado da mensagem na suposição 2 usando a função $H_{SHA3}\{Msg_{AB}\}$ confirma a autenticidade e integridade da mensagem. Ela aplica a regra de verificação do identificador na suposições 5 usando as fórmulas ta. E, finalmente, aplicando a regra da jurisdição, nas suposições 8 e 10, obtém a credencial de autorização temporária $\#(A \overset{CredAutA}{\Longleftrightarrow} C)$ e autoriza a entidade **A** a consumir a requisição $Urlrequest_A$.

Resultado : **B** autoriza **A** a partir da nova credencial $(A \overset{CredAutA}{\Longleftrightarrow} C)$ a consumir a requisição $Urlrequest_A$

5 AVALIAÇÃO E ANÁLISE DE DESEMPENHO

Após a definição da estrutura da arquitetura de referencia SOA, conforme citado no capitulo anterior, que utiliza web services REST, e onde foi proposto um protocolo de autenticação e autorização, que tem por finalidade prover segurança. Se faz necessário realizar experimentos que possibilitem a mensuração do impacto desta arquitetura na infraestrutura da PCDF. Desta forma, este capítulo apresenta a avaliação de desempenho da referida arquitetura.

5.1 *TESTES DE DESEMPENHO*

Testes de desempenho são de?nidos como uma investigação técnica realizada para determinar a capacidade de resposta, produtividade, con?abilidade e ou escalabilidade de um sistema sob uma determinada carga de trabalho (MEIER et al., 2007). A análise de desempenho possibilita identificar problemas que geralmente são encontrados em sistema computacionais. Tais problemas podem ser agrupados em tópicos de: comparação de sistemas, identificação de gargalos, caracterização de cargas de trabalho, configuração de sistemas e a previsão de desempenho (Braghetto, 2011). Sendo assim, a análise de desempenho da arquitetura propostas está dividida em dois estudos de caso, conforme descrito a seguir.

5.2 *ESTUDO DE CASO 1*

Nesta seção será apresentada a metodologia empregada para a realização do teste de desempenho bem como os resultados e análise obtidos com sua aplicação.

Para este estudo de caso inicial foram utilizados dois web services, o primeiro é traz informações sobre ocorrências policiais, tais como quais tipos de ocorrências criminais estão registras, dados gerais da vítima, autor dentre outros. Já segundo, apresenta informações sobre procedimentos policiais, como por exemplo, informações sobre mandado de prisões, medidas cautelares, pedidos de prisão dentre outros. Ambos são importantes e amplamente utilizados pela PCDF e por órgão parceiros. Eles foram construídos utilizando a tecnologia SOAP,

desenvolvidos com a ferramenta Visual Studio 2005.

Para a realização da análise foram desenvolvidos outros dois web services, que implementam os mesmos serviços, só que utilizando a tecnologia REST. Com isso, busca-se realizar uma comparação de desempenho entre as duas tecnologias. Para isso, veri?car-se-á o tempo médio de resposta.

5.2.1 *CONFIGURAÇÃO DO AMBIENTE DE TESTE*

Ambiente de teste foi o da Divisão de Tecnologia da Policia Civil do Distrito Federal, uma vez que se busca resultados mais próximos da realidade hoje vivenciado pela PCDF.

Para a sua realização dos testes foram utilizados configurações distintas de computadores, conforme tabela abaixo.

Máquina	***Quantidade***	***Configuração***
Provedor de Serviço	1	Servidor Intel com 4 processadores Intel Xeon E7 4870 2,4 GHz, 8 Gb de RAM e 120 Gb de HD
Provedor de Banco de Dados	1	Servidor Intel com 4 processadores Intel Xeon E7 4870 2,4 GHz, 8 Gb de RAM e 120 Gb de HD
Cliente	100	Desktop HP Intel Core I5-2500 3,3 GHz, 4 Gb RAM, 500 HD

Tabela 2: Ambiente de configuração estudo de caso 1

É importante frisar que essa estrutura é utilizada atualmente para disponibilização dos serviços. Os serviços estão publicados no provedor de serviço que utilizam o sistema operacional Windows Server 2008 R2 Enterprise Edition x64, utilizando o ISS 7.0 como servidor Web. Já o provedor de Banco de dados é o SQL 2008 r2 que utiliza o mesmo sistema operacional.

5.2.2 *PLANEJAMENTO DO EXPERIMENTO*

Essa é uma fase de extrema importância, uma vez que nela será realizada a identificação dos cenários de uso, a determinação da variabilidade entre os usuários, a identificação e geração dos dados de teste e a especificação das métricas que serão coletadas. Em síntese serão criadas as bases e perfis para cargas de trabalho (MEIER et al., 2007). Os termos que mais se destacam e que são utilizados durante a etapa de projeto e experimentos de

um teste de desempenho em são: as Variáveis de Resposta, Fatores, Níveis e Interação (Jain, 1991).

As Variáveis de resposta são as medidas de desempenho do sistema e representam o resultado de um experimento. Os Fatores, são termos relacionados com variáveis que influenciam a resposta do sistema. Já os Níveis tem relação direta com os valores que um fator pode assumir. Finalmente a Interação, que aponta a dependência entre os fatores que foram avaliados. Dessa forma, com intuito de verificar qual tecnologia tem melhor desempenho, optou-se por realizar um teste automatizado utilizando as ferramentas SOAPUI e ou JMeter. Sendo assim, a abordagem utilizada foi a da utilização de dois fatores: tecnologia adotada (REST/SOAP) e o número de clientes (10, 100) com dois níveis respectivamente. Para realizar o experimento serão realizados 100 requisições por cada cliente, sendo adotado um índice de 95% para intervalo de confiança. Já a variável de resposta adotada, será o tempo médio de resposta das requisições.

5.3 *ESTUDO DE CASO 2*

Nesta seção será realizado um estudo cuja finalidade será a de verificar o desempenho do web service REST com a aplicação do protocolo de segurança proposto na arquitetura de referência, objeto dessa dissertação, que envolve a aplicação de criptografia e assinatura e a utilização do Web service REST sem nenhum mecanismo de segurança.

Os web services adotados serão os mesmos definidos no estudo de caso 1, web service de ocorrências criminais, que retornam informações gerais do sistema de ocorrência e o de procedimentos policiais, que retorna informações criminais do sistema PROCED.

5.3.1 *CONFIGURAÇÃO DO AMBIENTE DE TESTE*

Ambiente de teste foi o da Divisão de Tecnologia da Policia Civil do Distrito Federal, uma vez que se buscam resultados mais próximos da realidade hoje vivenciados pela PCDF.

Para a sua realização dos testes foram utilizados configurações distintas de computadores, conforme tabela abaixo. Essa configuração é idêntica a utilizada no estudo de caso 1.

Com a finalidade de se automatizar o teste de desempenho e de emular os clientes foi adotada a utilização da ferramenta de teste SOAPUI ou JMeter.

Máquina	*Quantidade*	*Configuração*
Provedor de Serviço	1	Servidor Intel com 4 processadores Intel Xeon E7 4870 2,4 GHz, 8 Gb de RAM e 120 Gb de HD
Provedor de Banco de Dados	1	Servidor Intel com 4 processadores Intel Xeon E7 4870 2,4 GHz, 8 Gb de RAM e 120 Gb de HD
Cliente	100	Desktop HP Intel Core I5-2500 3,3 GHz, 4 Gb RAM, 500 HD

Tabela 3: Ambiente de configuração estudo de caso 2

5.3.2 *PLANEJAMENTO DO EXPERIMENTO*

O planejamento adotado neste estudo de caso procurou abordar combinações considerando os fatores e níveis. Sendo assim, a abordagem utilizada foi a da utilização de três fatores: Utilização de Criptografia e Assinatura utilizando algoritmos criptográficos RSA 1.5 para criptografa e RSA SHA-1 para assinatura com dois níveis (SIM e NÃO), Utilização de Criptografia e Assinatura utilizando algoritmos criptográficos ECDSA (Curvas elípticas) para criptografa e assinatura com dois níveis (SIM e NÃO) e o número de clientes com três níveis (1,10 e 100). Para realizar o experimento serão realizados 100 requisições por cada cliente, sendo adotado um índice de 95% para intervalo de confiança. Já a variável de resposta adotada, será o tempo médio de resposta das requisições. Ao final do estudo será também realizado mais um teste de hipótese, com nível de significância de 5%, onde se procura verificar se há um impacto significativo na utilização da arquitetura proposta ou não. Logo, as hipóteses investigadas foram:

H_0: Não há impacto na arquitetura com utilização dos mecanismos de segurança com o Web service REST.

$H_{alternativa}$: Há um impacto significativo com utilização dos mecanismos de segurança com o Web service REST.

REFERÊNCIAS

BERTINO, E. et al. **Security for Web Services and Service-Oriented Architectures**. [S.l.]: Springer, 2010. I-XII, 1-226 p. ISBN 978-3-540-87741-7.

BHALLA, N.; KAZEROONI, S. Web services vulnerabilities. **Security Compass Inc**, 2007.

BIANCO, P.; KOTERMANSKI, R.; MERSON, P. **Evaluating a Service-Oriented Architecture**. [S.l.], 2007.

BOOTH, D. et al. **Web Services Architecture**. February 2004. World Wide Web Consortium, Note NOTE-ws-arch-20040211.

BURROWS, M.; ABADI, M.; NEEDHAM, R. A logic of authentication. **ACM Trans. Comput. Syst.**, ACM, New York, NY, USA, v. 8, n. 1, p. 18–36, fev. 1990. ISSN 0734-2071. Disponível em: <http://doi.acm.org/10.1145/77648.77649>.

CERAMI, E. **Web Services Essentials**. 1st. ed. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2002. ISBN 0596002246.

CLEMENTS, P. et al. **Documenting Software Architectures: Views and Beyond**. 2nd. ed. [S.l.]: Addison-Wesley Professional, 2010. ISBN 0321552687, 9780321552686.

COETZEE, M. Towards a holistic information security governance framework for soa. **2012 Seventh International Conference on Availability, Reliability and Security**, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 155–160, 2012.

COYLE, F. P. **XML, Web Services, and the Data Revolution**. [S.l.]: Boston: Addison-Wesley, 2002.

DAGOSTINI, S. et al. An ontology for run-time verification of security certificates for soa. **2012 Seventh International Conference on Availability, Reliability and Security**, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 525–533, 2012.

DELESSY, N. A.; FERNANDEZ, E. **A pattern-driven process for secure service-oriented applications**. Tese (Doutorado), Boca Raton, FL, USA, 2008.

ERL, T. **Soa Principios De Design De Serviços**. [S.l.]: PRENTICE HALL BRASIL, 2009. ISBN 9788576051893.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Tese (Doutorado), 2000. AAI9980887.

FURTADO, C. et al. **Arquitetura Orientada a Serviço - Conceituação**. [S.l.], 2009.

HAMMER-LAHAV, E. **The OAuth 1.0 Protocol**. Fremont, CA, USA, abr. 2010. Disponível em: <http://www.rfc-editor.org/rfc/rfc5849.txt>.

HARDT, D. **The OAuth 2.0 Authorization Framework**. Fremont, CA, USA, out. 2012. Disponível em: <http://www.rfc-editor.org/rfc/rfc6749.txt>.

JENSEN, M. et al. Soa and web services: New technologies, new standards - new attacks. In: **ECOWS**. [S.l.]: IEEE Computer Society, 2007. p. 35–44.

JOSUTTIS, N. M. **SOA in Practice: The Art of Distributed System Design**. Beijing: O'Reilly Media, Inc., 2007. ISBN 978-0-596-52955-0.

KANNEGANTI, R.; CHODAVARAPU, P. **SOA security**. Greenwich, CT, USA: Manning Publications Co., 2008. ISBN 9781932394689.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing systematic literature reviews in software engineering**. [S.l.], 2007.

LAKSHMIRAGHAVAN, B. **Pro ASP.NET Web API Security: Securing ASP.NET Web API**. 1st. ed. Berkely, CA, USA: Apress, 2013. ISBN 1430257822, 9781430257820.

LOWIS, L.; ACCORSI, R. Vulnerability analysis in soa-based business processes. **IEEE Transactions on Services Computing**, IEEE Computer Society, Los Alamitos, CA, USA, v. 4, n. 3, p. 230–242, 2011. ISSN 1939-1374.

MADSEN, P. et al. **SAML V2.0 Executive Overview**. [S.l.], abr. 2005. Disponível em: <http://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf>.

MARKS, E.; BELL, M. **Service Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology**. [S.l.]: Wiley, 2006. (Guías de España). ISBN 9780471768944.

MEIER, J. et al. **Performance testing guidance for web applications: patterns & practices**. Redmond, WA, USA: Microsoft Press, 2007. ISBN 9780735625709.

MORADIAN, E.; HåKANSSON, A. Possible attacks on xml web services. **IJCSNS International Journal of Computer Science and 154 Network Security**, VOL.6 No.1B, 2006.

NORDBOTTEN, N. A. Xml and web services security standards. **IEEE Communications Surveys and Tutorials**, v. 11, n. 3, p. 4–21, 2009.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: **Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering**. Swinton, UK, UK: British Computer Society, 2008. (EASE'08), p. 68–77. Disponível em: <http://dl.acm.org/citation.cfm?id=2227115.2227123>.

PINHO, S. C. S. de. **Arquitectura de Segurança num ambiente SOA**. Dissertação (Mestrado) — Faculdade de Engenharia da Universidade do Porto, 2008.

PULIER, E.; TAYLOR, H. **Understanding Enterprise SOA**. [S.l.]: Manning Publications, 2005. Paperback. ISBN 1932394591.

RECORDON, D.; REED, D. Openid 2.0: A platform for user-centric identity management. In: **Proceedings of the Second ACM Workshop on Digital Identity Management**. New York, NY, USA: ACM, 2006. (DIM '06), p. 11–16. ISBN 1-59593-547-9. Disponível em: <http://doi.acm.org/10.1145/1179529.1179532>.

RICHARDSON, L.; RUBY, S. **Restful Web Services**. First. [S.l.]: O'Reilly, 2007. ISBN 9780596529260.

SIDDAVATAM, I.; GADGE, J. Comprehensive test mechanism to detect attack on web services. **Networks, 2008. ICON 2008. 16 IEEE Conferência Internacional sobre**, p. 1 – 6, 2008.

TANENBAUM, M. V. S. A. S. **SISTEMAS DISTRIBUÍDOS PRINCIPIOS E PARADIGMAS 2ª EDIÇÃO**. [S.l.]: SÃO PAULO, 2007.

VERISSIMO, P.; RODRIGUES, L. **Distributed Systems for System Architects**. Norwell, MA, USA: Kluwer Academic Publishers, 2001. ISBN 0792372662.

WEBBER, J.; ROBINSON, I.; PARASTATIDIS, S. **REST in Practice: Hypermedia and Systems Architecture**. [S.l.]: O'Reilly, 2010. ISBN 978-0-596-80582-1.

WEBER, S.; AUSTEL, P.; MCINTOSH, M. A framework for multi-platform soa security analyses. In: **2007 IEEE International Conference on Web Services (ICWS 2007), July 9-13, 2007, Salt Lake City, Utah, USA**. [S.l.]: IEEE Computer Society, 2007. p. 102–109.

XU, J. et al. Dynamic authentication for cross-realm soa-based business processes. **IEEE Transactions on Services Computing**, IEEE Computer Society, Los Alamitos, CA, USA, v. 5, n. 1, p. 20–32, 2012. ISSN 1939-1374.

ZHANG, Y.; CHEN, J.-L. A delegation solution for universal identity management in soa. **IEEE Transactions on Services Computing**, IEEE Computer Society, Los Alamitos, CA, USA, v. 4, n. 1, p. 70–81, 2011. ISSN 1939-1374.