

**UNIVERSIDADE FEDERAL DE SANTA CATARINA  
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA  
BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

**Giovani Milanez Espindola**

**Um Framework Orientado a Objetos para Gestão de  
Certificados de Atributos nos Padrões da ICP-Brasil**

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação.

**Jeandré Monteiro Sutil**  
Orientador

**Prof. Ricardo Felipe Custódio, Dr.**  
Co-Orientador

Florianópolis, julho de 2014

# **Um Framework Orientado a Objetos para Gestão de Certificados de Atributos nos Padrões da ICP-Brasil**

Giovani Milanez Espindola

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Sistemas de Informação e aprovada em sua forma final pelo Departamento de Informática e Estatística da Universidade Federal de Santa Catarina.

---

Prof. Leandro J. Komosinski

Coordenador do Curso

Banca Examinadora

---

Ruy Ramos

---

Prof. Ricardo Pereira e Silva, Dr.

---

*... Quando verificares, com tristeza, que nada sabes, terás  
feito teu primeiro progresso no aprendizado.  
Jigoro Kano*

# Resumo

O certificado de atributo X.509 é um documento eletrônico assinado digitalmente que qualifica uma entidade, sendo muito utilizado para estabelecer atribuições, ou privilégios a uma pessoa, organização ou mesmo um bem material. Em julho de 2012, a tecnologia foi padronizada para uso no Brasil pela ICP-Brasil - Infraestrutura de Chaves Públicas Brasileira. Por ser uma tecnologia relativamente nova no país, o suporte para desenvolvimento de sistemas que utilizem certificado de atributo X.509 ainda carece de estudos e referências de implementação. Este trabalho tem por objetivo prover uma orientação no uso da tecnologia, que venha a facilitar sua adoção, fornecendo um suporte para desenvolvimento de aplicações que precisam lidar com certificados de atributo X.509, em aderência aos normativos da ICP-Brasil. O autor propõe um protocolo de requisição e resposta para emissão, busca e revogação desses certificados. Tal protocolo foi implementado na forma de um framework orientado a objetos que permite a construção de aplicações que gerenciem o ciclo de vida de certificados de atributos. Este framework foi utilizado na construção de uma aplicação protótipo, que utiliza os serviços de uma Entidade Emissora de Atributos no controle da bilhetagem de sessões de cinema. Como resultado do trabalho, foi produzida ainda uma aplicação para dispositivos móveis, capaz de validar e armazenar os tickets emitidos para um usuário, permitindo que este os apresente na entrada do cinema.

**Palavras chave:** Certificado de atributos, ICP-Brasil, Framework, Entidade Emissora de Atributos, X.509, Criptografia assimétrica, Autenticidade, Autorização, Integridade.

# Lista de Figuras

2.1	Funcionamento do algoritmo de hash . . . . .	8
2.2	Confidencialidade com criptografia assimétrica . . . . .	9
2.3	Criptografia simétrica . . . . .	9
2.4	Processo de assinatura digital e verificação . . . . .	11
2.5	Hierarquia de uma ICP . . . . .	12
2.6	Biblioteca vs Framework . . . . .	15
2.7	Uso Biblioteca vs Framework . . . . .	16
2.8	Aplicação desenvolvida utilizando framework . . . . .	17
3.1	Modelo <i>push</i> de distribuição de CA . . . . .	23
3.2	Modelo <i>pull</i> de distribuição de CA . . . . .	23
3.3	Ciclo de vida do Certificado de Atributo . . . . .	25
3.4	Ciclo de vida do Certificado de Atributo ICP-Brasil . . . . .	27
4.1	Protocolo de requisição . . . . .	29
5.1	Interfaces do framework . . . . .	45
5.2	Utilização do Framework . . . . .	46
6.1	Arquitetura da solução de ingressos . . . . .	57
6.2	Interface de gerenciamento de modelos de tickets . . . . .	59
6.3	Interface de gerenciamento de estudantes . . . . .	60
6.4	Lista dos certificados de atributos emitidos . . . . .	60
6.5	Interface de gerenciamento de sessões . . . . .	61

6.6	Interface de gerenciamento de tickets . . . . .	62
6.7	Interface de emissão de tickets . . . . .	63
6.8	Interface de impressão de tickets . . . . .	64
6.9	Interface de busca de tickets . . . . .	65
6.10	Interface de resultado da busca de tickets . . . . .	66
6.11	Interface do verificador . . . . .	67
6.12	Interface de configuração de autoridade confiável . . . . .	68
6.13	Interface para leitura de tickets . . . . .	68
6.14	Interface de informações sobre o tickets . . . . .	69
A.1	Diagrama de visão geral de interação do AAFW . . . . .	79
A.2	Diagrama de casos de uso do AAFW . . . . .	80
A.3	Diagrama de classes simplificado do AAFW . . . . .	81
A.4	Diagrama de atividades do AAFW: Tratar Requisição . . . . .	82
A.5	Diagrama de atividades do AAFW: Emitir Certificado de Atributo . . . . .	83
A.6	Diagrama de atividades do AAFW: Buscar Certificado de Atributo . . . . .	84
A.7	Diagrama de atividades do AAFW: Revogar Certificado de Atributo . . . . .	84
A.8	Diagrama de atividades do AAFW: Escutar Por Requisições . . . . .	85
A.9	Diagrama de atividades do AAFW: Parar de escutar . . . . .	85
A.10	Diagrama de sequência do AAFW: Tratar Requisição . . . . .	86
A.11	Diagrama de sequência do AAFW: Emitir Certificado de Atributo . . . . .	87
A.12	Diagrama de sequência do AAFW: Buscar Certificado de Atributo . . . . .	88
A.13	Diagrama de sequência do AAFW: Revogar Certificado de Atributo . . . . .	88
A.14	Diagrama de sequência do AAFW: Escutar Por Requisições . . . . .	89
A.15	Diagrama de sequência do AAFW: Parar de escutar . . . . .	89

# Lista de Tabelas

5.1	Implementações das interfaces do framework . . . . .	48
-----	--	----

# Lista de Siglas

<b>CA</b>	Certificado de Atributo
<b>EEA</b>	Entidade Emissora de Certificado de Atributo
<b>AA</b>	Autoridade de Atributo
<b>TCP</b>	Transmission Control Protocol
<b>HTTP</b>	Hypertext Transfer Protocol
<b>DER</b>	Distinguished Encoding Rules
<b>ASN.1</b>	Abstract Syntax Notation One
<b>AC</b>	Autoridade Certificadora
<b>AC-Raiz</b>	Autoridade Certificadora Raiz
<b>AR</b>	Autoridade de Registro
<b>CMS</b>	Cryptographic Message Syntax
<b>ICP</b>	Infra-estrutura de Chaves Públicas
<b>ICP-Brasil</b>	Infraestrutura de Chaves Públicas Brasileira
<b>LCR</b>	Lista de Certificados Revogados
<b>LACR</b>	Lista de Certificados de Atributo Revogados
<b>RFC</b>	Request For Comment
<b>SHA</b>	Secure Hash Algorithm
<b>X.509</b>	Padrão de gerência de certificados digitais
<b>OCSP</b>	Online Certificate Status Protocol
<b>SSL</b>	Secure Sockets Layer
<b>ACL</b>	Access Control List
<b>RBAC</b>	Role-based Access Control
<b>RPC</b>	Remote Procedure Call
<b>IDL</b>	Interface Definition Language



# Sumário

<b>Resumo</b>	<b>iv</b>
<b>Lista de Figuras</b>	<b>v</b>
<b>Lista de Tabelas</b>	<b>vii</b>
<b>Lista de Siglas</b>	<b>viii</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Contextualização . . . . .	1
1.2 Objetivos . . . . .	3
1.2.1 Geral . . . . .	3
1.2.2 Específicos . . . . .	4
1.3 Justificativa . . . . .	4
1.4 Metodologia . . . . .	5
1.5 Limitações do Trabalho . . . . .	5
1.6 Organização do Trabalho . . . . .	5
<b>2 Conceitos Básicos</b>	<b>7</b>
2.1 Resumo Criptográfico . . . . .	7
2.2 Criptografia Assimétrica . . . . .	8
2.3 Certificado Digital . . . . .	10
2.4 Assinatura Digital . . . . .	10
2.5 Infraestrutura de Chaves Públicas . . . . .	12

	x
2.5.1	Autoridade Certificadora . . . . . 13
2.5.2	Autoridade de Registro . . . . . 13
2.5.3	Lista de Certificados Revogados . . . . . 14
2.6	ICP-Brasil . . . . . 14
2.7	Framework Orientado a Objeto . . . . . 15
<b>3</b>	<b>Certificação de Atributos</b> <b>18</b>
3.1	Certificado Digital x Certificado de Atributo . . . . . 18
3.2	Estrutura de um Certificado de Atributo . . . . . 19
3.3	Entidade Emissora de Certificado de Atributo . . . . . 22
3.4	Uso dos Certificados de Atributos . . . . . 23
3.5	Aplicação verificadora . . . . . 24
3.6	Ciclo de vida de um Certificado de Atributo . . . . . 25
3.7	Certificados de Atributos na ICP-Brasil . . . . . 26
<b>4</b>	<b>Protocolo para Gerenciamento de Certificados de Atributo</b> <b>28</b>
4.1	Premissas do Protocolo . . . . . 29
4.2	Requisição . . . . . 30
4.2.1	AttributeCertificateReq . . . . . 31
4.2.2	AttributeCertificateIssueInfo . . . . . 32
4.2.3	AttributeCertificateRevInfo . . . . . 33
4.2.4	AttributeCertificateSearchInfo . . . . . 35
4.3	Resposta . . . . . 36
4.3.1	AttributeCertificateResp . . . . . 36
4.4	Autenticação e Controle de Acesso . . . . . 38
4.5	Transporte . . . . . 39
4.5.1	Baseado em Socket TCP . . . . . 39
4.5.2	HTTP . . . . . 39
4.5.3	Chamada de Procedimento Remoto . . . . . 40

<b>5</b>	<b>Framework AAFW</b>	<b>41</b>
5.1	Definindo o Comportamento da EEA . . . . .	41
5.1.1	Definindo o tratamento dos atributos . . . . .	42
5.2	Utilização do Framework . . . . .	45
5.2.1	Criando uma EEA Básica . . . . .	48
5.2.2	Implementando a SystemFactory . . . . .	49
5.2.3	Criando Validador de Atributo . . . . .	51
5.2.4	Criando Template . . . . .	52
5.3	Tecnologias Envolvidas . . . . .	53
5.3.1	OpenSSL . . . . .	54
5.3.2	Cryptobase . . . . .	54
5.3.3	POCO C++ Libraries . . . . .	54
5.3.4	Apache Thrift . . . . .	55
<b>6</b>	<b>Aplicações Protótipo</b>	<b>56</b>
6.1	Aplicação para controle de ingressos . . . . .	56
6.1.1	EEA para Acesso aos Cinemas . . . . .	58
6.1.1.1	Modelo de tickets . . . . .	58
6.1.1.2	Cadastramento de estudantes . . . . .	59
6.1.1.3	Tickets Emitidos . . . . .	60
6.1.2	Filial . . . . .	61
6.1.2.1	Gerenciamento de sessões . . . . .	61
6.1.2.2	Gerenciamento de tickets . . . . .	62
6.1.2.3	Emissão de tickets . . . . .	62
6.1.2.4	Busca de tickets . . . . .	64
6.1.3	Verificador de tickets . . . . .	66
6.1.3.1	Configurações . . . . .	67
6.1.3.2	Leitura de ticket . . . . .	68
6.2	Outras aplicações . . . . .	69

	xii
<b>7 Considerações Finais</b>	<b>71</b>
7.1 Contribuições . . . . .	72
7.2 Trabalhos Futuros . . . . .	72
<b>Referências</b>	<b>75</b>
<b>Apêndice</b>	<b>77</b>
<b>A Documentação UML</b>	<b>78</b>
A.1 Diagrama de visão geral de interação . . . . .	78
A.2 Diagrama de casos de uso . . . . .	79
A.3 Diagrama de classes . . . . .	80
A.4 Diagrama de atividades . . . . .	81
A.4.1 Tratar Requisição . . . . .	81
A.4.2 Emitir Certificado de Atributo . . . . .	82
A.4.3 Buscar Certificado de Atributo . . . . .	83
A.4.4 Revogar Certificado de Atributo . . . . .	84
A.4.5 Escutar Por Requisições . . . . .	85
A.4.6 Parar de escutar . . . . .	85
A.5 Diagrama de sequência . . . . .	86
A.5.1 Tratar Requisição . . . . .	86
A.5.2 Emitir Certificado de Atributo . . . . .	86
A.5.3 Buscar Certificado de Atributo . . . . .	87
A.5.4 Revogar Certificado de Atributo . . . . .	88
A.5.5 Escutar Por Requisições . . . . .	89
A.5.6 Parar de escutar . . . . .	89
<b>B Exemplos Funcionais</b>	<b>90</b>
B.1 EEA Básica . . . . .	90
B.2 Implementando SystemFactory . . . . .	91

	xiii
B.3 Implementando Validador de Atributo . . . . .	93
B.4 Implementando Template . . . . .	95
B.5 Requisitando busca CAV via TCP . . . . .	97
B.6 Requisitando emissão via PHP com Thrift . . . . .	98
<b>C Artigo</b>	<b>101</b>

# Capítulo 1

## Introdução

### 1.1 Contextualização

O certificado digital, ou CD, é um documento que identifica uma entidade no meio eletrônico, atuando como um documento de identidade digital [COO 08]. Ele estabelece a ligação dessa pessoa ou organização com um par de chaves criptográficas, pública e privada, que passam a representá-la nas relações em meio virtual. A operação mais evidente do uso dos CD é a assinatura digital [PIN 08].

Muitas aplicações utilizam o CD também nos processos de autenticação de usuários, através da tecnologia SSL [FRE 11]. Este cenário de uso é uma alternativa que agrega segurança aos métodos tradicionais, como o uso de login e senha.

Nos cenários de uso atual, são muitas vezes adicionadas ao certificado digital informações que não só identificam o usuário, como também o qualificam. Um exemplo claro é o da qualificação profissional do titular do certificado, como o número de registro de um advogado na Ordem dos Advogados do Brasil (OAB). O principal problema dessa abordagem é que a vida útil de cada atributo pode ser diferente. Quanto mais atributos são concentrados em um mesmo certificado, maior as chances desse documento precisar ser reemitido, por conta de uma mudança em um dos atributos ou mesmo da inclusão de outros. No caso do registro junto à OAB, por exemplo, caso um advogado tenha seu registro cassado, será necessária a revogação de seu certificado

digital. Entretanto, o nome e número no Cadastro Nacional de Pessoas Físicas (CPF) do titular continuam os mesmos e seriam revogados junto com o certificado. Outra deficiência desse modelo advém do fato de que nem sempre a entidade responsável por identificar o usuário e emitir o certificado digital também ser a responsável pela emissão dos atributos que o qualificam. Como ocorre com o exemplo da OAB, que teve de constituir uma Autoridade Certificadora (emissora de certificados digitais) para que pudesse incluir os números de registros no certificado digital. Para tratar problemas como os expostos acima, surgiu o Certificado de Atributos.

Um certificado de atributos, ou CA, é um documento eletrônico assinado digitalmente que estabelece qualificações a uma entidade, denominada de titular do CA. Essas atribuições podem ser das mais variadas, como por exemplo, atribuir que o titular pertence a um grupo, exerce uma função ou possui determinado nível de acesso. Um exemplo mais prático seria o de uma universidade que possa emitir um CA para cada estudante, atestando que são alunos regulares. Outro exemplo é o de um cinema, que ao invés de utilizar ingressos convencionais para seus filmes, poderia emitir um CA, atestando que o ticket foi pago e que o usuário terá direito de acesso ao filme durante seu horário especificado.

Em 2000 foi especificado o formato de um CA pelo padrão ITU-T X.509 ou ISO/IEC 9594-8 [ITU 00] e em 2002 foi publicado um perfil de uso de certificado de atributo X.509 na Internet pela RFC 3281 [FAR 02]. Isso forneceu uma base para construir aplicações que fazem uso de CA, garantindo a interoperabilidade entre elas.

No Brasil, o Comitê Gestor da Infraestrutura de Chaves Públicas Brasileira, ou ICP-Brasil, que é regida pelo ITI - Instituto Nacional de Tecnologia da Informação - aprovou, no dia 5 de julho de 2012, a criação dos certificados de atributo no âmbito da ICP-Brasil, apresentando nos DOC ICP nº 16 [ITI b] e 16.1 [ITI a] a visão geral, perfil de uso e requisitos para geração e verificação de certificados de atributos na ICP-Brasil.

Apesar de já estarem definidos um formato e um perfil de uso, os normativos da ICP-Brasil ou mesmo os internacionais não especificam claramente como

uma entidade pode requisitar a emissão de um certificado de atributos a uma Entidade Emissora de Certificado de Atributos (EEA) e nem os detalhes da comunicação entre ambas.

Buscando preencher essas lacunas na adoção da tecnologia, este trabalho apresenta uma proposta de protocolo que suporte os processos de emissão, busca e revogação de certificados de atributos, como também sugere estruturas que comportarão essas trocas de mensagens.

O trabalho propõe também um framework baseado no protocolo proposto, para desenvolvimento de aplicações que necessitem gerenciar o ciclo de vida desses certificados. O framework poderá ser utilizado por desenvolvedores de aplicações que possam se beneficiar com a tecnologia de certificação de atributos, como EEAs. São diversas as candidatas à EEA, como por exemplo:

- Universidades que desejam emitir atestados de matrícula, diplomas entre outros documentos que fornecem atribuições.
- Empresas de eventos que desejam emitir ingressos para seus eventos.
- Classes trabalhadoras que emitam certificados para seus profissionais como OAB e CRM.

O framework também serve também como forma de avaliar o protocolo proposto neste trabalho.

## **1.2 Objetivos**

### **1.2.1 Geral**

Como objetivo geral do trabalho, almeja-se desenvolver um framework orientado a objetos que permita o desenvolvimento de aplicações gerenciadoras de certificados de atributos X.509, aderente aos padrões da ICP-Brasil.



## 1.2.2 Específicos

- Definir um protocolo que suporte a gestão do ciclo de vida de certificados de atributos;
- Projetar e implementar um framework orientado a objetos que ajudará o desenvolvedor a requisitar, aprovar, emitir, revogar, distribuir, verificar e obter certificados de atributo X.509;
- Desenvolver um protótipo de aplicação para Entidade Emissora de Certificado de Atributo;
- Desenvolver um protótipo de aplicação (terceira parte) capaz de interagir com a EEA, solicitando certificados de atributos;
- Desenvolver um protótipo de aplicação capaz de verificar a validade de certificados de atributos, autorizando ou negando acesso a determinados recursos de acordo com o status de validação.

## 1.3 Justificativa

A padronização do certificado de atributo X.509 é um grande passo para a interoperabilidade entre sistemas, uma vez que se trata de um documento estruturado, passível de interpretação por máquina e com o diferencial da garantia de autenticidade, integridade e autoria. Tarefas árduas, como por exemplo a implementação de um Controle de Acesso Baseado em Papéis (RBAC) [ZHO ] podem ser implementadas utilizando CA com o atributo do tipo papel (Role), definido na RFC 5755 [FAR 10], e ser reaproveitado em diversos sistemas. Sua utilização traz grande avanço para automatização segura de processos que requerem autorização e controle. O recente interesse na tecnologia, por entidades de classe, juntas comerciais, órgãos públicos e instituições privadas em geral, com sua regulamentação no Brasil, denotam a relevância do presente trabalho para a sociedade brasileira.

## **1.4 Metodologia**

Inicialmente, foram revisado os principais normativos técnicos que envolvem o uso de certificados de atributo X.509 (CA X.509), bem como as regulamentações nacionais, da ICP-Brasil. Com base nessa revisão bibliográfica, foram levantadas as premissas a serem seguidas na definição dos formatos e mensagens que definem o protocolo de gestão de certificados de atributo.

Com base no protocolo definido, foi projetado um framework orientado a objetos que suporta todo o ciclo de vida de um certificado de atributos. Definido o modelo do framework, este foi implementado e tornou-se base para as aplicações exemplo, que fazem uso de suas funcionalidades. Por fim, foram analisados os resultados obtidos na utilização das aplicações, com relação à aplicabilidade do framework.

## **1.5 Limitações do Trabalho**

A falta de aplicações que utilizam certificado de atributo não permite garantir que o CA gerado pelo framework seja interoperável, apesar de ser implementado conforme formato definido na RFC 5755 [FAR 10]. Também não é possível ainda avaliar se o protocolo proposto atende à todas as necessidades do mercado. Além disso, o protocolo possui limitações de privilégios na requisição de certificado de atributo: A mesma pessoa que pode solicitar a busca de certificado de atributo, pode também revogá-lo.

## **1.6 Organização do Trabalho**

No capítulo 2 serão apresentados os conceitos que fundamentam o funcionamento da tecnologia de certificação digital e de atributos. O capítulo 3 apresenta em detalhes o formato e perfil de uso do certificado de atributo, com uma revisão dos documentos RFC 5755, DOC-ICP 16 e DOC-ICP 16.01. O capítulo 4 mostra o formato proposto para requisição e resposta às solicitações de CA, além de formas de

comunicação entre EEA e cliente requisitante. O capítulo 5 apresentará uma visão do framework desenvolvido e instruções gerais de como utilizá-lo. No capítulo 6, serão detalhadas as aplicações desenvolvidas com base no framework. Por fim, no capítulo 7, serão tecidas algumas considerações finais, bem como relacionadas as propostas para trabalhos futuros.

# Capítulo 2

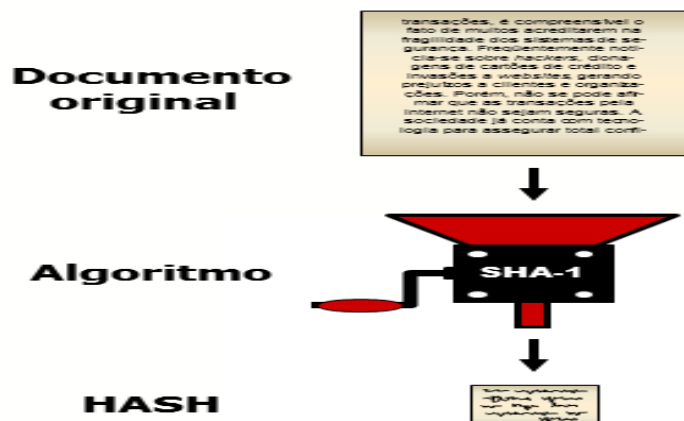
## Conceitos Básicos

A fim de compreender melhor sobre o framework construído nesse trabalho, é necessário ao leitor entendimento de algumas das tecnologias que ele se baseia. Será explanado nesse capítulo conceitos sobre certificação digital e também sobre framework orientado a objetos. O capítulo seguinte será dedicado a certificação de atributos.

### 2.1 Resumo Criptográfico

A função de resumo criptográfico, também conhecida como função hash, é um procedimento matemático que visa transformar um dado de tamanho arbitrário em um outro, de tamanho reduzido e fixo. Uma característica importante de uma função de hash é que, a partir de uma mesma informação, será produzido sempre o mesmo resultado. Daí o nome de resumo criptográfico, pois o valor resumido identifica a informação de entrada da função. Outra característica fundamental para uma boa função de hash é que, a mudança de um único bit em sua entrada, produza uma saída completamente distinta. Dadas essas propriedades, a função hash é frequentemente utilizada na garantia de integridade sobre os dados, onde envia-se a um destinatário a informação, acompanhada de seu resumo. A parte receptora processa a informação recebida utilizando a mesma função de hash e compara o resultado com o resumo

recebido. Alguns algoritmos de hash utilizados são SHA-1, SHA-256 e SHA-512.



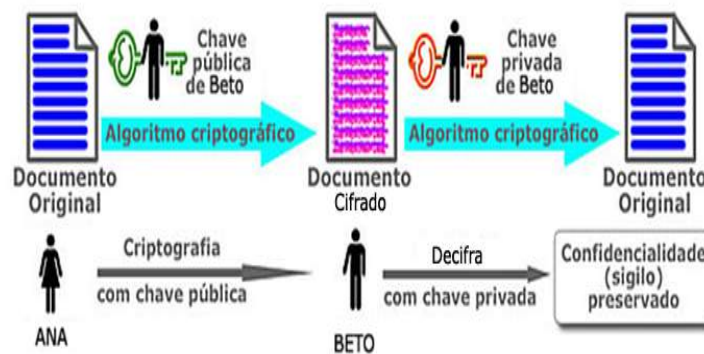
**Figura 2.1:** Funcionamento do algoritmo de hash.

## 2.2 Criptografia Assimétrica

A criptografia assimétrica consiste em um processo de transformação de textos claros em textos aparentemente sem sentido. É uma técnica muito utilizada para conferir sigilo e autenticidade a informações transmitidas em meio eletrônico. Esse processamento baseia-se no conhecimento de um segredo, conhecido como chave criptográfica. Somente as partes que conhecem essa chave são capazes de reverter o processo chamado de cifragem [CHO 02]. A criptografia assimétrica é aquela em que há não uma, mais sim duas chaves envolvidas na proteção da informação, o par de chaves. Uma dessas chaves é pública e distribuída livremente. A segunda deve ser mantida em sigilo (chave privada). O texto que uma cifra, a outra (e somente a outra) conseguirá decifrar: esse é o princípio fundamental da criptografia assimétrica. [CHO 02]

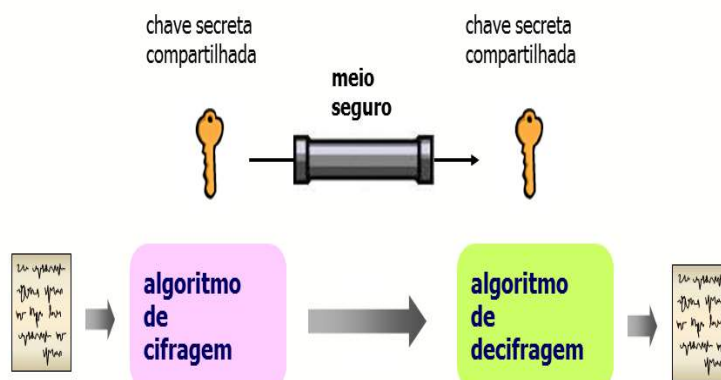
Com isso, é possível haver troca de mensagens confidenciais entre duas partes, desde que ambas conheçam a chave pública uma da outra. Antes de enviar, deve-se cifrar a informação com a chave pública da outra parte. A parte recebedora deve decifrar a informação utilizando a chave privada correspondente. Tem-se assim a

certeza que somente a entidade detentora da chave privada correspondente conseguirá ler a mensagem.



**Figura 2.2:** Confidencialidade com criptografia assimétrica.

Cifrar mensagens ou documentos grandes utilizando criptografia assimétrica é bastante custoso em termos de esforço computacional. Uma alternativa para esse problema é estabelecer uma chave única, utilizando o processo descrito anteriormente, e a partir daí realizar a comunicação criptografada com esta chave. Tem-se assim um acordo de chaves baseado em criptografia assimétrica e a comunicação baseada na criptografia chamada simétrica (uma única chave).



**Figura 2.3:** Criptografia simétrica.

## 2.3 Certificado Digital

O certificado digital é um documento eletrônico que identifica uma entidade, pois ele fornece a ligação entre ela e seu par de chaves criptográficas (pública e privada). Através dessa ligação é possível a duas entidades que nunca foram apresentadas pessoalmente, identificar-se em meio eletrônico com a segurança de que são realmente quem dizem ser. Um certificado digital, para que possa ser reconhecido como válido, deve ter sido emitido por um terceiro confiável a essas duas entidades, as chamadas Autoridades Certificadoras 2.5.1.

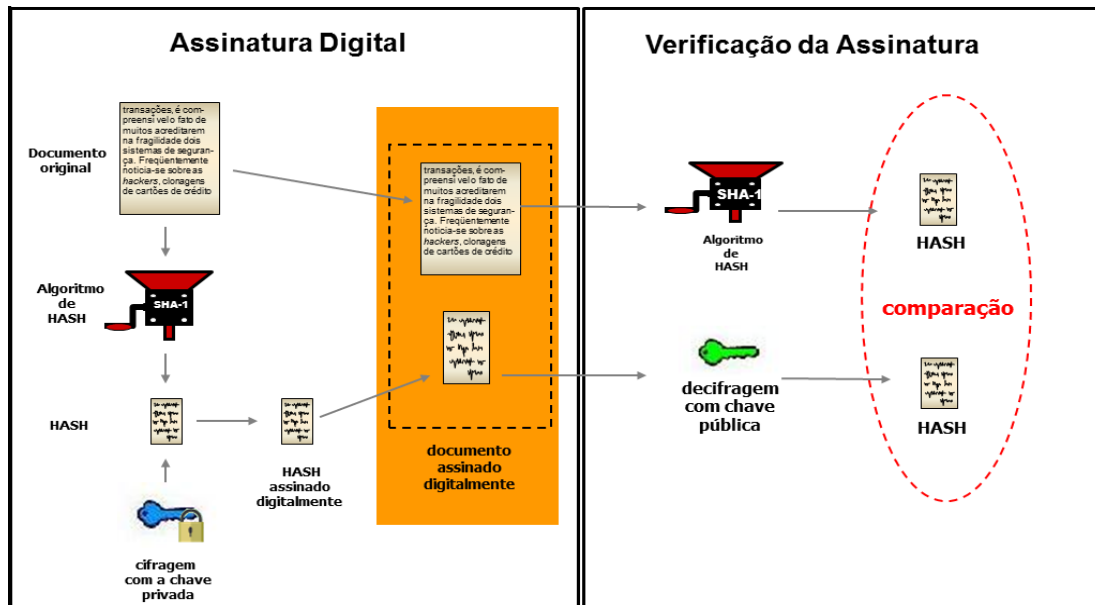
São campos comuns em um certificado digital:

- Número serial: Número inteiro único, fornecido pelo emissor;
- Emissor: A entidade que emitiu o CD;
- Validade: O período de validade do certificado;
- Titular: A entidade a quem se destina esse CD que está associada com o campo Chave Pública.
- Chave Pública: A chave pública do Titular, correspondente a sua chave privada
- Extensões: Pode conter informações adicionais, não previstas na estrutura do CD, como por exemplo o número de registro OAB do profissional.

## 2.4 Assinatura Digital

A assinatura digital é uma das principais aplicações do uso de criptografia assimétrica em conjunto com a certificação digital. Ela consiste no processo de cifragem de um resumo criptográfico que identifica um determinado documento ou mensagem, utilizando a chave privada da entidade assinante. A parte recebedora da informação poderá verificar se a assinatura (resumo assinado), foi produzida pelo detentor da chave privada, revertendo o processo com a chave pública do mesmo. Calcu-

lando um novo resumo sobre o documento recebido e comparando-o com o dado decifrado, o receptor poderá verificar a autoria e autenticidade da informação. [CHO 02]



**Figura 2.4:** Processo de assinatura digital e verificação.

Dessa forma, a assinatura digital agregada ao uso de certificado digital traz as seguintes garantias de segurança:

- **Integridade:** que a informação está completa, ou seja, não se encontra corrompida;
- **Autenticidade:** que a informação provém da fonte esperada e não foi adulterada;
- **Autoria:** que o criador da informação, ou parte a quem ela interessa, pode ser identificado inequivocamente;
- **Tempestividade:** em que exato momento foi realizada a assinatura;
- **Não Repúdio:** não é possível para o autor da assinatura negar esse ato.

A utilização do certificado digital possibilita atestar a autoria da informação e, quando utilizado um certificado emitido por uma terceira parte confiável, obtém-se



o não-repúdio, ou seja, não há como um titular de um certificado digital argumentar não ter sido ele o responsável por produzir aquela informação assinada, uma vez que somente ele conhece sua chave privada.

A propriedade de tempestividade pode ser alcançada com o uso de carimbos do tempo, que associados à assinatura, determinam uma data confiável em que elas foram realizadas.

## 2.5 Infraestrutura de Chaves Públicas

O elemento básico de uma infraestrutura de chaves públicas (ICP), é o certificado digital. Uma ICP é formada, por recursos humanos, físicos e procedimentais necessários para o gerenciamento de certificados digitais.

Sua função, de acordo com [HOU 01], é a de facilitar o uso de criptografia assimétrica através da emissão de certificados digitais e lista de certificados revogados. O modelo mais empregado de ICP consiste no estabelecimento de uma cadeia hierárquica de confiança. No topo dessa estrutura está a âncora de confiança, a Autoridade Certificadora Raiz, ou AC Raiz, que emite certificados digitais para ACs intermediárias e as chamadas ACs finais, responsáveis por emitir certificados para os entidades finais.

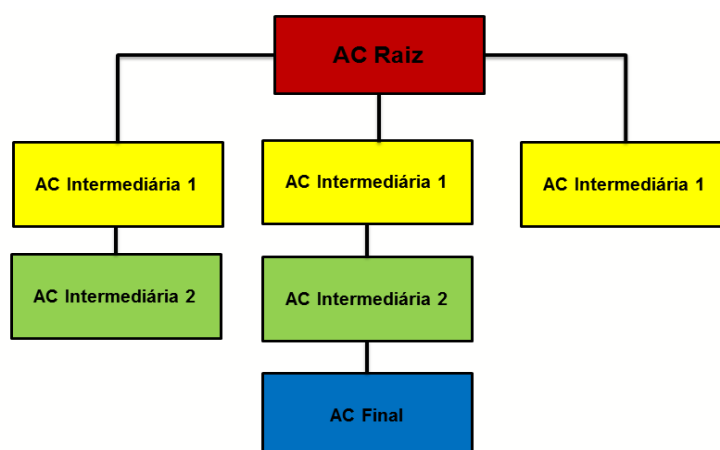


Figura 2.5: Hierarquia de uma ICP.

### 2.5.1 Autoridade Certificadora

Uma autoridade certificadora (AC) também é formada por recursos humanos, físicos e procedimentais. A sua principal tarefa é a de emitir certificados digitais. Ela faz isso preenchendo os campos do certificado digital 2.3 a ser emitido e depois os assina com sua chave privada. A autoridade contudo, só pode realizar a emissão do certificado se as informações a serem preenchidas estiverem corretas. Para isso ela conta com a Autoridade de Registro 2.5.2.

A confiança das informações contidas no certificado está na verificação dessa assinatura com a chave pública da autoridade certificadora. Portanto a proteção de sua chave privada é o elemento essencial para a confiança da AC.

Outra tarefa das autoridades certificadoras é a de revogar certificados, caso deseje-se cancelar seu uso antes do seu período de validade acabar. Essas informações precisam ser publicadas para que sistemas que utilizem CD verifiquem se este continua válido para uso. A estrutura que contém essas informações é denominada Lista de Certificados Revogados (LCR).

**Autoridade Certificadora Raiz** Esta no topo da hierárquica de uma ICP. Segue políticas e normas rígidas operacionais e de segurança. Emite LCRs e certificado para autoridades certificadoras intermediárias bem como fiscaliza os componentes da ICP regularmente.

**Autoridade Certificadora Intermediária** Está subordinada a AC raiz. Emite LCRs e certificados para outras ACs como também para usuários finais.

**Autoridade Certificadora Final** Pode emitir certificado somente para usuários finais. A emissão desses certificado se dá através de comunicação com a autoridade de registro 2.5.2.

### 2.5.2 Autoridade de Registro

É a entidade responsável pela interação entre a autoridade certificadora e os usuários finais. A autoridade de registro (AR) tem o papel de confirmar os dados do

usuário para constar no seu certificado a ser emitido. Um dos motivos para existência da AR é de que alguns dados do usuário são bastante específicos, como o número de registro do profissional (OAB, CRM) e requer a comunicação com entidades externas para essa validação. Esse tipo de tarefa não compete a AC. Outro motivo é que a AC geralmente atende a diversas requisições de emissão de certificado, já estando bastante ocupada.

### **2.5.3 Lista de Certificados Revogados**

A lista de certificados revogados é um documento assinado emitido por autoridades certificadoras que contém informações sobre os certificados que ela revogou. Essas informações incluem o número serial do certificado revogado, a data de revogação e um motivo. A revogação pode acontecer por roubo da chave privada do titular, por desistência por parte do titular de fazer seu uso ou pela perda de algum privilégio contido no seu certificado digital como seu número OAB, no caso de advogados.

Toda aplicação que faz uso de certificação digital deve sempre obter a LCR mais atual para verificação do estado do certificado digital em questão. Também deve ser feito essa verificação para todos os certificados de sua cadeia.

## **2.6 ICP-Brasil**

Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil) é uma ICP instituída pelo governo brasileiro pela Medida Provisória 2200 [BRA ].

*Art. 1o Fica instituída a Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil, para garantir a autenticidade, a integridade e a validade jurídica de documentos em forma eletrônica, das aplicações de suporte e das aplicações habilitadas que utilizem certificados digitais, bem como a realização de transações eletrônicas seguras.*

## 2.7 Framework Orientado a Objeto

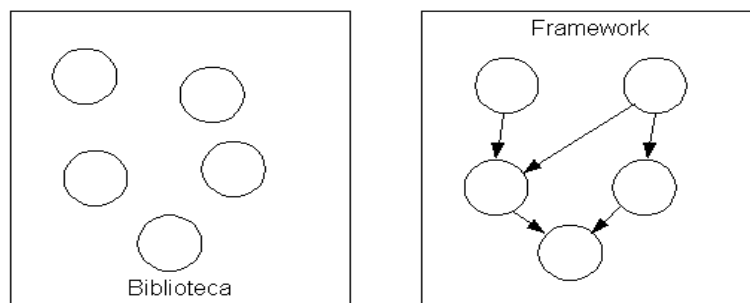
“Um framework consiste em um conjunto de classes, interfaces e padrões dedicados a solucionar um grupo de problemas através de uma arquitetura de programação flexível e extensível.” [Govoni 1999]

A demanda da indústria de software de produzir cada vez mais softwares complexos com menor tempo de desenvolvimento, é conhecida como paradoxo da indústria de software (Silva).

Uma das formas de alcançar esse objetivo é com o reuso de código, que pode ser feito com a implementação de artefatos de softwares reutilizáveis, como uma biblioteca de classes para manipulação de arquivos.

Um framework orientado a objetos consiste em uma outra forma de reuso de software, classificado como reuso de projeto. Ele é projetado para generalizar um domínio específico de aplicações, de maneira que novas possam ser construídas sobre o framework, com elevado grau de reuso. Isso é possível pois ele é composto por um conjunto de classes, sendo que algumas estão com sua implementação inacabada, bastando apenas a aplicação consumidora fornecer as partes faltantes, que a distingam das demais. Essas partes faltantes são chamadas de *hotspots* e representam os pontos de flexibilização do framework.

A figura a seguir mostra a diferença entre uma biblioteca e um framework: No primeiro, cabe as aplicações que a utilizam realizar a colaboração entre as classes. No segundo, a forma com que as classes colaboram entre si já está definida.

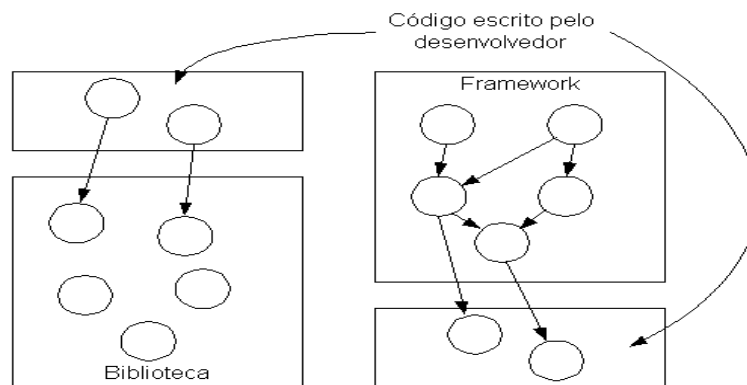


**Figura 2.6:** Biblioteca vs Framework (extraído de [Jac ])

O framework orientado a objetos confina as aplicações consumidoras à sua arquitetura de classes e fluxo de execução de métodos. Apesar disso, é possível acrescentar ou alterar suas funcionalidades, uma vez que o framework é construído sobre uma série de padrões de projetos voltados justamente ao reuso e especialização de suas funcionalidades.

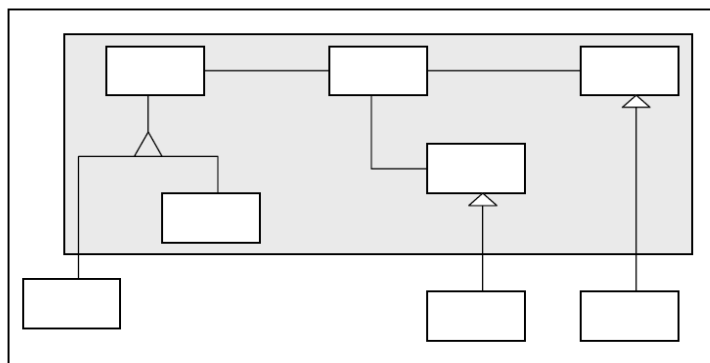
Uma de suas características principais é a inversão de controle, em que o framework sabe quando deve chamar os métodos inacabados que são implementados pela aplicação. Logo o programador da aplicação não tem controle do fluxo de execução, mas sim sobre a forma como cada atividade nele prevista será executada. Essa característica também é conhecida como princípio de Hollywood "*Don't call us, we'll call you*".

A figura a seguir mostra o framework realizando a inversão de controle, ou seja 'chamando' a parte de código desenvolvida fora dele, o que não ocorre com uma biblioteca.



**Figura 2.7:** Uso de Biblioteca vs Framework (extraído de [Jac ])

Evidentemente que para utilização de um framework o usuário deve ter conhecimento de seu projeto assim como sua estrutura de classes. Deve saber como adaptar e estender suas funcionalidades. Isso geralmente acarreta num tempo de aprendizagem mais demorado, porém o esforço é recompensado devido ao alto grau de reuso e conseqüentemente, produtividade.



**Figura 2.8:** Aplicação desenvolvida utilizando framework (Silva, 2000)

# Capítulo 3

## Certificação de Atributos

No capítulo anterior foram apresentados alguns conceitos fundamentais de criptografia e certificação digital, bem como os princípios de reusabilidade e engenharia de software utilizados na construção de frameworks orientados a objetos. Neste capítulo, apresentaremos o segundo tipo de certificado definido pelo padrão X.509, o de atributos, que complementa o certificado digital de chaves públicas, trazendo qualificações ao seu titular.

### 3.1 Certificado Digital x Certificado de Atributo

O documento RFC 5755 [FAR 10] propõe o perfil de certificado de atributos a ser utilizado pelos padrões da internet. Pode-se entender a diferença entre certificados digitais e certificados de atributos através de uma analogia entre passaporte e visto de permanência.

O primeiro pode ser visto como um passaporte. Ele identifica o titular, tem uma validade que tende a ser longa, é emitido por uma autoridade confiável e possui um número que identifica o titular perante essa autoridade, no caso o Departamento de Polícia Federal.

Já o certificado de atributos assemelha-se a um visto de permanência, consistindo em um benefício temporário: a permissão de permanência em um país por um

período máximo de tempo. Este benefício é emitido por uma autoridade diferente e tem uma validade mais curta.

Perceba então que o CD identifica enquanto que o CA confere qualificações, como por exemplo se o usuário poderá trabalhar, apenas estudar ou exercer outras atividades durante sua validade. Pode-se concluir então que um certificado digital possui atributos estáticos, de longa duração, sempre relacionados à identidade do titular. Para sua emissão, o titular deve se submeter a um criterioso processo de identificação, que muitas vezes exige uma validação presencial da identidade do indivíduo. O certificado de atributos, por sua vez, agrega ao primeiro propriedades dinâmicas, que podem ter menor duração e podem ser substituídas, removidas ou incluídas sem a rigidez com que ocorre uma emissão de CD.

## 3.2 Estrutura de um Certificado de Atributo

A definição ASN.1 do CA é apresentada abaixo:

```
AttributeCertificate ::= SEQUENCE {
    acinfo          AttributeCertificateInfo,
    signatureAlgorithm AlgorithmIdentifier,
    signatureValue   BIT STRING
}
```

O campo **acinfo** contém as informações sobre o certificado de atributo. Essas informações serão assinadas, o campo **signatureValue** conterá o valor da assinatura e **signatureAlgorithm** o identificador do algoritmo utilizado.

```
AttributeCertificateInfo ::= SEQUENCE {
    version          AttCertVersion, -- version is v2
    holder           Holder,
    issuer           AttCertIssuer,
    signature        AlgorithmIdentifier,
```



```

serialNumber          CertificateSerialNumber,
attrCertValidityPeriod AttCertValidityPeriod,
attributes            SEQUENCE OF Attribute,
issuerUniqueID        UniqueIdentifier OPTIONAL,
extensions            Extensions OPTIONAL
}

```

Essa estrutura representa as informações contidas no certificado de atributo. Os campos mais importantes são **holder** que identifica o titular do CA, seja ele uma pessoa física, jurídica ou mesmo um equipamento. Também está contido nessa estrutura no campo **issuer**, informações que permitem identificar quem emitiu o CA. O campo **attributes** contém os atributos associados ao titular do CA, ou seja, os benefícios que se deseja lhe conceder. Outro campo importante é **attrCertValidityPeriod** que delimita a janela de validade do CA.

A identificação do titular do certificado de atributos pode ser feita de três formas. Mais de uma pode ser especificada para o mesmo CA, porém é recomendado utilizar somente uma forma de identificação por CA. A forma *baseCertificateID* é utilizada para vincular o titular com seu certificado digital, através de seu número serial e emissor, por isso é recomendada para uso em ambientes que já utilizem autenticação baseada em certificados digitais. A outra forma é *entityName* que é utilizada para vincular o CA a um nome que identifique a entidade unicamente. É útil em casos que a entidade não possui certificado digital. A última forma é *objectDigestInfo* que serve para identificar o titular através do resumo criptográfico de um objeto, por exemplo seu CD.

Os atributos devem conter um conjunto de privilégios a ser associados com o titular (holder). Ele é formado por um identificador de objeto (OID) e um valor qualquer, sua definição ASN.1 é a seguinte:

```

Attribute ::= SEQUENCE {
    type          AttributeType,
    values        SET OF AttributeValue
}

```

```

    -- at least one value is required
}

AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY DEFINED BY AttributeType

```

Alguns tipos de atributos padrão foram definidos, considerando um perfil de uso na Internet. A maioria deles é baseado no *IetfAttrSyntax* [FAR 10], uma sintaxe que permite identificar, além dos atributos, a entidade de política de atributo. São eles:

**Service Authentication Information** Identifica o titular para um serviço ou servidor e pode incluir informações de autorização como usuário/senha, neste caso o atributo deve ser cifrado.

**Access Identity** Identifica o titular para um serviço ou servidor que pode ser utilizado pelo verificador para autorizar ações no sistema.

**Charging Identity** Identifica uma entidade para fins de cobrança, geralmente diferente do titular. Por exemplo, a empresa do titular pode ser cobrada por um serviço.

**Group** Contém informações sobre grupo o qual o titular é membro.

**Role** Identifica os papéis associados ao titular. Por exemplo, administrador.

**Clearance** Carrega informações de liberação de acesso sobre titular, através de um identificador da política de acesso.

Os tipos fornecem apenas uma indicação para uso, o que não impede de serem definidos outros atributos uma vez que a estrutura *Attribute* é bastante flexível.

As extensões fornecem informações sobre o CA e algumas foram definidas para uso nesse perfil:

**Audit Identity** Em casos em que a identidade do titular não pode ser diretamente inferida pelo campo holder, essa extensão pode ser utilizada para fins de auditoria e registro.

**AC Targeting** Utilizado para direcionar o uso do CA para serviços ou servidores específicos.

**Authority Key Identifier** Utilizado para ajudar o verificador a verificar a assinatura do CA, pois a extensão contém o AuthorityKeyIdentifier do seu emissor.

**Authority Information Access** Utilizado para ajudar o verificador a checar o estado de revogação do CA, pois fornece informações sobre onde obter a Lista de Certificados de Atributo Revogado (LCAR) ou endereço do servidor OCSP do emissor.

**CRL Distribution Points** Indica o local da publicação do LCAR para checar o estado de revogação do CA.

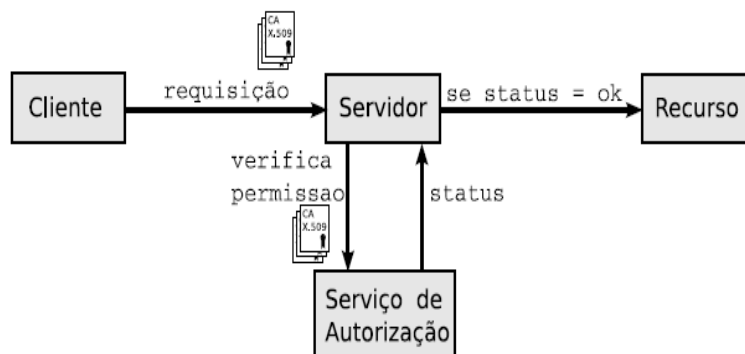
**No Revocation Available** Indica que nenhuma informação de revogação vai estar disponível para este CA.

### 3.3 Entidade Emissora de Certificado de Atributo

Uma Entidade Emissora de Atributos (EEA) é sinônimo do termo Autoridade de Atributos (AA). Segundo a RFC 5755 [FAR 10] ela é qualquer entidade que emita (assina) certificados de atributos. De acordo com a ICP-Brasil, ela pode ser qualquer pessoa jurídica detentora da prerrogativa legal para emissão de determinado atributo [ITI b]. Tal entidade deve possuir, para a assinatura dos certificados de atributos, um certificado digital ICP-Brasil emitido para Pessoas Jurídicas, do tipo A3 ou A4.

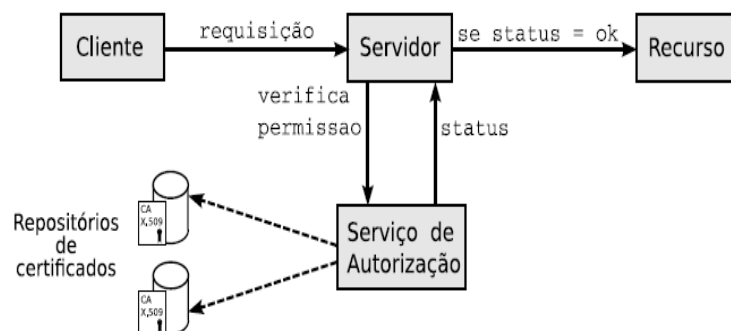
### 3.4 Uso dos Certificados de Atributos

Como técnica de distribuição de CA a RFC define dois modelos. O modelo *push*, no qual o cliente de um serviço apresenta seu certificado de atributo sempre que um terceiro precisar validá-lo antes de liberar acesso a determinado recurso.



**Figura 3.1:** Modelo *push* de distribuição de CA (Extraída de [GUI 08])

O segundo modelo é o *pull*, no qual o servidor, mediante requisição de um serviço, busca automaticamente o certificado de atributo a partir de um repositório confiável. Este último caso é especialmente interessante pois não há necessidade de alterar o protocolo de comunicação já existente entre cliente-servidor. Apesar das definições desses modelos, não é definido explicitamente como deve ser feita a troca de mensagens entre essas entidades, o que dificulta a interoperabilidade entre sistemas.



**Figura 3.2:** Modelo *pull* de distribuição de CA (Extraída de [GUI 08])

### 3.5 Aplicação verificadora

Para se considerar um certificado de atributo como válido, algumas considerações devem ser feitas. No caso do titular estar vinculado ao seu CD, ele deve ser achado pelo verificador e toda sua cadeia de certificação deve ser verificada. A assinatura do CA deve estar criptograficamente correta. A cadeia de certificação do emissor do CA também deve ser verificada. O emissor deve ser confiável pelo verificador. No momento da verificação, a hora deve estar entre o período de validade do CA.

Outra entidade envolvida no ciclo de vida do certificado de atributo são as aplicações verificadoras. São as aplicações que necessitam de um certificado de atributo válido a fim de permitir seu uso. Para garantir que um certificado de atributo seja válido, segundo a ICP-Brasil, as seguintes verificações devem ser feitas:

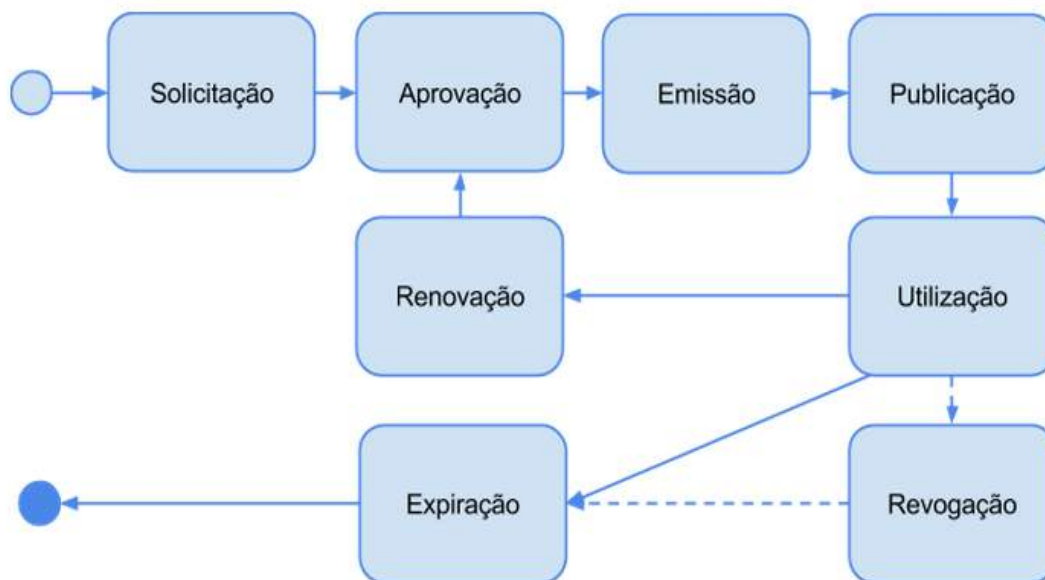
1. Validação da cadeia de certificação e do certificado de assinatura da EEA.
2. O certificado de atributo não deve constar na LCR ou consulta OCSP da EEA, quando a mesma optar por utilizar revogação.
3. O certificado de atributo não deve estar expirado.
4. A integridade da assinatura do certificado de atributo puder ser verificada com o certificado digital da EEA.
5. Obter e fazer uma comparação direta dos atributos desejados no certificado de atributo.

Nota-se também que é de responsabilidade da aplicação verificadora garantir a autenticidade do titular do certificado de atributo informado ou obtido por ela. Esse método de validação, contudo, pode expor informações ao validador que são da privacidade do titular, como seu nome e, caso esteja associado ao certificado digital, sua chave pública [OIK 06].

Um exemplo de aplicação que verifica ingressos de cinema, codificados como certificado de atributo, é implementada nesse trabalho 6.1.3.

### 3.6 Ciclo de vida de um Certificado de Atributo

O ciclo de vida do certificado de atributo começa com sua solicitação para EEA por uma entidade interessada. A solicitação é processada e será verificado a situação do titular a fim de aprovar sua emissão. Após isso, o CA será emitido. Sua publicação utilizando um dos métodos *push* ou *pull* 3.4 é necessária para que ele possa ser consumido. Uma vez publicado, o seu titular pode gozar dos benefícios por ele trazidos, até que o certificado expire. Caso ela desejar obter novamente seus privilégios, pode ser solicitada sua renovação, o que implica em novo processo de aprovação, emissão e assim sucessivamente. Ao longo de sua validade ele pode vir a ser revogado devido à perda dos referidos atributos, por exemplo, passando a constar na lista de certificados de atributos revogados. O ciclo termina quando seu prazo de validade expira.



**Figura 3.3:** Ciclo de vida do Certificado de Atributo

### 3.7 Certificados de Atributos na ICP-Brasil

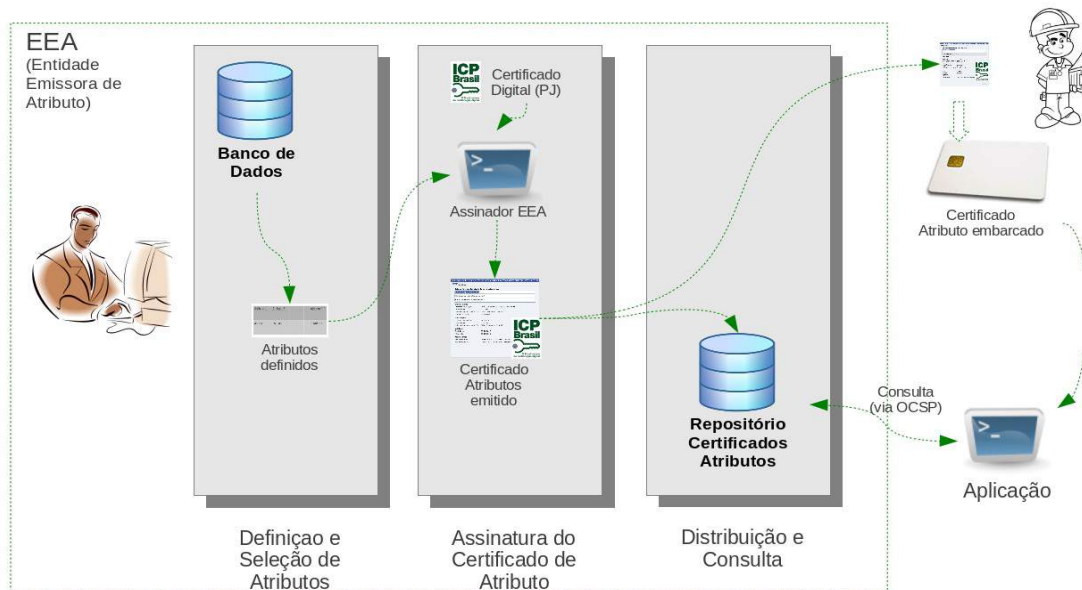
A ICP-Brasil regulamentou a adoção do certificado de atributo para uso no Brasil, especificado nos documentos DOC-ICP 16 [ITI b] e DOC-ICP 16.01 [ITI a].

Os documentos são baseados na RFC 5755 [FAR 10] e definem dois tipos de certificado de atributo:

**Certificado de Atributo Autônomo(CAA)** A vinculação deste certificado com o seu titular se dá através de um nome que o identifique unicamente, como RG ou CPF, logo não é necessário que o mesmo possua um certificado digital. É utilizada a forma de nome de entidade *entityName* para vinculação do titular. Esse tipo é bastante útil no caso de emissão de atestados/declarações/tickets.

**Certificado de Atributo Vinculado (CAV)** A vinculação deste certificado com o seu titular se dá através do seu certificado digital ICP-Brasil. É utilizado a forma *baseCertificateID* para vinculação do titular.

O ciclo de vida do certificado de atributo sugerido pela ICP-Brasil [ITI b], começa com a autoridade de atributo definindo os atributos que ela suporta e selecionando quais deles farão parte do certificado de atributo. De posse de sua chave privada e certificado digital do tipo PJ (A3 ou A4) ela é solicitada para emitir um certificado de atributo que é então salvo no seu repositório, podendo ser também embarcado num *smart card* do titular. Aplicações verificadoras que desejam obter um certificado de atributo de alguma entidade devem realizar a consulta no repositório da autoridade de atributo. Certificados de atributo emitido por entidades cujos certificados são pertencentes à cadeia da ICP-Brasil tem prerrogativa de validade jurídica.



**Figura 3.4:** Ciclo de vida do Certificado de Atributo (Extraída de [ITI b])

Outro aspecto do ciclo de vida que não consta na figura é a revogação do certificado de atributo.

Um candidato à autoridade de atributo seria o Conselho Regional de Medicina. Novos médicos poderão solicitar seu CRM, na forma de certificado de atributo, que poderá ser gravado num *smart card* ou qualquer mídia do seu titular. Médicos poderão ser cassados e seus certificados de atributos revogados e publicados.

Uma EEA foi implementada nesse trabalho 6.1.1.



## Capítulo 4

# Protocolo para Gerenciamento de Certificados de Atributo

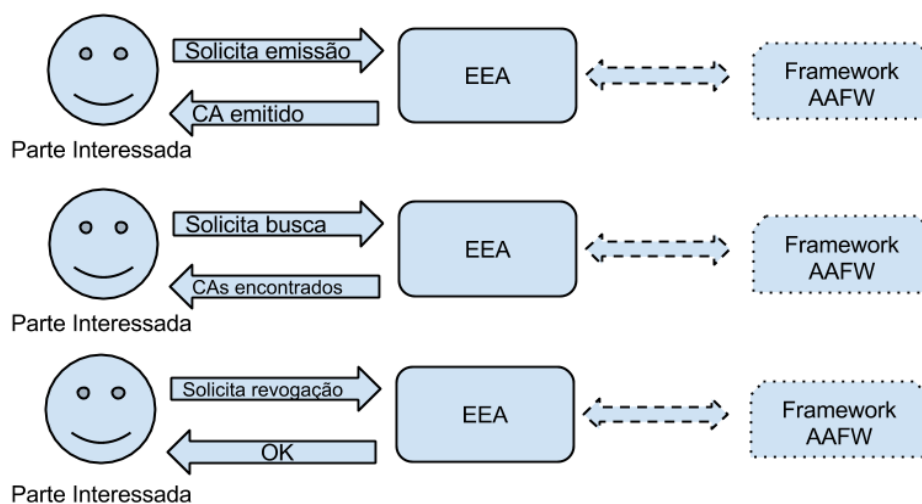
A RFC 5755 define o formato do certificado de atributo e apresentou dois modelos para sua distribuição, *push* e *pull*, porém o documento abstrai os detalhes técnicos relativos à publicação do certificado. Caso uma EEA deseje realizar a distribuição pelo modelo *pull* utilizando como repositório um diretório de rede, as aplicações interessadas em obter o certificado de atributo devem conhecer o repositório, as formas de autenticação, sua organização, entre outros. Nota-se que esta forma seria uma especificidade dessa EEA. Outras poderão implementar a distribuição de outra maneira, o que implica em uma falta de padronização dos protocolos de acesso. Outro ponto que fica a cargo dos implementadores é a forma como serão requisitados, consultados e revogados os certificados.

O protocolo proposto visa resolver essas questões, uma vez que define um conjunto de estruturas, na forma de requisição e resposta, a serem trocadas entre solicitante e EEA na emissão, revogação ou busca de certificado de atributo.

A requisição é enviada por uma parte interessada em realizar uma dessas três operações e deve ser processada e respondida pela entidade emissora de atributo.

A figura 4.1 mostra as três situações abrangidas pela protocolo. Note que o framework AAFW, que será apresentado no capítulo 5 não faz parte do protocolo,

porém age como um componente facilitador para autoridade de atributo conseguir realizar essas ações.



**Figura 4.1:** Protocolo de requisição

O documento intitulado *Attribute Certificate Request Message Format* [YEE ] descreve um formato de requisição para emissão de certificados de atributos. Embora tenha sido utilizado como referência na construção do presente protocolo, o rascunho de padrão não define um formato para resposta às requisições, ou mesmo uma forma de solicitação de revogação ou busca por certificados de atributos.

## 4.1 Premissas do Protocolo

Para o desenvolvimento do protocolo proposto, foram seguidas as seguintes premissas:

### Gerais

- permitir o suporte à gestão de mais de uma EEA pelo mesmo servidor
- suportar os métodos *push* e *pull*, conforme definido na RFC 5755

- ser aderente aos normativos da ICP-Brasil

### **Emissão de Certificados de Atributos**

- permitir a solicitação de um certificado para um atributo e titular específico
- permitir que a EEA decida quais os atributos podem ou não ser emitidos
- permitir à EEA negar a emissão de um certificado de atributos
- permitir a solicitação de um certificado de atributos com base em um modelo pré-definido
- permitir a emissão de certificados de atributos vinculados e não vinculados, conforme estabelecido pela ICP-Brasil

### **Busca por Atributos**

- buscar por um ou mais atributos de um titular
- buscar por atributos emitidos por uma EEA específica

### **Revogação de Certificados de Atributos**

- permitir revogação de um certificado específico
- permitir revogação de um ou mais atributos específicos
- permitir a revogação de todos os atributos de um titular
- permitir a revogação de atributos emitidos por uma EEA específica

## **4.2 Requisição**

O protocolo proposto inicia com o preenchimento de uma requisição, por parte do solicitante do certificado de atributos, a ser submetida à EEA. Essa requisição encontra-se definida na estrutura *AttributeCertificateReq*.

### 4.2.1 AttributeCertificateReq

Uma requisição pode conter uma solicitação de emissão 4.2.2, revogação 4.2.3 ou busca 4.2.4 de certificado de atributo, conforme mostrado na estrutura apresentada a seguir.

```
AttributeCertificateReq ::= SEQUENCE {
    version                INTEGER { v1(1) },
    reqType                ACReqType,
    reqInfo                AttributeCertificateReqInfo
}
```

A estrutura `AttributeCertificateReq` contém uma versão, um tipo e as respectivas informações que detalham-na. Os tipos de requisição são os definidos em `ACReqType`, ou seja, valores inteiros representando emissão, busca ou revogação.

```
ACReqType ::= ENUMERATED {
    issue                (1),
    -- Used in issuing requests
    revoke                (2),
    -- Used in revocation requests
    search                (3)
    -- Used in search requests
}
```

De acordo com o tipo de requisição definido, o campo `reqInfo` deve ser preenchido em conforme, com uma das estruturas previstas em `AttributeCertificateReqInfo`.

```
AttributeCertificateReqInfo ::= CHOICE {
    issueInfo            AttributeCertificateIssueReq,
    revInfo              AttributeCertificateRevReq,
    searchInfo           AttributeCertificateSearchReq
}
```

As seções 4.2.2 a 4.2.4 detalham as estruturas de cada um dos três tipos de mensagem previstos no protocolo.

## 4.2.2 AttributeCertificateIssueInfo

A estrutura `AttributeCertificateIssueInfo` comporta as informações necessárias para que a EEA emita um certificado de atributo.

```
AttributeCertificateIssueReq ::= SEQUENCE {
    templateId          INTEGER OPTIONAL,
    -- if templateId is supplied then the validityPeriod,
    -- attributes and extensions fields must not be present.
    -- Those values will be automatically provided by the AA,
    -- based on templateId.
    holder              Holder,
    issuer              AttCertIssuer OPTIONAL,
    validityPeriod      AttrCertValidtyPeriod OPTIONAL,
    attributes          SEQUENCE OF Attribute OPTIONAL,
    extensions          Extensions OPTIONAL
}
```

A informação mínima que deve ser passada a uma EEA é a identificação do titular que receberá o certificado de atributos. Essa informação deve ser fornecida no campo `holder`, que representa a estrutura `Holder` definida na RFC 5755. Essa informação pode ser tanto uma referência a um certificado digital, quanto um nome qualquer, sem um vínculo específico com um CD. Estas formas distintas são as que diferenciam um Certificado de Atributos Vinculado de um Autônomo na ICP-Brasil, conforme visto no capítulo 3.

Há basicamente duas formas de solicitar a emissão de um certificado de atributos. A primeira delas é identificar na requisição um modelo pré estabelecido pela EEA, através do campo `templateId`. Essa forma é indicada para EEAs que emitam

certificados de atributos que seguem um modelo específico, como por exemplo uma carteira de motorista digital emitida pelo órgão de trânsito. Embora haja diferenças tais como as categorias a que o motorista está habilitado, a carteira em si segue um formato específico. Em outras palavras, muda-se os valores dos atributos, mas há um conjunto fixo dessas estruturas presentes no certificado, que consistem no seu modelo comum.

A segunda forma de solicitar um certificado de atributos consiste na definição desse modelo por parte do próprio solicitante. Há cenários em que pode interessar a definição do modelo de certificado na própria solicitação, como em um sistema que utilize certificados de atributos, por exemplo, para encaminhar prestação de contas para um órgão fiscalizador. À medida que o órgão solicite informações mais detalhadas, o sistema poderá incluir atributos adicionais, emitindo certificados de forma mais flexível. Esta forma de requisição é suportada através dos campos específicos como o `attributes`, `extensions` e `validityPeriod`.

Além dos campos já citados, a estrutura permite especificar qual a EEA para que a requisição está destinada, através do campo `issuer`, representado pela estrutura padrão definida na RFC `AttCertIssuer`. Esse campo é útil para permitir, a um mesmo sistema servidor, a gestão de mais de uma EEA. Assim, se houver mais de um emissor para um mesmo tipo de atributo ou no caso anteriormente exposto de certificados flexíveis, é possível especificar qual o destinatário da requisição.

### 4.2.3 AttributeCertificateRevInfo

Essa estrutura transporta informações relacionadas a um pedido de revogação de um ou mais certificados de atributos emitidos por uma EEA.

A exemplo da emissão, há também na revogação duas formas de solicitar a invalidação de um certificado de atributos. A primeira é informando um conjunto de identificadores únicos para os certificados. A segunda busca revogar o atributo em si, não apenas um certificado. Cabe aqui uma ressalva de que não foi encontrado na literatura uma forma de revogar um atributo, outra que através da revogação do

certificado que o contém. Entretanto, a ICP-Brasil estabelece, no item 4.2.2 do DOC-ICP-16.01 [ITI a], que uma EEA deve especificar as condições e regras de revogação para atributos. Buscando conformidade com os normativos, foi prevista no protocolo uma forma de revogar todos os certificados de atributos que contém determinado atributo, para um mesmo titular. A estrutura de uma requisição de revogação, `AttributeCertificateRevReq`, é apresentada a seguir.

```
AttributeCertificateRevReq ::= CHOICE {
    attrCerts          SEQUENCE OF IssuerSerial,
    attrRevInfos       SEQUENCE OF AttributeRevocationInfo
}

AttributeRevocationInfo ::= SEQUENCE {
    holder             Holder,
    attributes         SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
    caIssuer           AttCertIssuer OPTIONAL
}
```

De acordo com a estrutura `AttributeCertificateRevReq`, um conjunto de certificados pode ser especificado através da estrutura `IssuerSerial`, conforme definido na RFC 5755. Esta forma permite a revogação de certificados específicos, permitindo limitá-la a um contexto específico.

A segunda forma baseia-se em identificar os atributos a serem revogados a partir de uma lista de identificadores, os chamados Object Identifiers (OID). Além dos identificadores, outro parâmetro permite uma maior granularidade no processo de revogação. Com o parâmetro `caIssuer`, é possível restringir a revogação aos certificados de atributos emitidos por uma única EEA. É importante perceber que o único campo obrigatório para a revogação de atributos é a identificação do próprio titular, tornando a solicitação flexível ao ponto de remover todos os certificados de atributo emitidos para uma entidade, ou apenas revogar um dos certificados emitidos.

#### 4.2.4 AttributeCertificateSearchInfo

A estrutura `AttributeCertificateSearchInfo` concentra informações utilizadas na busca por certificados de atributo no repositório da EEA. A busca pode ser por todos os certificados de atributos emitidos para um mesmo titular, caso em que somente o campo `holder` deve se preenchido. Um solicitante pode também buscar por um conjunto específico de atributos a serem procurados, utilizando para tanto uma ou mais estruturas de busca `AttributeCertificateSearchFilter`. A definição das estruturas em ASN.1, a seguir, apresenta o formato de uma busca.

```
AttributeCertificateSearchReq ::= SEQUENCE {
    holder          Holder,
    issuer          AttCertIssuer OPTIONAL,
    attributesFilter SEQUENCE OF AttributeCertificateSearchFilter
    OPTIONAL
}
```

```
AttributeCertificateSearchFilter ::= SEQUENCE {
    attributeOid    OBJECT IDENTIFIER,
    values          SET OF AttributeValue OPTIONAL
}
```

```
AttributeValue ::= ANY DEFINED BY OBJECT IDENTIFIER
```

De acordo com a estrutura `AttributeCertificateSearchFilter`, nota-se que a busca pode ser realizada sobre um conjunto de atributos quaisquer, representados pelos seus identificadores ou ainda por um subconjunto destes, em que os valores condigam com os parâmetros informados em `values`.



## 4.3 Resposta

Toda requisição `AttributeCertificateReq` recebida pela autoridade de atributo deverá gerar uma resposta `AttributeCertificateResp`, que conterá o resultado da operação. Se a operação tiver sido executada com sucesso, conterá também um ou mais certificados de atributos. Sua estrutura é detalhada a seguir, em 4.3.1.

### 4.3.1 `AttributeCertificateResp`

A resposta de um requisição pode ter sido processada com sucesso, isto é, a operação que foi solicitada foi efetuada como o esperado pelo requerente, bem como pode ter falhado. A estrutura a seguir representa um `AttributeCertificateResp`.

```
AttributeCertificateResp ::= SEQUENCE {
    status                ACStatusInfo,
    attributeCertificate  SEQUENCE OF AttributeCertificate OPTIONAL
    -- when the response is the result of a granted issue request,
    -- attributeCertificate must contain only ONE AttributeCertificate.
}
```

```
ACStatusInfo ::= SEQUENCE {
    status            ACStatus,
    statusString     PKIFreeText OPTIONAL,
    failInfo         ACFailureInfo OPTIONAL
}
```

```
ACStatus ::= INTEGER {
    granted            (0),
    rejection          (1)
}
```

```

ACFailureInfo ::= INTEGER {
    badRequest          (2),
        -- transaction not permitted or supported
    badDataFormat      (5),
        -- the data submitted has the wrong format
    notApproved        (15),
        -- the AA declined to grant the holder's attributes
    unacceptedExtension (16),
        -- the requested extension is not supported by the AA.
    untrustedHolder    (18),
        -- the holder is not trusted by the AA.
    unsupportedAttribute (19),
        -- one of the attributes requested is not supported by the AA.
    unsupportedTemplate (20),
        -- the templateId supplied in AttributeCertificateInfoReq is
        -- not known or supported by the AA.
    unknownSerial      (21),
        -- the serial supplied for revocation in AttributeCertificateReq
        -- is not known.
    systemFailure      (25)
        -- the request cannot be handled due to system failure
}

```

No caso do status da resposta ser `granted`, a requisição terá sido devidamente processada. Se a requisição foi pela emissão de um CA, ele estará presente na resposta. Caso tenha sido requisitada a busca por CAs, eles também estarão presentes na resposta, se encontrados. Caso tenha sido requisitada uma revogação de CAs, ele será incluído na próxima LCAR e também estarão presentes na resposta.

A estrutura `ACStatusInfo` apresenta os possíveis resultados para uma requisição, que pode também ser rejeitada com o status `rejected`. O protocolo permite também a descrição textual da causa do resultado, através do `statusString` e o detalhamento dos motivos, especificando-se o `failInfo`. Abaixo são detalhados os significados de cada código de erro possível.

**badRequest:** Transação não permitida ou suportada

**badDataFormat:** A requisição enviada possui formato inválido

**notApproved:** A autoridade de atributo rejeitou a emissão do atributo requisitado para o titular

**unacceptedExtension:** A extensão requisitada não é suportada

**untrustedHolder:** O titular não é confiável

**unsupportedAttribute:** O atributo requisitado não é suportado

**unsupportedTemplate:** O template requisitado não é suportado

**unknownSerial:** O serial fornecido para revogação não é conhecido

**systemFailure:** A requisição não pode ser tratada devido a uma falha no sistema

Um ponto importante a ressaltar é que o protocolo proposto dá suporte ao método *push* através do retorno do certificado de atributos na resposta à requisição de emissão e ao método *pull* através da busca por certificados de atributos, uma vez que o requisitante pode ser tanto o titular quanto um terceiro interessado no atributo.

## 4.4 Autenticação e Controle de Acesso

Está fora do escopo deste trabalho a definição de controles de acesso às diferentes funções suportadas por uma EEA, bem como tratar questões tais como a

privacidade dos atributos emitidos pela EEA. Entretanto, há tecnologias bastante difundidas para tratar o controle de acesso, como o emprego de Listas de Controle de Acesso (ACL), bem como para prover autenticação. Sugere-se para este segundo objetivo, encapsular as mensagens deste protocolo como o conteúdo do tipo `SignedData` ou `AuthenticatedData` de um pacote padrão CMS [HOU 09].

## 4.5 Transporte

Este trabalho também propõe alguns meios para se transmitir as mensagens definidas no capítulo `AttributeCertificateReq` 4.2.1 e `AttributeCertificateReq` 4.3.1.

### 4.5.1 Baseado em Socket TCP

O protocolo espera que exista um processo escutando por requisições numa porta definida (50005) no lado da autoridade de atributo. Clientes que desejam solicitar a emissão, revogação ou busca de certificado de atributo devem enviar uma mensagem cujos primeiros 4 bytes sejam o tamanho do resto da mensagem e o resto da mensagem é o ASN.1 do `AttributeCertificateReq` codificado em DER (4.2.1). A autoridade de atributo por sua vez, deve processar o pedido e responder com uma mensagem cujos primeiros 4 bytes sejam o tamanho do resto da mensagem e o resto da mensagem é o ASN.1 do `AttributeCertificateResp` codificado em DER(4.3.1).

Também pode ser utilizado canal seguro SSL para garantir, principalmente, que os requisitores são confiáveis, além de garantir a cifragem dos dados trocados.

### 4.5.2 HTTP

Funciona de maneira semelhante ao TCP, porém o autoridade de atributo deve estar escutando por requisições HTTP. O cliente deve realizar um GET/POST cujo content-type seja `application/ac-req` e o conteúdo o ASN.1 do `AttributeCertificateReq` codificado em DER (4.2.1). A autoridade de atributo deve responder com

content-type application/ac-resp e o conteúdo o ASN.1 do AttributeCertificateResp codificado em DER(4.3.1).

### **4.5.3 Chamada de Procedimento Remoto**

Pode ser implementado utilizando soluções como Web Service ou frameworks específicos como [FOU ].

# Capítulo 5

## Framework AAFW

O framework foi denominado de AAFW (Attribute Authority Framework) pois ele deve ser utilizado por autoridades de atributos que desejam gerenciar o ciclo de vida dos seus certificados de atributos. Ele é baseado exclusivamente no modelo de dados proposto no capítulo 4 e na RFC 5755 [FAR 10].

Sua proposta é conseguir tratar uma requisição de emissão, busca ou revogação de certificado de atributo, bastando para isso a autoridade realizar poucas configurações. Essas requisições, representam no mais baixo nível um `AttributeCertificateReq` (ver 4.2.1) e a resposta da autoridade um `AttributeCertificateResp`, definido em 4.3.1.

O framework é voltado para construir aplicações completas e não subsistema, porém ele conta com suporte para adição de subsistemas (como plug-ins) que são inicializados e destruídos junto com o framework.

### 5.1 Definindo o Comportamento da EEA

A fim de conseguir tratar o ciclo de vida do certificado de atributo, autoridades de atributos que desejam utilizar o framework AAFW devem informar ao framework como realizar algumas operações.

- Como carregar o certificado digital, algoritmo e chave privada para realização de

assinatura.

- Como armazenar e pesquisar num repositório os certificados de atributos emitidos.
- Como obter o próximo número serial do certificado de atributo a ser emitido.
- Como receber e enviar as mensagens `AttributeCertificateReq` 4.2.1 e `AttributeCertificateResp` 4.3.1.
- Como e onde publicar os certificados de atributo revogados, se desejar utilizar revogação.
- Como validar se os atributos presente numa requisição de emissão são aceitáveis para o titular requisitado.
- Como tratar o campo `templateId` presente numa requisição.

Essas operações representam o domínio do problema e são mapeadas para interfaces dentro do framework. São os “pedaços” de software que precisam ser desenvolvidos e informado ao framework para poder formar uma aplicação completa. A modelagem detalhada do Framework está disponível no Anexo A.

Para facilitar a implementação de uma EEA básica, o framework já oferece uma ou mais opções para realizar as operações descritas acima, permitindo que implementadores da entidade emissora de atributos especifiquem apenas comportamentos que fujam aos já contemplados.

### **5.1.1 Definindo o tratamento dos atributos**

Como definido no protocolo 4, é possível requisitar a emissão de atributos específicos ou simplesmente informando o campo `templateId`, que representa um identificador de modelo de certificado de atributo reconhecido pela EEA.

**Tratando atributos específicos** A forma com que a EEA consegue tratar solicitações de atributos específicos é através do registro validadores de atributo, representado pela interface *AttributeValidator*. Desta maneira, quando uma solicitação de emissão para determinado atributo é feita, o framework irá procurar por validadores nele registrado. Se achar um validador registrado para aquele OID do atributo solicitado, então é acionado seu método de validação. O método de validação deve informar se os atributos presentes numa requisição de emissão são aceitáveis para o titular requisitado.

Através do registro de validadores, a EEA pode responder por diversos atributos. Por exemplo, se forem registrados dois validadores, então ela pode emitir um certificado de atributo com um ou dois atributos, dependendo dos atributos presentes na requisição. Diferente da solicitação baseada no *templateID*, essa forma de requisição permite que o solicitante especifique os atributos que irão compor o certificado de atributos. Cabe à EEA apenas verificar se pode ou não emitir aquele atributo para o titular especificado no campo `holder` da requisição, conforme exposto no Capítulo 4.

**Tratando atributos com templates** Outra maneira de se realizar a emissão é através do registro de modelos de certificado de atributo, representado pela interface *ACTemplate*. Um modelo deve informar a validade, as extensões e os atributos que farão parte dos certificados a serem emitidos segundo seus critérios. O requerente deve simplesmente informar na requisição o campo `templateId` e o `holder` que receberá o certificado de atributos, deixando os demais campos vazios. O framework irá mapear o *templateId* recebido para um template nele registrado.

Através do registro de template, a EEA emite somente modelos de certificados de atributos previamente definidos. Por exemplo, se for registrado um modelo que define dois atributos, quando for solicitada a emissão utilizando o *templateId* correspondente, a EEA irá sempre emitir um certificado de atributos com os dois atributos definidos. Nesse caso a seleção dos atributos que irão compor o



certificado é de responsabilidade da EEA.

**Suporte à Revogação de atributos** Conforme requisito do DOC-ICP 16.01 [ITI a], a EEA deve indicar se suporta a revogação de atributos. A forma com que o framework dá suporte a essa funcionalidade é através do registro de publicador de LCR, representado pela interface *CRLPublisher*. O publicador de LCR diz onde publicar a LCR (LDAP, sistemas de arquivos, entre outros), além de configurações como seu período de validade e periodicidade de publicação. Um certificado de atributos terá codificada em sua estrutura a extensão *crlDistributionPoints*, com o ponto de distribuição informado pelo publicador.

Para que um atributo seja passível de revogação, basta registrar um publicador de LCR para ele. Com isso, sempre que for recebida uma requisição de revogação o framework irá verificar se existe publicador associado ao atributo a ser revogado. Caso exista, utilizará o publicador para realizar a revogação. O framework também fica encarregado de nunca deixar expirar as LCRs publicadas pelos publicadores. Sempre que uma LCR estiver prestes a expirar, uma outra será emitida em sua substituição.

A figura 5.1 mostra todas as interfaces do framework. Trata-se dos “pedaços” de software que necessitam ser implementados pelo usuário, conforme exposto anteriormente.

A classe central *AttributeAuthority* representa a autoridade de atributo. Ela contém todo o fluxo de tratamento da requisição. Durante esse fluxo, as interfaces são solicitadas para completar o algoritmo.

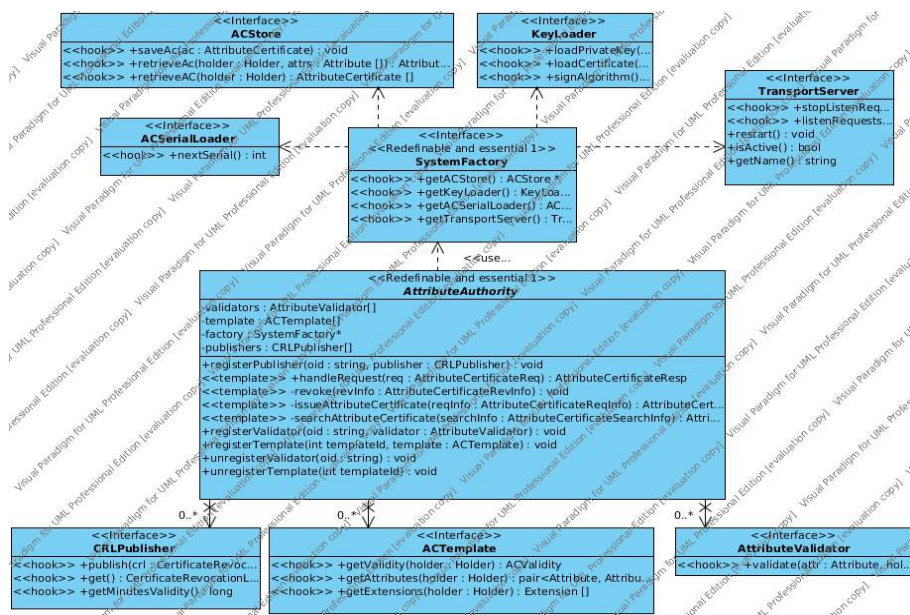
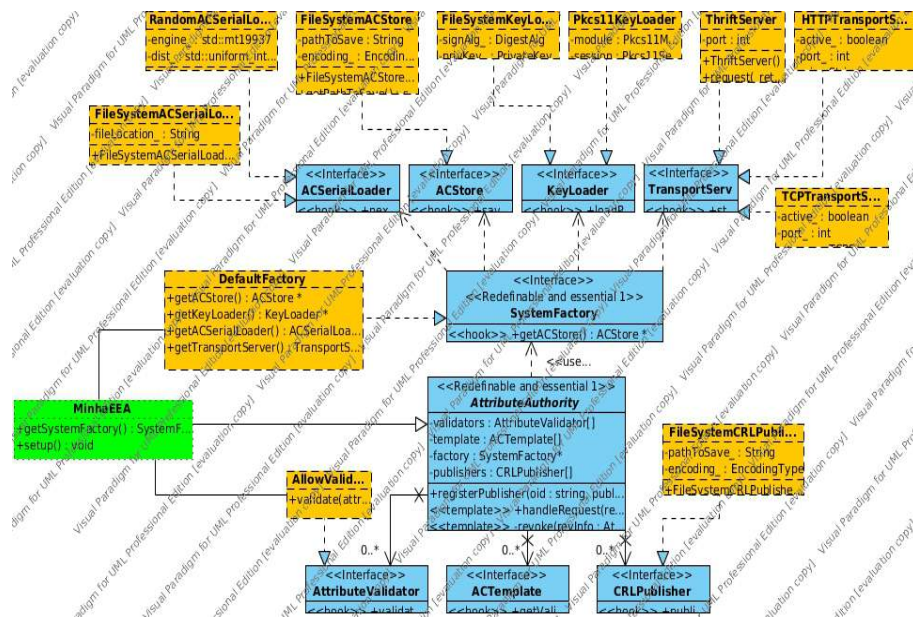


Figura 5.1: Interfaces do framework

## 5.2 Utilização do Framework

Para conseguir utilizar o framework o usuário deve herdar a classe *AttributeAuthority*, implementando seus métodos abstratos *getSystemFactory* e *setup*.

Na figura 5.2, os retângulos em azul representam as interfaces do framework, como mostrado na figura 5.1. Os retângulos em laranja (de borda tracejada) representam as implementações dessas interfaces, fornecidas pelo framework. O retângulo em verde (de borda pontilhada) representa uma possível aplicação implementada utilizando o framework.



**Figura 5.2:** Utilização do Framework

A figura mostra um caso onde o usuário criou uma subclasse de *AttributeAuthority* chamada de *MinhaEEA*. Pela associação com *DefaultFactory*, entende-se que será retornado uma instância de *DefaultFactory* no método *getSystemFactory*. O método *setup* dá a chance para a *MinhaEEA* registrar os seus validadores de atributos, template de atributos e revogadores de atributo. Pela associação com *AllowValidator*, entende-se que será registrado o validador *AllowValidator* para um ou mais atributo de *MinhaEEA*.

As implementações das interfaces do framework (representada pelos retângulos em laranja, de borda tracejada) são as maneiras de alteração do comportamento do framework. A tabela a seguir descreve o comportamento de cada uma dessas implementações.

Interface	Implementação	Comportamento
<i>ACSerialLoader</i>	<i>FileSystemACSerialLoader</i>	Utiliza um arquivo no sistemas de arquivos que contém o número serial do último certificado de atributo emitido. Sempre que um novo for emitido, o número serial será incrementado de forma sequencial.

<i>ACSerialLoader</i>	<i>RandomACSerialLoader</i>	Sempre gera um número pseudo aleatório para ser utilizado como número serial do próximo certificado de atributo.
<i>ACStore</i>	<i>FileSystemACStore</i>	Utiliza um diretório no sistema de arquivos para realizar a persistência e busca dos certificados de atributos emitidos.
<i>KeyLoader</i>	<i>FileSystemKeyLoader</i>	Faz o carregamento do certificado digital e chave privada para assinatura do certificado de atributo através de arquivos no sistemas de arquivos.
<i>KeyLoader</i>	<i>Pkcs11KeyLoader</i>	Faz o carregamento do certificado digital e chave privada para assinatura do certificado de atributo através de dispositivo que implemente a interface PKCS#11 [LAB 09].
<i>TransportServer</i>	<i>TCPTransportServer</i>	Utiliza o protocolo TCP para envio e recebimento de requisição e resposta, como consta em 4.5.1
<i>TransportServer</i>	<i>HTTPTransportServer</i>	Utiliza o protocolo HTTP para envio e recebimento de requisição e resposta, como consta em 4.5.2
<i>TransportServer</i>	<i>ThriftServer</i>	Utiliza chamada de procedimento remoto (RPC) para envio e recebimento de requisição e resposta. Implementa a parte servidor, com a tecnologia Apache Thrift.
<i>CRLPublisher</i>	<i>FileSystemCRLPublisher</i>	Utiliza um diretório no sistema de arquivos para publicação de LCARs.
<i>AttributeValidator</i>	<i>AllowValidator</i>	Não realiza nenhuma validação sobre os atributos a serem emitidos para o titular. Ou seja, sempre permite sua emissão.

<i>SystemFactory</i>	<i>DefaultFactory</i>	Utiliza <i>FileSystemACStore</i> para busca e persistência de CA, <i>FileSystemKeyLoader</i> para carregamento de chave privada e certificado (que são gerados auto assinados somente uma vez), <i>FileSystemACSerialLoader</i> para carregamento de número serial, e <i>TCPTransportServer</i> para o transporte de mensagens.
----------------------	-----------------------	---

Tabela 5.1: Implementações das interfaces do framework

A seguir serão mostrados exemplos de como alterar o comportamento do framework utilizando as implementações fornecidas ou criando sua própria.

### 5.2.1 Criando uma EEA Básica

A aplicação mais simples possível de construir utilizando o framework faz uso de suas implementações de interface já fornecidas. Como explicado na seção anterior, deve-se herdar a classe *AttributeAuthority*, implementando seus métodos abstratos *getSystemFactory* e *setup*.

Na implementação do método *getSystemFactory* ela deve fazer uso da *DefaultFactory*. Na implementação de *setup* deve registrar um validador para o atributo que se deseja emitir, utilizando o *AllowValidator* para não realizar nenhuma validação sobre o titular. Também é possível registrar um publicador de LCR para suportar revogação desse atributo, através da *FileSystemCRLPublisher*. O comportamento dessas implementações são explicados na tabela 5.1.

Este trecho de código representa esse cenário.

```

1 class MinhaEEA : public AttributeAuthority
2 {
3 public:
4     std::unique_ptr<SystemFactory> getSystemFactory()

```

```

5  {
6      return std::unique_ptr<SystemFactory>(new DefaultFactory);
7  }
8  void setup()
9  {
10     registerValidator("2.30.50.1.1.1", new AllowValidator);
11     registerCRLPublisher("2.30.50.1.1.1", new ←
        FileSystemCRLPublisher(60, "http://minhaEEA.com/eea.crl", ←
            "C:\\\\eea.crl"));
12     denyUnknownAttributes();
13 }
14 };

```

Essa aplicação de está pronta para receber requisições de certificados de atributo 4.2 e responder de acordo.

No exemplo, sempre que for requisitado a emissão de um CA contendo o atributo de OID 2.30.50.1.1.1, o framework irá aplicar a validação implementada pelo *AllowValidator*, através de seu método *validate*. Além disso, sempre que uma requisição para revogação que contenha esse atributo for feita, o framework irá chamar o método *publish* da implementação *FileSystemCRLPublisher*.

O método *denyUnknownAttributes* diz que deve ser negado a emissão para qualquer atributo desconhecido (aqueles que não tem um validador associado). Para iniciar a aplicação deve-se invocar o método *run()* de uma instância da *MinhaEEA*.

Para melhor entendimento da estrutura e dinâmica do framework, aconselha-se verificar os diagramas UML, presentes no apêndice A desse trabalho.

Exemplo completo em B.1.

## 5.2.2 Implementando a SystemFactory

Este exemplo irá mostrar como implementar sua própria *SystemFactory* para customizar alguns dos comportamento do framework. A *SystemFactory* contém assinatura de métodos que devem retornar a implementação das outras interfaces do

sistema. Os comportamentos que se desejam obter são:

- Armazenar e buscar certificados de atributos num diretório específico do sistema de arquivos;
- Realizar carregamento de certificado chave privada contidos em HSM;
- Utilizar um arquivo num lugar específico do sistema de arquivos para fazer controle do número serial dos CAs;
- Utilizar o protocolo TCP na porta 50005 para envio e recebimento de mensagens;

```

1 class MinhaFactory : public SystemFactory
2 {
3 public:
4     std::unique_ptr<ACStore> getACStore() const
5     {
6         return std::unique_ptr<ACStore>(new FileSystemACStore("/home/
7             /giovani/acs", EncodingType::DER));
8     }
9     std::unique_ptr<KeyLoader> getKeyLoader() const
10    {
11        return std::unique_ptr<KeyLoader>(new Pkcs11KeyLoader(
12            DigestAlg::SHA1,
13            "/usr/lib/softhsm/libsofthsm.so", "Chave AA", "
14            123456", "label1", "Certificado AA"));
15    }
16    std::unique_ptr<ACSerialLoader> getACSerialLoader() const
17    {
18        return std::unique_ptr<ACSerialLoader>(new
19            FileSystemACSerialLoader("/home/giovani/serial.txt"));
20    }
21    std::unique_ptr<TransportServer> getTransportServer() const
22    {
23        return std::unique_ptr<TransportServer>(new
24            TCPTransportServer(50005));
25    }
26 };

```

```

20     }
21 };

```

Exemplo completo em B.2.

### 5.2.3 Criando Validador de Atributo

Este exemplo irá mostrar como validar os atributos presentes numa requisição de emissão 4.2.2. A validação que se deseja obter é a seguinte: Garantir que o titular seja do tipo autônomo (não vinculado ao certificado digital) e que o seu nome está contido numa lista de nomes confiáveis.

```

1  class IsTrustedValidator : public AttributeValidator
2  {
3  public:
4      void validate(const Attribute& attribute, const Holder& holder) const
5      {
6          if(holder.getType() != Holder::HolderType::ENTITY_NAME)
7              throw RejectRequestException("Requisição inválida, ←
8              esperado tipo Autônomo!", ACStatusInfo::ACFailureInfo::←
9              badRequest);
10
11         std::string holderName = holder.getHolderEntityName().←
12         getEntry(ObjectIdentifier(NID_commonName));
13
14         auto it = std::find(trustedNames_.begin(), trustedNames_.end()←
15         (), holderName);
16         if(it == trustedNames_.end())
17             throw RejectRequestException("Titular "+holderName+" não ←
18             é confiável!", ACStatusInfo::ACFailureInfo::←
19             untrustedHolder);
20     }
21 private:
22     std::vector<std::string> trustedNames_{"Giovani Milanez ←
23     Espindola", "Jeandré Monteiro Sutil", "Ruy Ramos", "Ricardo ←

```



```

17     "Felipe Custódio"};
};

```

O validador de atributos implementado deve ser registrado no método *setup*, como mostrado no exemplo 5.2.1. Sempre que uma requisição para emissão que contenha o atributo de OID 2.30.50.1.1.1 for feita, o framework irá chamar o método *validate* da implementação *IsTrustedValidator*. Caso o nome do titular não esteja na lista de nomes confiáveis ou ele não seja do tipo autônomo, uma exceção é lançada informando ao framework que a requisição deve ser negada. O exemplo considera que o titular do tipo autônomo deve ter codificado seu *Common Name*.

Exemplo completo em B.4.

## 5.2.4 Criando Template

Este exemplo mostra como tratar o campo *templateId* presente numa requisição de emissão 4.2.2. O comportamento que se deseja atingir é: Sempre que for recebido uma requisição de emissão com *templateId* de valor 10, deve-se emitir um CA com validade de 1 dia e com dois atributos. Um que diz que o aluno é regular e outro que diz que ele pertence ao curso de Sistemas de Informação.

```

1 class MyTemplate : public ACTemplate
2 {
3 public:
4     AttributeCertificateValidity getValidity(const Holder& holder) ←
5         const
6     {
7         return AttributeCertificateValidity(60 * 24);
8     }
9     std::vector<std::pair<Attribute, std::unique_ptr<←
10         AttributeValidator>>> getAttributes() const
11     {
12         std::vector<std::pair<Attribute, std::unique_ptr<←
13             AttributeValidator>>> attrs;
14         attrs.push_back(

```

```

12     std::make_pair(Attribute(ObjectIdentifier("2.30.50.1.1.2" ←
        ), "Aluno regular"), std::unique_ptr< ←
        AttributeValidator>(new AllowValidator));
13     attrs.push_back(
14         std::make_pair(Attribute(ObjectIdentifier("2.30.50.1.1.3" ←
        ), "Sistemas de Informação"), std::unique_ptr< ←
        AttributeValidator>(new AllowValidator));
15
16     return attrs;
17 }
18 std::vector<Extension> getExtensions(const Holder& holder) ←
19     const
20 {
21     return std::vector<Extension>();
22 };

```

Sempre que uma requisição para emissão contendo o campo `templateId` com valor 10 for feita, o framework irá chamar os métodos `getValidity`, `getAttributes` e `getExtensions` da implementação `MyTemplate`. Foi associado para os atributos o validador `AllowValidator`, que sempre irá permitir a emissão dos mesmos. Note que para uma implementação mais realista, deve-se implementar seu próprio `AttributeValidator` 5.2.3 que faça as verificações para garantir que o titular tenha o privilégio para recebê-los.

Exemplo completo em B.4.

## 5.3 Tecnologias Envolvidas

Para o tratamento de estruturas ASN.1 como chave pública, chave privada, certificado, certificado de atributo, etc. foi utilizada a biblioteca OpenSSL 5.3.1, encapsulada pela biblioteca Cryptobase 5.3.2. Para implementação da parte de comunicação TCP/HTTP foi utilizado a biblioteca POCO C++ Libraries 5.3.3 e para comunicação RPC a biblioteca Apache Thrift 5.3.4. O framework foi implementado em C++.

### 5.3.1 OpenSSL

O projeto OpenSSL [PRO ] é open source e fornece uma implementação dos protocolos SSL e TLS além de uma biblioteca de criptografia de propósito geral. Neste trabalho foi utilizado sua biblioteca de criptografia. Uma de suas utilidades é conseguir codificar e decodificar estruturas ASN.1, fornecendo suporte para tipos definidos pelo usuário. Essa funcionalidade foi utilizada para codificar/decodificar os tipos definido em 4.

### 5.3.2 Cryptobase

A cryptobase foi construída pelo autor desse trabalho devido à falta de boas opções disponíveis para uso. Essa biblioteca fornece toda a base de criptografia para o framework aafw apresentado neste capítulo. Implementa classes C++ que encapsulam as estruturas do OpenSSL, que é construído em C. Possibilita elevado nível de abstração ao manusear essas estruturas, como também para realização de operações criptográficas como assinatura digital e funções hash. Uma estrutura X509 é mapeada para uma classe `Certificate` com métodos para obter o número serial e validade, por exemplo. Ela será publicada junto com o código fonte do framework desse trabalho.

### 5.3.3 POCO C++ Libraries

A POCO C++ Libraries [GMB ] fornece bibliotecas de uso geral de alto nível de abstração para tarefas como manuseio do sistema de arquivos, sockets, stream, logging, processos, etc... Ela foi utilizada no framework principalmente para adicionar suporte a subsistemas, implementação de servidor TCP e HTTP e abstração do sistemas de arquivos, permitindo produzir um código multi plataforma.

### **5.3.4 Apache Thrift**

O Apache Thrift [FOU ] é um framework para RPC, chamada de procedimento remoto. Ele permite que seja definido estruturas de dados e serviços utilizando sua IDL, linguagem de definição de interface. Dado a IDL ele produz a parte cliente automaticamente e gera o esqueleto para implementação do servidor. Ele consegue produzir código para até 16 linguagens de programação diferentes. O thrift foi utilizado como uma implementação da interface TransportServer.

# Capítulo 6

## Aplicações Protótipo

Para avaliar o protocolo definido e framework que o implementa, foi desenvolvido um sistema prevendo um cenário real de uso. O cenário proposto foi o de uma rede de cinemas que deseja utilizar certificados de atributos para vender bilhetes virtuais seguros pela internet, para as sessões de seus cinemas. Esse cenário é explorado em mais detalhes ao longo deste capítulo.

### 6.1 Aplicação para controle de ingressos

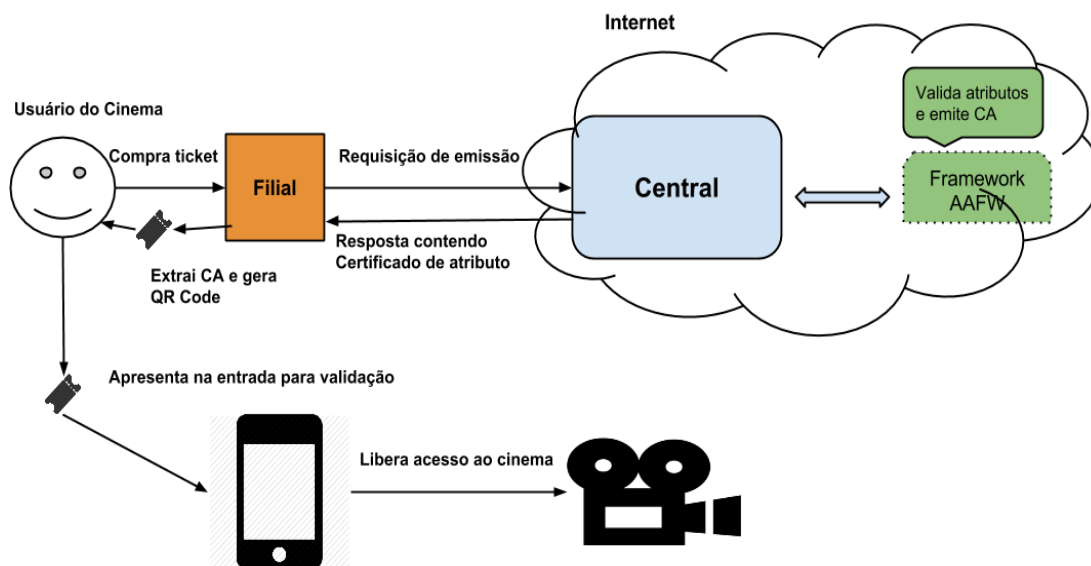
Foi desenvolvido um conjunto de aplicações para representar um cenário baseado nos seguintes requisitos:

1. Existe uma rede de cinemas que deseja melhorar a segurança e facilidade de acesso ao seus ingressos (tickets).
2. As filiais devem ser gerenciadas de modo que a emissão de ingressos seja previamente aprovada e feito por uma central.
3. Para aprovação de ingressos de estudantes eles devem estar cadastrados na central para garantir seu benefício. Para aprovação de ingressos para adultos nenhuma verificação adicional é feita. Supõe-se aqui que utiliza-se uma rede privada e que a central é capaz de restringir o acesso às suas filiais.

4. Cada ingresso emitido deve ser verificado quanto a sua validade, autenticidade e integridade a fim de permitir a entrada no cinema.

O cenário exposto justifica o uso de certificação de atributos, visto que o ingresso nada mais é do que uma permissão temporária para usufruir de um evento e que é emitido por uma entidade confiável com poder para tal.

O framework AAFW, em conjunto com a biblioteca *cryptobase* (ver 5.3.2), foram utilizados para implementar um sistema que atenda ao cenário apresentado na Figura 6.1.



**Figura 6.1:** Arquitetura da solução de ingressos.

O usuário se apresenta ao balcão de uma filial e solicita a compra de um ticket para determinada sessão. A filial se comunica com a central para realização da emissão. Ao receber a resposta, ela imprime o ticket no formato QR Code e o entrega para o usuário. O usuário, de posse do ticket impresso se encaminha para entrada onde ocorre a sessão. A seguir lhe é requisitado o ticket por um funcionário, que o escaneia através de um aplicativo instalado num *smart phone*. Será apresentado pelo aplicativo o resultado da validação do ticket bem como os dados da sessão comprada pelo usuário. Se o ticket estiver válido, o acesso à sessão lhe é liberado.

As aplicações construídas foram:

- A autoridade de atributo em si, uma central que irá responder por requisições de emissão, busca e revogação de ingressos de cinemas.
- As filiais que representam unidades de cinema, cada uma com seus próprios filmes e sessões. Elas serão responsáveis por requisitar a emissão, busca e revogação de ingressos de cinemas para a central.
- Uma aplicação que valida o ingresso para o cinema a fim de permitir a entrada do usuário comprador.

### **6.1.1 EEA para Acesso aos Cinemas**

Foi criada para rede de cinemas uma autoridade de atributo que tem a função de emitir e distribuir os ingressos para o filmes de suas filiais. Representa a central. Ela também deve manter uma base de estudantes para aprovação de emissão de tickets de estudantes.

Considerando que a rede de cinemas já possui um sistema que gerencie suas filiais ela terá pouco trabalho para a implementação desse cenário. Terá apenas de dizer ao framework AAFW como armazenar e buscar os certificados de atributos emitidos em seu próprio repositório e também como validar as requisições para o tipo estudante.

Feito isso, a atuação da EEA para emissão de ingresso será passiva. Ela irá receber uma requisição 4.2 de uma de suas filiais e automaticamente emitirá o ingresso, onde tudo é encapsulado pelo framework AAFW.

#### **6.1.1.1 Modelo de tickets**





Parte do sistema onde é definido um OID para cada modelo de ticket - estudante ou adulto - que será utilizado posteriormente como um atributo na codificação do certificado de atributo. Para cada atributo definido, é associado seu tipo de validação e também indicado se ele suporta ou não revogação. Esses dados serão mapeados para

implementações das interfaces `AttributeValidator` e `CRLPublisher` dentro do framework `aafw 5` e configurados no método `setup`.

CineAA Inicio Tickets Emitidos Estudantes Modelos de Tickets

### Autoridade de Atributo de Cinemas :: Modelos de Ticket

+ Novo

OID	Descrição	Tipo Validação	Suporte Revogação	Ações
2.30.50.1.1.1	Ticket Cinema CineAA - Adulto	Nenhuma	Nao	 
2.30.50.1.1.2	Ticket Cinema CineAA - Estudante	Cadastrado como estudante	Sim	 

Giovani Milanez Espindola, 2014

**Figura 6.2:** Interface de gerenciamento de modelos de tickets.

### 6.1.1.2 Cadastramento de estudantes

Para garantir o benefício de estudante a pessoa deve estar cadastrada na base de dados da autoridade de atributo.



CineAA Inicio Tickets Emitidos **Estudantes** Modelos de Tickets

### Autoridade de Atributo de Cinemas :: Estudantes

+ Novo

Nome	CPF	Ações
Giovani Milanez	07958151900	 
Jose	00000000000	 

Giovani Milanez Espindola, 2014

**Figura 6.3:** Interface de gerenciamento de estudantes.

### 6.1.1.3 Tickets Emitidos

Parte do sistema onde são mostrados os tickets (certificados de atributos) emitidos por essa autoridade de atributo.

CineAA Inicio **Tickets Emitidos** Estudantes Modelos de Tickets

### Autoridade de Atributo de Cinemas :: Tickets Emitidos

Nome	Início Validade	Fim Validade	Conteúdo
Jose	22/03/2014 22:30:00	22/03/2014 23:20:00	Cinema 3D - Tim Tim. SALA 4
Giovani	24/03/2014 00:30:00	24/03/2014 01:30:00	Cinema 2D - Thor. SALA 1

Giovani Milanez Espindola, 2014

**Figura 6.4:** Lista dos certificados de atributos (tickets) emitidos.

## 6.1.2 Filial

As filiais terão a responsabilidade de requisitar a emissão de tickets para a sua EEA. Representa a terceira parte interessada na emissão de certificado de atributo. Considerando que elas já possuem um sistema de emissão de tickets, então pouca alteração será necessária para a adequação do novo cenário. Seu trabalho será apenas o de montar uma requisição 4.2 com os dados da sessão (nome do filme, sala, horário, validade) codificado no atributo a ser requisitado (estudante ou adulto). Além disso ela terá que extrair o certificado de atributo (ticket) da resposta.

Para essa tarefa ela não precisará usar o framework AAFW, apenas a biblioteca cryptobase 5.3.2 que contém classes para manipulação da requisição e resposta. Ela também precisará saber como enviar e receber as mensagens, utilizando socket TCP por exemplo.

### 6.1.2.1 Gerenciamento de sessões

Parte do sistema onde são gerenciados as sessões dos filmes.

Venda Tickets Cinema :: Estudantes

- [Inicio](#)
- [Filmes](#)
- [Tickets](#)

[Voltar](#) [Novo Filme](#)

Nome	Sala	Tipo	Inicio	Duracao	Ações
Tim Tim	4	3D	23/03/2014 01:30:00	50	<a href="#">Editar</a> <a href="#">Remove</a>
Thor	1	2D	24/03/2014 03:30:00	60	<a href="#">Editar</a> <a href="#">Remove</a>
Forrest Gump	7	4D	22/03/2014 14:00:00	160	<a href="#">Editar</a> <a href="#">Remove</a>

Giovani Milanez Espindola, 2014

**Figura 6.5:** Interface de gerenciamento de sessões.

### 6.1.2.2 Gerenciamento de tickets

Parte do sistema onde são gerenciados os tickets. Nessa parte pode-se verificar os três casos de uso A.2 principais do framework AAFW: Emitir Certificado de Atributo, Buscar Certificado de Atributo e Revogar Certificado de Atributo.

Venda Tickets Cinema :: Tickets

[Inicio](#)  
[Filmes](#)  
[Tickets](#)

Cliente	Tipo	CPF	Filme	Horário	Ações
Giovani Milanez Espindola	estudante	07958151900	Forrest Gump	22/03/2014 14:00:00	<input type="button" value="Imprimir"/> <input type="button" value="Revogar"/>
Carlos Silveira	adulto		Tim Tim	23/03/2014 01:30:00	<input type="button" value="Imprimir"/> <input type="button" value="Revogar"/>

Giovani Milanez Espindola, 2014

**Figura 6.6:** Interface de gerenciamento de tickets.

### 6.1.2.3 Emissão de tickets

Parte do sistema onde são emitidos os tickets. Representa o caso de uso Emitir Certificado de Atributo. Ao preencher os dados, uma requisição de emissão `AttributeCertificateIssueReq` 4.2.2 é construída. Será utilizado o nome informado para solicitar um CA do tipo autônomo 3.7. Esse dado será codificado na estrutura `holder`. Pode ser escolhido também o tipo do ingresso: adulto ou estudante, o que resultará em atributos diferentes na requisição. O tipo de ingresso será codificado na estrutura `attributes`. Outro dado necessário é a validade `validityPeriod` que será preenchida com os dados de início e término da sessão escolhida.

A requisição depois de construída é então enviada para autoridade de atributo de cinemas 6.1.1 e ao receber a resposta, extraído o certificado de atributo.

Venda Tickets Cinema :: Emissão de Ticket

- [Inicio](#)
- [Filmes](#)
- [Tickets](#)

Nome:

CPF:

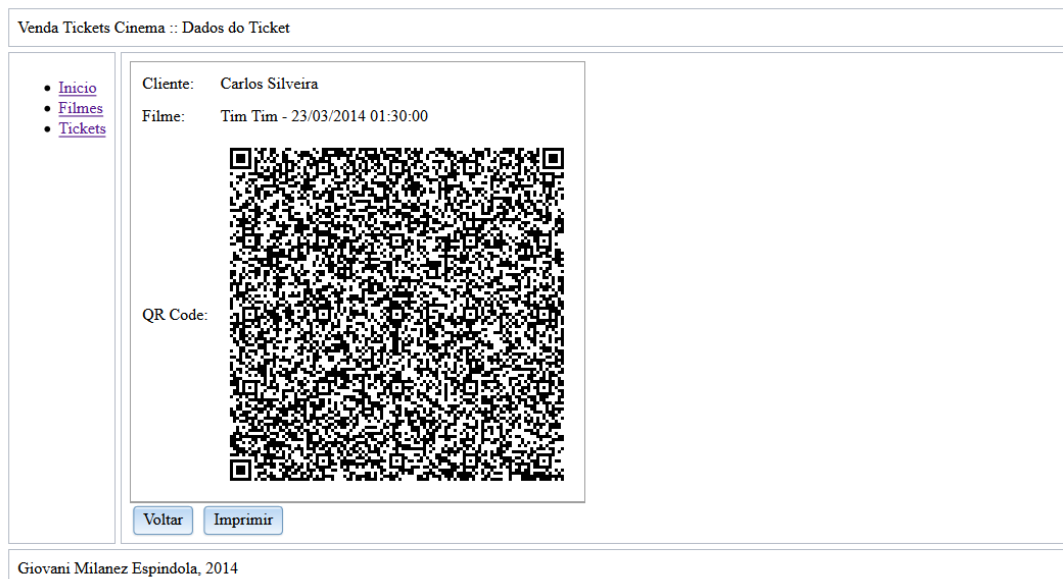
Tipo:

Filme:

Giovani Milanez Espindola, 2014

**Figura 6.7:** Interface de emissão de tickets.

**Impressão de tickets** Após emitir um ticket o usuário será redirecionado para interface de impressão. O QR code representa o certificado de atributo codificado em PEM. O usuário deve apresentá-lo para ter direito de assistir seu filme comprado. O ingresso passará pela verificação automática da sua validade, integridade e autenticidade realizado por um outro aplicativo 6.1.3.



**Figura 6.8:** Interface de impressão de tickets.

#### 6.1.2.4 Busca de tickets

Parte do sistema onde é permitido buscar por tickets emitidos pela autoridade de atributo de cinemas. Representa o caso de uso Buscar Certificado de Atributo. Ao preencher os dados, uma requisição de busca 4.2.4 é construída, enviada para autoridade de atributo de cinemas 6.1.1 e extraído os certificados de atributo recebidos.

A maneira com que essa requisição é construída assemelha-se ao caso de emissão 6.1.2.3. Porém, é populado uma estrutura `AttributeCertificateSearchReq`, fazendo um filtro pelo atributo especificado (estudante ou adulto) e pelo titular (nome) informado.

Venda Tickets Cinema :: Busca de tickets emitidos por terceiros

- [Inicio](#)
- [Filmes](#)
- [Tickets](#)

Nome:

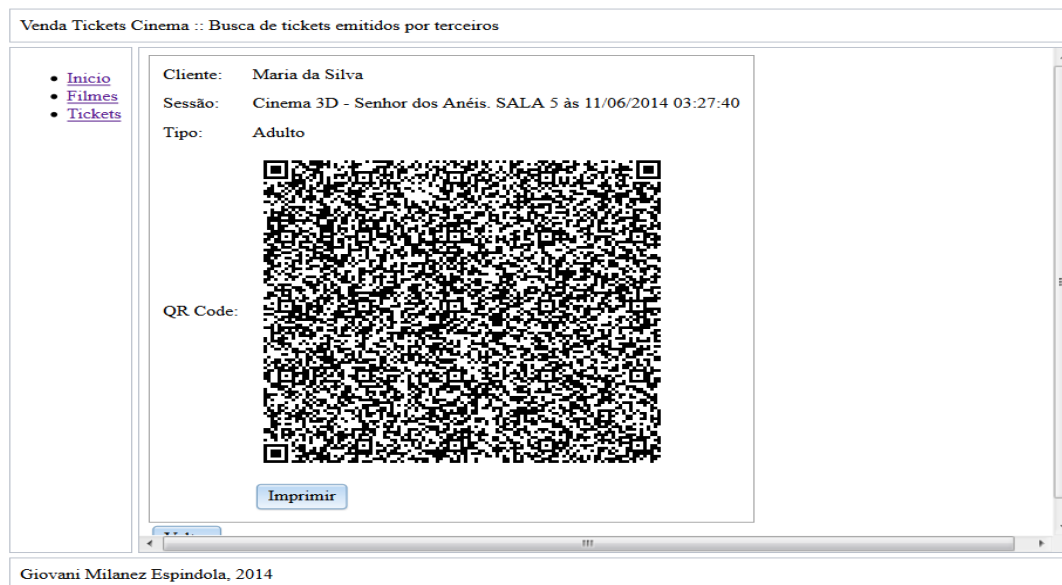
CPF:

Tipo:

Giovani Milanez Espindola, 2014

**Figura 6.9:** Interface de busca de tickets.

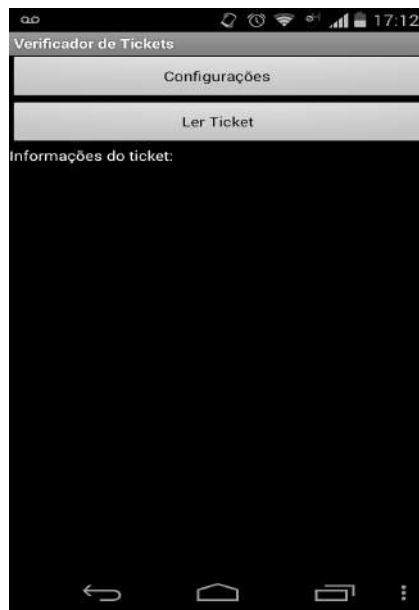
**Resultado da busca** Após realizar a busca o usuário será redirecionado para interface de resultado da busca. Nenhum ou vários tickets podem ser obtidos. Perceba que foi procurado pelo nome 'Maria da Silva' e que não havia sido emitido nenhum ticket para ela por esta filial, como mostra na figura 6.6. Porém, um ticket ticket foi emitido pela EEA para essa pessoa, como consta no resultado. Esse ticket pode ter sido emitido por outra filial, operando um site de vendas online, por exemplo.



**Figura 6.10:** Interface de resultado da busca de tickets.

### 6.1.3 Verificador de tickets

Para conseguir acesso à sessão, o usuário deverá apresentar o ticket para verificação. Para isso foi desenvolvido uma aplicação android que utiliza a biblioteca cryptobase 5.3.2. A verificação consiste na validação da integridade da assinatura, na validade do certificado de atributo, na presença dos atributos de adulto ou estudante 6.1.1.1 e na confiança do seu emissor (autoridade de atributo de cinemas). Processo detalhado em 3.5

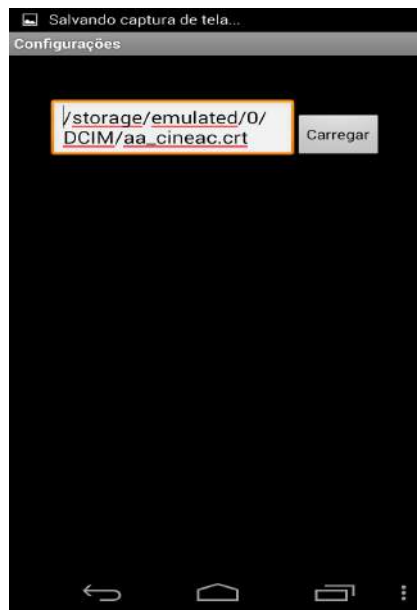


**Figura 6.11:** Interface do verificador

### 6.1.3.1 Configurações

Parte do sistema onde é configurado a autoridade de atributo confiável. A configuração é feita mediante do upload do seu certificado, que será utilizado para verificar a assinatura do certificado de atributo.





**Figura 6.12:** Interface de configuração de autoridade confiável

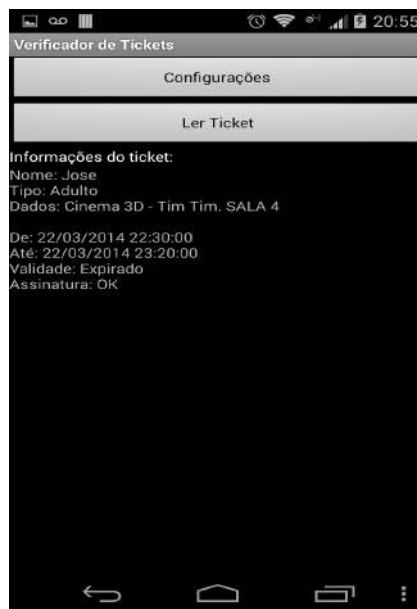
### 6.1.3.2 Leitura de ticket

Parte do sistema onde é feita a leitura dos tickets. O usuário deve imprimir seu ticket e apresentá-lo para poder ter acesso ao cinema.



**Figura 6.13:** Interface para leitura de tickets

**Resultado da leitura** A leitura mostra que a assinatura está íntegra e confiável, porém a validade está expirada, portanto o usuário não poderá ter acesso ao cinema.



**Figura 6.14:** Interface de informações sobre o tickets

## 6.2 Outras aplicações

Diversos outros tipos de aplicações podem ser construídas utilizando o framework. Tudo que representa determinado poder, privilégio ou atribuições a uma entidade pode ser representado por um certificado de atributo e portanto passível de ser utilizado por uma autoridade para emissão e controle dos mesmos. O caso citado acima, ingressos de cinema, pode ser abstraído para qualquer tipo de ingresso para qualquer evento. Outros exemplo são:

**Atestados de matrícula em universidades** Universidades/Escolas podem controlar a situação de seus alunos com certificado de atributo

Possíveis atributos seriam:

- Curso do aluno: Sistemas de informação, Ciência da computação

- Estado do aluno: regular, trancado

Os atributos seriam escolhidos para formar o certificado de atributo e ser emitido por uma entidade, por exemplo a UFSC. A parte interessada para solicitar a emissão seria o próprio aluno ou a UFSC, e as partes interessadas em consumir o certificado de atributo seriam cinemas, teatros, empresas de ônibus

**Controle de acesso** Sistemas informatizados podem fazer uso do certificado de atributo para fazer seu controle de acesso à recursos.

Possíveis atributos seriam:

- Tipo do usuário: Administrador, Operador, Convidado

Pode ser utilizado por websites em que fazem o papel de solicitar a emissão e também realizar a verificação dos mesmo. Pode ser utilizado também por portas com controle de acesso, que fazem o papel de verificar o certificado de atributo a fim de se destrancar.

**Consulta de crédito** Serviços como SPC e SERASA podem aderir ao uso de certificado de atributo.

Possíveis atributos seriam:

- situação: adimplente, inadimplente

No caso, a solicitação de emissão viria do próprio SPC. Lojistas que queiram verificar a situação de algum cliente basta solicitar a busca no repositório do SPC.

# Capítulo 7

## Considerações Finais

O desenvolvimento do trabalho foi feito levando-se em consideração os documentos principais DOC-ICP-16 e DOC-ICP-16.01 que falam sobre certificação de atributos para uso no âmbito da ICP-Brasil. As características do framework que compravam essa afirmação são:

- Permite a gestão do ciclo de vida dos certificados de atributos pela Entidade Emissora de Certificado de Atributo;
- Permite que a EEA escolha utilizar suporte a revogação de certificados de atributo ou não;
- Permite realizar a assinatura do certificado de atributo utilizando certificado digital de tipo Pessoa-Jurídica A3 ou A4 pertencente à cadeia de confiança da ICP-Brasil. Suporte para dispositivos criptográficos, inclusive HSM, que forneçam implementação PKCS#11;
- Permite o uso de certificado de atributo, vinculando o titular ao certificado digital ou de forma autônoma (nome de entidade);
- Segue o padrão ITU X.509, de acordo com o perfil estabelecido na RFC 5755, facilitando a interoperabilidade entre os sistemas.

## 7.1 Contribuições

Como contribuições desse trabalho, podemos citar:

1. Proposta de um protocolo para solicitação de certificado de atributo, considerando seu ciclo de vida;
2. Implementação de um framework de código aberto e multiplataforma que utiliza o protocolo proposto para desenvolvimento de sistemas de autoridades de atributo;
3. Implementação de uma biblioteca de código aberto e multiplataforma para manipulação em alto nível de estruturas ASN.1 e criptografia;
4. Implementação de aplicação verificadora de certificado de atributo;
5. Um incentivo para aplicação da tecnologia de certificação de atributo para uso no Brasil e exterior.

## 7.2 Trabalhos Futuros

Como trabalhos futuros, podemos citar:

1. Prever forma para validação da autenticidade do titular. Ou seja, validar que quem apresenta o certificado de atributo para uso é realmente o seu titular, como codificado em sua estrutura.
2. Um controle de privilégios embutido no protocolo sobre quem pode solicitar a emissão, busca ou revogação do certificado de atributo (ver 4.4). Uma maneira de resolver esse problema seria encapsulando a estrutura `AttributeCertificateReq` num pacote assinado CMS do tipo `Signed-data`[HOU 09]. A EEA, ao receber a mensagem, teria a informação do assinante e a assinatura da estrutura, podendo decidir se permite ou não o processamento dessa requisição;

3. Prever a solicitação de múltiplas requisições na mesma mensagem;
4. Fornecer uma implementação no framework para trabalhar com LDAP (na interface ACStore, por exemplo);
5. Avaliação de performance do recebimento de solicitação e envio de respostas;
6. Avaliação de segurança do protocolo e implementação do framework;



# Referências

- [ADA 08] ADAMS, C. et al. **Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)**. RFC 3161 (Proposed Standard).
- [BRA ] BRASIL. **Medida provisória n.º 2.200, de 28 de junho de 2001. Institui a Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil, e dá outras providências. Diário Oficial da União, Brasília, DF, 29 jun. de 2001.**
- [CAS ] CASTLE, B. **The Legion of the Bouncy Castle**. Disponível em <<http://www.bouncycastle.org/>>. Acesso em: 15 de maio de 2012.
- [CCI 88a] CCITT. **Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1)**.
- [CCI 88b] CCITT. **Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)**.
- [CHO 02] CHOUDHURY, S.; BHATNAGAR, K.; HAQUE, W. **Public Key Infrastructure Implementation and Design**. 1st. ed. New York, NY, USA: John Wiley & Sons, Inc., 2002.
- [COO 08] COOPER, D. et al. **Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile**. RFC 5280 (Proposed Standard).
- [dTdI ] DE TECNOLOGIA DA INFORMAÇÃO, I. N. **DOC-ICP**. Disponível em <<http://www.iti.gov.br/twiki/bin/view/Certificacao/DocIcp>>. Acesso em: 22 de maio de 2012.
- [FAR 02] FARRELL, S.; HOUSLEY, R. **An Internet Attribute Certificate Profile for Authorization**. RFC 3281 (Proposed Standard). Obsoleted by RFC 5755.
- [FAR 10] FARRELL, S.; HOUSLEY, R.; TURNER, S. **An Internet Attribute Certificate Profile for Authorization**. RFC 5755 (Proposed Standard).



- [FOU ] FOUNDATION, A. S. **Apache Thrift**. Disponível em <<http://thrift.apache.org/>>. Acesso em: 10 de junho de 2014.
- [FRE 11] FREIER, A.; KARLTON, P.; KOCHER, P. **The Secure Sockets Layer (SSL) Protocol Version 3.0**. RFC 6101 (Historic).
- [GMB ] GMBH, A. I. S. E. **POCO C++ Libraries**. Disponível em <<http://pocoproject.org/>>. Acesso em: 10 de junho de 2014.
- [GUI 08] GUILHEN, S. N. **Um serviço de autorização Java EE baseado em certificados de atributos X.509**. Universidade de São Paulo, agosto de 2008. Dissertação de Mestrado.
- [HOU 01] HOUSLEY, R.; POLK, T. **Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure**. New York, NY, USA: John Wiley & Sons, Inc., 2001.
- [HOU 09] HOUSLEY, R. **Cryptographic Message Syntax (CMS)**. RFC 5652 (INTERNET STANDARD).
- [ITI a] ITI. **PERFIL DE USO GERAL E REQUISITOS PARA GERAÇÃO E VERIFICAÇÃO DE CERTIFICADOS DE ATRIBUTO NA ICP-BRASIL. Versão 1.0. Brasília, Dezembro 2012. DOC-ICP-16.01.**
- [ITI b] ITI. **VISÃO GERAL SOBRE CERTIFICADO DE ATRIBUTO PARA A ICP-BRASIL. Versão 1.0. Brasília, Julho 2012. DOC-ICP-16.**
- [ITU 00] ITU, I. T. U. Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. Geneva: International Telecommunication Union, ago, 2000. Series x: Data networks, open system communications and security directory.
- [Jac ] Jacques Philippe Sauvé. **O que é um framework?** Disponível em <<http://www.dsc.ufcg.edu.br/jacques/cursos/map/html/frame/oque.htm>>. Acesso em: 12 de junho de 2014.
- [KAL 98] KALISKI, B. **PKCS #7: Cryptographic Message Syntax Version 1.5**. RFC 3852 (Proposed Standard).
- [LAB 09] LABORATORIES, R. **PCKCS #11 Base Functionality v2.30: Cryptoki Draft 4**.
- [NYS 00] NYSTROM, M.; KALISKI, B. **PKCS #10: Certification Request Syntax Specification Version 1.7**. RFC 2986 (Informational).
- [OIK 06] OIKAWA, M.; TAGAWA, Y. **Attribute certificate validation method and device**. US Patent App. 11/340,788.

- [PIN 08] PINKAS, D.; POPE, N.; ROSS, J. **CMS Advanced Electronic Signatures (CAAdES)**. RFC 5126.
- [PRO ] PROJECT, T. O. **OpenSSL: The Open Source toolkit for SSL/TLS**. <https://www.openssl.org/>.
- [SCH 05] SCHAAD, J. **Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)**. RFC 4211 (Proposed Standard).
- [SIL 07] SILVA, R. P. **UML: Modelagem Orientada a Objetos**. Visual Books, 2007.
- [SIL 09] SILVA, R. P. **Como modelar com UML 2**. Visual Books, 2009.
- [TEC 02] TECHNICAL, R. L. Pkcs #1: Rsa cryptography standard v2.1. RSA Data Security, 2002. Relatório técnico.
- [YEE ] YEE, P. **Attribute Certificate Request Message Format**.
- [ZHO ] ZHOU, W.; MEINEL, C. **Implement Role-Based Access Control with Attribute Certificates**.

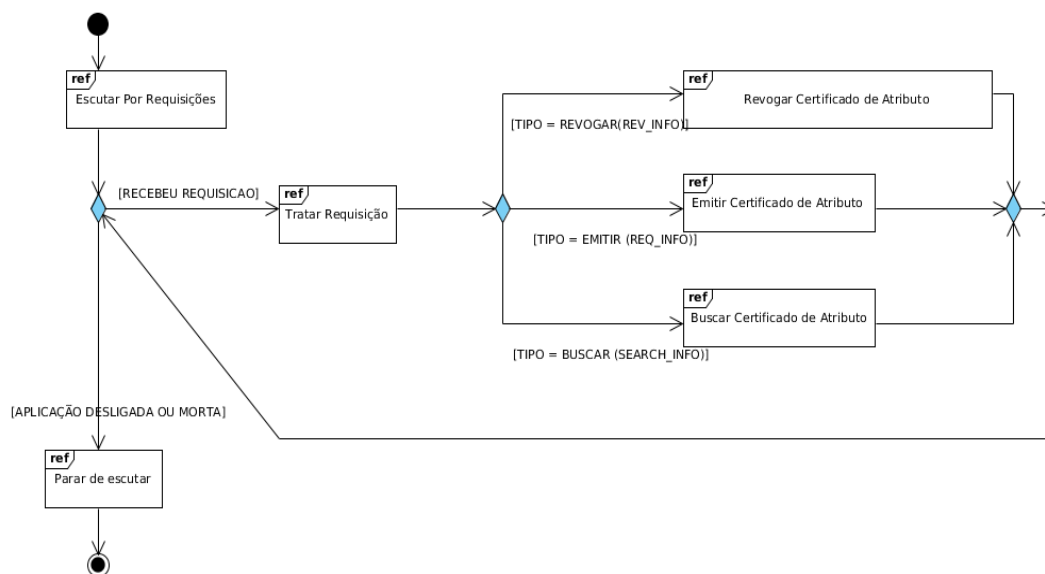
# **Apêndice A**

## **Documentação UML**

Para melhor entendimento por parte dos usuários desenvolvedores foi produzido alguns diagramas UML abordando modelagem estrutural e dinâmica.

### **A.1 Diagrama de visão geral de interação**

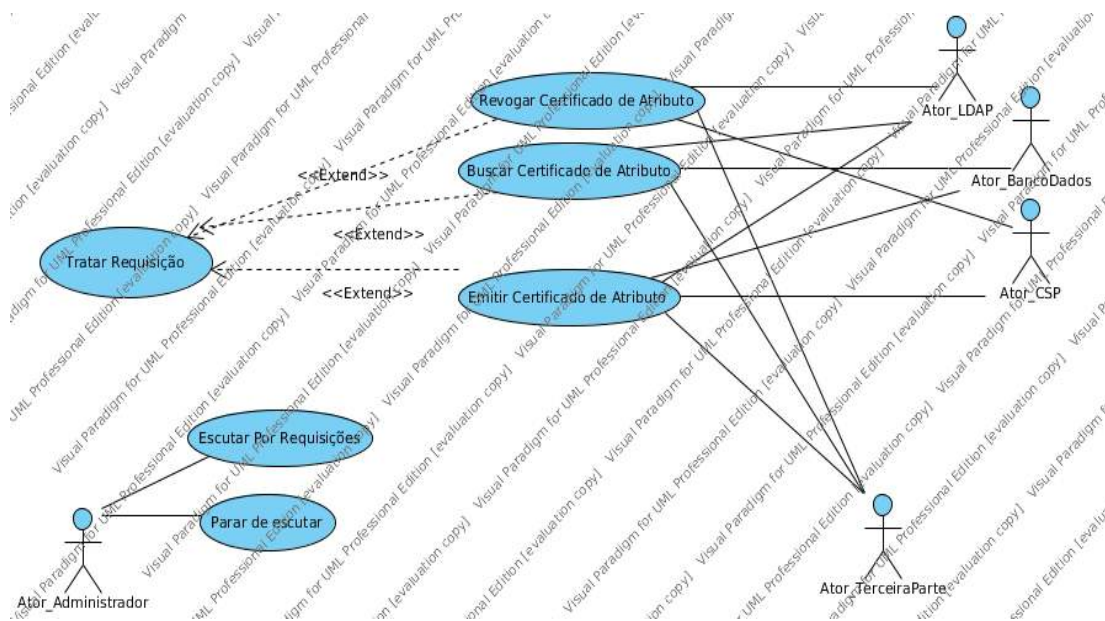
O diagrama de visão geral de interação mostra uma visão macro do fluxo das ocorrências dos casos de uso do sistema.



**Figura A.1:** Diagrama de visão geral de interação do AAFW.

## A.2 Diagrama de casos de uso

O diagrama de casos de uso mostra as funcionalidades do sistema - o que ele faz - e como interage com o meio externo.



**Figura A.2:** Diagrama de casos de uso do AAFW.

As funcionalidades do sistema são visíveis ao usuário externo (terceira parte) por meio do Ator\_TerceiraParte, são elas: Emitir Certificado de Atributo, Buscar Certificado de Atributo, Revogar Certificado de Atributo. Em tais casos podem ocorrer a participação de entidades externas como um HSM para realização de assinatura e banco de dados para persistência dos certificados de atributos.

O Ator\_TerceiraParte consegue se comunicar como o framework através do protocolo de transporte das mensagens de requisição e resposta por ele utilizado, como por exemplo o Baseado em Socket TCP 4.5.1

### A.3 Diagrama de classes

O diagrama de classes representa os elementos do domínio do problema, como explicado em 5.1. Os problemas foram mapeados para interfaces, que permitem que seja definido o que se espera que aconteça, porém sem fornecer a sua implementação. Implementação essa que será definida pela usuário do framework.

As interfaces e suas implementações embutidas no framework são expli-

cadras na tabela 5.1 O seu usuário pode escolher uma das implementações ou também definir suas próprias maneiras para a realização dessas tarefas.

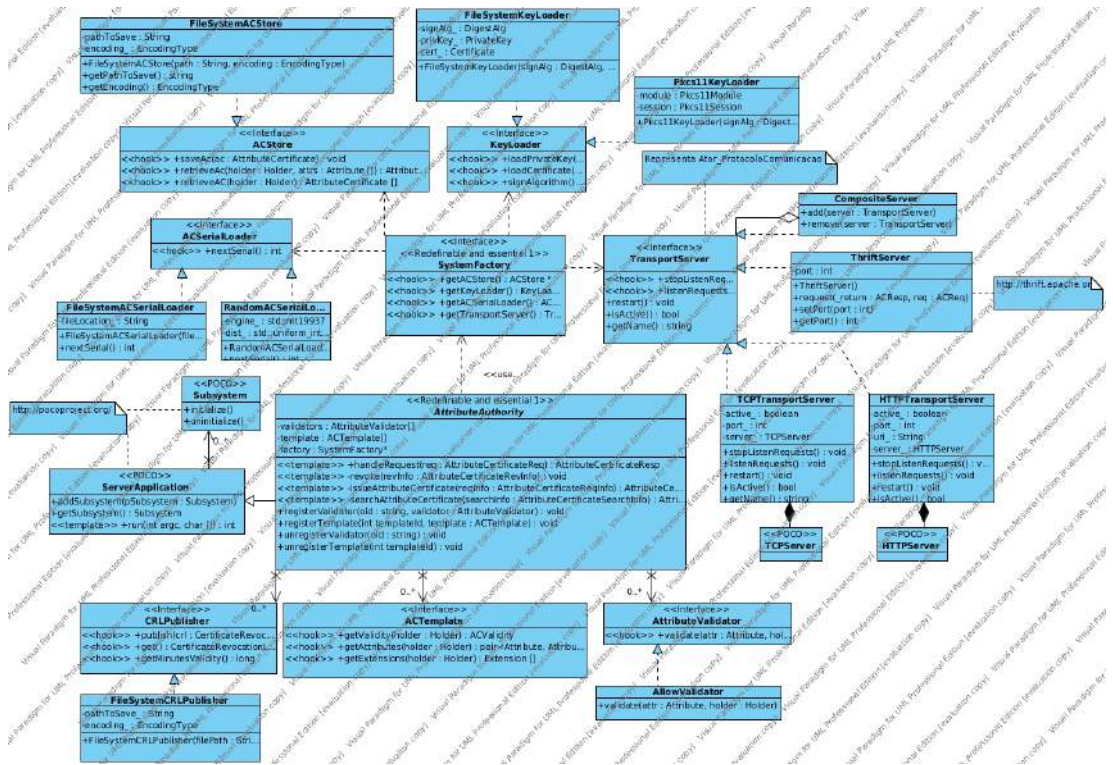


Figura A.3: Diagrama de classes simplificado do AAFW.

## A.4 Diagrama de atividades

O diagrama de atividades mostra o comportamento do programa - o que ele faz quando em execução.

### A.4.1 Tratar Requisição

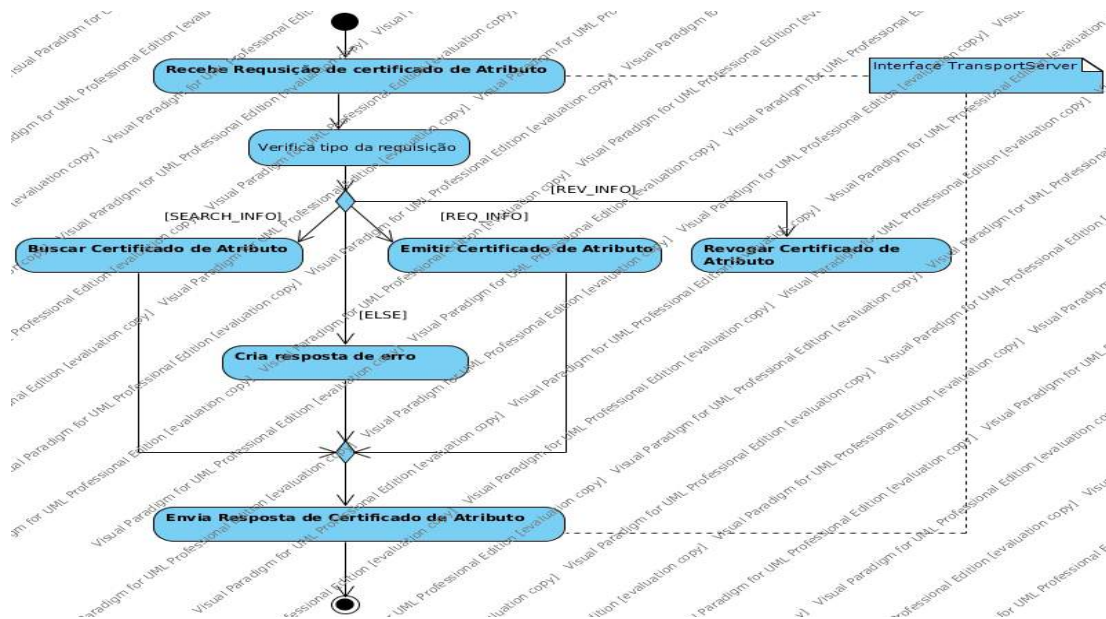


Figura A.4: Diagrama de atividades do AAFW: Tratar Requisição.

## A.4.2 Emitir Certificado de Atributo

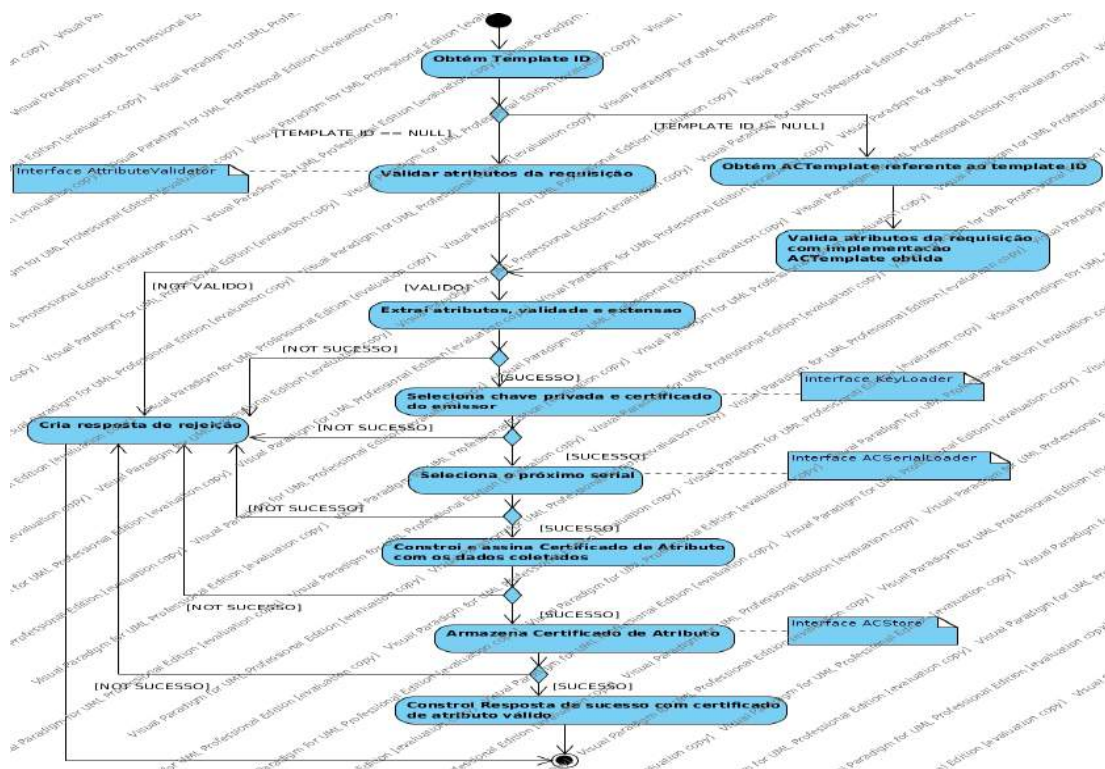


Figura A.5: Diagrama de atividades do AAFW: Emitir Certificado de Atributo.

### A.4.3 Buscar Certificado de Atributo



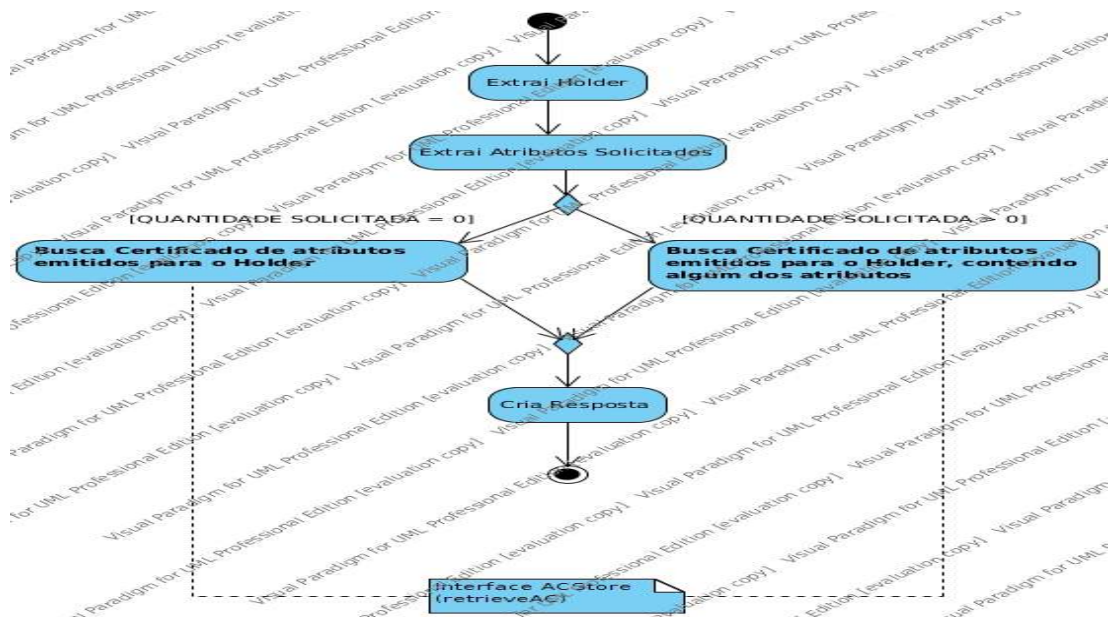


Figura A.6: Diagrama de atividades do AAFW: Buscar Certificado de Atributo.

#### A.4.4 Revogar Certificado de Atributo

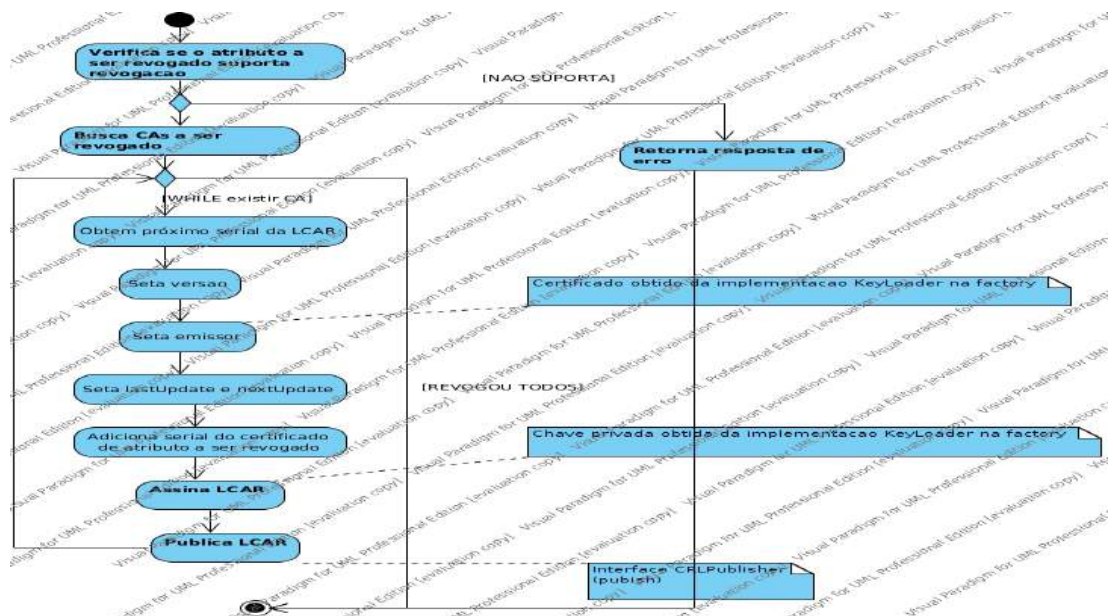


Figura A.7: Diagrama de atividades do AAFW: Revogar Certificado de Atributo.

## A.4.5 Escutar Por Requisições

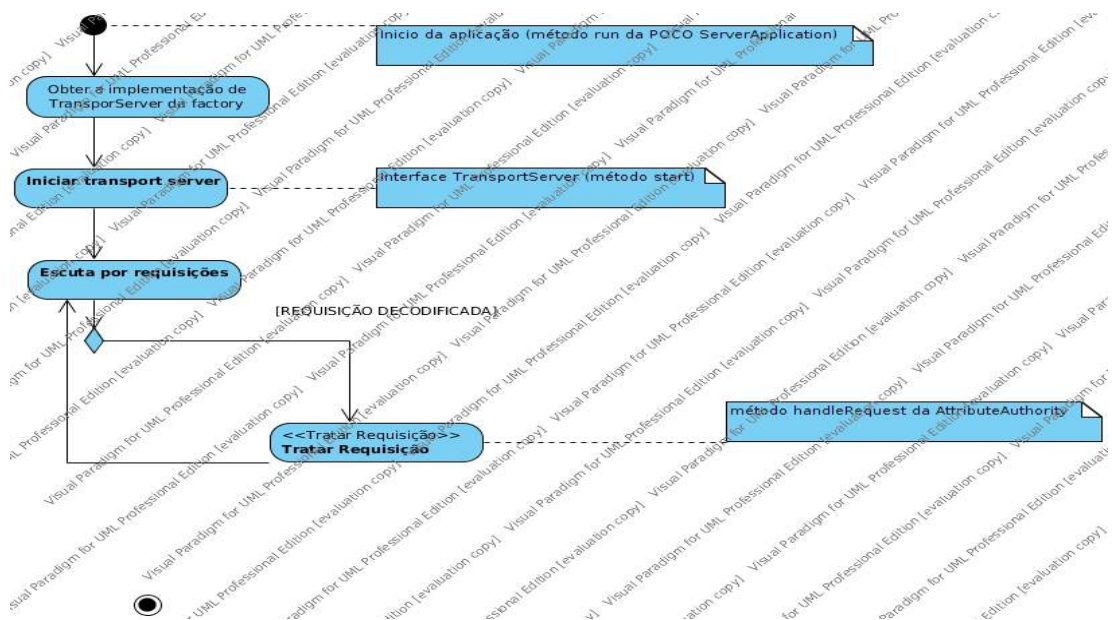


Figura A.8: Diagrama de atividades do AAFW: Escutar Por Requisições.

## A.4.6 Parar de escutar

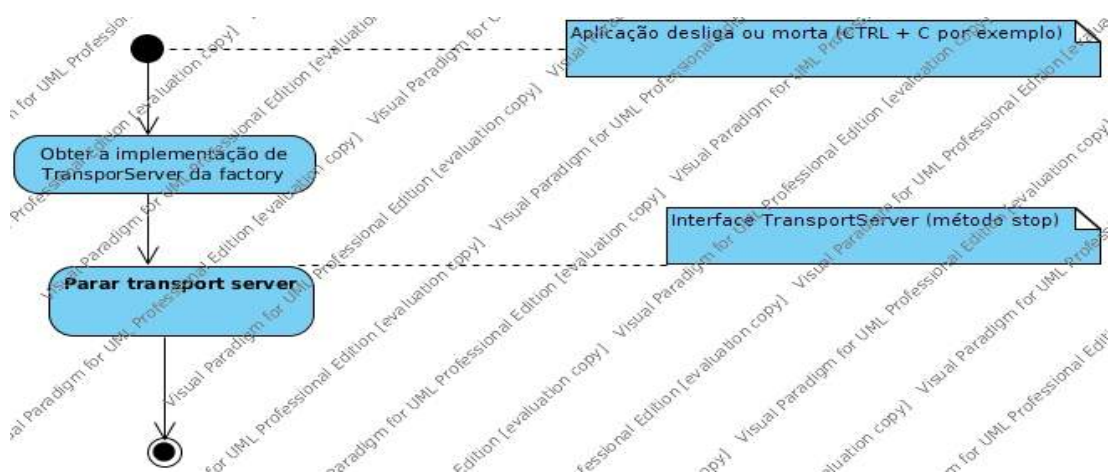


Figura A.9: Diagrama de atividades do AAFW: Parar de escutar.

## A.5 Diagrama de sequência

O diagrama de sequência também mostra o comportamento do programa, porém focado na interação entre os objetos do sistema.

### A.5.1 Tratar Requisição

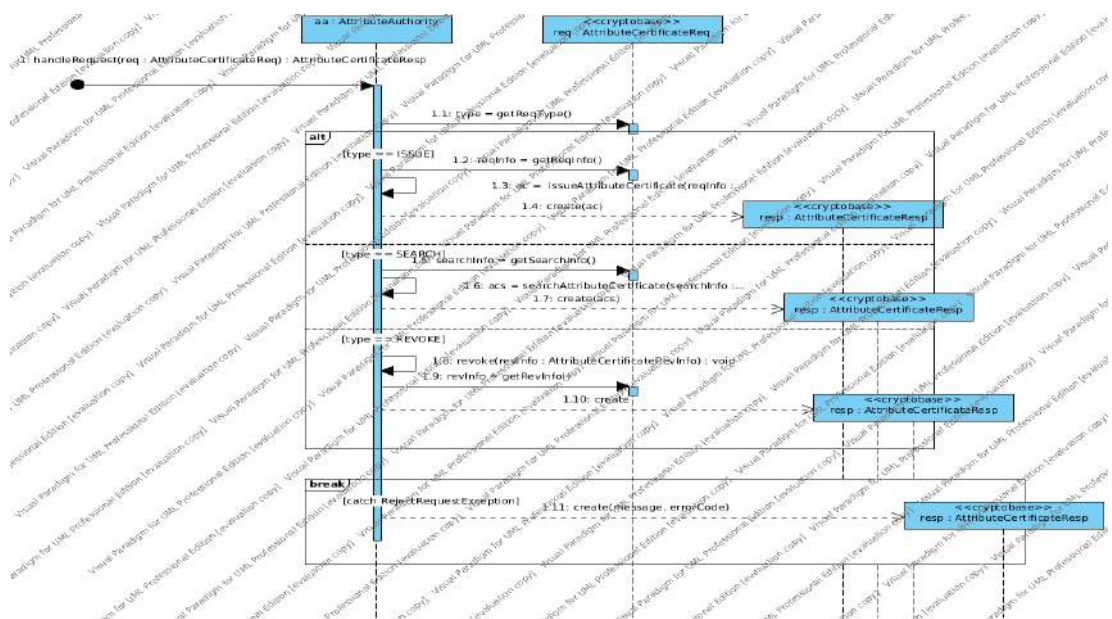


Figura A.10: Diagrama de sequência do AAFW: Tratar Requisição.

### A.5.2 Emitir Certificado de Atributo

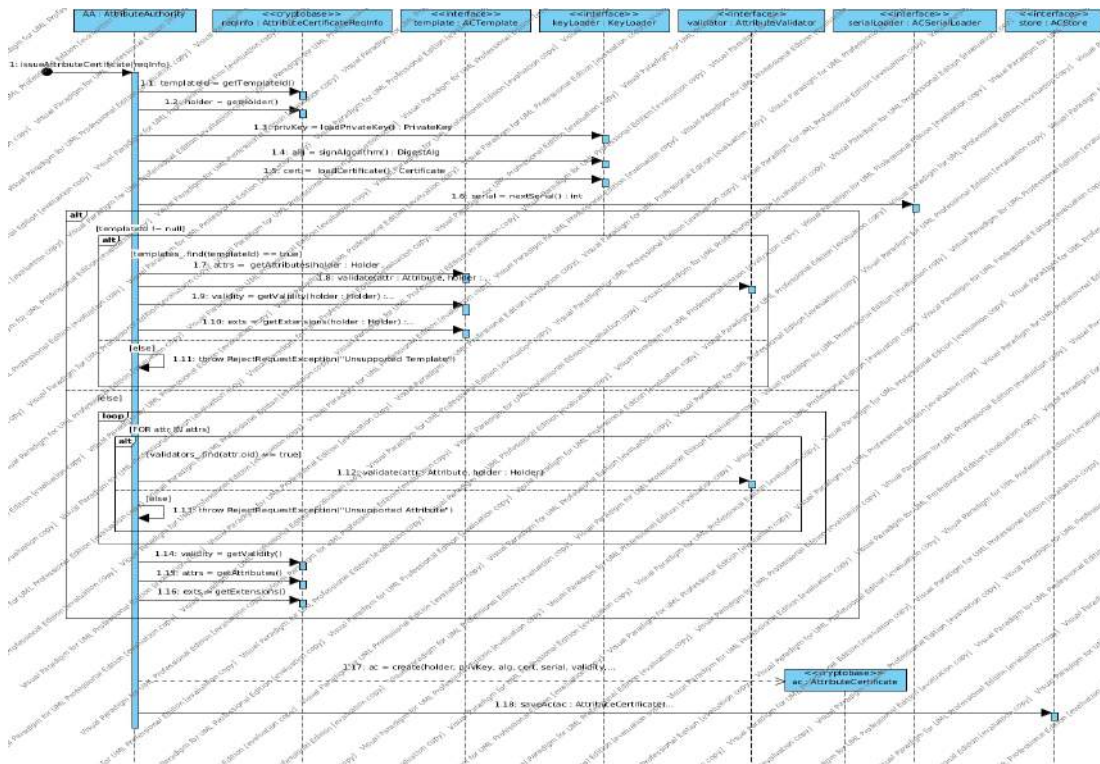


Figura A.11: Diagrama de sequência do AAFW: Emitir Certificado de Atributo.

### A.5.3 Buscar Certificado de Atributo

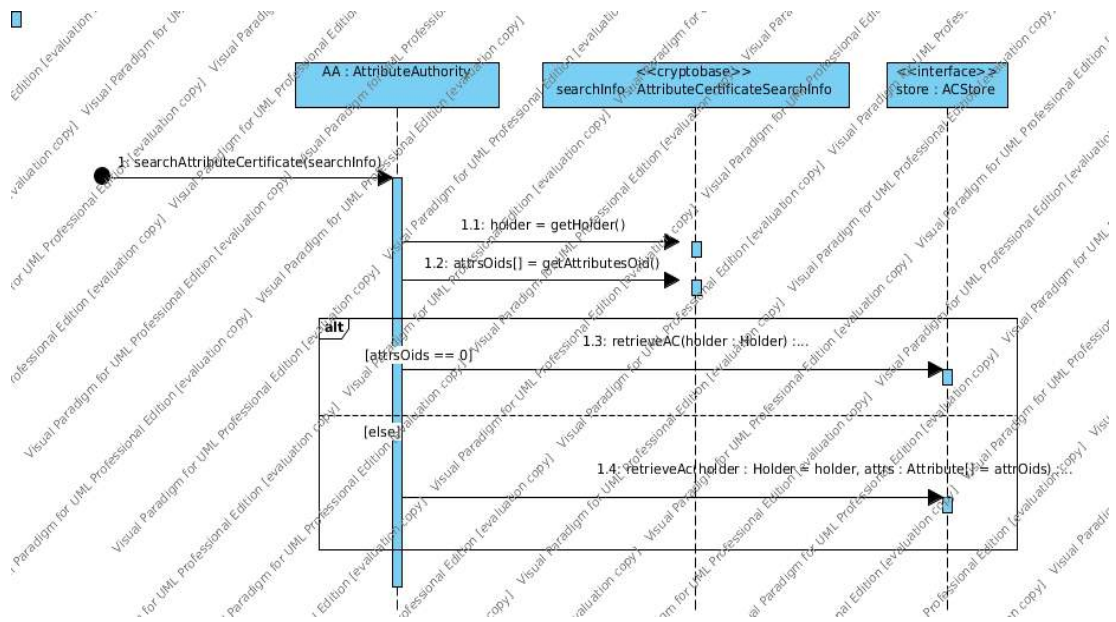


Figura A.12: Diagrama de sequência do AAFW: Buscar Certificado de Atributo.

### A.5.4 Revogar Certificado de Atributo

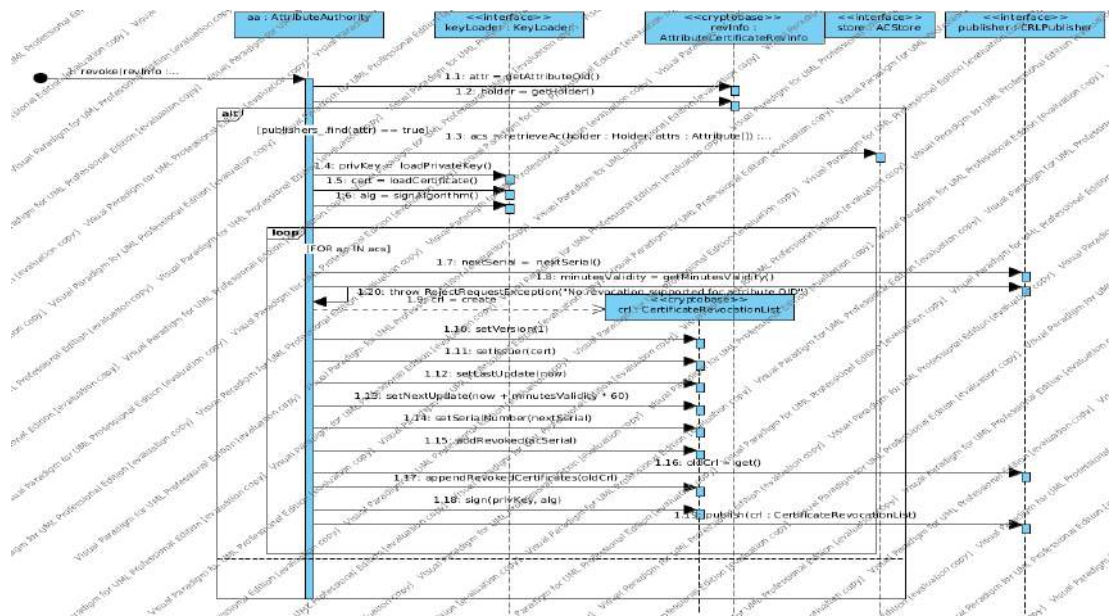


Figura A.13: Diagrama de sequência do AAFW: Revogar Certificado de Atributo.

### A.5.5 Escutar Por Requisições

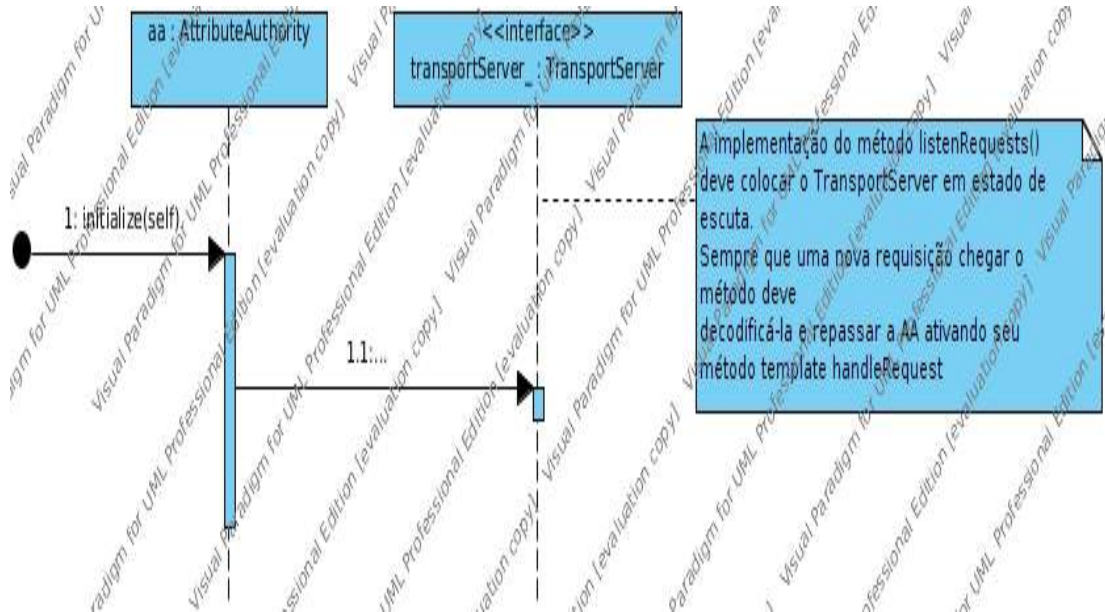


Figura A.14: Diagrama de sequência do AAFW: Escutar Por Requisições.

### A.5.6 Parar de escutar

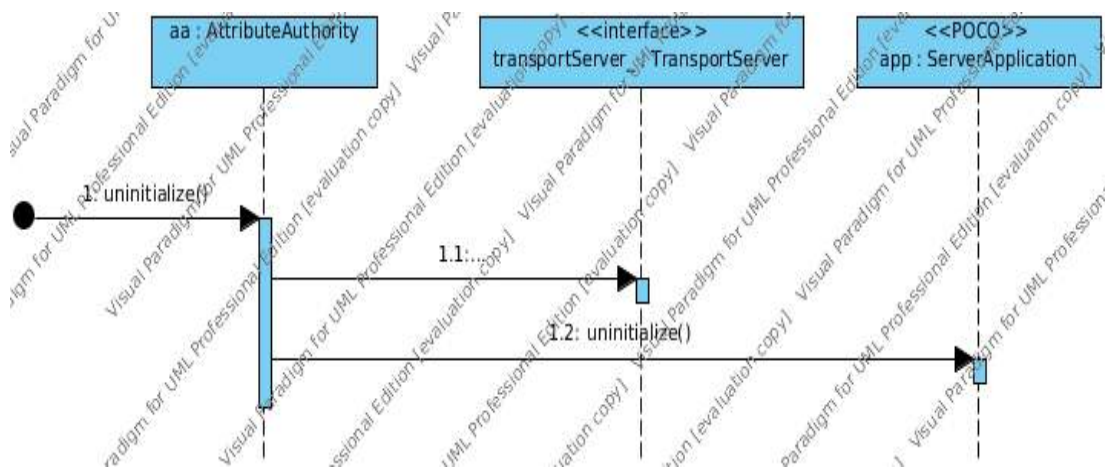


Figura A.15: Diagrama de sequência do AAFW: Parar de escutar.

# Apêndice B

## Exemplos Funcionais

Os exemplos aqui implementados foram testados e compilados utilizando g++ 4.7 (linux) e VS2012 (windows).

### B.1 EEA Básica

```
1 #include "aafw/AttributeAuthority.hpp"
2 #include "aafw/DefaultFactory.hpp"
3 #include "aafw/AllowValidator.hpp"
4 #include "aafw/FileSystemCRLPublisher.hpp"
5
6 #include <iostream>
7
8 using namespace aafw;
9 using namespace cryptobase;
10
11 class MinhaEEA : public AttributeAuthority
12 {
13 public:
14     std::unique_ptr<SystemFactory> getSystemFactory()
15     {
16         return std::unique_ptr<SystemFactory>(new DefaultFactory);
17     }
18 }
```

```

18 void setup()
19 {
20     registerValidator("2.30.50.1.1.1", new AllowValidator);
21     registerCRLPublisher("2.30.50.1.1.1", new ↵
        FileSystemCRLPublisher(60, "http://minhaEEA.com/eea.crl", ↵
            "C:\\eea.crl"));
22     denyUnknownAttributes();
23 }
24 };
25
26 int main(int argc, char ** argv)
27 {
28     try
29     {
30         MinhaEEA eea;
31         return eea.run(argc, argv);
32     }
33     catch (...)
34     {
35         std::cerr << "something bad happened" << std::endl;
36     }
37 }

```

## B.2 Implementando SystemFactory

```

1 #include "aafw/AttributeAuthority.hpp"
2 #include "aafw/AllowValidator.hpp"
3 #include "aafw/FileSystemACStore.hpp"
4 #include "aafw/Pkcs11KeyLoader.hpp"
5 #include "aafw/FileSystemACSerialLoader.hpp"
6 #include "aafw/TCPTransportServer.hpp"
7
8 #include <iostream>
9

```



```

10 using namespace aafw;
11 using namespace cryptobase;
12
13 class MinhaFactory : public SystemFactory
14 {
15 public:
16     std::unique_ptr<ACStore> getACStore() const
17     {
18         return std::unique_ptr<ACStore>(new FileSystemACStore("/home/←
19             /giovani/acs", EncodingType::DER));
20     }
21     std::unique_ptr<KeyLoader> getKeyLoader() const
22     {
23         return std::unique_ptr<KeyLoader>(new Pkcs11KeyLoader(←
24             DigestAlg::SHA1,
25             "/usr/lib/softhsm/libsofthsm.so", "Chave AA", "←
26             123456", "label1", "Certificado AA"));
27     }
28     std::unique_ptr<ACSerialLoader> getACSerialLoader() const
29     {
30         return std::unique_ptr<ACSerialLoader>(new ←
31             FileSystemACSerialLoader("/home/giovani/serial.txt"));
32     }
33     std::unique_ptr<TransportServer> getTransportServer() const
34     {
35         return std::unique_ptr<TransportServer>(new ←
36             TCPTransportServer(50005));
37     }
38 };
39
40 class MinhaEEA : public AttributeAuthority
41 {
42 public:
43     std::unique_ptr<SystemFactory> getSystemFactory()
44     {

```

```

40     return std::unique_ptr<SystemFactory>(new MinhaFactory);
41 }
42 void setup()
43 {
44     registerValidator("2.30.50.1.1.1", new AllowValidator);
45     denyUnknownAttributes();
46 }
47 };
48
49 int main(int argc, char ** argv)
50 {
51     try
52     {
53         MinhaEEA eea;
54         return eea.run(argc, argv);
55     }
56     catch (...)
57     {
58         std::cerr << "something bad happened" << std::endl;
59     }
60 }

```

### B.3 Implementando Validador de Atributo

```

1 #include "aafw/AttributeAuthority.hpp"
2 #include "aafw/DefaultFactory.hpp"
3 #include "aafw/Exception.hpp"
4
5 #include <iostream>
6
7 using namespace aafw;
8 using namespace cryptobase;
9
10 class IsTrustedValidator : public AttributeValidator

```

```

11 {
12 public:
13     IsTrustedValidator()
14     {
15         trustedNames_.push_back("Giovani Milanez Espindola");
16         trustedNames_.push_back("Jeandré Monteiro Sutil");
17         trustedNames_.push_back("Ruy Ramos");
18         trustedNames_.push_back("Ricardo Felipe Custódio");
19     }
20     void validate(const Attribute& attribute, const Holder& holder)←
21         const
22     {
23         if(holder.getType() != Holder::HolderType::ENTITY_NAME)
24             throw RejectRequestException("Requisição inválida, ←
25                 esperado tipo Autônomo!", ACStatusInfo::ACFailureInfo←
26                 ::badRequest);
27
28         std::string holderName = holder.getHolderEntityName().←
29             getEntry(ObjectIdentifier(NID_commonName));
30
31         auto it = std::find(trustedNames_.begin(), trustedNames_.end←
32             (), holderName);
33
34         if(it == trustedNames_.end())
35             throw RejectRequestException("Titular "+holderName+" não ←
36                 é confiável!", ACStatusInfo::ACFailureInfo::←
37                 untrustedHolder);
38     }
39 private:
40     std::vector<std::string> trustedNames_;
41 };
42 class MinhaEEA : public AttributeAuthority
43 {
44 public:
45     std::unique_ptr<SystemFactory> getSystemFactory()
46     {

```

```

39     return std::unique_ptr<SystemFactory>(new DefaultFactory);
40 }
41 void setup()
42 {
43     registerValidator("2.30.50.1.1.1", new IsTrustedValidator);
44     denyUnknownAttributes();
45 }
46 };
47
48 int main(int argc, char ** argv)
49 {
50     try
51     {
52         MinhaEEA eea;
53         return eea.run(argc, argv);
54     }
55     catch (...)
56     {
57         std::cerr << "something bad happened" << std::endl;
58     }
59 }

```

## B.4 Implementando Template

```

1 #include "aafw/AttributeAuthority.hpp"
2 #include "aafw/DefaultFactory.hpp"
3 #include "aafw/AllowValidator.hpp"
4
5 #include <iostream>
6
7 using namespace aafw;
8 using namespace cryptobase;
9
10 class MyTemplate : public ATemplate

```

```

11 {
12 public:
13     AttributeCertificateValidity getValidity(const Holder& holder) ←
14         const
15     {
16         return AttributeCertificateValidity(60 * 24);
17     }
18     std::vector<std::pair<Attribute, std::unique_ptr<←
19         AttributeValidator>>> getAttributes() const
20     {
21         std::vector<std::pair<Attribute, std::unique_ptr<←
22             AttributeValidator>>> attrs;
23         attrs.push_back(
24             std::make_pair(Attribute(ObjectIdentifier("2.30.50.1.1.2"←
25                 ), "Aluno regular"), std::unique_ptr<←
26                 AttributeValidator>(new AllowValidator)));
27         attrs.push_back(
28             std::make_pair(Attribute(ObjectIdentifier("2.30.50.1.1.3"←
29                 ), "Sistemas de Informação"), std::unique_ptr<←
30                 AttributeValidator>(new AllowValidator)));
31     }
32     return attrs;
33 }
34 std::vector<Extension> getExtensions(const Holder& holder) ←
35     const
36 {
37     return std::vector<Extension>();
38 }
39 };
40 class MinhaEEA : public AttributeAuthority
41 {
42 public:
43     std::unique_ptr<SystemFactory> getSystemFactory()
44     {
45         return std::unique_ptr<SystemFactory>(new DefaultFactory);

```

```

38     }
39     void setup()
40     {
41         registerAttributeCertificateTemplate(10, new MyTemplate);
42         denyUnknownAttributes();
43     }
44 };
45
46 int main(int argc, char ** argv)
47 {
48     try
49     {
50         MinhaEEA eea;
51         return eea.run(argc, argv);
52     }
53     catch (...)
54     {
55         std::cerr << "something bad happened" << std::endl;
56     }
57 }

```

## B.5 Requisitando busca CAV via TCP

```

1 #include "cryptobase/AttributeCertificateSearchInfo.hpp"
2 #include "cryptobase/AttributeCertificate.hpp"
3 #include "cryptobase/Certificate.hpp"
4
5 #include "TcpAAClient.hpp"
6
7 #include <iostream>
8
9 using namespace cryptobase;
10
11 int main(int argc, char ** argv)

```

```

12 {
13     try
14     {
15         Certificate cert = Certificate::fromFile("C:\\giovani.crt");
16         Holder h(cert);
17         AttributeCertificateSearchInfo searchInfo(h);
18         ObjectIdentifier obj("2.30.50.1.1.1");
19         searchInfo.setAttributesOid(obj);
20
21         AttributeCertificateReq req(searchInfo);
22         TcpAAClient client("localhost", 50005);
23         AttributeCertificateResp resp = client.send(req);
24         if(resp.granted())
25         {
26             for(auto ac : resp.getAcs())
27             {
28                 std::cout << "Encontrado " << ac.getPemEncoded() << ←
29                 std::endl;
30             }
31         }
32         else
33         {
34             std::cerr << resp.getStatusInfo().getText() << std::endl;
35         }
36     }
37     catch (...)
38     {
39         std::cerr << "something bad happened" << std::endl;
40     }

```

## B.6 Requisitando emissão via PHP com Thrift

```
<?php
```

```
namespace aafw\php;

error_reporting(E_ALL);

require_once 'C:\\Desenvolvimento\\thrift-0.9.1\\lib\\php\\↵
    lib\\Thrift\\ClassLoader\\ThriftClassLoader.php';

use Thrift\\ClassLoader\\ThriftClassLoader;

$GEN_DIR = 'C:\\Users\\Gica\\Desktop\\gen-php';

$loader = new ThriftClassLoader();
$loader->registerNamespace('Thrift', 'C:\\Desenvolvimento\\↵
    thrift-0.9.1\\lib\\php\\lib');
$loader->registerDefinition('aafw', $GEN_DIR);
$loader->register();

use Thrift\\Protocol\\TBinaryProtocol;
use Thrift\\Transport\\TSocket;
use Thrift\\Transport\\THttpClient;
use Thrift\\Transport\\TBufferedTransport;
use Thrift\\Exception\\TException;

try {
    $socket = new TSocket('localhost', 9090);
    $transport = new TBufferedTransport($socket, 1024, 1024);
    $protocol = new TBinaryProtocol($transport);
    $client = new \\aafw\\AAServiceClient($protocol);

    $transport->open();
```



```

$entityName["2.5.4.3"] = "Maria da Silva"; // Common Name
$holder = new \aafw\ACHolder(array("entityName" => $entityName));

$validity = new \aafw\ACValidity(array("notBeforeEpoch" => time(), "notAfterEpoch" => time() + 60 * 60 * 24 * 100)); // 100 days validity from now
$attributes[] = new \aafw\Attribute(array("oid" => "2.30.50.1.1.1", "values" => array("Cinema 3D - A Ilha. SALA 5")));
$reqInfo = new \aafw\ACIssueInfo(array("holder" => $holder, "attributes" => $attributes, "validity" => $validity));
$req = new \aafw\ACReq(array("issueInfo" => $reqInfo));

$resp = $client->request($req);

var_dump($resp);

$transport->close();

} catch (Exception $tx) {
    print 'Exception: ' . $tx->getMessage() . "\n";
}
?>

```

# **Apêndice C**

## **Artigo**

# Um Framework Orientado a Objetos para Gestão de Certificados de Atributos nos Padrões da ICP-Brasil

Giovani M. Espindola<sup>1</sup>

<sup>1</sup>Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

**Abstract.** *The X.509 attribute certificate is a digitally signed document that establish assignments to a entity. In July 2012, the technology was standardized for use in Brazil by ICP-Brasil. Because it is a relatively new technology in that country, the development support for systems that uses attribute certificate still lack of studies and references. The author proposes a request-response protocol for attribute certificate issuance, search and revocation. Such protocol was implemented in a object oriented framework that allows the construction of applications that need to manage the life cycle of that certificates. The framework was used to build a prototype application for cinema ticket management.*

**Resumo.** *O certificado de atributo X.509 é um documento eletrônico assinado digitalmente que estabelece atribuições a uma entidade. Em julho de 2012, a tecnologia foi padronizada para uso no Brasil pela ICP-Brasil. Por ser uma tecnologia relativamente nova no país, o suporte para desenvolvimento de sistemas que utilizem certificado de atributo X.509 ainda carece de estudos e referências de implementação. O autor propõe um protocolo de requisição e resposta para emissão, busca e revogação desses certificados. Tal protocolo foi implementado na forma de um framework orientado a objetos que permite a construção de aplicações que gerenciem o ciclo de vida de certificados de atributos. Este framework foi utilizado na construção de uma aplicação protótipo, para gestão de bilhetagem de sessões de cinema.*

## 1. Introdução

O certificado digital, ou CD, é um documento que identifica uma entidade no meio eletrônico, atuando como um documento de identidade digital [Cooper et al. 2008]. Ele estabelece a ligação dessa pessoa ou organização com um par de chaves criptográficas, pública e privada, que passam a representá-la nas relações em meio virtual. A operação mais evidente do uso dos CD é a assinatura digital [Pinkas et al. 2008]. Muitas aplicações utilizam o CD também nos processos de autenticação de usuários, através da tecnologia SSL [Freier et al. 2011]. Este cenário de uso é uma alternativa que agrega segurança aos métodos tradicionais, como o uso de login e senha.

Nos cenários de uso atual, são muitas vezes adicionadas ao certificado digital informações que não só identificam o usuário, como também o qualificam. Um exemplo claro é o da qualificação profissional do titular do certificado, como o número de registro de um advogado na Ordem dos Advogados do Brasil (OAB). O principal problema dessa abordagem é que a vida útil de cada atributo pode ser diferente. Quanto mais atributos são concentrados em um mesmo certificado, maior as chances desse documento precisar ser reemitido, por conta de uma mudança em um dos atributos ou mesmo da inclusão de

outros. Outra deficiência desse modelo advém do fato de que nem sempre a entidade responsável por identificar o usuário e emitir o certificado digital também ser a responsável pela emissão dos atributos que o qualificam. Para tratar problemas como os expostos acima, surgiu o Certificado de Atributos.

Apesar de já estarem definidos um formato e um perfil de uso, os normativos da ICP-Brasil ou mesmo os internacionais não especificam claramente como uma entidade pode requisitar a emissão de um certificado de atributos a uma Entidade Emissora de Certificado de Atributos (EEA) e nem os detalhes da comunicação entre ambas.

Buscando preencher essas lacunas na adoção da tecnologia, este trabalho apresenta uma proposta de protocolo que suporte os processos de emissão, busca e revogação de certificados de atributos, como também sugere estruturas que comportarão essas trocas de mensagens.

O trabalho propõe também um framework baseado no protocolo proposto, para desenvolvimento de aplicações que necessitem gerenciar o ciclo de vida desses certificados. O framework poderá ser utilizado por desenvolvedores de aplicações que possam se beneficiar com a tecnologia de certificação de atributos, como EEAs.

## **2. Conceitos Básicos**

### **2.1. Resumo Criptográfico**

A função de resumo criptográfico, também conhecida como função hash, é um procedimento matemático que visa transformar um dado de tamanho arbitrário em um outro, de tamanho reduzido e fixo. Uma característica importante de uma função de hash é que, a partir de uma mesma informação, será produzido sempre o mesmo resultado. Daí o nome de resumo criptográfico, pois o valor resumido identifica a informação de entrada da função. Outra característica fundamental para uma boa função de hash é que, a mudança de um único bit em sua entrada, produza uma saída completamente distinta. Dadas essas propriedades, a função hash é frequentemente utilizada na garantia de integridade sobre os dados, onde envia-se a um destinatário a informação, acompanhada de seu resumo. A parte receptora processa a informação recebida utilizando a mesma função de hash e compara o resultado com o resumo recebido. Alguns algoritmos de hash utilizados são SHA-1, SHA-256 e SHA-512.

### **2.2. Criptografia Assimétrica**

A criptografia assimétrica consiste em um processo de transformação de textos claros em textos aparentemente sem sentido. É uma técnica muito utilizada para conferir sigilo e autenticidade a informações transmitidas em meio eletrônico. Esse processamento baseia-se no conhecimento de um segredo, conhecido como chave criptográfica. Somente as partes que conhecem essa chave são capazes de reverter o processo chamado de cifragem [Choudhury et al. 2002]. A criptografia assimétrica é aquela em que há não uma, mais sim duas chaves envolvidas na proteção da informação, o par de chaves. Uma dessas chaves é pública e distribuída livremente. A segunda deve ser mantida em sigilo (chave privada).

### **2.3. Certificado Digital**

O certificado digital é um documento eletrônico que identifica uma entidade, pois ele fornece a ligação entre ela e seu par de chaves criptográficas (pública e privada). Através

dessa ligação é possível a duas entidades que nunca foram apresentadas pessoalmente, identificar-se em meio eletrônico com a segurança de que são realmente quem dizem ser.

## **2.4. Assinatura Digital**

A assinatura digital é uma das principais aplicações do uso de criptografia assimétrica em conjunto com a certificação digital. Ela consiste no processo de cifragem de um resumo criptográfico que identifica um determinado documento ou mensagem, utilizando a chave privada da entidade assinante. A parte recebedora da informação poderá verificar se a assinatura (resumo assinado), foi produzida pelo detentor da chave privada, revertendo o processo com a chave pública do mesmo. Calculando um novo resumo sobre o documento recebido e comparando-o com o dado decifrado, o recebedor poderá verificar a autoria e autenticidade da informação. [Choudhury et al. 2002]

## **2.5. Infraestrutura de Chaves Públicas**

O elemento básico de uma infraestrutura de chaves públicas (ICP), é o certificado digital. Uma ICP é formada, por recursos humanos, físicos e procedimentais necessários para o gerenciamento de certificados digitais.

Sua função, de acordo com [Housley and Polk 2001], é a de facilitar o uso de criptografia assimétrica através da emissão de certificados digitais e lista de certificados revogados. O modelo mais empregado de ICP consiste no estabelecimento de uma cadeia hierárquica de confiança. No topo dessa estrutura está a âncora de confiança, a Autoridade Certificadora Raiz, ou AC Raiz, que emite certificados digitais para ACs intermediárias e as chamadas ACs finais, responsáveis por emitir certificados para as entidades finais.

### **2.5.1. Autoridade Certificadora**

Uma autoridade certificadora (AC) também é formada por recursos humanos, físicos e procedimentais. A sua principal tarefa é a de emitir certificados digitais. Ela faz isso preenchendo os campos do certificado digital 2.3 a ser emitido e depois os assina com sua chave privada. A autoridade contudo, só pode realizar a emissão do certificado se as informações a serem preenchidas estiverem corretas. Para isso ela conta com a Autoridade de Registro 2.5.2.

### **2.5.2. Autoridade de Registro**

É a entidade responsável pela interação entre a autoridade certificadora e os usuários finais. A autoridade de registro (AR) tem o papel de confirmar os dados do usuário para constar no seu certificado a ser emitido. Um dos motivos para existência da AR é de que alguns dados do usuário são bastante específicos, como o número de registro do profissional (OAB, CRM) e requer a comunicação com entidades externas para essa validação.

### **2.5.3. Lista de Certificados Revogados**

A lista de certificados revogados é um documento assinado emitido por autoridades certificadoras que contém informações sobre os certificados que ela revogou. Essas informações

incluem o número serial do certificado revogado, a data de revogação e um motivo. A revogação pode acontecer por roubo da chave privada do titular, por desistência por parte do titular de fazer seu uso ou pela perda de algum privilégio contido no seu certificado digital como seu número OAB, no caso de advogados.

## **2.6. ICP-Brasil**

Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil) é uma ICP instituída pelo governo brasileiro pela Medida Provisória 2200 [BRASIL ].

## **2.7. Framework Orientado a Objeto**

*Um framework consiste em um conjunto de classes, interfaces e padrões dedicados a solucionar um grupo de problemas através de uma arquitetura de programação flexível e extensível.* [Govoni 1999]

A demanda da indústria de software de produzir cada vez mais softwares complexos com menor tempo de desenvolvimento, é conhecida como paradoxo da indústria de software (Silva).

Uma das formas de alcançar esse objetivo é com o reuso de código, que pode ser feito com a implementação de artefatos de softwares reutilizáveis, como uma biblioteca de classes para manipulação de arquivos.

Um framework orientado a objetos consiste em uma outra forma de reuso de software, classificado como reuso de projeto. Ele é projetado para generalizar um domínio específico de aplicações, de maneira que novas possam ser construídas sobre o framework, com elevado grau de reuso. Isso é possível pois ele é composto por um conjunto de classes, sendo que algumas estão com sua implementação inacabada, bastando apenas a aplicação consumidora fornecer as partes faltantes, que a distingam das demais. Essas partes faltantes são chamadas de *hotspots* e representam os pontos de flexibilização do framework.

O framework orientado a objetos confina as aplicações consumidoras à sua arquitetura de classes e fluxo de execução de métodos. Apesar disso, é possível acrescentar ou alterar suas funcionalidades, uma vez que o framework é construído sobre uma série de padrões de projetos voltados justamente ao reuso e especialização de suas funcionalidades.

## **3. Certificação de Atributos**

### **3.1. Certificado Digital x Certificado de Atributo**

O documento RFC 5755 [Farrell et al. 2010] propõe o perfil de certificado de atributos a ser utilizado pelos padrões da internet.

O certificado digital identifica seu titular enquanto que o CA lhe confere qualificações, como por exemplo, no caso de um visto para o exterior, se o usuário poderá trabalhar, apenas estudar ou exercer outras atividades durante sua validade. Pode-se concluir então que um certificado digital possui atributos estáticos, de longa duração, sempre relacionados à identidade do titular. Para sua emissão, o titular deve se submeter a um criterioso processo de identificação, que muitas vezes exige uma validação presencial da

identidade do indivíduo. O certificado de atributos, por sua vez, agrega ao primeiro propriedades dinâmicas, que podem ter menor duração e podem ser substituídas, removidas ou incluídas sem a rigidez com que ocorre uma emissão de CD.

### 3.2. Estrutura de um Certificado de Atributo

A definição ASN.1 do CA é apresentada abaixo:

```
AttributeCertificate ::= SEQUENCE {
    acinfo             AttributeCertificateInfo,
    signatureAlgorithm AlgorithmIdentifier,
    signatureValue     BIT STRING
}
```

O campo **acinfo** contém as informações sobre o certificado de atributo. Essas informações serão assinadas, o campo **signatureValue** conterá o valor da assinatura e **signatureAlgorithm** o identificador do algoritmo utilizado.

A estrutura **acinfo** representa as informações contidas no certificado de atributo. Os campos mais importantes são **holder** que identifica o titular do CA, seja ele uma pessoa física, jurídica ou mesmo um equipamento. Também está contido nessa estrutura no campo **issuer**, informações que permitem identificar quem emitiu o CA. O campo **attributes** contém os atributos associados ao titular do CA, ou seja, os benefícios que se deseja lhe conceder. Outro campo importante é **attrCertValidityPeriod** que delimita a janela de validade do CA.

### 3.3. Entidade Emissora de Certificado de Atributo

Uma Entidade Emissora de Atributos (EEA) é sinônimo do termo Autoridade de Atributos (AA). Segundo a RFC 5755 [Farrell et al. 2010] ela é qualquer entidade que emita (assina) certificados de atributos. De acordo com a ICP-Brasil, ela pode ser qualquer pessoa jurídica detentora da prerrogativa legal para emissão de determinado atributo [ITI b]. Tal entidade deve possuir, para a assinatura dos certificados de atributos, um certificado digital ICP-Brasil emitido para Pessoas Jurídicas, do tipo A3 ou A4.

### 3.4. Aplicação verificadora

Para se considerar um certificado de atributo como válido, algumas considerações devem ser feitas. No caso do titular estar vinculado ao seu CD, ele deve ser achado pelo verificador e toda sua cadeia de certificação deve ser verificada. A assinatura do CA deve estar criptograficamente correta. A cadeia de certificação do emissor do CA também deve ser verificada. O emissor deve ser confiável pelo verificador. No momento da verificação, a hora deve estar entre o período de validade do CA.

Outra entidade envolvida no ciclo de vida do certificado de atributo são as aplicações verificadoras. São as aplicações que necessitam de um certificado de atributo válido a fim de permitir seu uso.

### 3.5. Ciclo de vida de um Certificado de Atributo

O ciclo de vida do certificado de atributo começa com sua solicitação para EEA por uma entidade interessada. A solicitação é processada e será verificado a situação do titular a

fim de aprovar sua emissão. Após isso, o CA será emitido. Sua publicação utilizando um dos métodos *push* ou *pull* é necessária para que ele possa ser consumido. Uma vez publicado, o seu titular pode gozar dos benefícios por ele trazidos, até que o certificado expire. Caso ela desejar obter novamente seus privilégios, pode ser solicitada sua renovação, o que implica em novo processo de aprovação, emissão e assim sucessivamente. Ao longo de sua validade ele pode vir a ser revogado devido à perda dos referidos atributos, por exemplo, passando a constar na lista de certificados de atributos revogados. O ciclo termina quando seu prazo de validade expira.

#### **4. Protocolo para Gerenciamento de Certificados de Atributo**

A RFC 5755 define o formato do certificado de atributo e apresentou dois modelos para sua distribuição, *push* e *pull*, porém o documento abstrai os detalhes técnicos relativos à publicação do certificado. Caso uma EEA deseje realizar a distribuição pelo modelo *pull* utilizando como repositório um diretório de rede, as aplicações interessadas em obter o certificado de atributo devem conhecer o repositório, as formas de autenticação, sua organização, entre outros. Nota-se que esta forma seria uma especificidade dessa EEA. Outras poderão implementar a distribuição de outra maneira, o que implica em uma falta de padronização dos protocolos de acesso. Outro ponto que fica a cargo dos implementadores é a forma como serão requisitados, consultados e revogados os certificados.

O protocolo proposto visa resolver essas questões, uma vez que define um conjunto de estruturas, na forma de requisição e resposta, a serem trocadas entre solicitante e EEA na emissão, revogação ou busca de certificado de atributo.

A requisição é enviada por uma parte interessada em realizar uma dessas três operações e deve ser processada e respondida pela entidade emissora de atributo.

##### **4.1. Premissas do Protocolo**

Para o desenvolvimento do protocolo proposto, foram seguidas as seguintes premissas:

##### **Gerais**

- permitir o suporte à gestão de mais de uma EEA pelo mesmo servidor
- suportar os métodos *push* e *pull*, conforme definido na RFC 5755
- ser aderente aos normativos da ICP-Brasil

##### **Emissão de Certificados de Atributos**

- permitir a solicitação de um certificado para um atributo e titular específico
- permitir que a EEA decida quais os atributos podem ou não ser emitidos
- permitir à EEA negar a emissão de um certificado de atributos
- permitir a solicitação de um certificado de atributos com base em um modelo pré-definido
- permitir a emissão de certificados de atributos vinculados e não vinculados, conforme estabelecido pela ICP-Brasil

##### **Busca por Atributos**

- buscar por um ou mais atributos de um titular
- buscar por atributos emitidos por uma EEA específica



## Revogação de Certificados de Atributos

- permitir revogação de um certificado específico
- permitir revogação de um ou mais atributos específicos
- permitir a revogação de todos os atributos de um titular
- permitir a revogação de atributos emitidos por uma EEA específica

### 4.2. Requisição

O protocolo proposto inicia com o preenchimento de uma requisição, por parte do solicitante do certificado de atributos, a ser submetida à EEA. Essa requisição encontra-se definida na estrutura *AttributeCertificateReq*.

#### 4.2.1. AttributeCertificateReq

Uma requisição pode conter uma solicitação de emissão 4.2.2, revogação 4.2.3 ou busca 4.2.4 de certificado de atributo, conforme mostrado na estrutura apresentada a seguir.

```
AttributeCertificateReq ::= SEQUENCE {
    version                INTEGER { v1(1) },
    reqType                ACRReqType,
    reqInfo                AttributeCertificateReqInfo
}
```

A estrutura *AttributeCertificateReq* contém uma versão, um tipo e as respectivas informações que detalham-na. Os tipos de requisição são os definidos em *ACRReqType*, ou seja, valores inteiros representando emissão, busca ou revogação.

```
ACRReqType ::= ENUMERATED {
    issue                (1),
    -- Used in issuing requests
    revoke                (2),
    -- Used in revocation requests
    search                (3)
    -- Used in search requests
}
```

De acordo com o tipo de requisição definido, o campo *reqInfo* deve ser preenchido em conforme, com uma das estruturas previstas em *AttributeCertificateReqInfo*.

```
AttributeCertificateReqInfo ::= CHOICE {
    issueInfo            AttributeCertificateIssueReq,
    revInfo              AttributeCertificateRevReq,
    searchInfo           AttributeCertificateSearchReq
}
```

As seções 4.2.2 a 4.2.4 detalham as estruturas de cada um dos três tipos de mensagem previstos no protocolo.

#### 4.2.2. AttributeCertificateIssueInfo

A estrutura `AttributeCertificateIssueInfo` comporta as informações necessárias para que a EEA emita um certificado de atributo.

```
AttributeCertificateIssueReq ::= SEQUENCE {
    templateId          INTEGER OPTIONAL,
    -- if templateId is supplied then the validityPeriod,
    -- attributes and extensions fields must not be present.
    -- Those values will be automatically provided by the AA,
    -- based on templateId.
    holder              Holder,
    issuer              AttCertIssuer OPTIONAL,
    validityPeriod     AttrCertValidityPeriod OPTIONAL,
    attributes         SEQUENCE OF Attribute OPTIONAL,
    extensions         Extensions OPTIONAL
}
```

A informação mínima que deve ser passada a uma EEA é a identificação do titular que receberá o certificado de atributos. Essa informação deve ser fornecida no campo `holder`, que representa a estrutura `Holder` definida na RFC 5755. Essa informação pode ser tanto uma referência a um certificado digital, quanto um nome qualquer, sem um vínculo específico com um CD. Estas formas distintas são as que diferenciam um Certificado de Atributos Vinculado de um Autônomo na ICP-Brasil, conforme visto no capítulo 3.

Há basicamente duas formas de solicitar a emissão de um certificado de atributos. A primeira delas é identificar na requisição um modelo pré estabelecido pela EEA, através do campo `templateId`. Essa forma é indicada para EEAs que emitam certificados de atributos que seguem um modelo específico, como por exemplo uma carteira de motorista digital emitida pelo órgão de trânsito. Embora haja diferenças tais como as categorias a que o motorista está habilitado, a carteira em si segue um formato específico. Em outras palavras, muda-se os valores dos atributos, mas há um conjunto fixo dessas estruturas presentes no certificado, que consistem no seu modelo comum.

A segunda forma de solicitar um certificado de atributos consiste na definição desse modelo por parte do próprio solicitante.

#### 4.2.3. AttributeCertificateRevInfo

Essa estrutura transporta informações relacionadas a um pedido de revogação de um ou mais certificados de atributos emitidos por uma EEA.

A exemplo da emissão, há também na revogação duas formas de solicitar a invalidação de um certificado de atributos. A primeira é informando um conjunto de identificadores únicos para os certificados. A segunda busca revogar o atributo em si, não apenas um certificado. A estrutura de uma requisição de revogação, `AttributeCertificateRevReq`, é apresentada a seguir.

```
AttributeCertificateRevReq ::= CHOICE {
```

```

    attrCerts          SEQUENCE OF IssuerSerial,
    attrRevInfos       SEQUENCE OF AttributeRevocationInfo
}

AttributeRevocationInfo ::= SEQUENCE {
    holder             Holder,
    attributes         SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
    caIssuer           AttCertIssuer OPTIONAL
}

```

#### 4.2.4. AttributeCertificateSearchInfo

A estrutura `AttributeCertificateSearchInfo` concentra informações utilizadas na busca por certificados de atributo no repositório da EEA. A busca pode ser por todos os certificados de atributos emitidos para um mesmo titular, caso em que somente o campo `holder` deve se preenchido. Um solicitante pode também buscar por um conjunto específico de atributos a serem procurados, utilizando para tanto uma ou mais estruturas de busca `AttributeCertificateSearchFilter`. A definição das estruturas em ASN.1, a seguir, apresenta o formato de uma busca.

```

AttributeCertificateSearchReq ::= SEQUENCE {
    holder             Holder,
    issuer             AttCertIssuer OPTIONAL,
    attributesFilter   SEQUENCE OF AttributeCertificateSearchFilter
    OPTIONAL
}

AttributeCertificateSearchFilter ::= SEQUENCE {
    attributeOid       OBJECT IDENTIFIER,
    values             SET OF AttributeValue OPTIONAL
}

AttributeValue ::= ANY DEFINED BY OBJECT IDENTIFIER

```

### 4.3. Resposta

Toda requisição `AttributeCertificateReq` recebida pela autoridade de atributo deverá gerar uma resposta `AttributeCertificateResp`, que conerá o resultado da operação. Se a operação tiver sido executada com sucesso, conerá também um ou mais certificados de atributos. Sua estrutura é detalhada a seguir, em 4.3.1.

#### 4.3.1. AttributeCertificateResp

A resposta de um requisição pode ter sido processada com sucesso, isto é, a operação que foi solicitada foi efetuada como o esperado pelo requerente, bem como pode ter falhado. A estrutura a seguir representa um `AttributeCertificateResp`.

```

AttributeCertificateResp ::= SEQUENCE {
    status                ACStatusInfo,
    attributeCertificate   SEQUENCE OF AttributeCertificate OPTIONAL
    -- when the response is the result of a granted issue request,
    -- attributeCertificate must contain only ONE AttributeCertificate.
}

ACStatusInfo ::= SEQUENCE {
    status                ACStatus,
    statusString          PKIFreeText OPTIONAL,
    failInfo              ACFailureInfo OPTIONAL
}

ACStatus ::= INTEGER {
    granted                (0),
    rejection              (1)
}

```

No caso do status da resposta ser `granted`, a requisição terá sido devidamente processada. Se a requisição foi pela emissão de um CA, ele estará presente na resposta. Caso tenha sido requisitada a busca por CAs, eles também estarão presentes na resposta, se encontrados. Caso tenha sido requisitada uma revogação de CAs, ele será incluído na próxima LCAR e também estarão presentes na resposta.

A estrutura `ACStatusInfo` apresenta os possíveis resultados para uma requisição, que pode também ser rejeitada com o status `rejected`. O protocolo permite também a descrição textual da causa do resultado, através do `statusString` e o detalhamento dos motivos, especificando-se o `failInfo`.

## 5. Framework AAFW

O framework foi denominado de AAFW (Attribute Authority Framework) pois ele deve ser utilizado por autoridades de atributos que desejam gerenciar o ciclo de vida dos seus certificados de atributos. Ele é baseado exclusivamente no modelo de dados proposto no capítulo 4 e na RFC 5755 [Farrell et al. 2010].

Sua proposta é conseguir tratar uma requisição de emissão, busca ou revogação de certificado de atributo, bastando para isso a autoridade realizar poucas configurações. Essas requisições, representam no mais baixo nível um `AttributeCertificateReq` (ver 4.2.1) e a resposta da autoridade um `AttributeCertificateResp`, definido em 4.3.1.

### 5.1. Definindo o Comportamento da EEA

A fim de conseguir tratar o ciclo de vida do certificado de atributo, autoridades de atributos que desejam utilizar o framework AAFW devem informar ao framework como realizar algumas operações.

- Como carregar o certificado digital, algoritmo e chave privada para realização de assinatura.

- Como armazenar e pesquisar num repositório os certificados de atributos emitidos.
- Como obter o próximo número serial do certificado de atributo a ser emitido.
- Como receber e enviar as mensagens *AttributeCertificateReq* 4.2.1 e *AttributeCertificateResp* 4.3.1.
- Como e onde publicar os certificados de atributo revogados, se desejar utilizar revogação.
- Como validar se os atributos presente numa requisição de emissão são aceitáveis para o titular requisitado.
- Como tratar o campo *templateId* presente numa requisição.

Essas operações representam o domínio do problema e são mapeadas para interfaces dentro do framework. São os “pedaços” de software que precisam ser desenvolvidos e informado ao framework para poder formar uma aplicação completa.

Para facilitar a implementação de uma EEA básica, o framework já oferece uma ou mais opções para realizar as operações descritas acima, permitindo que implementadores da entidade emissora de atributos especifiquem apenas comportamentos que fujam aos já contemplados.

### 5.1.1. Definindo o tratamento dos atributos

Como definido no protocolo 4, é possível requisitar a emissão de atributos específicos ou simplesmente informando o campo *templateId*, que representa um identificador de modelo de certificado de atributo reconhecido pela EEA.

**Tratando atributos específicos** A forma com que a EEA consegue tratar solicitações de atributos específicos é através do registro validadores de atributo, representado pela interface *AttributeValidator*. Desta maneira, quando uma solicitação de emissão para determinado atributo é feita, o framework irá procurar por validadores nele registrado. Se achar um validador registrado para aquele OID do atributo solicitado, então é acionado seu método de validação. O método de validação deve informar se os atributos presentes numa requisição de emissão são aceitáveis para o titular requisitado.

**Tratando atributos com templates** Outra maneira de se realizar a emissão é através do registro de modelos de certificado de atributo, representado pela interface *ACTemplate*. Um modelo deve informar a validade, as extensões e os atributos que farão parte dos certificados a serem emitidos segundo seus critérios. O requerente deve simplesmente informar na requisição o campo *templateId* e o *holder* que receberá o certificado de atributos, deixando os demais campos vazios. O framework irá mapear o *templateId* recebido para um template nele registrado.

**Suporte à Revogação de atributos** Conforme requisito do DOC-ICP 16.01 [ITI a], a EEA deve indicar se suporta a revogação de atributos. A forma com que o framework dá suporte a essa funcionalidade é através do registro de publicador de LCR, representado pela interface *CRLPublisher*. O publicador de LCR diz onde publicar a LCR (LDAP, sistemas de arquivos, entre outros), além de configurações como seu período de validade e periodicidade de publicação. Um certificado de atributos terá codificada em sua estrutura a extensão *crlDistributionPoints*, com o ponto de distribuição informado pelo publicador.

Para que um atributo seja passível de revogação, basta registrar um publicador de LCR para ele. Com isso, sempre que for recebida uma requisição de revogação o framework irá verificar se existe publicador associado ao atributo a ser revogado. Caso exista, utilizará o publicador para realizar a revogação. O framework também fica encarregado de nunca deixar expirar as LCRs publicadas pelos publicadores. Sempre que uma LCR estiver prestes a expirar, uma outra será emitida em sua substituição.

## 5.2. Utilização do Framework

Para conseguir utilizar o framework o usuário deve herdar a classe *AttributeAuthority*, implementando seus métodos abstratos *getSystemFactory* e *setup*.

Na figura 1, os retângulos em azul representam as interfaces do framework. Os retângulos em laranja (de borda tracejada) representam as implementações dessas interfaces, fornecidas pelo framework. O retângulo em verde (de borda pontilhada) representa uma possível aplicação implementada utilizando o framework.

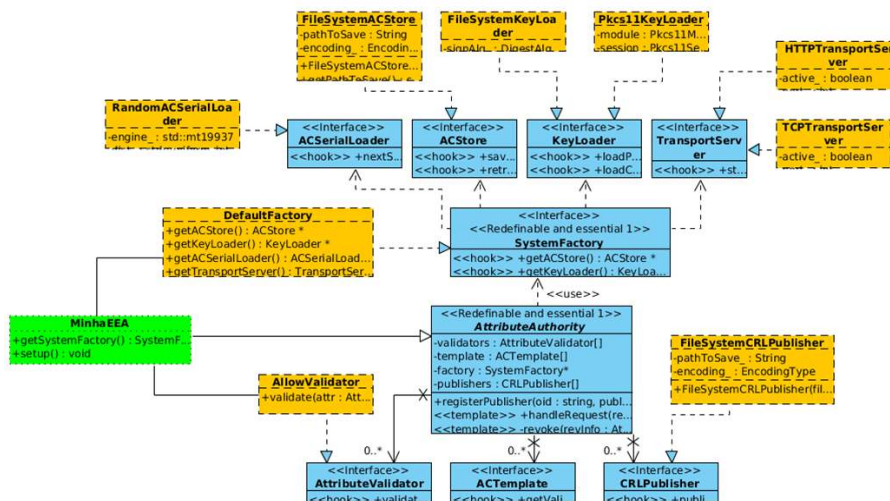


Figure 1. Utilização do Framework

A figura mostra um caso onde o usuário criou uma subclasse de *AttributeAuthority* chamada de *MinhaEEA*. Pela associação com *DefaultFactory*, entende-se que será retornado uma instância de *DefaultFactory* no método *getSystemFactory*. O método *setup* dá a chance para a *MinhaEEA* registrar os seus validadores de atributos, template de atributos e revogadores de atributo. Pela associação com *AllowValidator*, entende-se que será registrado o validador *AllowValidator* para um ou mais atributo de *MinhaEEA*.

As implementações das interfaces do framework (representada pelos retângulos em laranja, de borda tracejada) são as maneiras de alteração do comportamento do framework.

## 6. Aplicações Protótipo

Para avaliar o protocolo definido e framework que o implementa, foi desenvolvido um sistema prevendo um cenário real de uso. O cenário proposto foi o de uma rede de cinemas que deseja utilizar certificados de atributos para vender bilhetes virtuais seguros pela

internet, para as sessões de seus cinemas. Esse cenário é explorado em mais detalhes ao longo deste capítulo.

## 6.1. Aplicação para controle de ingressos

Foi desenvolvido um conjunto de aplicações para representar um cenário baseado nos seguintes requisitos:

1. Existe uma rede de cinemas que deseja melhorar a segurança e facilidade de acesso ao seus ingressos (tickets).
2. As filiais devem ser gerenciadas de modo que a emissão de ingressos seja previamente aprovada e feito por uma central.
3. Para aprovação de ingressos de estudantes eles devem estar cadastrados na central para garantir seu benefício. Para aprovação de ingressos para adultos nenhuma verificação adicional é feita. Supõe-se aqui que utiliza-se uma rede privada e que a central é capaz de restringir o acesso às suas filiais.
4. Cada ingresso emitido deve ser verificado quanto a sua validade, autenticidade e integridade a fim de permitir a entrada no cinema.

O usuário se apresenta ao balcão de uma filial e solicita a compra de um ticket para determinada sessão. A filial se comunica com a central para realização da emissão. Ao receber a resposta, ela imprime o ticket no formato QR Code e o entrega para o usuário. O usuário, de posse do ticket impresso se encaminha para entrada onde ocorre a sessão. A seguir lhe é requisitado o ticket por um funcionário, que o escanea através de um aplicativo instalado num *smart phone*. Será apresentado pelo aplicativo o resultado da validação do ticket bem como os dados da sessão comprada pelo usuário. Se o ticket estiver válido, o acesso à sessão lhe é liberado.

### 6.1.1. EEA para Acesso aos Cinemas

Foi criada para rede de cinemas uma autoridade de atributo que tem a função de emitir e distribuir os ingressos para o filmes de suas filiais. Representa a central. Ela também deve manter uma base de estudantes para aprovação de emissão de tickets de estudantes.

Considerando que a rede de cinemas já possua um sistema que gerencie suas filiais ela terá pouco trabalho para a implementação desse cenário. Terá apenas de dizer ao framework AAFW como armazenar e buscar os certificados de atributos emitidos em seu próprio repositório, através da interface *ACStore* e também como validar as requisições para o tipo estudante.

Feito isso, a atuação da EEA para emissão de ingresso será passiva. Ela irá receber uma requisição 4.2 de uma de suas filiais e automaticamente emitirá o ingresso, onde tudo é encapsulado pelo framework AAFW.

O sistema é dividido em 3 partes:

**Modelo de tickets** Parte do sistema onde é definido um OID para cada modelo de ticket - estudante ou adulto - que será utilizado posteriormente como um atributo na codificação do certificado de atributo. Para cada atributo definido, é associado seu tipo de validação

e também indicado se ele suporta ou não revogação. Esses dados serão mapeados para implementações das interfaces `AttributeValidator` e `CRLPublisher` dentro do framework AAFW e configurados no método `setup`.

**Cadastramento de estudantes** Para garantir o benefício de estudante a pessoa deve estar cadastrada na base de dados da autoridade de atributo. Parte do sistema onde é gerenciado os estudantes.

**Tickets Emitidos** Parte do sistema onde são mostrados os tickets (certificados de atributos) emitidos por essa autoridade de atributo.

### 6.1.2. Filial

As filiais terão a responsabilidade de requisitar a emissão de tickets para a sua EEA. Representa a terceira parte interessada na emissão de certificado de atributo. Considerando que elas já possuem um sistema de emissão de tickets, então pouca alteração será necessária para a adequação do novo cenário. Seu trabalho será apenas o de montar uma requisição 4.2 com os dados da sessão (nome do filme, sala, horário, validade) codificado no atributo a ser requisitado (estudante ou adulto). Além disso ela terá que extrair o certificado de atributo (ticket) da resposta.

O sistema é dividido em 5 partes:

**Gerenciamento de sessões** Parte do sistema onde são gerenciados as sessões dos filmes. É possível especificar a data e hora da sessão, bem como nome do filme, duração, sala onde irá ocorrer, etc.

**Gerenciamento de tickets** Parte do sistema onde são gerenciados os tickets. Nessa parte pode-se verificar os três casos de uso principais do framework AAFW: Emitir Certificado de Atributo, Buscar Certificado de Atributo e Revogar Certificado de Atributo. Além disso, também é possível visualizar e imprimir os tickets já emitidos.

**Emissão de tickets** Parte do sistema onde são emitidos os tickets. Representa o caso de uso Emitir Certificado de Atributo. Ao preencher os dados, uma requisição de emissão `AttributeCertificateIssueReq` é construída. Será utilizado o nome informado para solicitar um CA do tipo autônomo. Esse dado será codificado na estrutura `holder`. Pode ser escolhido também o tipo do ingresso: adulto ou estudante, o que resultará em atributos diferentes na requisição. O tipo de ingresso será codificado na estrutura `attributes`. Outro dado necessário é a validade `validityPeriod` que será preenchida com os dados de início e término da sessão escolhida.

A requisição depois de construída é então enviada para autoridade de atributo de cinemas e ao receber a resposta, extraído o certificado de atributo.



**Impressão de tickets** Após emitir um ticket o usuário será redirecionado para interface de impressão. O QR code representa o certificado de atributo codificado em PEM. O usuário deve apresentá-lo para ter direito de assistir seu filme comprado. O ingresso passará pela verificação automática da sua validade, integridade e autenticidade realizado por um outro aplicativo 6.1.3.

**Busca de tickets** Parte do sistema onde é permitido buscar por tickets emitidos pela autoridade de atributo de cinemas. Representa o caso de uso Buscar Certificado de Atributo. Ao preencher os dados, uma requisição de busca é construída, enviada para autoridade de atributo de cinemas 6.1.1 e extraído os certificados de atributo recebidos.

A maneira com que essa requisição é construída assemelha-se ao caso de emissão. Porém, é populado uma estrutura `AttributeCertificateSearchReq`, fazendo um filtro pelo atributo especificado (estudante ou adulto) e pelo titular (nome) informado.

Após realizar a busca o usuário será redirecionado para interface de resultado da busca, onde os tickets obtidos podem ser impressos. Nenhum ou vários tickets podem ser obtidos.

### 6.1.3. Verificador de tickets

Para conseguir acesso à sessão, o usuário deverá apresentar o ticket para verificação. Para isso foi desenvolvido uma aplicação android. A verificação consiste na validação da integridade da assinatura, na validade do certificado de atributo, na presença dos atributos de adulto ou estudante e na confiança do seu emissor (autoridade de atributo de cinemas).

O sistema é dividido em 3 partes:

**Configurações** Parte do sistema onde é configurado a autoridade de atributo confiável. A configuração é feita mediante do upload do seu certificado, que será utilizado para verificar a assinatura do certificado de atributo.

**Leitura de Ticket** Parte do sistema onde é feita a leitura dos tickets. O usuário deve imprimir seu ticket e apresentá-lo para poder ter acesso ao cinema.

**Resultado da Leitura** A leitura mostra se a assinatura está íntegra, sua validade está expirada, além dos dados da sessão extraídos do ticket, como sala filme, tipo (estudante ou adulto), etc.

## 6.2. Outras aplicações

Diversos outros tipos de aplicações podem ser construídas utilizando o framework. Tudo que representa determinado poder, privilégio ou atribuições a uma entidade pode ser representado por um certificado de atributo e portanto passível de ser utilizado por uma autoridade para emissão e controle dos mesmos. O caso citado acima, ingressos de cinema, pode ser abstraído para qualquer tipo de ingresso para qualquer evento.

## 7. Considerações Finais

O trabalho apresentado atende os requisitos dos documentos DOC-ICP-16 e DOC-ICP-16.01 que falam sobre certificação de atributos para uso no âmbito da ICP-Brasil, além de seguir o formato definido pela RFC 5755.

Como contribuições desse trabalho, podemos citar:

1. Proposta de um protocolo para solicitação de certificado de atributo, considerando seu ciclo de vida;
2. Implementação de um framework de código aberto e multiplataforma que utiliza o protocolo proposto para desenvolvimento de sistemas de autoridades de atributo;
3. Implementação de uma biblioteca de código aberto e multiplataforma para manipulação em alto nível de estruturas ASN.1 e criptografia;
4. Implementação de aplicação verificadora de certificado de atributo;

Como trabalhos futuros, podemos citar:

1. Prever forma para validação da autenticidade do titular. Ou seja, validar que quem apresenta o certificado de atributo para uso é realmente o seu titular, como codificado em sua estrutura.
2. Um controle de privilégios embutido no protocolo sobre quem pode solicitar a emissão, busca ou revogação do certificado de atributo. Uma maneira de resolver esse problema seria encapsulando a estrutura `AttributeCertificateReq` num pacote assinado CMS do tipo `Signed-data` [Housley 2009]. A EEA, ao receber a mensagem, teria a informação do assinante e a assinatura da estrutura, podendo decidir se permite ou não o processamento dessa requisição;
3. Prever a solicitação de múltiplas requisições na mesma mensagem;

## References

- BRASIL. Medida provisória n.º 2.200, de 28 de junho de 2001. Institui a Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil, e dá outras providências. Diário Oficial da União, Brasília, DF, 29 jun. de 2001.
- Choudhury, S., Bhatnagar, K., and Haque, W. (2002). *Public Key Infrastructure Implementation and Design*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition.
- Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard).
- Farrell, S., Housley, R., and Turner, S. (2010). An Internet Attribute Certificate Profile for Authorization. RFC 5755 (Proposed Standard).
- Freier, A., Karlton, P., and Kocher, P. (2011). The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (Historic).
- Housley, R. (2009). Cryptographic Message Syntax (CMS). RFC 5652 (INTERNET STANDARD).
- Housley, R. and Polk, T. (2001). *Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure*. John Wiley & Sons, Inc., New York, NY, USA.

ITI. PERFIL DE USO GERAL E REQUISITOS PARA GERAÇÃO E VERIFICAÇÃO DE CERTIFICADOS DE ATRIBUTO NA ICP-BRASIL. Versão 1.0. Brasília, Dezembro 2012. DOC-ICP-16.01.

ITI. VISÃO GERAL SOBRE CERTIFICADO DE ATRIBUTO PARA A ICP-BRASIL. Versão 1.0. Brasília, Julho 2012. DOC-ICP-16.

Pinkas, D., Pope, N., and Ross, J. (2008). Cms advanced electronic signatures (cades). RFC 5126.