Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Proposta de Protocolo de Autenticação e Autorização seguro para aplicações SOA

Rogério Alves da Conceição

Dissertação apresentada como requisito parcial
para conclusão do Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Rodrigo Bonifácio de Almeida

Coorientadora
Prof.ª Dr.ª Edna Dias Canedo

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado Profissional em Computação Aplicada

Coordenadora: Prof. Dr. Marcelo Ladeira

Banca examinadora composta por:

Prof. Dr. Rodrigo Bonifácio de Almeida (Orientador) — CIC/UnB
Prof.ª Dr.ª Membra da Banca — MEC
Prof. Dr. Membro do Banco — CIC/UnB

CIP — Catalogação Internacional na Publicação

Conceição, Rogério Alves da.

Proposta de Protocolo de Autenticação e Autorização seguro para aplicações SOA / Rogério Alves da Conceição. Brasília : UnB, 2014.
81 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2014.

1. Interoperabilidade, 2. Segurança em arquiteturas orientadas a serviço, 3. Web Services, 4. Arquitetura de referência em SOA

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Proposta de Protocolo de Autenticação e Autorização seguro para aplicações SOA

Rogério Alves da Conceição

Dissertação apresentada como requisito parcial
para conclusão do Mestrado Profissional em Computação Aplicada

Prof. Dr. Rodrigo Bonifácio de Almeida (Orientador)
CIC/UnB

Prof.ª Dr.ª Membra da Banca
MEC

Prof. Dr. Membro do Banco
CIC/UnB

Prof. Dr. Marcelo Ladeira
Coordenadora do Mestrado Profissional em Computação Aplicada

Brasília, 24 de dezembro de 2014

Dedicatória

Citando o poeta: “Eu dedico essa música a primeira garota que tá sentada ali na fila. Brigado!”

Agradecimentos

Agradeço ao Prof. José Ralha, cujos esforços na versão anterior foram bem "adaptados" a este trabalho.

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

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 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 sistematico da literatura para compreender o estado da arte e da pratica 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

Este trabalho está organizado em seis capítulos. No capítulo 2 é apresentado um mapeamento sistemático e os resultados obtidos com a sua realização. No capítulo 3 é realizada uma revisão da literatura onde são abordados os conceitos gerais sobre da Arquitetura Orientada a Serviços (SOA), Web Services, REST, segurança e vulnerabilidades em SOA. Além disso, também são apresentados alguns protocolos de autenticação e autorização. No capítulo 4 é apresentado o protocolo de autenticação e autorização proposto e objeto deste trabalho. Nele são descritos os requisitos e a arquitetura do protocolo. É realizada

uma análise formal do protocolo utilizando-se a lógica BAN. Neste capítulo também e descrita a implementação de um protótipo do protocolo proposto bem como uma análise de segurança. No capítulo 5 é realizada uma avaliação e análise de desempenho e são apresentados os resultados dos experimentos realizados. No capítulo 6 são apresentas as conclusões do trabalho.

Capítulo 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 [33]. 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 [40].

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 chaves a serem pesquisadas e também onde as buscas serão realizadas [28]. 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 [28], 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 2.5 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 apresentada na figura 2.1.

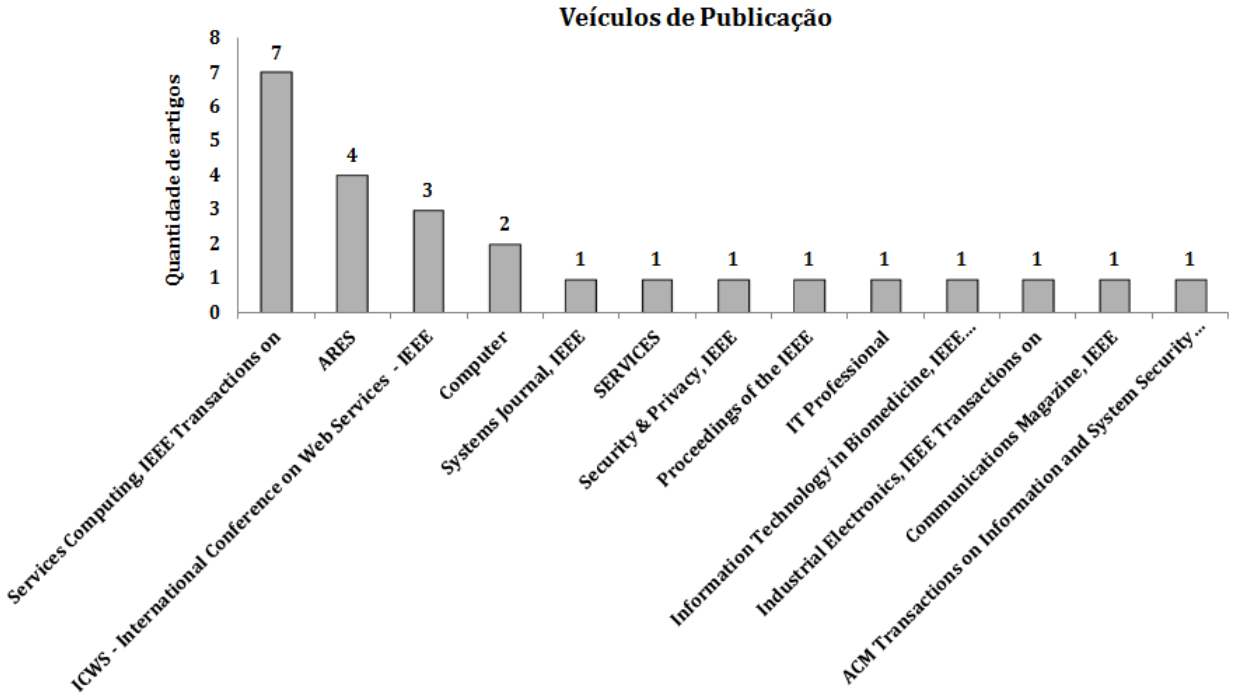

Figura 2.1: Gráfico representativo da quantidade de contribuições por veículo de publicação.

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 [51] que foi classificado como sendo uma Solução e uma Validação. Um outro exemplo pode ser observado no artigo [31] 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 [31], que propõem um novo método denominado ATLIST, que realiza análise de vulnerabilidades em processos de negócios e serviços baseados em SOA. Este 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 [11], 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 [49]. Nesta solução, é proposta uma ferramenta que tem por objetivo identificar possíveis violações de segurança em ambientes SOA. Na ferramenta 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 [9]. Este artigo analisa os desafios de segurança enfrentados em arquiteturas orientadas a serviço. Os autores propõe 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 [50]. 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 conversão de credencial nem estabelecimento de qualquer caminho de autenticação entre os serviços que participam de uma sessão de negócios. No protocolo são realizadas provas e experimentos para verificar a viabilidade da nova técnica. Este trabalho também poderá 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, foram 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 relevante. Porém, como eles se enquadraram nos critérios de busca estabelecidos, foram considerados.

A figura 2.2 apresenta os resultados categorizados pelo tipo da contribuição sendo apresentado o quantitativo de publicações.

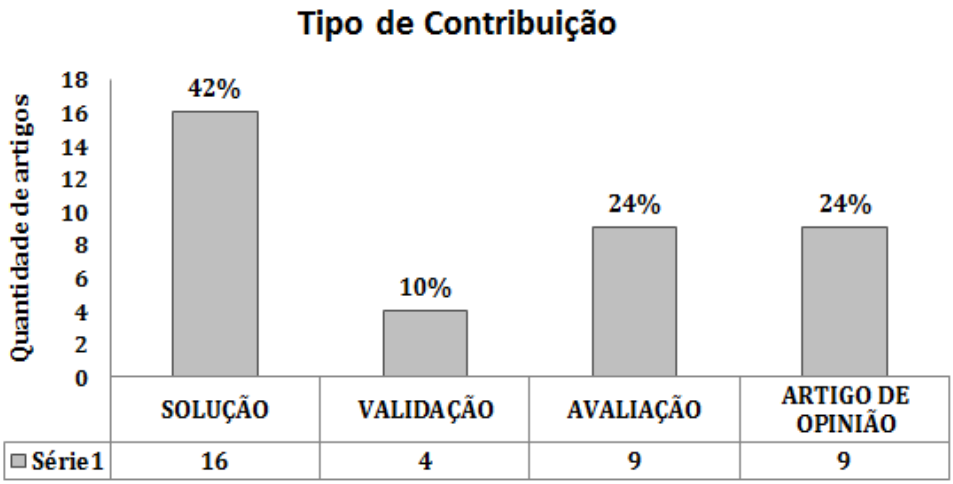

Figura 2.2: Gráfico representativo dos tipos de contribuições

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 [12] 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 2.3 apresenta graficamente essa categorização.

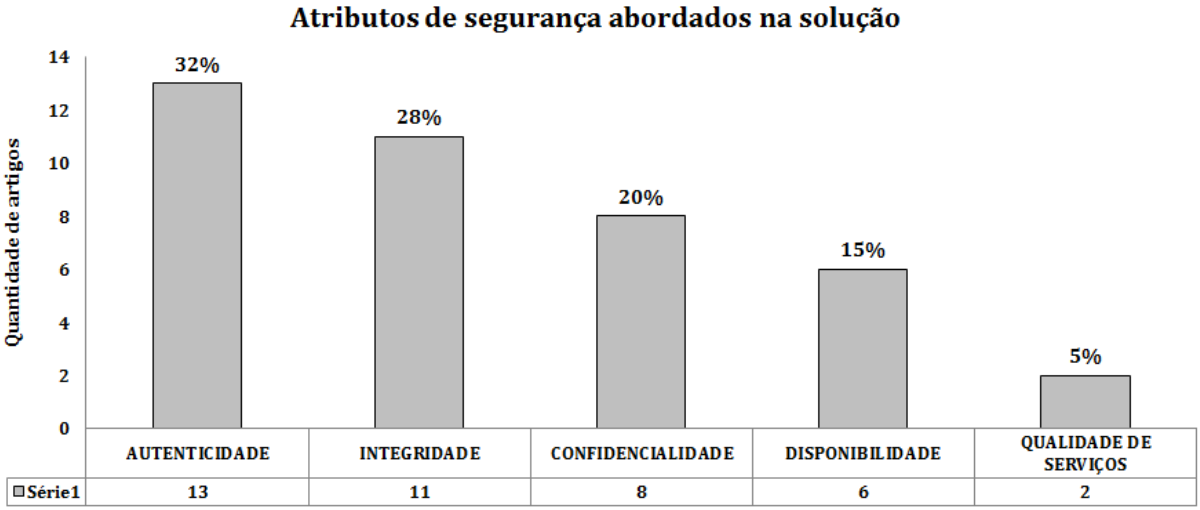

Figura 2.3: Gráfico dos atributos de segurança abordados na solução

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 dos artigos classificados como solução. Em seguida foram categorizados de acordo com os tipos: técnica, método ou ferramenta/arquitetura. Verificou-se que entre os artigos mapeados um artigo que fez referência a mais de um tipo de solução. Esse artigo é descrito na publicação de [50] 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.

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. 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 figura 2.3 apresenta graficamente essa categorização.

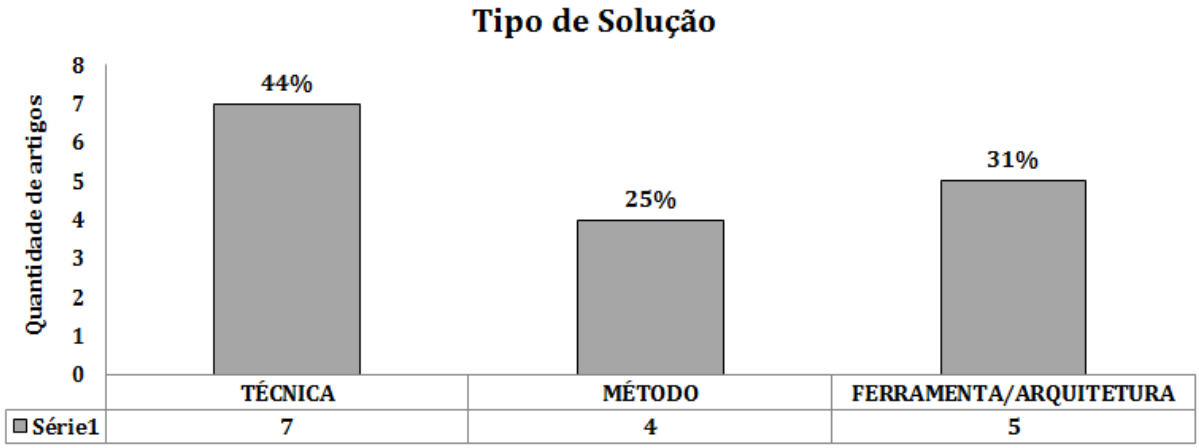

Figura 2.4: Gráfico dos tipos de Solução mais propostos

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 Síntese do capítulo

O mapeamento sistemático realizado, possibilitou aprofundar os conhecimentos referentes à segurança em SOA. Sendo 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 [49], 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 [12] onde são abordados todos os atributos de segurança: integridade, autenticidade, disponibilidade e confidencialidade. Neste trabalho é proposto 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.

O artigo descrito em [9], 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.

Capítulo 3

Revisão de Literatura

Este capítulo apresenta uma revisão dos principais conceitos relacionados ao tema deste trabalho, envolvendo Arquitetura Orientada a Serviço, web service, REST, segurança aplicada a SOA, vulnerabilidades em SOA e protocolos de autenticação e autorização.

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 [8], tem sido amplamente utilizada por um grande número de empresas.

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 [25].

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 [13]. O funcionamento de SOA baseia-se em três conceitos: serviços, interoperabilidade e baixo acoplamento [25].

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 dispo-

nibilizados 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.

SOA é um estilo arquitetural e representa propositadamente uma tecnológica neutra. Para implementá-la podem ser usadas diversas tecnologias, como por exemplo: Web services, serviços REST-*(Representational State Transfer)* e componentes distribuídos [14]. Porém, atualmente as tecnologias mais empregadas para implementar SOA são os web services e os serviços REST que são descritos nas seções 3.2 e 3.3 respectivamente.

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 [5].

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 Web service é, portanto, um sistema de software que pode agir a pedido de qualquer computador conectado à rede e que se comunica usando padrões XML [41].

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 [4]. 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 [13].
- Baixo acoplamento: Acoplamento refere-se a uma conexão do relacionamento entre dois elementos. Uma medida de acoplamento se compara a um nível de dependência [13]. 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 [13]. Logo, todo serviço possui um contrato entre o requisitante e o provedor deste serviço.

Segundo [4], 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 [10].

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 [4]. 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 3.1.

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.

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.

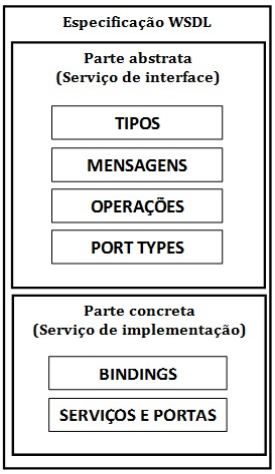

Figura 3.1: Especificação de serviços WSDL, adaptado de [4]

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 [1].

O UDDI provê um método padronizado para a publicação e descoberta de informações, permitindo que as empresas tanto publiquem como encontrem Web Services. Segundo [7], 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 [15].

Dessa forma, para que os princípios arquiteturais que permeiam *REST* sejam seguidos, um conjunto de restrições deve ser implementado [15]. Um aplicação que esteja em conformidade com essas restrições, a seguir descritas, são classificadas com RESTful [43].

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 Identifier* (URI). Que é definido como uma sequência de caracteres que identificam um recurso físico ou abstrato [3]. 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, DELETE E e adicionalmente os métodos HEADER e OPTIONS.

Neste contexto, o método GET é utilizado recuperar uma representação de um recurso. PUT para criar um novo recurso ou modificar um existente, DELETE é utilizado para remover um recurso e POST é comumente utilizado para criação de um novo recurso. HEADER é empregado para recuperar metadados de uma representação, podendo ser usado para empregar melhoria no cache. OPTIONS que retornar uma descrição de serviço ou explicação dos métodos disponíveis de uma determinada URI.

Os serviços web que são desenvolvidos baseados na arquitetura REST, ou seja, serviços Web RESTful são relativamente simples, pois a arquitetura REST segue padrões W3C/IETF3 bem conhecidos tais como (HTTP, XML, URI, MIME). Além disso, a sua adoção não é difícil de ser implantada, pois pode ser implementada em várias linguagens e em diferentes sistemas operacionais [39].

Serviços Web RESTful, são escaláveis e oferecem suporte a cache através do protocolo HTTP, *clustering* e balanceamento de carga. Outra vantagem e a possibilidade de otimização de desempenho de web services, uma vez que podem utilizar formatos de mensagem mais leves, como por exemplo, o *JavaSript Objetc Notation*(JSON) [39].

3.4 Segurança em SOA

3.4.1 Conceitos básicos

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 [4].

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 [47], 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 [4], 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 [26], 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.

3.4.2 Criptografia

A palavra criptografia origina do Grego kryptós, “escondido”, e gráphein, “escrever”, e pode ser conceituada como sendo a ciência a arte de manter mensagens seguras [44]. Ela está associada a vários aspectos da segurança da informação, tais como: privacidade ou confidencialidade, integridade do dados, autenticação e não-repúdio [36]. Em síntese, o processo de criptografia consiste em transformar uma informação que está em texto claro

(*plaintext*), em uma informação cifrada (*ciphertext*), criptografada. O processo inverso, que é o de transformar uma informação cifrada em um texto claro, não codificado é denominado decriptografia. A Figura 3.2 exemplifica esse processo.

Figura 3.2: Relação entre os termos básicos de criptografia adaptado de [44].

Para realizar o processo de criptografia é necessário a utilização de um sistema criptográfico que é formado pelo conjunto de textos em claro, textos cifrados e chaves. Atualmente existem dois tipos de sistemas criptográficos: o de chave simétrica e o de chave assimétrica.

3.4.3 Criptografia de Chave Simétrica

A criptografia de Chave Simétrica pode ser definida como sendo aquela que utiliza uma chave única tanto para cifrar quanto para decifrar um texto claro [45]. Esse sistema de criptografia possui basicamente cinco elementos: Texto claro, que é a mensagem original; Algoritmo de criptografia, que o responsável por realizar as transformações no texto claro; A chave secreta, que é a chave compartilhada entre o emissor da mensagem e o receptor, utilizada como entrada para o algoritmo de criptografia; Texto cifrado, que é a mensagem em que foi aplicada o algoritmo criptográfico, é a saída e algoritmo de criptografia: Que é basicamente o mesmo algoritmo de criptografia só que executado de forma inversa. Para isso, ele aplica ao texto cifrado a chave secreta compartilhada, utilizada no momento da cifragem, e obtém o texto claro original. O processo do Sistema Criptografia de Chave Simétrica é ilustrado na figura 3.3

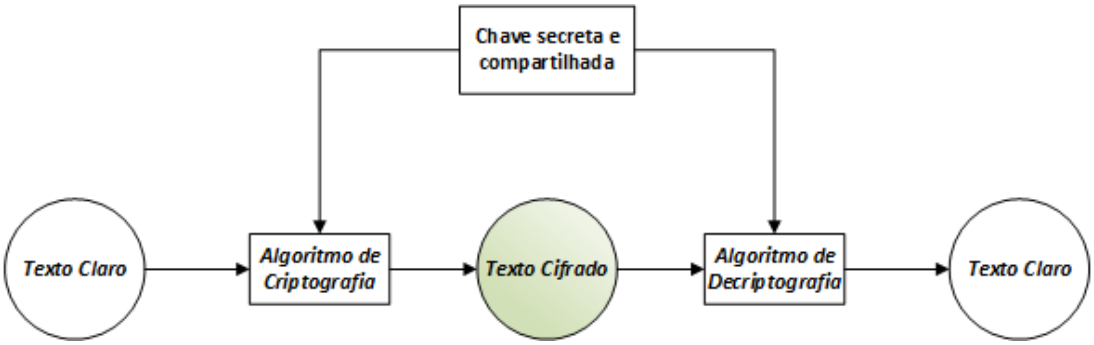

Figura 3.3: Processo de criptografia simétrica adaptado de [45].

Existem dois tipos de algoritmos de criptografia de chave simétrica: os de cifras de fluxo e cifras de blocos. Cifras de fluxo, cifram e decifram dados bit a bit, ou seja, um

bit a cada unidade de tempo, não sendo muito adequado a implementações em software. Já a cifras de blocos, realizam as mesmas operações em blocos de bits, sendo mais fáceis de implementar em softwares [44]. São exemplos de algoritmos de criptografia de chave simétrica: AES(Advanced Encryption Standard), BlowFish, Triple-DES e DES (Data Encryption Standard).

3.4.4 Criptografia de Chave Assimétrica

A criptografia de Chave Assimétrica ou de chave pública é aquela que utiliza uma chave para criptografia e outra chave diferente, porém relacionadas matematicamente, para decriptografia [45]. Dessa forma, cada usuário possui um par de chaves, uma pública e outra privada, que são utilizadas no processo de ciframento e deciframento das mensagens. Na utilização dessa técnica, todos os participantes têm acesso a chave pública porém as chaves privadas devem ser de conhecimento apenas do seu proprietário, devendo permanecer protegida e secreta [45]. A figura 3.4 exemplifica o processo de Criptografia de Chave Assimétrica.

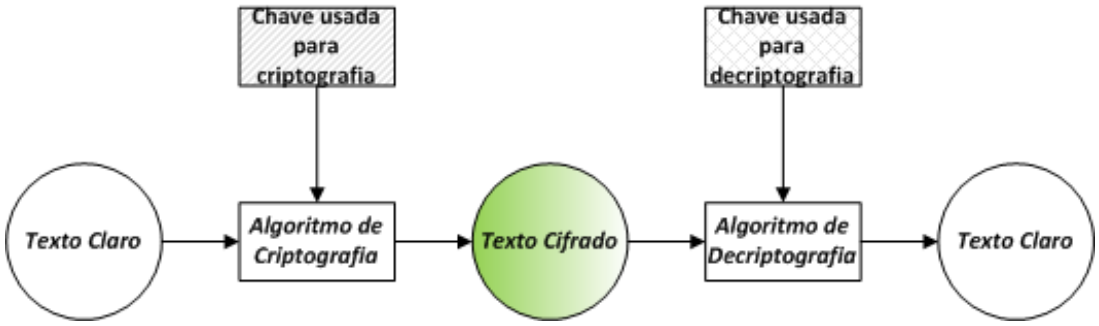

Figura 3.4: Processo de criptografia assimétrica.

A criptografia assimétrica ou de chave pública pode ser utilizada tanto para prover a confidencialidade como para possibilitar a autenticidade da mensagem. Quando usada para prover a confidencialidade dos dados, a mensagem é criptografada com a chave pública do receptor e só pode ser decifrada com a chave privada do destinatário. Vários algoritmos de criptografia de chave pública foram propostos, porém os que mais se destacaram tanto para criptografia quanto para assinaturas digitais foram o RSA, ElGamal e Rabin [44].

3.4.5 Funções de Hash

Uma função de *hash* é aquela que ao ser aplicada a uma mensagem de comprimento variável, produz como saída um valor *hash*, de tamanho fixo [44], também chamado de resumo de mensagem. Ela é amplamente utilizadas em aplicações criptográficas, tais

como: assinaturas digitais, esquemas de proteção de senha e autenticação de mensagens. A finalidade da função de *hash* é produzir um identificador único em um arquivo, mensagem ou bloco de dados. Para isso, ela deve satisfazer aos seguintes requisitos [45]:

- H pode ser aplicado a um bloco de dados de qualquer tamanho.
- H produz uma saída de comprimento fixo.
- Dado uma mensagem qualquer de entrada, deve ser fácil calcular o seu valor hash.
- Para qualquer valor *hash*(h) dado, é computacionalmente inviável encontrar x tal que H(x)= h. Ou seja, deve ser resistente a primeira inversão, de forma que seja fácil gerar um código dada uma mensagem, mas muito difícil gerar um mensagem dado um código.
- Para qualquer bloco dado x, é computacionalmente inviável encontrar y diferente x tal que H(y)= H(x). Isso é conhecido como resistência fraca a colisões.
- É computacionalmente inviável encontrar qualquer par(x,y) tal que H(x)= H(y). Ou seja, deve possuir resistência forte a colisões.

Vários algorimos de *hash* foram desenvolvidos ao logo dos anos. Como exemplo podem ser citados os algoritmos de resumo criptográfico ou MD(*Message Digest*), desenvolvidos por Ron Rivest, e cuja a última versão foi o MD5. Porém a funções de *hash* da familia MD foram considerados inseguros e devem ser evitados [16].

Outro exemplo, que merece destaque, são os algoritmos de hash seguro(SHA - *Secure Hash Algorithm*). Eles foram desenvolvidos como alternativa a insegurança dos algoritmos MD(*Message Digest*). O SHA é um padrão desenvolvido pelo Instituo Nacional de Padrões e Tecnologia (NIST-*National Institute of Standards and Technology*). Atualmente, o SHA, encontra-se na versão 3,(SHA-3), sendo considerado o padrão recomendado para aplicações atuais e futuras [16].

3.4.6 Assinatura digital

A assinatura digital pode ser definida como o mecanismo de autenticação que permite que o criador de uma mensagem possa anexar um código que atue como assinatura e garanta origem e integridade da mensagem [45]. As assinaturas digitais podem ser geradas tanto por sistemas criptográficos assimétricos, usualmente os mais empregados, quanto simétricos. No caso da geração da assinatura digital com sistemas criptográficos assimétricos, o processo ocorre da seguinte maneira: a mensagem é criptografada com a chave privada do remetente, de forma que qualquer pessoa que possua a chave pública do remetente possa verificar a autenticidade da mensagem.

Porém a utilização da criptografia assimétrica e lenta, e como alternativa utiliza-se a combinação dos sistemas criptográficos com as funções de *hash*. O que torna o processo mais rápido. Isso ocorre porque não se criptografa toda mensagem e sim apenas o resumo criptográfico, que é o resultado da aplicação da função de hash sobre a mensagem. Assinar um resumo criptográfico não é apenas mais eficiente do que assinar a própria mensagem, mas também uma defesa contra ataque do homem-do-meio, descritos na seção 3.5.4 [17].

Dessa forma, para gerar uma assinatura digital, utilizando a combinação da criptografia assimétrica com a função de *hash*, deve-se seguir os seguintes passos, conforme descrito na figura 3.5. O emissor da mensagem aplica uma função de hash na mensagem que deseja enviar a um receptor, criando dessa forma, um resumo da mensagem com tamanho fixo. Em seguida emissor criptografa esse resumo com a sua chave privada e envia ao receptor juntamente com a mensagem original. O receptor, por sua vez, recebe o documento e realiza o processo de verificação da assinatura, de forma a determinar a autenticidade e integridade da mensagem enviada. Para realizar essa verificação, ele aplica a mesma função de hash utilizada pelo emissor na mensagem original, e na sequência, o receptor utiliza a chave pública do emissor para decifrar o resumo de mensagem enviado, se o valor de *hash* gerado pelo receptor for igual ao valor de *hash* enviado então o receptor terá a certeza que o documento não foi adulterado e que ele foi enviado realmente pelo emissor, detentor da chave privada. Em outras palavras a assinatura digital é válida.

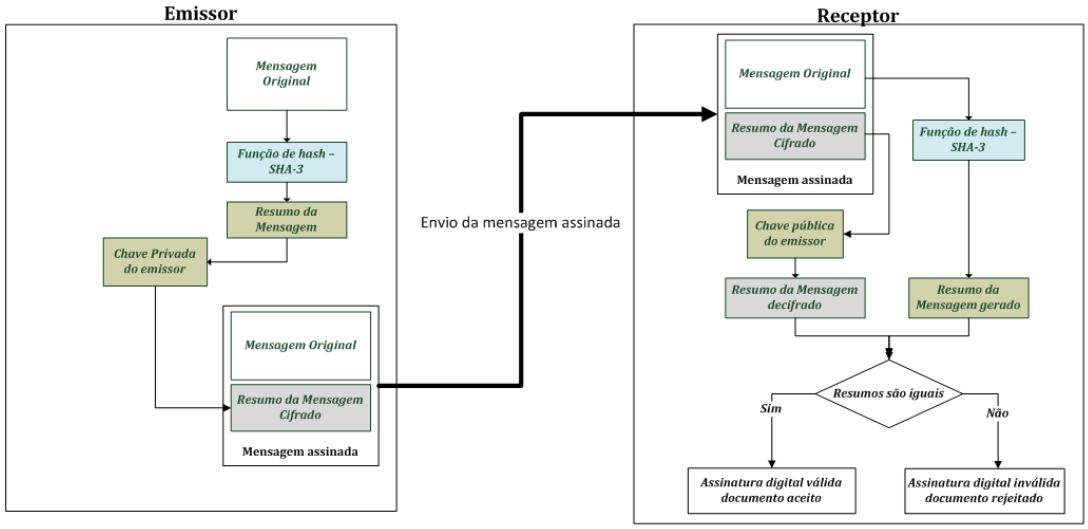

Figura 3.5: Processo de assinatura digital com função de hash.

3.4.7 Certificados Digitais

Um certificado digital é um documento eletrônico que possui informações que comprovam a identidade do seu emissor. Esse documento eletrônico é gerado e assinado por uma

terceira parte confiável, denominada Autoridade Certificadora (CA). Uma CA é responsável por associar uma chave pública a uma entidade(pessoa, processo, servidor etc) [16]. Dessa forma, uma das finalidades de um certificado digital é garantir que a chave pública contida no certificado pertence à entidade à qual ele foi emitido, possibilitando assim, a identificação inequívoca das partes envolvidas em uma comunicação.

O padrão inicialmente adotado para os certificados digitais foi esquema X.509 que é uma recomendação do ITU-T(*International Telecommunication Union*) e faz parte da série de recomendações X.500, que definem um serviço de diretório que mantém informações sobre usuários [45].

Um certificado digital X.509 normalmente contém os seguintes campos:

- Versão: Diferencia entre versões sucessivas do formato de certificado; V1, V2, V3 ou, designando a versão padrão do certificado.
- Número de série: Um número único de série emitido por uma CA que identifica o certificado.
- Identificador do algoritmo de assinatura: O algoritmo usado pela CA para assinar o certificado.
- Nome do emissor: Um identificador exclusivo para a autoridade certificadora que emitiu o certificado.
- Validade: Início e fim do período de validade do certificado.
- Nome do titular: Um nome único que identifica a entidade para a qual o certificado foi emitido.
- Informações de chave pública do titular: Contém a chave pública para a entidade identificada juntamente com um identificador para o algoritmo de criptografia usado com a chave (próximos dois itens).
- Algoritmo de assinatura: Especifica o algoritmo utilizado pela CA para assinar o certificado.
- Assinatura . Abrange toso os outros campos do certificado; ela contém o código hash dos outros campos, criptografafados com a chave privada da CA.
- Identificador exclusivo do emissor(opcional): Um identificador exclusivo usado para identificar o CA emissora caso o nome X.500 tenha sido reutilizado para entidades diferentes.

- Identificador exclusivo do titular (opcional): Um identificador exclusivo usado para identificar o titular caso o nome X.500 tenha sido reutilizado para entidades diferentes.
- Extensões (opcional): Extensões opcionais que tratam de casos e restrições especiais, tais como um certificado que identifica um CA e um certificado restrito ao uso para um propósito específico, como por exemplo: apenas para assinaturas.

3.5 Vulnerabilidades em SOA

Segundo [47], 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.

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 injeta ou manipula código SQL de forma que seja possível manipular um banco de dados de maneira inesperada[20]. Esses procedimentos podem retornar informações não previstas, alterar dados constantes nas bases de dados ou provocar indisponibilidade do próprio serviço web. Ataques de

injeção de SQL são projetados para romper a segurança do banco de dados e acessar as informações contidas nele de forma não autorizada.

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 [22]. 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, que permitem que um atacante execute consultas XML Path Language (XPath) em um banco de dados XML, controlado por um usuário. O objetivo deste ataque é realizar consultas a informações cujo acesso não foi autorizado [20].

3.5.2 Negação de Serviços e ataques de negação de serviço distribuídos

Ataques de negação de serviço(DoS) são uma tentativa coordenada para negar um serviço, fazendo com que um computador execute excessivamente uma tarefa improdutiva tornando o sistema indisponível para realizar operações legítimas [27]. 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.

O ataque de negação de serviço distribuído (DDoS) é um tipo de ataque DoS. Trata-se de inundar um ou mais computadores de destino com falsos pedidos. Isso sobrecarrega os computadores e impede que usuários legítimos tenham acesso aos serviços disponibilizados [27]. Os ataques DDoS são diferentes de ataques normais de DoS. Neste caso, o atacante sequestra centenas ou mesmo milhares de computadores na Internet, instala programas nesses computadores de forma a automatizar o ataque. O atacante então instrui os computadores sequestrados a atacar o site ou serviço alvo com mensagens forjadas. Isso causa uma sobrecarga no servidor que bloqueia tráfego legítimo. O ataque coordenado resultante dessa manobra é particularmente devastador, pois vem de muitas direções ao mesmo tempo.

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). Com esse ataque, os intrusos podem obter informações sigilosas que permitemo acesso não autorizado a um sistema [27].

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. Os invasores usam ataques man-in-the-middle para roubar informações, para executar ataques de negação de serviço, para corromper os dados transmitidos, para obter acesso a computadores e recursos de rede interna de uma organização, e introduzir novas informações em sessões de rede [27].

3.6 Protocolos de Autenticação e Autorização

A comunidade web tem desenvolvido uma série de protocolos que abordam questões como identidade e confiança [48]. 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 [29].

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 [29].E autorização que é o processo que analisa se um usuário após ser autenticado tem permissão para executar uma determinada ação.

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 [42]. 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 apresentado na Figura 3.6.

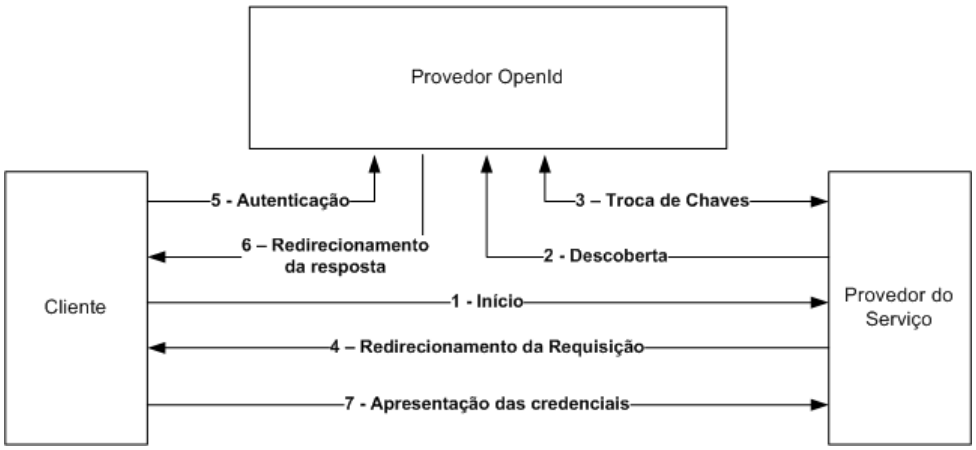

Figura 3.6: Fluxo do protocolo OpenId, adaptado de [19]

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 [32]

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 [32, 38, 4]. 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. 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.

As 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.

Os perfis, são aqueles que definem um conjunto de restrições e ou extensões para o suporte da utilização de SAML em uma 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 [18]. 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 [19].

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 [19].

Na maioria dos casos, o papel do servidor de autorização e o servidor de recursos podem ser representados por uma única entidade. A Figura 3.7 apresenta de forma abstrata o fluxo do protocolo OAuth 2.0 e descreve a interação entre os quatro papéis.

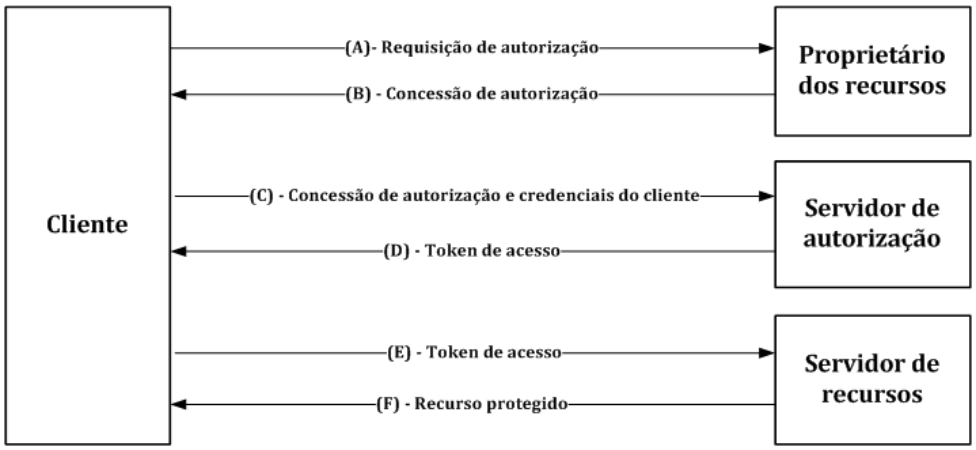

Figura 3.7: Fluxo do protocolo OAuth 2.0 adaptado de [19]

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 válidas, 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 Traust

O sistema Traust é um serviço flexível de autorização voltado a sistemas abertos de larga escala. Ele utiliza sistemas de negociação de confiança para permitir que os clientes possam estabelecer uma confiança bilateral com provedores de recursos até então desconhecidos em tempo real, além de negociar o acesso a novos recursos em tempo de execução [30].

Traust foi projetado para permitir sua integração direta com novas aplicações de reconhecimento de confiança ou indiretamente com aplicações legadas existentes. Essa flexibilidade abre o caminho para a adoção incremental de tecnologias de negociação de confiança sem a necessidade atualizações de protocolo ou de software [30].

Em síntese, o Traust atua como um intermediário que fornece tokens de segurança que permitem acesso aos recursos localizados em um determinado domínio de segurança. Os formatos dos tokens utilizados podem variar e ser de qualquer formato, incluindo usuário e senha, bilhetes Kerberos, afirmações SAML e ou certificados X.509 [30].

Dessa forma, ele pode ser integrado diretamente a aplicativos mais novos, que sejam baseados em confiança e que tenham por finalidade prover a autorização a sistemas abertos.

3.6.5 eXtensible Access Control Markup Language

O eXtensible Access Control Markup Language (XACML) foi proposto pelo grupo OASIS e atualmente está na versão 3.0 [46]. O XACML é um padrão baseado em XML que define linguagens de marcação e possibilita especificar politicas de controle de acesso e um modelo de processamento para avaliar solicitações de autorização de acordo com as politicas definidas que são utilizadas para controlar acesso a conteúdos e informações protegidas.

O XACML descreve uma linguagem para políticas de controle de acesso e têm pontos de extensão padrão para definição de novas funções, tipos de dados e combinações lógicas o que o torna flexível e extensível.

Uma dos diferenciais do XACML em relação a outros padrões proprietários é que ele é independente de uma implementação específica o que permite que ele possa ser utilizado tanto para prover controle de acesso a sistemas robustos bem como a um recurso específico. Outro ponto importante e que o XACML pode ser combinado ao SAML e assim possibilitar um a criação de um sistema de autorização completo.

3.7 Síntese do capítulo

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.

Capítulo 4

Protocolo de Autenticação e Autorização proposto

A Polícia Civil do Distrito Federal, diante da necessidade de compartilhar suas informações com órgãos parceiros, de forma eficiente e segura, objetiva estabelecer uma arquitetura de referência para a adoção de uma arquitetura orientada a serviços. Essa arquitetura deve primar pela segurança, dada a criticidade e sensibilidade das informações que são tratadas no âmbito da PCDF, conforme discutido no capítulo 1. Dessa forma, para implementar uma arquitetura orientada a serviços na instituição optou-se pela adoção de uma solução baseada no modelo REST, que é escalável e possibilita a otimização de desempenho dos serviços, uma vez que podem utilizar formatos de mensagem mais leves, como por exemplo, o *JavaSript Objetc Notation*(JSON), conforme apresentado no capitulo e seção. Neste cenário, essa dissertação contribui com um protocolo de autenticação e autorização que atende às necessidades particulares da PCDF, mas que também pode ser adotado em outras instituições. Este capítulo inicia a apresentação dos requisitos do protocolo e de sua arquitetura, onde é apresentada uma visão geral do protocolo de autenticação e autorização proposto. Em seguida é realizada a análise formal do protocolo utilizando-se a lógica BAN. Por fim, este capítulo termina com a descrição da implementação de um protótipo do protocolo proposto.

4.1 Requisitos do Protocolo

Conforme discutido, o protocolo de autenticação e autorização deve ser aderente à arquitetura REST, haja vista que sua adoção é fácil, pois pode ser implementada em várias linguagens e em diferentes sistemas operacionais. Além disso, permite utilizar o formato de mensagem JSON, que é o formato de mensagem padrão utilizado pelo protocolo

de autenticação e autorização proposto. Os requisitos inerentes ao protocolo são descritos nesta seção.

RQ1 Segurança de sessão. Toda comunicação entre o cliente e o servidor deve ser 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 Certificadora(AC), que esteja subordinada à hierarquia da ICP-Brasil, para encriptar as comunicações e garantir a autenticidade do servidor e do cliente. Os clientes devem realizar a validação do certificado antes de interagir com o servidor.

RQ2 Segurança na troca de mensagens. Deve ser utilizada criptografia assimétrica, para promover a segurança na troca de mensagens realizada entre o Cliente e a PCDF. Todas as mensagens deverão ser assinadas digitalmente. Para isso, será utilizado uma função *Hash*, com o algoritmo SHA (Secure Hash Algorithm).

RQ3 Autorização e escalabilidade. O protocolo deve permitir acesso aos serviços ofertados apenas às instituições autorizadas. Desta forma, a autenticação e autorização devem seguir os padrões definidos na política de segurança estabelecida pela PCDF. Logo, para ser autenticado, o usuário deve apresentar credenciais válidas, que são declarações de identidade (*Claims*), que serão utilizadas no processo de autenticação e autorização. Essas credenciais devem ser criptografadas, assinadas e enviadas no cabeçalho do protocolo HTTPS. O protocolo de autenticação e autorização proposto deve ser escalável em termos de sobrecarga, tamanho do domínio de proteção e de manutenção. Além disso, o protocolo deve permitir a preservação de privacidade, uma vez que para proteger a instituições e a PCDF de entidades maliciosas, suas interações deverão revelar o mínimo de informações possíveis.

RQ4 Flexibilidade. A autenticação e autorização deve ser 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.

RQ5 Contratos previamente definidos. A política de autenticação e autorização proposta no protocolo será estabelecida por meio de contratos, onde serão definidos todas as regras que deverão ser atendidas pelas instituições conveniadas e pela PCDF.

Dessa forma, para que qualquer instituição possa ter acesso aos serviços ofertados pela Divisão de Tecnologia da PCDF, faz-se necessário o estabelecimento de um contrato

prévio de acesso. Ou seja, primeiramente devem ser estabelecidas as restrições de acesso e autorização. Uma vez cadastrada, a instituição conveniada deve informar as credenciais para comprovar a sua identidade no momento da autenticação, de forma que ela possa ser autorizada de acordo com o seus privilégios ou permissões.

No momento do estabelecimento do contrato, serão geradas para o cliente múltiplas credenciais, que são declarações de identidade (*Claims*), que devem ser utilizadas no processo de autenticação e autorização. Essas informações devem ser compartilhadas entre a instituição conveniada, o servidor de Autenticação e Autorizaçãoe o servidor de fachada REST, que são detalhados na seção 4.2. Além disso, o Contrato poderá ter acesso a múltiplos serviços. A Figura 4.1 apresenta o relacionamento entre o contrato, as credenciais e os serviços.

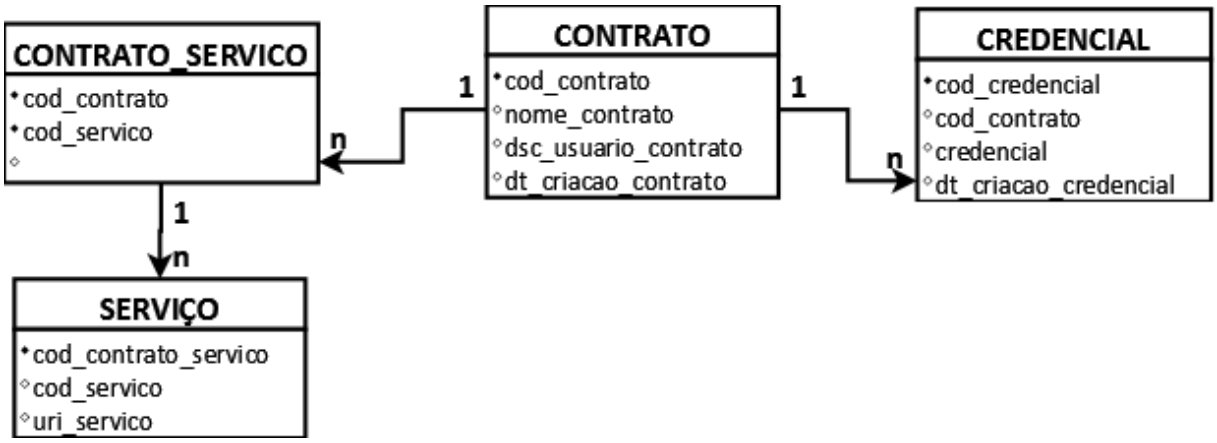

Figura 4.1: Diagrama de relacionamento entre contrato, credenciais e usuários

4.2 Arquitetura do Protoloco

A arquitetura do protocolo proposto é apresentada na Figura 4.2. O protocolo é composto por quatro componentes: Cliente, servidor de Autenticação e Autorização, servidor de fachada REST, e servidor de Banco de Dados- que gerencia contratos, credenciais e *tokens* de acesso.

O componente Cliente na arquitetura do protocolo representa as Instituições ou Órgãos conveniados, que após firmar um contrato, podem consumir os serviços ofertados pela PCDF. O Servidor de Autenticação e Autorização tem um papel fundamental na arquitetura do protocolo, pois é nele que o gerenciamento de autenticação e autorização é realizado. Desta forma, o servidor de Autenticação e Autorização é responsável por realizar os processos de verificação e validação de credenciais, criação dos desafios de autenticação, criação de *tokens* JSON e a criação e o gerenciamento das credenciais de

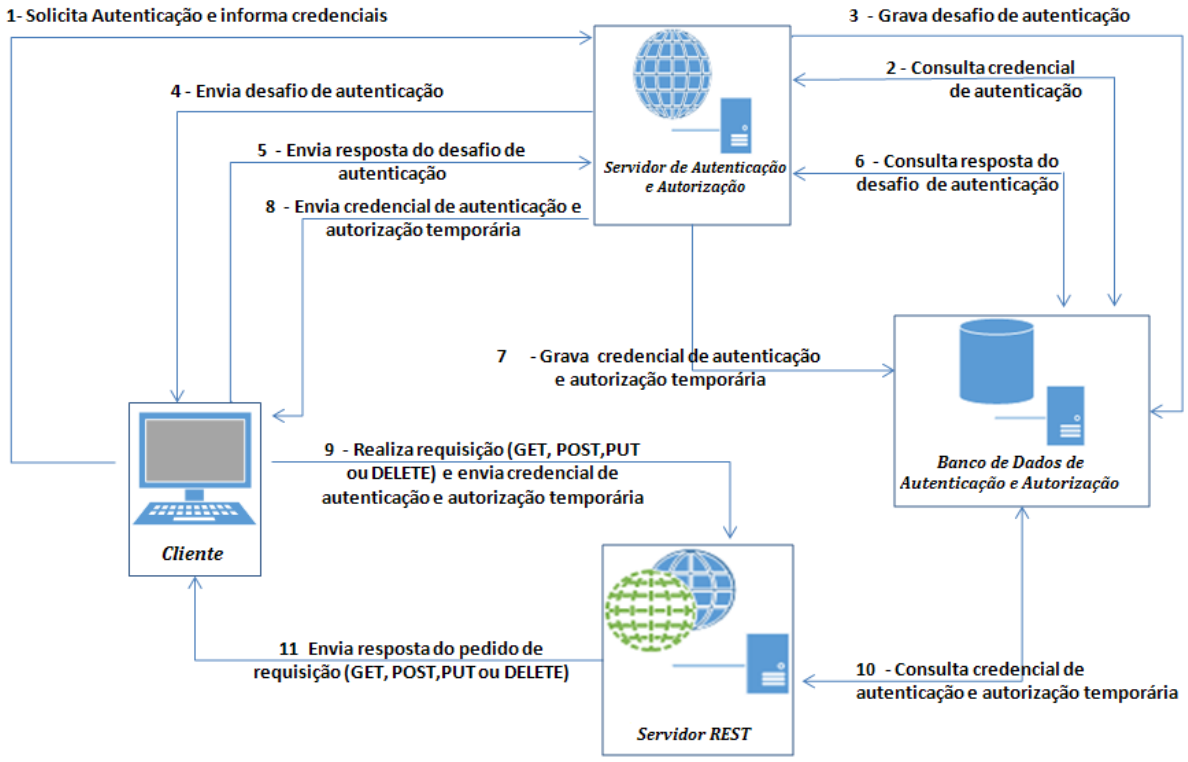

Figura 4.2: Fluxo do protocolo de autenticação/autorização proposto, 1º cenário.

autenticação e autorização temporárias, que são utilizados pelos Clientes para consumir os serviços requisitados. O servidor de fachada REST atua como uma fachada, abstraindo toda lógica necessária para o consumo dos serviços. Neste servidor estão concentrados os serviços REST disponibilizados pela PCDF. Desta forma, quando um Cliente necessita acessar um serviço, primeiramente ele deve ser autenticado e autorizado no servidor de Autenticação e Autorização. Após esse processo, o Cliente faz a requisição ao servidor de fachada REST, que realiza as verificações necessárias para saber se o Cliente tem privilégios ou não para acessar o serviço. O servidor de fachada REST acessa a base de dados de Autenticação e Autorização para confirmar as credenciais de autenticação e autorização temporárias informadas e, caso elas sejam válidas, permite que o Cliente acesse o serviço requerido. Um ponto importante a ser destacado é que os desenvolvedores, ao desenvolver um serviço, não necessitam ter preocupações referentes à autenticação e autorização, pois essas preocupações são de responsabilidade do servidor de fachada REST. Finalmente, o servidor de Banco de Dados mantém as informações necessárias para o funcionamento dos serviços de autenticação e autorização. É neste servidor que são salvas os usuários, os desafios e as credenciais de autorização e autenticação temporária.

4.2.1 Visão geral do protocolo de Autenticação e Autorização proposto

Para ter acesso a API REST, referente aos serviços ofertados, o Cliente deve ser autenticado e autorizado a acessar o serviço. Para isso, o protocolo usa a autenticação baseada em *tokens* de segurança, que são recipientes de reivindicações da autoridade emissora. Os *tokens* de segurança utilizados (*Web Tokens*) seguem o formato *JSON*. Esse formato, 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 do protocolo são assinadas e criptografadas de forma assimétrica. O processo de autenticação e autorização é descrito em dois cenários distintos. No primeiro cenário, representado na Figura 4.3, o Cliente não está autenticado. No segundo cenário, o Cliente está autenticado e possui uma credencial de autorização.

4.2.1.1 Primeiro cenário

No primeiro cenário, o Cliente não está autenticado e deve solicitar a autenticação pela primeira vez, conforme descrito a seguir:

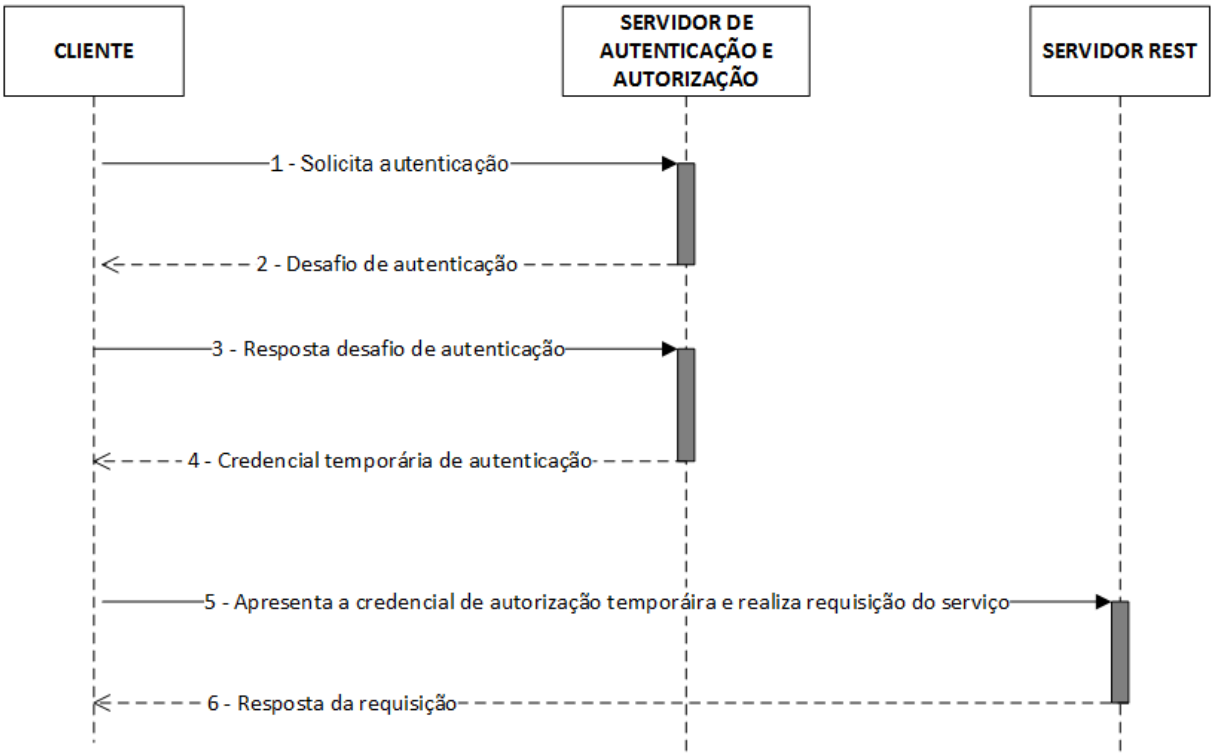

Figura 4.3: Fluxo do protocolo de autenticação/autorização proposto, 1º cenário.

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 uma mensagem

(mensagem 1 da Figura 4.3)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. O *token* é assinado digitalmente pelo Cliente e cifrado com a chave pública do servidor de Autenticação e Autorização. 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.

Na segunda mensagem, 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, o código HTTP 401 (usuário não autorizado) é retornada ao Cliente. Também é feita a verificação de *timestamp*, que se refere ao tempo de envio da mensagem. Caso a mensagem tenha sido enviada em um período de tempo superior ao pré-estabelecido no contrato, o Cliente recebe como resposta o código HTTP 401 (usuário não autorizado).

Caso não ocorram problemas, procede-se com o processo de validação da credencial informada. Tal processo consiste em consultar a credencial em uma base de dados e verificar se a credencial é válida e associada ao Cliente. Em seguida, 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, os dados relacionados ao desafio, a data e hora de geração do desafio e a resposta que o Cliente deverá fornecer são persistidos no servidor de Banco de Dados. Finalmente, o *token* JSON, contendo o código do desafio, o código da credencial e um *timestamp* representando a data e hora de criação do desafio é enviado ao Cliente. Tal *token* é assinado digitalmente pelo servidor de Autenticação e Autorização e cifrado com a chave pública do Cliente que está solicitando a autenticação.

Na terceira mensagem, após receber o desafio do servidor de Autenticação e Autorização, o Cliente extrai o *token* cifrado com sua chave privada e verifica a autenticidade e integridade da requisição por meio da verificação da assinatura digital do servidor de Autenticação e Autorização. Em seguida, o Cliente analisa o *timestamp* com o objetivo de verificar se a mensagem foi enviada em um período de tempo superior ao pré-estabelecido no contrato. Se for detectada alguma inconsistência, o processo de autenticação atual é descartado e inicia-se um novo processo de autenticação.

Caso nenhuma inconsistência seja identificada, o Cliente verifica e responde o desafio solicitado, enviando-o juntamente com um *timestamp* e o código do serviço que deseja consumir, para o servidor de Autenticação e Autorização por meio de um *token* JSON, que é assinado digitalmente pelo Cliente e cifrado com a chave pública do servidor de

Autenticação e Autorização.

Na quarta mensagem, o servidor de Autenticação e Autorização recebe a resposta do desafio de autenticação, decifra o *token* e verifica a autenticidade e integridade da requisição por meio da verificação da assinatura digital do Cliente. Não ocorrendo nenhuma violação, inicia-se o processo de verificação da resposta. A primeira verificação realizada refere-se ao tempo de geração do desafio, por meio do *timestamp*. Se a resposta tiver sido enviada em um período de tempo superior ao pré-estabelecido em contrato, o servidor de Autenticação e Autorização responde com o código HTTP 401 (usuário não autorizado). Caso contrário, procede com a verificação do desafio que consiste em realizar uma consulta na tabela de desafios verificando se a resposta ao desafio corresponde a esperada. Caso a resposta esteja correta o servidor de Autenticação e Autorização autentica o Cliente. Em seguida o servidor de Autenticação e Autorização verifica, considerando o código do serviço requisitado, se o Cliente tem privilégios necessários para consumir o serviço requisitado.

Caso o Cliente possua os privilégios necessários, o servidor de Autenticação e Autorização gera uma credencial de autenticação e autorização temporária para o serviço solicitado. Tal credencial é gravada em uma tabela de credencias de autorização temporária, juntamente com a data e hora de criação, a data de expiração e o código do Cliente. A tabela de credencias de autorização temporária é posteriormente acessada pelo servidor de fachada REST para verificar quais privilégios o Cliente tem acesso e se o mesmo está autenticada. O *token*, contendo a credencial de autenticação e autorização temporária, é assinado digitalmente pelo servidor de Autenticação e Autorização e cifrado com a chave pública do Cliente. Após esse processo, o *token* é enviado ao Cliente.

Caso a resposta do desafio esteja em desacordo com a esperada ou se o Cliente não tiver privilégios suficientes para acessar o serviço requisitado, a resposta do servidor de Autenticação e Autorização contém o código HTTP 401 (usuário não autorizado). É importante destacar que a credencial de autenticação e autorização temporária será gerada apenas para o serviço que o Cliente tenha solicitado e possua o privilégio de acesso para utilizá-la. Nesse caso, a credencial será válida por um período de tempo que será definido no momento da assinatura do contrato de prestação de serviço, entre o órgão conveniado e a PCDF.

Na quinta mensagem, o Cliente, extrai o *token* cifrado com sua chave privada e verifica a autenticidade e integridade da requisição por meio da verificação da assinatura digital do servidor de Autenticação e Autorização. Em seguida consulta o *timestamp* para verificar se a mensagem foi enviada em um período de tempo superior ao pré-estabelecido no contrato. Caso ocorra alguma inconsistência, o processo de autenticação atual é descartado e inicia-se um novo processo de autenticação.

Caso não ocorram inconsistências, o Cliente verifica a data e hora de validade da credencial de autorização temporária para saber se a mesma é válida. Confirmada sua validade, o Cliente envia ao servidor de fachada REST, a requisição do serviço que deseja consumir, juntamente com a credencial de autenticação e autorização temporária. O *token* de autenticação e autorização temporária é assinado com a chave privada do Cliente e cifrado com chave pública do servidor de fachada REST, sendo enviado no cabeçalho da requisição.

Finalmente, após receber a requisição, o servidor de fachada REST extrai o token cifrado com sua chave privada e verifica a autenticidade e integridade da requisição por meio da verificação da assinatura digital do servidor de Autenticação e Autorização. Em seguida consulta o *timestamp* para checar se a mensagem foi enviada em um período de tempo superior ao pré-estabelecido no contrato. Não havendo inconsistências, o servidor de fachada RESTverifica se a credencial de autenticação e autorização temporária é valida. Para isso, ele realiza uma consulta na tabela de credenciais temporárias, com a finalidade de confirmar se a credencial informada não expirou, se foi realmente gerada para o Cliente e se está associada ao serviço solicitado.

Nas situações em que a verificação é consistente, o Cliente recebe os dados referentes à sua requisição. Havendo qualquer problema ele recebe uma resposta contendo o código HTTP 401 (usuário não autorizado).

4.2.1.2 Segundo cenário

No segundo cenário, o Cliente possui uma credencial de autorização temporária,que deve ser apresentada quando requisitar algum serviço que possui acesso. Neste caso, o Cliente envia um *token* ao servidor de fachada REST contendo uma credencial temporária no cabeçalho da requisição do serviço que deseja consumir. O servidor de fachada RESTrecebe o *token* de autenticação e autorização, faz a verificação na tabela de credenciais temporárias para confirmar que o Cliente possui uma credencial válida e, caso afirmativo, verifica quais são os privilégios de autorização da credencial e verifica se o Cliente possui a permissão necessária para acessar o serviço. Neste cenário, a requisição do Cliente é atendida. Por outro lado, 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. Esse processo é descrito no fluxo alternativo da figura 4.4.

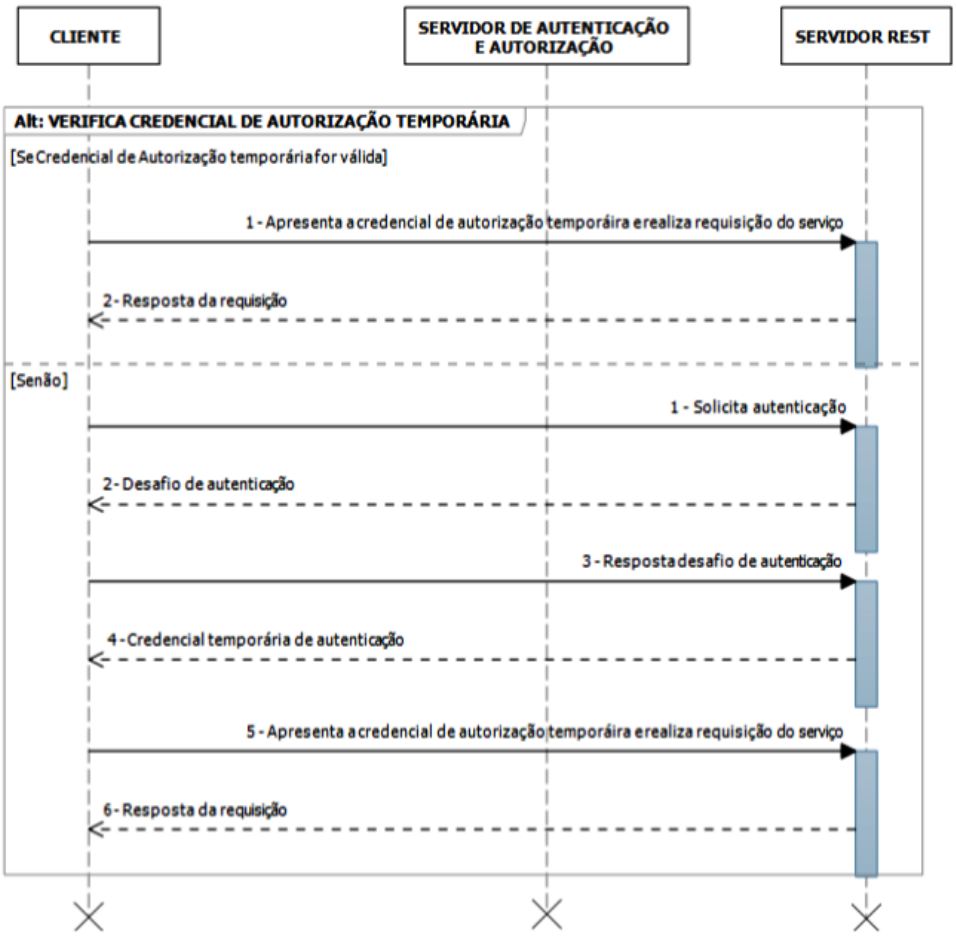

Figura 4.4: Fluxo do protocolo de autenticação/autorização proposto, 2º cenário.

4.3 Formalização do protocolo

O uso de especificações formais na área de criptografia (em particular para especificar protocolos criptográficos) não é recente. Parte dos trabalhos nesta área foram desenvolvidos ainda na década de 90 [34], possibilitando uma análise mais detalhada dos protocolos. Com isso, o principal objetivo tem sido o de verificar se os objetivos de segurança propostos pelos autores dos protocolos criptográficos são alcançados.

Neste trabalho, o protocolo proposto foi descrito formalmente utilizando a lógica BAN, com o intuito de favorecer uma melhor comunicação e entendimento utilizando uma linguagem mais precisa. Além disso, a propriedade de terminação com a geração da credencial temporária de autenticação e autorização foi verificada com um programa escrito em Prolog.

4.3.1 Lógica BAN

A lógica BAN foi desenvolvida por Burrows, Abadi e Needham em 1989, tendo alcançado certa popularidade para a análise de confiança e de conhecimento entre os participantes dos protocolos criptográficos [6]. BAN é uma lógica pioneira na especificação de protocolos criptográficos, em particular protocolos usados na autenticação e distribuição de chaves [6].

4.3.1.1 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, sendo usualmente referenciada como uma declaração lógica. Em geral, os símbolos A, B e S denotam entidades ou participantes; 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. Finalmente, Na, Nb e Ns são os identificadores gerados pelos participantes (referenciados como *nonces* na literatura). As construções usadas com maior frequência são apresentadas na Tabela 4.1:

Expressão	***Leitura/Significado***
$P \mid\equiv X$	P **acredita** X: P confia em X, ou P acredita que X é verdadeiro.
$P \triangleleft X$	P **recebeu** X: Alguém enviou uma mensagem para P contendo X; ou seja, P recebeu X.
$P \mid\sim X$	P **disse** X: P alguma vez compartilhou X. Pode-se assumir que 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 e deve ser confiável.
#(X)	novo(X): a fórmula X não foi usada em mensagens anteriores à execução atual do protocolo.
$P \overset{k}{\longleftrightarrow} Q$	(lê-se "k é uma chave satisfatória para P e Q"). A chave k não pode ser descoberta por qualquer outro participante, exceto P, Q ou por alguém que ambos confiam.
$\{X\}_K$	fórmula X foi cifrada com a chave K. As mensagens cifradas são legíveis e verificáveis apenas pelos participantes que conhecem a chave K.
$P \overset{k}{\rightleftharpoons} Q$	k é um segredo compartilhado entre P e Q e possivelmente as entidades de confiança de P e Q. Somente P e Q podem usar k para provar suas identidades.
$\overset{K}{\mapsto} P$:	k é a chave pública de P.

Tabela 4.1: Notação básica da Lógica BAN, adaptação de [6].

4.3.1.2 Postulados lógicos

No estudo de protocolos de segurança é importante diferenciar o tempo das demonstrações ou eventos. Caso isso não seja observado, problemas como a não detecção do reenvio de mensagens podem ocorrer. Dessa forma, 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 [6]. Essa divisão de tempo é suficiente para facilitar o entendimento sobre como a lógica pode ser manipulada. Para realizar a análise dos protocolos de segurança, a lógica BAN possui uma série de postulados lógicos. No restante dessa seção apresentamos alguns desses postulados, reforçando que maiores detalhes podem ser encontrados no artigo que introduziu a notação BAN [6]

(P1) Regra de significado da mensagem. Esta regra faz parte da interpretação das mensagens. Para as chaves secretas compartilhadas, temos que:

$$\frac{P \mid\equiv P \overset{k}{\longleftrightarrow} Q \quad P \triangleleft \{ X \}_k}{P \mid\equiv Q \mid\sim X}$$

Ou seja, assumindo que P acredita que k é uma chave satisfatória para se comunicar com Q, e que P recebeu a mensagem X cifrada com a chave k, então P acredita que Q uma vez disse X. De forma similar, esse postulado quando consideramos chaves públicas estabelece que:

$$\frac{P \mid\equiv \overset{K}{\mapsto} Q \quad P \triangleleft \{ X \}_{k^{-1}}}{P \mid\equiv Q \mid\sim X}$$

(P2) Regra de verificação do identificador. Essa regra estabelece que, dada uma mensagem recente, enviada durante a execução atual do protocolo é possível assumir que o emissor confia na mensagem.

$$\frac{P \mid\equiv \#(X), \quad P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$$

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

(P3) Regra da jurisdição. Esta regra representa a confiança e a autoridade de uma entidade sobre as declarações.

$$\frac{P \mid\equiv Q \Rightarrow X, \quad P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$$

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 os postulados fundamentais na análise formal do protocolo criptográfico proposto nessa dissertação. A utilização destas regras, juntamente com as notações descritas na sessão anterior, possibilita o estabelecimento da confiança entre os participantes de um protocolo.

4.3.2 Análise formal do protocolo proposto

Nesta sessão apresentamos a análise formal do protocolo de autenticação e autorização proposto, seguindo o processo descrito em [6]. De acordo com o referido processo, a análise de um protocolo é realizada em quatro etapas, que serviram para organizar o restante dessa seção.

4.3.2.1 Idealização do protocolo

Para especificar o protocolo formalmente, foram utilizadas algumas notações para representar os elementos participantes. Logo, os símbolos A, B e C são utilizadas para representar respectivamente as entidades que trocam mensagens entre si. No protocolo proposto, temos como participantes os nós Cliente, servidor de fachada REST e servidor de Autenticação e Autorização. As chaves públicas das entidades A, B e C são representadas, respectivamente, por Ka, Kb e Kc. As chaves privadas, seguindo o mesmo pressuposto, são representadas pelos símbolos Ka^{-1}, Kb^{-1} e Kc^{-1}. Os elementos $Cred_A$ e Cod_{Srv_A} representam, respectivamente, a credencial utilizada pela entidade A e o código que identifica o serviço que a entidade A deseja consumir.

O desafio, gerado pela entidade C e enviado a entidade A, é representado pela fórmula N_{CA}. Essa fórmula contempla o código do desafio gerado pela entidade C e o código da credencial aleatória da entidade A. A resposta do desafio gerada pela entidade C e enviada à entidade A é representada pela fórmula $Resp_{AC}$. Essa fórmula corresponde ao código do desafio gerado pela entidade C e a credencial solicitada pela entidade A. Ts_A e Ts_C são, respectivamente, os *timestamps* emitidos pelas entidades A e C.

O símbolo Msg_{AC} representa o resumo da mensagem enviada pela entidade A à entidade C, Msg_{CA} representa o resumo da mensagem enviada pela entidade C à entidade

A e Msg_{AB} representa o resumo da mensagem enviada pela entidade A à entidade B. O elemento H representa a aplicação de uma função *Hash* a uma mensagem. Finalmente, o símbolo Exp_A representa a data/hora de expiração da credencial temporária de autorização e autenticação, C_{Aut} corresponde à credencial temporária de autorização e autenticação gerada para a entidade A e Req_A refere-se à requisição de serviços realizadas a partir da entidade A, utilizando algum dos métodos do protocolo HTTP (GET, PUT, POST ou DELETE).

Para especificar formalmente um protocolo de segurança utilizando a lógica BAN, é necessário idealizar o protocolo e, a partir da aplicação dos postulados e das suposições iniciais, verificar se ele atinge ou não o seu objetivo. A idealização do protocolo proposto é descrito na figura 4.5, representando o fluxo de troca de mensagens executado pelo protocolo.

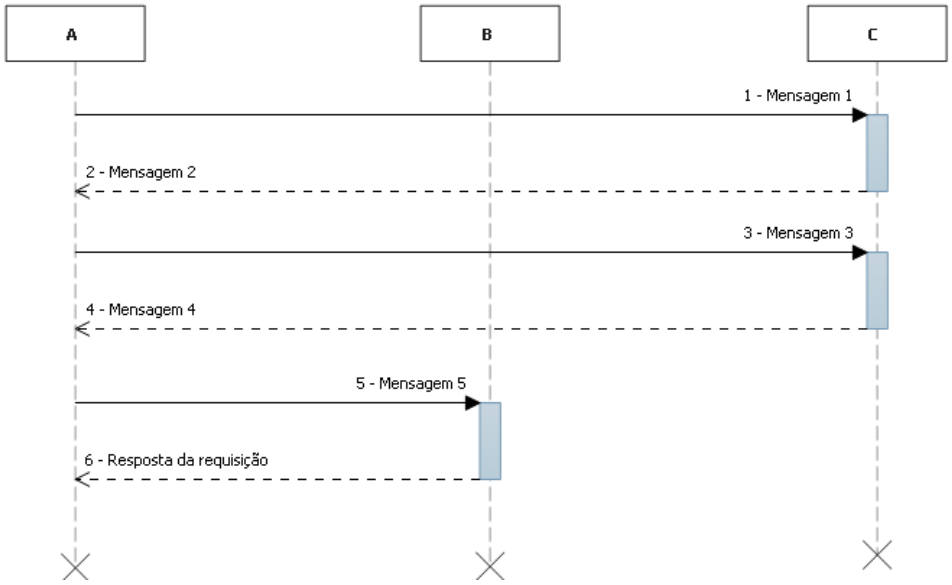

Figura 4.5: Diagrama com a idealização do protocolo de autenticação/autorização proposto

1. Mensagem 1: $A \longrightarrow C : \{ Ts_A, Cred_A, H\{ Msg_{AC} \}_{Ka^{-1}} \}_{Kc}$.
2. Mensagem 2: $C \longrightarrow A : \{ Ts_C, N_{CA}, H\{ Msg_{CA} \}_{Kc^{-1}} \}_{Ka}$.
3. Mensagem 3: $A \longrightarrow C : \{ Ts_A, Resp_{AC}, Cod_{Srv_A}, H\{ Msg_{AC} \}_{Ka^{-1}} \}_{Kc}$.
4. Mensagem 4: $C \longrightarrow A : \{ Ts_C, Exp_A, \#(A \overset{C_{Aut}}{\rightleftharpoons} C), H\{ Msg_{CA} \}_{Kc^{-1}} \}_{Ka}$.
5. Mensagem 5: $A \longrightarrow B : \{ Ts_A, (A \overset{C_{Aut}}{\rightleftharpoons} C), H\{ Msg_{AB} \}_{Ka^{-1}} \}_{Ka}, Req_A$.

4.3.2.2 Suposições

O objetivo do protocolo é fazer com que a entidade A seja autenticada pela entidade C e obtenha uma credencial de autenticação e autorização temporária, referente a uma requisição de um serviço que a entidade A deseja consumir. Com isso, a credencial de autenticação e autorização temporária pode ser utilizada pela entidade A no momento da requisição do serviço à entidade B e, assim, realizar a computação que deseja via o consumo a um serviço forma segura. Para isso, algumas suposições iniciais são estabelecidas e, juntamente com a aplicação dos postulados da lógica BAN, busca-se verificar se o protocolo alcança o objetivo proposto. Todas as suposições, apresentadas na Tabela 4.2, são baseadas no uso de um canal seguro de comunicação SSL/TSL.

Suposição	**Descrição**	**Suposição**	**Descrição**
1 -	$A \mid\equiv \overset{Kc}{\mapsto} C$	**9 -**	$B \mid\equiv C \Rightarrow \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$
2 -	$B \mid\equiv \overset{Ka}{\mapsto} A$	**10 -**	$A \mid\equiv C \mid\equiv \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$
3 -	$C \mid\equiv \overset{Ka}{\mapsto} A$	**11 -**	$B \mid\equiv C \mid\equiv \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$
4 -	$A \mid\equiv \#Ts_C$	**12 -**	$A \mid\equiv \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$
5 -	$B \mid\equiv \#Ts_A$	**13 -**	$B \mid\equiv \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$
6 -	$C \mid\equiv \#Ts_A$		
7 -	$A \mid\equiv Exp_A$		
8 -	$A \mid\equiv C \Rightarrow \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$		

Tabela 4.2: Suposições aplicadas ao protocolo proposto.

Dessa forma, temos que as suposições 1, 2 e 3 garantem que as entidades participantes A, B e C confiam nas chaves públicas das entidades que farão as trocas de mensagem. As suposições 4, 5 e 6 são *timestamps*, o que denota que as entidades A, B e C devem estar sincronizadas. Sendo assim, a entidade A acredita que o *timestamp* Ts_C é novo e foi gerado recentemente. Da mesma forma que as entidades B e C acreditam que o *timestamp* Ts_A também é novo e foi gerado recentemente. A suposição 7 é utilizada pela entidade A para garantir que a credencial de autenticação e autorização gerado pela entidade C não expirou e que pode ser utilizada. As suposições 8 e 9 denotam que as entidades A e B acreditam que entidade C tem jurisdição sobre a credencial de autenticação e autorização gerada. Portanto, as suposições 10 e 11 garantem que as entidades A, B acreditam que a credencial de autenticação e autorização gerada é nova é foi realmente gerada pela entidade C. Finalmente, as suposições 12 e 13 garantem que as entidades A e B acreditam na nova credencial de autenticação e autorização temporária representada por C_{Aut}.

4.3.2.3 Provas

Como o objetivo final do protocolo é autenticar a entidade A, de forma que ela obtenha uma credencial de autenticação e autorização temporária e referente a uma requisição de serviço desejado. Nessa seção, será realizada uma análise de cada mensagem do protocolo idealizado, aplicando os postulados lógicos e suposições com o intuito de provar que o protocolo consegue atingir o objetivo proposto.

Na primeira mensagem, a entidade A envia sua credencial, um *timestamp* e o código do serviço que ele está querendo consumir ao servidor de Autenticação e Autorização, representado pela entidade C. A mensagem enviada é assinada com a chave privada do participante A e cifrada com a chave pública do participante C. Após o recebimento dessa mensagem, o estado da execução do protocolo evolui conforme a descrição

Mensagem 1: $A \longrightarrow C : \{ Ts_A, Cred_A, H\{ Msg_{AC} \}_{Ka^{-1}} \}_{Kc}$.

$C \triangleleft \{ Ts_A, Cred_A, Cod_{Srv_A}, H\{ Msg_{AC} \}_{Ka^{-1}} \}_{Kc}$

$C \mid\equiv A \mid\sim H\{Msg_{AC}\}$

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

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

Dessa forma, C recebe a fórmula $\{ Ts_A, Cred_A, Cod_{Srv_A}, H\{ Msg_{AC} \}_{Ka^{-1}} \}_{Kc}$ e, usando sua chave privada, decifra a fórmula recebida. Após decifrar a fórmula, e aplicando a regra do significado da mensagem na suposição 3 e usando a função $H\{Msg_{AC}\}$ confirma a autenticidade e integridade da mensagem. Por fim, aplica a regra de verificação do identificador na suposição 6 usando a fórmula Ts_A para obter a credencial da entidade A (identificada pela fórmula $Cred_A$).

Na segunda mensagem, após receber e validar os dados enviados por A, a entidade C gera um desafio de autenticação, N_{CA}. O desafio consiste em fazer uma busca aleatória à tabela de credenciais e selecionar um código de credencial que esteja associado à entidade A. Em seguida grava-se o desafio, a data e hora de geração do desafio e a resposta esperada em uma base de dados. Na sequência, uma mensagem é enviada uma mensagem á entidade A. Tal mensagem, que contem o desafio e um *timestamp*, precisa ser assinada com a chave privada da entidade C e cifrada com a chave pública da entidade A. Logo, temos que:

Mensagem 2: $C \longrightarrow A : \{ Ts_C, N_{CA}, H\{ Msg_{CA} \}_{Kc^{-1}} \}_{Ka}$.

$A \triangleleft \{ Ts_C, N_{CA}, H\{ Msg_{CA} \}_{Kc^{-1}} \}_{Ka}$

$A \mid\equiv C \mid\sim H\{Msg_{CA}\}$

$A \mid\equiv C \mid\sim Ts_C$

$A \mid\equiv N_{CA}$

A entidade A recebe a fórmula $\{ Ts_C, N_{CA}, H\{ Msg_{CA} \}_{Kc^{-1}} \}_{Ka}$ e a decifra usando sua chave privada. Em seguida, aplicando a regra do significado da mensagem na suposição 1

usando a função $H\{Msg_{CA}\}$, confirma a autenticidade e integridade da mensagem. Por fim, aplica a regra de verificação do identificador na suposição 4, usando a fórmula Ts_C para obter o desafio N_{CA} gerado pela entidade C. Como resultado, A obtém o desafio de autenticação gerado pela entidade C.

Na terceira mensagem, a entidade A, após receber e validar o desafio gerado pela entidade C, envia a resposta $Resp_{AC}$, conforme solicitado. Essa resposta consiste em informar o código do desafio gerado pela entidade C e a credencial associada ao código de credencial solicitada pela entidade C. Além disso, a entidade A deve informar o código do serviço que deseja consumir. Dessa forma, temos:

Mensagem 3: $A \longrightarrow C : \{ Ts_A, Resp_{AC}, Cod_{Srv_A}, H\{ Msg_{AC} \}_{Ka^{-1}} \}_{Kc}$.

$C \triangleleft \{ Ts_A, Resp_{AC}, H\{ Msg_{AC} \}_{Ka^{-1}} \}_{Kc}$

$C \mid\equiv A \mid\sim H\{Msg_{AC}\}$

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

$C \mid\equiv Resp_{AC}$

A entidade C recebe a fórmula $\{ Ts_A, Resp_{AC}, H\{ Msg_{AC} \}_{Ka^{-1}} \}_{Kc}$ e a decifra usando sua chave privada. Em seguida, aplicando a regra do significado da mensagem na suposição 3 e usando a função $H\{Msg_{AC}\}$, confirma a autenticidade e integridade da mensagem. Por fim, aplica a regra de verificação do identificador na suposição 6 usando a fórmula Ts_A para obter os dados da resposta do desafio $Resp_{AC}$. Com isso, é possível validar a resposta enviada e autenticar a entidade A, dando início ao processo de autorização. Como resultado temos que a entidade C autentica a entidade A.

Na mensagem 4, após autenticar a entidade A, a entidade C procede com o processo de autorização, que consiste em verificar qual serviço a entidade está querendo consumir. Para isso, a entidade C verifica o código de serviço solicitado pela entidade A, Cod_{Srv_A}, que foi informado no envio da mensagem 3. Se a entidade A possuir os privilégios necessários para consumir o serviço requisitado, a entidade C gera uma nova credencial de autorização temporária para o serviço solicitado. Em seguida, a entidade C grava a credencial ($A \overset{C_{Aut}}{\rightleftharpoons} C$), o código do serviço que a entidade A está requerendo, a data e hora de criação, a data de expiração e o código do contrato da entidade A. Terminado esse procedimento, a entidade C, envia uma mensagem contendo a credencial ($A \overset{C_{Aut}}{\rightleftharpoons} C$), a data e hora de expiração da credencial (Exp_A) e um *timestamp* a entidade A. Esta mensagem é assinada com a chave privada da entidade C e cifrada com a chave pública da entidade A.

Mensagem 4: $C \longrightarrow A : \{ Ts_C, Exp_A, \#(A \overset{C_{Aut}}{\rightleftharpoons} C), H\{ Msg_{CA} \}_{Kc^{-1}} \}_{Ka}$.

$A \triangleleft C \longrightarrow A : \{ Ts_C, Exp_A, \#(A \overset{C_{Aut}}{\rightleftharpoons} C), H\{ Msg_{CA} \}_{Kc^{-1}} \}_{Ka}$.

$A \mid\equiv C \mid\sim H\{Msg_{CA}\}$

$A \mid\equiv C \mid\sim Ts_C$

$A \mid\equiv C \mid\sim Exp_A$
$A \mid\equiv C \Rightarrow \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$
$A \mid\equiv C \mid\equiv \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$
$A \mid\equiv \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$

A entidade A recebe a fórmula $\{ Ts_C, Exp_A, \#(A \overset{C_{Aut}}{\rightleftharpoons} C), H\{ Msg_{CA} \}_{Kc^{-1}} \}_{Ka}$ e a decifra usando sua chave privada. Em seguida, aplicando a regra do significado da mensagem na suposição 1 e usando a função $H\{Msg_{CA}\}$, confirma a autenticidade e integridade da mensagem. A entidade A aplica a regra de verificação do identificador nas suposições 4 e 7 usando as fórmulas Ts_C e Exp_A. E, finalmente, aplicando a regra da jurisdição, nas suposições 7 e 9, obtém a credencial de autorização temporária $\#(A \overset{C_{Aut}}{\rightleftharpoons} C)$.

Finalmente, na quinta mensagem, a entidade A após receber e validar a mensagem enviada pela entidade C, obtendo assim a credencial de autenticação e autorização temporária, $\#(A \overset{C_{Aut}}{\rightleftharpoons} C)$, envia uma mensagem à entidade B contendo a requisição do serviço que deseja consumir juntamente com a credencial de autorização temporária. Esta mensagem é assinada com a chave privada da entidade A e cifrada com a chave pública da entidade B.

Mensagem 5: $A \longrightarrow B : \{ Ts_A, (A \overset{C_{Aut}}{\rightleftharpoons} C), H\{ Msg_{AB} \}_{Ka^{-1}} \}_{Kb}, Req_A$.
$B \triangleleft \{ Ts_A, (A \overset{C_{Aut}}{\rightleftharpoons} C), H\{ Msg_{AB} \}_{Ka^{-1}} \}_{Kb}, Req_A$
$B \mid\equiv A \mid\sim H\{Msg_{AB}\}$
$B \mid\equiv A \mid\sim Ts_A$
$B \mid\equiv A \Rightarrow \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$
$B \mid\equiv A \mid\equiv \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$
$B \mid\equiv \#(A \overset{C_{Aut}}{\rightleftharpoons} C)$

Como resultado, a entidade B recebe a fórmula $\{ Ts_A, (A \overset{C_{Aut}}{\rightleftharpoons} C), H\{ Msg_{AB} \}_{Ka^{-1}} \}_{Kb}, Req_A$, e a decifra, usando sua chave privada. Em seguida, aplicando a regra do significado da mensagem na suposição 2 e usando a função $H\{Msg_{AB}\}$, confirma a autenticidade e integridade da mensagem. A entidade B aplica a regra de verificação do identificador na suposições 5 usando as fórmulas Ts_A. Finalmente, aplicando a regra da jurisdição nas suposições 8 e 10, obtém a credencial de autorização temporária $\#(A \overset{C_{Aut}}{\rightleftharpoons} C)$ e autoriza a entidade A a consumir a requisição Req_A. Como resultado, B autoriza A, a partir da nova credencial $(A \overset{C_{Aut}}{\rightleftharpoons} C)$, a consumir a requisição Req_A

4.3.2.4 Análise

A análise demonstra que o protocolo de autenticação e autorização proposto alcança os objetivos apresentados— ou seja, a autenticação da entidade A e a emissão da credencial de autenticação e autorização temporária. Isso permite que a entidade A consuma o

serviço requerido. É importante frisar que a lógica BAN foi utilizada para (a) comunicar de forma mais precisa a execução do protocolo e (b) permitir uma verificação mais precisa sobre possíveis falhas no protocolo.

4.4 Implementação

Após a definição do protocolo proposto, e para atender o interesse na realização de testes de desempenho que objetivam avaliar o impacto da solução, foi feita a implementação de um protótipo para o protocolo de autenticação e autorização. Esse protótipo foi desenvolvido em conjunto com um trabalho de conclussão de curso do aluno Alexandre Lucchesi (da Engenharia da Computação, UnB).

4.4.1 Protótipo

Para implementar o protótipo do protocolo proposto foi necessário desenvolver cada um dos componentes (Cliente, servidor de Autenticação e Autorizaçãoe servidor de fachada REST) descritos na arquitetura do protocolo na seção 4.2. Para isso, foi utilizada a linguagem de programação funcional Haskell, com o compilador GHC (Glasgow Haskell Compiler). Para o controle de dependências e gerenciamento de *build* de aplicações foi utilizada a ferramenta Cabal, que é um o gerenciador de pacotes do Haskell. Com ele é possível construir aplicações e bibliotecas de forma padronizada, organizada e portável. Para criar o protótipo ainda foi utilizado o framework Haskell para desenvolvimento web, denominado Snap *Framework* que é necessário para lidar com as requisições HTTP. Tais requisições são utilizadas pelo protocolo de autenticação e autorização proposto. Além disso, foi utilizado o HsOpenSSL que é um OpenSSL vinculativo para Haskell, utilizado para garantir a utilização do HTTPS em todas as trocas de mensagem a partir de código Haskell.

Para a assinatura digital das mensagens utilizadas pelo protocolo foi utilizado o algoritmo RSASSAPKCSv1_5 SHA256 conforme orientação descrita na publicação [24] e para criptografia foi utilizado o algoritmo de criptografia assimétrica, RSAESPKCS1-V1_5, para cifrar a chave simétrica utilizada no algoritmo de criptografia simétrica AES_128_CBC_HMAC_SHA_256 que foi utilizado para cifrar a mensagem. Procurou-se seguir a orientação do que é preconizado na publicação [23].

O servidor de banco de dados foi implementado utilizando o banco de dados não relacional *Apache CouchDB*, que é um banco de dados flexível e tolerante a falhas que usa *JSON* para armazenar os dados e JavaScript como sua linguagem de consulta usando o MapReduce. Além disso, o *Apache CouchDB* oferece uma API de acesso de acordo com o estilo REST.

O protótipo implementado não objetivou a otimização de código. Seu objetivo foi o de possibilitar a realização dos testes de desempenho em uma configuração mais rudimentar em termos de desempenho, o que leva a possibilidades de melhoria em relação ao processamento das requisiçõe. Além disso, o protótipo permitiu verificar as funcionalidades do protocolo proposto.

4.5 Síntese do capítulo

Este capítulo apresentou os requisitos e a arquitetura do protocolo de autenticação e autorização proposto. Além disso, o capítulo apresentou a formalização do protocolo utilizando à lógica BAN. Finalmente, foi apresentada de forma sucinta a implementação de um protótipo para protocolo proposto. No próximo capítulo fazemos uma análise de segurança do protocolo proposto (conforme o trabalho apresentado em [30]) e apresentamos uma análise de desempenho a fim de mensurar o impacto da adoção protocolo de autenticação e autorização proposto.

Capítulo 5

Análise de Segurança e Desempenho

Com a definição da arquitetura de referência REST e do protocolo de autenticação e autorização (Capítulo 4), tornou-se necessária a realização de uma análise de segurança e de experimentos que possibilitam a mensuração do impacto da sua utilização no tratamento das requisições realizadas à PCDF.

Este capítulo inicialmente apresenta uma análise dos mecanismos de segurança propostos no protocolo. Essa primeira análise segue um procedimento informal, mas está de acordo com trabalhos descritos na literatura [] citar dissertações, teses e artigos que fazem uma análise similar.. Em seguida, este capítulo apresenta uma avaliação empírica que busca analisar o desempenho do protocolo de Autenticação e Autorização Proposto.

5.1 Análise de Segurança

Nesta seção serão discutidas as propriedades de segurança do protocolo de autenticação e autorização proposto. São abordadas as propriedades e alguns ataques que podem ser realizados. Essa seção segue a estrutura e se fundamenta (parcialmente) no discurso utilizado na análise do protocolo Traust [30].

5.1.1 Segurança da sessão

O protocolo de Autenticação e Autorização proposto utiliza para segurança de sessão e camada de transporte, o protocolo TLS/SSL. A utilização desse protocolo tem por objetivo evitar o ataque *man-in-the-middle*. Para isso, é exigido, tanto para o órgão conveniado quanto para a própria PCDF, a utilização de certificados digitais padrão X.509, emitidos e garantidos por uma CA que esteja subordinada à hierarquia da ICP-Brasil.

Com isso, ambas as partes envolvidas no processo de comunicação podem estabelecer um processo de confiança mútua no nível de transporte. Todo o tráfego que flui sobre uma

sessão bilateral certificada tem uma fonte confiável. Isso permite que implementações de serviços possam autorizar ou desautorizar interações com base na fonte bem conhecida de uma solicitação HTTP.

Além da segurança oferecida pela utilização da segurança na camada de transporte, com utilização do TLS/SSL, o protocolo utiliza mecanismos de segurança tais como criptografia assimétrica e assinatura digital. Isso Permite que outras partes mal intencionadas não consigam ter acesso ao conteúdo das mensagens trocadas pelo protocolo no processo de comunicação.

Diferentemente do sistema Traust [30], o protocolo de Autenticação e Autorização proposto, será executado apenas em ambientes que tanto o cliente quanto o servidor possuem chaves públicas certificadas. Procura-se, dessa forma, atenuar problemas relacionados ao protocolo TLS, que quando executado em ambientes em que as partes não possuem chaves públicas certificadas, relacionados a ataques *man-in-the-middle* durante o estabelecimento da sessão [30].

5.1.2 Responsabilização dos usuários conveniados

Uma das ameaças verificadas diz respeito a possibilidade do uso indevido, por parte dos órgãos conveniados, das informações disponibilizadas nos serviços ofertados pela PCDF. Isso decorre porque as informações disponibilizadas são sensíveis e possuírem caráter sigiloso. Dessa forma, para evitar esse problema é exigido dos órgãos conveniados que assinem um contrato para o consumo do serviço. No momento da assinatura do contrato as instituições recebem uma tabela contendo várias credenciais que servem como identidade e que deverão ser utilizadas no processo de autenticação, conforme descrito no seção 4.1. Isso possibilita que ocorra uma autenticação mútua entre a PCDF e os consumidores dos serviços. Além disso, com a utilização do protocolo de autenticação e autorização proposto, são empregados mecanismos de criptografia e assinaturas digitais que são geradas a partir de certificados digitais padrão X.509 vinculados aos órgãos e garantidos por AC. Dessa forma, busca-se evitar o não-repúdio por parte das instituições conveniadas,atribuindo-lhes responsabilidades em caso do mau uso dos serviço ofertados pela PCDF. Logo, uma vez detectado algum tipo de vazamento de dados proveniente dos serviços ofertados o órgão poderá ser identificado e após uma apuração minuciosa, responsabilizado pelos danos causados à PCDF.

5.1.3 Ataques de repetição

Esta ameaça, conforme descrita na Seção 3.5, se empregada contra o protocolo de autenticação e autorização proposto, tem baixa probabilidade de sucesso. Isso porque

utilizamos *timestamps* na maior parte das mensagens trocadas pelo protocolo. Além disso, são gerados números únicos, que identificam os desafios de autenticação gerados e que podem ser utilizados como *nonces*, valores gerados de forma aleatória e que podem ser empregados contra esse tipo de ataque.

5.1.4 Ataques de negação de serviço

Esta ameaça é muito difícil de ser evitada, uma vez que pode ser fruto de ataques via rede, do consumo excessivo de recursos da máquina ou, ainda, ser resultante da exploração de qualquer tipo de vulnerabilidade que implique na indisponibilidade do serviço ou de um recurso. O protocolo de autenticação e autorização proposto é baseado no esquema de desafio-resposta. Os desafios são gerados de forma aleatória a partir das credenciais que identificam unicamente os consumidores dos serviços, o que possibilita uma autenticação mútua. Além disso, também são empregados outros mecanismos de segurança como utilização da segurança na camada de transporte e mecanismos de criptografia e assinaturas digitais. Isso minimiza a ameaça de ataques de negação de serviço. Assim como o protocolo Traust [30], o servidor de autenticação e autorização proposto pode ser replicado para outros servidores, o que possibilita um balanceamento de carga e minimiza a criação de gargalos. Isso permite que o servidor de autenticação seja escalável e esteja disponível em situações críticas, como no caso dos ataques de negação de serviço.

5.1.5 Roubo de credenciais de autenticação e autorização

O protocolo de autenticação a autorização proposto, após executado corretamente, emite um token de segurança que será utilizado para a obtenção do serviço desejado, conforme apresentado na seção 4.2. Cabe ressaltar que se um atacante conseguir burlar os mecanismos de segurança utilizados pelo protocolo e conseguir acesso a essa credencial de autenticação e autorização, o atacante terá acesso apenas um serviço, pois o protocolo emite credenciais para um único serviço por vez e com tempo de expiração determinado no momento de sua geração. Essa é uma situação muito difícil de ser verificada, porém caso o extravio de credenciais ocorra, buscamos minimizar o problema com o isolamento de uma credencial por serviço requisitado, diminuindo o impacto que pode ocorrer.

Outro problema que pode ocorrer é o roubo de credenciais de autenticação, que são as credenciais que o órgão conveniado recebe no momento da assinatura do contrato de oferecimento do serviço. Essa credenciais são utilizadas para responder aos desafios que serão realizadas pelo protocolo de autenticação proposto. Caso o extravio o ocorra e seja identificado, a PCDF pode, de forma flexível e transparente, desativar as credenciais que

julga terem sido comprometidas sem que o usuário seja prejudicado. Em um caso mais extremo, outras credenciais podem ser geradas e redistribuídas às instituições conveniadas.

Porém, cabe ressaltar que, caso ocorra o extravio de credenciais, o órgão que teve o problema será investigado e poderá ser responsabilizado— caso seja detectada uma má gestão na guarda das credenciais, conforme descrito na subseção 5.1.5

5.1.6 Ponto único de ataque

Uma das possibilidades verificadas é que, ocorrendo um ataque, o alvo possa não ser protocolo de autenticação e autorização proposto e sim o próprio servidor de Autenticação e Autorização. Neste caso, se o atacante for bem sucedido o servidor poder ficar vulnerável. Esse servidor é responsável pela geração dos desafios de autenticação e pela geração das credenciais de autenticação e autorização. Porém, apesar de realizar estas atividades ele armazena e consulta os dados gerados no processo de autenticação e autorização em outra máquina, que é em um servidor de banco de dados. Em outras palavras, o servidor de Autenticação e Autorizaçãoestá em uma máquina diferente do servidor de banco de dados, o que minimiza os problemas relacionados a esse tipo de ataque, pois o servidor pode ser replicado para outra máquina e o que estiver com problemas pode ser facilmente substituído.

5.1.7 Síntese da análise de segurança

- descrever os pontos positivos
- descrever os pontos que requerem mais atenção

5.2 Testes de desempenho

Testes de desempenho são definidos como investigações técnicas realizadas para determinar a capacidade de resposta, a confiabilidade ou escalabilidade de um sistema, sob uma determinada carga de trabalho [35]. A análise de desempenho possibilita identificar problemas que geralmente são encontrados em sistemas computacionais. Esses problemas podem ser agrupados em termos de comparação e configuração de sistemas, identificação de gargalos, caracterização de cargas de trabalho e a previsão de desempenho [21]. Dessa forma, foram conduzidos experimentos com o objetivo de analisar o desempenho do protocolo de autenticação e autorização proposto. O restante dessa seção descreve os procedimentos e os resultados obtidos com a análise de desempenho.

5.2.1 Objetivos, Questões e Métricas

A avaliação de desempenho do protocolo de autenticação e autorização proposto foi organizada com o uso da abordagem GQM (*Goals, Questions, and Metrics*) [2]. O resultado da aplicação da GQM é a especificação de um sistema de medição visando um conjunto particular de problemas e um conjunto de regras para a interpretação dos dados de medição [2]. O modelo de avaliação resultante tem três níveis: nível conceitual, onde são definidos os objetivos da medição; nível operacional, que é aquele onde são verificadas as questões que caracterizam o objeto da medição; e o nível quantitativo, que é o nível onde são definidas as métricas que identificam a medidas necessárias para responder as questões levantadas [2].

5.2.1.1 Objetivos

A realização de testes de desempenho objetivam determinanr se o desempenho de um algoritmo, protocolo ou sistema está de acordo com os requisitos. Outro objetivo dos testes de desempenho é o de validar a capacidade de resposta, a vazão, a confiabilidade e a escalabilidade de um sistema sob uma determinada carga [35]. Em relação ao trabalho desenvolvido nessa dissertação, a avaliação de desempenho do protocolo de autenticação e autorização proposto tem como objetivo:

- Verificar o impacto do protocolo de autenticação e autorização proposto;
- Identificar a viabilidade do uso do protocolo em cenários de carga esperados para a PCDF.

5.2.1.2 Questões

Seguindo o que é preconizado na metodologia GQM, os objetivos são definidos em um nível abstrato, de forma que devem ser formuladas questões, em um nível operacional, que têm por finalidade responder se os objetivos definidos serão atendidos. Logo, com base nos objetivos formulados para a análise de desempenho do protocolo de autenticação e autorização proposto, as seguintes questões foram investigadas:

(Q1) *Qual o impacto observado no tempo de resposta às requisições com o uso do protocolo?*

(Q2) *Um protótipo funcional, sem foco em otimização, consegue suportar a demanda prevista?*

5.2.1.3 Métricas

O desempenho é descrito quantitativamente por meio de métricas. Uma métrica pode ser definida como um conjunto de dados que é definido para responder uma questão de maneira quantitativa. Dessa forma, foram utilizadas as seguintes métricas na avaliação do desempenho do protocolo de autenticação e autorização proposto.

- quantidade de usuários: variável de controle que representa a quantidade de usuários utilizados na avaliação de desempenho.
- utilização do protocolo: variável de controle que indica a utilização ou não do protocolo proposto na dissertação. Ela é definida em dois níveis, SIM ou NÃO.
- tempo médio de resposta: variável de resposta que consiste no tempo entre a solicitação de um serviço por um usuário até o momento em que ele recebe uma resposta completa [37]. Para a análise de desempenho do protocolo proposto o tempo médio será dado em milissegundos.
- Vazão (*throughput*): variável de resposta que corresponde ao número de operações que podem ser tratadas pelo protocolo em um determinado período de tempo [37].

Dessa forma, após definir as métricas utilizadas na avaliação de desempenho, foi criado um plano de medição que descreve como as variáveis de resposta serão mensuradas e quais procedimentos serão realizados durante o período de execução dos experimentos.

Logo, o primeiro passo foi a criação de um serviço REST, que retorna informações sobre ocorrências policiais, tais como: quais tipos de ocorrências criminais estão registradas, dados gerais da vítima, autor, dentre outras informações. Esse serviço é consumido no momento da execução dos testes de desempenho da solução proposta. Foram realizados 10 cenários de testes, conforme representado na Tabela 5.1.

N. de usuários	Sem utilização do protocolo	Com utilização do Protocolo
10	Teste 1	Teste 2
20	Teste 3	Teste 4
30	Teste 5	Teste 6
40	Teste 7	Teste 8
50	Teste 9	Teste 10

Tabela 5.1: Cenários de testes realizados

Para a execução dos testes, que utilizam as métricas definidas, foram estabelecidas as seguintes regras: Cada teste deve realizar várias requisições ao serviço REST criado e o número de requisições é obtido pela fórmula $numeroUsuarios \times 100$. A frequência

de lançamento das threads, que representam os usuários virtuais, é definido pela fórmula $numeroUsuarios \times 2s$. Exemplificando, de acordo com a Tabela 5.1, ao selecionar o Teste 1, serão executadas 10 threads a cada 20 segundos. Importa salientar que cada thread corresponde a um usuário, é iniciada 2 segundos após a thread anterior e executa 100 requisições— o que totaliza 1000 requisições ao serviço no cenário Teste 1. Esse procedimento foi adotado para os demais cenários de testes. Por fim, as variáveis de resposta adotadas para a análise de desempenho, conforme discutido, foram o tempo médio de resposta das requisições e a vazão média de requisições por segundo.

5.2.2 Configuração do ambiente de teste

O ambiente de teste envolveu estações de trabalho tanto do Laboratório de Engenharia de Software da Universidade de Brasília quanto na Divisão de Tecnologia da Policia Civil do Distrito Federal. Para a realização dos testes foram utilizadas configurações distintas de computadores, conforme apresentado na tabela 5.2.

Estação	Quantidade	Configuração
Servidor de Autenticação e Autorização	1	Desktop DELL Intel Core I5-2450 2,5 GHz, 4 Gb RAM, 500 HD
Servidor de Fachada REST	1	Desktop DELL Intel Core I5-2450 2,5 GHz, 4 Gb RAM, 500 HD
Cliente REST	1	Desktop DELL Intel Core I5-2450 2,5 GHz, 4 Gb RAM, 500 HD
Servidor do Serviço web REST PCDF	1	Intel Xeon E7 4870 2,4 GHz, 8 Gb RAM, 120 HD

Tabela 5.2: Ambiente utilizado na análise de desempenho

Os servidores de Autenticação e Autorização, Fachada REST e Cliente REST, foram prototipados utilizando a linguagem de programação funcional *Haskell*. Estes servidores acessam uma base de dados não relacional *CouchDB*, conforme apresentado na seção 4.4. O sistema operacional utilizado nessas máquinas foi o Linux Ubuntu 12.04 LTS-64 bits. No caso do serviço REST desenvolvido pela PCDF, para atender a demanda da análise de desempenho, foi desenvolvido utilizando a linguagem C# acessando um banco de dados *SQL Server 2008 r2* que mantém os registros das ocorrências policiais. O serviço foi publicado em um servidor que utiliza o sistema operacional *Windows Server 2008 r2, Enterprise Edition x64*, utilizando o ISS 7.0 como servidor Web. A Figura 5.1 descreve os ambientes de configuração utilizados na análise de desempenho do (tem que padronizar. em alguns pontos está maiúsculo, em outros minúsculo)Protocolo de Autenticação e Autorização proposto.

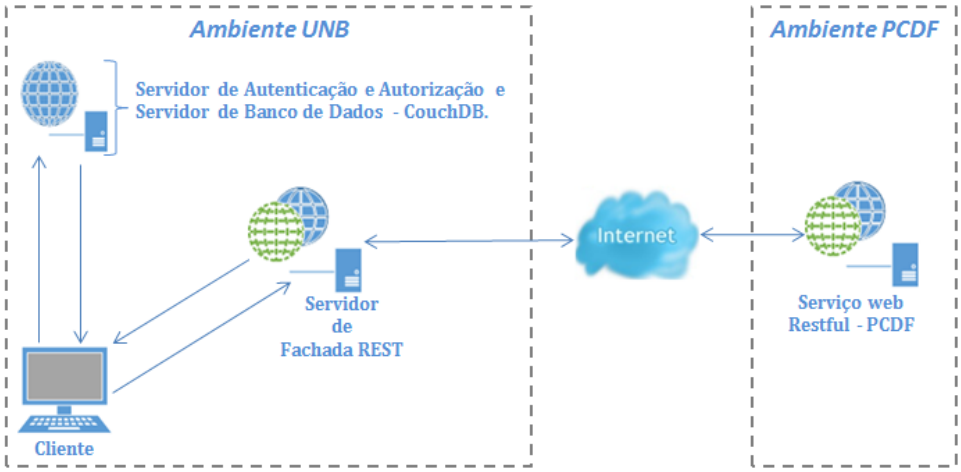

Figura 5.1: Ambiente de teste de desempenho do Protocolo de Autenticação e Autorização.

5.2.3 Análise dos resultados

Os testes objetivam (a) verificar o impacto no tempo de resposta às requisições com o uso do protocolo e (b) avaliar se um protocolo funcional, sem foco em otimização, suportaria a demanda de requisições prevista para a PCDF. Para atender aos objetivos supracitados, foram realizados testes considerando requisições diretas, sem o uso do protocolo de autenticação e autorização; e requisições seguras e que seguem as trocas de mensagens definidas no protocolo. Conforme discutido nas seções anteriores, as análises resultaram em um universo de amostras com 1000, 2000, 3000, 4000 e 5000 requisições, correspondendo respectivamente a grupos de 10, 20, 30, 40 e 50 usuários. Os dados coletados foram analisados, e resultados estatísticos são apresentados nas tabelas 5.3 e 5.4.

Dessa forma, com o objetivo de responder a primeira questão relacionada ao teste de performance (*Qual o impacto observado no tempo de resposta às requisições com o uso do protocolo?*), consideramos que os resultados obtidos evidenciam um impacto significativo com a utilização do protocolo de autenticação e autorização proposto. Pode-se observar que, com a utilização do protocolo para 10 usuários simultâneos, o tempo médio de resposta é de 683,84 milissegundos. Por outro lado, quando comparado ao acesso ao serviço sem a utilização do protocolo, o tempo médio de resposta às requisições é de 42,26 milissegundos. Estas comparações estendem-se aos demais cenários de testes, para 20, 30, 40 e 50 usuários. Em todos casos o impacto é significativo quando comparado aos resultados obtidos sem a utilização do protocolo de autenticação e autorização. Porém, essa variação era de certa forma esperada, pois o protocolo proposto incorpora mecanismos de segurança que requerem algoritmos de criptografia e assinatura digital e segurança na

camada de transporte com a utilização do SSL/TLS. Além disso, a implementação do protocolo será ainda alvo de otimizações.

De outra forma, ao analisar o tempo médio dos experimentos que utilizaram o protocolo, nota-se que este tempo é aceitável, pois com 10 usuários ele fica abaixo de 1 segundo, com 20 o tempo médio é 1,5 segundos, com 30 usuários esse tempo sobe para aproximadamente 2,5 segundos, aumentado de forma aceitável até 50 usuários em que o tempo médio verificado foi de aproximadamente 4,5 segundos, conforme apresentado nas tabelas 5.3 e 5.4 e no gráfico 5.2. De forma que esses resultados estão dentro do esperado para utilização. Atualmente o número de convênios na PCDF não supera 10 usuários, estima-se que com a utilização do protocolo esse número suba para no máximo 30 órgãos conveniados. Logo, com a análise dos resultados percebe-se que, apesar do impacto da utilização do protocolo de autenticação e autorização, o tempo médio de resposta não configura como um limitador para sua utilização. Relaciona com os estudos que avaliam o impacto no tempo de resposta com a adoção de WS-Security, WS-*.

Qtd Usuários	Tempo Médio	Erro Padrão	Mediana	Desv Padrão	Qtd Req.
10	683,84	12,53	622,5	396,36	1000
20	1.431,14	21,63	1.274,5	967,30	2000
30	2.466,28	35,86	2.067	1964,30	3000
40	3.249,76	35,83	3.084,5	2266,50	4000
50	4.677,64	44,84	4.375,5	3170,80	5000

Tabela 5.3: Análise de desempenho considerando o protocolo de autenticação e autorização.

Qtd Usuários	Tempo Médio	Erro Padrão	Mediana	Desv Padrão	Qtd Req.
10	42,26	1,12	43	35,51	1000
20	40,96	0,65	39	29,35	2000
30	42,79	1,20	37	65,97	3000
40	40,60	0,40	43	25,69	4000
50	42,64	0,48	44	34,39	5000

Tabela 5.4: Análise de desempenho sem considerar o protocolo de autenticação e autorização.

Além disso, a quantidade de usuários simultâneos tem impacto no desempenho da aplicação SOA que utiliza o protocolo de autenticação e autorização proposto. Verifica-se que o tempo médio de resposta sobe de acordo com a quantidade de usuários. No caso da não utilização do protocolo, observa-se que o tempo médio de resposta é constante e que a variação não é tão significante com o aumento do número de usuários simultâneos, conforme observado na figura 5.2.

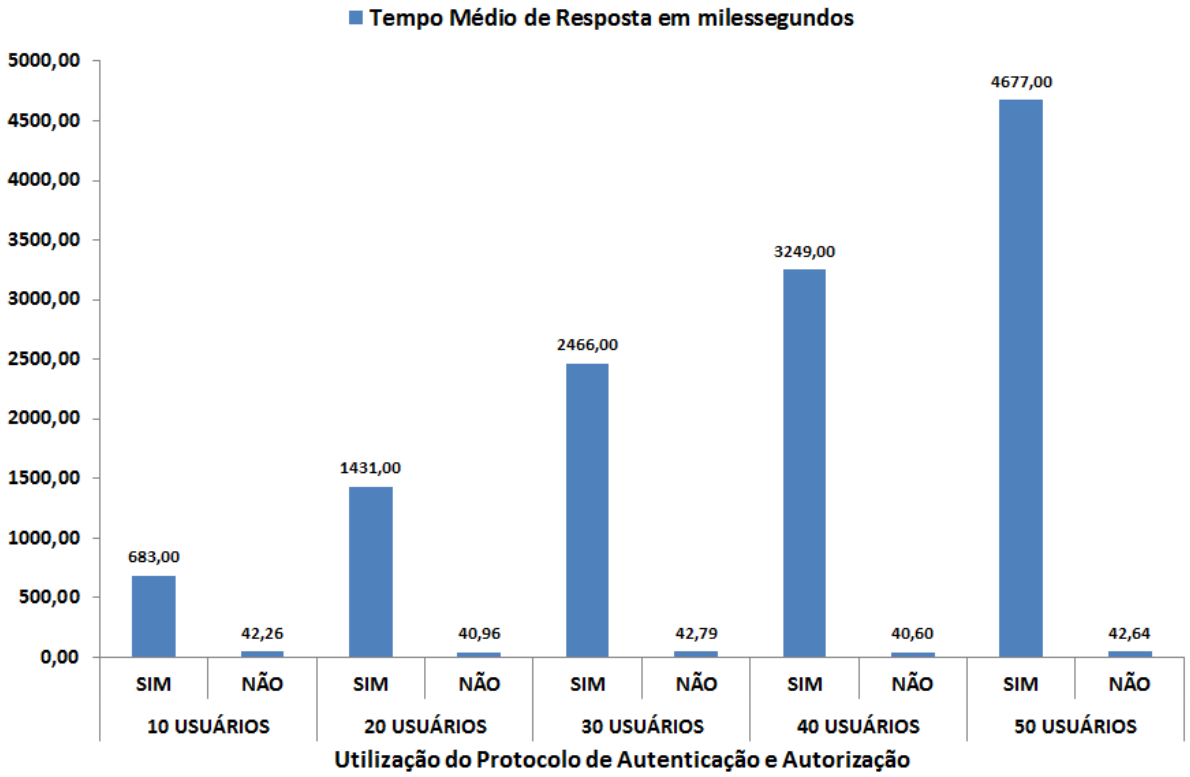

Figura 5.2: Fluxo do protocolo de autenticação/autorização proposto, 1º cenário.

Para responder a segunda questão relacionada à análise de desemepenho (*Um protótipo funcional, sem foco em otimização, consegue suportar a demanda prevista?*), novamente foram realizadas análises dos dados coletados nos testes realizados. A análise dos resultados obtidos com a execução do experimento ocorreu com a observação do tempo médio de resposta e com a verificação da vazão— ou seja, o número de requisições atendidas por segundo, que foi coletado ao final da execução do Teste 10, aplicado a um grupo de 50 usuários, conforme apresentado na Tabela 5.1. Dessa forma, foram coletadas 5000 amostras que utilizaram o protocolo de autenticação e autorização proposto em um período de 11 minutos por que 11 minutos?. Os resultados estatísticos são apresentados na tabela 5.5.

Observa-se que a PCDF, em um cenário extremo, pode ter que responder a uma média de 1200 requisições de um serviço que fornece informações sobre ocorrências policiais em um período de uma hora, ou seja, duas requisições por segundo. Neste caso, os resultados obtidos com o experimento realizado com os 50 usuários, demonstrou que o tempo médio de resposta foi de 4677 milissegundos (ou aproximadamente 4,677 segundos) e que a vazão média, que é o número de requisições atendidas por segundo, foi de 8,62 requisições por segundo. Ao se realizar a comparação deste cenário com o cenário extremo vivenciado pela PCDF, verificou-se que a arquitetura, do ponto de vista de vazão, atende ao cenário

mais crítico, que é de 2 duas requisições por segundo. O que garante a qualidade do serviço ofertado pelo protocolo de autenticação e autorização proposto.

essa tabela está muito ruim. sugiro remover

Informação	**Valor**
Nº de Requisições	5000
Média	4677
Mínimo	118
Máximo	21665
Desvio Padrão	3170,49
% Erro	0,0
Vazão/seg	8,62
Kbps	11,8
Média de Bytes	1401,05

Tabela 5.5: Estatística básica com a utilização do protocolo de autenticação e autorização para 50 usuários.

5.3 Síntese do capítulo

revisa essa síntese. está furada. fala apenas do teste de desempenho

Neste capítulo foi realizada uma análise de desempenho, onde foram definidos objetivos e questões que foram respondidas após a execução e análise dos testes realizados. O estudo evidenciou que o impacto no desempenho é significativo quando comparado com a não utilização da proposta. Porém os tempos médios obtidos estão dentro de um tempo médio esperado, que é abaixo de 5 segundos. Observou-se ainda, que em um cenário extremo de utilização, onde a PCDF tenha que atender a uma média de 2 duas requisições por segundo, e um período de 1 uma hora, ou seja 1200 requisições, a arquitetura proposta atende perfeitamente, pois consegue atender a 8,62 requisições por segundo em um cenário extremo onde foram realizados 5000 requisições em aproximadamente em 10 minutos.

Referências

[1] Maarten Van Steen Andrew S. Tanenbaum. *Sistemas Distribídos Principios e Paradigmas 2ª Edição*. São Paulo, 2007. 19

[2] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994. 60

[3] T. Berners-Lee, R. Fielding, and L. Masinter. Rfc 3986, uniform resource identifier (uri): Generic syntax, 2005. 21

[4] Elisa Bertino, Lorenzo Martino, Federica Paci, and Anna Cinzia Squicciarini. *Security for Web Services and Service-Oriented Architectures*. Springer, 2010. xi, 17, 18, 19, 22, 23, 33

[5] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Mike Champion, Christopher Ferris, and David Orchard. Web services architecture. World Wide Web Consortium, Note NOTE-ws-arch-20040211, February 2004. 16

[6] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, February 1990. xii, 46, 47, 48

[7] Ethan Cerami. *Web Services Essentials*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1st edition, 2002. 19

[8] Paul Clements, David Garlan, Felix Bachmann, James Ivers, Judith Stafford, Len Bass, and Paulo Merson. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Professional, 2nd edition, 2010. 15

[9] Marijke Coetzee. Towards a holistic information security governance framework for soa. *2012 Seventh International Conference on Availability, Reliability and Security*, 0:155–160, 2012. 10, 14

[10] F. P Coyle. *XML, Web Services, and the Data Revolution*. Boston: Addison-Wesley, 2002. 17

[11] Stefania DAgostini, Valentina Di Giacomo, Claudia Pandolfo, and Domenico Presenza. An ontology for run-time verification of security certificates for soa. *2012 Seventh International Conference on Availability, Reliability and Security*, 0:525–533, 2012. 10

[12] Nelly A. Delessy and Eduardo.B Fernandez. *A pattern-driven process for secure service-oriented applications*. PhD thesis, Boca Raton, FL, USA, 2008. 12, 14

[13] T. Erl. *Soa Principios De Design De Serviços*. PRENTICE HALL BRASIL, 2009. 15, 17

[14] T. Erl, B. Carlyle, C. Pautasso, R. Balasubramanian, H. Wilhelmsen, and D. Booth. *SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST*. The Prentice Hall Service Technology Series from Thomas Erl. Pearson Education, 2012. 16

[15] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000. AAI9980887. 20

[16] B.A. Forouzan and F. Mosharraf. *Redes de Computadores: Uma Abordagem Top-Down*. 2013. 26, 28

[17] M.T. Goodrich and R. Tamassia. *Introdução à Segurança de Computadores*. Bookman, 2013. 27

[18] E. Hammer-Lahav. The OAuth 1.0 Protocol. Technical Report 5849, RFC Editor, Fremont, CA, USA, April 2010. 34

[19] Dick Hardt. The OAuth 2.0 authorization framework. RFC 6749, RFC Editor, Fremont, CA, USA, October 2012. xi, 32, 34

[20] M. Harwood. *Security Strategies in Web Applications and Social Networking*. Information Systems Security & Assurance. Jones & Bartlett Learning, 2010. 29, 30

[21] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley Professional Computing. Wiley, 1991. 59

[22] Meiko Jensen, Nils Gruschka, Ralph Herkenhöner, and Norbert Luttenberger. Soa and web services: New technologies, new standards - new attacks. In *ECOWS*, pages 35–44. IEEE Computer Society, 2007. 30

[23] M. Jones, D. Balfanz, J. Bradley, Y. YaronGoland, J. Panzer, N. Sakimura, and P. Tarjan. JSON Web Token (JWT). Internet-Draft draft-jones-json-web-token-20.txt,, Internet Engineering Task Force, April 2014. Work in progress. 54

[24] M. Jones, J. Bradley, and N. Sakimura. JSON Web Signature (JWS). Internet-Draft draft-ietf-jose-json-web-signature-23, Internet Engineering Task Force, March 2014. Work in progress. 54

[25] Nicolai M. Josuttis. *SOA in Practice: The Art of Distributed System Design*. O'Reilly Media, Inc., Beijing, 2007. 15

[26] Ramarao Kanneganti and Prasad Chodavarapu. *SOA security*. Manning Publications Co., Greenwich, CT, USA, 2008. 23

[27] David Kim and Michael G. Solomon. *Fundamentals of Information Systems Security*. Jones & Bartlett Learning, 2010. 30, 31

[28] Barbara Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007. 6, 7

[29] Badrinarayanan Lakshmiraghavan. *Pro ASP.NET Web API Security: Securing ASP.NET Web API*. Apress, Berkely, CA, USA, 1st edition, 2013. 31, 32

[30] Adam J. Lee, Marianne Winslett, Jim Basney, and Von Welch. The traust authorization service. *ACM Trans. Inf. Syst. Secur.*, 11(1), 2008. 35, 55, 56, 57, 58

[31] Lutz Lowis and Rafael Accorsi. Vulnerability analysis in soa-based business processes. *IEEE Transactions on Services Computing*, 4(3):230–242, 2011. 10

[32] Paul Madsen, Eve Maler, Thomas Wisniewski, Tony Nadalin, Scott Cantor, Jeff Hodges, and Prateek Mishra. Saml v2.0 executive overview. Technical report, April 2005. 33

[33] E.A. Marks and M. Bell. *Service Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology*. Guías de España. Wiley, 2006. 5

[34] Catherine A. Meadows. Formal verification of cryptographic protocols: A survey. pages 133–150. Springer-Verlag, 1995. 45

[35] J. Meier, Carlos Farre, Prashant Bansode, Scott Barber, and Dennis Rea. *Performance testing guidance for web applications: patterns & practices*. Microsoft Press, Redmond, WA, USA, 2007. 59, 60

[36] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996. 23

[37] Ian Molyneaux. *The Art of Application Performance Testing: Help for Programmers and Quality Assurance*. O'Reilly Media, Inc., 1st edition, 2009. 61

[38] Nils Agne Nordbotten. Xml and web services security standards. *IEEE Communications Surveys and Tutorials*, 11(3):4–21, 2009. 33

[39] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. "big"' web services: Making the right architectural decision. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 805–814, New York, NY, USA, 2008. ACM. 22

[40] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77, Swinton, UK, UK, 2008. British Computer Society. 5

[41] Eric Pulier and Hugh Taylor. *Understanding Enterprise SOA*. Manning Publications, 2005. 16

[42] David Recordon and Drummond Reed. Openid 2.0: A platform for user-centric identity management. In *Proceedings of the Second ACM Workshop on Digital Identity Management*, DIM '06, pages 11–16, New York, NY, USA, 2006. ACM. 32

[43] Leonard Richardson and Sam Ruby. *Restful Web Services*. O'Reilly, first edition, 2007. 20

[44] Bruce Schneier. *Applied Cryptography (2Nd Ed.): Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995. xi, 23, 24, 25

[45] W. Stallings. *Criptografia e segurança de redes: princípios e práticas*. Pearson Prentice Hall, 2008. xi, 24, 25, 26, 28

[46] The OASIS technical commitee. XACML: eXtensible Access Control Markup Language, 2005. 35

[47] Paulo Verissimo and Luis Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, Norwell, MA, USA, 2001. 22, 29

[48] J Webber, I Robinson, and S Parastatidis. *REST in Practice: Hypermedia and Systems Architecture*. O'Reilly, 2010. 31

[49] Sam Weber, Paula Austel, and Michael McIntosh. 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*, pages 102–109. IEEE Computer Society, 2007. 10, 14

[50] Jie Xu, Dacheng Zhang, Lu Liu, and Xianxian Li. Dynamic authentication for cross-realm soa-based business processes. *IEEE Transactions on Services Computing*, 5(1):20–32, 2012. 11, 12

[51] Yang Zhang and Jun-Liang Chen. A delegation solution for universal identity management in soa. *IEEE Transactions on Services Computing*, 4(1):70–81, 2011. 10