

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

Alex Sandro da Silva Pereira

**Aplicabilidade do Padrão de Assinatura PAdES no
Âmbito 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 2012

Aplicabilidade do Padrão de Assinatura PAdES no Âmbito da ICP-Brasil.

Alex Sandro da Silva Pereira

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

Thiago Acórdi Ramos

Darlan Vivian

Lucas Ferraro

Agradecimentos

Agradeço a todos que fizeram parte desse trabalho, ao Jeandré Monteiro Sutil que auxiliou-me em tudo, sempre pronto a ajudar orientou-me da melhor forma, fazendo com que cada palavra que compõe este trabalho fosse colocada da melhor maneira possível, um ser cujo espelhei-me muito ao longo do caminho. Ao Thiago Acórdi Ramos o agradecimento de amigo, pela paciência, todo apoio técnico e respostas quase que em tempo real que me ajudaram muito em todas as etapas da confecção desse trabalho. Ao Darlan Vivian pelas inúmeras contribuições e apoio. Ao Prof. Ricardo Felipe Custódio e Lucas Ferraro cuja força foi fundamental. A toda minha família e amigos em especial minha mãe amada e a Rosângela Fátima Resmini. No momento não consigo encontrar palavras que possam descrever tamanha alegria, o que deixo a vocês é o meu muito obrigado.

(Alex Sandro da Silva Pereira)

Resumo

O PDF (Formato de Documento Portátil), criado em 1993 pela Adobe Systems e publicado como norma ISO 32000-1, desde 2008, é um padrão para visualização de documentos que permite encapsular assinaturas digitais no próprio arquivo. Em julho de 2008 o ETSI (European Telecommunications Standard Institute), organização responsável pela padronização de tecnologias da informação e comunicação na Europa, definiu uma série de especificações de como assinaturas digitais podem ser integradas e visualizadas nesse formato portátil, conhecida como PAdES - PDF Advanced Electronic Signatures. Essas especificações estão em conformidade com as regulamentações legais da União Europeia. No contexto nacional, o mecanismo que viabiliza a emissão de certificados digitais e garante a validade jurídica de documentos em forma eletrônica é a Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil), que possui políticas e normas para os processos de produção e verificação de assinaturas digitais. No entanto, até o momento, a ICP-Brasil não disponibiliza regulamentações referentes ao processo de assinatura digital no padrão PAdES. O estudo da viabilidade da utilização do formato PAdES em conformidade com as normas e políticas da ICP-Brasil é objetivo de estudo deste trabalho.

O que resultou na apresentação de propostas de representação das assinaturas digitais sob cada política de assinatura definida na ICP-Brasil, com exceção da assinatura digital com referências para validação, sendo possível analisar algumas mudanças na regulamentação do Padrão Brasileiro de Assinatura Digital decorrentes da adoção do padrão PAdES.

Palavras chave: PAdES, ICP-Brasil, Assinatura Digital, PDF.

Abstract

The PDF (Portable Document Format), created in 1993 by Adobe Systems and published as ISO 32000-1, since 2008 is a standard for viewing documents which allows you to encapsulate digital signatures in the file itself. In July 2008, the ETSI (European Telecommunications Standards Institute), an organization responsible for standardizing information and communication technologies in Europe, defined a set of specifications that describes how digital signatures can be integrated and visualized in this portable format, known as PAdES - Advanced PDF Electronic Signatures. These specifications are in accordance with legal regulations of the European Union. In the national context, the mechanism that enables the issuance of digital certificates and ensures the legal validity of electronic documents is the Brazilian Public Key Infrastructure (ICP-Brasil), which has policies and standards for the process of generation and verification of digital signatures. However, until now, the ICP-Brasil does not provide regulations for the digital signature process in PAdES standard. The feasibility of using the format PAdES in accordance with the rules and policies of the ICP-Brasil is the objective of this study. This work resulted in proposals for representation of digital signatures on each signature policy defined by the ICP-Brasil, except for the digital signature with references for validation, being possible to analyze the feasible changes in the regulation of the Brazilian's Digital Signature Standard for the adoption of PAdES standard.

Keywords: PAdES, ICP-Brasil, Digital Signature, PDF.

Lista de Figuras

2.1	Operação de criptografia assimétrica. (Imagem extraída de (Moecke))	8
2.2	Assinatura Digital.(Imagem extraída de (Moecke))	10
3.1	Estrutura CMS (<i>signed-data</i>).	16
3.2	Estrutura CMS (<i>signerInfo</i>).	17
3.3	CAdES	19
4.1	Arquitetura hierárquica da ICP-Brasil.	26
4.2	Relação entre os documentos ICP-Brasil e os formatos de assinatura. (<i>Extraído do DOC-ICP-15 (ITI).</i>)	29
4.3	Formatos de Assinatura Digital ICP-Brasil. (*Opcional para a política AD-RA)	30
5.1	Assinatura digital em documentos PDF.	42
5.2	Múltiplas Assinaturas	46
6.1	Dicionário DSS e <i>Signature VRI Dictionary</i> (Imagem extraída do ETSI TS 102 7784 (Institute 2009)).	55
6.2	Aplicação de Sucessivos Carimbos (Imagem extraída do ETSI TS 102 7784 (Institute 2009)).	56
7.1	Assinatura digital com Referência Básica	62
7.2	Assinatura digital com Referência do Tempo	63
7.3	Problema Assinatura Incremental.	64

7.4	AD-RC utilizando PAdES-LTV.	67
7.5	Assinatura Digital com Referências para Arquivamento.	68
8.1	Assinador PAdES	72
8.2	Assinatura PAdES AD-RB	73
8.3	Atributos Assinados da Estrutura da Assinatura	74
8.4	Verificação em Software	75
8.5	Geração de Assinatura	75
8.6	Configuração CADES Equivalent	76
8.7	Estrutura Assinatura Acrobat Pro.	77
8.8	Integração com o Repositório do Windows.	78
8.9	Verificação da Assinatura com o <i>Adobe Reader X</i>	78
8.10	Assinatura que Exige LCR	78

Lista de Siglas

AD-RB	Assinatura Digital com Referência Básica
AD-RT	Assinatura Digital com Referência do Tempo
AD-RV	Assinatura Digital com Referências para Validação
AD-RC	Assinatura Digital com Referências Completas
AD-RA	Assinatura Digital com Referências para Arquivamento
CAdES	CMS Advanced Electronic Signature
CAdES-A	CAdES with Archive validation data
CAdES-BES	CAdES Basic Electronic Signature
CAdES-C	CAdES with Complete validation data
CAdES-EPES	CAdES Explicit Policy Electronic Signature
CAdES-T	CAdES with Time
CAdES-X	CAdES with Extended validation data
CadES-X Long	CAdES with Extended Long validation data
CMS	Cryptographic Message Syntax
ETSI	European Telecommunications Standards Institute
ICP	Infraestrutura de Chaves Públicas
ICP-Brasil	Infraestrutura de Chaves Públicas Brasileira
PAdES	PDF Advanced Electronic Signature
PAdES-BES	PAdES Basic Electronic Signature
PAdES-EPES	PAdES Explicit Policy Electronic Signature
PAdES-PAdES-LTV	PAdES Long Term Validation
PBAD	Padrão Brasileiro de Assinatura Digital
PDF	Portable Document Format
PKCS	Public Key Cryptography Standards
PKCS#7	Public Key Cryptography Standards #7
XAdES	XML Advanced Electronic Signatures
XML	eXtensible Markup Language
XMLDSig	XML digital signature

Sumário

Lista de Figuras	vii
1 Introdução	1
1.1 Contextualização	1
1.2 Objetivos	3
1.2.1 Geral	4
1.2.2 Específicos	4
1.3 Justificativa	4
1.4 Metodologia	5
1.5 Limitações do Trabalho	5
1.6 Organização do Trabalho	6
2 Conceitos Básicos	7
2.1 Introdução	7
2.2 Criptografia Assimétrica	7
2.3 Assinatura Digital	8
2.3.1 Elementos Associados a uma Assinatura	9
2.3.2 Política de Assinatura	10
2.4 Resumo Criptográfico	11
2.5 Infraestrutura de Chaves Públicas	12
2.5.1 Certificado Digital	13
2.6 Considerações Finais do Capítulo	13

3	Padrões de Assinatura Digital	15
3.1	Introdução	15
3.2	Cryptographic Message Syntax (CMS)	15
3.3	CMS Advanced Electronic Signatures (CAAdES)	18
3.3.1	Tipos de Assinatura	18
3.4	XMLDSig e XML Advanced Electronic Signatures	23
3.5	Considerações Finais	24
4	Padrão Brasileiro de Assinatura Digital	25
4.1	Introdução	25
4.2	Estrutura normativa	26
4.3	Políticas de Assinatura Digital	29
4.3.1	AD-RB	30
4.3.2	AD-RT	31
4.3.3	AD-RV	32
4.3.4	AD-RC	34
4.3.5	AD-RA	34
4.4	Considerações Finais do Capítulo	35
5	Formato de Documento Portátil	36
5.1	Introdução	36
5.2	Estrutura Básica	37
5.3	Atualização	39
5.4	Assinatura Digital	39
5.4.1	Assinatura de Certificação	44
5.4.2	Múltiplas Assinaturas	45
5.5	Considerações Finais do Capítulo	47
6	PDF Advanced Electronic Signatures	48
6.1	Introdução	48
6.2	PAdES Basic	49

6.3	PAdES Enhanced - PAdES-BES and PAdES-EPES	50
6.4	PAdES Long Term - PAdES-LTV Profile	52
6.5	Considerações Finais do Capítulo	57
7	Utilização do Padrão PAdES no Âmbito da ICP-Brasil	58
7.1	Introdução	58
7.2	Perfil de uso Geral	58
7.3	Políticas de Assinatura	61
7.3.1	AD-RB	61
7.3.2	AD-RT	62
7.3.3	AD-RV	63
7.3.4	AD-RC	65
7.3.5	AD-RA	66
7.4	Impactos e Mudanças na Normatização do PBAD	68
7.5	Considerações Finais do Capítulo	70
8	Protótipo Assinador PBAD-PAdES	71
8.1	Introdução	71
8.2	Assinatura	72
8.3	Verificação	73
8.4	Compatibilidade com Adobe Acrobat	74
8.4.1	Geração de Assinaturas	74
8.4.2	Verificação da Assinaturas	77
8.5	Considerações Finais do Capítulo	79
9	Considerações Finais	80
9.1	Contribuições	81
9.2	Trabalhos Futuros	81
	Referências	83
A	Apêndice A –Código Fonte	88

B Apêndice B –Artigo

1 Introdução

1.1 Contextualização

Um documento é uma fonte de informação, na forma material, capaz de ser usada para referência, estudo ou prova (Buckland 1997). Um documento eletrônico, de acordo com projeto de lei sobre o comércio eletrônico (BRASIL 2001) que "Dispõe sobre o valor probante do documento eletrônico e da assinatura digital, regula a certificação digital, institui normas para as transações de comércio eletrônico e dá outras providências. ", em seu art. 2, inciso I, é "a informação gerada, enviada, recebida, armazenada ou comunicada por meios eletrônicos, ópticos, opto-eletrônicos ou similares". Para que haja confiança nas informações transitadas eletronicamente, tais documentos devem produzir efeito de prova, conhecida como eficácia probante. Para possuir essa característica, devem contar com algumas propriedades, tais como: a autenticidade, a integridade, a autoria e a tempestividade. Um meio de fornecer esse conjunto de propriedades, fundamentais aos documentos eletrônicos, dá-se através do uso de assinaturas digitais.

Assinatura digital é uma tecnologia que permite a tramitação de documentos, processos e outras informações com segurança. Baseada no uso da certificação digital, um dos papéis dessa tecnologia é prover um meio pelo qual pode-se identificar a validade da informação contida em um documento eletrônico. Vários esforços foram desenvolvidos com esse objetivo, dando origem a diferentes padrões de assinaturas. Um formato bastante popular na representação de documentos eletrônicos é o Formato de Documento Portátil (PDF), capaz também de representar documentos di-

gitalmente assinados. O padrão PDF suporta a assinatura digital do conteúdo representado, baseando-se, para tanto, em outras tecnologias de assinatura eletrônica, são elas CADES (INSTITUTE 2010), XAdES (INSTITUTE 2010) e CMS (Housley 2009). O formato de documentos eletrônicos PDF conta com o suporte do popular Adobe Reader à geração e verificação de assinaturas.

Este formato foi desenvolvido em 1993, pela Adobe Systems e permite a comunicação de material visual entre diferentes sistemas e aplicações. O PDF pode conter diferentes elementos como texto, gráficos, fontes e imagens. Sua capacidade de deixar a aparência do documento fixa, independente de hardware e sistema operacional, auxiliou muito em sua popularização. Em 1 de julho de 2008 o PDF, versão 1.7, tornou-se um padrão aberto e foi publicado pela ISO, organização internacional responsável pela publicação de variados padrões aceitos mundialmente, transformando-se na norma ISO 32000-1:2008 (Standard 2008). Atualmente, existem diversas bibliotecas para criação e manipulação de arquivos PDF, tornando o formato amplamente difundido entre diferentes plataformas e sistemas. É cada vez mais comum a presença nativa de visualizadores PDF incorporados a distribuições de sistemas operacionais. Linux e OS X são alguns exemplos, estes possuem também o suporte oficial da Adobe. Para os que não contam nativamente com os visualizadores, há inúmeras opções gratuitas disponíveis para *download*, o que contribui para a disseminação do formato.

O principal diferencial encontrado no formato PDF, através do editor da Adobe, é o suporte à representação gráfica das assinaturas no próprio documento, aproximando-o do documento real e, conseqüentemente, colaborando com sua aceitação.

Um ano após a publicação do formato PDF pela ISO, em julho de 2009, o ETSI (European Telecommunications Standard Institute) publicou um padrão que define uma série de especificações, de como pode ser utilizada assinatura digital em documentos PDF, conhecida como PAdES –PDF Advanced Electronic Signatures –que documentam e estendem o suporte de assinatura digital especificado na ISO 32000-1. Essa série é dividida em cinco partes, sendo que a primeira apresenta os recursos da

assinatura PDF apenas considerando aspectos gerais. A partir da segunda, é descrito o formato básico do PAdES e subsequentemente as especificações para implementadores. Criado em 1988, o ETSI é uma organização responsável pela padronização de tecnologias da informação e comunicação na Europa. Então, a adição do PAdES ao conjunto de soluções para assinatura em PDF está em conformidade com as regulamentações legais da União Europeia.

No contexto nacional, a Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil), mantida pelo Instituto Nacional de Tecnologia da Informação (ITI), é a instituição que regulamenta os critérios para a confiança e aceitação de documentos eletrônicos. A infraestrutura foi instituída pela Medida Provisória 2.200-2 (BRASIL 2001) que "Institui a Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil, transforma o Instituto Nacional de Tecnologia da Informação em autarquia, e dá outras providências.", de 24 de agosto de 2001, visando garantir a integridade, a autenticidade e a validade jurídica de documentos na forma eletrônica. Cabe à ICP-Brasil a emissão e gerenciamento de certificados digitais, bem como a regulamentação dos procedimentos, padrões e políticas que garantam a segurança no uso de documentos eletrônicos. Assim, a utilização de formatos padronizados pela ICP-Brasil é importante para garantir confiabilidade e credibilidade no processo de criação e validação das assinaturas. O padrão de assinatura digital PAdES encontra-se em estudo para normatização segundo as regras definidas pela ICP-Brasil. Este trabalho pretende analisar os impactos da adoção do padrão PAdES pela ICP-Brasil, buscando verificar quais as adaptações necessárias à normatização.

1.2 Objetivos

Vistas as preocupações na adequação do formato de assinatura digital no âmbito da ICP-Brasil, são apresentados os objetivos deste trabalho.

1.2.1 Geral

O objetivo deste trabalho é analisar soluções e impactos no processo de adaptação do padrão de assinatura digital PAdES aos requisitos da ICP-Brasil.

1.2.2 Específicos

- Avaliar o formato de documento portátil (PDF), o padrão PAdES e as políticas de assinatura da ICP-Brasil;
- Analisar a viabilidade de utilização do padrão PAdES sob as regras e políticas de uso de assinatura digital no âmbito da ICP-Brasil;
- Propor modelos de assinaturas PAdES aderentes aos requisitos do padrão brasileiro de assinatura digital, para cada uma das políticas definidas;
- Avaliar os impactos da adoção do padrão pela ICP-Brasil.
- Desenvolver uma aplicação para gerar e verificar assinaturas digitais de acordo com o padrão PAdES, em concordância com as premissas de uma assinatura digital válida no âmbito da ICP-Brasil.

1.3 Justificativa

Os documentos eletrônicos, aliados à flexibilidade dos meios digitais, possibilitam, de forma rápida e fácil, o registro, a recuperação e a gestão da informação. Esses documentos, em determinados cenários, podem ser utilizados como uma alternativa ao papel, sendo utilizados até mesmo para registro de atos jurídicos, conforme permite a Lei 11.419 (BRASIL 2006) que "Dispõe sobre a informatização do processo judicial; altera a Lei no 5.869, de 11 de janeiro de 1973 –Código de Processo Civil; e dá outras providências.". Assim, a existência de um formato que permita representar um documento, mesmo em diferentes sistemas e aplicações, mantendo uma aparência

única e provendo características como autenticidade, integridade, autoria e tempestividade é altamente relevante para a evolução da forma como as informações sensíveis são gerenciadas.

O formato de documento portátil (PDF), permite a comunicação de material visual entre diferentes sistemas e aplicações, tendo como característica principal a capacidade de manter, da melhor forma possível, a aparência do documento fixa, independente de hardware e sistema operacional. A avaliação dos impactos do uso desta tecnologia para a representação de documentos em âmbito nacional é, dessa forma, extremamente importante, permitindo a fusão entre um dos formatos de representação de documentos mais utilizados na atualidade e a eficácia probante fornecida pelo padrão brasileiro de assinatura digital.

1.4 Metodologia

O passo inicial da confecção deste trabalho foi através da realização de uma pesquisa bibliográfica sobre os principais conceitos envolvidos no processo de assinatura digital de documentos. Foi estudado também o padrão PDF, de forma a melhor entender sua arquitetura e forma de representação de informações. Além disso, para avaliar a adequação à regulamentação do formato de assinatura PAdES, no contexto nacional, estudou-se os procedimentos e políticas propostas no padrão de assinatura digital do Brasil. Com base nesse estudo, propôs-se um modelo para os variados tipos de assinatura possíveis no âmbito da ICP-Brasil, utilizando-se o padrão PAdES. Para validar os modelos propostos, foi implementado um gerador e verificador de assinaturas como prova de conceito. Por fim, buscou-se avaliar ainda os possíveis impactos e adaptações necessárias, dada a natureza peculiar do padrão PAdES.

1.5 Limitações do Trabalho

Apesar deste trabalho ter íntima ligação com aspectos jurídicos referentes à validade de documentos, este trabalho restringe-se ao estudo da assinatura digital,

apenas sob os aspectos técnicos. O padrão PAdES suporta o uso de estruturas especificadas no padrão CAdES e XAdES, para representação de assinaturas. Neste trabalho, serão consideradas apenas as assinaturas CAdES, ficando as assinaturas PAdES-XAdES como proposta de trabalhos futuros. A normatização do padrão PAdES no âmbito da ICP-Brasil encontra-se em desenvolvimento. Este trabalho apresenta apenas uma proposta para os modelos de assinatura tratados pela ICP-Brasil, não define formatos de políticas de assinatura PAdES e nem formas de codificação de políticas para este tipo de assinatura, não implicando no formato final de como o padrão PAdES será adotado pela infraestrutura. Por vezes, no texto, utiliza-se o termo PBAD-PAdES como referência ao modelo proposto. O uso deste termo restringe-se ao escopo deste trabalho, sem qualquer relação com o padrão em atual desenvolvimento.

1.6 Organização do Trabalho

No capítulo 2, serão apresentadas definições dos principais conceitos relacionados ao processo da confecção de assinaturas eletrônicas. O capítulo 3, apresenta os principais padrões de assinatura digital. O capítulo 4 a Infraestrutura de Chaves Públicas Brasileira e mostra os formatos de assinaturas digitais aprovados no âmbito da ICP-Brasil. O capítulo 5 apresenta o formato de documento portátil PDF e a forma como este padrão suporta assinatura digital. Seguido pelo capítulo 6 que mostra o padrão PAdES, que mantém compatibilidade com o padrão PDF. O capítulo 7 apresenta a análise da viabilidade da utilização do PAdES no âmbito da ICP-Brasil. O capítulo 8 exibe os resultados da implementação de um Assinador PAdES que gera assinaturas no formato AD-RB. Por último o capítulo 9 apresenta algumas considerações finais e possíveis trabalhos futuros.

2 Conceitos Básicos

2.1 Introdução

Para a compreensão dos formatos de assinaturas digitais é necessário um apanhado geral dos principais conceitos envolvidos no processo de assinatura e verificação de documentos eletrônicos. Neste capítulo serão apresentadas definições sobre criptografia assimétrica, assinatura digital e certificado digital, base fundamental para a aplicação de assinaturas digitais.

2.2 Criptografia Assimétrica

Criptografia é a escrita em códigos ou escrita de forma ilegível. Cripto, do grego “kryptos”, significa escondido, oculto, e grafia, também do grego “graphos”, representa escrita. De acordo com Schneier (1996), criptografia é a arte e a ciência de manter mensagens seguras. Este conceito surgiu da necessidade de transmitir informações de maneira sigilosa e consiste no simples embaralhamento de letras, ou até operações matemáticas sobre números que representam o código de um alfabeto.

Uma aplicação muito conhecida da criptografia é a cifragem. No mundo digital, como as informações são representadas de forma binária, o processo de cifragem utiliza uma função criptográfica para transformar um conteúdo inteligível em algo aparentemente sem qualquer significado. Para tanto, faz-se uso de uma chave secreta de conhecimento apenas do criador da informação. O resultado é o chamado conteúdo cifrado. O processo inverso da cifragem é a decifragem. Se uma única chave é uti-

lizada para os dois processos, cifragem e decifragem, atribui-se o nome criptografia simétrica.

A criptografia assimétrica (Diffie e Hellman 1976) foi proposta por Whitfield Diffie e Martin E. Hellman e, publicada em 1976. Este conceito tem a mesma função da criptografia simétrica, cifrar e decifrar dados, porém, diferente dela, este processo é realizado por chaves distintas, ou seja, não utiliza uma única chave e sim, um par (uma pública e outra privada). A chave privada deve ser mantida em segredo, já a pública pode ser disponibilizada em locais públicos. O que é cifrado por uma é, somente, decifrado pela outra, e vice-versa. Este conceito, que possibilitou a troca de chaves sobre canais inseguros, tem sua segurança baseada na dificuldade de cálculos de logaritmos discretos. Por outro lado possui custo computacional mais elevado. Um exemplo simples do uso da criptografia assimétrica é ilustrado pela figura 2.1:



Figura 2.1: Operação de criptografia assimétrica. (Imagem extraída de (Moecke))

A criptografia assimétrica é a base fundamental para construção de assinaturas digitais, pois permite, mantendo-se uma das chaves em segredo e publicando-se a segunda, estabelecer uma espécie de “identidade” em meio digital, conforme será apresentado em 2.3 e 2.5.1.

2.3 Assinatura Digital

A assinatura digital possibilita a autenticidade e não repúdio do signatário, permitindo verificar a integridade dos dados. Ela é uma aplicação importante da criptografia assimétrica e resulta da utilização da chave privada (ver seção 2.2) sobre os dados. De acordo com Schneier (1996), uma assinatura digital, idealmente, deve possuir as seguintes propriedades:

- Prover autenticidade. O receptor tem certeza da identidade do signatário;
- Impossível de ser falsificada;
- A partir de um documento assinado não pode existir a possibilidade de transferir esta assinatura para outro documento;
- Proteger a integridade. O documento não pode ser alterado após assinado;
- Não permitir que o signatário negue a sua assinatura.

Como a utilização da criptografia assimétrica exige um custo computacional elevado, ela é inadequada para cifragem de conteúdos extensos. Logo, a forma tradicional de assinatura digital não ocorre sobre o documento em si, mas sim sobre uma identificação única do mesmo (NIST 1994). Esta identificação é obtida através de uma função resumo (ver seção 2.4), que tem como resultado um identificador para o conteúdo. Portanto, o processo de assinatura digital é a utilização de uma função resumo sobre a mensagem e a subsequente aplicação de um algoritmo de assinatura que faz uso da chave privada sobre este resumo, resultando no resumo cifrado. Já o processo de verificação da assinatura, novamente, a função resumo deve ser aplicada sobre a mensagem e o receptor deve obter a chave pública do signatário para recuperar o resumo cifrado afim de compara-los. Um exemplo do processo de assinatura e verificação é ilustrado pela figura 2.2:

Quando mais de uma assinatura é gerada sobre o mesmo conjunto de dados, dá-se o nome de coassinaturas. Já o processo de assinar uma assinatura irá resultar em uma contra-assinatura.

2.3.1 Elementos Associados a uma Assinatura

Uma assinatura digital é o resultado da utilização de um algoritmo de assinatura, o qual faz uso da chave privada, sobre o hash do conteúdo a ser assinado. Porém, esta forma elementar de assinatura não permite a verificação de sua autenticidade nem sequer possibilita atestar que a assinatura foi realizada em determinada

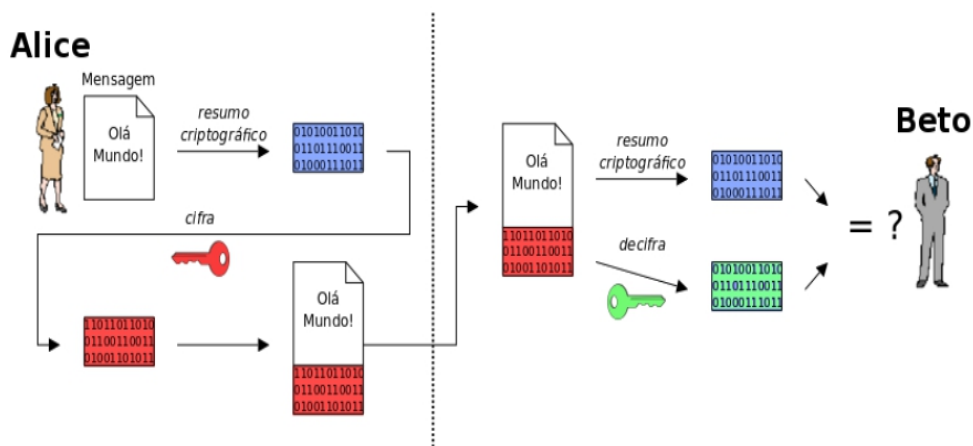


Figura 2.2: Assinatura Digital.(Imagem extraída de (Moecke))

período do tempo. Estas características e muitas outras podem ser adquiridas através da associação da assinatura a outros elementos.

Como elementos principais podemos citar:

- Certificado digital 2.5.1. Possibilita a verificação da autenticidade do signatário;
- Atributos assinados. São informações que, em conjunto com o documento a ser assinado, fazem parte do cálculo da assinatura e;
- Atributos não assinados. São informações relevantes que podem ser associadas a assinatura após o momento de sua criação sem que esta seja invalidada. Um exemplo de atributo não assinado é o carimbo do tempo, este permite que assinatura seja associada a uma data segura e também provar que ela ocorreu antes de determinado momento. Mais informações sobre carimbo do tempo podem ser encontradas na RFC 3161 (Adams et al. 2008).

2.3.2 Política de Assinatura

Uma política de assinatura, de acordo com a especificação ETSI TR 102 272 (Institute 2003), define um conjunto de regras utilizadas para a criação e validação de assinaturas digitais. Além disso, a política pode ser identificada de forma implícita

ou explícita. A forma implícita refere-se a identificar os atributos que caracterizam o tipo da assinatura. Por exemplo, se uma assinatura é formada por elementos que caracterizam uma política específica, conclui-se que a assinatura está em conformidade com esta política. Já a forma explícita exige que a ligação da assinatura com o identificador da política ocorra através da utilização deste identificador no cálculo da assinatura.

A política de assinatura pode ser especificada no formato livre ou estruturado. A forma livre refere-se a um formato legível por humanos, o que possibilita sua interpretação e avaliação no contexto em que foi aplicada. Já a forma estruturada é utilizada para codificar regras eletrônicas que facilitam os processos de geração e validação de assinaturas digitais.

Políticas de assinaturas devem conter regras que se aplicam a funcionalidade, que podem implicar no uso de políticas de certificação e relacionadas ao ambiente utilizado pelo signatário. Um exemplo de regras de funcionalidades, é a especificada no ETSI TR 102 272, através da política de verificação de assinatura, que possui regras para uso de provedores de serviço confiáveis e que definem os componentes da assinatura, fornecido pelo signatário, como requisitos para validação a longo term. As especificações ETSI TR 102 272 (Institute 2003) e ETSI TR 102 038 (Institute 2002) apresentam mais detalhes sobre políticas de assinatura.

2.4 Resumo Criptográfico

A FIPS PUB 186-3(NIST 1994), descreve função resumo como um processo que mapeia uma sequência de bits de tamanho arbitrário para uma sequência de comprimento fixo. Dessa forma, não importa o comprimento da informação, o resultado terá sempre o mesmo tamanho. Este resultado é conhecido como resumo criptográfico, impressão digital ou simplesmente *hash*. As funções resumo são projetadas para satisfazer as seguintes propriedades:

- Ser de difícil inversão. É computacionalmente inviável encontrar o conteúdo resumido a partir do resumo;

- Para entradas idênticas, devem ser geradas saídas idênticas.
- Resistentes a colisões. Para entradas distintas, deve ser improvável gerar uma mesma saída.

Assim, um resumo criptográfico pode ser utilizado para facilitar o processo de verificação da integridade dos documentos, melhorar a organização das informações auxiliando na indexação, dentre outros.

2.5 Infraestrutura de Chaves Públicas

A assinatura digital (ver seção 2.3), utiliza a chave para garantir integridade dos dados, não repúdio e autenticidade do signatário. Porém, a autenticidade e não repúdio, referem-se a chave pública estar associada a sua, correspondente, privada. É necessário contudo amarrar a chave privada de assinatura a uma entidade, o titular responsável pela chave. A tecnologia utilizada para este fim é o certificado digital, que pode estar baseada em uma cadeia de confiança hierárquica.

De acordo com Housley e Polk (2001), uma infraestrutura de chaves públicas (ICP) é projetada para facilitar o uso da criptografia assimétrica e possibilita construir uma cadeia de confiança hierárquica. A entidade autoassinada, topo desta hierarquia conhecida como Autoridade Raiz, certifica e pode transferir a tarefa de certificação para outras entidades, ditas intermediárias ou subordinadas e finais, mediante a utilização da assinatura digital. O processo de construção do caminho de confiança entre a Autoridade Raiz e a Entidade Final é conhecido como caminho de certificação que segundo (Steve e PKI 2002), deve ser construído e validado para estabelecer um ponto de confiança e verificar uma assinatura digital.

Uma infraestrutura de chaves pública não é formada apenas por soluções tecnológicas e certificados digitais, mas também por pessoas responsáveis por estabelecer regras operacionais e protocolos para a gestão de todos os sistemas da infraestrutura.

O certificado digital é um componente importante desta infraestrutura e será descrito de forma sucinta a seguir.

2.5.1 Certificado Digital

De acordo com a RFC 5280 (Cooper et al. 2008), o certificado digital contém propriedades como:

- A chave pública;
- Informações de uso da chave;
- Informações do detentor da chave;
- Informações do seu emissor;
- Assinatura digital do seu emissor. De acordo com Housley e Polk (2001), um certificado é protegido através da assinatura do seu emissor. Ou seja, ele é considerado uma mensagem assinada do ponto de vista do certificado que o emitiu. Além disso, esta assinatura permite atestar a confiança sobre este certificado.;
- Uma período de validade.

Assim, um certificado digital é um conjunto de informações que permitem identificar, de forma segura, o proprietário do par de chaves. Dessa forma, a realização de assinaturas digitais, utilizando certificados digitais, provê à assinatura características de autenticidade e não repúdio.

2.6 Considerações Finais do Capítulo

Este capítulo apresentou, de forma geral, os principais componentes envolvidos no processo de assinatura digital. Abordou também a forma mais elementar da assinatura digital. Vale ressaltar que, para que a assinatura digital seja aplicada com segurança, esta deve conter uma série de atributos, não detalhados neste capítulo. De

fato, existem várias propostas que visam a correta manutenção da integridade e autenticidade de documentos, implicando em diferentes formatos de assinatura digital, prevendo inclusive a manutenção por longo prazo. Os principais padrões de assinatura digital serão descritos no capítulo 3.

3 Padrões de Assinatura Digital

3.1 Introdução

Este capítulo apresentará os principais padrões de assinatura utilizados na atualidade. Estes padrões, direta ou indiretamente, fazem parte do padrão brasileiro de assinatura digital. Inicialmente, será exposto o padrão *Cryptographic Message Syntax*, que define uma sintaxe não só voltada à assinatura digital de conteúdos, mas para proteção de forma geral. Em seguida, será apresentado o padrão CAdES, que define diretrizes de utilização visando interoperabilidade, bem como adiciona suporte à preservação de assinaturas por longo prazo. Na sequência, será apresentado o padrão XMLDSig, padrão de assinatura representado em XML. Por fim, será abordado o padrão XAdES, evolução do XMLDSig que, como o CAdES, adiciona suporte à preservação de longa duração.

3.2 Cryptographic Message Syntax (CMS)

Cryptographic Message Syntax (CMS)(Housley 2009) do Internet Engineering Task Force (IETF) é um padrão que descreve uma sintaxe para proteção de dados, utilizando criptografia. Este padrão é baseado no PKCS#7 versão 1.5, publicado pelo RSA Data Security (RSA) como *RSA Laboratories Technical Note* em novembro de 1993 (Technical 1993). A proteção dos dados resulta de uma estrutura que contém vários campos, a utilização de determinados campos depende do tipo de proteção escolhido, este pode ser: *data*, *signed-data*, *enveloped-data*, *digested-data*, *encrypted-data*

e *authenticated-data*.

Para representar os diferentes tipos de estruturas, o pacote CMS contém, inicialmente, dois campos: *contentType* e o *content*. O primeiro define o tipo de conteúdo, enquanto o segundo consiste na estrutura correspondente ao tipo especificado. A estrutura CMS é representada utilizando um padrão formal para especificação de tipos de dados abstratos denominado ASN.1 (CCITT 1988) e, posteriormente, codificada no formato binário, utilizando BER (Basic Encoding Rules) (CCITT 1988) e DER (Distinguished Encoding Rules) (CCITT 1988). Este tipo de codificação permite a representação das informações na forma de um conjunto de octetos, que pode ser transmitido entre vários sistemas de diferentes arquiteturas.

No caso de assinaturas digitais, o identificador *contentType* irá receber o valor *signed-data*. A estrutura *signed-data* permite a representação de uma ou mais assinaturas digitais referentes ao mesmo documento, através do campo *signerInfos*. A figura 3.1 apresenta uma simplificação da estrutura CMS, para um conteúdo assinado.

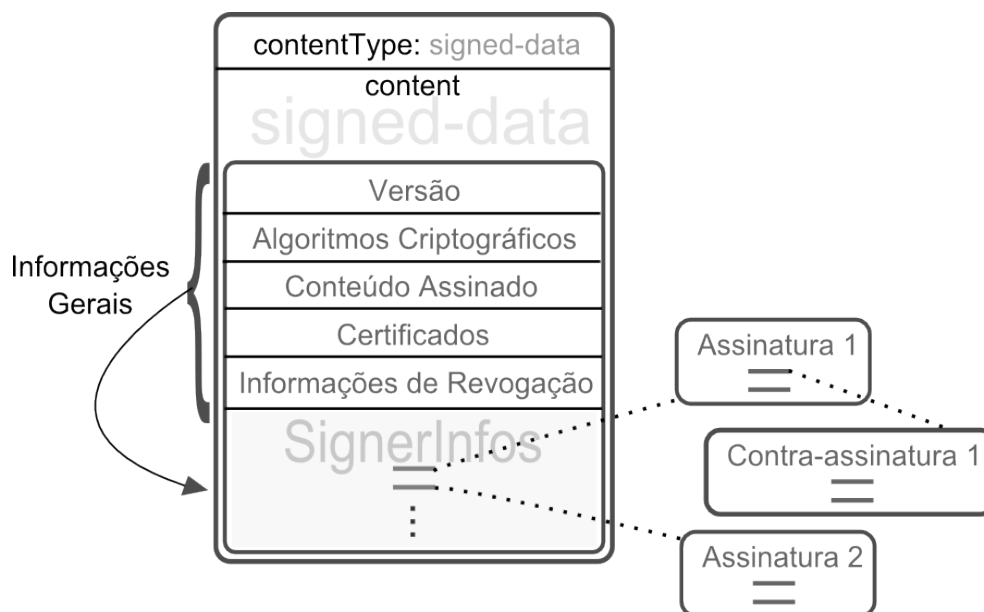


Figura 3.1: Estrutura CMS (*signed-data*).

Os campos agrupados como informações gerais armazenam dados referentes a todas as assinaturas, a começar pela versão da sintaxe CMS. Outras informações

disponíveis são os algoritmos de resumos criptográficos utilizados pelos signatários sobre o objeto da assinatura, o próprio conteúdo assinado (no caso de uma assinatura com o conteúdo anexado), uma coleção de certificados e uma coleção de informações de revogação.

Dentre as informações gerais, podem ainda estar presentes as cadeias de certificação dos certificados utilizados nas assinaturas, bem como as informações de revogação que podem comprovar a validade destas credenciais.

O campo contendo as assinaturas, *signerInfos*, pode conter zero ou mais estruturas que armazenam informações referentes a uma única assinatura. Ou seja, pode conter informações de diferentes signatários, as chamadas co-assinaturas ou assinaturas em paralelo. Cada co-assinatura pode ainda encapsular contra-assinaturas ou assinaturas em série. Estas assinaturas referem-se não ao conteúdo diretamente, mas sim à assinatura anterior, servindo como uma forma de reconhecimento de firma desta. A figura 3.2 detalha uma estrutura *SignerInfo*.

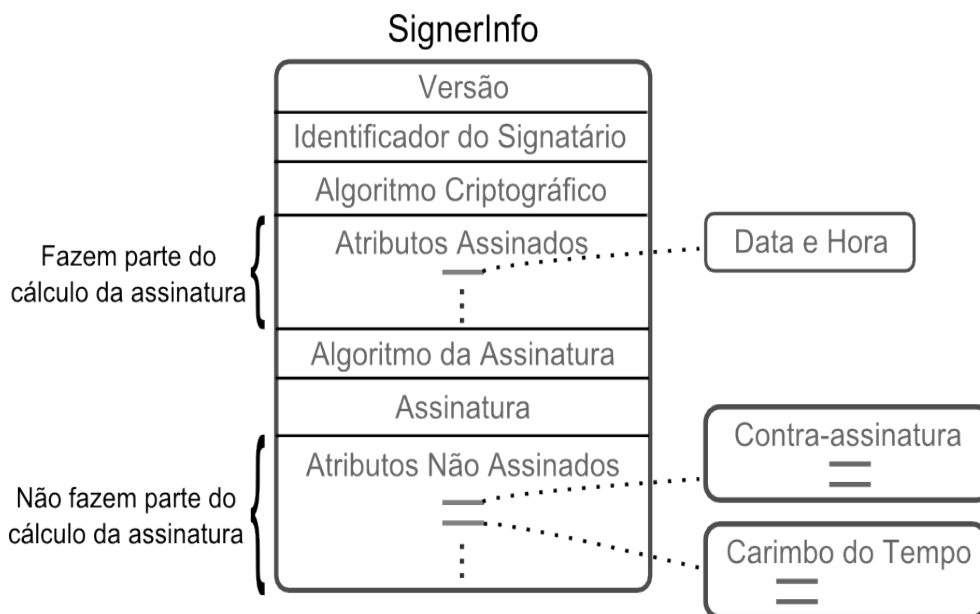


Figura 3.2: Estrutura CMS (*signerInfo*).

Tal como ilustrado na figura 3.2, são atributos de uma assinatura: a versão em que ela foi gerada, um identificador do algoritmo utilizado para calcular o resumo

criptográfico do conteúdo assinado e uma coleção de atributos assinados e não assinados. Por fim, a estrutura conta também com o identificador do algoritmo de assinatura e a assinatura propriamente dita.

Embora seja um padrão de assinatura bastante completo, falta ao CMS a definição de um conjunto de regras e extensões que permitem uma preservação a longo prazo. Esta deficiência foi sanada através da definição de um padrão avançado sobre o CMS, o CAAdES, descrito a seguir.

3.3 CMS Advanced Electronic Signatures (CAAdES)

O CMS Advanced Electronic Signatures (CAAdES) (INSTITUTE 2010), do European Telecommunications Standards Institute (ETSI) é um padrão de assinatura construído sobre o CMS (Housley 2009), adicionando a este a predefinição de um conjunto de atributos assinados e não assinados, bem como o uso de políticas de assinatura, as quais são utilizadas para definir como uma assinatura deve ser gerada e validada. Sua especificação técnica descreve diferentes tipos de assinaturas que podem ser utilizadas em diferentes cenários, enumerando características e práticas de uso. O núcleo dessa proposta é a preservação da integridade de informações por longo prazo.

3.3.1 Tipos de Assinatura

Os tipos de assinatura possíveis com esta especificação, são construídos a partir de um básico (CAAdES-BES), este constitui o núcleo de todos os demais tipos de assinatura. Após definido o formato base, é necessário especificar se a assinatura será baseada em política explícita (CAAdES-EPES) ou implícita. Ou seja, deve-se definir se a assinatura será do tipo (CAAdES-BES) ou (CAAdES-EPES) para depois fazer uso dos formatos que garantem validade a longo prazo, a saber: com carimbo de tempo (CAAdES-T), com referências ou dados de validação (CAAdES-C, CAAdES-X Long, CAAdES-X Type 1, CAAdES-X Type 2 e CAAdES-X Long Type 1 or 2) e de arqui-

vamento (CAAdES-A).

A figura 3.3 ilustra a relação entre os diferentes tipos de assinaturas definidas no padrão CAAdES, que serão descritos, resumidamente, nas seções a seguir.

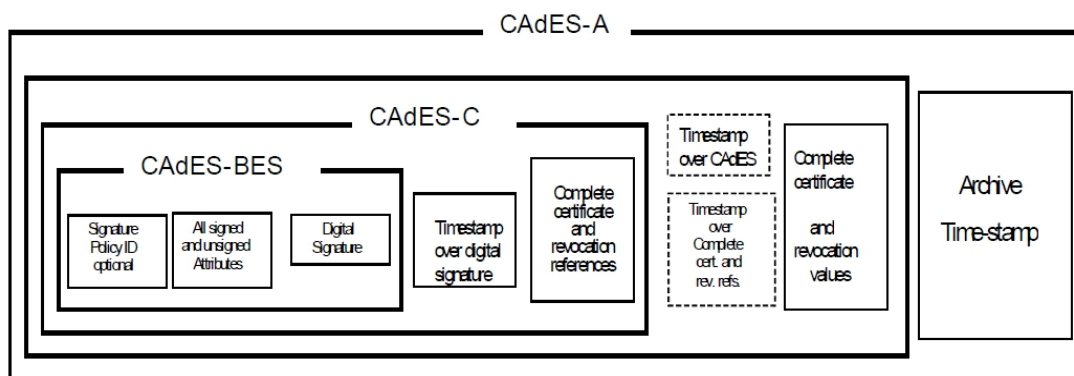


Figura 3.3: Tipos de Assinatura CAAdES. (Imagem extraída de (INSTITUTE 2010))

3.3.1.1 CAAdES Basic Electronic Signature (CAAdES-BES)

Esta forma de assinatura deve ser utilizada quando não existe a necessidade de confiar na data em que a assinatura foi gerada, pois fornece proteção apenas sobre a integridade e autenticidade do conteúdo. Dessa forma, este formato pode ser utilizado em processos de autenticação por exemplo, já que a validação da assinatura ocorre utilizando informações do momento atual.

Como componentes de sua estrutura podemos citar o dado (objeto que será assinado), os atributos assinados obrigatórios, os atributos assinados opcionais e a assinatura calculada sobre o dado e os atributos assinados. Dada a natureza de sua utilização, este formato possui os seguintes atributos assinados obrigatórios:

Content-type: Indica o tipo do conteúdo assinado;

Message-digest: Resumo criptográfico do dado, objeto da assinatura;

ESS signing-certificate-v1 ou v2: Armazena o certificado do signatário. Prevenindo ataques de substituição e reemissão.

O tipo básico é a assinatura mais simples que pode ser gerada seguindo a especificação CAAdES e é elementar para a construção dos demais tipos de assinatura.

3.3.1.2 CAAdES Explicit Policy-based ES (CAAdES-EPES)

As regras de conformidade de uma assinatura podem estar descritas de forma implícita ou explícita. A forma implícita compreende a identificação dos atributos que caracterizam o tipo da assinatura. Por exemplo, se uma assinatura é formada somente pelos elementos descritos no tipo básico 3.3.1.1, conclui-se que a assinatura é do tipo básica. Já a forma explícita consiste na adição de um novo atributo assinado que identifique, unicamente, a política de assinatura que deve ser usada no momento da validação. Em outras palavras, uma assinatura baseada em política explícita é aquela contendo um atributo que indique sob qual política esta deve ser validada. Ela pode conter ainda outros atributos necessários para atender os requisitos descritos na política.

3.3.1.3 Assinaturas com Dados de Validação

Como percebemos, o tipo de assinatura básico não fornece informações suficientes para uma validação a longo prazo, nem mesmo se a assinatura for baseada em política explícita. Para que esse tipo de validação seja possível, é necessário que a assinatura forneça dados como:

- Certificados relacionados à assinatura;
- Informações sobre o status de revogação para cada certificado;
- Informações sobre o momento em que a assinatura foi realizada. Através de um carimbo de tempo ou uma garantia externa provida por uma fonte confiável.

Estes dados possibilitam uma validação a longo prazo e permitem gerar os formatos que serão descritos a seguir.

3.3.1.4 Electronic Signature with Time (CADES-T)

Possuir característica de tempo confiável é o primeiro passo para prover uma validação a longo prazo. Uma assinatura com ligação a uma data segura, além de provar que ela ocorreu antes de determinado momento, permite sua validação mesmo após o certificado do signatário ter expirado. Esta característica pode ser adquirida através da inclusão de um carimbo do tempo ou por uma marca do tempo. O carimbo do tempo é adicionado como um atributo não assinado e permite que uma assinatura possa ser validada a qualquer momento no futuro, desde que o certificado da autoridade de carimbo do tempo seja considerado válido. Já a marca do tempo não é adicionada na assinatura e, é de responsabilidade do provedor de serviço do tempo confiável prover evidência da marca quando for solicitado.

Não é necessário que a característica do tempo confiável seja incluída no momento da construção da assinatura. Por se tratar de um atributo não assinado, este pode ser adicionado posteriormente. É fundamental, entretanto, que a marca temporal seja aposta enquanto o certificado do signatário e os certificados da cadeia de certificação do signatário estiverem válidos.

3.3.1.5 ES with Complete Validation Data References (CADES-C)

É possível que uma assinatura mantenha apenas referência para todos os certificados e informações de revogação necessários para sua validação, reduzindo assim, o tamanho da assinatura em termos de espaço físico, pois permite armazenar os valores das informações de validação em local externo à assinatura.

Idealmente, a assinatura com referência completa aos dados de validação deve ser construída sobre a que prove garantia de tempo confiável 3.3.1.4. Pois as informações de validação devem ser validadas contra uma determinada marca temporal. Logo, armazenar referencia a estas informações e não prover uma data confiável, impossibilita comprovar que a assinatura realmente ocorreu antes daquele período, nem se a marca temporal foi forjada para um momento em que as informações de validação não denunciassessem a invalidade das assinaturas.

Por estas referências de validação serem atributos não assinados, elas podem ser adicionadas posteriormente ao momento da assinatura pelo verificador. Assim, é possível obter informações de revogação mais precisas àquelas disponíveis no momento da realização da assinatura. O tempo de espera pelas melhores informações de revogação é chamado *Grace Period*. Este refere-se a um período de tempo que permite que as informações de revogação de um certificado se propaguem. Assim, caso um certificado seja revogado minutos antes da realização da assinatura, esta informação estará presente, apenas, na próxima lista emitida. Assim, a espera pelo fim do *Grace Period* para obtenção de referências de revogação adequadas é um fato crucial nos formatos que referenciam ou contém dados de validação.

3.3.1.6 Extended Electronic Signature Formats

Possuir ligação a uma data confiável e manter as informações de validação externos a assinatura, possibilita uma economia de espaço, porém estas informações devem estar sempre acessíveis no momento da validação. O formato de assinatura com referência completa aos dados de validação 3.3.1.5 permite validação a longo prazo de forma online, uma vez que deve haver a validação de dados armazenados externamente.

Afim de possibilitar uma proteção a longo prazo de forma offline, o formato CADES-C 3.3.1.5 pode ser estendido para:

Extended Long Electronic Signature (CADES-X Long): Permite evitar que os dados das informações se percam, pois, além das referências, os dados propriamente ditos são adicionados na assinatura. Apesar de demandar mais espaço, garante a disponibilidade dos dados de validação;

Extended Electronic Signature with Time Type 1 (CADES-X Type 1): Adiciona um carimbo de tempo sobre toda a assinatura. Permite proteção da integridade e fornece uma marcação de tempo confiável. Possibilita também proteção aos dados de validação, por exemplo, contra o comprometimento da chave da Autoridade Certificadora;

Extended Electronic Signature with Time Type 2 (CAAdES-X Type 2): Inclui um carimbo do tempo sobre as referências dos certificados e informações de revogação. Difere do formato anterior, pois provê proteção, apenas, aos dados de validação;

Extended Long ES with Time (CAAdES-X Long Type 1 or 2): É a combinação do formato CAAdES-X Long com um dos dois formatos CAAdES-X Type 1 ou CAAdES-X Type 2. Além de fornecer proteção sobre as referências dos dados de validação, também garante sua disponibilidade.

3.3.1.7 Archival Electronic Signature (CAAdES-A)

O formato de arquivamento oferece proteção completa a todos os dados da assinatura, através da adição de um carimbo de tempo sobre o (CAAdES-X) Long ou (CAAdES-X Long Type 1 ou 2). Assim sendo, toda a assinatura fica condicionada à validade do carimbo de tempo, que pode ser renovado com a inclusão de novos carimbos e assim sucessivamente.

Os formatos de assinatura CMS e CAAdES possuem equivalentes no formato XML os quais serão descritos a seguir.

O CAAdES-A pode ser realizado sobre formas mais simples, não necessariamente sobre as formas X.

3.4 XMLDSig e XML Advanced Electronic Signatures

O formato XML Signature (XMLDSig) (IETF/W3C), originalmente publicado como *W3C Recommendation* pelo IETF/World Wide Web Consortium (W3C), *XML Signature Working Group*, é uma recomendação que especifica a sintaxe e regras de assinaturas digitais XML. Atualmente encontra-se em sua segunda edição (IETF/W3C), publicada como *W3C Recommendation 10 June 2008* e mantida pelo *W3C XML Security Specifications Maintenance Working Group*.

A assinatura XML pode ser aplicada sobre qualquer tipo de dado, incluindo

binários. Além disso, este dado pode ser uma referência externa à assinatura, pode estar envolvido por ela ou, no caso do documento ser um XML, este pode incluir a assinatura como um elemento de sua estrutura. Para representar estas funcionalidades, o termo utilizado na especificação para o primeiro caso é *detached*, para o segundo, *enveloping* e *Enveloped* para o terceiro.

O XMLDSig é equivalente ao CMS, ou seja, não possui características para preservação a longo prazo. Para que isso seja possível, deve-se fazer uso do XML Advanced Electronic Signatures (XAdES) (INSTITUTE 2010), equivalente ao CAdES. Os formatos de assinaturas em XML serão abordados somente de forma sucinta, pois estão fora do escopo deste trabalho.

3.5 Considerações Finais

Este capítulo apresentou quatro padrões que possibilitam associar elementos a uma assinatura para prover melhor proteção dos dados. Percebeu-se que o CMS e o XMLDSig, são formatos equivalentes, o primeiro é codificado em binário, enquanto o segundo detém de todas características da forma de representação em XML. Estes formatos, apesar de permitirem o uso do carimbo do tempo, não possibilitam uma preservação a longo prazo. Faz-se necessário utilizar modelos que provejam esta funcionalidade, encontrada nos padrões avançados CAdES e XAdES.

O capítulo seguinte apresenta a Infraestrutura de Chaves Públicas Brasileira e mostra os formatos de assinaturas digitais aprovados no âmbito da ICP-Brasil.

4 Padrão Brasileiro de Assinatura Digital

4.1 Introdução

A Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil), foi instituída pela medida provisória 2200-2, de 28 de Junho de 2001 (BRASIL 2001), para garantir a integridade, a autenticidade e a validade jurídica de documentos em formato eletrônico. Assim, a Infraestrutura de Chaves Públicas Brasileira é a instituição que regulamenta os critérios para a confiança e aceitação de documentos eletrônicos.

Sua estrutura é formada por uma hierarquia e uma autoridade gestora de políticas, intitulada Comitê Gestor da ICP-Brasil, com a responsabilidade de governar a infraestrutura. São responsabilidades do comitê gestor: estabelecer regras operacionais, diretrizes, licenciar e autorizar o funcionamento dos componentes da hierarquia, dentre outras. A hierarquia conta com um ponto máximo de confiança, a Autoridade Certificadora Raiz (AC Raiz), Autoridades Certificadoras Subordinadas (ACs) e Autoridades de Registro (ARs). A AC Raiz é o topo da hierarquia de certificação (vide figura 4.1), gerenciada pelo Instituto Nacional de Tecnologia da Informação, entidade incumbida de executar atividades relacionadas à gestão das autoridades. São responsabilidades do ITI: auditar, fiscalizar, gerenciar o ciclo de vida da AC-Raiz, credenciar ACs de segundo nível, dentre outras.

Uma Autoridade Certificadora subordinada é a entidade credenciada, direta ou indiretamente, à AC Raiz, com a responsabilidade pelo gerenciamento dos

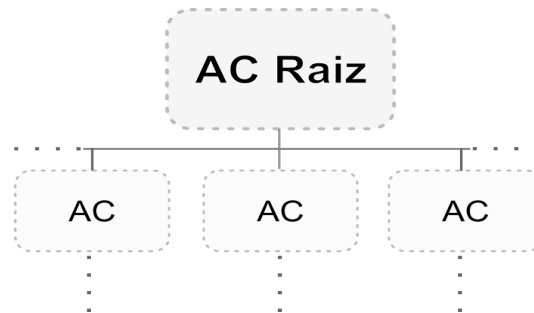


Figura 4.1: Arquitetura hierárquica da ICP-Brasil.

certificados por ela emitidos, bem como pelos processos de distribuição e revogação destes certificados. As Autoridades de Registro, por sua vez, são entidades vinculadas a uma determinada AC, com a finalidade de identificação e registro dos usuários que fazem uso dos serviços de certificação.

Este capítulo apresenta uma visão geral sobre a estrutura normativa da ICP-Brasil e descreve os formatos de assinatura digital aprovados em seu âmbito.

4.2 Estrutura normativa

A Infraestrutura de Chaves Públicas Brasileira contém um conjunto de documentos normativos, classificados como: DOC-ICP, ADE-ICP, MCT, dentre outros de caráter informativo. Cada DOC-ICP consiste em um conjunto de normas técnicas e procedimentais a serem seguidas, voltadas a determinado componente da infraestrutura. Os DOC-ICPs são instituídos por um conjunto de resoluções, decisões tomadas pelo Comitê Gestor da ICP-Brasil, cada DOC-ICP é representado por um identificador numérico, como por exemplo: DOC-ICP-nn e DOC-ICP-nn.mm. O primeiro é tratado como documento principal e o segundo como seu complemento. O documento principal precisa ser aprovado pelo Comitê Gestor sempre que houver modificações, emitindo uma Resolução de aprovação. Já os documentos complementares, após aprovados uma vez, não precisam mais de aprovação do Comitê Gestor. Normalmente estes são mais técnicos que os principais, trazendo mais detalhes sobre as regras e im-

plementações. A tabela 4.1 mostra alguns documentos atualmente publicados no (*site*) do ITI.

<i>DOC-ICP</i>	<i>Assunto</i>
DOC-ICP-01	Declaração de Práticas de Certificação da Autoridade Certificadora Raiz da ICP-Brasil - v.4.1.
DOC-ICP-01.01	Padrões e Algoritmos Criptográficos da ICP-Brasil - v.2.2.

Tabela 4.1: Exemplo - DOC-ICP.(*Extraído do (site) do ITI (Informação).*)

O ADE-ICP corresponde a um adendo, vinculado a um DOC-ICP, que contém formulários e outros elementos passíveis de alterações frequentes. Sua classificação é semelhante a de um DOC-ICP, ADE-ICP-nn.aa e ADE-ICP-nn.mm.aa, porém não utiliza apenas números. O ADE-ICP-nn.aa faz referência ao DOC-ICP-nn e o ADE-ICP-nn.mm.aa ao DOC-ICP-nn.mm. Outro tipo de documento da estrutura normativa é o Manual de Condutas Técnicas (MCT), documento que detalha as especificações técnicas necessárias para homologação de componentes tecnológicos sensíveis, como o software que gerencia os certificados, o hardware que protege as chaves privadas da infraestrutura, entre outros. Os MCTs são utilizados junto aos LEAs (Laboratório de Ensaios e Auditoria) como base para as homologações da conformidade dos equipamentos e aplicações relacionadas aos processos de certificação digital na ICP-Brasil. A tabela 4.2 mostra alguns documentos atualmente vigentes.

<i>ADE e MCT</i>	<i>Assunto</i>
ADE-ICP-01.A v.1.0	Formulário revalidação dos dados cadastrais e solicitação de novo certificado
ADE-ICP-10.07.A - v1.0	Formulário de credenciamento de LEA
MCT 3	Requisitos Materiais Documentos Tokens
MCT 4	Requisitos Materiais Documentos para Softwares de Assinatura

Tabela 4.2: Documentos técnicos gerenciados pela ICP-Brasil

Dentre os demais documentos da ICP-Brasil, estão os que regulamentam a geração e verificação de assinaturas digitais. Os DOC-ICP-15 (ITI) e seus complemen-

tos (vide tabela 4.3) descrevem as características técnicas necessárias para a utilização de assinaturas digitais no âmbito da ICP-Brasil. Nesse sentido, o DOC-ICP-15 exhibe uma visão geral sobre assinaturas digitais, além de resgatar conceitos de criptografia assimétrica, resumos criptográficos e o processo de assinatura digital, já vistos na seção 2. A tabela 4.3 apresenta o conjunto do DOC-ICP voltados ao padrão brasileiro de assinatura digital.

<i>DOC-ICP</i>	<i>Assunto</i>
DOC-ICP-15	Visão Geral sobre Assinaturas Digitais na ICP-Brasil.
DOC-ICP-15.01	Requisitos Mínimos para Geração e Verificação de Assinaturas Digitais na ICP-Brasil.
DOC-ICP-15.02	Perfil de uso geral para assinaturas digitais na ICP-Brasil.
DOC-ICP-15.03	Requisitos das políticas de assinatura digital na ICP-Brasil.

Tabela 4.3: Normatização da Assinatura Digital

De acordo com o DOC-ICP-15, a ICP-Brasil permite o uso dos formatos de assinatura CADES (INSTITUTE 2010) e XAdES (INSTITUTE 2010). Como exposto na seção 3, estes formatos possibilitam o uso de uma série de atributos ou propriedades que tornam a geração de uma assinatura mais flexível. Essa flexibilidade, entretanto, pode resultar em baixa interoperabilidade entre os sistemas, devido à possibilidade de seleção de propriedades arbitrárias ou mesmo proprietárias para confeccionar uma assinatura. Assim, foi definido um perfil para uso geral (ITI), construído a partir da seleção de um subconjunto de atributos do formato CADES e propriedades do XAdES. Este perfil mostra os principais atributos ou propriedades que podem ser utilizados no momento da confecção de uma assinatura, tendo o objetivo de padronizar o uso desses atributos, primando assim pela interoperabilidade.

Além disso, foi estabelecido que as assinaturas devem ter uma política explícita, tornado regra pelo DOC-ICP-15.03 (ITI). Ou seja, os formatos de assinatura ICP-Brasil baseados em CADES e XAdES possuem política explícita (CADES-EPES e XAdES-EPES), que além de definir o subconjunto de características desse perfil geral, definem a forma como a assinatura deve ser criada e verificada. O uso de políticas

é obrigatório e permite uma proteção adequada, de acordo com os requisitos de cada processo. A figura 4.2 exibe a relação entre os documentos da ICP-Brasil e os padrões internacionais de assinatura.

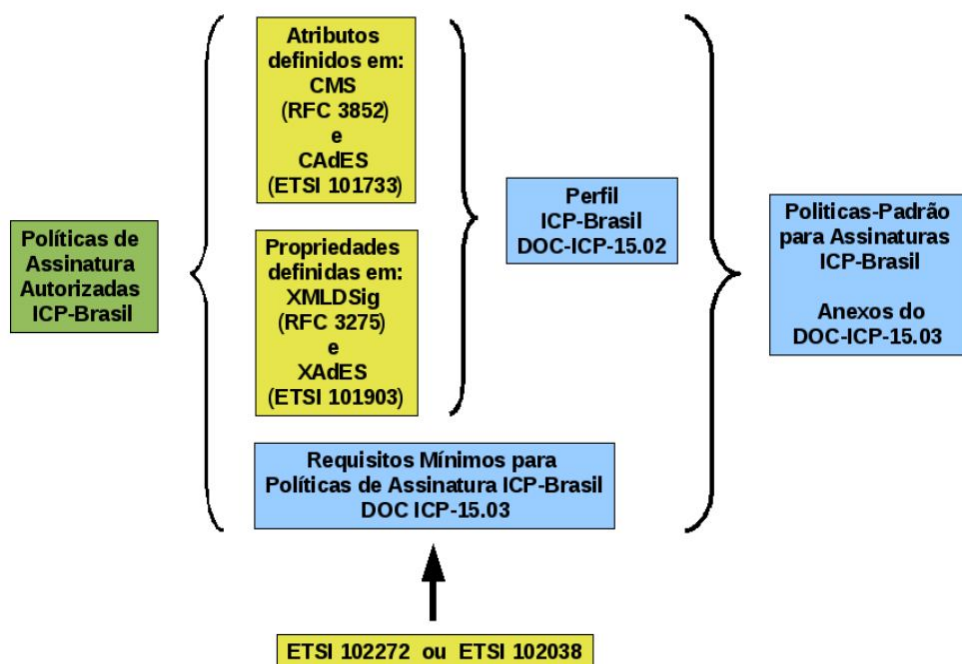


Figura 4.2: Relação entre os documentos ICP-Brasil e os formatos de assinatura. (Extraído do DOC-ICP-15 (ITI).)

O documento que fornece os requisitos técnicos para geração e verificação de assinaturas através da utilização de políticas explícitas é o DOC-ICP-15.01 (ITI). Já o DOC-ICP-15.03 (ITI), aborda os requisitos de cada política. Esses documentos apresentam também as diferentes formas aprovadas pela ICP-Brasil, para assinar documentos digitalmente. Estas formas, organizadas em políticas de assinatura, são apresentadas na seção 4.3.

4.3 Políticas de Assinatura Digital

As políticas de assinaturas digitais permitidas na ICP-Brasil são cinco: assinatura digital com Referência Básica (AD-RB), com Referência do Tempo (AD-RT),

com Referências para Validação (AD-RV), com Referências Completas (AD-RC) e com Referências para Arquivamento (AD-RA). Cada política encapsula características de sua antecessora, sendo a AD-RB o ponto de partida para todas as demais, conforme mostra a figura 4.3.

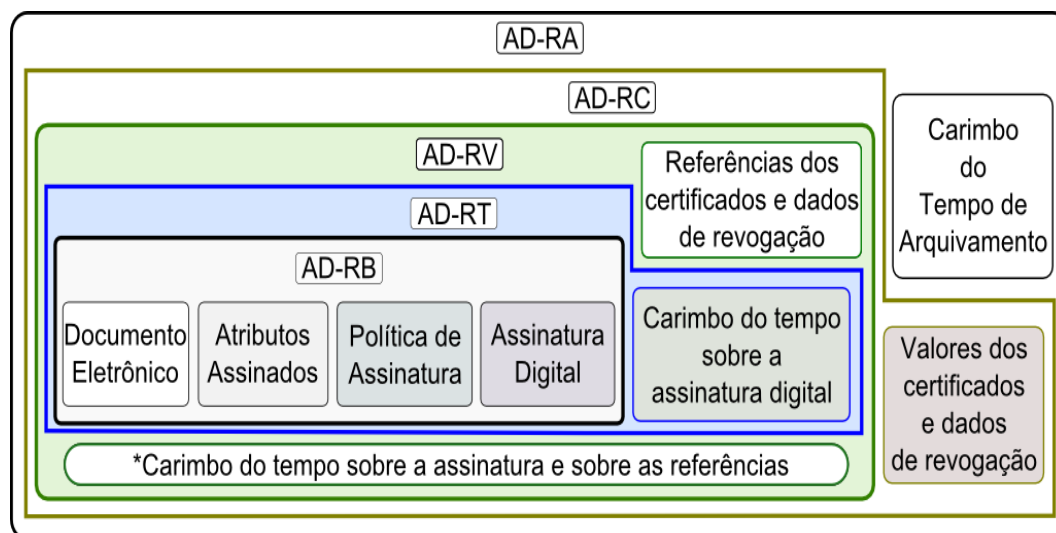


Figura 4.3: Formatos de Assinatura Digital ICP-Brasil. (*Opcional para a política AD-RA)

Como pode ser observado na figura 4.3, os formatos subsequentes detêm todas as características dos anteriores e outras adicionais, que os diferenciam do antecessor. Os cinco formatos podem ser utilizados tanto para CADES quanto para XAdES. As seções 4.3.1 a 4.3.5 apresentam mais detalhes acerca de cada política de assinatura, mostrando as diferenças na forma como estas são implementadas nos padrões CADES e XAdES.

4.3.1 AD-RB

A utilização do formato de assinatura AD-RB, equivalente ao CADES-EPES e XAdES-EPES, ocorre quando não é necessário uma validação a longo prazo da assinatura pois esta não possui características de tempo confiável. Assim, este formato deve ser utilizado, apenas quando não for necessário prover uma validação após o prazo de validade do signatário. Ou seja, este formato permite apenas a validação da

assinatura enquanto o certificado do signatário e os certificados da cadeia de certificação do signatário estiverem válidos. Ou seja, a assinatura é validada utilizando a data atual, caso o certificado tenha sido revogado após a o momento da assinatura, esta será considerada inválida. Este formato exige os atributos exibidos na tabela 7.3.

<i>Atributos</i>	<i>Representação em CAdES</i>
Tipo do Conteúdo	<i>id-contentType.</i>
Resumo criptográfico da mensagem	<i>id-messageDigest.</i>
Informações sobre o signatário	<i>id-aa-signingCertificate(V2).</i>
Identificador da política	<i>id-aa-ets-sigPolicyId.</i>

Tabela 4.4: Atributos exigidos pela política de assinatura AD-RB

Estes atributos são elementos que fazem parte do cálculo da assinatura, ou seja são atributos assinados (vide seção 3.3.1.1). O tipo do conteúdo refere-se ao modo como o objeto da assinatura deve ser codificado, caso este faça parte da assinatura, geralmente em uma sequência de bytes *Octet Strings*. As informações do signatário resultam de uma estrutura que contém o seu número de série e resumo criptográfico mais o nome do certificado emissor. Por último o identificador da política contém informações que identificam a política de assinatura como o resumo criptográfico e o endereço da política.

4.3.2 AD-RT

Caso seja necessário que a assinatura possua referências temporais deve-se optar por outro formato, ao iniciar pelo AD-RT. O formato com referência do tempo (AD-RT), equivalente ao CAdES-T e XAdES-T, é construído sobre o com referência básica e possui ligação a uma data segura provida pelo uso do carimbo do tempo como mostra a tabela 4.5. A referência temporal não é só importante quando se deseja saber a data, mas também traz a segurança de que uma posterior revogação ou expiração não comprometam a assinatura realizada, desde que a data da assinatura seja anterior ao evento.

<i>Atributos</i>	<i>Representação em CAdES</i>
<i>Atributos AD-RB</i>	
Carimbo do tempo sobre a assinatura	<i>id-aa-signatureTimeStampToken.</i>

Tabela 4.5: Atributos exigidos pela política de assinatura AD-RT

Esta referência do tempo pode ser adicionada a qualquer momento na estrutura da assinatura sem interferir em sua integridade, exceto caso o certificado seja considerado inválido. Nesse sentido, a assinatura é associada a um referência temporal e este formato de assinatura deve ser utilizada caso necessite provar que ela ocorreu antes de um determinado momento e verificar a assinatura mesmo após o período de validade do certificado do signatário, desde que o carimbo do tempo seja considerado válido.

Apesar de prover uma data confiável as informações de validação estão externas a assinatura e muitas vezes os dados de revogação não são considerados os mais adequados para validar a assinatura (vide problema do *Grace Period* na seção 3.3.1.5) ou estes podem estar indisponíveis devido a fatores como indisponibilidade da rede, cadeia de certificação expirada, dentre outros.

4.3.3 AD-RV

Para que uma assinatura possa referenciar inequivocamente os dados de validação, deve-se utilizar outro formato de assinatura, a começar pelo AD-RV.

Uma assinatura com Referências para Validação (AD-RV), equivalente ao CAdES-C e XAdES-C, detém todas as características da com carimbo do tempo. Soma-se a sua estrutura referências para os dados de validação e um carimbo do tempo sobre estas referências, estes dados são considerados obrigatórios por esta política de assinatura conforme mostra a tabela 4.6.

Armazenar as referências de validação permite que uma assinatura possa ser associada inequivocamente ao conjunto de informações necessárias para a sua validação e estes sejam adquiridos de forma mais eficiente. Estas referências por serem

<i>Atributos</i>	<i>Representação em CADES</i>
<i>Atributos AD-RT</i>	
Referências completas aos certificados	<i>id-aa-ets-certificateRefs.</i>
Referências completas aos dados de revogação	<i>id-aa-ets-revocationRefs.</i>
Carimbo do tempo sobre as referências e a assinatura	<i>id-aa-ets-escTimeStamp.</i>

Tabela 4.6: Atributos exigidos pela política de assinatura AD-RV

atributos não assinados, podem ser adicionados posteriormente ao momento da assinatura, o qual possibilita o tempo de espera pelas melhores informações de revogação *Grace Period*. As referências completas aos certificados dizem respeito a toda cadeia de certificação, exceto o certificado do signatário. O número de série do certificado, o resumo criptográfico e o nome do certificado emissor é o que constitui a informação de cada certificado da cadeia. As referências completas aos dados de revogação são informações sobre dados como as listas de certificados revogados de cada certificado. Uma referência de validação contém dados como: o resumo criptográfico da lista, o nome do emissor e a data de emissão da lista. Após a inclusão das referências de validação a política AD-RV exige que estas sejam associadas a um tempo seguro através da utilização do carimbo do tempo. Ou seja, este formato de assinatura contém dois carimbos do tempo, um sobre a assinatura e outro sobre as referências de validação e a assinatura. O carimbo sobre as referências de validação protege contra o comprometimento de algum certificado da cadeia de certificação do signatário e amarra estas informações à assinatura. Dessa forma este tipo de assinatura deve ser utilizado quando é necessário verificar a assinatura a qualquer momento, desde que os dados de validação referenciados estejam disponíveis.

Possuir referências para os dados de validação possibilita uma economia de espaço, já que os valores encontram-se externos à assinatura. Porém, nem sempre - devido a razões como falta de conectividade, perda dos dados, dentre outras - as informações de validação estão disponíveis o que impossibilita uma validação a longo prazo.

4.3.4 AD-RC

Para evitar que esses tipos de problemas ocorram, pode-se fazer uso do formato de assinatura com referências completas (AD-RC). O formato de assinatura AD-RC, equivalente ao CAdES-X e XAdES-X, além de armazenar as referências de validação exige que a assinatura encapsule os valores destas referências conforme mostra a tabela 4.7.

<i>Atributos</i>	<i>Representação em CAdES</i>
<i>Atributos AD-RV</i>	
Valores dos certificados	<i>id-aa-ets-certValues.</i>
Valores de revogação	<i>id-aa-ets-revocationValues.</i>

Tabela 4.7: Atributos exigidos pela política de assinatura AD-RC

Assim, apesar de exigir mais espaço, este formato garante que os dados estejam sempre disponíveis e deve ser utilizado quando existe a necessidade que a assinatura contenha os dados de validação na própria estrutura o que possibilita uma maior proteção a longo prazo. Como este formato é construído sobre o AD-RV, possui uma data associada a assinatura e um carimbo aplicado sobre as referências para evitar o comprometimento de alguma autoridade certificadora da cadeia de certificação do signatário.

4.3.5 AD-RA

Caso seja necessário realizar um arquivamento dos dados assinados, é importante considerar que o estado dos algoritmos criptográficos utilizados para compor a assinatura sofre influência do tempo. Assim, a garantia de arquivamento esta ligada ao nível de segurança dos algoritmos que com o passar do tempo perde a força. O formato de assinatura com referências para arquivamento (AD-RA), equivalente ao CAdES-A e XAdES-A, construído sobre o AD-RC, porém não faz uso do carimbo do tempo sobre as referências. Este formato deve ser utilizado quando é necessário arquivar documentos a longo prazo mantendo aspectos como integridade, autenticidade

e confidencialidade. Para isso, faz uso de um carimbo do tempo para proteger toda a assinatura e este pode ser protegido com a aplicação de outros carimbos este ilustrado pela tabela 4.8.

<i>Atributos</i>	<i>Representação em CADES</i>
<i>Atributos AD-RC, exceto id-aa-ets-escTimeStamp</i>	
<i>Carimbo do tempo de arquivamento</i>	<i>id-aa-ets-archiveTimestampV2.</i>

Tabela 4.8: Atributos exigidos pela política de assinatura AD-RA

Desta forma, caso o carimbo aplicado com o passar do tempo seja considerado frágil - possui um tamanho de chave fraco por exemplo, pode-se aplica um outro carimbo do tempo com tamanho de chave maior sobre tudo e assim sucessivamente. Importante mencionar que o carimbo é realizado sobre toda a assinatura, incluindo os atributos não assinados (e inclusive os carimbos de arquivamento anteriores).

4.4 Considerações Finais do Capítulo

Este capítulo abordou a estrutura normativa da ICP-Brasil e os formatos de assinatura digital atualmente permitidos por ela. A Infraestrutura de Chaves Públicas Brasileira contém um conjunto de documentos normativos (DOC-ICP) que refletem as resoluções vigentes da ICP-Brasil e as tornam mais organizadas, melhorando sua compreensão. Estes documentos fornecem as informações necessárias sobre os formatos de assinatura digital. Os formatos são baseados em política explícita, podendo serem implementadas tanto no padrão CADES quanto no XAdES. O uso dessas políticas provê um guia para que cada parte saiba como gerar e validar de forma correta as assinaturas digitais presentes no documento, com vista a obter proteção adequada.

O capítulo seguinte apresentará o formato de documento portátil PDF e a forma como este padrão suporta assinatura digital.

5 Formato de Documento Portátil

5.1 Introdução

A Adobe Systems desenvolveu, em 1993, o formato de documento portátil (*PDF*). Em 2007 a especificação deste formato foi descontinuado e preparada para transformar-se em norma ISO, o que ocorreu de fato no ano seguinte e resultou a ISO 32000-1:2008 (Standard 2008). A versão base do *PDF* descrita nesta norma é a 1.7 porém esta encapsula funcionalidades das versões 1.0 a 1.6. O principal objetivo do *PDF* é permitir a comunicação de material visual entre diferentes sistemas e aplicações, independentemente do ambiente que ele foi criado. Ou seja, idealmente o conteúdo contido neste documento portátil deve ser exibido de forma idêntica em diferentes máquinas e sistemas operacionais.

A estrutura básica de um documento PDF é derivada do *PostScript* que é uma linguagem de descrição de páginas que possui a função principal de descrever a aparência do texto, dos vetores, das imagens, dentre outros, independente do dispositivo utilizado. A estrutura do *PostScript* (Incorporated 1999) precisou sofrer algumas variações ao ser integrada ao *PDF* afim de melhorar a performance. Atualmente, existem diversas bibliotecas para criação e manipulação de arquivos PDF. Pode-se gerar um documento portátil diretamente em um editor de arquivos PDF ou obtê-lo através da conversão de outros formatos, o que contribui para que ele seja muito utilizado para troca de documentos eletrônicos.

O *PDF* permite adicionar restrições de segurança no documento para que o autor possa ter controle sobre ele, permite também a edição de forma colabora-

tiva assim como ser exibido por aplicações web de forma parcial, sem precisar carregar o documento inteiro. Além disso, conforme especificado na ISO 32000-1:2008 (Standard 2008), este formato de documento pode armazenar um vasto conjunto de objetos em sua estrutura como texto, imagem, vídeos, anotações, assinaturas digitais, etc.

Deter o conhecimento total destas funcionalidades é extremamente importante para a análise do impacto da adoção deste padrão como meio de representação de documentos eletrônicos seguros, ou seja, assinados digitalmente. Por ser um padrão extremamente abrangente e complexo, este capítulo limitar-se-á às características que têm impacto direto no modo como é realizada uma assinatura digital neste tipo de documento.

5.2 Estrutura Básica

Um documento PDF pode ser interpretado como uma árvore de objetos, que tem como primeiro nó uma estrutura chamada de catálogo do documento. Este catálogo permite acesso a todos os objetos o que inclui as páginas, o número de páginas, o conteúdo das páginas, informações de como o conteúdo deve ser exibido, etc.

O PDF possui oito tipos básicos de objetos: booleanos, numéricos, literais, nomes, objeto nulo, arrays, dicionários e streams. Os três últimos são considerados containers e podem abrigar outros objetos resultando em tipos complexos. Os objetos de um arquivo PDF, em sua maioria, são *containers*. A identificação de um objeto ocorre através de um rótulo, o que permite que os mesmos sejam referenciados por outros objetos. Quando isso ocorre o objeto rotulado passa a ser considerado um objeto indireto, conceito muito utilizado na navegação entre a estrutura do documento. Estes objetos estão localizados por toda a estrutura do *PDF*, que inicialmente é constituída por quatro partes, conforme ilustra a tabela a seguir:

Cabeçalho: identifica a versão do PDF. O valor *%PDF-1.4*, por exemplo, indica que o arquivo está em conformidade com a versão 1.4. É importante mencionar que as

Cabeçalho
Corpo
Tabela de Referência Cruzada
Trailer

Tabela 5.1: Estrutura básica de um arquivo PDF

características de uma versão anterior são suportadas pelas versões posteriores, podendo não ser pelas anteriores;

Corpo: contém uma sequencia de objetos indiretos, estes objetos representam os componentes do documento como fontes, paginas, imagens etc.

Tabela de Referência Cruzada: mostra a localização dos objetos no *Corpo*, indica se o mesmo está ativo ou inativo e permite o acesso aleatório a eles. Uma característica importante desta tabela é o modo como ela informa a localização dos objetos. Cada objeto possui uma posição no arquivo, e esta posição é demarcada pela distância (em bytes) entre o início do documento e o início do objeto. Em um documento *PDF* pode-se encontrar esta tabela procurando pelo identificador *xref*, que será seguido por números que indicam a localização dos objetos. Um exemplo de uma entrada desta tabela é *0000000322 00000 n*, o número *0000000322* indica que o objeto está na posição 322, o *00000* é um número serial marcando a versão do objeto, inicialmente 0 e alterada sempre que o mesmo é atualizado. Por último, o *n* informa que o objeto é um objeto ativo, sendo marcado como *f*, caso contrário. Um objeto inativo não será exibido pelo leitor de arquivos PDF.

Trailer: informa a localização da *Tabela de Referência Cruzada*, do dicionário de catalogo do documento e pode referenciar alguns objetos do *Corpo*. Caso existam atualizações incrementais, a *Tabela de Referência Cruzada* da revisão anterior também faz parte do conjunto de informações do *Trailer*.

5.3 Atualização

Muitas vezes é necessário alterar um documento ou até mesmo incluir uma nova funcionalidade não prevista para a versão do PDF, descrita no cabeçalho. Para tratar esses casos, pode-se adicionar uma entrada no dicionário de catálogo do documento, informando da nova versão. Esta tem prioridade sobre a descrita no cabeçalho. Essa possibilidade permite que o arquivo possa evoluir ao longo do tempo e trabalhar com funcionalidades que no momento de sua criação sequer existiam.

A atualização de um documento, por exemplo pela simples adição de um objeto, implica na alteração não só do corpo do documento, mas pode afetar a posição de todos os objetos subsequentes. A tabela de referência cruzada deverá ser alterada para refletir a mudança e, conseqüentemente, sua localização também deverá ser atualizada no *trailer*.

Alterar a estrutura de um documento muitas vezes não é interessante, pois uma pequena alteração pode exigir a reescrita do arquivo inteiro. Para evitar esse alto custo e deixar a estrutura original intacta, o padrão PDF suporta o conceito de “revisões”, encapsulamentos que permitem a atualização incremental de um documento, através da adição de novo conteúdo ao final do arquivo.

Para comportar o novo conteúdo é necessário criar uma nova estrutura: corpo, tabela de referência cruzada e trailer. Esta nova estrutura, que não irá conter um cabeçalho, tem a mesma função da primeira, com exceção do *trailer*, que além de referenciar a tabela cruzada desta revisão irá também referenciar a antiga. Este processo pode-se repetir inúmeras vezes e é ilustrado pela tabela 5.2.

5.4 Assinatura Digital

O documento *PDF* permite a aplicação de assinatura digital para a verificação da autenticidade do documento, bem como sua autoria. Para que isso seja possível, é necessário armazenar informações sobre o signatário e o estado do documento. Isto

Cabeçalho
Corpo
Tabela de Referência Cruzada
Trailer
Corpo (2)
Tabela de Referência Cruzada (2)
Trailer (2)
...
Corpo (n)
Tabela de Referência Cruzada (n)
Trailer (n)

Tabela 5.2: Atualização incremental de um arquivo PDF.

é realizado com o auxílio de uma estrutura chamada dicionário de assinatura. A assinatura pode ocorrer através da utilização da assinatura digital ou através de biometria. A forma de validação de uma assinatura deve ser através de um módulo especial, chamado *signature handler*. Este módulo encontra-se integrado aos leitores de arquivos *PDF*, sendo possível também a sua extensão, por terceiros, na forma de plug-ins, afim de verificar ou gerar uma assinatura com características específicas.

As entradas principais do dicionário de assinatura são:

Filter: Deve conter o nome do módulo principal que irá realizar a verificação da assinatura (*signature handler*);

SubFilter: Refere-se à codificação da assinatura, armazenada na entrada *Contents*;

Contents: Contém a estrutura da assinatura, no formato PKCS #1 ou PKCS #7 codificada em Hexadecimal;

Certs: Deve-se incluir a cadeia de certificação do signatário, utilizado apenas para assinaturas PKCS #1;

ByteRange: Indica qual o conteúdo assinado;

Reference: Pode conter outros dicionários que auxiliam na detecção de modificações do documento;

M: Indica a data e hora da assinatura;

Location: Indica o local da assinatura;

Reason: Indica a razão da assinatura;

ContactInfo: Informações de contato do signatário como por exemplo o número do telefone;

Name: O nome do signatário. Utilizado somente se não for possível extrair o nome do signatário a partir da assinatura;

Changes: Indica as mudanças realizadas desde a última assinatura;

R: A versão do *Signature Handler* utilizado para criar a assinatura;

O dicionário de assinatura está relacionado a uma segunda estrutura: o Campo de Assinatura. O Campo de Assinatura consiste em outro container, criado no momento da assinatura, que serve para definir a forma gráfica do campo de assinatura, além de associar alguns atributos adicionais ao dicionário de assinatura. O Campo de Assinatura é uma anotação, que pode ser posicionada e exibida em uma página específica. Caso não seja especificada uma anotação pelo signatário (sem assinatura visível), seus limites terão tamanho 0 e este não será exibido em nenhuma página.

Informações referentes à assinatura podem ser incluídas na entrada *seed value* desta estrutura. Esta entrada faz referência para um outro container que provê informações de restrições que devem ser aplicadas no momento da assinatura. Estas restrições incluem: tipo de algoritmo que pode ser utilizado, uso da chave do certificado do signatário, tipo de estrutura da assinatura, etc. Através dessa funcionalidade, pode-se gerar um campo para assinatura e deixá-lo em branco, funcionalidade bastante comum em assinaturas em papel, onde os campos de assinatura são pré-reservados, recebendo as assinaturas posteriormente. Com os atributos os autores podem obrigar o uso de um algoritmo específico, por exemplo.

O dicionário de assinatura, assim como todo o conteúdo, deve ser assinado, com exceção da entrada *Contents*, que receberá posteriormente a assinatura propriamente dita. Ou seja, a assinatura em documentos PDF é aplicada sobre todo o conteúdo, inclusive sobre o dicionário de assinatura. Esta é uma característica bastante peculiar do documento PDF, que implica na seguinte sequência de passos:

1. Criação das estruturas Campo de Assinatura e do Dicionário de Assinatura, alocando o espaço necessário para armazenar o conteúdo da assinatura;
2. Preenchimento dos atributos do Dicionário de Assinatura, com exceção do *Contents*;
3. Cálculo do resumo criptográfico sobre o documento, incluindo o dicionário de assinatura, com exceção do *Contents*;
4. Preenchimento do *Contents* com os valores reais codificados em Hexadecimal.

A figura a seguir 5.1 ilustra o processo de assinatura.

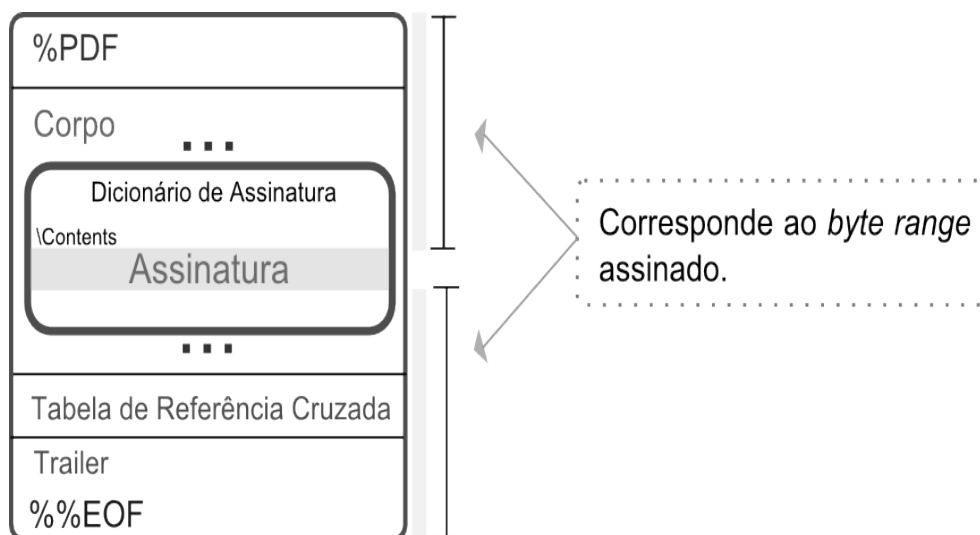


Figura 5.1: Assinatura digital em documentos PDF.

Conforme descrito na seção anterior, 5.3, ao atualizar um documento, tanto a tabela de referência cruzada quanto o *trailer* precisarão também ser atualizados.

Como eles também são assinados com o documento, qualquer alteração anterior deverá necessariamente ser feita através de uma nova revisão. Ou seja, caso uma assinatura for realizada, esta é inserida no corpo do documento. Dessa forma, uma nova assinatura não pode ser adicionada no mesmo corpo porque iria invalidar a anterior, esta deve estar em uma revisão. Uma vez que essa nova revisão esteja também assinada, não será mais possível alterar qualquer propriedade da assinatura anterior. Esta é a característica incremental da assinatura PDF. A estrutura da assinatura armazenada na entrada *Contents* do dicionário de assinatura pode ser tipo *PKCS#1* (Technical 2002) ou *PKCS#7* (Kaliski 1998).

O padrão *PKCS#7* provê uma forma mais completa de representação informações assinadas digitalmente, suportando o uso de referências de validação e suporte a atributos adicionais. Esse padrão deve ser utilizado conforme as regras definidas na *RFC 3852* (Housley 2004) e *RFC 2315* (Kaliski 1998). O Capítulo 3 apresenta maiores detalhes acerca do *Cryptographic Message Syntax*, sucessor do *PKCS #7*, descrito na *RFC 5652*, versão mais atual da *RFC 3852*.

O CMS permite realizar assinaturas com o documento anexado junto à estrutura. A assinatura PDF também provê alternativa equivalente, porém ao invés de anexar o documento, será anexado o resumo criptográfico do *Byte Range* assinado. A escolha do tipo de assinatura é indicada pela entrada *SubFilter* do dicionário de assinatura, que pode ser *adbe.pkcs7.detached* ou *adbe.pkcs7.sha1* o que irá indicar para o *Signature Handler* que a assinatura é com o resumo criptográfico anexado ou não. A escolha do segundo sub-filtro (*adbe.pkcs7.sha1*) permite a realização da assinatura com o resumo do documento anexado na estrutura da assinatura. Porém, este resumo deve ser do tipo *SHA-1*, posteriormente pode-se utilizar outro tipo de algoritmo de resumo sobre este. Ou seja, calcula-se o resumo *SHA-1* do *byte range* que será assinado e o adiciona na estrutura do documento no campo *encapContentInfo*. A partir desse momento pode-se utilizar outro algoritmo de resumo criptográfico sobre este campo para compor a assinatura.

Como a assinatura digital em documentos PDF faz uso da estrutura do

documento, é importante ficar atento a algumas restrições desta estrutura. As informações de validação por exemplo podem ser incluídas no PKCS #7, onde a cadeia de certificação irá no campo *certificates [0]* e as informações de revogação no atributo assinado *adbe-revocationInfoArchival*. Importante citar que este atributo armazena somente os valores de revogação e não referências, assim, caso seja necessário realizar uma assinatura que contenha dados de validação, estes devem ser adquiridos antes da realização desta assinatura. Outro detalhe é a restrição de que a estrutura da assinatura deva conter a assinatura de apenas um signatário, ou seja, não é permitida a realização de coassinaturas. Uma alternativa seria através do uso das revisões.

Como discutido na seção que trata da atualização de documentos 5.3, ao alterar o tamanho da estrutura, será alterado o *byte range* do conteúdo assinado, o mesmo ocorre com a inclusão de atributos na estrutura do *PKCS#7*, caso estes atributos ultrapassem o tamanho pre-estabelecido no momento da assinatura, a inclusão deles a invalidará. Assim, a mesma atenção deve ser dada ao utilizar um carimbo de tempo sobre a assinatura. A assinatura PDF permite o uso do carimbo de tempo que é armazenado na estrutura *PKCS#7*, como um atributo não assinado. Porém é aconselhável que o mesmo seja realizado imediatamente após a realização da assinatura, não apenas para que a data do carimbo seja próxima à data da assinatura, mas também para evitar que a assinatura seja bloqueada por uma segunda, que por ventura seja incluída momentos após a primeira. Como a assinatura subsequente também irá assinar a primeira, por meio das revisões, caso esta não contenha o carimbo de tempo, ficará inviável adicioná-lo posteriormente.

5.4.1 Assinatura de Certificação

Um outro tipo de assinatura que pode ser aplicada é a assinatura de certificação. Essa assinatura trabalha em conjunto com a funcionalidade de permissão de detecção de modificações *MDP*, permitindo que um documento possa ser modificado sem alterar a validade das assinaturas existentes. Somente uma assinatura de certificação pode ser aplicada sobre o documento. As informações de como é realizada a

detecção das modificações ocorrem através do preenchimento da entrada *Reference* do dicionário de assinatura. Esta irá conter as entradas *TransformMethod* e *TransformParams*, definindo as regras de detecção de modificação sobre o documento. A definição dessas estruturas permitem determinar quais objetos podem ser incluídos, excluídos ou alterados após a certificação.

O método de transformação que deve ser utilizado com documentos assinados digitalmente é o *DocMDP (Modification Detection and Prevention)*, que permite à pessoa que aplicou a primeira assinatura especificar quais mudanças são permitidas no documento e quais invalidam a assinatura do autor. A restrição pode ser indicada pelo valor *1* que especifica que nenhuma alteração é permitida, pelo valor *2*, que permite o preenchimento de formulários, criação de templates de páginas e a adição de assinaturas digitais ou ainda pelo valor *3* que inclui as ações do *2* e possibilita criar anotações, deletar e modificar.

O padrão PDF permite a inclusão de um conjunto de ações que possibilitam um comportamento inesperado do conteúdo do documento, como por exemplo a utilização de conteúdo dinâmico, em linguagem *JavaScript*. Características como esta devem ser levadas em consideração ao se aplicar uma assinatura de certificação sobre um documento assinado. Assim, é importante que a assinatura de certificação contenha um dicionário de comprovação legal *legal attestation dictionary*, que permite salientar todos os conteúdos que podem resultar em um comportamento inesperado.

5.4.2 Múltiplas Assinaturas

Como discutido na seção sobre atualização de um documento PDF 5.3, ao incluir algum objeto no corpo do documento, toda sua estrutura é alterada. Para que isso não ocorra deve-se fazer uso do conceito da atualização incremental através da criação de revisões. O conteúdo original irá ficar intacto, permitindo que uma nova assinatura seja adicionada sem invalidar a outra. Como o *Byte Range* da assinatura anterior é preservado, é possível recriar o estado do documento correspondente àquela assinatura.

A inclusão de uma nova assinatura ocorre da forma tradicional, porém esta não irá assinar apenas a nova revisão, mas o documento inteiro, inclusive a assinatura anterior. O processo de inclusão de novas assinaturas, assim como a criação de revisões, podem ser repetidas várias vezes, conforme ilustrado pela figura 5.2.

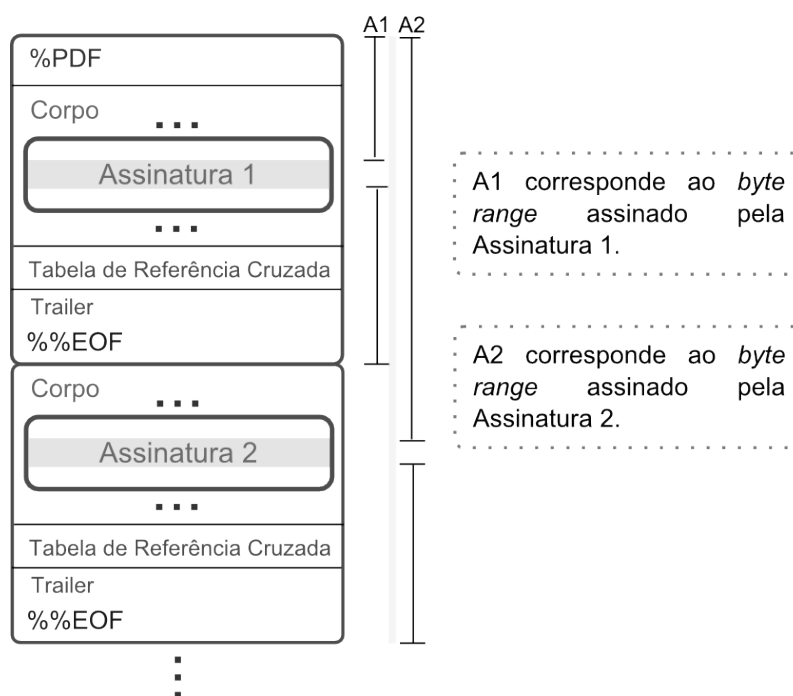


Figura 5.2: Múltiplas Assinaturas.

Dessa forma, percebe-se que uma assinatura incremental detém características diferentes da co e contra-assinaturas tradicionais. Na verdade esta detém características dos dois conceitos anteriores, bem como características peculiares. Tal como na co-assinatura, a assinatura incremental está atrelada diretamente ao conteúdo do documento. Entretanto, de forma equivalente à contra-assinatura, a assinatura incremental está também atrelada à assinatura anterior. Diferentemente dos dois tipos de assinaturas, a alteração, ainda que mínima, na assinatura anterior ou em partes do documento assinado implica na invalidação de todas as subsequentes.

5.5 Considerações Finais do Capítulo

Neste capítulo foi apresentado o formato de documento portátil PDF, com foco nas características relacionadas à assinatura digital. Pôde-se perceber que uma assinatura PDF trabalha em conjunto com a estrutura do documento, permitindo adicionar restrições e exibir informações referentes a esta assinatura. Um dos pontos mais importantes, a natureza incremental da assinatura PDF, torna a manutenção da assinatura por longo prazo uma limitação para o padrão. No Capítulo 6, será apresentado o padrão PAdES, que mantém compatibilidade com o padrão PDF, incluindo, entretanto, suporte à preservação por longo prazo de documentos eletrônicos.

6 PDF Advanced Electronic Signatures

6.1 Introdução

PDF Advanced Electronic Signatures (PAdES), do European Telecommunications Standards Institute (ETSI), é uma especificação técnica que estende e aplica restrições as assinaturas digitais PDF normatizadas pela ISO 32000-1 (Standard 2008) para obterem status de assinaturas avançadas. Além disso, provê suporte para assinaturas XML e inclui características que possibilitam gerar assinaturas equivalentes as descritas nas especificações CAdES (INSTITUTE 2010) e XAdES (INSTITUTE 2010).

Esta especificação é dividida em seis partes a qual inicia pelo *PAdES Overview* (Institute 2009) considerado um documento de carácter informativo, que descreve sobre o suporte para assinatura XML e apresenta as características dos documentos(*PAdES Basic* (Institute 2009), *PAdES Enhanced - PAdES-BES and PAdES-EPES Profiles* (Institute 2010), *PAdES-LTV* (Institute 2009) e *PAdES for XML Content* (Institute 2009)) descrevendo como os perfis podem ser combinados.

Além disso, o *PAdES Overview* apresenta como a assinatura é realizada conforme a norma ISO 32000-1 e quais os tipos de assinaturas do documento PDF (vide seção 5.4) e informa a existência do *PDF/A-1* que pode ser utilizado para manter a representação fixa do documento por longo prazo. Esta parte não descreve, apenas, sobre a representação visual da assinatura digital apresentada na parte seis intitulada *Visual Representations of Electronic Signatures* (Institute 2010), esta descreve

requisitos e recomendações para a representação de assinaturas digitais avançadas em documentos PDF.

Neste capítulo serão apresentados os principais conceitos relacionados a assinaturas CADES em documentos PDF. Sabe-se que a total compreensão da especificação PAdES é vital para a adoção deste padrão de assinatura junto ao contexto nacional, e características dos elementos XML assim como da aparência das assinaturas devem ser analisadas. Porém como o objetivo deste trabalho esta relacionada a adoção do PAdES utilizando CADES, este capítulo irá descrever com mais detalhes as especificações *PAdES Basic*, *PAdES Enhanced - PAdES-BES and PAdES-EPES Profiles* e *PAdES-LTV* pois fazem referência direta a este tipo de assinatura.

6.2 PAdES Basic

A especificação técnica ETSI TS 102 7782 (Institute 2009) utiliza a assinatura baseada em CMS conforme especificada na ISO 320001 (Standard 2008). E tem por objetivo maximizar a interoperabilidade deste tipo de assinatura, fornecendo restrições adicionais àquelas descritas na ISO. Estas assinaturas suportam: assinaturas incrementais, opcionalmente incluem carimbo de tempo, informações revogação e um dicionário de comprovação legal (vide seção 5.4.1) e podem fazer uso dos campos (Razão, Local e Contato) do dicionário de assinatura.

Conforme exposto na ISO, a assinatura deve cobrir todo do documento, exceto a entrada *Contents* do dicionário e, a estrutura da assinatura deve ser do tipo PKCS #7 o qual irá incluir um único *SignerInfo* (vide seção CMS3) e no mínimo, o certificado do signatário. A entrada do dicionário de assinatura *SubFilter* deve fazer referência apenas aos valores (*adbe.pkcs7.detached* e *adbe.pkcs7.sha1*) e a especificação recomenda o uso de algoritmos mais fortes ao do tipo *SHA1*.

A adição do carimbo do tempo e das informações de revogação não sofreram nenhuma restrição, estas são incluídas conforme especificado na ISO 32000-1. O carimbo deve ser adicionado imediatamente para indicar o tempo mais próximo da

assinatura e as informações de revogação são adicionadas nos atributos assinados. Já o uso de certificados de atributo não é recomendado e a utilização dos *seed values*, exposto na seção 5.4, não podem ir contra os requisitos da especificação, ou seja, utilizar um *seed values* que especifique que a assinatura deve ser do tipo PKCS #1 não é permitido.

A utilização de *Signature Handlers* é encorajada pela especificação, estes devem estar aptos a verificar a assinatura. O leitor de arquivos PDF deve utilizar o *Signature Handler* adequado o qual irá verificar conforme procedimentos tradicionais: comparar resumos criptográficos, validar a cadeia de certificação, etc.

6.3 PAdES Enhanced - PAdES-BES and PAdES-EPES

A especificação ETSI TS 102 7783 (Institute 2010) apresenta formas de assinatura alternativas as especificadas na ISO 32000-1. Estas são mapeadas para os formatos PAdES-BES e PAdES-EPES que podem opcionalmente conter um carimbo do tempo, sendo formatos equivalentes ao CAdES-BES 3.3.1.1, CAdES-EPES 3.3.1.2 e CAdES-T 3.3.1.4. Assim, semelhante a especificação CAdES (INSTITUTE 2010), o formato PAdES-EPES possui política explícita e é construída sobre o PAdES-BES, porém apesar permitir o uso do carimbo do tempo não possui a nomenclatura "T".

A estrutura da assinatura deve estar em conformidade com a RFC 3852 (Housley 2004) e ser adicionada ao documento conforme especificado na ISO 32000-1 (Standard 2008)(seção 12.8.1), o *byte range* assinado é o documento inteiro excluindo a entrada *Contents* que irá conter a estrutura CAdES. Como esta forma de assinatura tem como base a tradicional, que utiliza o dicionário de assinatura. Atributos deste dicionário tem preferência a alguns utilizados no CAdES para evitar redundância. Acresce que, nem todos os atributos do CAdES podem ser utilizadas por este perfil, recomenda-se que atributos não assinados não descritos na especificação sejam ignorados, ao menos que outros perfis os requeiram.

Os atributos assinados obrigatórios dos perfis PAdES-BES e PAdES-EPES

são:

- *content-type*. Indica o tipo do conteúdo assinado e deve ter apenas o valor "*id-data*";
- *message-digest*. Resumo criptográfico do *byte range* assinado;
- *Signing Certificate Reference*. Armazena informações sobre o certificado do signatário, afim de evitar ataques de substituição e reemissão. Seu uso é igual o especificado pela especificação CADES.

A seguir será listado um conjunto dos atributos CADES que a especificação trata como opcionais ao compor os formatos PAdES-BES ou PAdES-EPES:

- *signature-policy-identifier*. Política de assinatura que deve ser um atributo assinado e estar em conformidade com o CADES. Estas não devem ser confundidas com os *seed values* (vide seção 5.4). Os *seed values* representam restrições ao fluxo de criação da assinatura, ou seja, impõem restrições ao software assinador, enquanto as políticas definem regras acordadas entre o signatário e o verificador. Apesar disso, especificação define elementos que possibilitam restringir a utilização da política de assinatura, através da adição dos valores (*SignaturePolicyOID*, *SignaturePolicyHashValue*, *SignaturePolicyHashAlgorithm* e *SignaturePolicyCommitmentType*) no *seed values*.
- *signature-time-stamp*. Carimbo do tempo conforme descrito norma ISO 32000-1.
- *signing-time*. A data da assinatura deve ser incluída na entrada "*M*" do dicionário de assinatura;
- *counter-signature*. Refere-se a contra-assinatura e este atributos não pode ser utilizado;
- *content-reference e content-identifier*. Permite referenciar dados externos a assinatura. Não pode ser utilizado;

- *content-hints*. Descreve o conteúdo assinado. Não deve ser utilizado;
- *commitment-type-indication*. Define o compromisso do assinante e pode ser utilizada para o perfil PAdES-EPES mas não para o PAdES-BES.
- *signer-location*. A localização deve ser incluída na entrada "*Location*" do dicionário de assinatura;
- *signer-attributes*. Deve ser utilizado conforme especificado no CAdES, porém, certificados de atributo não podem ser utilizados.
- *content-time-stamp*. Carimbo do tempo sobre o dado. Deve ser utilizado conforme definido no CAdES.

Como a ISO 32000-1 permite que extensões e esta especificação fez uso deste recurso. Deve-se adicionar no dicionário de extensões *Extensions Dictionary* o valor:

```
<</ESIC<</BaseVersion/1.7
    /ExtensionLevel 2>>
>>
```

Apesar da assinatura possuir como suporte a estrutura tradicional da assinatura PDF a qual exige que a estrutura contenha apenas um *SignerInfo*, esta não detém de todas as características especificadas na ISO 32000-1. Assinaturas, PKCS#1 ou com o sub-filtro (*adbe.pkcs7.detached* e *adbe.pkcs7.sha1*) não são permitidas, somente o sub-filtro *ETSI.CAdES.detached*. O modo de validação da assinatura ocorre da forma tradicional a verificação PDF com adição da verificação dos campos CAdES.

6.4 PAdES Long Term - PAdES-LTV Profile

Uma assinatura digital exige que informações de validação, como a cadeia de certificação e dados de revogação, estejam disponíveis no momento da verificação. Os tipos de assinatura supracitados provêm auxílio ao armazenamento destes dados,

porém o faz através dos campos *certificates [0]* e *adbe-revocationInfoArchival* da estrutura da assinatura. Como as informações de revogação desta estrutura é assinada e dada as características das assinaturas incrementais (vide seção 5.4) tornar as assinaturas anteriores imutáveis, caso os dados de revogação não estejam disponíveis antes da realização da assinatura esta não poderá ser realizada ou não irá conter os dados de validação. Uma alternativa à solução deste problema é através da especificação PAdES-LTV que permite que os dados de validação possam ser adicionados posteriormente a realização da assinatura sem invalidar as assinaturas incrementais caso estas tenham sido realizadas.

Isto é feito através da especificação ETSI TS 102 7784 (Institute 2009) que adiciona suporte a validação a longo prazo através do PAdES-LTV às assinaturas baseadas em CMS conforme especificado na ISO 320001 (Standard 2008). Ela, define como as informações de validação devem ser adicionadas ao documento PDF e posteriormente como protege-lo através da inclusão de carimbos do tempo. Além disso, esta proteção a longo prazo pode ser adicionada aos perfis PAdES Basic (Institute 2009), PAdES-BES e PAdES-EPES (Institute 2010). O que permite obter funcionalidades equivalentes ao CAdES-X Long e CAdES-A. Acresce que, o PAdES-LTV pode também ser utilizado conjuntamente com a especificação ETSI TS 102 778-5 *PAdES for XAdES signatures* (Institute 2009) o qual permite construir formas equivalentes ao XAdES-XL e XAdES-A.

Para prover tais recursos, este perfil faz uso de uma extensão ao ISO 32000-1 chamado *Document Security Store* (DSS), localizado no corpo da estrutura do PDF, que irá conter os dados necessários para a validação das assinaturas. Assim, o *Document Security Store* é um dicionário utilizado para armazenar os valores dos dados de validação. A estrutura irá incluir também os dados de validação do carimbo do tempo realizado sobre a assinatura. Além disso, o DSS não refere-se a apenas uma assinatura, caso o documento possua assinaturas incrementais, as informações de validação de todas as assinaturas serão armazenada nesta estrutura. Ou seja, o DSS tem o objetivo de fornecer um repositório comum o que evita armazenar dados duplicados, porém

não tem suporte para referenciar dados externos. Sua utilização deve ser através da inclusão de uma nova entrada no dicionário de catálogo do documento chamada *DSS* que irá referenciar as informações relacionadas as assinaturas. A tabela 6.1 ilustra a estrutura do DSS.

Type	(Opcional). Deve conter o valor DSS.
VRI	(Opcional). Contém referência para os dicionários que indicam as informações relacionadas a cada assinatura.
Certs	(Opcional). Contém referência aos certificados utilizados para a validação
OCSPs	(Opcional). Contém referência as informações de revogação (<i>Online Certificate Status Protocol</i>).
CRLs	(Opcional). Contém referência as informações de revogação (Lista de certificados revogados).

Tabela 6.1: Dicionário DSS.

O *Document Security Store* pode fazer uso de uma estrutura auxiliar chamada *validation-related information* (VRI), representada pelo *Signature VRI Dictionary*, esta estrutura contém o resumo criptográfico *SHA1* da assinatura a qual referencia. Isto possibilita acessar os dados de validação de uma assinatura específica diretamente, o que melhora a performance da verificação. Assim, caso um documento PDF contenha a estrutura VRI vinculada a um DSS, o processo de verificação deve dar preferência para esta estrutura. Caso o documento contenha apenas o DSS e este não possua informações sobre a assinatura, o verificador deve verificar se a estrutura da assinatura contém os dados de validação. Caso estes dados não sejam encontrados internos ao documento PDF, deve-se utilizar o repositório local (repositório do Windows por exemplo) ou on-line. A figura 6.1 ilustra o *Document Security Store* e o *Signature VRI Dictionary*.

Para possibilitar a proteção a longo prazo do documento e das informações de validação, esta especificação apresenta uma outra extensão a ISO 32000-1 chamada *Document Time-stamp*. Este é um dicionário de assinatura padrão, semelhante ao utilizado para realizar assinaturas (vide seção 5), porém com algumas alterações. A entrada

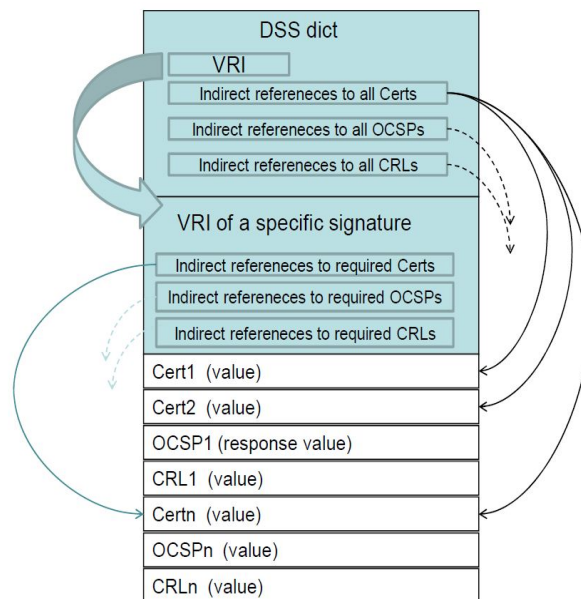


Figura 6.1: Dicionário DSS e *Signature VRI Dictionary* (Imagem extraída do ETSI TS 102 7784 (Institute 2009)).

Contents conterá um carimbo do tempo codificado em hexadecimal que deve estar em conformidade com a RFC 3161 (Adams et al. 2008), este será realizado sobre todo o documento, igual a uma assinatura incremental (vide seção 5.4). A entrada *SubFilter* deve conter o valor *ETSI.RFC3161* para indicar o formato dos dados contido na entrada *Contents*, outro detalhe é sobre o valor da entrada "V" do dicionário de assinatura, que se existir deve conter o valor 0 (zero). Os seguintes campos foram removidos: *Cert*, *Reference*, *Changes*, *R*, *PropAuthTime*, e *PropAuthType*. Caso o carimbo do tempo contenha informações equivalentes ao *Name*, *M*, *Location*, *Reason*, e *ContactInfo*, estes não devem estar no dicionário e podem ser ignorados pelo verificador.

É recomendado que a utilização do DSS seja seguida de um *Document Time-stamp*, caso seja preciso prolongar a proteção do documento deve-se adicionar sucessivos DSS seguidos de um *Document Time-stamp*. A validação é realizada considerando a data do último carimbo para validar os mais interno. A figura 6.2 ilustra a utilização de sucessivos carimbos.

Como esta especificação introduz extensões a ISO 32000-1, o dicionário

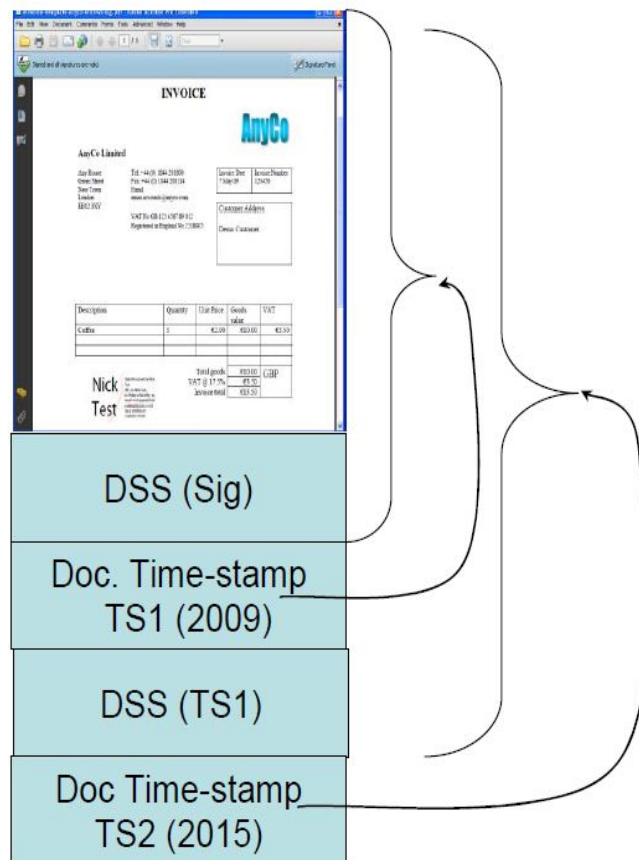


Figura 6.2: Aplicação de Sucessivos Carimbos (Imagem extraída do ETSI TS 102 7784 (Institute 2009)).

de catalogo do documento deve conter o valor:

```
<</ESIC
<</BaseVersion /1.7
/ExtensionLevel 1
>>
>>
```

A identificação do DSS também ocorre através do valor:

```
<</ADBE
<</BaseVersion /1.7
```

```
/ExtensionLevel 5
```

```
>>
```

6.5 Considerações Finais do Capítulo

Este capítulo apresentou o formato avançado de assinatura sobre documentos PDF, com foco no padrão CAAdES. Após o entendimento deste padrão pode-se então partir para a análise da viabilidade da utilização do PAdES no âmbito da ICP-Brasil, o que será descrito no capítulo 7.

7 Utilização do Padrão PAdES no Âmbito da ICP-Brasil

7.1 Introdução

Com a revisão dos principais padrões de assinatura utilizados tanto em âmbito mundial como no cenário nacional, com o PBAD, torna-se possível avaliar os requisitos necessários para a utilização do padrão PAdES no contexto do padrão proposto pela Infraestrutura de Chaves Públicas Brasileira. Este capítulo apresenta uma análise das alterações necessárias, limitações e impactos da adoção do padrão PAdES no contexto do Padrão Brasileiro de Assinatura Digital. Para tanto, serão apresentadas propostas de representação das assinaturas digitais sob cada política de assinatura definida na ICP-Brasil, utilizando os perfis *PAdES Enhanced - PAdES-BES and PAdES-EPES Profiles* (Institute 2010) e *PAdES-LTV* (Institute 2009). Por fim, serão ainda apresentadas algumas considerações sobre as possíveis mudanças na regulamentação da ICP-Brasil, com vista a suportar as tecnologias e peculiaridades inerentes ao uso do padrão PAdES.

7.2 Perfil de uso Geral

Conforme exposto no capítulo 6, o padrão PAdES permite a utilização do padrão CAAdES para a composição da assinatura. Assim, buscou-se avaliar a possibilidade em utilizar os atributos e perfis presentes neste formato para compor a assinatura

digital a ser utilizada no PBAD. Entretanto, analisando-se o padrão PAdES de forma mais detalhada, pode-se perceber que a assinatura neste tipo de documento possui algumas peculiaridades, diferentes de todos os outros padrões estudados. De acordo com o padrão PAdES, a assinatura não é só a estrutura CAdES em si, mas sim a união dela com algumas outras estruturas componentes do documento PDF.

Para avaliar a possibilidade de utilização do padrão PAdES segundo os critérios da ICP-Brasil, foram então analisadas as políticas de assinaturas digitais previstas no PBAD e levantada a lista de atributos aprovados, conforme as definições do DOC-ICP-15.02 (ITI). Com a lista dos atributos aprovados pela ICP-Brasil, buscou-se o mapeamento dos mesmos em atributos equivalentes, definidos no PAdES. Após construída uma tabela de equivalência, buscou-se mapear os diferentes perfis de assinatura, de acordo com cada política.

A tabela 7.1 apresenta os atributos equivalentes aos atributos assinados, conforme definido na tabela 2.1 do DOC-ICP-15.02 (ITI). É importante salientar que a assinatura, segundo o padrão PDF, é realizada sobre todo o documento, conforme apresentado no capítulo 5. Dessa forma, mesmo utilizando campos externos à estrutura CAdES para garantir a integridade e autenticidade do documento, estes encontram-se também assinados. Em outras palavras, mesmo que a referida tabela aponte para atributos externos à assinatura, presentes no conteúdo do documento, tratam-se de atributos assinados.

Atributo	Ref (Institute 2010)	Observações
id-aa-ets-contentTimestamp	4.5.11	Utiliza-se o atributo do CAdES (INSTITUTE 2010).
id-aa-ets-signerAttr	4.5.10	Utiliza-se o atributo do CAdES. Obs.: Certificado de atributo não pode ser utilizado conforme descrito em ISO 32000-1 Seção 12.8.3.3.1.

Atributo	Ref (Institute 2010)	Observações
id-aa-ets-signerLocation	4.5.9	Deve ser utilizado o campo "Location" do dicionário de assinatura especificado em ISO 32000-1 (Standard 2008) Seção 12.8.1.
id-signingTime	4.5.3	Deve ser utilizado o campo "M" do dicionário de assinatura especificado em ISO 32000-1 Seção 12.8.1.
id-contentType	4.4.1	Utiliza-se o atributo do CADES. Obs.: Somente o valor <i>id-data</i> pode ser usado.
id-messageDigest	4.4.2	Utiliza-se o atributo do CADES.
id-aa-signingCertificate(V2)	4.4.3	Utiliza-se o atributo do CADES.
id-aa-ets-sigPolicyId	4.5.1	Utiliza-se o atributo do CADES.

Tabela 7.1: Atributos assinados e suas equivalências no padrão PAdES.

Além dos atributos assinados definidos pela ICP-Brasil, para estar em conformidade com a especificação PAdES seriam ainda necessários definir os seguintes atributos:

Atributo	Ref (Institute 2010)	Observações
<i>SubFilter</i>	4.2 (e)	O valor da chave <i>SubFilter</i> do dicionário de assinatura especificado em ISO 32000-1 Seção 12.8.1 deve ser <i>ETSI.CADES.detached</i> .

Atributo	Ref (Institute 2010)	Observações
<i>Extensions Dictionary</i>	4.7 (e)	Extensão incluída no dicionário <i>catalogo do documento</i>

Tabela 7.2: Atributos auxiliares.

7.3 Políticas de Assinatura

7.3.1 AD-RB

A assinatura com referência básica deve fazer obrigatoriamente uso dos seguintes atributos:

<i>Atributos</i>	<i>Representação em PAdES-CAdES</i>
Tipo do Conteúdo	<i>id-contentType</i> .
Resumo criptográfico da mensagem	<i>id-messageDigest</i> .
Informações sobre o signatário	<i>id-aa-signingCertificate(V2)</i> .
Identificador da política	<i>id-aa-ets-sigPolicyId</i> .
<i>SubFilter</i>	<i>ETSI.CAdES.detached</i> .
<i>Extension Dictionary</i>	<</ESIC<</BaseVersion/1.7 /ExtensionLevel 2>>>>

Tabela 7.3: Atributos exigidos pela política de assinatura PBAD-PAdES(AD-RB)

O dicionário de assinatura contém, no mínimo, a estrutura da assinatura no formato CAdES e uma indicação da utilização deste formato. A estrutura CAdES deve estar codificada em hexadecimal e armazenada na entrada *Contents*, já o nome que indica que a estrutura CAdES esta sendo utilizada deve estar na entrada *SubFilter* e possuir o valor *ETSI.CAdES.detached*. A estrutura deve possuir o certificado do signatário armazenado no campo *Certificates [0]* e os atributos assinados: *id-contentType*, *id-messageDigest*, *id-aa-signingCertificateV2* e *id-aa-ets-sigPolicyId*.

O dicionário de catálogo do documento deve indicar que o arquivo PDF contém características definidas na especificação (Institute 2010), através da inclusão do valor

```
<</ESIC<</BaseVersion/1.7/ExtensionLevel 2>>>>
```

na entrada *Extension*. A figura 7.1 mostra uma visão geral da estrutura do documento com estas características.

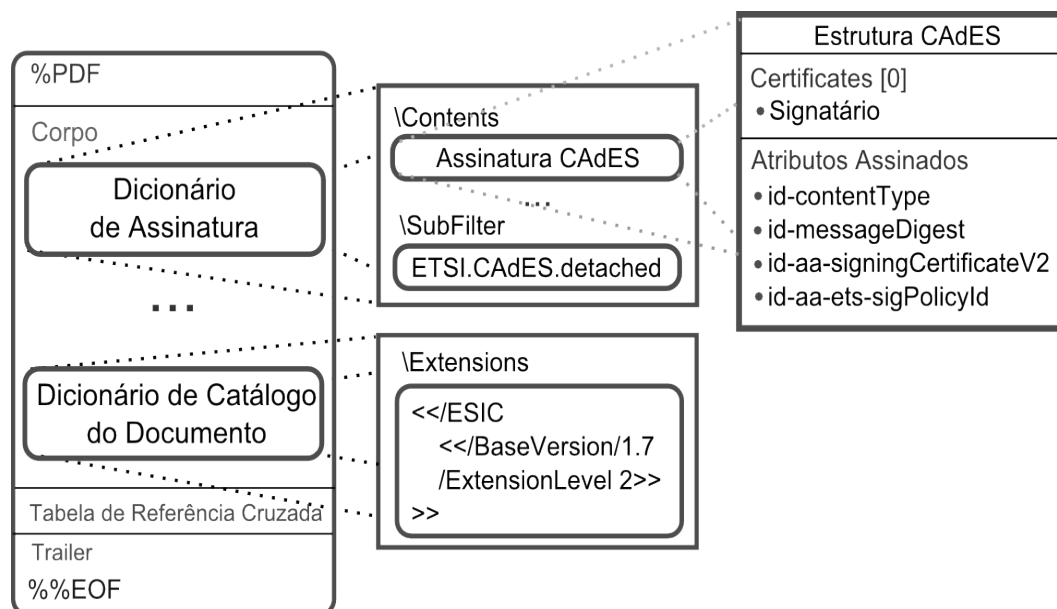


Figura 7.1: Assinatura digital com Referência Básica.

7.3.2 AD-RT

A assinatura digital com referência do tempo é construída sobre a básica. Assim, este perfil detém todas as características supracitadas. Além disso, utiliza o *id-aa-signatureTimeStampToken* como um atributo não assinado da estrutura CADES (vide tabela 7.4), o qual irá conter o carimbo do tempo sobre a assinatura como mostra a figura 7.2.

<i>Atributos</i>	<i>Representação em PAdES-CADES</i>
<i>Atributos PBAD-PAdES(AD-RB)</i>	
Carimbo do tempo sobre a assinatura	<i>id-aa-signatureTimeStampToken.</i>

Tabela 7.4: Atributos exigidos pela política de assinatura PBAD-PAdES(AD-RT)

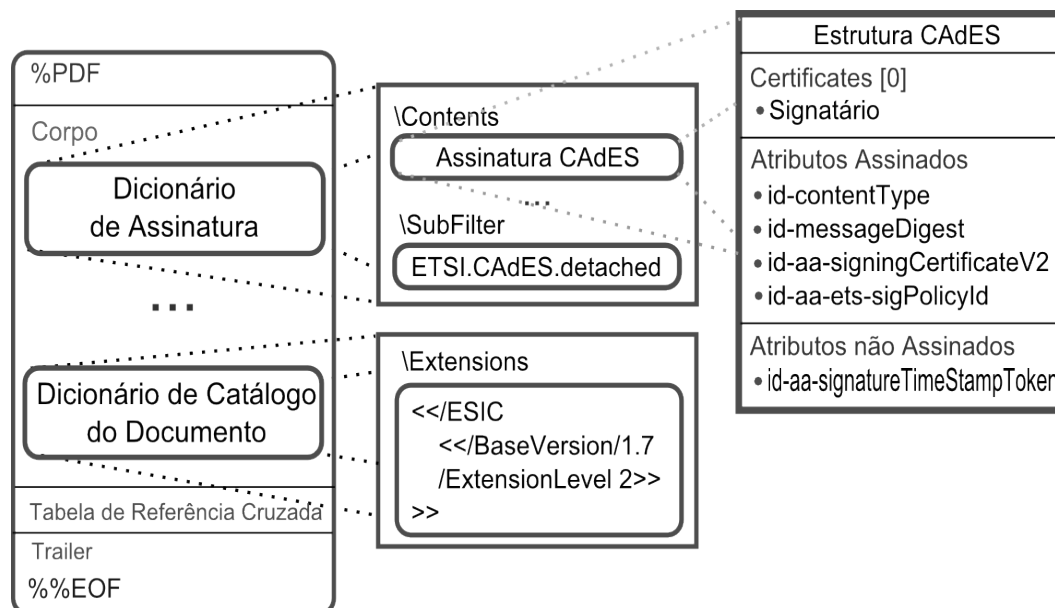


Figura 7.2: Assinatura digital com Referência do Tempo.

7.3.3 AD-RV

Uma assinatura AD-RV é semelhante à assinatura CADES-C, no sentido de que ambas agregam, à assinatura com carimbo de tempo, referências para as estruturas necessárias à validação. Em ambas as assinaturas, tais referências são inseridas através dos atributos não assinados *Complete Certificate References* e *Complete Revocation References*. Para construir uma assinatura digital equivalente em PAdES, seria necessário adicionar as referências supracitadas na estrutura CADES, interna ao documento. Contudo, essa abordagem é desencorajada pela própria especificação, dada a natureza incremental da assinatura PAdES, conforme nota explicativa número 1 do Anexo B, no documento ETSI TS 102 778-4 (Institute 2009). O problema está relacionado à manutenção da assinatura com referências, sobretudo com relação ao *Grace*

Period, conforme exposto na seção 3.3.1.5. É frequente a necessidade de atualização das referências de validação ou mesmo a inserção de tais referências em momento posterior à assinatura, pelo verificador, conforme previsto pela política AD-RV, na seção 5.2.1.2 do DOC-ICP-15.03 (ITI). Como na assinatura PDF a estrutura CADES torna-se imutável após a aplicação de uma segunda assinatura (vide seção 5.4), ficaria inviável a atualização posterior dos atributos não assinados, que de fato passariam a estar assinados no contexto do padrão PAdES. Para melhor ilustrar o problema, a figura 7.3 apresenta um cenário em que a adição posterior das referências de validação é inviável.

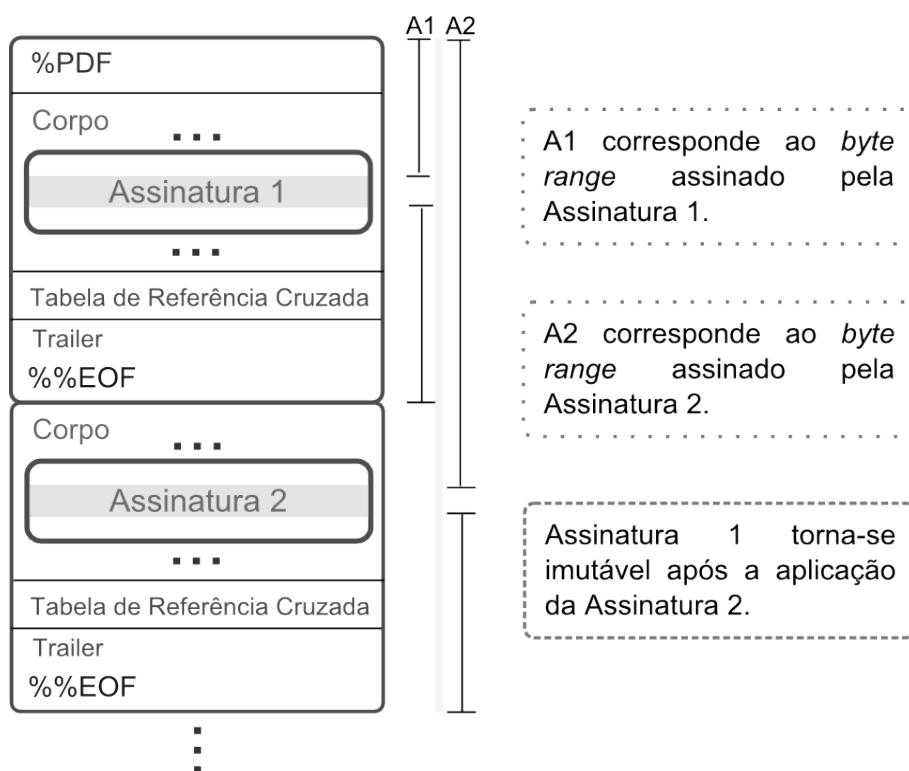


Figura 7.3: Problema Assinatura Incremental.

A especificação do PAdES de fato desencoraja o uso de atributos não assinados, salvo quando expressamente descritos no perfil, tal como exposto no item h, seção 4.2, do documento ETSI TS 102 778-3 (Institute 2010).

Sendo assim, não é possível representar uma assinatura AD-RV segundo o padrão PAdES. Para adicionar suporte a este tipo de assinatura, muito útil em ambi-

entes de alta demanda, com assinaturas em lote e grande fluxo de documentos, seria necessário estender o padrão. O *Document Security Store* ao invés de conter apenas valores, poderia permitir incluir referências de validação, o que possibilitaria que as informações de revogação fossem adicionadas posteriormente ao momento da assinatura, mesmo que o documento contenha assinaturas incrementais. Porém, caso seja realizada uma assinatura incremental após a inclusão das referências estas ficariam imutáveis. Assim, esta opção permite apenas estender o prazo da inclusão das referências.

Uma segunda alternativa seria criar um *DSS Security Timestamp* o qual aplicaria o carimbo do tempo apenas sobre o *Document Security Store* o qual permitiria que este seja destacável, caso fosse realizada uma assinatura incremental, bastaria destacar a revisão que contém as referências de validação, aplicar a assinatura incremental e anexar a revisão destacada novamente. Desse modo, as informações de revogação poderiam ser substituídas sem afetar nenhuma assinatura do documento. Porém, o estudo do PAdES no Âmbito da ICP-Brasil restringe-se a utilização dele no contexto nacional e qualquer forma de estender a especificação está fora do escopo deste trabalho.

7.3.4 AD-RC

A assinatura equivalente à política AD-RC pode ser obtida da utilização da estrutura *Long Term Validation - LTV*, para armazenar os valores dos dados de revogação. A seção subsequente apresenta esta abordagem.

7.3.4.1 AD-RC utilizando PAdES-LTV

O formato de assinatura digital com referências completas pode ser obtido através do uso do próprio suporte a validação por longo prazo definido no padrão (PAdES-LTV). Segundo essa alternativa, o *Document Security Store (DSS)* é utilizado para armazenar os valores dos dados de validação. A estrutura irá incluir também os dados de validação do carimbo do tempo realizado sobre a assinatura.

Posteriormente, deve-se aplicar um carimbo de tempo sobre o documento *Document Timestamp* que, como na abordagem anterior, irá também amarrar os dados de validação conforme mostra a tabela 7.5.

<i>Atributos</i>	<i>Representação em PAdES-CADES</i>
<i>Atributos PBAD-PAdES(AD-RT)</i>	
Valores de validação	<i>Document Security Store.</i>
Carimbo do tempo sobre todo o documento	<i>Document Timestamp.</i>

Tabela 7.5: Atributos exigidos pela política de assinatura PBAD-PAdES(AD-RC) LTV

Caso assinaturas incrementais sejam aplicadas ao documento antes da inclusão do DSS, estas permanecerão íntegras. Caso adicionadas após, as informações de validação permanecerão imutáveis, dado as características das assinaturas incrementais conforme exposto na seção 5.4.

Então, a utilização do DSS permite prolongar o prazo para a obtenção dos dados de revogação mais adequados à assinatura. E estes serão amarrados através da aplicação de um carimbo do tempo sobre todo o documento conforme exposto na figura 7.4.

O dicionário de catálogo do documento deve conter os valores:

```
<</ESIC<</BaseVersion/1.7
    /ExtensionLevel 1>>
/ESIC<</BaseVersion/1.7
    /ExtensionLevel 2>>
>>
```

7.3.5 AD-RA

A assinatura digital com referências para arquivamento deve ser construída sobre o perfil AD-RC e conter uma restrição à inserção de novas assinaturas ao documento, ou seja, deve ser uma assinatura de certificação. Nas assinaturas PBAD-

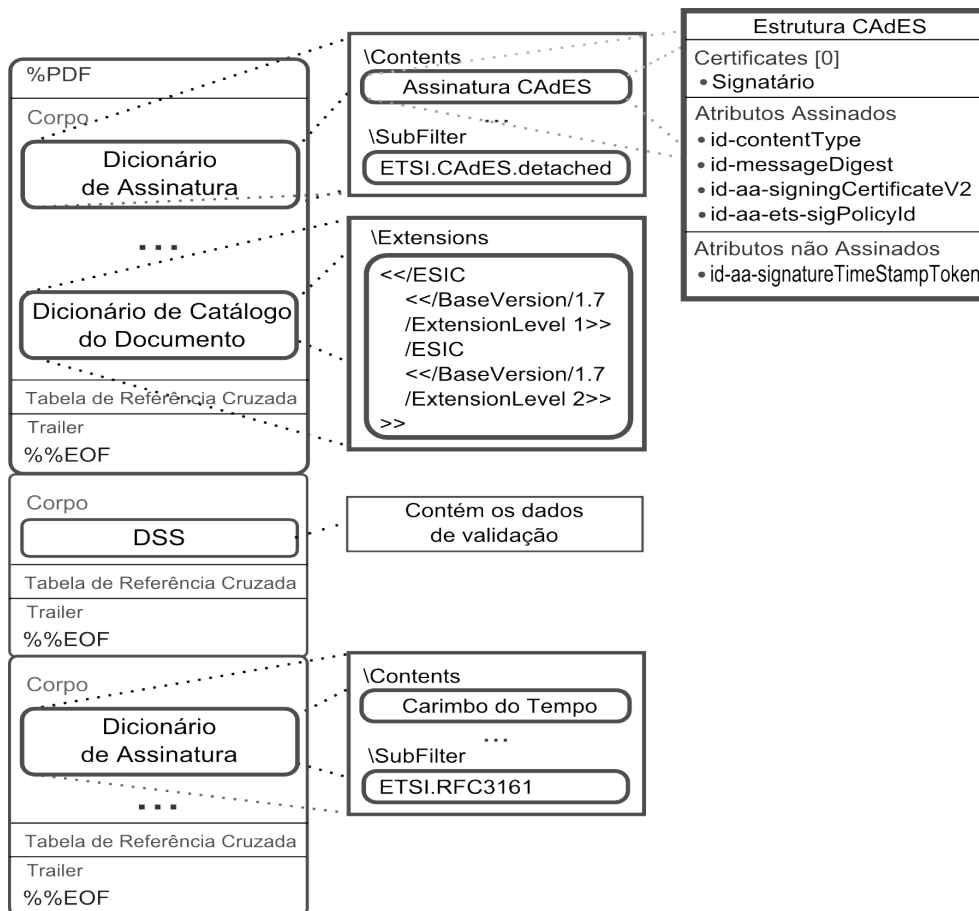


Figura 7.4: AD-RC utilizando PAdES-LTV.

CAAdES, esta restrição é imposta através da aposição de um carimbo do tempo de arquivamento, encapsulando todo o conteúdo assinado. Já no padrão PDF, tal bloqueio é realizado através da utilização do método de transformação DocMDP (*Modification, Detection and Prevention*) onde a chave *P* deve conter o valor que indica que o documento não pode sofrer alterações (vide seção 5.4.1). Apesar desta restrição não permitir que o documento seja alterado, o leitor PDF deve permitir as modificações referentes à inclusão do *Document Security Store* (DSS) e *Document Security Timestamp*.

Assim, uma assinatura digital com referências para arquivamento (vide figura 7.5) detém as mesmas características da assinatura AD-RC utilizando PAdES-LTV, soma-se à ela o método de transformação DocMDP (*Modification, Detection and*

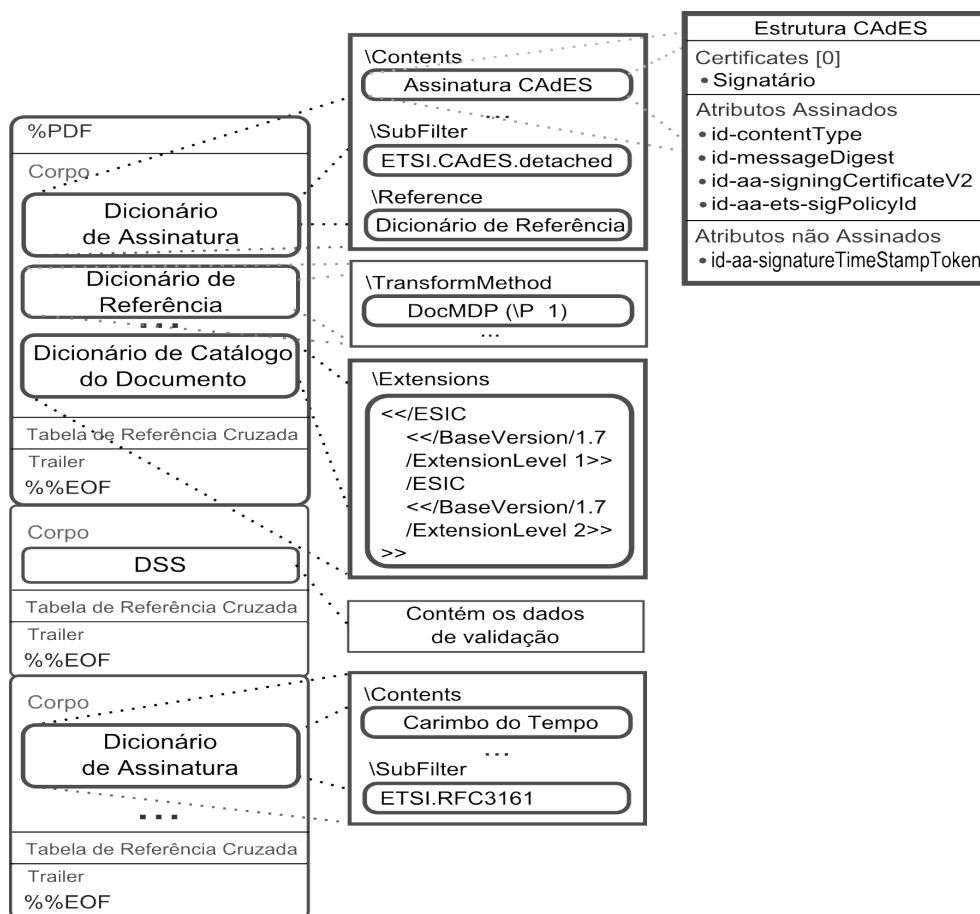


Figura 7.5: Assinatura Digital com Referências para Arquivamento.

Prevention) que deve indicar que o documento não pode sofrer alterações.

7.4 Impactos e Mudanças na Normatização do PAdES

Os tipos de assinaturas utilizados no contexto nacional permitem o uso de coassinaturas, contra-assinaturas e assinaturas com documento anexado. Primeiramente, a especificação PAdES não permite a realização de assinaturas separadas do documento. De fato as assinaturas PAdES encontram-se incorporadas ao conteúdo assinado. Embora esta seja a característica mais marcante do padrão PDF e certamente uma das principais responsáveis por sua aceitação, a forma como foi definida a alocação do campo destinado à assinatura, em área limitada interna ao documento, acaba

por trazer sérias implicações para a manutenção das características de segurança necessárias à manutenção do documento eletrônico por longo prazo. O principal impacto desta abordagem é a necessidade de assinaturas incrementais, operando na forma de revisões, em que uma assinatura é realizada sobre todo o conteúdo do documento mais as assinaturas anteriores. Dessa forma, o PAdES não permite a utilização de múltiplas assinaturas em uma mesma estrutura CMS, não suportando, por consequência, a realização de co e contra assinaturas.

A inserção de um novo contexto relacionado às assinaturas, as incrementais, deverá implicar na revisão dos referidos conceitos, no conjunto normativo da ICP-Brasil. O documento DOC-ICP-15 (ITI) e seus derivados, documentos que definem o uso de assinaturas digitais no âmbito da ICP-Brasil, precisarão ser adaptados ao uso do PAdES. Um exemplo é a seção 6.8 do documento principal, que limita os contextos de assinatura aos três atualmente existentes no âmbito do CADES e XAdES: assinatura, co e contra-assinaturas.

O procedimento de verificação de assinaturas incrementais, quando comparado ao processo de verificação de múltiplas assinaturas em uma estrutura CMS, exige um maior esforço computacional. Enquanto o cálculo do resumo criptográfico do documento assinado pode ser feito uma única vez, nas co assinaturas CADES. Já as assinaturas incrementais exigem o recálculo do resumo criptográfico do conteúdo assinado a cada nova assinatura, onerando o processo de acordo com o tamanho do conteúdo de cada revisão.

Outro ponto a ser considerado é a possibilidade do documento possuir conteúdos dinâmicos, que provoquem um comportamento inesperado, como com a utilização de rotinas em linguagem *JavaScript* para modificar o conteúdo do texto em tempo de exibição. Caso o documento possua tais elementos, deve-se primeiramente realizar uma assinatura de certificação, com vista a evidenciar a existência de quaisquer elementos que representem risco ao verificador.

Um outro problema seria o formato das políticas: ASN.1 para o CADES referencia os atributos com OID e as XML para o XAdES com URI. Seria necessário

um novo padrão de políticas para o PDF para, por exemplo, obrigar a inclusão das extensões e subfilter.

7.5 Considerações Finais do Capítulo

Este capítulo buscou avaliar os impactos, as limitações e os requisitos necessários para a adoção do padrão PAdES, no âmbito da ICP-Brasil. Primeiramente, deixou claro que existe forma equivalente da maioria dos atributos requeridos pelos perfis de assinaturas digitais da Infraestrutura de Chaves Pública Brasileira, exceto os exigidos pelo formato AD-RV. Nesse sentido, foram apresentadas propostas de representação das políticas de assinaturas digitais adotadas pela ICP-Brasil.

Para avaliar a possibilidade de utilização desta proposta foi implementado o Assinador PBAD-PAdES para gerar assinaturas no formato PBAD-PAdES(AD-RB), os resultados desta análise estão descritos no capítulo 8.

8 Protótipo Assinador PBAD-PAdES

8.1 Introdução

Para comprovar a viabilidade da confecção dos perfis propostos, foi criado o *Assinador PBAD-PAdES* utilizando a linguagem de programação *Java*. Este assinador faz uso biblioteca *iText* (Corp.) para manipular a estrutura do PDF e a *Bouncy Castle* (Castle) para efetuar o processo de assinatura digital.

O *iText* é uma biblioteca gratuita que permite a criação e manipulação de documentos PDF sobre os mais variados aspectos. Seu principal objetivo é possibilitar a automatização do processo de gerenciamento do formato de documento portátil. Sobre o ponto de vista de utilização, ela é de fácil entendimento e possui boa documentação. O processo de criptografia foi realizado com o auxílio da biblioteca *Bouncy Castle*. Esta permite o controle total sobre o processo criptográfico através de funções de baixo nível. Estas funções possibilitam ações como: acessar o repositório de chaves, criar e verificar assinaturas digitais, manipular a estrutura da assinatura, dentre outros.

Este capítulo apresentar o *Assinador PBAD-PAdES*. A seção 8.4 analisa a compatibilidade do Acrobat Reader sobre o aspecto de geração de assinatura. Já a seção seguinte, pretende utilizar o tradicional *Adobe Reader* para verificar assinaturas geradas com o *Assinador PBAD-PAdES*. Por fim, a última parte analisa uma assinatura gerada com o *Adobe Acrobat 10* e mostra em linhas gerais sobre a confecção de plugins.

8.2 Assinatura

O Assinador PBAD-PAdES permite realizar assinaturas com referência básica (AD-RB) e explorar a utilização do *Documento Security Store* (DSS) o qual irá permitir criar os outros tipos de assinatura. Além disso, através deste software, pode-se verificar a estrutura da assinatura gerada e criar múltiplas assinaturas. A visão geral da interface do software é ilustrada pela figura 8.1.

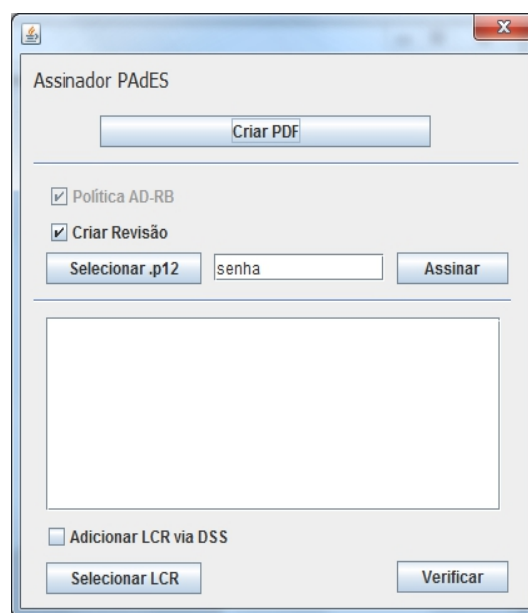


Figura 8.1: Assinador PAdES.

O processo de assinatura ocorre através do uso do repositório de chaves, que deve ser especificado clicando no botão "*Selecionar .p12*". Após este passo, pode-se clicar em "*Assinar*" o que resultará em um documento PDF assinado e um arquivo *.pkcs7* que contém a estrutura da assinatura. O arquivo *.pkcs7* é gerado apenas para permitir a visualização da estrutura de assinatura que foi anexada ao documento. Ou seja, o documento PDF de saída sozinho é o resultado principal do processo de assinatura. A figura 8.2 exibe o comportamento do software após executar este passo.

A estrutura da assinatura gerada é ilustrada pela figura 8.3. Esta contém uma política, o campo *SingningCertificateV2*, e não dispõe do *signingTime* como atri-

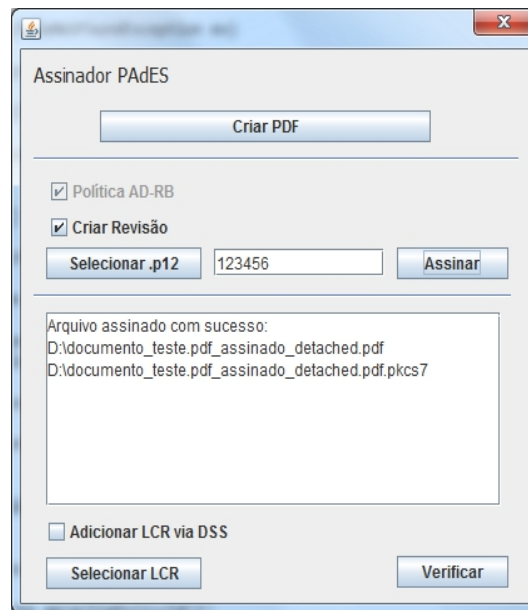


Figura 8.2: Assinatura PAdES AD-RB.

buto assinado anexado a estrutura. Este, conforme a especificação *PAdES* recomenda, foi adicionado no dicionário de assinatura assim como o tipo de codificação, o local e o motivo. Contém as entradas: */SubFilter/ETSI.CAdES.detached /Location(Local Assinatura) /M(D:20120515031156-03'00')* */Reason(Razão Assinatura)*. A extensão *ESIC ExtensionLevel 2* foi incluída no dicionário *catalogo do documento*.

8.3 Verificação

A processo de verificação ocorre através do botão "*Verificar*" que valida a integridade do documento, a política de assinatura utilizada, o atributo *SingingCertificateV2* e permite extrair detalhes sobre as atualizações incrementais. Um exemplo de verificação é ilustrado pela figura 8.4.

```

[0] Context-Specific, Length = 439
├── SEQUENCE, Length = 24
│   ├── OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 9 3 } contentType
│   └── SET, Length = 11
│       ├── OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 7 1 } data
│       └── SEQUENCE, Length = 47
│           ├── OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 9 4 } messageDigest
│           └── SET, Length = 34
│               ├── SEQUENCE, Length = 145
│               │   ├── OBJECT IDENTIFIER, Length = 11, Value = { 1 2 840 113549 1 9 16 2 15 } id-aa-ets-sigPolicyId
│               │   └── SET, Length = 129
│               │       ├── SEQUENCE, Length = 127
│               │       │   ├── OBJECT IDENTIFIER, Length = 7, Value = { 2 16 76 1 7 1 1 } id-icpbr-adrb
│               │       │   ├── SEQUENCE, Length = 47
│               │       │   └── SEQUENCE, Length = 67
│               │       │       ├── SEQUENCE, Length = 65
│               │       │       │   ├── OBJECT IDENTIFIER, Length = 11, Value = { 1 2 840 113549 1 9 16 5 1 } id-spq-ets-uri
│               │       │       │   └── IAString, Length = 50, Value = "http://endereco.politica.com.br/politica_pades.der"
│               │       └── SEQUENCE, Length = 213
│               │           ├── OBJECT IDENTIFIER, Length = 11, Value = { 1 2 840 113549 1 9 16 2 47 } id-aa-signingCertificateV2
│               │           └── SET, Length = 197

```

Figura 8.3: Atributos Assinados da Estrutura da Assinatura.

8.4 Compatibilidade com Adobe Acrobat

O Adobe Acrobat (Systems) constitui uma família de aplicativos produzida pela Adobe Systems. Dois *softwares* muito utilizados desta família é o *Acrobat Reader* e *Adobe Acrobat Pro*, ambos atualmente na versão X. O primeiro é totalmente gratuito e contém um conjunto variado de funcionalidades, onde é possível navegar, visualizar e imprimir arquivos PDF. Já o segundo provê um conjunto maior de funcionalidades, porém sua licença não é gratuita e o período para testes é de trinta dias.

8.4.1 Geração de Assinaturas

Ambos os softwares, *Acrobat Reader* e *Adobe Acrobat Pro*, possuem suporte para trabalhar com assinaturas digitais, porém somente com o *Adobe Acrobat Pro* é possível visualizar dentro a aba de ferramentas, sem precisar efetuar nenhuma configuração, a opção para gerar assinatura digital. O Reader pode também assinar

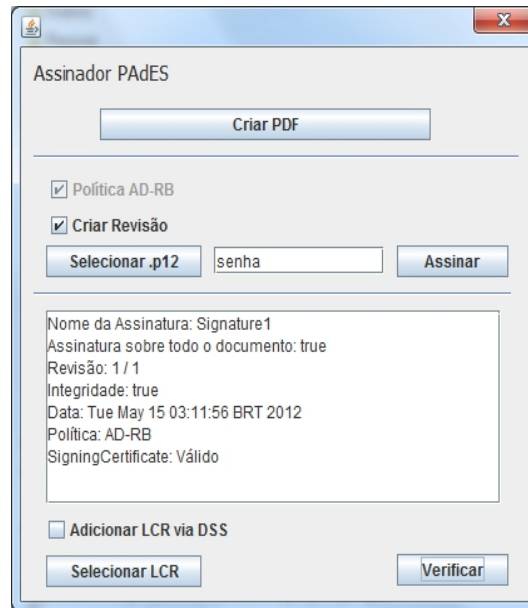


Figura 8.4: Verificação em Software.

desde que seja gerado o campo da assinatura no Acrobat (ou outro software).

O processo para geração de assinatura é muito simples, adiciona-se sobre página um campo que irá exibir os detalhes da assinatura. Estes detalhes constituem a aparência da assinatura e são altamente configuráveis. Uma outra opção disponibilizada pela ferramenta Adobe é a possibilidade de após a realização da assinatura, não permitir que nenhuma modificação seja feita no documento, através do *DocMDP* (ver seção 5). A figura 8.5 exibe a tela principal do módulo que realiza a assinatura.

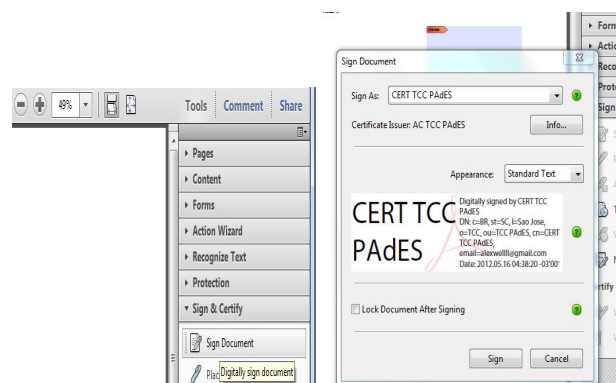


Figura 8.5: Geração de Assinatura.

8.4.1.1 Assinaturas PKCS #7 e Equivalente ao CAdES

O *Adobe Acrobat Pro* além de permitir realizar a assinatura tradicional (vide seção 5.4) também possui suporte para geração de assinaturas *PAdES*, e este é ativado após uma pequena configuração. Basta clicar em: *Editar/Preferências...*, acessar o item "Segurança" clicar em "Preferências Avançadas..." e localizar a aba "Criação". A figura 8.6 exibe a simples alteração que deve ser realizada.

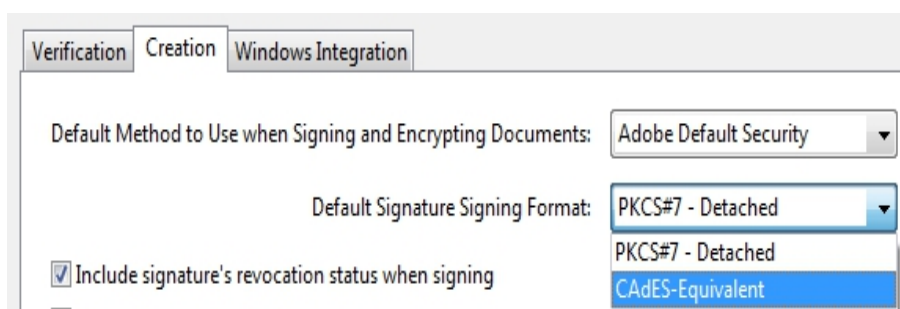


Figura 8.6: Configuração *CAdES Equivalent*.

A assinatura gerada possui como *SubFilter* o valor *ETSI.CAdES.detached* e os dados da entrada *Contents* após convertidos para binário mostraram que a estrutura da assinatura CAdES continha toda a cadeia de certificação no campo *Certificates[0]* e os dados do signatário no atributo assinado *Signing Certificate V2* e as informações de revogação no atributo assinado *adbe-revocationInfoArchival*.

Pode-ser perceber que a estrutura abriga toda a cadeia de certificação, contém também os valores dos dados de revogação e o atributo do conforme mostrado nas figuras 8.7.

O problema ao incluir os dados de validação na estrutura da assinatura, está diretamente relacionado ao *Grace Period*, que diz respeito a obtenção dos melhores dados de revogação, outro ponto importante é que estes dados devem estar disponíveis antes da geração da assinatura. Assim, o Acrobat disponibiliza a opção de incluir os dados de revogação após o momento da assinatura (Systems) e utiliza o *Document Security Store (DSS)* para adicionar os dados de validação. Deve-se realizar uma assinatura sem a inclusão dos dados de validação, e posteriormente clicar com

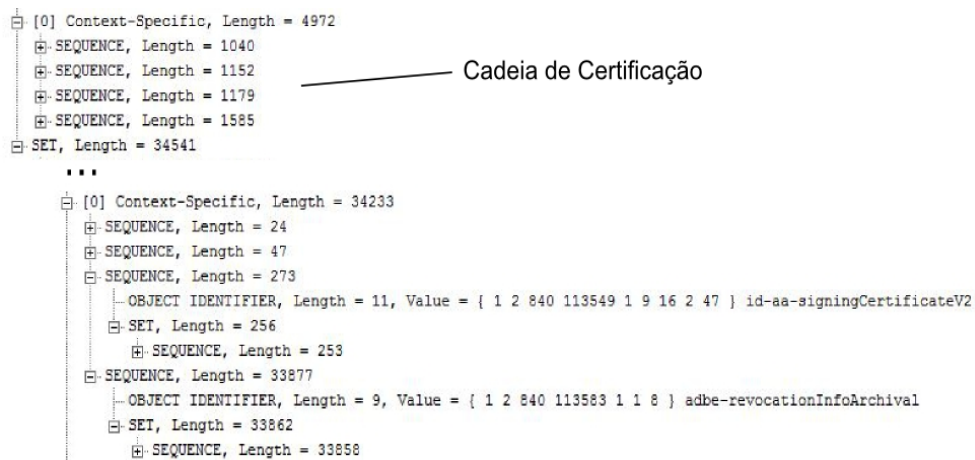


Figura 8.7: Estrutura Assinatura Acrobat Pro.

o botão direito do mouse sobre a assinatura e selecionar "Adicionar informações de verificação". Desse modo, concluí-se que o tipo de assinatura implementada no *Adobe Acrobat Pro* é equivalente ao *CAdES-BES* e este possui suporte para a estrutura *DSS* do *PAdES-LTV*.

8.4.2 Verificação da Assinaturas

Após gerar a assinatura com o software *Assinador PBAD-PAdES*, sem inserir as informações de validação, foi realizada uma tentativa de verificação com o *Adobe Reader*. Este informou que o caminho de certificação não pode ser verificado. Para resolver o problema, foi preciso configurá-lo incluindo o certificado raiz no repositório do windows e informando que este deve ser utilizado para validar a autenticidade do signatário. A configuração ocorreu da seguinte maneira: deve-se clicar em *Editar/Preferências...*, acessar o item "*Segurança*" clicar em "*Preferências Avançadas...*" e localizar a aba "*Integração com o Windows*". A figura 8.8 mostra as opções desta aba.

Ao abrir o leitor novamente, a assinatura foi verificada com sucesso conforme mostra a imagem 8.9. Pode-se perceber que o *Adobe Reader* considerou a assinatura como *CAdES-BES*, pois não exibiu nenhum aviso sobre a política de assinatura.

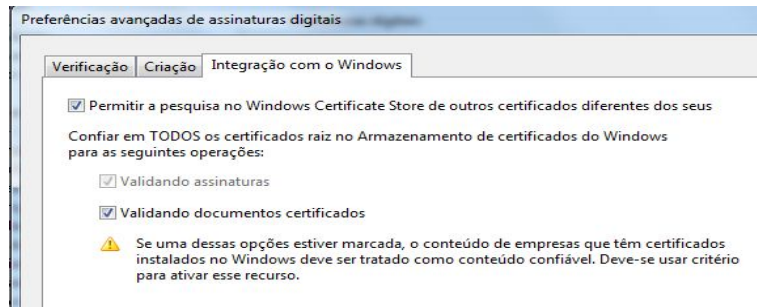


Figura 8.8: Integração com o Repositório do Windows.

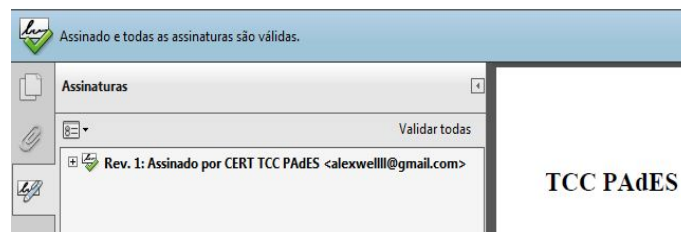


Figura 8.9: Verificação da Assinatura com o *Adobe Reader X*.

Foi gerada uma nova assinatura, e removida a lista de certificados revogados. Ao utilizar o *Adobe Reader* o mesmo exibiu a mensagem: "A assinatura é válida, mas o cancelamento da identidade do assinante não pôde ser verificada" conforme mostra a figura 8.10.

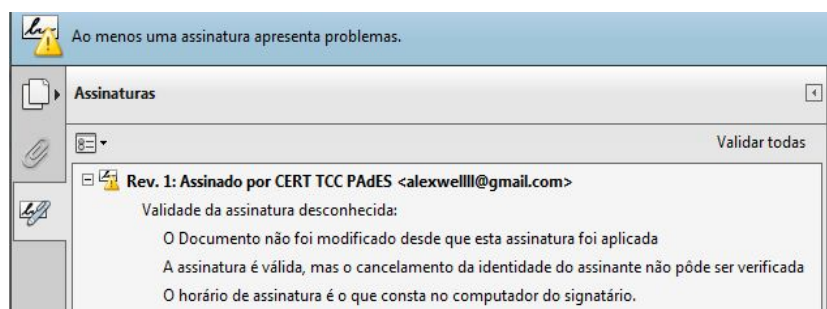


Figura 8.10: Assinatura que Exige LCR.

Utilizou-se o *Assinador PBAD-PAdES* para incluir, somente, a lista de certificados revogados na estrutura *Document Security Store DSS*. Novamente, fez-se uso

do Adobe Reader para verificar a documento. Ele validou com sucesso. Ou seja, o Adobe Reader utilizou a lista de certificados revogados que foi armazenada no *Document Security Store* DSS. Desse modo, concluí-se que o tipo de verificação de assinatura implementada no *Adobe Reader* é equivalente ao *CAdES-BES* e este possui suporte para a estrutura *DSS* do *PAdES-LTV*. O mesmo comportamento ocorreu com o *Acrobat Pro*.

Para permitir a verificação de assinaturas com política explícita deve-se desenvolver um plug-in para o Acrobat Reader. Uma opção seria através do uso do Adobe Acrobat X SDK (Systems). Porém, as aplicações devem estar em conformidade com a licença (Adobe Reader Integration Key License (RIKLA) Program) (Systems) que faz necessário obter uma chave para integração com o Reader, esta integração exige um contrato anual. A aplicação depois de confeccionada necessita ser aprovada, o que de acordo com o RIKLA demora cerca de duas semanas.

8.5 Considerações Finais do Capítulo

Este capítulo apresentou a implementação do *Assinador PBAD-PAdES* confeccionado utilizando a linguagem de programação *Java*. E mostrou uma análise da compatibilidade com o Acrobat Reader sobre o aspecto de geração de assinatura comparando as assinaturas geradas com o *Adobe Acrobat 10*.

9 Considerações Finais

A utilização do formato de representação de documentos (*PDF*) associado a eficácia probante da Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil) trará grandes avanços sobre aspectos da utilização da assinatura digital no Brasil. Assim, este trabalho apresentou uma análise sobre soluções e impactos no processo de adaptação do padrão de assinatura digital PAdES aos moldes da ICP-Brasil, avaliando as relações com as políticas assinaturas atualmente descritas sendo consideradas apenas as assinaturas CAdES. Apesar deste trabalho ter íntima ligação com aspectos de políticas de assinatura, como descrito na seção 1.5, este trabalho apresentou apenas uma proposta para os modelos de assinatura tratados pela ICP-Brasil, não definindo formatos de políticas de assinatura PAdES e nem formas de codificação de políticas para este tipo de assinatura, estes tratados como trabalhos futuros.

No capítulo 2 foi apresentado sobre criptografia assimétrica, assinatura digital e certificado digital, julgados como principais componentes envolvidos no processo de assinatura digital. Seguidos das principais propostas de padrões de assinatura digital apresentados no capítulo 3. Estas, direta ou indiretamente, fazem parte do padrão brasileiro de assinatura, visam a correta manutenção da autenticidade e integridade, prevendo inclusive a manutenção de assinaturas por longo prazo. Partiu-se então para a compreensão da Infraestrutura de Chaves Públicas Brasileira e seus formatos de assinaturas digitais no capítulo 4.

Ainda, foram apresentados, nos capítulos 5 e 6, o formato de documento portátil e o formato avançado de assinaturas digitais sobre este tipo de documento. Pôde-se perceber que uma assinatura PDF trabalha em conjunto com a estrutura do

documento, sendo a natureza incremental da assinatura PDF um fator de grande impacto para o padrão brasileiro de assinatura digital.

Após a revisão dos principais padrões de assinatura utilizados tanto no cenário nacional, com o PBAD, e em âmbito mundial, tornou-se possível avaliar algumas limitações e impactos da adoção do padrão PAdES no contexto do Padrão Brasileiro de Assinatura Digital, utilizando o perfil PAdES Enhanced (Institute 2010), sob cada política de assinatura definida na ICP-Brasil. O capítulo 7 contempla essas informações.

Por fim, o capítulo 8 apresentou uma implementação do Assinador PBAD-PAdES para gerar assinaturas no formato PBAD-PAdES(AD-RB) e uma breve análise sobre a compatibilidade do Acrobat Reader sobre o aspecto de geração de assinatura.

9.1 Contribuições

Destaca-se como principais contribuições deste trabalho:

- Revisão dos principais padrões de assinatura existentes;
- Avaliação dos impactos, mudanças e adaptações necessárias para aplicação do PAdES no âmbito da ICP-Brasil;
- Implementação de um protótipo de Software de Assinatura PBAD-PAdES com política AD-RB;
- Avaliação do suporte do Software Adobe Acrobat Reader a assinaturas digitais, sobretudo com assinaturas PAdES;
- Avaliação da compatibilidade da assinatura gerada pelo software desenvolvido, Assinador PBAD-PAdES, com o Adobe Reader X.

9.2 Trabalhos Futuros

Sugere-se como trabalhos futuros:

1. A proposição de formatos de assinaturas baseados no padrão XAdES, bem como uma análise comparativa com o CAdES.
2. Levar em consideração a parte visual das assinaturas assim como os vários tipos de conteúdos que o PDF pode encapsular.
3. Desenvolver um plug-in para o Adobe Reader, de forma que seja totalmente compatível com o padrão brasileiro de assinatura digital da ICP-Brasil;
4. Aprimorar o assinador PBAD-PAdES para suporte a outras políticas de assinatura, como AD-RC e AD-RA;
5. Avaliar a performance da verificação de assinaturas utilizando o PBAD-PAdES;
6. Realizar estudo sobre os principais ataques realizados sobre a assinatura PDF.
7. Realizar estudo sobre a representação dos atributos de uma política PAdES. Deve-se verificar se nos documentos PDF ou mesmo na especificação PAdES existem identificadores únicos para cada política, tais como o OID no CAdES. E posteriormente realizar definir uma política de assinatura e a melhor forma de codificação para o PAdES.

Referências

Adams et al. 2008 ADAMS, C.; CAIN, P.; PINKAS, D.; ZUCCHERATO, R. *Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)*. [S.l.]: IETF, 2008. RFC 3161 (Proposed Standard). (Request for Comments, 3161).

BRASIL 2001 BRASIL. *Medida provisória n.º 2.200, de 24 de agosto de 2001. Institui a Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil, transforma o Instituto Nacional de Tecnologia da Informação em autarquia, e dá outras providências*. [S.l.]: Diário Oficial da União, Brasília, DF, 27 ago. de 2001.

BRASIL 2001 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*. [S.l.]: Diário Oficial da União, Brasília, DF, 29 jun. de 2001.

BRASIL 2001 BRASIL. *Projeto de Lei n.º 4.906, de 21 de junho de 2001. Dispõe sobre o valor probante do documento eletrônico e da assinatura digital, regula a certificação digital, institui normas para as transações de comércio eletrônico e dá outras providências*. [S.l.]: Diário Oficial da União, Brasília, DF, 21 jun. de 2001.

BRASIL 2006 BRASIL. *Lei n.º 11.419, de 19 de dezembro de 2006. Dispõe sobre a informatização do processo judicial; altera a Lei no 5.869, de 11 de janeiro de 1973 - Código de Processo Civil; e dá outras providências*. [S.l.]: Diário Oficial da União, Brasília, DF, 20 dez. de 2006.

Buckland 1997 BUCKLAND, M. K. What is a "document"? *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 1997.

Castle CASTLE, B. *The Legion of the Bouncy Castle*. Disponível em: <<http://www.bouncycastle.org/>>. Acessado em: 08 de maio de 2012.

CCITT 1988 CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1). ITU-T, 1988.

CCITT 1988 CCITT. Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). ITU-T, 1988.

CCITT 1988 CCITT. Recommendation X.509: The Directory - Authentication Framework. ITU-T, 1988.

Cooper et al. 2008 COOPER, D.; SANTESSON, S.; FARRELL, S.; BOEYEN, S.; HOUSLEY, R.; POLK, W. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. [S.l.]: IETF, maio 2008. RFC 5280 (Proposed Standard). (Request for Comments, 5280).

Corp. CORP. iText S. *iText*. Disponível em: <<http://itextpdf.com/>>. Acessado em: 15 de maio de 2012.

Diffie e Hellman 1976 DIFFIE, W.; HELLMAN, M. E. New directions in cryptography. *IEEE Transactions on Information Theory*, Invited Paper, S.L, n. 6, p. 1–12, 1976.

Housley 2004 HOUSLEY, R. *Cryptographic Message Syntax (CMS)*. [S.l.]: IETF, 2004. RFC 3852 (Proposed Standard). (Request for Comments, 3852).

Housley 2009 HOUSLEY, R. *Cryptographic Message Syntax (CMS)*. [S.l.]: IETF, set. 2009. RFC 5652 (Proposed Standard). (Request for Comments, 5652).

IETF/W3C IETF/W3C. *XML-Signature Syntax and Processing*. 27 de fevereiro de 2012. <http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/>.

IETF/W3C IETF/W3C. *XML Signature Syntax and Processing (Second Edition)*. 10 de junho de 2008. <http://www.w3.org/TR/xmlsig-core/>.

Incorporated 1999 INCORPORATED, A. S. *PostScript Language Reference Manual*. 3rd. ed. [S.l.]: ADOBE, 1999. 912 p. Reference Manual. ISBN 0-201-37922-8.

Informação INFORMAÇÃO, I. N. de Tecnologia da. *DOC-ICP. Disponível em: <<http://www.itl.gov.br/twiki/bin/view/Certificacao/DocIcp>>. Acessado em: 22 de maio de 2012.*

Institute 2002 INSTITUTE, E. T. S. Technical Specification, *TC Security - Electronic Signatures and Infrastructures (ESI); XML format for signature policies*. April 2002.

Institute 2003 INSTITUTE, E. T. S. Technical Specification, *Electronic Signatures and Infrastructures (ESI); ASN.1 format for signature policies*. December 2003.

Institute 2009 INSTITUTE, E. T. S. Technical Specification, *Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 5: PAdES for XML Content - Profiles for XAdES signatures*. December 2009.

Institute 2009 INSTITUTE, E. T. S. Technical Specification, *ETSI TS 102 778-1 V1.1.1 Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 1: PAdES Overview - a framework document for PAdES*. July 2009.

Institute 2009 INSTITUTE, E. T. S. Technical Specification, *ETSI TS 102 778-2 V1.2.1 Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 1: PAdES Overview - a framework document for PAdES*. July 2009.

Institute 2009 INSTITUTE, E. T. S. Technical Specification, *ETSI TS 102 778-4 V1.1.2 Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 4: PAdES Long Term - PAdES-LTV Profile*. July 2009.

INSTITUTE 2010 INSTITUTE, E. T. S. *Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)*. [S.l.]: ETSI, 2010. ETSI TS 101 733 V1.8.3 (2011-01).

Institute 2010 INSTITUTE, E. T. S. Technical Specification, *Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 6: Visual Representations of Electronic Signatures*. December 2010.

INSTITUTE 2010 INSTITUTE, E. T. S. *Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES)*. [S.l.]: ETSI, 2010. ETSI TS 101 903 V1.4.2 (2010-12).

Institute 2010 INSTITUTE, E. T. S. Technical Specification, *ETSI TS 102 778-3 v1.2.1 Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 3: PAdES Enhanced - PAdES-BES and PAdES-EPES Profiles*. July 2010.

ITI ITI. *PERFIL DE USO GERAL PARA ASSINATURAS DIGITAIS NA ICP-BRASIL. Versão 2.0. Brasília, Abril 2010. DOC-ICP-15.02*. [S.l.].

ITI ITI. *REQUISITOS DAS POLÍTICAS DE ASSINATURA DIGITAL NA ICP-BRASIL. Versão 5.0. Brasília, Março 2012. DOC-ICP-15.03*. [S.l.].

ITI ITI. *REQUISITOS PARA GERAÇÃO E VERIFICAÇÃO DE ASSINATURAS DIGITAIS NA ICP-BRASIL. Versão 2.0. Brasília, Abril 2010. DOC-ICP-15.01*. [S.l.].

ITI ITI. *VISÃO GERAL SOBRE ASSINATURAS DIGITAIS NA ICP-BRASIL. Versão 2.0. Brasília, Abril 2010. DOC-ICP-15*. [S.l.].

Kaliski 1998 KALISKI, B. *PKCS #7: Cryptographic Message Syntax Version 1.5*. [S.l.]: IETF, 1998. RFC 3852 (Proposed Standard). (Request for Comments, 2315).

Moecke MOECKE, C. T. *ASSINATURA DIGITAL DE DOCUMENTOS ELETRÔNICOS NA ICP-BRASIL. 2008. p. 103. Monografia de Conclusão de Curso (Ciências da Computação) - Universidade Federal de Santa Catarina, Florianópolis*.

NIST 1994 NIST. *Digital Signature Standard*. [S.l.], May 1994.

Standard 2008 STANDARD, I. *Document management –Portable document format –Part 1: PDF 1.7*. [S.l.]: ISO, 2008. 756 p. ISO 32000-1.

Steve e PKI 2002 STEVE, L.; PKI, F. *Understanding Certification Path Construction*. [S.l.]: PKI Forum, 2002. White Paper.

Systems SYSTEMS, A. *Acrobat Developer Center*. Disponível em: <<http://www.adobe.com/devnet/acrobat.html>>. Acessado em: 16 de maio de 2012.

Systems SYSTEMS, A. *Adobe Acrobat*. Disponível em: <<http://www.adobe.com/products/acrobat.html>>. Acessado em: 15 de maio de 2012.

Systems SYSTEMS, A. *Adobe Reader Integration Key License (RIKLA) Program*. Disponível em: <<http://www.adobe.com/devnet/reader/ikla.html>>. Acessado em: 16 de maio de 2012.

Systems SYSTEMS, A. *Establish long-term signature validation*. Disponível em: <http://help.adobe.com/en_US/acrobat/pro/using/WS934c23d7cc8877da1172e0811fde233c98-8000.html#WS934c23d7cc8877da1172e0811fde3067ed-7fff>. Acessado em: 15 de maio de 2012.

Technical 1993 TECHNICAL, R. L. *PKCS #7 - Cryptographic Message Syntax Standard*. [S.l.], November 1993.

Technical 2002 TECHNICAL, R. L. *PKCS #1: RSA Cryptography Standard v2.1*. [S.l.], 2002.

A Apêndice A –Código Fonte

```
1 package core ;
3 import java . io . ByteArrayOutputStream ;
import java . io . FileInputStream ;
5 import java . io . FileOutputStream ;
import java . io . IOException ;
7 import java . math . BigInteger ;
import java . security . KeyStore ;
9 import java . security . MessageDigest ;
import java . security . PrivateKey ;
11 import java . security . cert . CertStore ;
import java . security . cert . Certificate ;
13 import java . security . cert . CollectionCertStoreParameters ;
import java . security . cert . X509Certificate ;
15 import java . util . ArrayList ;
import java . util . Arrays ;
17 import java . util . Collection ;
import java . util . GregorianCalendar ;
19 import java . util . HashMap ;
import java . util . Iterator ;
21 import java . util . Properties ;

23 import org . bouncecastle . asn1 . ASN1Encodable ;
import org . bouncecastle . asn1 . ASN1EncodableVector ;
25 import org . bouncecastle . asn1 . ASN1InputStream ;
import org . bouncecastle . asn1 . ASN1Integer ;
27 import org . bouncecastle . asn1 . ASN1Object ;
import org . bouncecastle . asn1 . ASN1ObjectIdentifier ;
29 import org . bouncecastle . asn1 . ASN1OctetString ;
import org . bouncecastle . asn1 . ASN1Set ;
31 import org . bouncecastle . asn1 . DERIA5String ;
import org . bouncecastle . asn1 . DEROctetString ;
33 import org . bouncecastle . asn1 . DEROutputStream ;
import org . bouncecastle . asn1 . DERSet ;
35 import org . bouncecastle . asn1 . cms . Attribute ;
import org . bouncecastle . asn1 . cms . AttributeTable ;
37 import org . bouncecastle . asn1 . esf . OtherHashAlgAndValue ;
import org . bouncecastle . asn1 . esf . SPuri ;
39 import org . bouncecastle . asn1 . esf . SigPolicyQualifierInfo ;
import org . bouncecastle . asn1 . esf . SigPolicyQualifiers ;
41 import org . bouncecastle . asn1 . esf . SignaturePolicyId ;
import org . bouncecastle . asn1 . esf . SignaturePolicyIdentifier ;
43 import org . bouncecastle . asn1 . ess . ESSCertIDv2 ;
import org . bouncecastle . asn1 . ess . SigningCertificateV2 ;
45 import org . bouncecastle . asn1 . pkcs . PKCSObjectIdentifiers ;
import org . bouncecastle . asn1 . x509 . AlgorithmIdentifier ;
47 import org . bouncecastle . asn1 . x509 . GeneralName ;
import org . bouncecastle . asn1 . x509 . GeneralNames ;
49 import org . bouncecastle . asn1 . x509 . IssuerSerial ;
import org . bouncecastle . cert . X509CertificateHolder ;
51 import org . bouncecastle . cms . CMSProcessable ;
```

```

import org.bouncycastle.cms.CMSSignedData;
53 import org.bouncycastle.cms.CMSSignedDataGenerator;
import org.bouncycastle.cms.CMSSignedDataParser;
55 import org.bouncycastle.cms.SignerInformation;
import org.bouncycastle.cms.SignerInformationStore;
57 import org.bouncycastle.jce.PrincipalUtil;
import org.bouncycastle.jce.X509Principal;
59 import org.bouncycastle.operator.jcajce.
    JcaDigestCalculatorProviderBuilder;
import org.bouncycastle.util.Store;
61
import util.MyCMSProcessableRange;
63 import util.MyCMSSignedDataGenerator;
import util.MyCrlClientImp;
65
import com.itextpdf.text.pdf.AcroFields;
67 import com.itextpdf.text.pdf.LtvVerification;
import com.itextpdf.text.pdf.PdfDate;
69 import com.itextpdf.text.pdf.PdfDictionary;
import com.itextpdf.text.pdf.PdfName;
71 import com.itextpdf.text.pdf.PdfNumber;
import com.itextpdf.text.pdf.PdfObject;
73 import com.itextpdf.text.pdf.PdfPKCS7;
import com.itextpdf.text.pdf.PdfReader;
75 import com.itextpdf.text.pdf.PdfSignature;
import com.itextpdf.text.pdf.PdfSignatureAppearance;
77 import com.itextpdf.text.pdf.PdfStamper;
import com.itextpdf.text.pdf.PdfString;
79
public class AssinadorPAdES {
81     private static String OID_ALG_SHA256 = "2.16.840.1.101.3.4.2.1";
        // SHA-256 http://tools.ietf.org/html/rfc5754#section-2.2
        private static String POLITICA_ADRB_OID = "2.16.76.1.7.1.1.1";
83     private static byte[] POLITICA_ADRB_HASH = { 1,2,3,4,5,6,7,8,9,10,
        1,2,3,4,5,6,7,8,9,10,
85         1,2,3,4,5,6,7,8,9,10,
        1,2};
87
    public static Properties properties = new Properties();
89
    public void adicionarDSS(String entrada, String saida)
91    {
        try
93        {
            //String caminhoLCR = properties.getProperty("LCR");
95
            PdfReader r = new PdfReader(entrada);
97            FileOutputStream fout = new FileOutputStream(saida);
            PdfStamper stp = new PdfStamper(r, fout, '\0', true);
99            LtvVerification v = stp.getLtvVerification();
            AcroFields af = stp.getAcroFields();
101            for (String sigName : af.getSignatureNames())
                {
103                v.addVerification(sigName,
                    null,
105                    new MyCrlClientImp(),
                    LtvVerification.CertificateOption.WHOLE_CHAIN,
107                    LtvVerification.Level.CRL,
                    LtvVerification.CertificateInclusion.NO);
109                }
            stp.close();
111        }
        catch (Exception ex)
113        {
            ;
115        }
117    }

```

```

119 public String assinar(String entrada, boolean criarRevisao)
120 {
121     String retorno = "";
122     String password = properties.getProperty("PASSWORD");
123     String p12 = properties.getProperty("PKCS12");
124     String saida = entrada + "_assinado.pdf";
125     try
126     {
127         KeyStore ks = KeyStore.getInstance("PKCS12");
128         ks.load(new FileInputStream(p12), password.toCharArray());
129         String alias = (String)ks.aliases().nextElement();
130         PrivateKey key = (PrivateKey)ks.getKey(alias, password.
131             toCharArray());
132         Certificate[] chain = ks.getCertificateChain(alias);
133
134         PdfReader reader = new PdfReader(entrada);
135         PdfStamper stp = PdfStamper.createSignature(reader, new
136             FileOutputStream(saida), '\0', null, criarRevisao); // Cria
137             uma nova revisão para incluir multiplas assinaturas, null,
138             true);
139
140         // Adiciona Extensão no dicionário, conforme seção 4.7 da
141         // Parte 3 do PAdES
142         PdfDictionary pdfDic = new PdfDictionary();
143         pdfDic.put(PdfName.BASEVERSION, new PdfName("1.7"));
144         pdfDic.put(PdfName.EXTENSIONLEVEL, new PdfNumber(2));
145         PdfDictionary auxPdfDic = new PdfDictionary();
146         auxPdfDic.put(new PdfName("ESIC"), pdfDic);
147         reader.getCatalog().put(new PdfName("Extensions"), auxPdfDic);
148
149         PdfSignatureAppearance sap = stp.getSignatureAppearance();
150         //sap.setVisibleSignature(new Rectangle(100, 100, 300, 200),
151             1, null);
152         sap.setSigDate(new GregorianCalendar());
153         sap.setCrypto(null, chain, null, PdfSignatureAppearance.
154             WINCER_SIGNED);
155         sap.setReason("Razão Assinatura");
156         sap.setLocation("Local Assinatura");
157         sap.setAcro6Layers(true);
158         //sap.setRenderingMode(PdfSignatureAppearance.RenderingMode.
159             DESCRIPTION);
160
161         PdfSignature dic = new PdfSignature(PdfName.ADOBE_PPCLITE, new
162             PdfName("ETSI.CAdES.detached"));
163
164         dic.setDate(new PdfDate(sap.getSigDate()));
165         dic.setName(PdfPKCS7.getSubjectFields((X509Certificate)chain
166             [0]).getField("CN"));
167         if (sap.getReason() != null)
168             dic.setReason(sap.getReason());
169         if (sap.getLocation() != null)
170             dic.setLocation(sap.getLocation());
171         sap.setCryptoDictionary(dic);
172         int csize = 4000;
173         HashMap exc = new HashMap();
174         exc.put(PdfName.CONTENTS, new Integer(csize * 2 + 2));
175         sap.preClose(exc);
176
177         MyCMSSignedDataGenerator generator = new
178             MyCMSSignedDataGenerator();
179
180         ASN1EncodableVector atributos = new ASN1EncodableVector();
181         atributos.add(gerarSigPolicyId());
182         atributos.add(gerarSigningCertificateV2(chain[0]));
183
184         AttributeTable signedAttr = new AttributeTable(atributos);
185         generator.addSigner(key, (X509Certificate)chain[0],
186             CMSSignedDataGenerator.DIGEST_SHA256, signedAttr, null);

```

```

177     ArrayList list = new ArrayList();
178     for (int i = 0; i < chain.length; i++)
179         list.add(chain[i]);
181     CertStore chainStore = CertStore.getInstance("Collection", new
        CollectionCertStoreParameters(list), "BC");
182     generator.addCertificatesAndCRLs(chainStore);
183     CMSSignedData signedData;
185     CMSProcessable content = new MyCMSProcessableRange(sap);
186     signedData = generator.generate(content, false, "BC");
187
188     byte[] pk = convertToDefiniteLength(signedData.getEncoded());
189
190     try
191     {
192         FileOutputStream fos = new FileOutputStream(saida+".pkcs7");
193         fos.write(pk);
194         fos.close();
195     }
196     catch (Exception ex)
197     {
198         throw new Exception("IOException: Erro ao salvar .pkcs7");
199     }
201     byte[] outc = new byte[csize];
202
203     PdfDictionary dic2 = new PdfDictionary();
204
205     System.arraycopy(pk, 0, outc, 0, pk.length);
206
207     dic2.put(PdfName.CONTENTS, new PdfString(outc).setHexWriting(
        true));
208     sap.close(dic2);
209
210     retorno = saida+"\n"+ saida+".pkcs7";
211 }
212 catch (Exception ex)
213 {
214     retorno = ex.getMessage();
215 }
216
217 return retorno;
218 }
219
220 public Attribute gerarSigPolicyId()
221 {
222     AlgorithmIdentifier algId = new AlgorithmIdentifier(
        OID_ALG_SHA256);
223     DEROctetString hashValue = new DEROctetString(POLITICA_ADRB_HASH
        );
224     OtherHashAlgAndValue hashAlgAndValue = new org.bouncycastle.asn1
        .esf.OtherHashAlgAndValue(algId, hashValue);
225
226     SPuri spuri = new SPuri(new DERIA5String("http://endereco.
        politica.com.br/politica_pades.der"));
227     // OID uri "1.2.840.113549.1.9.16.5.1" ETSI TS 101 733 secao:
        // [5.8.1] id-spq-ets-uri OBJECT IDENTIFIER ::= { iso(1) member-
        // body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id-
        // -sq(5) 1 }
228     SigPolicyQualifierInfo spqiArray[] = new SigPolicyQualifierInfo
        [1];
229     spqiArray[0] = new SigPolicyQualifierInfo(new
        ASN1ObjectIdentifier("1.2.840.113549.1.9.16.5.1"), spuri.
        toASN1Primitive());
230
231     SignaturePolicyId sigPolicyId = new SignaturePolicyId(new
        ASN1ObjectIdentifier(POLITICA_ADRB_OID), hashAlgAndValue, new

```



```

        SigPolicyQualifiers (spqiArray));
SignaturePolicyIdentifier sigPolicyIdentifier = new
233 SignaturePolicyIdentifier (sigPolicyId);

Attribute attr = new Attribute (PKCSObjectIdentifiers.
235 id_aa_ets_sigPolicyId, new DERSet (sigPolicyIdentifier));

return attr;
237 }

public Attribute gerarSigningCertificateV2 (Certificate signer)
239 {
Attribute retorno = null;
241 try
{
243 MessageDigest md = MessageDigest.getInstance ("SHA-256");
md.update (signer.getEncoded ());
245 byte [] hashValue = md.digest ();

247 BigInteger serialnumber = ((X509Certificate) signer).
getSerialNumber ();

249 X509Principal principal = PrincipalUtil.getIssuerX509Principal
(((X509Certificate) signer));
251 GeneralNames issuer = new GeneralNames (new GeneralName (
principal));
IssuerSerial issuerSerial = new IssuerSerial (issuer, new
ASN1Integer (serialnumber));
253 ESSCertIDv2 cert = new ESSCertIDv2 (new AlgorithmIdentifier (
OID_ALG_SHA256), hashValue, issuerSerial);
ESSCertIDv2 [] certs = new ESSCertIDv2 [1];
255 certs [0] = cert;
SigningCertificateV2 signCert = new SigningCertificateV2 (certs
);
257 retorno = new Attribute (PKCSObjectIdentifiers.
id_aa_signingCertificateV2, new DERSet (signCert));
}
259 catch (Exception ex)
{
261 ex.printStackTrace ();
}

return retorno;
263 }

265 }

public String verificar (String entrada)
267 {
String retorno = "";
269 try {
271 KeyStore kall = PdfPKCS7.loadCacertsKeyStore ();
PdfReader reader = new PdfReader (entrada);
273 AcroFields af = reader.getAcroFields ();
ArrayList names = af.getSignatureNames ();
275 for (int k = 0; k < names.size (); ++k) {
String name = (String) names.get (k);
277 retorno = "Nome da Assinatura: " + name;
retorno += "\nAssinatura sobre todo o documento: " + af.
signatureCoversWholeDocument (name);
279 retorno += "\nRevisão: " + af.getRevision (name) + " / " +
af.getTotalRevisions ();

281 // Start revision extraction
// FileOutputStream out = new FileOutputStream ("revisão_" +
af.getRevision (name) + ".pdf");
283 // byte bb[] = new byte [8192];
// InputStream ip = af.extractRevision (name);
285 // int n = 0;
// while ((n = ip.read (bb)) > 0)
287 // out.write (bb, 0, n);

```

```

289 //         out.close();
//         ip.close();
// End revision extraction
291 PdfPKCS7 pk = af.verifySignature(name);
293
295 retorno += "\nIntegridade: " + pk.verify();
295 retorno += "\nData: " + pk.getSignDate().getTime();
297 // Extrai Estrutura CAdES do Dicionário de assinatura para
// utilizar a BouncyCastle
299 PdfDictionary dic = af.getSignatureDictionary(name);
PdfObject pdfObject = dic.get(new PdfName("Contents"));
301 CMSSignedDataParser sp = new CMSSignedDataParser(new
    JcaDigestCalculatorProviderBuilder().setProvider("BC").
    build(), pdfObject.getBytes());
Store certStore = sp.getCertificates();
303 SignerInformationStore signers = sp.getSignerInfos();
Collection c = signers.getSigners();
305 Iterator it = c.iterator();
307 while (it.hasNext())
309 {
    SignerInformation signer = (SignerInformation)it.
        next();
311
    Collection certCollection = certStore.getMatches(
        signer.getSID());
    Iterator certIt = certCollection.iterator();
313 X509CertificateHolder cert = (X509CertificateHolder)
        certIt.next();
315
    AttributeTable signedAttr = signer.getSignedAttributes
        ();
317
    Attribute sigPolicyIdentifier = signedAttr.get(
        PKCSObjectIdentifiers.id_aa_ets_sigPolicyId);
    Attribute signingCertificateV2 = signedAttr.get(
        PKCSObjectIdentifiers.id_aa_signingCertificateV2);
319 retorno += "\n" + validarPolitica(sigPolicyIdentifier);
retorno += "\n" + validarSigningCertificateV2(
    signingCertificateV2, cert);
321 retorno += "\n";
    }
323 }
325 catch (Exception ex)
327 {
    retorno = ex.getMessage();
329 }
return retorno;
331 }
333 public String validarPolitica(Attribute sigPolicyIdentifier)
    throws Exception
335 {
    String retorno = "";
    try
337 {
        if(sigPolicyIdentifier == null)
339         throw new Exception("Não Possui Política Explícita.");
341 ASN1Set attr = sigPolicyIdentifier.getAttrValues();
343 ASN1Encodable dob = attr.getObjectAt(0);
    SignaturePolicyIdentifier auxsigPolicyIdentifier =
        SignaturePolicyIdentifier.getInstance(dob);

```

```

345     SignaturePolicyId sigPolicyId = auxsigPolicyIdentifier.
346         getSignaturePolicyId();
347
348     if(!sigPolicyId.getSigPolicyId().equals(new
349         ASN1ObjectIdentifier(POLITICA_ADRB_OID)))
350         throw new Exception("Erro: Política Desconhecida.");
351
352     OtherHashAlgAndValue hashAlgAndValue = sigPolicyId.
353         getSigPolicyHash();
354     ASN1OctetString asn1Hash = hashAlgAndValue.getHashValue();
355
356     if(!Arrays.equals(asn1Hash.getOctets(), POLITICA_ADRB_HASH))
357         throw new Exception("Erro: Hash Política Desconhecida.");
358
359     retorno = "Política: AD-RB";
360 }
361 catch(Exception ex)
362 {
363     retorno = ex.getMessage();
364 }
365
366 return retorno;
367 }
368
369 public String validarSigningCertificateV2(Attribute
370     signingCertificateV2, X509CertificateHolder signer) throws
371     Exception
372 {
373     String retorno = "";
374     try
375     {
376         if(signingCertificateV2 == null)
377             throw new Exception("Não Possui SigningCertificate.");
378
379         ASN1Set attr = signingCertificateV2.getAttrValues();
380
381         ASN1Encodable dob = attr.getObjectAt(0);
382         SigningCertificateV2 auxsigningCertificateV2 =
383             SigningCertificateV2.getInstance(dob);
384
385         ESSCertIDv2[] certs = auxsigningCertificateV2.getCerts();
386         ESSCertIDv2 cert = certs[0];
387         byte[] certHash = cert.getCertHash();
388
389         MessageDigest md = MessageDigest.getInstance("SHA-256");
390         md.update(signer.getEncoded());
391         byte[] hashSignatario = md.digest();
392
393         if(!Arrays.equals(certHash, hashSignatario))
394             throw new Exception("Erro: SigningCertificate (Hash Inválido
395                 ).");
396
397         IssuerSerial issuerSerial = cert.getIssuerSerial();
398
399         BigInteger serialnumber = signer.getSerialNumber();
400         GeneralNames issuer = new GeneralNames(new GeneralName(signer.
401             getIssuer()));
402         IssuerSerial issuerSerialSignatario = new IssuerSerial(issuer,
403             new ASN1Integer(serialnumber));
404
405         if(!issuerSerial.equals(issuerSerialSignatario))
406             throw new Exception("Erro: SigningCertificate (IssuerSerial
407                 Inválido).");
408
409         retorno = "SigningCertificate: Válido";
410     }
411     catch(Exception ex)
412     {

```

```

405     retorno = ex.getMessage();
406     }
407     return retorno;
408     }
409
410 private byte[] convertToDefiniteLength(byte[] pk) throws Exception
411 {
412     ByteArrayOutputStream bOut = new ByteArrayOutputStream();
413     DEROutputStream      dOut = new DEROutputStream(bOut);
414     dOut.writeObject(( fromByteArray(pk) ));
415     dOut.close();
416     byte[] retorno = bOut.toByteArray();
417
418     return retorno;
419 }
420
421 private ASN1Object fromByteArray(byte[] data)
422     throws IOException
423 {
424     ASN1InputStream aIn = new ASN1InputStream(data);
425
426     try
427     {
428         return (ASN1Object) aIn.readObject();
429     }
430     catch (ClassCastException e)
431     {
432         throw new IOException("cannot recognise object in stream
433                                ");
434     }
435 }

```

implementacao/ProjetoPADES_ICPBrasil/src/core/AssinadorPADES.java

```

1 package core;
2 import java.io.FileOutputStream;
3
4 import com.itextpdf.text.Anchor;
5 import com.itextpdf.text.BadElementException;
6 import com.itextpdf.text.BaseColor;
7 import com.itextpdf.text.Chapter;
8 import com.itextpdf.text.Document;
9 import com.itextpdf.text.DocumentException;
10 import com.itextpdf.text.Element;
11 import com.itextpdf.text.Font;
12 import com.itextpdf.text.List;
13 import com.itextpdf.text.ListItem;
14 import com.itextpdf.text.Paragraph;
15 import com.itextpdf.text.Phrase;
16 import com.itextpdf.text.Section;
17 import com.itextpdf.text.pdf.PdfDeveloperExtension;
18 import com.itextpdf.text.pdf.PdfName;
19 import com.itextpdf.text.pdf.PdfPCell;
20 import com.itextpdf.text.pdf.PdfPTable;
21 import com.itextpdf.text.pdf.PdfWriter;
22
23
24 public class PdfSimples {
25
26     private static Font catFont = new Font(Font.FontFamily.TIMES_ROMAN
27         , 18,
28         Font.BOLD);
29     private static Font redFont = new Font(Font.FontFamily.TIMES_ROMAN
30         , 12,
31         Font.NORMAL, BaseColor.RED);
32     private static Font subFont = new Font(Font.FontFamily.TIMES_ROMAN
33         , 16,

```

```

    Font.BOLD);
32 private static Font smallBold = new Font(Font.FontFamily.
    TIMES_ROMAN, 12,
    Font.BOLD);
34
36 public static void criar(String titulo , String caminho)
    {
38     try {
        Document document = new Document();
40     PdfWriter.getInstance(document, new FileOutputStream(caminho))
        ; // .addDeveloperExtension(new PdfDeveloperExtension(new
        PdfName("ESIC"), PdfWriter.PDF_VERSION_1_7, 2));
        document.open();
42     //addMetaData(document);
        addTitlePage(document, titulo);
44     //addContent(document);
        document.close();
46     } catch (Exception e) {
        e.printStackTrace();
48     }
50 }
52 // iText allows to add metadata to the PDF which can be viewed in
    your Adobe
    // Reader
    // under File -> Properties
54 private static void addMetaData(Document document) {
56     document.setTitle("My first PDF");
        document.addSubject("Using iText");
58     document.addKeywords("Java, PDF, iText");
        document.addAuthor("Lars Vogel");
60     document.addCreator("Lars Vogel");
    }
62
64 private static void addTitlePage(Document document, String titulo)
    throws DocumentException {
    Paragraph preface = new Paragraph();
66     // We add one empty line
    addEmptyLine(preface, 1);
68     // Lets write a big header
    preface.add(new Paragraph(titulo, catFont));
70
72     //     addEmptyLine(preface, 1);
        // Will create: Report generated by: _name, _date
        //     preface.add(new Paragraph(
74     //         "Report generated by: " + System.getProperty("user.
        name") + ", " + new Date(), //$NON-NLS-1$ //$NON-NLS-2$ //
        //$NON-NLS-3$
        //         smallBold));
76     //     addEmptyLine(preface, 3);
        //     preface.add(new Paragraph(
78     //         "This document describes something which is very
        important ",
        //         smallBold));
80     //
        //     addEmptyLine(preface, 8);
82     //
        //     preface.add(new Paragraph(
84     //         "This document is a preliminary version and not
        subject to your license agreement or any other agreement with
        vogella.com ;-).",
        //         redFont));
86
88     document.add(preface);
    // Start a new page
    document.newPage();

```

```

90 }
91
92 private static void addContent(Document document) throws
    DocumentException {
93     Anchor anchor = new Anchor("First Chapter", catFont);
94     anchor.setName("First Chapter");
95
96     // Second parameter is the number of the chapter
97     Chapter catPart = new Chapter(new Paragraph(anchor), 1);
98
99     Paragraph subPara = new Paragraph("Subcategory 1", subFont);
100    Section subCatPart = catPart.addSection(subPara);
101    subCatPart.add(new Paragraph("Hello"));
102
103    subPara = new Paragraph("Subcategory 2", subFont);
104    subCatPart = catPart.addSection(subPara);
105    subCatPart.add(new Paragraph("Paragraph 1"));
106    subCatPart.add(new Paragraph("Paragraph 2"));
107    subCatPart.add(new Paragraph("Paragraph 3"));
108
109    // Add a list
110    createList(subCatPart);
111    Paragraph paragraph = new Paragraph();
112    addEmptyLine(paragraph, 5);
113    subCatPart.add(paragraph);
114
115    // Add a table
116    createTable(subCatPart);
117
118    // Now add all this to the document
119    document.add(catPart);
120
121    // Next section
122    anchor = new Anchor("Second Chapter", catFont);
123    anchor.setName("Second Chapter");
124
125    // Second parameter is the number of the chapter
126    catPart = new Chapter(new Paragraph(anchor), 1);
127
128    subPara = new Paragraph("Subcategory", subFont);
129    subCatPart = catPart.addSection(subPara);
130    subCatPart.add(new Paragraph("This is a very important message")
131        );
132
133    // Now add all this to the document
134    document.add(catPart);
135
136 }
137
138 private static void createTable(Section subCatPart)
139     throws BadElementException {
140     PdfPTable table = new PdfPTable(3);
141
142     // t.setBorderColor(BaseColor.GRAY);
143     // t.setPadding(4);
144     // t.setSpacing(4);
145     // t.setBorderWidth(1);
146
147     PdfPCell c1 = new PdfPCell(new Phrase("Table Header 1"));
148     c1.setHorizontalAlignment(Element.ALIGN_CENTER);
149     table.addCell(c1);
150
151     c1 = new PdfPCell(new Phrase("Table Header 2"));
152     c1.setHorizontalAlignment(Element.ALIGN_CENTER);
153     table.addCell(c1);
154
155     c1 = new PdfPCell(new Phrase("Table Header 3"));
156     c1.setHorizontalAlignment(Element.ALIGN_CENTER);
157     table.addCell(c1);
158     table.setHeaderRows(1);

```

```

158     table.addCell("1.0");
160     table.addCell("1.1");
162     table.addCell("1.2");
164     table.addCell("2.1");
166     table.addCell("2.2");
168     table.addCell("2.3");
170     subCatPart.add(table);
172 }
174 private static void createList(Section subCatPart) {
176     List list = new List(true, false, 10);
178     list.add(new ListItem("First point"));
180     list.add(new ListItem("Second point"));
182     list.add(new ListItem("Third point"));
184     subCatPart.add(list);
186 }
188 private static void addEmptyLine(Paragraph paragraph, int number)
190 {
192     for (int i = 0; i < number; i++) {
194         paragraph.add(new Paragraph(" "));
196     }
198 }
200 }

```

implementacao/ProjetoPAdES_ICPBrasil/src/core/PdfSimples.java

```

1 package util;
2 import java.io.IOException;
3 import java.io.InputStream;
4
5 import org.bouncycastle.cms.CMSEException;
6 import org.bouncycastle.cms.CMSProcessable;
7
8 import com.itextpdf.text.pdf.PdfSignatureAppearance;
9
10 public class MyCMSProcessableRange implements CMSProcessable {
11     private PdfSignatureAppearance sap;
12     private byte[] buf = new byte[8192];
13
14     public MyCMSProcessableRange(PdfSignatureAppearance sap) {
15         this.sap = sap;
16     }
17
18     public void write(java.io.OutputStream outputStream) throws
19         IOException, CMSEException {
20
21         InputStream s = sap.getRangeStream();
22         int read = 0;
23         while ((read = s.read(buf, 0, buf.length)) > 0) {
24             outputStream.write(buf, 0, read);
25         }
26     }
27
28     public Object getContent() {
29         return sap;
30     }
31 }

```

implementacao/ProjetoPAdES_ICPBrasil/src/util/MyCMSProcessableRange.java

```

1 package util;
2 import java.io.ByteArrayOutputStream;
3 import java.io.IOException;

```

```

import java.io.OutputStream;
5 import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
7 import java.security.PrivateKey;
import java.security.Provider;
9 import java.security.SecureRandom;
import java.security.cert.CertificateEncodingException;
11 import java.security.cert.X509Certificate;
import java.util.ArrayList;
13 import java.util.Iterator;
import java.util.List;
15
import org.bouncycastle.asn1.ASN1EncodableVector;
17 import org.bouncycastle.asn1.ASN1ObjectIdentifier;
import org.bouncycastle.asn1.ASN1OctetString;
19 import org.bouncycastle.asn1.ASN1Set;
import org.bouncycastle.asn1.BEROctetString;
21 import org.bouncycastle.asn1.DERSet;
import org.bouncycastle.asn1.cms.AttributeTable;
23 import org.bouncycastle.asn1.cms.CMSObjectIdentifiers;
import org.bouncycastle.asn1.cms.ContentInfo;
25 import org.bouncycastle.asn1.cms.SignedData;
import org.bouncycastle.asn1.cms.SignerInfo;
27 import org.bouncycastle.cms.CMSAbsentContent;
import org.bouncycastle.cms.CMSAttributeTableGenerator;
29 import org.bouncycastle.cms.CMSException;
import org.bouncycastle.cms.CMSProcessable;
31 import org.bouncycastle.cms.CMSProcessableByteArray;
import org.bouncycastle.cms.CMSSignedData;
33 import org.bouncycastle.cms.CMSSignedGenerator;
import org.bouncycastle.cms.CMSTypedData;
35 import org.bouncycastle.cms.SignerInfoGenerator;
import org.bouncycastle.cms.SignerInformation;
37 import org.bouncycastle.cms.SignerInformationStore;
import org.bouncycastle.cms.SimpleAttributeTableGenerator;
39 import org.bouncycastle.cms.jcajce.JcaSignerInfoGeneratorBuilder;
import org.bouncycastle.operator.ContentSigner;
41 import org.bouncycastle.operator.OperatorCreationException;
import org.bouncycastle.operator.bc.BcDigestCalculatorProvider;
43 import org.bouncycastle.operator.jcajce.JcaContentSignerBuilder;
45
public class MyCMSSignedDataGenerator
47     extends CMSSignedGenerator
{
49     private List signerInfs = new ArrayList();
51     private class SignerInf
{
53         final PrivateKey          key;
54         final Object              signerIdentifier;
55         final String              digestOID;
56         final String              encOID;
57         final CMSAttributeTableGenerator sAttr;
58         final CMSAttributeTableGenerator unsAttr;
59         final AttributeTable      baseSignedTable;
61     SignerInf(
        PrivateKey          key,
63         Object              signerIdentifier,
        String              digestOID,
65         String              encOID,
        CMSAttributeTableGenerator sAttr,
67         CMSAttributeTableGenerator unsAttr,
        AttributeTable      baseSignedTable)
69     {
71         this.key = key;
        this.signerIdentifier = signerIdentifier;

```



```

73     this.digestOID = digestOID;
74     this.encOID = encOID;
75     this.sAttr = sAttr;
76     this.unsAttr = unsAttr;
77     this.baseSignedTable = baseSignedTable;
78 }
79 SignerInfoGenerator toSignerInfoGenerator(
80     SecureRandom random,
81     Provider sigProvider,
82     boolean addDefaultAttributes)
83     throws IOException, CertificateEncodingException,
84     CMSException, OperatorCreationException,
85     NoSuchAlgorithmException
86 {
87     String digestName = MyCMSSignedHelper.
88         INSTANCE.getDigestAlgName(digestOID);
89     String signatureName = digestName + "with"
90     + MyCMSSignedHelper.INSTANCE.getEncryptionAlgName(
91     encOID);
92     JcaSignerInfoGeneratorBuilder builder = new
93     JcaSignerInfoGeneratorBuilder(new
94     BcDigestCalculatorProvider());
95     if (addDefaultAttributes)
96     {
97         builder.setSignedAttributeGenerator(sAttr);
98     }
99     builder.setDirectSignature(!addDefaultAttributes);
100    builder.setUnsignedAttributeGenerator(unsAttr);
101    JcaContentSignerBuilder signerBuilder;
102    try
103    {
104        signerBuilder = new JcaContentSignerBuilder(
105        signatureName).setSecureRandom(random);
106    }
107    catch (IllegalArgumentException e)
108    {
109        throw new NoSuchAlgorithmException(e.getMessage());
110    }
111    if (sigProvider != null)
112    {
113        signerBuilder.setProvider(sigProvider);
114    }
115    ContentSigner contentSigner = signerBuilder.build(key);
116    if (signerIdentifier instanceof X509Certificate)
117    {
118        return builder.build(contentSigner, (X509Certificate)
119        signerIdentifier);
120    }
121    else
122    {
123        return builder.build(contentSigner, (byte[])
124        signerIdentifier);
125    }
126 }
127 /**
128  * base constructor
129  */
130 public MyCMSSignedDataGenerator()
131 {

```

```

131
132  /**
133   * constructor allowing specific source of randomness
134   * @param rand instance of SecureRandom to use
135   */
136 public MyCMSSignedDataGenerator(
137     SecureRandom rand)
138 {
139     super(rand);
140 }
141
142 /**
143  * add a signer – no attributes other than the default ones will
144  * be
145  * provided here.
146  * @param key signing key to use
147  * @param cert certificate containing corresponding public key
148  * @param digestOID digest algorithm OID
149  * @deprecated use addSignerInfoGenerator
150  */
151 public void addSigner(
152     PrivateKey key,
153     X509Certificate cert,
154     String digestOID)
155     throws IllegalArgumentException
156 {
157     addSigner(key, cert, getEncOID(key, digestOID), digestOID);
158 }
159
160 /**
161  * add a signer, specifying the digest encryption algorithm to
162  * use – no attributes other than the default ones will be
163  * provided here.
164  * @param key signing key to use
165  * @param cert certificate containing corresponding public key
166  * @param encryptionOID digest encryption algorithm OID
167  * @param digestOID digest algorithm OID
168  * @deprecated use addSignerInfoGenerator
169  */
170 public void addSigner(
171     PrivateKey key,
172     X509Certificate cert,
173     String encryptionOID,
174     String digestOID)
175     throws IllegalArgumentException
176 {
177     doAddSigner(key, cert, encryptionOID, digestOID,
178         new MySignedAttributeTableGenerator(), null, null);
179 }
180
181 /**
182  * add a signer – no attributes other than the default ones will
183  * be
184  * provided here.
185  * @deprecated use addSignerInfoGenerator
186  */
187 public void addSigner(
188     PrivateKey key,
189     byte [] subjectKeyID,
190     String digestOID)
191     throws IllegalArgumentException
192 {
193     addSigner(key, subjectKeyID, getEncOID(key, digestOID),
194         digestOID);

```

```

195  /**
    * add a signer, specifying the digest encryption algorithm to
    * use – no attributes other than the default ones will be
197  * provided here.
    * @deprecated use addSignerInfoGenerator
199  */
    public void addSigner(
201      PrivateKey    key,
        byte[]      subjectKeyID,
203      String        encryptionOID,
        String        digestOID)
205      throws IllegalArgumentException
    {
207        doAddSigner(key, subjectKeyID, encryptionOID, digestOID,
            new MySignedAttributeTableGenerator(), null, null);
209    }

211  /**
    * add a signer with extra signed/unsigned attributes.
213  *
    *
215  * @param key signing key to use
    * @param cert certificate containing corresponding public key
217  * @param digestOID digest algorithm OID
    * @param signedAttr table of attributes to be included in
    * signature
219  * @param unsignedAttr table of attributes to be included as
    * unsigned
    * @deprecated use addSignerInfoGenerator
221  */
    public void addSignerPAdES(
223      PrivateKey    key,
        X509Certificate cert,
225      String        digestOID,
        AttributeTable signedAttr,
227      AttributeTable unsignedAttr)
        throws IllegalArgumentException
    {
229      addSignerPAdES(key, cert, getEncOID(key, digestOID),
        digestOID, signedAttr, unsignedAttr);
231    }

233  /**
    * add a signer with extra signed/unsigned attributes.
235  *
    *
237  * @param key signing key to use
    * @param cert certificate containing corresponding public key
239  * @param digestOID digest algorithm OID
    * @param signedAttr table of attributes to be included in
    * signature
    * @param unsignedAttr table of attributes to be included as
    * unsigned
241  * @deprecated use addSignerInfoGenerator
    */
243  public void addSigner(
        PrivateKey    key,
245      X509Certificate cert,
        String        digestOID,
247      AttributeTable signedAttr,
        AttributeTable unsignedAttr)
249      throws IllegalArgumentException
    {
251      addSigner(key, cert, getEncOID(key, digestOID), digestOID,
        signedAttr, unsignedAttr);
    }
253
    /**

```

```

255 * add a signer , specifying the digest encryption algorithm ,
      * with extra signed/unsigned attributes .
257 *
      * @param key signing key to use
      * @param cert certificate containing corresponding public key
259 * @param encryptionOID digest encryption algorithm OID
      * @param digestOID digest algorithm OID
261 * @param signedAttr table of attributes to be included in
      * signature
      * @param unsignedAttr table of attributes to be included as
263 * unsigned
      * @deprecated use addSignerInfoGenerator
      */
265 public void addSignerPAdES(
267     PrivateKey key ,
      X509Certificate cert ,
269     String encryptionOID ,
      String digestOID ,
271     AttributeTable signedAttr ,
      AttributeTable unsignedAttr)
      throws IllegalArgumentException
273 {
      doAddSigner(key , cert , encryptionOID , digestOID ,
275     new MySignedAttributeTableGenerator( signedAttr ) ,
      new SimpleAttributeTableGenerator( unsignedAttr ) ,
277     signedAttr);
      }
279 /**
      * add a signer , specifying the digest encryption algorithm ,
      * with extra signed/unsigned attributes .
281 *
      * @param key signing key to use
      * @param cert certificate containing corresponding public key
283 * @param encryptionOID digest encryption algorithm OID
      * @param digestOID digest algorithm OID
285 * @param signedAttr table of attributes to be included in
      * signature
287 * @param unsignedAttr table of attributes to be included as
      * unsigned
      * @deprecated use addSignerInfoGenerator
289 *
      */
291 public void addSigner(
      PrivateKey key ,
293     X509Certificate cert ,
      String encryptionOID ,
295     String digestOID ,
      AttributeTable signedAttr ,
297     AttributeTable unsignedAttr)
      throws IllegalArgumentException
299 {
      doAddSigner(key , cert , encryptionOID , digestOID ,
301     new MySignedAttributeTableGenerator( signedAttr ) ,
      new SimpleAttributeTableGenerator( unsignedAttr ) ,
303     signedAttr);
      }
305 /**
      * add a signer with extra signed/unsigned attributes .
307 *
      * @param key signing key to use
      * @param subjectKeyID subjectKeyID of corresponding public key
309 * @param digestOID digest algorithm OID
      * @param signedAttr table of attributes to be included in
      * signature
311 * @param unsignedAttr table of attributes to be included as
      * unsigned
      * @deprecated use addSignerInfoGenerator

```

```

313     */
314     public void addSigner(
315         PrivateKey    key,
316         byte[]        subjectKeyID,
317         String        digestOID,
318         AttributeTable signedAttr,
319         AttributeTable unsignedAttr)
320         throws IllegalArgumentException
321     {
322         addSigner(key, subjectKeyID, getEncOID(key, digestOID),
323             digestOID, signedAttr,
324             unsignedAttr);
325     }
326
327     /**
328      * add a signer, specifying the digest encryption algorithm,
329      * with extra signed/unsigned attributes.
330      *
331      * @param key signing key to use
332      * @param subjectKeyID subjectKeyID of corresponding public key
333      * @param encryptionOID digest encryption algorithm OID
334      * @param digestOID digest algorithm OID
335      * @param signedAttr table of attributes to be included in
336      * signature
337      * @param unsignedAttr table of attributes to be included as
338      * unsigned
339      * @deprecated use addSignerInfoGenerator
340      */
341     public void addSigner(
342         PrivateKey    key,
343         byte[]        subjectKeyID,
344         String        encryptionOID,
345         String        digestOID,
346         AttributeTable signedAttr,
347         AttributeTable unsignedAttr)
348         throws IllegalArgumentException
349     {
350         doAddSigner(key, subjectKeyID, encryptionOID, digestOID,
351             new MySignedAttributeTableGenerator(signedAttr),
352             new SimpleAttributeTableGenerator(unsignedAttr),
353             signedAttr);
354     }
355
356     /**
357      * add a signer with extra signed/unsigned attributes based on
358      * generators.
359      * @deprecated use addSignerInfoGenerator
360      */
361     public void addSigner(
362         PrivateKey    key,
363         X509Certificate cert,
364         String        digestOID,
365         CMSAttributeTableGenerator signedAttrGen,
366         CMSAttributeTableGenerator unsignedAttrGen)
367         throws IllegalArgumentException
368     {
369         addSigner(key, cert, getEncOID(key, digestOID), digestOID,
370             signedAttrGen, unsignedAttrGen);
371     }
372
373     /**
374      * add a signer, specifying the digest encryption algorithm,
375      * with extra signed/unsigned attributes based on generators.
376      * @deprecated use addSignerInfoGenerator
377      */
378     public void addSigner(
379         PrivateKey    key,
380         X509Certificate cert,

```

```

373     String                encryptionOID ,
374     String                digestOID ,
375     CMSAttributeTableGenerator signedAttrGen ,
376     CMSAttributeTableGenerator unsignedAttrGen)
377     throws IllegalArgumentException
378 {
379     doAddSigner(key, cert, encryptionOID, digestOID,
380                signedAttrGen,
381                unsignedAttrGen, null);
382 }
383 /**
384  * add a signer with extra signed/unsigned attributes based on
385  * generators.
386  * @deprecated use addSignerInfoGenerator
387  */
388 public void addSigner(
389     PrivateKey                key,
390     byte []                   subjectKeyID ,
391     String                    digestOID ,
392     CMSAttributeTableGenerator signedAttrGen ,
393     CMSAttributeTableGenerator unsignedAttrGen)
394     throws IllegalArgumentException
395 {
396     addSigner(key, subjectKeyID, getEncOID(key, digestOID),
397              digestOID, signedAttrGen,
398              unsignedAttrGen);
399 }
400 /**
401  * add a signer, including digest encryption algorithm, with
402  * extra signed/unsigned attributes based on generators.
403  * @deprecated use addSignerInfoGenerator
404  */
405 public void addSigner(
406     PrivateKey                key,
407     byte []                   subjectKeyID ,
408     String                    encryptionOID ,
409     String                    digestOID ,
410     CMSAttributeTableGenerator signedAttrGen ,
411     CMSAttributeTableGenerator unsignedAttrGen)
412     throws IllegalArgumentException
413 {
414     doAddSigner(key, subjectKeyID, encryptionOID, digestOID,
415                 signedAttrGen, unsignedAttrGen, null);
416 }
417 private void doAddSigner(
418     PrivateKey                key,
419     Object                    signerIdentifier ,
420     String                    encryptionOID ,
421     String                    digestOID ,
422     CMSAttributeTableGenerator signedAttrGen ,
423     CMSAttributeTableGenerator unsignedAttrGen ,
424     AttributeTable            baseSignedTable)
425     throws IllegalArgumentException
426 {
427     signerInfos.add(new SignerInf(key, signerIdentifier,
428                                   digestOID, encryptionOID,
429                                   signedAttrGen, unsignedAttrGen, baseSignedTable));
430 }
431 /**
432  * generate a signed object that for a CMS Signed Data
433  * object using the given provider.
434  * @deprecated use generate() method not taking provider.
435  */
436 public CMSSignedData generate(

```

```

437     CMSProcessable content,
        String sigProvider)
        throws NoSuchAlgorithmException, NoSuchProviderException,
            CMSEException
439     {
        return generate(content, MyCMSUtils.getProvider(sigProvider)
441     );
    }
443     /**
444     * generate a signed object that for a CMS Signed Data
445     * object using the given provider.
446     * @deprecated use generate() method not taking provider.
447     */
448     public CMSSignedData generate(
449         CMSProcessable content,
450         Provider sigProvider)
451         throws NoSuchAlgorithmException, CMSEException
452     {
453         return generate(content, false, sigProvider);
454     }
455     /**
456     * generate a signed object that for a CMS Signed Data
457     * object using the given provider – if encapsulate is true a
458     * copy
459     * of the message will be included in the signature. The content
460     * type
461     * is set according to the OID represented by the string
462     * signedContentType.
463     * @deprecated use generate(CMSTypedData, boolean)
464     */
465     public CMSSignedData generate(
466         String eContentType,
467         CMSProcessable content,
468         boolean encapsulate,
469         String sigProvider)
470         throws NoSuchAlgorithmException, NoSuchProviderException,
471             CMSEException
472     {
473         return generate(eContentType, content, encapsulate,
474             MyCMSUtils.getProvider(sigProvider),
475             true);
476     }
477     /**
478     * generate a signed object that for a CMS Signed Data
479     * object using the given provider – if encapsulate is true a
480     * copy
481     * of the message will be included in the signature. The content
482     * type
483     * is set according to the OID represented by the string
484     * signedContentType.
485     * @deprecated use generate(CMSTypedData, boolean)
486     */
487     public CMSSignedData generate(
488         String eContentType,
489         CMSProcessable content,
490         boolean encapsulate,
491         Provider sigProvider)
492         throws NoSuchAlgorithmException, CMSEException
493     {
494         return generate(eContentType, content, encapsulate,
495             sigProvider, true);
496     }
497     /**
498     * Similar method to the other generate methods. The additional
499     * argument

```

```

493     * addDefaultAttributes indicates whether or not a default set
      * of signed attributes
      * need to be added automatically. If the argument is set to
      * false, no
495     * attributes will get added at all.
      * @deprecated use generate(CMSTypedData, boolean)
497     */
public CMSSignedData generate(
499     String eContentType,
      CMSProcessable content,
501     boolean encapsulate,
      String sigProvider,
503     boolean addDefaultAttributes)
      throws NoSuchAlgorithmException, NoSuchProviderException,
          CMSException
505     {
      return generate(eContentType, content, encapsulate,
507         MyCMSUtils.getProvider(sigProvider),
          addDefaultAttributes);
509     }
    /**
511     * Similar method to the other generate methods. The additional
      * argument
      * addDefaultAttributes indicates whether or not a default set
      * of signed attributes
513     * need to be added automatically. If the argument is set to
      * false, no
      * attributes will get added at all.
515     */
public CMSSignedData generate(
517     String eContentType,
      final CMSProcessable content,
519     boolean encapsulate,
      Provider sigProvider,
521     boolean addDefaultAttributes)
      throws NoSuchAlgorithmException, CMSException
523     {
      boolean isCounterSignature = (eContentType == null);
525
      final ASN1ObjectIdentifier contentTypeOID =
          isCounterSignature
527             ? null
              : new ASN1ObjectIdentifier(eContentType);
529
      for (Iterator it = signerInfs.iterator(); it.hasNext(); )
531     {
          SignerInf signer = (SignerInf) it.next();
533
          try
535             {
                  signerGens.add(signer.toSignerInfoGenerator(rand,
537                     sigProvider,
                      addDefaultAttributes));
539             }
          catch (OperatorCreationException e)
541             {
                  throw new CMSException("exception creating signerInf
                      ", e);
543             }
          catch (IOException e)
545             {
                  throw new CMSException("exception encoding
                      attributes", e);
547             }
          catch (CertificateEncodingException e)
549             {
                  throw new CMSException("error creating sid.", e);

```



```

551     }
553     signerInfs.clear();
555     if (content != null)
557     {
559         return generate(new CMSTypedData()
561             {
563                 public ASN1ObjectIdentifier getContentType()
565                 {
567                     return contentTypeOID;
569                 }
571                 public void write(OutputStream out)
573                 throws IOException, CMSEException
575                 {
577                     content.write(out);
579                 }
581                 public Object getContent()
583                 {
585                     return content.getContent();
587                 }
589             }, encapsulate);
591     }
593     else
595     {
597         return generate(new CMSAbsentContent(contentTypeOID),
599             encapsulate);
601     }
603 }
605 /**
607  * generate a signed object that for a CMS Signed Data
609  * object using the given provider – if encapsulate is true a
611  * copy
613  * of the message will be included in the signature with the
615  * default content type "data".
617  * @deprecated use generate(CMSTypedData, boolean)
619  */
621 public CMSSignedData generate(
623     CMSProcessable content,
625     boolean encapsulate,
627     String sigProvider)
629     throws NoSuchAlgorithmException, NoSuchProviderException,
631     CMSEException
633 {
635     if (content instanceof CMSTypedData)
637     {
639         return this.generate(((CMSTypedData) content).
641             getContentType().getId(), content, encapsulate,
643             sigProvider);
645     }
647     else
649     {
651         return this.generate(DATA, content, encapsulate,
653             sigProvider);
655     }
657 }
659 /**
661  * generate a signed object that for a CMS Signed Data
663  * object using the given provider – if encapsulate is true a
665  * copy
667  * of the message will be included in the signature with the
669  * default content type "data".
671  * @deprecated use generate(CMSTypedData, boolean)
673  */

```

```

613 public CMSSignedData generate(
        CMSProcessable content,
615     boolean encapsulate,
        Provider sigProvider)
        throws NoSuchAlgorithmException, CMSEException
617     {
        if (content instanceof CMSTypedData)
619         {
            return this.generate(((CMSTypedData)content).
                getContentType().getId(), content, encapsulate,
                sigProvider);
621         }
        else
623         {
            return this.generate(DATA, content, encapsulate,
                sigProvider);
625         }
        }
627
        public CMSSignedData generate(
629     CMSTypedData content)
        throws CMSEException
631     {
        return generate(content, false);
633     }
635
        public CMSSignedData generate(
        // FIXME Avoid accessing more than once to support
        // CMSProcessableInputStream
637     CMSTypedData content,
        boolean encapsulate)
639     throws CMSEException
        {
        if (!signerInfs.isEmpty())
641         {
            throw new IllegalStateException("this method can only be
643                 used with SignerInfoGenerator");
        }
645
        // TODO
647 //     if (signerInfs.isEmpty())
        //     {
649 //         /* RFC 3852 5.2
        //         * "In the degenerate case where there are no signers,
        //         the
651 //         * EncapsulatedContentInfo value being "signed" is
        //         irrelevant. In this
        //         * case, the content type within the
        //         EncapsulatedContentInfo value being
653 //         * "signed" MUST be id-data (as defined in section 4),
        //         and the content
        //         * field of the EncapsulatedContentInfo value MUST be
        //         omitted."
655 //         */
        //         if (encapsulate)
657 //         {
            throw new IllegalArgumentException("no signers,
        encapsulate must be false");
659 //         }
        //         if (!DATA.equals(eContentType))
661 //         {
            throw new IllegalArgumentException("no signers,
        eContentType must be id-data");
663 //         }
        //     }
665 //
        //     if (!DATA.equals(eContentType))
667 //     {

```

```

669 //          /* RFC 3852 5.3
//          * [The 'signedAttrs' ]...
//          * field is optional , but it MUST be present if the
content type of
671 //          * the EncapsulatedContentInfo value being signed is
not id-data.
//          */
673 //          // TODO signedAttrs must be present for all signers
//          }
675
ASN1EncodableVector  digestAlgs = new ASN1EncodableVector();
677 ASN1EncodableVector  signerInfos = new ASN1EncodableVector()
;
679
digests.clear(); // clear the current preserved digest
state
681
//
// add the precalculated SignerInfo objects.
683 //
684 for (Iterator it = _signers.iterator(); it.hasNext(); )
685 {
686     SignerInformation signer = (SignerInformation)it.next();
687     digestAlgs.add(MyCMSSignedHelper.INSTANCE.fixAlgID(
signer.getDigestAlgorithmID()));
689
// TODO Verify the content type and calculated digest
match the precalculated SignerInfo
signerInfos.add(signer.toASN1Structure());
691 }
693
//
// add the SignerInfo objects
695 //
ASN1ObjectIdentifier contentTypeOID = content.getContentType
();
697
ASN1OctetString octs = null;
699
if (content != null)
701 {
702     ByteArrayOutputStream bOut = null;
703
704     if (encapsulate)
705     {
706         bOut = new ByteArrayOutputStream();
707     }
709
710     OutputStream cOut = MyCMSUtils.
attachSignersToOutputStream(signerGens , bOut);
711
// Just in case it's unencapsulated and there are no
signers!
cOut = MyCMSUtils.getSafeOutputStream(cOut);
713
714     try
715     {
716         content.write(cOut);
717
718         cOut.close();
719     }
720     catch (IOException e)
721     {
722         throw new CMSEException("data processing exception: "
+ e.getMessage(), e);
723     }
725
726     if (encapsulate)
727     {

```

```

727         octs = new BEROctetString(bOut.toByteArray());
729     }
731     for (Iterator it = signerGens.iterator(); it.hasNext();)
732     {
733         SignerInfoGenerator sGen = (SignerInfoGenerator)it.next
734         ();
735         SignerInfo inf = sGen.generate(contentTypeOID);
736         digestAlgs.add(inf.getDigestAlgorithm());
737         signerInfos.add(inf);
738         byte[] calcDigest = sGen.getCalculatedDigest();
739         if (calcDigest != null)
740         {
741             digests.put(inf.getDigestAlgorithm().getAlgorithm().
742             getId(), calcDigest);
743         }
744     }
745     ASN1Set certificates = null;
746     if (certs.size() != 0)
747     {
748         certificates = MyCMSUtils.createBerSetFromList(certs);
749     }
750     ASN1Set certrevlist = null;
751     if (crls.size() != 0)
752     {
753         certrevlist = MyCMSUtils.createBerSetFromList(crls);
754     }
755     ContentInfo encInfo = new ContentInfo(contentTypeOID, octs);
756     SignedData sd = new SignedData(
757         new DERSet(digestAlgs),
758         encInfo,
759         certificates,
760         certrevlist,
761         new DERSet(signerInfos));
762     ContentInfo contentInfo = new ContentInfo(
763         CMSObjectIdentifiers.signedData, sd);
764     return new CMSSignedData(content, contentInfo);
765 }
766
767 /**
768  * generate a set of one or more SignerInformation objects
769  * representing counter signatures on
770  * the passed in SignerInformation object.
771  *
772  * @param signer the signer to be countersigned
773  * @param sigProvider the provider to be used for counter
774  * signing.
775  * @return a store containing the signers.
776  * @deprecated use generateCounterSigners(SignerInformation)
777  */
778 public SignerInformationStore generateCounterSigners(
779     SignerInformation signer, Provider sigProvider)
780     throws NoSuchAlgorithmException, CMSException
781 {
782     return this.generate(null, new CMSProcessableByteArray(
783         signer.getSignature()), false, sigProvider).
784         getSignerInfos();
785 }

```

```

789     }
791     /**
793     * generate a set of one or more SignerInformation objects
795     * representing counter signatures on
797     * the passed in SignerInformation object.
799     * @param signer the signer to be countersigned
801     * @param sigProvider the provider to be used for counter
803     * signing.
805     * @return a store containing the signers.
807     * @deprecated use generateCounterSigners(SignerInformation)
809     */
811     public SignerInformationStore generateCounterSigners(
813     SignerInformation signer, String sigProvider)
815     throws NoSuchAlgorithmException, NoSuchProviderException,
817     CMSEException
    {
        return this.generate(null, new CMSProcessableByteArray(
            signer.getSignature()), false, MyCMSUtils.getProvider(
                sigProvider)).getSignerInfos();
    }
    /**
    * generate a set of one or more SignerInformation objects
    * representing counter signatures on
    * the passed in SignerInformation object.
    * @param signer the signer to be countersigned
    * @return a store containing the signers.
    */
    public SignerInformationStore generateCounterSigners(
        SignerInformation signer)
        throws CMSEException
    {
        return this.generate(new CMSProcessableByteArray(null,
            signer.getSignature()), false).getSignerInfos();
    }
}

```

implementacao/ProjetoPADES_ICPBrasil/src/util/MyCMSSignedDataGenerator.java

```

1 package util;
2 import java.io.ByteArrayInputStream;
3 import java.io.IOException;
4 import java.security.InvalidAlgorithmParameterException;
5 import java.security.NoSuchAlgorithmException;
6 import java.security.Provider;
7 import java.security.cert.CRLException;
8 import java.security.cert.CertStore;
9 import java.security.cert.CertificateException;
10 import java.security.cert.CertificateFactory;
11 import java.security.cert.CollectionCertStoreParameters;
12 import java.util.ArrayList;
13 import java.util.Enumeration;
14 import java.util.HashMap;
15 import java.util.List;
16 import java.util.Map;
17
18 import org.bouncycastle.asn1.ASN1Encodable;
19 import org.bouncycastle.asn1.ASN1Primitive;
20 import org.bouncycastle.asn1.ASN1Sequence;
21 import org.bouncycastle.asn1.ASN1Set;
22 import org.bouncycastle.asn1.ASN1TaggedObject;
23 import org.bouncycastle.asn1.DERNull;
24 import org.bouncycastle.asn1.DERObjectIdentifier;
25 import org.bouncycastle.asn1.cryptopro.CryptoProObjectIdentifiers;
26 import org.bouncycastle.asn1.eac.EACObjectIdentifiers;

```

```

27 import org.bouncycastle.asn1.nist.NISTObjectIdentifiers;
import org.bouncycastle.asn1.oiw.OIWObjectIdentifiers;
29 import org.bouncycastle.asn1.pkcs.PKCSObjectIdentifiers;
import org.bouncycastle.asn1.teletrust.TeleTrustObjectIdentifiers;
31 import org.bouncycastle.asn1.x509.AlgorithmIdentifier;
import org.bouncycastle.asn1.x509.X509ObjectIdentifiers;
33 import org.bouncycastle.asn1.x9.X9ObjectIdentifiers;
import org.bouncycastle.cms.CMSException;
import org.bouncycastle.cms.CMSSignedDataGenerator;
35 import org.bouncycastle.x509.NoSuchStoreException;
import org.bouncycastle.x509.X509CollectionStoreParameters;
37 import org.bouncycastle.x509.X509Store;
import org.bouncycastle.x509.X509V2AttributeCertificate;
39
41 class MyCMSSignedHelper
{
43     static final MyCMSSignedHelper INSTANCE = new MyCMSSignedHelper
        ();

45     private static final Map      encryptionAlgs = new HashMap();
private static final Map      digestAlgs = new HashMap();
47     private static final Map      digestAliases = new HashMap();

49     private static void addEntries(DERObjectIdentifier alias, String
        digest, String encryption)
    {
51         digestAlgs.put(alias.getId(), digest);
        encryptionAlgs.put(alias.getId(), encryption);
53     }

55     static
    {
57         addEntries(NISTObjectIdentifiers.dsa_with_sha224, "SHA224",
            "DSA");
        addEntries(NISTObjectIdentifiers.dsa_with_sha256, "SHA256",
            "DSA");
59         addEntries(NISTObjectIdentifiers.dsa_with_sha384, "SHA384",
            "DSA");
        addEntries(NISTObjectIdentifiers.dsa_with_sha512, "SHA512",
            "DSA");
61         addEntries(OIWObjectIdentifiers.dsaWithSHA1, "SHA1", "RSA");
        addEntries(OIWObjectIdentifiers.md4WithRSA, "MD4", "RSA");
63         addEntries(OIWObjectIdentifiers.md4WithRSAEncryption, "MD4",
            "RSA");
        addEntries(OIWObjectIdentifiers.md5WithRSA, "MD5", "RSA");
65         addEntries(OIWObjectIdentifiers.sha1WithRSA, "SHA1", "RSA");
        addEntries(PKCSObjectIdentifiers.md2WithRSAEncryption, "MD2",
            "RSA");
67         addEntries(PKCSObjectIdentifiers.md4WithRSAEncryption, "MD4",
            "RSA");
        addEntries(PKCSObjectIdentifiers.md5WithRSAEncryption, "MD5",
            "RSA");
69         addEntries(PKCSObjectIdentifiers.sha1WithRSAEncryption, "
            SHA1", "RSA");
        addEntries(PKCSObjectIdentifiers.sha224WithRSAEncryption, "
            SHA224", "RSA");
71         addEntries(PKCSObjectIdentifiers.sha256WithRSAEncryption, "
            SHA256", "RSA");
        addEntries(PKCSObjectIdentifiers.sha384WithRSAEncryption, "
            SHA384", "RSA");
73         addEntries(PKCSObjectIdentifiers.sha512WithRSAEncryption, "
            SHA512", "RSA");
        addEntries(X9ObjectIdentifiers.ecdsa_with_SHA1, "SHA1", "
            ECDSA");
75         addEntries(X9ObjectIdentifiers.ecdsa_with_SHA224, "SHA224",
            "ECDSA");
        addEntries(X9ObjectIdentifiers.ecdsa_with_SHA256, "SHA256",
            "ECDSA");

```

```

77 addEntries(X9ObjectIdentifiers.ecdsa_with_SHA384, "SHA384",
    "ECDSA");
addEntries(X9ObjectIdentifiers.ecdsa_with_SHA512, "SHA512",
79 "ECDSA");
addEntries(X9ObjectIdentifiers.id_dsa_with_sha1, "SHA1", "
    DSA");
addEntries(EACObjectIdentifiers.id_TA_ECDSA_SHA_1, "SHA1", "
    ECDSA");
81 addEntries(EACObjectIdentifiers.id_TA_ECDSA_SHA_224, "SHA224
    ", "ECDSA");
addEntries(EACObjectIdentifiers.id_TA_ECDSA_SHA_256, "SHA256
    ", "ECDSA");
83 addEntries(EACObjectIdentifiers.id_TA_ECDSA_SHA_384, "SHA384
    ", "ECDSA");
addEntries(EACObjectIdentifiers.id_TA_ECDSA_SHA_512, "SHA512
    ", "ECDSA");
85 addEntries(EACObjectIdentifiers.id_TA_RSA_v1_5_SHA_1, "SHA1"
    , "RSA");
addEntries(EACObjectIdentifiers.id_TA_RSA_v1_5_SHA_256, "
    SHA256", "RSA");
87 addEntries(EACObjectIdentifiers.id_TA_RSA_PSS_SHA_1, "SHA1",
    "RSAandMGF1");
addEntries(EACObjectIdentifiers.id_TA_RSA_PSS_SHA_256, "
    SHA256", "RSAandMGF1");
89
encryptionAlgs.put(X9ObjectIdentifiers.id_dsa.getId(), "DSA"
    );
91 encryptionAlgs.put(PKCSObjectIdentifiers.rsaEncryption.getId
    (), "RSA");
encryptionAlgs.put(TeleTrustObjectIdentifiers.
    teleTrustRSASignatureAlgorithm, "RSA");
93 encryptionAlgs.put(X509ObjectIdentifiers.id_ea_rsa.getId(),
    "RSA");
encryptionAlgs.put(CMSSignedDataGenerator.ENCRYPTION_RSA_PSS
    , "RSAandMGF1");
95 encryptionAlgs.put(CryptoProObjectIdentifiers.gostR3410_94.
    getId(), "GOST3410");
encryptionAlgs.put(CryptoProObjectIdentifiers.gostR3410_2001
    .getId(), "ECGOST3410");
97 encryptionAlgs.put("1.3.6.1.4.1.5849.1.6.2", "ECGOST3410");
encryptionAlgs.put("1.3.6.1.4.1.5849.1.1.5", "GOST3410");
99 encryptionAlgs.put(CryptoProObjectIdentifiers.
    gostR3411_94_with_gostR3410_2001.getId(), "ECGOST3410");
encryptionAlgs.put(CryptoProObjectIdentifiers.
    gostR3411_94_with_gostR3410_94.getId(), "GOST3410");
101
digestAlgs.put(PKCSObjectIdentifiers.md2.getId(), "MD2");
103 digestAlgs.put(PKCSObjectIdentifiers.md4.getId(), "MD4");
digestAlgs.put(PKCSObjectIdentifiers.md5.getId(), "MD5");
105 digestAlgs.put(OIWObjectIdentifiers.idSHA1.getId(), "SHA1");
digestAlgs.put(NISTObjectIdentifiers.id_sha224.getId(), "
    SHA224");
107 digestAlgs.put(NISTObjectIdentifiers.id_sha256.getId(), "
    SHA256");
digestAlgs.put(NISTObjectIdentifiers.id_sha384.getId(), "
    SHA384");
109 digestAlgs.put(NISTObjectIdentifiers.id_sha512.getId(), "
    SHA512");
digestAlgs.put(TeleTrustObjectIdentifiers.ripemd128.getId(),
    "RIPEMD128");
111 digestAlgs.put(TeleTrustObjectIdentifiers.ripemd160.getId(),
    "RIPEMD160");
digestAlgs.put(TeleTrustObjectIdentifiers.ripemd256.getId(),
    "RIPEMD256");
113 digestAlgs.put(CryptoProObjectIdentifiers.gostR3411.getId(),
    "GOST3411");
digestAlgs.put("1.3.6.1.4.1.5849.1.2.1", "GOST3411");
115

```

```

117     digestAliases.put("SHA1", new String[] { "SHA-1" });
118     digestAliases.put("SHA224", new String[] { "SHA-224" });
119     digestAliases.put("SHA256", new String[] { "SHA-256" });
120     digestAliases.put("SHA384", new String[] { "SHA-384" });
121     digestAliases.put("SHA512", new String[] { "SHA-512" });
122 }
123 /**
124  * Return the digest algorithm using one of the standard JCA
125  * string representations rather than the algorithm identifier (if
126  * possible).
127  */
128 String getDigestAlgName(
129     String digestAlgOID)
130 {
131     String algName = (String)digestAlgs.get(digestAlgOID);
132     if (algName != null)
133     {
134         return algName;
135     }
136     return digestAlgOID;
137 }
138 /**
139  * Return the digest encryption algorithm using one of the
140  * standard
141  * JCA string representations rather the the algorithm
142  * identifier (if
143  * possible).
144  */
145 String getEncryptionAlgName(
146     String encryptionAlgOID)
147 {
148     String algName = (String)encryptionAlgs.get(encryptionAlgOID
149 );
150     if (algName != null)
151     {
152         return algName;
153     }
154     return encryptionAlgOID;
155 }
156
157
158 X509Store createAttributeStore(
159     String type,
160     Provider provider,
161     ASN1Set certSet)
162     throws NoSuchStoreException, CMSEException
163 {
164     List certs = new ArrayList();
165     if (certSet != null)
166     {
167         Enumeration e = certSet.getObjects();
168         while (e.hasMoreElements())
169         {
170             try
171             {
172                 ASN1Primitive obj = ((ASN1Encodable)e.
173                     nextElement()).toASN1Primitive();
174                 if (obj instanceof ASN1TaggedObject)
175                 {

```



```

179         ASN1TaggedObject tagged = (ASN1TaggedObject)
                obj;
181         if (tagged.getTagNo() == 2)
183             {
                certs.add(new X509V2AttributeCertificate
                    (ASN1Sequence.getInstance(tagged,
                        false).getEncoded()));
185             }
            }
187         catch (IOException ex)
189             {
                throw new CMSEException(
                    "can't re-encode attribute certificate!"
                    , ex);
191             }
        }
193    }
195    try
197    {
        return X509Store.getInstance(
            "AttributeCertificate/" +type, new
            X509CollectionStoreParameters(certs),
            provider);
199    }
201    catch (IllegalArgumentException e)
203    {
        throw new CMSEException("can't setup the X509Store", e);
205    }
}

X509Store createCertificateStore(
207    String type,
    Provider provider,
209    ASN1Set certSet)
    throws NoSuchStoreException, CMSEException
211    {
    List certs = new ArrayList();
213
    if (certSet != null)
215        {
            addCertsFromSet(certs, certSet, provider);
217        }
219    try
221    {
        return X509Store.getInstance(
            "Certificate/" +type, new
            X509CollectionStoreParameters(certs),
            provider);
223    }
225    catch (IllegalArgumentException e)
227    {
        throw new CMSEException("can't setup the X509Store", e);
229    }
}

X509Store createCRLsStore(
231    String type,
    Provider provider,
233    ASN1Set crlSet)
    throws NoSuchStoreException, CMSEException
235    {
    List crls = new ArrayList();
237
    if (crlSet != null)
239    {

```

```

241     addCRLsFromSet(crls , crlSet , provider);
242 }
243 try
244 {
245     return X509Store.getInstance(
246         "CRL/" +type, new
247         X509CollectionStoreParameters(crls),
248         provider);
249 }
250 catch (IllegalArgumentException e)
251 {
252     throw new CMSEException("can't setup the X509Store", e);
253 }
254 }
255 CertStore createCertStore(
256     String type,
257     Provider provider,
258     ASN1Set certSet,
259     ASN1Set crlSet)
260     throws CMSEException, NoSuchAlgorithmException
261 {
262     List certsAndcrls = new ArrayList();
263     //
264     // load the certificates and revocation lists if we have any
265     //
266     if (certSet != null)
267     {
268         addCertsFromSet(certsAndcrls , certSet , provider);
269     }
270     if (crlSet != null)
271     {
272         addCRLsFromSet(certsAndcrls , crlSet , provider);
273     }
274     try
275     {
276         if (provider != null)
277         {
278             return CertStore.getInstance(type, new
279                 CollectionCertStoreParameters(certsAndcrls),
280                 provider);
281         }
282         else
283         {
284             return CertStore.getInstance(type, new
285                 CollectionCertStoreParameters(certsAndcrls));
286         }
287     }
288     catch (InvalidAlgorithmParameterException e)
289     {
290         throw new CMSEException("can't setup the CertStore", e);
291     }
292 }
293 private void addCertsFromSet(List certs , ASN1Set certSet ,
294     Provider provider)
295     throws CMSEException
296 {
297     CertificateFactory cf;
298     try
299     {
300         if (provider != null)
301         {

```

```

303         cf = CertificateFactory.getInstance("X.509",
304             provider);
305     }
306     else
307     {
308         cf = CertificateFactory.getInstance("X.509");
309     }
310 }
311 catch (CertificateException ex)
312 {
313     throw new CMSEException("can't get certificate factory.",
314         ex);
315 }
316 Enumeration e = certSet.getObjects();
317 while (e.hasMoreElements())
318 {
319     try
320     {
321         ASN1Primitive obj = ((ASN1Encodable)e.nextElement())
322             .toASN1Primitive();
323         if (obj instanceof ASN1Sequence)
324         {
325             certs.add(cf.generateCertificate(
326                 new ByteArrayInputStream(obj.getEncoded())));
327         }
328     }
329     catch (IOException ex)
330     {
331         throw new CMSEException(
332             "can't re-encode certificate!", ex);
333     }
334     catch (CertificateException ex)
335     {
336         throw new CMSEException(
337             "can't re-encode certificate!", ex);
338     }
339 }
340
341 private void addCRLsFromSet(List crls, ASN1Set certSet, Provider
342     provider)
343     throws CMSEException
344 {
345     CertificateFactory cf;
346     try
347     {
348         if (provider != null)
349         {
350             cf = CertificateFactory.getInstance("X.509",
351                 provider);
352         }
353         else
354         {
355             cf = CertificateFactory.getInstance("X.509");
356         }
357     }
358     catch (CertificateException ex)
359     {
360         throw new CMSEException("can't get certificate factory.",
361             ex);
362     }
363     Enumeration e = certSet.getObjects();
364     while (e.hasMoreElements())
365     {

```

```

365         try
366         {
367             ASN1Primitive obj = ((ASN1Encodable)e.nextElement())
368                 .toASN1Primitive();
369             crls.add(cf.generateCRL(
370                 new ByteArrayInputStream(obj.getEncoded())));
371         }
372     catch (IOException ex)
373     {
374         throw new CMSEException("can't re-encode CRL!", ex);
375     }
376     catch (CRLEException ex)
377     {
378         throw new CMSEException("can't re-encode CRL!", ex);
379     }
380     }
381 }
382
383 AlgorithmIdentifier fixAlgID(AlgorithmIdentifier algId)
384 {
385     if (algId.getParameters() == null)
386     {
387         return new AlgorithmIdentifier(algId.getObjectId(),
388             DERNull.INSTANCE);
389     }
390     return algId;
391 }
392
393 void setSigningEncryptionAlgorithmMapping(DERObjectIdentifier
394     oid, String algorithmName)
395 {
396     encryptionAlgs.put(oid.getId(), algorithmName);
397 }
398
399 void setSigningDigestAlgorithmMapping(DERObjectIdentifier oid,
400     String algorithmName)
401 {
402     digestAlgs.put(oid.getId(), algorithmName);
403 }

```

implementacao/ProjetoPADES_ICPBrasil/src/util/MyCMSSignedHelper.java

```

1 package util;
2 import java.io.IOException;
3 import java.io.InputStream;
4 import java.io.OutputStream;
5 import java.security.NoSuchProviderException;
6 import java.security.Provider;
7 import java.security.Security;
8 import java.security.cert.CRLEException;
9 import java.security.cert.CertStore;
10 import java.security.cert.CertStoreException;
11 import java.security.cert.CertificateEncodingException;
12 import java.security.cert.X509CRL;
13 import java.security.cert.X509Certificate;
14 import java.util.ArrayList;
15 import java.util.Collection;
16 import java.util.Iterator;
17 import java.util.List;
18
19 import org.apache.commons.io.output.NullOutputStream;
20 import org.bouncycastle.asn1.ASN1Encodable;
21 import org.bouncycastle.asn1.ASN1EncodableVector;
22 import org.bouncycastle.asn1.ASN1InputStream;

```

```

23 import org.bouncycastle.asn1.ASN1Object;
import org.bouncycastle.asn1.ASN1Primitive;
25 import org.bouncycastle.asn1.ASN1Set;
import org.bouncycastle.asn1.BEROctetStringGenerator;
27 import org.bouncycastle.asn1.BERSet;
import org.bouncycastle.asn1.DERSet;
29 import org.bouncycastle.asn1.DERTaggedObject;
import org.bouncycastle.asn1.cms.ContentInfo;
31 import org.bouncycastle.asn1.cms.IssuerAndSerialNumber;
import org.bouncycastle.asn1.x509.CertificateList;
33 import org.bouncycastle.asn1.x509.TBSCertificateStructure;
import org.bouncycastle.asn1.x509.X509CertificateStructure;
35 import org.bouncycastle.cert.X509AttributeCertificateHolder;
import org.bouncycastle.cert.X509CRLHolder;
37 import org.bouncycastle.cert.X509CertificateHolder;
import org.bouncycastle.cms.CMSException;
39 import org.bouncycastle.cms.SignerInfoGenerator;
import org.bouncycastle.operator.DigestCalculator;
41 import org.bouncycastle.util.Store;
import org.bouncycastle.util.io.Streams;
43 import org.bouncycastle.util.io.TeeInputStream;
import org.bouncycastle.util.io.TeeOutputStream;
45
public class MyCMSUtils
47 {
    static ContentInfo readContentInfo(
49     byte[] input)
        throws CMSException
51     {
53         // enforce limit checking as from a byte array
        return readContentInfo(new ASN1InputStream(input));
55     }
57     static ContentInfo readContentInfo(
        InputStream input)
        throws CMSException
59     {
61         // enforce some limit checking
        return readContentInfo(new ASN1InputStream(input));
63     }
65     static List getCertificatesFromStore(CertStore certStore)
        throws CertStoreException, CMSException
67     {
        List certs = new ArrayList();
69         try
71         {
            for (Iterator it = certStore.getCertificates(null).
                iterator(); it.hasNext();)
73             {
                X509Certificate c = (X509Certificate) it.next();
75                 certs.add(X509CertificateStructure.getInstance(
                    ASN1Primitive
                        .fromByteArray(
                            c.
                                getEncoded(
                                    ))));
77             }
79             return certs;
81         } catch (IllegalArgumentException e)
            {
83             throw new CMSException("error processing certs", e);
85         } catch (IOException e)

```

```

87     {
88         throw new CMSEException("error processing certs", e);
89     }
90     catch (CertificateEncodingException e)
91     {
92         throw new CMSEException("error encoding certs", e);
93     }
94 }
95 static List getCertificatesFromStore(Store certStore)
96     throws CMSEException
97 {
98     List certs = new ArrayList();
99     try
100    {
101        for (Iterator it = certStore.getMatches(null).iterator()
102            ; it.hasNext();)
103        {
104            X509CertificateHolder c = (X509CertificateHolder)it.
105                next();
106            certs.add(c.toASN1Structure());
107        }
108        return certs;
109    }
110    catch (ClassCastException e)
111    {
112        throw new CMSEException("error processing certs", e);
113    }
114 }
115
116 static List getAttributeCertificatesFromStore(Store attrStore)
117     throws CMSEException
118 {
119     List certs = new ArrayList();
120     try
121    {
122        for (Iterator it = attrStore.getMatches(null).iterator()
123            ; it.hasNext();)
124        {
125            X509AttributeCertificateHolder attrCert = (
126                X509AttributeCertificateHolder)it.next();
127            certs.add(new DERTaggedObject(false, 2, attrCert.
128                toASN1Structure()));
129        }
130        return certs;
131    }
132    catch (ClassCastException e)
133    {
134        throw new CMSEException("error processing certs", e);
135    }
136 }
137
138 static List getCRLsFromStore(CertStore certStore)
139     throws CertStoreException, CMSEException
140 {
141     List crls = new ArrayList();
142     try
143    {
144        for (Iterator it = certStore.getCRLs(null).iterator()
145            ; it.hasNext();)
146        {
147            X509CRL c = (X509CRL)it.next();

```

```

149         crls.add(CertificateList.getInstance(ASN1Primitive.
150             fromByteArray(c.getEncoded())));
151     }
152     }
153     return crls;
154 }
155 catch (IllegalArgumentException e)
156 {
157     throw new CMSEException("error processing crls", e);
158 }
159 catch (IOException e)
160 {
161     throw new CMSEException("error processing crls", e);
162 }
163 catch (CRLEException e)
164 {
165     throw new CMSEException("error encoding crls", e);
166 }
167 }
168
169 static List getCRLsFromStore(Store crlStore)
170     throws CMSEException
171 {
172     List certs = new ArrayList();
173
174     try
175     {
176         for (Iterator it = crlStore.getMatches(null).iterator();
177             it.hasNext();)
178         {
179             X509CRLHolder c = (X509CRLHolder)it.next();
180             certs.add(c.toASN1Structure());
181         }
182
183         return certs;
184     }
185     catch (ClassCastException e)
186     {
187         throw new CMSEException("error processing certs", e);
188     }
189 }
190
191 static ASN1Set createBerSetFromList(List derObjects)
192 {
193     ASN1EncodableVector v = new ASN1EncodableVector();
194
195     for (Iterator it = derObjects.iterator(); it.hasNext();)
196     {
197         v.add((ASN1Encodable)it.next());
198     }
199
200     return new BERSet(v);
201 }
202
203 static ASN1Set createDerSetFromList(List derObjects)
204 {
205     ASN1EncodableVector v = new ASN1EncodableVector();
206
207     for (Iterator it = derObjects.iterator(); it.hasNext();)
208     {
209         v.add((ASN1Encodable)it.next());
210     }
211
212     return new DERSet(v);
213 }
214
215 static OutputStream createBEROctetOutputStream(OutputStream s,

```

```

        int tagNo, boolean isExplicit, int bufferSize) throws
        IOException
217     {
218         BEROctetStringGenerator octGen = new BEROctetStringGenerator
219             (s, tagNo, isExplicit);
220
221         if (bufferSize != 0)
222         {
223             return octGen.getOctetOutputStream(new byte[bufferSize])
224                 ;
225         }
226         return octGen.getOctetOutputStream();
227     }
228
229     static TBSCertificateStructure getTBSCertificateStructure(
230         X509Certificate cert)
231     {
232         try
233         {
234             return TBSCertificateStructure.getInstance(
235                 ASN1Primitive.fromByteArray(cert.getTBSCertificate())
236             );
237         }
238         catch (Exception e)
239         {
240             throw new IllegalArgumentException(
241                 "can't extract TBS structure from this cert");
242         }
243     }
244
245     static IssuerAndSerialNumber getIssuerAndSerialNumber(
246         X509Certificate cert)
247     {
248         TBSCertificateStructure tbsCert = getTBSCertificateStructure
249             (cert);
250         return new IssuerAndSerialNumber(tbsCert.getIssuer(),
251             tbsCert.getSerialNumber().getValue());
252     }
253
254     private static ContentInfo readContentInfo(
255         ASN1InputStream in)
256         throws CMSEException
257     {
258         try
259         {
260             return ContentInfo.getInstance(in.readObject());
261         }
262         catch (IOException e)
263         {
264             throw new CMSEException("IOException reading content.", e);
265         }
266         catch (ClassCastException e)
267         {
268             throw new CMSEException("Malformed content.", e);
269         }
270         catch (IllegalArgumentException e)
271         {
272             throw new CMSEException("Malformed content.", e);
273         }
274     }
275
276     public static byte[] streamToByteArray(
277         InputStream in)
278         throws IOException
279     {
280         return Streams.readAll(in);
281     }

```



```
277 public static byte[] streamToByteArray(
279     InputStream in,
281     int limit)
    throws IOException
    {
283     return Streams.readAllLimited(in, limit);
    }
285
286 public static Provider getProvider(String providerName)
    throws NoSuchProviderException
    {
289     if (providerName != null)
        {
291         Provider prov = Security.getProvider(providerName);
293         if (prov != null)
            {
295             return prov;
            }
297         throw new NoSuchProviderException("provider " +
299             providerName + " not found.");
        }
301     return null;
    }
303
304 static InputStream attachDigestsToInputStream(Collection digests
    , InputStream s)
    {
305     InputStream result = s;
307     Iterator it = digests.iterator();
309     while (it.hasNext())
        {
311         DigestCalculator digest = (DigestCalculator)it.next();
            result = new TeeInputStream(result, digest.
                getOutputStream());
        }
313     return result;
    }
315
316 static OutputStream attachSignersToOutputStream(Collection
    signers, OutputStream s)
    {
317     OutputStream result = s;
319     Iterator it = signers.iterator();
321     while (it.hasNext())
        {
323         SignerInfoGenerator signerGen = (SignerInfoGenerator)it.
            next();
            result = getSafeTeeOutputStream(result, signerGen.
                getCalculatingOutputStream());
        }
325     return result;
    }
327
328 static OutputStream getSafeOutputStream(OutputStream s)
    {
329     return s == null ? new NullOutputStream() : s;
    }
331
332 static OutputStream getSafeTeeOutputStream(OutputStream s1,
    OutputStream s2)
    {
333     return s1 == null ? getSafeOutputStream(s2)
335         : s2 == null ? getSafeOutputStream(s1) : new
337         TeeOutputStream(
            s1, s2);
    }
```

```
339 }
}
```

implementacao/ProjetoPAdES_ICPBrasil/src/util/MyCMSUtils.java

```

package util;
2
import com.itextpdf.text.log.Logger;
4 import com.itextpdf.text.log.LoggerFactory;
import com.itextpdf.text.pdf.CrlClient;
6
import java.io.InputStream;
8 import java.net.URL;
import java.security.cert.CertificateFactory;
10 import java.security.cert.X509CRL;
import java.security.cert.X509Certificate;
12
import org.bouncycastle.asn1.ASN1InputStream;
14 import org.bouncycastle.asn1.ASN1OctetString;
import org.bouncycastle.asn1.DERIA5String;
16 import org.bouncycastle.asn1.x509.CRLDistPoint;
import org.bouncycastle.asn1.x509.DistributionPoint;
18 import org.bouncycastle.asn1.x509.DistributionPointName;
import org.bouncycastle.asn1.x509.GeneralName;
20 import org.bouncycastle.asn1.x509.GeneralNames;
22
/**
 *
24 * @author psoares
 */
26 public class MyCrlClientImp implements CrlClient {
    private static final Logger LOGGER = LoggerFactory.getLogger(
        MyCrlClientImp.class);
28    String caminhoLCR;
    public MyCrlClientImp()
30    {
32    }
34    public byte [] getEncoded(X509Certificate checkCert, String url)
    {
36        byte [] retorno = null;
        String urlAux = "";
38        try
        {
40            ASN1InputStream aIn = new ASN1InputStream(checkCert.
                getExtensionValue("2.5.29.31"));
            ASN1OctetString octs = (ASN1OctetString)aIn.readObject();
42            aIn = new ASN1InputStream(octs.getOctets());
            CRLDistPoint crldp = CRLDistPoint.getInstance(aIn.readObject
                ());
            DistributionPoint[] dsp = crldp.getDistributionPoints();
            DistributionPointName dspn = dsp[0].getDistributionPoint();
44            GeneralName[] genNames = GeneralNames.getInstance(dspn.
                getName()).getNames();
46
            for (int i = 0; i < genNames.length; ++i)
            {
50                if (genNames[i].getTagNo() == GeneralName.
                    uniformResourceIdentifier)
                    {
52                    urlAux = DERIA5String.getInstance(genNames[i].getName()
                        ).getString();
54
                    X509CRL lcr = downloadLCR(urlAux);
56
                    retorno = lcr.getEncoded();
58                }
            }
        }
    }
}

```

```

    }
60     }
    catch (Exception ex)
62     {
        if (urlAux.startsWith("c:/"))
64         {
            retorno = UtilArquivo.read(urlAux);
66         }
    }
68     return retorno;
70 }

72 private X509CRL downloadLCR(String url) throws Exception
74 {
    X509CRL retorno = null;
76     URL objUrl = new URL(url);
        InputStream lcrStream = objUrl.openStream();
78     CertificateFactory cf = CertificateFactory.getInstance("X.509"
        );
        retorno = (X509CRL) cf.generateCRL(lcrStream);
80     lcrStream.close();
82     return retorno;
    }
84 }

```

implementacao/ProjetoPADES_ICPBrasil/src/util/MyCrlClientImp.java

```

package util;
2 import java.util.Date;
import java.util.Hashtable;
4 import java.util.Map;

6 import org.bouncycastle.asn1.ASN1ObjectIdentifier;
import org.bouncycastle.asn1.DERIA5String;
8 import org.bouncycastle.asn1.DERObjectIdentifier;
import org.bouncycastle.asn1.DEROctetString;
10 import org.bouncycastle.asn1.DERSet;
import org.bouncycastle.asn1.cms.Attribute;
12 import org.bouncycastle.asn1.cms.AttributeTable;
import org.bouncycastle.asn1.cms.CMSAttributes;
14 import org.bouncycastle.asn1.cms.Time;
import org.bouncycastle.asn1.esf.OtherHashAlgAndValue;
16 import org.bouncycastle.asn1.esf.SPuri;
import org.bouncycastle.asn1.esf.SigPolicyQualifierInfo;
18 import org.bouncycastle.asn1.esf.SigPolicyQualifiers;
import org.bouncycastle.asn1.esf.SignaturePolicyId;
20 import org.bouncycastle.asn1.esf.SignaturePolicyIdentifier;
import org.bouncycastle.asn1.x509.AlgorithmIdentifier;
22 import org.bouncycastle.cms.CMSAttributeTableGenerator;

24 /**
 * Default signed attributes generator.
 */
26 public class MySignedAttributeTableGenerator
    implements CMSAttributeTableGenerator
28 {
    private final Hashtable table;

30     /**
 * Initialise to use all defaults
 */
32     public MySignedAttributeTableGenerator()
34     {
        table = new Hashtable();
36     }
38 }

```

```

40  /**
41  * Initialise with some extra attributes or overrides.
42  *
43  * @param attributeTable initial attribute table to use.
44  */
45  public MySignedAttributeTableGenerator(
46      AttributeTable attributeTable)
47  {
48      if (attributeTable != null)
49      {
50          table = attributeTable.toHashtable();
51      }
52      else
53      {
54          table = new Hashtable();
55      }
56  }
57
58  /**
59  * Create a standard attribute table from the passed in
60  * parameters – this will
61  * normally include contentType, signingTime, and messageDigest.
62  * If the constructor
63  * using an AttributeTable was used, entries in it for
64  * contentType, signingTime, and
65  * messageDigest will override the generated ones.
66  *
67  * @param parameters source parameters for table generation.
68  * @return a filled in Hashtable of attributes.
69  */
70  protected Hashtable createStandardAttributeTable(
71      Map parameters)
72  {
73      Hashtable std = (Hashtable)table.clone();
74
75      if (!std.containsKey(CMSAttributes.contentType))
76      {
77          DERObjectIdentifier contentType = (DERObjectIdentifier)
78              parameters.get(CMSAttributeTableGenerator.
79                  CONTENT_TYPE);
80
81          // contentType will be null if we're trying to generate
82          // a counter signature.
83          if (contentType != null)
84          {
85              Attribute attr = new Attribute(CMSAttributes.
86                  contentType,
87                  new DERSet(contentType));
88              std.put(attr.getAttrType(), attr);
89          }
90
91          // PAdES(CAdES) não permite o uso do atributo signing-time.
92          // Deve-se utilizar o "M" dicionário de assinatura.
93          // if (!std.containsKey(CMSAttributes.signingTime))
94          // {
95          //     Date signingTime = new Date();
96          //     Attribute attr = new Attribute(CMSAttributes.
97          // signingTime,
98          //         new DERSet(new Time(signingTime)));
99          //     std.put(attr.getAttrType(), attr);
100         }
101
102         if (!std.containsKey(CMSAttributes.messageDigest))
103         {
104             byte[] messageDigest = (byte[])parameters.get(
105                 CMSAttributeTableGenerator.DIGEST);

```

```

100         Attribute attr = new Attribute(CMSAttributes.
            messageDigest,
            new DERSet(new DEROctetString(messageDigest)));
102         std.put(attr.getAttrType(), attr);
104     }
106     return std;
108 }
109 /**
110  * @param parameters source parameters
111  * @return the populated attribute table
112  */
113 public AttributeTable getAttributes(Map parameters)
114 {
115     return new AttributeTable(createStandardAttributeTable(
        parameters));
116 }

```

implementacao/ProjetoPAdES_ICPBrasil/src/util/MySignedAttributeTableGenerator.java

```

package util;
2
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
6
public class UtilArquivo {
8
    public static byte[] read(String aInputFileName){
10        File file = new File(aInputFileName);
11        byte[] result = new byte[(int) file.length()];
12        try {
13            InputStream input = null;
14            try {
15                int totalBytesRead = 0;
16                input = new BufferedInputStream(new FileInputStream(file));
17                while(totalBytesRead < result.length){
18                    int bytesRemaining = result.length - totalBytesRead;
19                    int bytesRead = input.read(result, totalBytesRead,
20                        bytesRemaining);
21                    if (bytesRead > 0){
22                        totalBytesRead = totalBytesRead + bytesRead;
23                    }
24                }
25            }
26            finally {
27                input.close();
28            }
29        }
30        catch (Exception ex) {
31            ex.printStackTrace();
32        }
33        return result;
34    }
35 }
36 }

```

implementacao/ProjetoPAdES_ICPBrasil/src/util/UtilArquivo.java

```

package view;
2
import java.awt.BorderLayout;
import java.awt.FlowLayout;
4

```

```

6  import javax.swing.JButton;
import javax.swing.JDialog;
8  import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
10 import javax.swing.SwingConstants;
import java.awt.event.ActionListener;
12 import java.awt.event.ActionEvent;
import java.security.Security;

14
import javax.swing.JCheckBox;
16 import javax.swing.JFileChooser;
import javax.swing.JTabbedPane;
18 import javax.swing.JTextArea;
import javax.swing.JSeparator;

20
import core.AssinadorPADES;
import core.PdfSimples;
import javax.swing.JTextField;

24
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import javax.swing.JScrollPane;
import javax.swing.JLabel;
28 import java.awt.Font;

30 public class TelaAssinador extends JDialog implements ActionListener
    {

32     private static final ActionListener Internal = null;
private final JPanel contentPanel = new JPanel();
34     private JTextField txtSenha;
String CRIAR_DOCUMENTO = "Criar PDF";
36     String VERIFICAR = "Verificar";
String ASSINAR = "Assinar";
38     String SELECIONAR_P12 = "Selecionar .p12";
String SELECIONAR_LCR = "Adicionar DSS";
40     String caminhoP12;
String caminhoLCR;
42     JTextArea textAreaLog = new JTextArea();
JCheckBox cbCriarRevisao;

44
/**
46     * Launch the application.
*/
48     public static void main(String[] args) {
try {
50         TelaAssinador dialog = new TelaAssinador();
dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
52         dialog.setVisible(true);
} catch (Exception e) {
54             e.printStackTrace();
}
56     }

58     /**
* Create the dialog.
*/
60     public TelaAssinador() {
62         setBounds(100, 100, 423, 438);
getContentPane().setLayout(new BorderLayout());
64         contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
getContentPane().add(contentPanel, BorderLayout.CENTER);
66         contentPanel.setLayout(null);
{
68             JButton btnNewButton = new JButton("Assinar");
btnNewButton.addActionListener(this);
70             btnNewButton.setBounds(300, 143, 89, 23);
contentPanel.add(btnNewButton);
72         }

```

```

74  JButton btnNewButton_1 = new JButton("Criar PDF");
75  btnNewButton_1.addActionListener(this);
76  btnNewButton_1.setBounds(63, 44, 264, 23);
77  contentPanel.add(btnNewButton_1);
78
79  JCheckBox chckbxNewCheckBox = new JCheckBox("Pol\u00E9tica AD-RB
80  ");
81  chckbxNewCheckBox.setEnabled(false);
82  chckbxNewCheckBox.setSelected(true);
83  chckbxNewCheckBox.setBounds(20, 91, 264, 23);
84  contentPanel.add(chckbxNewCheckBox);
85
86  cbCriarRevisao = new JCheckBox("Criar Revis\u00E3o");
87  cbCriarRevisao.setSelected(true);
88  cbCriarRevisao.setBounds(20, 117, 264, 23);
89  contentPanel.add(cbCriarRevisao);
90
91  JButton btnVerificar = new JButton("Verificar");
92  btnVerificar.addActionListener(this);
93  btnVerificar.setBounds(300, 366, 89, 23);
94  contentPanel.add(btnVerificar);
95
96  JSeparator separator = new JSeparator();
97  separator.setBounds(10, 177, 381, 2);
98  contentPanel.add(separator);
99
100 JSeparator separator_1 = new JSeparator();
101 separator_1.setBounds(10, 78, 387, 2);
102 contentPanel.add(separator_1);
103
104 JButton btnSelecionarp = new JButton("Selecionar .p12");
105 btnSelecionarp.addActionListener(this);
106
107 btnSelecionarp.setBounds(20, 143, 124, 23);
108 contentPanel.add(btnSelecionarp);
109
110 txtSenha = new JTextField();
111 txtSenha.setText("senha");
112 txtSenha.setBounds(154, 144, 136, 20);
113 contentPanel.add(txtSenha);
114 txtSenha.setColumns(10);
115
116 JScrollPane scrollPane = new JScrollPane();
117 scrollPane.setBounds(20, 190, 363, 165);
118 contentPanel.add(scrollPane);
119 scrollPane.setViewportView(textAreaLog);
120 textAreaLog.setLineWrap(true);
121
122 JButton btnSelecionarLcr = new JButton("Adicionar DSS");
123 btnSelecionarLcr.addActionListener(this);
124 btnSelecionarLcr.setBounds(20, 367, 124, 23);
125 contentPanel.add(btnSelecionarLcr);
126
127 JLabel lblNewLabel = new JLabel("Assinador PAdES");
128 lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 15));
129 lblNewLabel.setBounds(10, 4, 238, 29);
130 contentPanel.add(lblNewLabel);
131 }
132
133 public void actionPerformed(ActionEvent e)
134 {
135     Security.addProvider(new BouncyCastleProvider());
136     JFileChooser chooseDialog = new JFileChooser();
137     int rVal = 0;
138     textAreaLog.setText("");
139
140     if(((JButton)e.getSource()).getText() == CRIAR_DOCUMENTO)
141     {
142         rVal = chooseDialog.showSaveDialog(TelaAssinador.this);
143     }
144 }

```

```

142     if (rVal == JFileChooser.APPROVE_OPTION)
143     {
144         PdfSimples.criar("TCC PAdES", chooseDialog.getSelectedFile()
145             .getAbsolutePath());
146     }
147 }
148 else if(((JButton)e.getSource()).getText() == SELECIONAR_P12)
149 {
150     rVal = chooseDialog.showOpenDialog(TelaAssinador.this);
151     if (rVal == JFileChooser.APPROVE_OPTION)
152     {
153         caminhoP12 = chooseDialog.getSelectedFile().getAbsolutePath
154             ();
155     }
156 }
157 else if(((JButton)e.getSource()).getText() == ASSINAR)
158 {
159     rVal = chooseDialog.showOpenDialog(TelaAssinador.this);
160     if (rVal == JFileChooser.APPROVE_OPTION)
161     {
162         chooseDialog.getSelectedFile().getAbsolutePath();
163         AssinadorPAdES obj = new AssinadorPAdES();
164         obj.properties.setProperty("PASSWORD", txtSenha.getText());
165         obj.properties.setProperty("PKCS12", caminhoP12);
166
167         String caminhoArquivo = chooseDialog.getSelectedFile().
168             getAbsolutePath();
169         boolean criarRevisao = cbCriarRevisao.isSelected();
170
171         textAreaLog.setText("Arquivo assinado:\n"+ obj.assinar(
172             caminhoArquivo, criarRevisao));
173     }
174 }
175 else if(((JButton)e.getSource()).getText() == VERIFICAR)
176 {
177     rVal = chooseDialog.showOpenDialog(TelaAssinador.this);
178     if (rVal == JFileChooser.APPROVE_OPTION)
179     {
180         String caminhoPDF = chooseDialog.getSelectedFile().
181             getAbsolutePath();
182         AssinadorPAdES obj = new AssinadorPAdES();
183
184         String statusVerificacao = obj.verificar(caminhoPDF);
185
186         textAreaLog.setText(statusVerificacao);
187     }
188 }
189 else if(((JButton)e.getSource()).getText() == SELECIONAR_LCR)
190 {
191     rVal = chooseDialog.showOpenDialog(TelaAssinador.this);
192     if (rVal == JFileChooser.APPROVE_OPTION)
193     {
194         String caminhoPDF = chooseDialog.getSelectedFile().
195             getAbsolutePath();
196         AssinadorPAdES obj = new AssinadorPAdES();
197
198         obj.adicionarDSS(caminhoPDF, caminhoPDF + "_DSS.pdf");
199         String statusVerificacao = "\nArquivo com DSS:\n"+
200             caminhoPDF + "_DSS.pdf";
201
202         textAreaLog.setText(statusVerificacao);
203     }
204 }
205 }

```

implementacao/ProjetoPAdES_ICPBrasil/src/view/TelaAssinador.java

B Apêndice B –Artigo

Aplicabilidade do Padrão de Assinatura PAdES no Âmbito da ICP-Brasil

Alex Sandro da Silva Pereira

INE-UFSC

Florianópolis-SC-Brasil

alexwellll@gmail.com

RESUMO

O PDF (Formato de Documento Portátil), criado em 1993 pela Adobe Systems e publicado como norma ISO 32000-1, desde 2008, é um padrão para visualização de documentos que permite encapsular assinaturas digitais no próprio arquivo. Em julho de 2008 o ETSI (European Telecommunications Standard Institute), definiu uma série de especificações de como assinaturas digitais podem ser integradas e visualizadas nesse formato portátil, conhecida como PAdES - PDF Advanced Electronic Signatures. No contexto nacional, o mecanismo que viabiliza a emissão de certificados digitais e garante a validade jurídica de documentos em forma eletrônica é a Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil), até o momento, a ICP-Brasil não disponibiliza regulamentações referentes ao processo de assinatura digital no padrão PAdES. O estudo da viabilidade da utilização do formato PAdES em conformidade com as normas e políticas da ICP-Brasil é objetivo de estudo deste trabalho. O que resultou na apresentação de propostas de representação das assinaturas digitais sob cada política de assinatura definida na ICP-Brasil, com exceção da assinatura digital com referências para validação, sendo possível analisar algumas mudanças na regulamentação do Padrão Brasileiro de Assinatura Digital decorrentes da adoção do padrão PAdES.

ABSTRACT

Digital repositories are structures serving the purpose of indexing and cataloging digital content. Collaborative systems are structures that allow users interaction to the performance of tasks. Collaborative systems may use mechanisms for regulating users actions, such as reputation systems. This work aims to propose a reputation system for digital repositories and the construction of a prototype. The PDF (Portable Document Format), created in 1993 by Adobe Systems and published as ISO 32000-1, since 2008 is a standard for viewing documents which allows you to encapsulate digital signatures in the file itself. In July 2008, the ETSI (European Telecommunications Standards Institute), defined a set of specifications that describes how digital signatures can be integrated and visualized in this portable format, known as PAdES - Advanced PDF Electronic Signatures. In the national context, the mechanism that enables the issuance of digital certificates and ensures the legal validity of electronic documents is the Brazilian Public Key Infrastructure (ICP-Brasil), until now, the ICP-Brasil does not provide regulations for the digital signature process in PAdES standard. The feasibility of using the format PAdES in accordance with the rules and policies of the ICP-Brasil is the objective of this study. This work resulted in proposals for representation of digital signatures on each signature policy defined by the ICP-Brasil, except for the digital signature with references for validation, being possible to analyze the feasible changes in the regulation of the Brazilian's Digital Signature Standard for the adoption of PAdES standard.

Keywords

PAdES, ICP-Brasil, Assinatura Digital, PDF.

1. Introdução

Um documento é uma fonte de informação, na forma material, capaz de ser usada para referência, estudo ou prova [1]. Um documento eletrônico, de acordo com projeto de lei sobre o comércio eletrônico [2], que "Dispõe sobre o valor probante do documento eletrônico e da assinatura digital, regula a certificação digital, institui normas para as transações de comércio eletrônico e dá outras providências.", em seu art. 2, inciso I, é "a informação gerada, enviada, recebida, armazenada ou comunicada por meios eletrônicos, ópticos, opto-eletrônicos ou similares". Para que haja confiança nas informações transitadas eletronicamente, tais documentos devem produzir efeito de prova, conhecida como eficácia probante.

Assinatura digital é uma tecnologia que permite a tramitação de documentos, processos e outras informações com segurança. Um formato bastante popular na representação de documentos eletrônicos é o Formato de Documento Portátil (PDF), capaz também de representar documentos digitalmente assinados. O padrão PDF suporta a assinatura digital do conteúdo representado, baseando-se, para tanto, em outras tecnologias de assinatura eletrônica, são elas CAdES [3], XAdES [4] e CMS [5].

Um ano após a publicação do formato PDF pela ISO, em julho de 2009, o ETSI (European Telecommunications Standard Institute) publicou um padrão que define uma série de especificações, de como pode ser utilizada assinatura digital em documentos PDF, conhecida como PAdES - PDF Advanced Electronic Signatures - que documentam e estendem o suporte de assinatura digital especificado na ISO 32000-1. Essa série é dividida em cinco partes, sendo que a primeira apresenta os recursos da assinatura PDF apenas considerando aspectos gerais. A partir da segunda, é descrito o formato básico do PAdES e subsequentemente as especificações para implementadores. Criado em 1988, o ETSI é uma organização responsável pela padronização de tecnologias da informação e comunicação na Europa. Então, a adição do PAdES ao conjunto de soluções para assinatura em PDF está em conformidade com as regulamentações legais da União Europeia.

O padrão de assinatura digital PAdES encontra-se em estudo para normatização segundo as regras definidas pela ICP-Brasil. Este artigo pretende analisar os impactos da adoção do padrão PAdES pela ICP-Brasil, buscando verificar quais as adaptações necessárias à normatização.

Apesar deste trabalho ter íntima ligação com aspectos jurídicos referentes à validade de documentos, este trabalho restringe-se ao estudo da assinatura digital, apenas sob os aspectos técnicos. O padrão PAdES suporta o uso de estruturas especificadas no padrão CAdES e XAdES, para representação de assinaturas. Neste trabalho, serão consideradas apenas as assinaturas CAdES, ficando as assinaturas PAdES-XAdES como proposta de trabalhos futuros. Este trabalho apresenta apenas uma proposta para os modelos de assinatura tratados pela ICP-Brasil, não define formatos de políticas de assinatura

PADES e nem formas de codificação de políticas para este tipo de assinatura, não implicando no formato final de como o padrão PADES será adotado pela infraestrutura.

Na seção 2, serão apresentadas definições dos principais conceitos relacionados ao processo da confecção de assinaturas eletrônicas. A seção 3, apresenta os principais padrões de assinatura digital. A seção 4 a Infraestrutura de Chaves Públicas Brasileira e mostra os formatos de assinaturas digitais aprovados no âmbito da ICP-Brasil. A seção 5 apresenta o formato de documento portátil PDF e a forma como este padrão suporta assinatura digital. Seguido pela seção 6 que mostra o padrão PADES, que mantém compatibilidade com o padrão PDF. A seção 7 apresenta a análise da viabilidade da utilização do PADES no âmbito da ICP-Brasil. A seção 8 exibe os resultados da implementação de um Assinador PADES que gera assinaturas no formato AD-RB. Por último a seção 9 apresenta algumas considerações finais.

2. Conceitos Básicos

Para a compreensão dos formatos de assinaturas digitais é necessário um apanhado geral dos principais conceitos envolvidos no processo de assinatura e verificação de documentos eletrônicos.

2.1. Criptografia Assimétrica

Criptografia é a escrita em códigos ou escrita de forma ilegível. Cripto, do grego "kryptos", significa escondido, oculto, e grafia, também do grego "graphos", representa escrita. De acordo com Schneier (1996), criptografia é a arte e a ciência de manter mensagens seguras.

A criptografia assimétrica [6] foi proposta por Whitfield Diffie e Martin E. Hellman e, publicada em 1976. Este conceito tem a mesma função da criptografia simétrica, cifrar e decifrar dados, porém, diferente dela, este processo é realizado por chaves distintas, ou seja, não utiliza uma única chave e sim, um par (uma pública e outra privada). A chave privada deve ser mantida em segredo, já a pública pode ser disponibilizada em locais públicos. O que é cifrado por uma é, somente, decifrado pela outra, e vice-versa.

2.2. Assinatura Digital

A assinatura digital possibilita a autenticidade e não repúdio do signatário, permitindo verificar a integridade dos dados. Ela é uma aplicação importante da criptografia assimétrica e resulta da utilização da chave privada sobre os dados. De acordo com Schneier (1996), uma assinatura digital, idealmente, deve possuir as seguintes propriedades:

- Prover autenticidade. O receptor tem certeza da identidade do signatário;
- Impossível de ser falsificada;
- A partir de um documento assinado não pode existir a possibilidade de transferir esta assinatura para outro documento;
- Proteger a integridade. O documento não pode ser alterado após assinado;
- Não permitir que o signatário negue a sua assinatura.

Como a utilização da criptografia assimétrica exige um custo computacional elevado, ela é inadequada para cifragem de conteúdos extensos. Logo, a forma tradicional de assinatura digital não ocorre sobre o documento em si, mas sim sobre uma identificação única do mesmo [7].

2.3. Elementos Associados a uma Assinatura

Como elementos principais podemos citar:

- Certificado digital. Possibilita a verificação da autenticidade do signatário;
- Atributos assinados. São informações que, em conjunto com o documento a ser assinado, fazem parte do cálculo da assinatura e;
- Atributos não assinados. São informações relevantes que podem ser associadas a assinatura após o momento de sua criação sem que esta seja invalidada. Um exemplo de atributo não assinado é o carimbo do tempo, este permite que assinatura seja associada a uma data segura e também provar que ela ocorreu antes de determinado momento. Mais informações sobre carimbo do tempo podem ser encontradas na RFC 3161 [8].

2.4. Política de Assinatura

Uma política de assinatura, de acordo com a especificação ETSI TR 102 272 [9], define um conjunto de regras utilizadas para a criação e validação de assinaturas digitais. Além disso, a política pode ser identificada de forma implícita ou explícita e especificada no formato livre ou estruturado. A forma livre refere-se a um formato legível por humanos, o que possibilita sua interpretação e avaliação no contexto em que foi aplicada. Já a forma estruturada é utilizada para codificar regras eletrônicas que facilitam os processos de geração e validação de assinaturas digitais.

2.5. Resumo Criptográfico

A FIPS PUB 186-3 [7], descreve função resumo como um processo que mapeia uma sequência de bits de tamanho arbitrário para uma sequência de comprimento fixo. Dessa forma, não importa o comprimento da informação, o resultado terá sempre o mesmo tamanho. Este resultado é conhecido como resumo criptográfico, impressão digital ou simplesmente *hash*. As funções resumo são projetadas para satisfazer as seguintes propriedades:

- Ser de difícil inversão. É computacionalmente inviável encontrar o conteúdo resumido a partir do resumo;
- Para entradas idênticas, devem ser geradas saídas idênticas.
- Resistentes a colisões. Para entradas distintas, deve ser improvável gerar uma mesma saída.

2.6. Infraestrutura de Chaves Públicas

De acordo com Housley e Polk (2001), uma infraestrutura de chaves públicas (ICP) é projetada para facilitar o uso da criptografia assimétrica e possibilita construir uma cadeia de confiança hierárquica. A entidade autoassinada, topo desta hierarquia conhecida como Autoridade Raiz, certifica e pode transferir a tarefa de certificação para outras entidades, ditas intermediárias ou subordinadas e finais, mediante a utilização da assinatura digital. O processo de construção do caminho de confiança entre a Autoridade Raiz e a Entidade Final é conhecido como caminho de certificação que segundo [10], deve ser construído e validado para estabelecer um ponto de confiança e verificar uma assinatura digital.

O certificado digital é um componente importante desta infraestrutura e será descrito de forma sucinta a seguir.

2.7. Certificado Digital

De acordo com a RFC 5280 [11], o certificado digital contém

propriedades como:

- A chave pública;
- Informações de uso da chave;
- Informações do detentor da chave;
- Informações do seu emissor;
- Assinatura digital do seu emissor. De acordo com Housley e Polk (2001), um certificado é protegido através da assinatura do seu emissor. Ou seja, ele é considerado uma mensagem assinada do ponto de vista do certificado que o emitiu. Além disso, esta assinatura permite atestar a confiança sobre este certificado.;
- Uma período de validade.

Assim, um certificado digital é um conjunto de informações que permitem identificar, de forma segura, o proprietário do par de chaves.

3. Padrões de Assinatura Digital

3.1. CMS Advanced Electronic Signatures

O CMS Advanced Electronic Signatures (CAAdES) [3], do European Telecommunications Standards Institute (ETSI) é um padrão de assinatura construído sobre o CMS [5], adicionando a este a predefinição de um conjunto de atributos assinados e não assinados, bem como o uso de políticas de assinatura, as quais são utilizadas para definir como uma assinatura deve ser gerada e validada.

3.1.1. CAAdES Basic Electronic Signature

Esta forma de assinatura deve ser utilizada quando não existe a necessidade de confiar na data em que a assinatura foi gerada, pois fornece proteção apenas sobre a integridade e autenticidade do conteúdo.

Como componentes de sua estrutura podemos citar o dado (objeto que será assinado), os atributos assinados obrigatórios, os atributos assinados opcionais e a assinatura calculada sobre o dado e os atributos assinados.

3.1.2. CAAdES Explicit Policy-based ES

As regras de conformidade de uma assinatura podem estar descritas de forma implícita ou explícita.

3.1.3. Electronic Signature with Time

Uma assinatura com ligação a uma data segura, além de provar que ela ocorreu antes de determinado momento, permite sua validação mesmo após o certificado do signatário ter expirado. Esta característica pode ser adquirida através da inclusão de um carimbo do tempo ou por uma marca do tempo. O carimbo do tempo é adicionado como um atributo não assinado e permite que uma assinatura possa ser validada a qualquer momento no futuro, desde que o certificado da autoridade de carimbo do tempo seja considerado válido. Já a marca do tempo não é adicionada na assinatura e, é de responsabilidade do provedor de serviço do tempo confiável prover evidência da marca quando for solicitado.

3.1.4. ES with Complete Validation Data

References

Idealmente, a assinatura com referência completa aos dados de validação deve ser construída sobre a que prove garantia de tempo confiável. Pois as informações de validação devem ser validadas contra uma determinada marca temporal. Logo, armazenar referência a estas informações e não prover uma

data confiável, impossibilita comprovar que a assinatura realmente ocorreu antes daquele período, nem se a marca temporal foi forjada para um momento em que as informações de validação não denunciassessem a invalidade das assinaturas.

3.1.5 Extended Electronic Signature Formats

Possuir ligação a uma data confiável e manter as informações de validação externos a assinatura, possibilita uma economia de espaço, porém estas informações devem estar sempre acessíveis no momento da validação. O formato de assinatura com referência completa aos dados de validação permite validação a longo prazo de forma online, uma vez que deve haver a validação de dados armazenados externamente.

Afim de possibilitar uma proteção a longo prazo de forma offline, o formato CAAdES-C pode ser estendido para:

- Extended Long Electronic Signature (CAAdES-X Long):
Permite evitar que os dados das informações se percam, pois, além das referências, os dados propriamente ditos são adicionados na assinatura. Apesar de demandar mais espaço, garante a disponibilidade dos dados de validação;
- Extended Electronic Signature with Time Type 1 (CAAdES-X Type 1):
Adiciona um carimbo de tempo sobre toda a assinatura. Permite proteção da integridade e fornece uma marcação de tempo confiável. Possibilita também proteção aos dados de validação, por exemplo, contra o comprometimento da chave da Autoridade Certificadora;
- Extended Electronic Signature with Time Type 2 (CAAdES-X Type 2):
Inclui um carimbo do tempo sobre as referências dos certificados e informações de revogação. Difere do formato anterior, pois provê proteção, apenas, aos dados de validação;
- Extended Long ES with Time (CAAdES-X Long Type 1 or 2):
É a combinação do formato CAAdES-X Long com um dos dois formatos CAAdES-X Type 1 ou CAAdES-X Type 2. Além de fornecer proteção sobre as referências dos dados de validação, também garante sua disponibilidade.

3.1.6 Archival Electronic Signature

O formato de arquivamento oferece proteção completa a todos os dados da assinatura, através da adição de um carimbo de tempo sobre o (CAAdES-X) Long ou (CAAdES-X Long Type 1 ou 2). Assim sendo, toda a assinatura fica condicionada à validade do carimbo de tempo, que pode ser renovado com a inclusão de novos carimbos e assim sucessivamente.

4. Padrão Brasileiro de Assinatura Digital

A Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil), foi instituída pela medida provisória 2200-2, de 28 de Junho de 2001 [12], para garantir a integridade, a autenticidade e a validade jurídica de documentos em formato eletrônico. Assim, a Infraestrutura de Chaves Públicas Brasileira é a instituição que regulamenta os critérios para a confiança e aceitação de documentos eletrônicos.

Sua estrutura é formada por uma hierarquia e uma autoridade gestora de políticas, intitulada Comitê Gestor da ICP-Brasil, com a responsabilidade de governar a infraestrutura. São responsabilidades do comitê gestor: estabelecer regras operacionais, diretrizes, licenciar e autorizar o funcionamento

dos componentes da hierarquia, dentre outras. A hierarquia conta com um ponto máximo de confiança, a Autoridade Certificadora Raiz (AC Raiz), Autoridades Certificadoras Subordinadas (ACs) e Autoridades de Registro (ARs). A AC Raiz é o topo da hierarquia de certificação, gerenciada pelo Instituto Nacional de Tecnologia da Informação, entidade incumbida de executar atividades relacionadas à gestão das autoridades. São responsabilidades do ITI: auditar, fiscalizar, gerenciar o ciclo de vida da AC-Raiz, credenciar ACs de segundo nível, dentre outras.

4.1. Políticas de Assinatura Digital

As políticas de assinaturas digitais permitidas na ICP-Brasil são cinco: assinatura digital com Referência Básica (AD-RB), com Referência do Tempo (AD-RT), com Referências para Validação (AD-RV), com Referências Completas (AD-RC) e com Referências para Arquivamento (AD-RA). Cada política encapsula características de sua antecessora, sendo a AD-RB o ponto de partida para todas as demais.

4.1.1. AD-RB

A utilização do formato de assinatura AD-RB, equivalente ao CADES-EPES e XAdES-EPES, ocorre quando não é necessário uma validação a longo prazo da assinatura pois esta não possui características de tempo confiável. Assim, este formato deve ser utilizado, apenas quando não for necessário prover uma validação após o prazo de validade do signatário.

4.1.2. AD-RT

O formato com referência do tempo (AD-RT), equivalente ao CADES-T e XAdES-T, é construído sobre o com referência básica e possui ligação a uma data segura provida pelo uso do carimbo do tempo. A referência temporal não é só importante quando se deseja saber a data, mas também traz a segurança de que uma posterior revogação ou expiração não comprometam a assinatura realizada, desde que a data da assinatura seja anterior ao evento.

4.1.3. AD-RV

Uma assinatura com Referências para Validação (AD-RV), equivalente ao CADES-C e XAdES-C, detém todas as características da com carimbo do tempo. Soma-se a sua estrutura referências para os dados de validação e um carimbo do tempo sobre estas referências, estes dados são considerados obrigatórios por esta política de assinatura.

4.1.4. AD-RC

Para evitar que esses tipos de problemas ocorram, pode-se fazer uso do formato de assinatura com referências completas (AD-RC). O formato de assinatura AD-RC, equivalente ao CADES-X e XAdES-X, além de armazenar as referências de validação exige que a assinatura encapsule os valores destas referências.

Assim, apesar de exigir mais espaço, este formato garante que os dados estejam sempre disponíveis e deve ser utilizado quando existe a necessidade que a assinatura contenha os dados de validação na própria estrutura o que possibilita uma maior proteção a longo prazo.

4.1.5. AD-RA

Caso seja necessário realizar um arquivamento dos dados assinados, é importante considerar que o estado dos algoritmos criptográficos utilizados para compor a assinatura sofre influência do tempo. Assim, a garantia de arquivamento esta ligada ao nível de segurança dos algoritmos que com o passar do tempo perde a força. O formato de assinatura com

referências para arquivamento (AD-RA), equivalente ao CADES-A e XAdES-A, construído sobre o AD-RC, porém não faz uso do carimbo do tempo sobre as referências. Este formato deve ser utilizado quando é necessário arquivar documentos a longo prazo mantendo aspectos como integridade, autenticidade e confidencialidade. Para isso, faz uso de um carimbo do tempo para proteger toda a assinatura e este pode ser protegido com a aplicação de outros carimbos.

5. Formato de Documento Portátil

A Adobe Systems desenvolveu, em 1993, o formato de documento portátil (*PDF*). Em 2007 a especificação deste formato foi descontinuado e preparada para transformar-se em norma ISO, o que ocorreu de fato no ano seguinte e resultou a ISO 32000-1:2008 [13]. A versão base do *PDF* descrita nesta norma é a 1.7 porém esta encapsula funcionalidades das versões 1.0 a 1.6.

A estrutura básica de um documento PDF é derivada do *PostScript* [14] que é uma linguagem de descrição de páginas que possui a função principal de descrever a aparência do texto, dos vetores, das imagens, dentre outros, independente do dispositivo utilizado.

Deter o conhecimento total sobre todas as funcionalidades do *PDF* é extremamente importante para a análise do impacto da adoção deste padrão como meio de representação de documentos eletrônicos seguros, ou seja, assinados digitalmente. Por ser um padrão extremamente abrangente e complexo, esta seção limitar-se-á às características que têm impacto direto no modo como é realizada uma assinatura digital neste tipo de documento.

5.1. Estrutura Básica

Um documento PDF pode ser interpretado como uma árvore de objetos, que tem como primeiro nó uma estrutura chamada de catálogo do documento. Este catálogo permite acesso a todos os objetos o que inclui as páginas, o número de páginas, o conteúdo das páginas, informações de como o conteúdo deve ser exibido, etc.

O PDF possui oito tipos básicos de objetos: booleanos, numéricos, literais, nomes, objeto nulo, *arrays*, dicionários e *streams*. Os três últimos são considerados *containers* e podem abrigar outros objetos resultando em tipos complexos. Estes objetos estão localizados por toda a estrutura do *PDF*, que inicialmente é constituída por quatro partes: Cabeçalho, Corpo, Tabela de Referência Cruzada e *Trailer*.

- Cabeçalho: identifica a versão do PDF. O valor *%PDF-1.4*, por exemplo, indica que o arquivo está em conformidade com a versão 1.4;
- Corpo: contém uma sequencia de objetos indiretos, estes objetos representam os componentes do documento como fontes, paginas, imagens etc;
- Tabela de Referência Cruzada: mostra a localização dos objetos no *Corpo*, indica se o mesmo está ativo ou inativo e permite o acesso aleatório a eles. Uma característica importante desta tabela é o modo como ela informa a localização dos objetos. Cada objeto possui uma posição no arquivo, e esta posição é demarcada pela distância (em bytes) entre o início do documento e o início do objeto;
- Trailer: informa a localização da *Tabela de Referência Cruzada*, do dicionário de catalogo do documento e pode referenciar alguns objetos do *Corpo*. Caso existam atualizações incrementais, a *Tabela de Referência Cruzada* da revisão anterior

também faz parte do conjunto de informações do *Trailer*.

5.2. Atualização

A atualização de um documento, por exemplo pela simples adição de um objeto, implica na alteração não só do corpo do documento, mas pode afetar a posição de todos os objetos subsequentes. A tabela de referência cruzada deverá ser alterada para refletir a mudança e, conseqüentemente, sua localização também deverá ser atualizada no *trailer*.

O padrão PDF suporta o conceito de "revisões", encapsulamentos que permitem a atualização incremental de um documento, através da adição de novo conteúdo ao final do arquivo. Para comportar o novo conteúdo é necessário criar uma nova estrutura: corpo, tabela de referência cruzada e trailer. Esta nova estrutura, que não irá conter um cabeçalho, tem a mesma função da primeira, com exceção do trailer, que além de referenciar a tabela cruzada desta revisão irá também referenciar a antiga.

5.3. Assinatura Digital

A assinatura pode ocorrer através da utilização da assinatura digital ou através de biometria. A forma de validação de uma assinatura deve ser através de um módulo especial, chamado *signature handler*. Este módulo encontra-se integrado aos leitores de arquivos *PDF*, sendo possível também a sua extensão, por terceiros, na forma de plug-ins, afim de verificar ou gerar uma assinatura com características específicas.

As entradas principais do dicionário de assinatura são:

- **Filter:** Deve conter o nome do módulo principal que irá realizar a verificação da assinatura (*signature handler*);
- **SubFilter:** Refere-se à codificação da assinatura, armazenada na entrada *Contents*;
- **Contents:** Contém a estrutura da assinatura, no formato PKCS#1 ou PKCS#7 codificada em Hexadecimal;
- **Certs:** Deve-se incluir a cadeia de certificação do signatário, utilizado apenas para assinaturas PKCS#1;
- **ByteRange:** Indica qual o conteúdo assinado;
- **Reference:** Pode conter outros dicionários que auxiliam na detecção de modificações do documento;
- **M:** Indica a data e hora da assinatura;
- **Location:** Indica o local da assinatura;
- **Reason:** Indica a razão da assinatura;
- **ContactInfo:** Informações de contato do signatário como por exemplo o número do telefone;
- **Name:** O nome o signatário. Utilizado somente se não for possível extrair o nome do signatário a partir da assinatura;
- **Changes:** Indica as mudanças realizadas deste a última assinatura;
- **R:** A versão do *Signature Handler* utilizado para criar a assinatura;

O dicionário de assinatura está relacionado a uma segunda estrutura: o Campo de Assinatura. O Campo de Assinatura consiste em outro *container*, criado no momento da assinatura, que serve para definir a forma gráfica do campo de assinatura, além de associar alguns atributos adicionais ao dicionário de

assinatura.

Informações referentes à assinatura podem ser incluídas na entrada *seed value* desta estrutura. Esta entrada faz referência para um outro *container* que provê informações de restrições que devem ser aplicadas no momento da assinatura. Estas restrições incluem: tipo de algoritmo que pode ser utilizado, uso da chave do certificado do signatário, tipo de estrutura da assinatura, etc.

O dicionário de assinatura, assim como todo o conteúdo, deve ser assinado, com exceção da entrada *Contents*, que receberá posteriormente a assinatura propriamente dita. Esta é uma característica bastante peculiar do documento PDF, que implica na seguinte sequência de passos:

- Criação das estruturas Campo de Assinatura e do Dicionário de Assinatura, alocando o espaço necessário para armazenar o conteúdo da assinatura;
- Preenchimento dos atributos do Dicionário de Assinatura, com exceção do *Contents*;
- Cálculo do resumo criptográfico sobre o documento, incluindo o dicionário de assinatura, com exceção do *Contents*;
- Preenchimento do *Contents* com os valores reais codificados em Hexadecimal.

A estrutura da assinatura armazenada na entrada *Contents* do dicionário de assinatura pode ser tipo PKCS#1 [15] ou PKCS#7 [16].

5.4. Assinatura de Certificação

Essa assinatura trabalha em conjunto com a funcionalidade de permissão de detecção de modificações *MDP*, permitindo que um documento possa ser modificado sem alterar a validade das assinaturas existentes. Somente uma assinatura de certificação pode ser aplicada sobre o documento. As informações de como é realizada a detecção das modificações ocorrem através do preenchimento da entrada *Reference* do dicionário de assinatura.

O padrão PDF permite a inclusão de um conjunto de ações que possibilitam um comportamento inesperado do conteúdo do documento, como por exemplo a utilização de conteúdo dinâmico, em linguagem *JavaScript*. Características como esta devem ser levadas em consideração ao se aplicar uma assinatura de certificação sobre um documento assinado. Assim, é importante que a assinatura de certificação contenha um dicionário de comprovação legal *legal attestation dictionary*, que permite salientar todos os conteúdos que podem resultar em um comportamento inesperado.

5.5. Múltiplas Assinaturas

Como discutido na seção sobre atualização de um documento PDF, ao incluir algum objeto no corpo do documento, toda sua estrutura é alterada. Para que isso não ocorra deve-se fazer uso do conceito da atualização incremental através da criação de revisões. O conteúdo original irá ficar intacto, permitindo que uma nova assinatura seja adicionada sem invalidar a outra.

A inclusão de uma nova assinatura ocorre da forma tradicional, porém esta não irá assinar apenas a nova revisão, mas o documento inteiro, inclusive a assinatura anterior.

6. PDF Advanced Electronic Signatures

PDF Advanced Electronic Signatures (PAES), do European Telecommunications Standards Institute (ETSI), é uma

especificação técnica que estende e aplica restrições as assinaturas digitais PDF normatizadas pela ISO 32000-1 [13] para obterem status de assinaturas avançadas. Além disso, provê suporte para assinaturas XML e inclui características que possibilitam gerar assinaturas equivalentes as descritas nas especificações CADES [3] e XAdES [4].

Esta especificação é dividida em seis partes a qual inicia pelo *PAdES Overview* [17] considerado um documento de caracter informativo, que descreve sobre o suporte para assinatura XML e apresenta as características dos documentos(*PAdES Basic* [18], *PAdES Enhanced - PAdES-BES and PAdES-EPES Profiles* [19], *PAdES-LTV* [20], *PAdES for XML Content* [21]) descrevendo como os perfis podem ser combinados e a parte seis intitulada *Visual Representations of Electronic Signatures* [22], esta descreve requisitos e recomendações para a representação de assinaturas digitais avançadas em documentos PDF.

Nesta seção serão apresentados os principais conceitos relacionados a assinaturas CADES em documentos PDF. Sabe-se que a total compreensão da especificação PAdES é vital para a adoção deste padrão de assinatura junto ao contexto nacional, e características dos elementos XML assim como da aparência das assinaturas devem ser analisadas. Porém como o objetivo deste trabalho esta relacionada a adoção do PAdES utilizando CADES, esta seção irá descrever com mais detalhes as especificações *PAdES Basic*, *PAdES Enhanced - PAdES-BES and PAdES-EPES Profiles* e *PAdES-LTV* pois fazem referência direta a este tipo de assinatura.

6.1. PAdES Basic

A especificação técnica ETSI TS 102 778-2 [18] utiliza a assinatura baseada em CMS conforme especificada na ISO 32000-1 [13]. E tem por objetivo maximizar a interoperabilidade deste tipo de assinatura, fornecendo restrições adicionais àquelas descritas na ISO. Estas assinaturas suportam: assinaturas incrementais, opcionalmente incluem carimbo de tempo, informações revogação e um dicionário de comprovação legal e podem fazer uso dos campos (Razão, Local e Contato) do dicionário de assinatura.

Conforme exposto na ISO, a assinatura deve cobrir todo do documento, exceto a entrada *Contents* do dicionário e, a estrutura da assinatura deve ser do tipo PKCS#7 o qual irá incluir um único *SignerInfo* e no mínimo, o certificado do signatário. A entrada do dicionário de assinatura *SubFilter* deve fazer referência apenas aos valores (*adbe.pkcs7.detached* e *adbe.pkcs7.sha1*) e a especificação recomenda o uso de algoritmos mais fortes ao do tipo *SHA1*.

6.2. PAdES Enhanced - PAdES-BES and PAdES-EPES

A especificação ETSI TS 102 778-3 [19] apresenta formas de assinatura alternativas as especificadas na ISO 32000-1. Estas são mapeadas para os formatos PAdES-BES e PAdES-EPES que podem opcionalmente conter um carimbo do tempo, sendo formatos equivalentes ao CADES-BES, CADES-EPES e CADES-T. Assim, semelhante a especificação CADES [3], o formato PAdES-EPES possui política explícita e é construída sobre o PAdES-BES, porém apesar permitir o uso do carimbo do tempo não possui a nomenclatura "\-T".

A estrutura da assinatura deve estar em conformidade com a RFC 3852 [5] e ser adicionada ao documento conforme especificado na ISO 32000-1 [13] (seção 12.8.1), o *byte range* assinado é o documento inteiro excluindo a entrada *Contents* que irá conter a estrutura CADES. Como esta forma de

assinatura tem como base a tradicional, que utiliza o dicionário de assinatura. Atributos deste dicionário tem preferência a alguns utilizados no CADES para evitar redundância. Acresce que, nem todos os atributos do CADES podem ser utilizadas por este perfil, recomenda-se que atributos não assinados não descritos na especificação sejam ignorados, ao menos que outros perfis os requeiram.

6.3. PAdES Long Term - PAdES-LTV Profile

Uma assinatura digital exige que informações de validação, como a cadeia de certificação e dados de revogação, estejam disponíveis no momento da verificação. Os tipos de assinatura supracitados provêm auxílio ao armazenamento destes dados, porém o faz através dos campos *certificates [0]* e *adbe-revocationInfoArchival* da estrutura da assinatura. Como as informações de revogação desta estrutura é assinada e dada as características das assinaturas incrementais tornar as assinaturas anteriores imutáveis, caso os dados de revogação não estejam disponíveis antes da realização da assinatura esta não poderá ser realizada ou não irá conter os dados de validação. Uma alternativa à solução deste problema é através da especificação PAdES-LTV que permite que os dados de validação possam ser adicionados posteriormente a realização da assinatura sem invalidar as assinaturas incrementais caso estas tenham sido realizadas.

Isto é feito através da especificação ETSI TS 102 778-4 [20] que adiciona suporte a validação a longo prazo através do PAdES-LTV às assinaturas baseadas em CMS conforme especificado na ISO 32000-1 [13]. Ela, define como as informações de validação devem ser adicionadas ao documento PDF e posteriormente como protege-lo através da inclusão de carimbos do tempo. Além disso, esta proteção a longo prazo pode ser adicionada aos perfis PAdES Basic [18], PAdES-BES e PAdES-EPES [19]. O que permite obter funcionalidades equivalentes ao CADES-X Long e CADES-A. Acresce que, o PAdES-LTV pode também ser utilizado conjuntamente com a especificação ETSI TS 102 778-5 *PAdES for XAdES signatures* [21] o qual permite construir formas equivalentes ao XAdES-XL e XAdES-A.

Para prover tais recursos, este perfil faz uso de uma extensão ao ISO 32000-1 chamado *Document Security Store* (DSS), localizado no corpo da estrutura do PDF, que irá conter os dados necessários para a validação das assinaturas. Assim, o *Document Security Store* é um dicionário utilizado para armazenar os valores dos dados de validação. A estrutura irá incluir também os dados de validação do carimbo do tempo realizado sobre a assinatura. Além disso, o DSS não refere-se a apenas uma assinatura, caso o documento possua assinaturas incrementais, as informações de validação de todas as assinaturas serão armazenada nesta estrutura.

Para possibilitar a proteção a longo prazo do documento e das informações de validação, esta especificação apresenta uma outra extensão a ISO 32000-1 chamada *Document Time-stamp*. Este é um dicionário de assinatura padrão, semelhante ao utilizado para realizar assinaturas, porém com algumas alterações.

É recomendado que a utilização do DSS seja seguida de um *Document Time-stamp*, caso seja preciso prolongar a proteção do documento deve-se adicionar sucessivos DSS seguidos de um *Document Time-stamp*. A validação é realizada considerando a data do último carimbo para validar os mais interno.

7. Utilização do Padrão PAdES no Âmbito da ICP-Brasil

Com a revisão dos principais padrões de assinatura utilizados tanto em âmbito mundial como no cenário nacional, com o PBAD, torna-se possível avaliar os requisitos necessários para a utilização do padrão PAdES no contexto do padrão proposto pela Infraestrutura de Chaves Públicas Brasileira. Esta seção apresenta uma análise das alterações necessárias, limitações e impactos da adoção do padrão PAdES no contexto do Padrão Brasileiro de Assinatura Digital. Para tanto, serão apresentadas propostas de representação das assinaturas digitais sob cada política de assinatura definida na ICP-Brasil, utilizando os perfis *PAdES Enhanced - PAdES-BES and PAdES-EPES Profiles* [19] e *PAdES-LTV* [20].

7.1. Perfil de uso Geral

Para avaliar a possibilidade de utilização do padrão PAdES segundo os critérios da ICP-Brasil, foram então analisadas as políticas de assinaturas digitais previstas no PBAD e levantada a lista de atributos aprovados, conforme as definições do DOC-ICP-15.02 [23]. Com a lista dos atributos aprovados pela ICP-Brasil, buscou-se o mapeamento dos mesmos em atributos equivalentes, definidos no PAdES. Após construída uma tabela de equivalência, buscou-se mapear os diferentes perfis de assinatura, de acordo com cada política.

A tabela abaixo apresenta os atributos equivalentes aos atributos assinados, conforme definido na tabela 2.1 do DOC-ICP-15.02 [23]. É importante salientar que a assinatura, segundo o padrão PDF, é realizada sobre todo o documento.. Dessa forma, mesmo utilizando campos externos à estrutura CADES para garantir a integridade e autenticidade do documento, estes encontram-se também assinados. Em outras palavras, mesmo que a referida tabela aponte para atributos externos à assinatura, presentes no conteúdo do documento, tratam-se de atributos assinados.

Atributo	Ref[19]	Observações
id-aa-etscontentTimestamp	4.5.11	Utiliza-se o atributo do CADES.
id-aa-ets-signerAttr	4.5.10	Utiliza-se o atributo do CADES. Obs.: Certificado de atributo não pode ser utilizado conforme descrito em ISO 32000-1 Seção 12.8.3.3.1.
id-aa-ets-signerLocation	4.5.9	Deve ser utilizado o campo "Location" do dicionário de assinatura especificado em ISO 32000-1 (Standard 2008) Seção 12.8.1.
id-signingTime	4.5.3	Deve ser utilizado o campo "M" do dicionário de assinatura especificado em ISO 32000-1 Seção 12.8.1.
id-contentType	4.4.1	Utiliza-se o atributo do CADES. Obs.: Somente o valor id-data pode ser usado.
id-messageDigest	4.4.2	Utiliza-se o atributo do CADES.
id-aasigninCertificate(V2)	4.4.3	Utiliza-se o atributo do CADES.
id-aa-ets-sigPolicyId	4.5.1	Utiliza-se o atributo do CADES.

Além dos atributos assinados definidos pela ICP-Brasil, para estar em conformidade com a especificação PAdES seriam ainda necessários definir os seguintes atributos:

Atributo	Ref[19]	Observações
----------	---------	-------------

SubFilter	4.2 (e)	O valor da chave SubFilter do dicionário de assinatura especificado em ISO 32000-1 Seção 12.8.1 deve ser <i>ETSI.CADES.detached</i> .
Extensions Dictionary	4.7 (e)	Extensão incluída no dicionário catalogo do documento.

7.2. Políticas de Assinatura

7.2.1. AD-RB

O dicionário de assinatura contém, no mínimo, a estrutura da assinatura no formato CADES e uma indicação da utilização deste formato. A estrutura CADES deve estar codificada em hexadecimal e armazenada na entrada *Contents*, já o nome que indica que a estrutura CADES esta sendo utilizada deve estar na entrada *SubFilter* e possuir o valor *ETSI.CADES.detached*. A estrutura deve possuir o certificado do signatário armazenado no campo *Certificates [0]* e os atributos assinados: *id-contentType*, *id-messageDigest*, *id-aa-signingCertificateV2* e *id-aa-ets-sigPolicyId*.

O dicionário de catalogo do documento deve indicar que o arquivo PDF contém características definidas na especificação [19], através da inclusão do valor `<</ESIC<</BaseVersion/1.7/ExtensionLevel 2>>>>` na entrada *Extension*.

7.2.2. AD-RT

A assinatura digital com referência do tempo é construída sobre a básica. Assim, este perfil detém todas as características supracitadas. Além disso, utiliza o *id-aa-signatureTimeStampToken* como um atributo não assinado da estrutura CADES, o qual irá conter o carimbo do tempo sobre a assinatura.

7.2.3. AD-RV

Uma assinatura AD-RV é semelhante à assinatura CADES-C, no sentido de que ambas agregam, à assinatura com carimbo de tempo, referências para as estruturas necessárias à validação. Em ambas as assinaturas, tais referências são inseridas através dos atributos não assinados *Complete Certificate References* e *Complete Revocation References*. Para construir uma assinatura digital equivalente em PAdES, seria necessário adicionar as referências supracitadas na estrutura CADES, interna ao documento. Contudo, essa abordagem é desencorajada pela própria especificação, dada a natureza incremental da assinatura PAdES, conforme nota explicativa número 1 do Anexo B, no documento ETSI TS 102 778-4 [20]. O problema está relacionado à manutenção da assinatura com referências, sobretudo com relação ao *Grace Period*. É frequente a necessidade de atualização das referências de validação ou mesmo a inserção de tais referências em momento posterior à assinatura, pelo verificador, conforme previsto pela política AD-RV, na seção 5.2.1.2 do DOC-ICP-15.03 [24]. Como na assinatura PDF a estrutura CADES torna-se imutável após a aplicação de uma segunda assinatura, ficaria inviável a atualização posterior dos atributos não assinados, que de fato passariam a estar assinados no contexto do padrão PAdES.

A especificação do PAdES de fato desencoraja o uso de atributos não assinados, salvo quando expressamente descritos no perfil, tal como exposto no item h, seção 4.2, do documento ETSI TS 102 778-3 [19]. Sendo assim, não é possível representar uma assinatura AD-RV segundo o padrão PAdES.

7.2.4. AD-RC

A assinatura equivalente à política AD-RC pode ser obtida da

utilização da estrutura `{\it Long Term Validation - LTV}`, para armazenar os valores dos dados de revogação. A seção subsequente apresenta esta abordagem.

Segundo essa alternativa, o *Document Security Store (DSS)* é utilizado para armazenar os valores dos dados de validação. A estrutura irá incluir também os dados de validação do carimbo do tempo realizado sobre a assinatura.

Posteriormente, deve-se aplicar um carimbo de tempo sobre o documento *Document Timestamp* que, como na abordagem anterior, irá também amarrar os dados de validação.

Caso assinaturas incrementais sejam aplicadas ao documento antes da inclusão do DSS, estas permanecerão íntegras. Caso adicionadas após, as informações de validação permanecerão imutáveis, dado as características das assinaturas incrementais.

7.2.5. AD-RA

A assinatura digital com referências para arquivamento deve ser construída sobre o perfil AD-RC e conter uma restrição à inserção de novas assinaturas ao documento, ou seja, deve ser uma assinatura de certificação. Nas assinaturas PBAD-CADES, esta restrição é imposta através da aposição de um carimbo do tempo de arquivamento, encapsulando todo o conteúdo assinado. Já no padrão PDF, tal bloqueio é realizado através da utilização do método de transformação DocMDP (*Modification, Detection and Prevention*) onde a chave *P* deve conter o valor que indica que o documento não pode sofrer alterações. Apesar desta restrição não permitir que o documento seja alterado, o leitor PDF deve permitir as modificações referentes à inclusão do *Document Security Store (DSS)* e *Document Security Timestamp*.

7.3. Impactos e Mudanças na Normatização do PBAD

Os tipos de assinaturas utilizados no contexto nacional permitem o uso de coassinaturas, contra-assinaturas e assinaturas com documento anexado. Primeiramente, a especificação PAdES não permite a realização de assinaturas separadas do documento. De fato as assinaturas PAdES encontram-se incorporadas ao conteúdo assinado. Embora esta seja a característica mais marcante do padrão PDF e certamente uma das principais responsáveis por sua aceitação, a forma como foi definida a alocação do campo destinado à assinatura, em área limitada interna ao documento, acaba por trazer sérias implicações para a manutenção das características de segurança necessárias à manutenção do documento eletrônico por longo prazo. O principal impacto desta abordagem é a necessidade de assinaturas incrementais, operando na forma de revisões, em que uma assinatura é realizada sobre todo o conteúdo do documento mais as assinaturas anteriores. Dessa forma, o PAdES não permite a utilização de múltiplas assinaturas em uma mesma estrutura CMS, não suportando, por consequência, a realização de co e contra assinaturas.

A inserção de um novo contexto relacionado às assinaturas, as incrementais, deverá implicar na revisão dos referidos conceitos, no conjunto normativo da ICP-Brasil. O documento DOC-ICP-15 [26] e seus derivados, documentos que definem o uso de assinaturas digitais no âmbito da ICP-Brasil, precisarão ser adaptados ao uso do PAdES. Um exemplo é a seção 6.8 do documento principal, que limita os contextos de assinatura aos três atualmente existentes no âmbito do CADES e XAdES: assinatura, co e contra-assinaturas.

O procedimento de verificação de assinaturas incrementais,

quando comparado ao processo de verificação de múltiplas assinaturas em uma estrutura CMS, exige um maior esforço computacional. Enquanto o cálculo do resumo criptográfico do documento assinado pode ser feito uma única vez, nas co assinaturas CADES. Já as assinaturas incrementais exigem o recálculo do resumo criptográfico do conteúdo assinado a cada nova assinatura, onerando o processo de acordo com o tamanho do conteúdo de cada revisão.

Outro ponto a ser considerado é a possibilidade do documento possuir conteúdos dinâmicos, que provoquem um comportamento inesperado, como com a utilização de rotinas em linguagem *JavaScript* para modificar o conteúdo do texto em tempo de exibição. Caso o documento possua tais elementos, deve-se primeiramente realizar uma assinatura de certificação, com vista a evidenciar a existência de quaisquer elementos que representem risco ao verificador.

Um outro problema seria o formato das políticas: ASN.1 para o CADES referencia os atributos com OID e as XML para o XAdES com URI. Seria necessário um novo padrão de políticas para o PDF para, por exemplo, obrigar a inclusão das extensões e subfilter.

8. Protótipo Assinador PBAD-PAdES

Para comprovar a viabilidade da confecção dos perfis propostos, foi criado o *Assinador PBAD-PAdES* utilizando a linguagem de programação *Java*. Este assinador faz uso biblioteca *iText* [27] para manipular a estrutura do PDF e a *Bouncy Castle* [28] para efetuar o processo de assinatura digital.

8.1. Assinatura

O Assinador PBAD-PAdES permite realizar assinaturas com referência básica (AD-RB) e explorar a utilização do *Documento Security Store (DSS)* o qual irá permitir criar os outros tipos de assinatura. Além disso, através deste software, pode-se verificar a estrutura da assinatura gerada e criar múltiplas assinaturas.

A estrutura da assinatura gerada contém uma política, o campo *SiginingCertificateV2*, e não dispõe do *signingTime* como atributo assinado anexado a estrutura. Este, conforme a especificação PAdES recomenda, foi adicionado no dicionário de assinatura assim como o tipo de codificação, o local e o motivo. Contém as entradas: `/SubFilter/ETSI.CADES.detached` `/Location(Local Assinatura)` `/M(D:20120515031156-03'00')` `/Reason(Razão Assinatura)}`. A extensão *ESIC ExtensionLevel 2* foi incluída no dicionário *catalogo do documento*.

8.2. Verificação

A processo de verificação ocorre através do botão "*Verificar*" que valida a integridade do documento, a política de assinatura utilizada, o atributo *SiginingCertificateV2* e permite extrair detalhes sobre as atualizações incrementais.

8.3. Compatibilidade com Adobe Acrobat

O Adobe Acrobat [28] constitui uma família de aplicativos produzida pela Adobe Systems. Dois *softwares* muito utilizados desta família é o *Acrobat Reader* e *Adobe Acrobat Pro*, ambos atualmente na versão X. O primeiro é totalmente gratuito e contém um conjunto variado de funcionalidades, onde é possível navegar, visualizar e imprimir arquivos PDF. Já o segundo provê um conjunto maior de funcionalidades, porém sua licença não é gratuita e o período para testes é de trinta dias.

8.3.1 Geração de Assinaturas

O processo para geração de assinatura é muito simples, adiciona-se sobre página um campo que irá exibir os detalhes da assinatura. Estes detalhes constituem a aparência da assinatura e são altamente configuráveis. Uma outra opção disponibilizada pela ferramenta Adobe é a possibilidade de após a realização da assinatura, não permitir que nenhuma modificação seja feita no documento, através do *DocMDP*.

8.3.2 Assinaturas PKCS#7 e Equivalente ao CADES

O *Adobe Acrobat Pro* além de permitir realizar a assinatura tradicional também possui suporte para geração de assinaturas PAdES, e este é ativado após uma pequena configuração. Basta clicar em: *Editar/Preferências...*, acessar o item "*Segurança*" clicar em "*Preferências Avançadas...*" e localizar a aba "*Criação*".

A assinatura gerada possui como *SubFilter* o valor *ETSI.CADES.detached* e os dados da entrada *Contents* após convertidos para binário mostraram que a estrutura da assinatura CADES continha toda a cadeia de certificação no campo *Certificates[0]* e os dados do signatário no atributo assinado *Signing Certificate V2* e as informações de revogação no atributo assinado *adbe-revocationInfoArchival*.

O problema ao incluir os dados de validação na estrutura da assinatura, está diretamente relacionado ao *Grace Period*, que diz respeito a obtenção dos melhores dados de revogação, outro ponto importante é que estes dados devem estar disponíveis antes da geração da assinatura. Assim, o *Acrobat* disponibiliza a opção de incluir os dados de revogação após o momento da assinatura [29] e utiliza o *Document Security Store (DSS)* para adicionar os dados de validação. Deve-se realizar uma assinatura sem a inclusão dos dados de validação, e posteriormente clicar com o botão direito do mouse sobre a assinatura e selecionar "Adicionar informações de verificação". Desse modo, conclui-se que o tipo de assinatura implementada no *Adobe Acrobat Pro* é equivalente ao *CADES-BES* e este possui suporte para a estrutura *DSS* do *PAdES-LTV*.

8.3.3. Verificação da Assinaturas

Após gerar a assinatura com o software *Assinador PBAD-PAdES*, sem inserir as informações de validação, foi realizada uma tentativa de verificação com o *Adobe Reader*. Este informou que o caminho de certificação não pode ser verificado. Para resolver o problema, foi preciso configurá-lo incluindo o certificado raiz no repositório do windows e informando que este deve ser utilizado para validar a autenticidade do signatário. A configuração ocorreu da seguinte maneira: deve-se clicar em *Editar/Preferências...*, acessar o item "*Segurança*" clicar em "*Preferências Avançadas...*" e localizar a aba "*Integração com o Windows*".

Ao abrir o leitor novamente, a assinatura foi verificada com sucesso.

Foi gerada uma nova assinatura, e removida a lista de certificados revogados. Ao utilizar o *Adobe Reader* o mesmo exibiu a mensagem: "A assinatura é válida, mas o cancelamento da identidade do assinante não pôde ser verificado".

Utilizou-se o *Assinador PBAD-PAdES* para incluir, somente, a lista de certificados revogados na estrutura *Document Security Store DSS*. Novamente, fez-se uso do *Adobe Reader* para

verificar a documento. Ele validou com sucesso. Ou seja, o *Adobe Reader* utilizou a lista de certificados revogados que foi armazenada no *Document Security Store DSS*. Desse modo, conclui-se que o tipo de verificação de assinatura implementada no *Adobe Reader* é equivalente ao *CADES-BES* e este possui suporte para a estrutura *DSS* do *PAdES-LTV*. O mesmo comportamento ocorreu com o *Acrobat Pro*.

Para permitir a verificação de assinaturas com política explícita deve-se desenvolver um plug-in para o *Acrobat Reader*. Uma opção seria através do uso do *Adobe Acrobat X SDK* [30]. Porém, as aplicações devem estar em conformidade com a licença (*Adobe Reader Integration Key License (RIKLA) Program*) [31] que faz necessário obter uma chave para integração com o *Reader*, esta integração exige um contrato anual. A aplicação depois de confeccionada necessita ser aprovada, o que de acordo com o *RIKLA* demora cerca de duas semanas.

9. Conclusões

A utilização do formato de representação de documentos (*PDF*) associado a eficácia probante da Infraestrutura de Chaves Públicas Brasileira (*ICP-Brasil*) trará grandes avanços sobre aspectos da utilização da assinatura digital no Brasil. Assim, este trabalho apresentou uma análise sobre soluções e impactos no processo de adaptação do padrão de assinatura digital PAdES aos moldes da *ICP-Brasil*, avaliando as relações com as políticas assinaturas atualmente descritas sendo consideradas apenas as assinaturas CADES. Apesar deste trabalho ter íntima ligação com aspectos de políticas de assinatura, este trabalho apresentou apenas uma proposta para os modelos de assinatura tratados pela *ICP-Brasil*, não definindo formatos de políticas de assinatura PAdES e nem formas de codificação de políticas para este tipo de assinatura, estes tratados como trabalhos futuros.

10. Referências

- [1] BRASIL. Projeto de Lei n.º 4.906, de 21 de junho de 2001. Dispõe sobre o valor probante do documento eletrônico e da assinatura digital, regula a certificação digital, institui normas para as transações de comércio eletrônico e dá outras providências. [S.l.]: Diário Oficial da União, Brasília, DF, 21 jun. de 2001.
- [2] BUCKLAND, M. K. What is a "document"? JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE, 1997.
- [3] INSTITUTE, E. T. S. Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CADES). [S.l.]: ETSI, 2010. ETSI TS 101 733 V1.8.3 (2011-01).
- [4] INSTITUTE, E. T. S. Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES). [S.l.]: ETSI, 2010. ETSI TS 101 903 V1.4.2 (2010-12).
- [5] HOUSLEY, R. Cryptographic Message Syntax (CMS). [S.l.]: IETF, 2004. RFC 3852 (Proposed Standard). (Request for Comments, 3852).
- [6] DIFFIE, W.; HELLMAN, M. E. New directions in cryptography. IEEE Transactions on Information Theory, Invited Paper, S.L, n. 6, p.1–12, 1976.
- [7] NIST. Digital Signature Standard. [S.l.], May 1994.
- [8] ADAMS, C.; CAIN, P.; PINKAS, D.; ZUCCHERATO, R. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). [S.l.]: IETF, 2008. RFC 3161 (Proposed Standard). (Request for Comments, 3161).
- [9] INSTITUTE, E. T. S. Technical Specification, Electronic Signatures and Infrastructures (ESI); ASN.1 format for

- signature policies. December 2003.
- [10] STEVE, L.; PKI, F. Understanding Certification Path Construction. [S.l.]: PKI Forum, 2002. White Paper.
- [11] COOPER, D.; SANTESSON, S.; FARRELL, S.; BOEYEN, S.; HOUSLEY, R.; POLK, W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. [S.l.]: IETF, maio 2008. RFC 5280 (Proposed Standard). (Request for Comments, 5280).
- [12] 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. [S.l.]: Diário Oficial da União, Brasília, DF, 29 jun. de 2001.
- [13] STANDARD, I. Document management - Portable document format -Part 1: PDF 1.7. [S.l.]: ISO, 2008. 756 p. ISO 32000-1.
- [14] INCORPORATED, A. S. PostScript Language Reference Manual. 3rd. ed. [S.l.]: ADOBE, 1999. 912 p. Reference Manual. ISBN 0-201-37922-8.
- [15] TECHNICAL, R. L. PKCS #1: RSA Cryptography Standard v2.1. [S.l.], 2002.
- [16] TECHNICAL, R. L. PCKCS #7 - Cryptographic Message Syntax Standard. [S.l.], November 1993.
- [17] INSTITUTE, E. T. S. Technical Specification, ETSI TS 102 778-1 V1.1.1 Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 1: PAdES Overview - a framework document for PAdES. July 2009.
- [18] INSTITUTE, E. T. S. Technical Specification, ETSI TS 102 778-2 V1.2.1 Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 1: PAdES Overview - a framework document for PAdES. July 2009.
- [19] INSTITUTE, E. T. S. Technical Specification, ETSI TS 102 778-3 v1.2.1 Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 3: PAdES Enhanced - PAdES-BES and PAdES-EPES Profiles. July 2010.
- [20] INSTITUTE, E. T. S. Technical Specification, ETSI TS 102 778-4 V1.1.2 Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 4: PAdES Long Term - PAdES-LTV Profile. July 2009.
- [21] INSTITUTE, E. T. S. Technical Specification, Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 5: PAdES for XML Content - Profiles for XAdES signatures. December 2009.
- [22] INSTITUTE, E. T. S. Technical Specification, Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 6: Visual Representations of Electronic Signatures. December 2010.
- [23] ITI ITI. PERFIL DE USO GERAL PARA ASSINATURAS DIGITAIS NA ICP-BRASIL. Versão 2.0. Brasília, Abril 2010. DOC-ICP-15.02. [S.l.].
- [24] ITI ITI. REQUISITOS DAS POLÍTICAS DE ASSINATURA DIGITAL NA ICP-BRASIL. Versão 5.0. Brasília, Março 2012. DOC-ICP-15.03. [S.l.].
- [25] ITI ITI. REQUISITOS PARA GERAÇÃO E VERIFICAÇÃO DE ASSINATURAS DIGITAIS NA ICP-BRASIL. Versão 2.0. Brasília, Abril 2010. DOC-ICP-15.01. [S.l.].
- [26] ITI ITI. VISÃO GERAL SOBRE ASSINATURAS DIGITAIS NA ICP-BRASIL. Versão 2.0. Brasília, Abril 2010. DOC-ICP-15. [S.l.].
- [27] CORP. iText S. iText. Disponível em: <<http://itextpdf.com/>>. Acessado em: 15 de maio de 2012.
- [28] SYSTEMS, A. Adobe Acrobat. Disponível em: <<http://www.adobe.com/products/acrobat.html>>. Acessado em: 15 de maio de 2012.
- [29] SYSTEMS, A. Establish long-term signature validation. Disponível em: <http://help.adobe.com/en_US/acrobat/pro/using/WS934c23d7cc8877da1172e0811fde233c98-8000.html#WS934c23d7cc8877da1172e0811fde3067ed-7fff>. Acessado em: 15 de maio de 2012.
- [30] SYSTEMS, A. Acrobat Developer Center. Disponível em: <<http://www.adobe.com/devnet/acrobat.html>>. Acessado em: 16 de maio de 2012.
- [31] SYSTEMS, A. Adobe Reader Integration Key License (RIKLA) Program. Disponível em: <<http://www.adobe.com/devnet/reader/ikla.html>>. Acessado em: 16 de maio de 2012.