UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Lucas Silveira

IMPLEMENTAÇÃO DO PADRÃO BRASILEIRO DE ASSINATURA DIGITAL

Florianópolis(SC)

2011

Lucas Silveira

IMPLEMENTAÇÃO DO PADRÃO BRASILEIRO DE ASSINATURA DIGITAL

Trabalho de Conclusão de Curso submetido ao Curso de Sistemas de Informação para a obtenção do Grau de Bacharel em Sistemas de Informação.
Orientador: Felipe Carlos Werlang
Coorientador: Prof. Ricardo Felipe Custódio, Dr.

Florianópolis(SC)

2011

Lucas Silveira

IMPLEMENTAÇÃO DO PADRÃO BRASILEIRO DE ASSINATURA DIGITAL

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de "Bacharel em Sistemas de Informação", e aprovado em sua forma final pelo Curso de Sistemas de Informação.

Florianópolis(SC), 19 de junho 2011.

Prof. Leandro José Komosinski, Dr.
Coordenador do Curso

Felipe Carlos Werlang
Orientador

Prof. Ricardo Felipe Custódio, Dr.
Coorientador

Banca Examinadora:

Lucas Ferraro

Thiago Acórdi Ramos, Msc.

Manuel Matos

Dedico esse trabalho a minha família, que me fornece toda a educação e amor necessário para me tornar mais forte na busca incansável de alcançar meus objetivos.
Pai, mãe, e irmã. Agradeço a vocês por serem compreensíveis e estarem ao meu lado em todos os momentos, me apoiando, incentivando, e ensinando em todos os aspectos da vida. Sem vocês, nada disso seria possível.

AGRADECIMENTOS

Em primeiro lugar, agradeço a meus pais que me deram a estrutura necessária e a oportunidade de ingressar e concluir um curso universitário com todo apoio que necessitei.

Agradeço também meus companheiros do LabSEC pela ajuda e amizade prestada em todos os momentos. Em especial os amigos Martín Augusto Gagliotti Vigil, que me ajudou e ensinou muito desde os meus primeiros passos dentro do LabSEC até hoje. Felipe Carlos Werlang, Thiago Acórdi Ramos e Lucas Ferraro, obrigado por toda ajuda e orientação no desenvolvimento deste trabalho.

Agradeço ao Professor Custódio pela oportunidade, confiança, e orientação nesses três anos de trabalho no LabSEC.

Obrigado ao ITI e ao CNB pelo apoio e confiança no trabalho. Meus agradecimentos também se estendem ao Nelson da Silva, que cedeu boa parte do código utilizado na carimbadora de testes.

Obrigado.

“Aprender é a única coisa de que a mente nunca se cansa, nunca tem medo e nunca se arrepende.”

Leonardo da Vinci

RESUMO

O presente trabalho visa a implementação dos códigos de referência do Padrão Brasileiro de Assinatura Digital (PBAD), especificado no conjunto normativo DOC-ICP-15 (BRASIL, 2010d). Este desenvolvimento engloba a criação de uma biblioteca de assinatura digital e um assinador minimalista de referência, os quais formam as estruturas principais dos códigos de referência. Ao longo do trabalho é apresentado a estrutura, organização, metodologia, desenvolvimento das principais funcionalidades, e os resultados obtidos ao final do projeto da biblioteca de referência e do assinador minimalista de referência.

Palavras-chave: Assinatura Digital, CAdES, ICP-Brasil, Padrão Brasileiro de Assinatura Digital, Política de Assinatura, XAdES.

LISTA DE FIGURAS

Figura 1 Assinatura Anexada 29
Figura 2 Assinatura Destacada 29
Figura 3 Assinatura Encapsuladoras 30
Figura 4 Assinatura Embarcada 30
Figura 5 Diagrama de classes - Estrutura integrada não dinâmica 45
Figura 6 Principais classes da estrutura integrada dinâmica 49
Figura 7 Geração de Assinatura 52
Figura 8 Verificação de Assinatura 53

LISTA DE ABREVIATURAS E SIGLAS

AC	Autoridade Certificadora
ACT	Autoridade de Carimbo do Tempo
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
ASN.1	*Abstract Syntax Notation One*
CAdES	*CMS Advanced Eletronic Signatures*
CMS	*Cryptographic Message Syntax*
CNB	*Colégio Notarial do Brasil*
ECC	*Elliptic Curve Cryptosystem*
ETSI	*European Telecommunications Standards Institute*
ICP	Infraestrutura de Chaves Públicas
INE	Departamento de Informática e Estatística
ITI	Instituto Nacional de Tecnologia da Informação
LabSEC	Laboratório de Segurança em Computação - INE - UFSC
LCR	Lista de Certificados Revogados
LPA	Lista de Políticas Aprovadas
OCSP	Online Certificate Status Protocol
PA	Política de Assinatura
PBAD	Padrão Brasileiro de Assinatura Digital
RSA	*Rivest-Shamir-Adleman*
RFC	*Request for Comments*
SHA	*Secure Hash Algorithm*
SVN	*Subversion*
UFSC	Universidade Federal de Santa Catarina
UML	*Unified Modeling Language*
XAdES	*XML Advanced Eletronic Signatures*
XML	*Extensible Markup Language*
XMLDsig	*XML Digital Signature*

SUMÁRIO

1 INTRODUÇÃO 21
1.1 OBJETIVOS 22
1.1.1 Geral 22
1.1.2 Específico 22
1.2 JUSTIFICATIVA 22
1.3 METODOLOGIA 23
2 FUNDAMENTAÇÃO TEÓRICA 25
2.1 CRIPTOGRAFIA 25
2.1.1 Criptografia Assimétrica 26
2.2 RESUMO CRIPTOGRÁFICO 27
2.3 ASSINATURA DIGITAL 27
2.3.1 Co-assinatura 28
2.3.2 Contra-assinatura 28
2.3.3 CAdES 28
2.3.4 XAdES 29
2.3.5 Política de Assinatura 30
2.4 CARIMBO DO TEMPO 31
2.5 INFRAESTRUTURA DE CHAVES PÚBLICAS 31
2.5.1 Certificado Digital 32
2.5.2 Lista de Certificados Revogados 32
2.5.3 Online Certificate Status Protocol 33
2.5.4 Infraestrutura de Chaves Públicas Brasileira 33
2.6 PADRÃO BRASILEIRO DE ASSINATURA DIGITAL 34
3 CÓDIGOS DE REFERÊNCIA 37
3.1 BIBLIOTECA DE REFERÊNCIA 37
3.2 ASSINADOR MINIMALISTA DE REFERÊNCIA 38
3.3 GERENCIADOR DE POLÍTICAS 39
3.4 CARIMBADORA DE TESTES 39
3.5 PRINCIPAIS TECNOLOGIAS EMPREGADAS 40
3.5.1 Linguagem de Programação Java 40
3.5.2 Trac 40
3.5.3 Subversion (SVN) 41
3.5.4 BouncyCastle 41
3.6 PROCESSO DE DESENVOLVIMENTO 41
4 ESTRUTURA DE INTEGRAÇÃO CADES E XADES 43
4.1 IMPLEMENTAÇÃO PERFIL RB E RT 43
4.2 INTEGRAÇÃO CADES E XADES 44

4.2.1 Implementação integrada não dinâmica 44
4.3 IMPLEMENTAÇÃO INTEGRADA DINÂMICA 46
4.3.1 Leitura dinâmica de Políticas de Assinatura 46
4.3.2 Uso de interfaces 47
4.3.3 GenericEncoding 47
4.3.4 Uso de Reflexão Java 48
5 FUNCIONALIDADES FUNDAMENTAIS 51
5.1 GERAÇÃO DE ASSINATURA 51
5.2 VERIFICAÇÃO DE ASSINATURA 52
5.3 ATRIBUTOS DE ASSINATURA 54
5.4 CONTRA-ASSINATURA 55
5.5 CO-ASSINATURA 56
6 CONSIDERAÇÕES FINAIS 57
6.1 TRABALHOS FUTUROS 58
Referências Bibliográficas 59

1 INTRODUÇÃO

Com a implantação da Medida Provisória 2200-2 (BRASIL, 2001), de 24 de agosto de 2001, o Brasil instituiu a Infra-Estrutura de Chaves Públicas Brasileira (ICP-Brasil). O objetivo dessa Medida Provisória é garantir autenticidade, integridade, e validade jurídica a documentos eletrônicos, impulsionando e estimulando o uso de certificação digital no país, e consequentemente, a utilização de assinaturas digitais em âmbito governamental e privado. (BRASIL, 2001)

O fato de diferentes instituições poderem utilizar diferentes tipos de softwares para a criação e validação de assinaturas digitais, pode acarretar problemas relativos a interoperabilidade desses sistemas. Assinaturas digitais produzidas em um software "A", não necessariamente serão possíveis de verificação em um software "B", e vice-versa. Isso acontece devido a falta de padronização dos sistemas na geração e verificação das assinaturas.

Para solucionar o problema, a ICP-Brasil, através de uma resolução de seu comitê gestor, regulamentou o DOC-ICP-15 (BRASIL, 2010d) e mais três instruções normativas do Instituto Nacional de Tecnologia da Informação (ITI). São elas: DOC-ICP-15.01 (BRASIL, 2010c), DOC-ICP-15.02 (BRASIL, 2010a), e DOC-ICP-15.03 (BRASIL, 2010b). O objetivo desses documentos é estabelecer a padronização na geração e na verificação das assinaturas digitais no âmbito da ICP-Brasil.

Esses documentos especificam os perfis, formatos e algoritmos que devem ser utilizados para uma assinatura digital estar de acordo com o Padrão Brasileiro de Assinatura Digital (PBAD). As especificações são feitas através da utilização de políticas de assinatura, seguindo o padrão europeu de assinatura digital, definido pelo *European Telecommunications Standards Institute* (ETSI).

O Laboratório de Segurança em Computação da Universidade Federal de Santa Catarina (LabSEC) ficou responsável pela implementação desse padrão, com o objetivo de desenvolver os códigos para posterior divulgação, através do financiamento do Colégio Notarial do Brasil (CNB).

1.1 OBJETIVOS

1.1.1 Geral

Implementar os códigos de referência do padrão brasileiro de assinatura digital.

1.1.2 Específico

- Implementar os artefatos de Políticas de Assinatura (PA);
- Implementar os artefatos da Lista Políticas de Aprovadas (LPA);
- Suportar todas as 10 políticas de assinatura existentes hoje no PBAD;
- Implementar os atributos/propriedades de assinatura assinados e não assinados definidos no PBAD;
- Implementar uma Carimbadora de Testes;
- Implementar componentes do Assinador Minimalista de Referência;
- Implementar componentes da Biblioteca de Referência.

1.2 JUSTIFICATIVA

Um dos problemas hoje enfrentados na utilização de assinaturas digitais no Brasil é a falta de sistemas que gerem assinaturas padronizadas. Isto limita em muitos casos a validade da assinatura. Com a definição do PBAD, esse problema pode ser resolvido.

Para que o PBAD se torne realidade, é fundamental a realização do projeto Códigos de Referência. Vista que o desenvolvimento desse tornará possível encontrar/descobrir possíveis problemas existentes no padrão e corrigi-los para tornar o PBAD uma realidade.

Além disso, a biblioteca de referência de assinatura digital permitirá que qualquer entidade, pública ou privada, que deseja desenvolver um assinador de acordo com o PBAD possa fazê-lo com maior facilidade. Será desenvolvido um assinador digital de referência, que terá por objetivo demonstrar a utilização da biblioteca de referência, e poderá ser utilizado livremente por toda sociedade brasileira para gerar e verificar assinaturas no contexto do PBAD.

1.3 METODOLOGIA

O desenvolvimento deste trabalho ocorreu no Laboratório de Segurança em Computação da Universidade Federal de Santa Catarina (LabSEC) e contou com as etapas de desenvolvimento apresentadas abaixo:

- Estudo dos formatos de assinatura CAdES e XAdES;
- Estudo do conjunto normativo da ICP-Brasil DOC-ICP-15;
- Projeto da biblioteca;
- Implementação do perfil RB e RT;
- Implementação da estrutura integrada não dinâmica;
- Implementação da estrutura integrada dinâmica;
- Implementação do suporte a contra-assinatura e co-assinatura;
- Implementação dos atributos/propriedades de assinatura CAdES e XAdES;
- Desenvolvimento de uma Carimbadora de Testes;
- Desenvolvimento de um Gerenciador de Políticas;
- Desenvolvimento do assinador minimalista de referência;

As ferramentas utilizadas para realização do projeto foram:

- Ambiente de Programação: Eclipse;
- Sistema Operacional: Linux (Ubuntu);
- Software de Gerenciamento de Projeto: Trac;
- Ferramenta para suporte UML: Visual Paradigm;
- Ferramenta para visualização de assinatura no formato CMS: dumpasn1;
- Ferramenta para controle de versão: Subversion (SVN);
- Biblioteca java para criação/verificação de assinatura: BouncyCastle;
- Biblioteca para utilização de serviços de infraestrutura de chaves públicas: OpenSSL.

Para o gerenciamento do projeto foram realizadas reuniões diárias entre o time de desenvolvimento. Nessas reuniões eram abordadas as tarefas e dificuldades encontradas no dia anterior, e as tarefas que seriam realizadas no dia corrente, objetivando sempre transparência e constante evolução no planejamento e desenvolvimento das tarefas.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 CRIPTOGRAFIA

É o método de armazenamento e transmissão de dados de uma forma que apenas as partes destinadas possam ler e processar. É considerada a ciência da proteção da informação através da codificação da escrita em um formato ilegível. Criptografia é uma forma efetiva de proteger informações sensíveis através do armazenamento em mídias e da transmissão de dados através de redes de comunicação não confiáveis (HARRIS, 2010).

A criptografia tem raízes por volta de 2000 A.C. no Egito, quando hieróglifos eram usados para decorar túmulos para contar as histórias de vida de um falecido. O objetivo dessa prática não era esconder as mensagens em si, mas sim fazer as histórias de vida dos mortos parecerem mais nobres e majestosas (HARRIS, 2010).

Os métodos de criptografia evoluíram e passaram a ser utilizados com a intenção de ocultar informações de terceiros. Esta evolução contou com a colaboração de alguns povos como os hebreus e espartanos. Mais tarde, em Roma, Júlio César (100-44 A.C.) desenvolveu um método simples de substituição de letras de um alfabeto. Ele simplesmente substituía as letras do alfabeto em três posições, como demonstrado no exemplo abaixo:

• **Alfabeto padrão:**
ABCDEFGHIJKLMNOPQRSTUVWXYZ
• **Alfabeto criptográfico:**
DEFGHIJKLMNOPQRSTUVWXYZABC (HARRIS, 2010).

Durante a Segunda Guerra Mundial, os dispositivos de criptografia foram utilizados para comunicações táticas, com um grande melhoramento devido as tecnologias mecânicas e eletromecânicas providas pelo telégrafo e a comunicação por rádio (HARRIS, 2010).

Segundo Menezes, Oorschot e Vanstone (2001) os objetivos da criptografia são garantir os seguintes serviços:

• **Privacidade ou Confidencialidade:** manter o conteúdo da informação oculto de todos, menos aqueles que são autorizados;

• **Integridade:** garantir que os dados não foram alterados sem autorização;

• **Autenticação:** garantir a identificação das partes envolvidas;

• **Não-repúdio:** prevenir que entidades neguem compromissos ou

ações anteriores.

2.1.1 Criptografia Assimétrica

Criptografia assimétrica, ou criptografia de chave pública, pode ser considerada uma tecnologia nova, tendo a ideia inicial introduzida por Whitfield Diffie, Martin Hellman e Ralph Merkle no ano de 1976 (PAAR; PELZL; PRENEEL, 2010).

Neste sistema de criptografia existe um par de chaves formado por uma chave pública e uma chave privada, podendo a pública ser de conhecimento de todos. A chave privada deve ser conhecida e utilizada apenas pelo seu proprietário. Normalmente as chaves públicas são mantidas em diretórios e bases de dados públicas para que qualquer pessoa tenha acesso a esta informação, facilitando assim a cifragem e decifragem na troca de mensagens (HARRIS, 2010).

Existe uma relação matemática entre as chaves pública e privada, porém qualquer pessoa que obtenha uma chave pública não é capaz de deduzir a chave privada correspondente. Assim é possível dizer que um atacante que detenha a chave pública de qualquer pessoa não será capaz de obter a chave privada da mesma pessoa através de funções matemáticas (HARRIS, 2010).

É possível utilizar tanto uma chave privada quanto uma chave pública para criptografar uma informação, o importante é saber que para se decifrar a mensagem sempre se deve utilizar a chave oposta a qual foi utilizada no processo de cifragem. Por exemplo, se Bob cifra uma mensagem com sua chave privada, Alice só conseguirá decifrá-la utilizando a chave pública de Bob e vice-versa.

A escolha da chave para cifragem irá depender do requisito de segurança que o emissor está empregando. Se o serviço de confidencialidade é mais importante a chave que deve ser utilizada para cifragem é a pública. Assim, só o destinatário possui a chave privada correspondente, e somente esse conseguirá decifrar a mensagem. Já se o serviço de segurança prioritário é a autenticidade, o emissor deve cifrar a mensagem com sua chave privada. Com isso, qualquer pessoa que utilizar sua chave pública conseguirá decodificar a mensagem, provando assim que o emissor é autêntico. Isso acontece pelo simples motivo de que só o detentor daquela chave privada seria capaz de cifrar uma mensagem decodificada pela respectiva chave pública.

Alguns exemplos de algoritmos de chave assimétrica segundo Harris (2010):

- RSA (Rivest-Shamir-Adleman)
- Elliptic curve cryptosystem (ECC)

- Diffie-Hellman
- El Gamal
- Knapsack

2.2 RESUMO CRIPTOGRÁFICO

Resumos criptográficos são amplamente utilizados em protocolos. Eles calculam o resumo de uma mensagem, que é uma pequena sequência de bytes de tamanho fixo. Para uma determinada mensagem, o resumo criptográfico pode ser como uma impressão digital da mensagem, ou seja, uma forma única de representação da mesma. São múltiplos os usos de funções de resumo na criptografia. Eles são uma parte essencial dos esquemas de assinatura digital e de autenticações de mensagem. São muito utilizados também em outros serviços, tais como armazenamento de senhas e derivação de chaves (PAAR; PELZL; PRENEEL, 2010).

Resumos criptográficos também podem ser utilizados para verificar a integridade de informações, ou seja, verificar se a informação não foi alterada. Devido o resumo criptográfico prover uma forma única de representação da informação qualquer alteração pode ser identificada através desta tecnologia.

Atualmente os algoritmos de resumo criptográfico mais utilizados são a família SHA: SHA-1, SHA-224, SHA-256, SHA-383, E SHA-512 (FERGUSON; SCHNEIER; KOHNO, 2010).

2.3 ASSINATURA DIGITAL

A primitiva criptográfica que é fundamental na autenticidade, autorização e não-repúdio é a assinatura digital (MENEZES; OORSCHOT; VANSTONE, 2001).

A assinatura digital é o processo de criptografar um resumo criptográfico, com a chave privada de quem deseja realizar a assinatura (HARRIS, 2010).

Se Bob deseja garantir que a mensagem que ele enviou para Alice não seja modificada e que ela saiba que a mensagem foi enviada realmente por ele, Bob pode assinar digitalmente esta mensagem. Isso significa que uma função de resumo criptográfico deve ser aplicada à mensagem, e então Bob deve cifrar o resultado desta função com sua chave privada. No momento em que Alice for verificar a mensagem enviada por Bob, ela deverá utilizar a mesma função de resumo criptográfico que ele utilizou e armazenar esta informação. Após essa operação, Alice deverá decifrar o valor do resumo criptográfico

enviado por Bob, utilizando a chave pública dele. Ela deve comparar o valor do resumo criptográfico obtido nesta operação com o valor do resumo criptográfico que ela tem armazenado, se os valores forem os mesmos ela pode ter certeza que a mensagem não foi alterada durante o processo de transmissão e que realmente foi enviada por Bob.

O resumo criptográfico garante a integridade da mensagem e a assinatura deste valor provê autenticidade e não-repúdio (HARRIS, 2010).

2.3.1 Co-assinatura

Co-assinaturas são conhecidas também como assinaturas digitais em paralelo (BRASIL, 2010b).

> A geração de co-assinaturas digitais ocorre quando duas ou mais assinaturas digitais são geradas de forma paralela e independente pelos signatários, utilizando conteúdos digitais idênticos. Cada co-assinatura gerada pode conter atributos assinados e não assinados próprios. (BRASIL, 2010d)

2.3.2 Contra-assinatura

Contra-assinatura também são chamadas de assinaturas digitais em série (BRASIL, 2010b).

"A geração de contra-assinaturas digitais ocorre quando uma ou mais assinaturas digitais são realizadas sobre a seqüência de bytes (bloco) que representa uma assinatura digital já existente. Uma contra-assinatura pode conter outros atributos assinados próprios." (BRASIL, 2010d)

Segundo DOC-ICP-15.02 (BRASIL, 2010a), a contra-assinatura deve ser utilizada "quando a ordem de aplicação das assinaturas é relevante, ou seja, quando a função da segunda assinatura é, no mínimo, atestar o recebimento do documento com a primeira assinatura já presente." O DOC-ICP-15.02 ainda especifica que para o uso de contra-assinaturas as partes geradora e verificadora devem estar de acordo com relação ao uso da mesma. O verificador deve estar ciente da sua presença, quantidade, e significado.

2.3.3 CAdES

CMS Advanced Electronic Signatures (CAdES) é um formato de assinatura eletrônica que pode permanecer válida por longos períodos. Isso inclui evidências sobre sua validade mesmo se as partes interessadas tentarem negar

a validade da assinatura (PINKAS; POPE; ROSS, 2008).

Esse formato pode ser considerado uma extensão da RFC 3852 (HOUSLEY, 2004) e RFC 2634 (HOFFMAN, 1999), adicionando atributos assinados e não assinados já definidos quando necessário (PINKAS; POPE; ROSS, 2008).

Nesse formato podem ser realizadas assinaturas com os seguintes tipos de encapsulamento: (BRASIL, 2010d)

• **Assinaturas Anexadas:** o conteúdo assinado está dentro da estrutura da assinatura;

Figura 1: Assinatura Anexada

• **Assinaturas Destacadas:** o conteúdo assinado está fora da estrutura da assinatura.

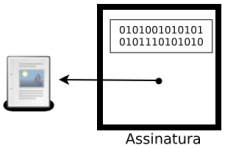

Figura 2: Assinatura Destacada

2.3.4 XAdES

XML Advanced Electronic Signatures (XAdES) estende o formato definido em XMLDSIG (Donald Eastlake, 2002) dentro de um contexto de assinaturas eletrônicas que permanecem válidas por longos períodos.

O XAdES incorpora informações adicionais úteis em casos de uso comuns. Ele inclui evidências sobre a validade da assinatura mesmo se o signatário ou verificador tentar negar (repudiar) a sua validade. Assim uma assinatura eletrônica de acordo com este formato pode ser usada para arbitragem em casos de disputas entre o signatário e o verificador que podem ocorrer em um momento posterior, mesmo anos mais tarde (Donald Eastlake, 2002).

Esse formato permite a realização de assinatura com três tipos de encapsulamento diferentes, são eles: (BRASIL, 2010d)

• **Assinaturas Destacadas:** o conteúdo assinado está externo a estrutura da assinatura. Esse tipo de assinatura é equivalente ao tipo destacada do formato CAdES, representado na Figura 1;

• **Assinaturas Encapsuladoras:** o conteúdo está dentro da estrutura da assinatura;

Figura 3: Assinatura Encapsuladoras

• **Assinaturas Embarcadas:** a assinatura está incluída na estrutura do documento que foi assinado. Esse tipo de assinatura só existe para documentos XML.

Figura 4: Assinatura Embarcada

2.3.5 Política de Assinatura

Uma Política de Assinatura (PA) é um conjunto de regras impostas para a criação e verificação de assinaturas eletrônicas, para que a validade da assinatura possa ser verificada (ROSS; PINKAS; POPE, 2001).

Uma política de assinatura tem uma referência global única, que é ligada a uma assinatura eletrônica pelo signatário como parte do cálculo da assinatura (ROSS; PINKAS; POPE, 2001). Essa referência global única pode ser um identificador único da política (*object identifier*), seu próprio resumo criptográfico, ou outra forma existente de representá-la unicamente.

Ela também precisa ser disponibilizada em formato legível para que se possam avaliar as suas exigências legais e o contexto contratual no qual a mesma está sendo aplicada (ROSS; PINKAS; POPE, 2001).

Para permitir a automação do processo de uma assinatura eletrônica uma outra parte da política de assinatura específica as regras de criação e validação em formato processável por máquina (ROSS; PINKAS; POPE, 2001).

2.4 CARIMBO DO TEMPO

Um carimbo do tempo é um serviço que visa evidenciar que uma determinada informação existia antes de um tempo específico. O Carimbo do Tempo é um documento eletrônico assinado pela Autoridade de Carimbo do Tempo, sendo esta uma terceira parte confiável (ADAMS et al., 2001).

Essa tecnologia é amplamente utilizada em assinaturas digitais para evidenciar que uma determinada assinatura foi criada antes de um dado tempo. Se um certificado de chave pública for revogado isso permite que um verificador possa saber se a assinatura foi criada antes ou depois da data de revogação (ADAMS et al., 2001).

2.5 INFRAESTRUTURA DE CHAVES PÚBLICAS

A Infraestrutura de chaves públicas (ICP) é formada por programas, formatos de dados, procedimentos, protocolos de comunicação, políticas de segurança, e mecanismos de criptografia de chave pública que trabalham em conjunto para possibilitar que pessoas se comuniquem de forma segura. Em outras palavras, uma ICP é responsável por estabelecer o nível de confiança em um ambiente (HARRIS, 2010).

Uma ICP prove suporte à serviços de autenticidade, confidencialidade, não repúdio, e integridade, cujo conceitos vimos na seção 2.1.

Existe uma diferença entre criptografia de chave pública e infraestrutura de chave pública. A primeira é um outro nome para algoritmos assimétricos, enquanto a segunda é o que o próprio nome já diz, uma infraestrutura. Esta infraestrutura assume que a identidade do receptor pode ser assegurada através de certificados digitais e algoritmos assimétricos. Portanto, a ICP contém as peças necessárias para identificar usuários, criar e distribuir certificados, manter e revogar certificados, distribuir e manter as chaves de criptografia, e todas as tecnologias necessárias para se alcançar o objetivo da comunicação criptografada e autêntica (HARRIS, 2010).

Qualquer pessoa que deseja participar de uma ICP deve requisitar um

certificado digital, que é uma credencial que contém a chave pública daquele indivíduo, juntamente com outras informações de identificação. O certificado é criado e assinado por uma terceira parte confiável, conhecida como Autoridade Certificadora (AC). Quando a AC assina um certificado, vincula-se a identidade do proprietário a uma chave pública, e a AC assume a responsabilidade pela autenticidade do indivíduo. É essa terceira parte confiável (AC) que permite que pessoas que nunca se encontraram possam se identificar umas com as outras para obter uma comunicação segura. Para isso basta que as partes envolvidas na comunicação confiem na mesma parte confiável (AC).

2.5.1 Certificado Digital

Uma das partes mais importantes dentro de uma ICP é o certificado digital. Ele é o mecanismo usado para associar uma chave pública com uma coleção de componentes de uma maneira suficiente para identificar o proprietário. O padrão que a AC utiliza para criar certificados é o X.509, que determina os diferentes campos utilizados no certificado e os valores que podem ser populados nestes campos. Atualmente este padrão está na versão 4, que é frequentemente denotada como x.509v4. Muitos protocolos criptográficos utilizam este tipo de certificado, incluindo o SSL (HARRIS, 2010).

Os certificados incluem um número serial, número da versão, informações de identidade, informações de algoritmos, data de emissão e expiração, e uma assinatura da autoridade emissora (HARRIS, 2010).

2.5.2 Lista de Certificados Revogados

Uma lista de certificados revogados, ou LCR, é uma base de dados que contém uma lista de certificados revogados. Qualquer um que desejar verificar um certificado deve verificar a LCR para ver se o certificado em questão já foi revogado. Uma vez que um certificado é adicionado à LCR transações envolvendo o mesmo não serão mais autorizadas. Revogação é muito confiável, e não existe um limite de quantos certificados podem ser revogados (FERGUSON; SCHNEIER; KOHNO, 2010).

2.5.3 Online Certificate Status Protocol

Outra forma de verificar a revogação é a verificação online de certificados. Esta é baseada no Online Certificate Status Protocol (OCSP). Por exemplo, para verificar um certificado Alice consulta uma parte confiável, como uma Autoridade Certificadora ou uma outra parte delegada, com o número serial do certificado em questão. A parte confiável verifica o estado do certificado em sua própria base de dados e então envia uma resposta assinada para Alice. Alice conhece a chave pública da parte confiável e com isso pode verificar a assinatura da resposta obtida. Se a parte confiável diz que o certificado é válido, Alice agora é capaz de saber que o mesmo não está revogado (FERGUSON; SCHNEIER; KOHNO, 2010).

2.5.4 Infraestrutura de Chaves Públicas Brasileira

Através da medida provisória 2.200-2, de 24 de agosto de 2001, o Brasil passou a ter instituída a Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil), transformando o ITI em autarquia (BRASIL, 2001).

> A Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil é uma cadeia hierárquica e de confiança que viabiliza a emissão de certificados digitais para identificação do cidadão quando transacionando no meio virtual, como a Internet. Observa-se que o modelo adotado pelo Brasil foi o de certificação com raíz única, sendo que o ITI além de desempenhar o papel de Autoridade Certificadora Raiz - AC Raiz, também, tem o papel de credenciar e descredenciar os demais participantes da cadeia, supervisionar e fazer auditoria dos processos (BRASIL, 2010a).

> A Autoridade Certificadora Raiz da ICP-Brasil é a primeira autoridade da cadeia de certificação. É executora das Políticas de Certificados e normas técnicas e operacionais aprovadas pelo Comitê Gestor da ICP-Brasil. Portanto, compete à AC-Raiz emitir, expedir, distribuir, revogar e gerenciar os certificados das autoridades certificadoras de nível imediatamente subseqüente ao seu. A AC-Raiz também está encarregada de emitir a lista de certificados revogados e de fiscalizar e auditar as autoridades certificadoras, autoridades de registro e demais prestadores de serviço habilitados na ICP-Brasil. Além disso, verifica se as Autoridades Certificadoras (ACs) estão atuando em conformidade com as diretrizes e normas técnicas estabelecidas pelo Comitê Gestor (BRASIL, 2010a).

2.6 PADRÃO BRASILEIRO DE ASSINATURA DIGITAL

O Padrão Brasileiro de Assinatura Digital (PBAD) é definido pela ICP-Brasil através do conjunto normativo DOC-ICP-15. Esse conjunto normativo é formado por quatro documentos, são estes:

• DOC-ICP-15: Visão Geral sobre Assinaturas Digitais na ICP-Brasil (BRASIL, 2010d);

• DOC-ICP-15.01: Requisitos Mínimos para Geração e Verificação de Assinaturas Digitais na ICP-Brasil (BRASIL, 2010c);

• DOC-ICP-15.02: Perfil de uso geral para assinaturas digitais na ICP-Brasil (BRASIL, 2010a);

• DOC-ICP-15.03: Requisitos das políticas de assinatura digital na ICP-Brasil (BRASIL, 2010b).

Atualmente esses documentos se encontram na versão 2.0.

Os formatos de assinatura digital reconhecidos pelo PBAD, são CAdES e XAdES, citados respectivamente nas seções 2.4.1 e 2.4.2. O PBAD engloba a utilização de Políticas de Assinatura (PAs), seção 2.4.3, definindo um conjunto composto por dez PAs. Cada PA define um perfil de assinatura diferente, dos quais cinco são baseados em CAdES e cinco em XAdES.

Os cinco perfis existentes hoje no PBAD são:

1 - Assinatura Digital com Referência Básica (AD-RB):

> Este tipo de assinatura deve ser utilizado em aplicações ou processos de negócio nos quais a assinatura digital agrega segurança à autenticação de entidades e verificação de integridade, permitindo sua validação durante o prazo de validade dos certificados dos signatários.
>
> Uma vez que não são usados carimbos do tempo, a validação posterior só será possível se existirem referências temporais que identifiquem o momento em que ocorreu a assinatura digital. Nessas situações, deve existir legislação específica ou um acordo prévio entre as partes definindo as referências a serem utilizadas.
>
> Segundo esta PA, é permitido o emprego de múltiplas assinaturas (BRASIL, 2010b).

2 - Assinatura Digital com Referência do Tempo (AD-RT):

> Este tipo de assinatura deve ser utilizado em aplicações ou processos de negócios nos quais a assinatura digital necessita de segurança em relação à irretratabilidade do momento de sua geração.
>
> Como esse tipo de assinatura não traz, de forma auto-contida, referências ou valores dos certificados e das informações de revogação (LCRs ou respostas OCSP) necessários para sua validação posterior, ele deve ser utilizado somente quando esses dados puderem ser obtidos por meios externos, de forma inequívoca. Uma assinatura desse tipo pode ter sua

capacidade probante diminuída, no caso de comprometimento da chave da AC que emitiu qualquer um dos certificados da cadeia de certificação.

Segundo esta PA, é permitido o emprego de múltiplas assinaturas (BRASIL, 2010b).

3 - Assinatura Digital com Referências para Validação (AD-RV):

Este tipo de assinatura inclui, no seu próprio corpo, referências sobre os certificados que compõem a cadeia de certificação e sobre as informações de revogação do certificado digital do signatário. Um carimbo do tempo provê a ligação entre essas informações e o conteúdo assinado.

Ele deve ser usado em aplicações onde se necessita verificar a assinatura a qualquer momento e onde os dados necessários para isso (que estão referenciados no corpo da assinatura), estejam disponíveis para recuperação.

Além de oferecer segurança quanto à irretratabilidade, ele permite que se verifique a validade da assinatura digital mesmo que ocorra comprometimento da chave privada da AC que emitiu o certificado do signatário, desde que o carimbo do tempo sobre as referências tenha sido colocado antes desse comprometimento.

Segundo esta PA, é permitido o emprego de múltiplas assinaturas (BRASIL, 2010b).

4 - Assinatura Digital com Referências Completas (AD-RC):

Este tipo de assinatura inclui, no seu próprio corpo, além das referências, os certificados que compõem a cadeia de certificação e as informações de revogação do certificado digital do signatário. Ele demanda uma maior capacidade de armazenamento.

Ele deve ser usado em situações onde é necessária a verificação completa da validade da assinatura digital a qualquer momento, pois os dados necessários estão auto-contidos na assinatura.

Além de oferecer segurança quanto à irretratabilidade, ele permite que se verifique a validade da assinatura digital mesmo que ocorra comprometimento da chave privada da AC que emitiu o certificado do signatário, desde que o carimbo do tempo sobre as referências/valores dos certificados tenha sido colocado antes desse comprometimento.

Segundo esta PA, é permitido o emprego de múltiplas assinaturas (BRASIL, 2010b).

5 - Assinatura Digital com Referências para Arquivamento (AD-RA):

Este tipo de assinatura é adequado para aplicações que necessitam realizar o arquivamento do conteúdo digital assinado por longos períodos, sabendo-se que podem surgir fraquezas, vulnerabilidades ou exposição a fragilidades dos algoritmos, funções e chaves criptográficas utilizadas no processo de geração de assinatura digital.

> Ele provê proteção contra fraqueza dos algoritmos, funções e tamanho de chaves criptográficas, desde que o carimbo do tempo de arquivamento seja realizado tempestivamente e utilize algoritmos, funções e tamanhos de chave considerados seguros no momento de sua geração.
>
> Além disso, oferece segurança quanto à irretratabilidade, e permite que se verifique a validade da assinatura digital mesmo que ocorra comprometimento da chave privada da AC que emitiu o certificado do signatário (desde que o carimbo do tempo sobre as referências/valores dos certificados tenha sido colocado antes desse comprometimento).
>
> Segundo esta PA, é permitido o emprego de múltiplas assinaturas (BRASIL, 2010b).

A utilização desses perfis para criar e validar uma assinatura ICP-Brasil possibilita a padronização entre a geração e a verificação das assinaturas no Brasil. É importante salientar que todos os cinco perfis apresentados acima podem ser usados em PAs com CAdES ou XAdES.

3 CÓDIGOS DE REFERÊNCIA

Com o desenvolvimento e a publicação do PBAD, sentiu-se a necessidade da criação de um software capaz de implementar todas as funcionalidades, requisitos e restrições impostas por este padrão, afim de facilitar a adoção do mesmo. Para isso foi criado o projeto nomeado como Códigos de Referência com o objetivo de desenvolver e disponibilizar, de forma simples e explicativa, um conjunto de softwares responsáveis pela geração e verificação de assinaturas digitais conforme o padrão brasileiro.

Esse projeto foi dividido em duas etapas fundamentais, sendo essas a implementação de uma biblioteca de referência de assinatura digital e um assinador minimalista de referência, que utiliza todos os recursos disponíveis na biblioteca, demonstrando assim sua utilização eficaz.

3.1 BIBLIOTECA DE REFERÊNCIA

A principal motivação para a criação da biblioteca de referência foi facilitar a implementação de terceiros que desejam criar seus próprios aplicativos conforme o PBAD.

A biblioteca de referência é completa do ponto de vista do PBAD. Ela trata todos os requisitos e imposições deste padrão, sendo possível gerar e verificar qualquer tipo de assinatura, contra-assinatura, e co-assinaturas dentro desse contexto.

Toda a estrutura da biblioteca foi feita para suportar a geração e verificação de assinaturas no formato CAdES e XAdES. Criou-se uma abstração entre esses formatos permitindo que qualquer aplicação que utilize a biblioteca não precise informar se está trabalhando com o formato CAdES ou XAdES. Essa abstração será apresentada em detalhes no capítulo 4.

Outro fator fundamental que a biblioteca de referência traz é o suporte a todas as dez políticas de assinaturas existentes hoje no PBAD. Qualquer outra política que venha a ser homologada pelo ITI também será suportada pela biblioteca, desde que essas políticas utilizem apenas atributos/propriedades da assinatura definidos pelo documento normativo DOC-ICP-15.02 (BRASIL, 2010a) do PBAD. Essa funcionalidade só pode ser implementada devido ao desenvolvimento de um Gerenciador de Políticas (seção 3.3), e a implementação dos atributos/propriedades de assinatura (seção 5.3).

A implementação de todos os atributos/propriedades da assinatura, especificados no normativo DOC-ICP-15.02 (BRASIL, 2010a), trazem uma grande vantagem a aplicação que utilize a biblioteca. Esses atributos/pro-

priedades foram implementados de forma completa, ou seja, toda a parte de construção e validação. Isso diminui um grande esforço do assinador, já que existem diversos atributos/propriedades e cada um possui uma construção e validação particular.

No caso de surgirem novos atributos no PBAD, os quais a biblioteca não conhecerá, esse problema pode ser facilmente resolvido pelo assinador. Para isso, deve-se criar uma nova classe que construa e valide o atributo, implementando a interface *SignatureAttribute* da biblioteca de referência. Desta maneira, o assinador poderá suportar qualquer política futura, com atributos hoje desconhecidos, sem precisar alterar nenhuma linha de código da biblioteca.

Todo o processo de geração da assinatura digital é feito pela biblioteca. O assinador só precisa passar os parâmetros corretos e chamar o método responsável pela geração. Da mesma forma, todo o processo de verificação da assinatura, incluindo a parte criptográfica e a parte do caminho de certificação, é realizado pela biblioteca.

Uma questão que também é interessante citar é o suporte a todos os tipos de encapsulamento de assinatura digital, ou seja, suporte a assinaturas destacadas, anexadas, encapsuladoras, e embarcadas, como visto nas subseções 2.3.3 e 2.3.4.

3.2 ASSINADOR MINIMALISTA DE REFERÊNCIA

O assinador minismalista de referência foi desenvolvido por dois motivos principais. O primeiro foi demonstrar o uso da biblioteca de referência, e o segundo foi a disponibilização para a sociedade de um aplicativo capaz de gerar e verificar assinaturas digitais de acordo com o PBAD. Isso inclui assinaturas no formato CAdES e XAdES, de todos os tipos (destacadas, anexadas, encapsuladoras, e embarcadas), e com qualquer política de assinatura definida no documento normativo DOC-ICP-15.03 (BRASIL, 2010b).

Este assinador utilizou todas as funcionalidades disponilizadas pela biblioteca de referência, o que tornou seu desenvolvimento muito mais rápido e simples. Toda a parte de implementação do assinador foi realizada pensando em desenvolver um código que demonstrasse da maneira mais clara possível como se utilizar a biblioteca de referência. Assim, o assinador pode ser uma referência para terceiros que desejam implementar seus próprios aplicativos utilizando a biblioteca de referência.

O assinador possui uma pré-configuração que permite criar uma assinatura de forma muito simples. É preciso apenas selecionar o arquivo a ser assinado e clicar no botão assinar. Essa pré-configuração permite a realiza-

ção de uma Assinatura Digital de Referência Completa (AD-RC), no formato CAdES e do tipo destacada. Para usuários avançados é possível alterar essa configuração.

3.3 GERENCIADOR DE POLÍTICAS

O Gerenciador de Políticas foi uma etapa muito importante no desenvolvimento da biblioteca de referência. Ele foi responsável pela escrita e leitura da LPA e das PAs nas linguagens de máquina ASN1 e XML.

A escrita da LPA e das PAs, proporcionada pelo gerenciador de políticas, foi fundamental para o desenvolvimento do projeto. Essa escrita tornou possível gerar a LPA e as PAs em linguagem de máquina (ASN1 e XML), o que só existia até então em forma textual. As PAs geradas em linguagem de máquina foram chamadas de Artefatos de PA.

Devido a esta implementação foi possível fazer com que a biblioteca de referência trabalhasse com as políticas de assinatura de forma dinâmica. Com a leitura de PA implementada foi possível conhecer quais atributos, regras e restrições se aplicavam a cada política, tornando possível gerar e verificar assinaturas com PA de maneira automática.

3.4 CARIMBADORA DE TESTES

Oito das dez políticas de assinatura do PBAD utilizam o carimbo do tempo como atributo não assinado obrigatório da assinatura. Para ser possível testar a criação e a verificação das assinaturas que obrigam o uso deste atributo, sentiu-se a necessidade de um serviço de emissão de carimbos do tempo.

Para ter esse serviço a disposição foi implementada uma estrutura de emissão de carimbos do tempo para dar suporte ao projeto. Essa estrutura foi implementada na linguagem de programação JAVA e contou com a criação de uma Autoridade de Carimbo de Tempo responsável pela emissão de duas carimbadoras, uma com o certificado válido e a outra com certificado revogado para fins de teste.

Esta aplicação está executando atualmente em um servidor hospedado no LabSEC. Ela está disponível para todos que desejam fazer requisições de carimbo do tempo. Por se tratar de uma aplicação de teste, a data e hora da protocolação do documento é definida de acordo com a hora especificada no servidor hospedeiro e o carimbo emitido não possui validade jurídica.

A carimbadora é simples e funciona da seguinte forma: a parte cliente

precisa enviar uma requisição para o servidor hospedeiro contendo apenas os bytes do resumo criptográfico da informação que se deseja carimbar. Após o envio da requisição o cliente solicita uma resposta do servidor. Essa resposta contém o resumo criptográfico da informação enviada e o momento (data e hora) da protocolação.

A verificação desse carimbo deve analisar se o resumo criptográfico da informação é o mesmo resumo criptográfico que consta no carimbo do tempo. Também deve-se verificar a assinatura e o caminho de certificação da carimbadora utilizada. Somente se esses requisitos forem verdadeiros o carimbo do tempo é confiável, e pode-se dizer que não foi forjado.

3.5 PRINCIPAIS TECNOLOGIAS EMPREGADAS

Para a realização deste projeto as principais tecnologias utilizadas são apresentadas nas subseções abaixo.

3.5.1 Linguagem de Programação Java

Java é uma linguagem de programação de alto nível que utiliza o paradigma de programação orientado a objetos. Essa linguagem proporciona portabilidade e independência de plataforma às suas aplicações. Isso acontece, devido ao fato da compilação de um código Java gerar código intermediário, conhecido como *bytecodes*. Esse código intermediário é executado pela Máquina Virtual Java (JVM). Sendo assim, é possível executar uma aplicação implementada em Java em qualquer plataforma que tenha uma JVM instalada.

Outro ponto a favor da linguagem Java é a manutenção de uma biblioteca de classe (APIs) extensa e bem documentada, gerando um excelente suporte aos usuários (Oracle Technology Network, 2011a).

3.5.2 Trac

O Trac é um projeto de software livre que é utilizado para o gerenciamento de projetos. Ele procura interferir o mínimo possível nos processos e nas políticas estabelecidas por um time de desenvolvimento.

O objetivo desse software é ajudar os desenvolvedores a escrever softwares grandes com qualidade. Ele utilizada uma abordagem minimalista para softwares de gerenciamento de projetos baseados na web.

Um fator interessante desse software é que ele pode ser facilmente integrado com o Subversion (subseção 3.5.3) ou qualquer outro software controle de versão através de instalações de alguns modelos de comunicação (The Trac Project, 2011).

3.5.3 Subversion (SVN)

O Subversion (SVN) é um software livre para controle de versão. Ele é desenvolvido como um projeto da fundação de software Apache, sendo parte de uma rica comunidade de desenvolvedores e usuários.

O SVN é mundialmente reconhecido e é caracterizado por sua confiabilidade na manutenção do versionamento de arquivos e sistemas. Algumas características desse software são a simplicidade no uso e a capacidade de suportar diversos usuários e projetos (Apache Subversion, 2011).

3.5.4 BouncyCastle

O BouncyCastle é uma bibilioteca criptográfica de caráter livre, disponibilizada nas linguagens de programação Java e C#. Essa biblioteca tem suporte a inúmeras funcionalidades de segurança da informação, tais como: criptografia, assinatura digital, carimbo do tempo, e resumos criptográficos (Bouncy Castle, 2011).

3.6 PROCESSO DE DESENVOLVIMENTO

O projeto contou com a participação de um gerente de projetos, um gerente de qualidade, um consultor externo e mais três programadores, sendo o autor um dos programadores. Perto do término do projeto mais dois programadores iniciantes se juntaram a equipe com o objetivo de serem introduzidos a conceitos como assinatura digital, resumo criptográfico, e criptografia.

O gerente de projetos era responsável por manter o planejamento do projeto, definindo prioridades e estabelecendo metas. Esse também era responsável pela comunicação da equipe com ITI, com o intuito de sanar dúvidas, apresentar sugestões de melhorias no PBAD, entre outros.

Utilizou-se o software Trac (The Trac Project, 2011) para o gerenciamento do projeto. Este software foi essencial para o desenvolvimento do projeto, provendo funcionalidades importantes tanto para as partes gerenciais quanto para as operacionais. Esse software permitiu que os gerentes pudes-

sem anotar as funcionalidades que deveriam ser implementadas ou ajustadas, dando o nome de *tickets* a essas anotações.

Com a utilização do software Trac, todos os programadores ao final da implementação de uma tarefa poderiam observar a lista de *tickets* e iniciar a implementação de uma nova tarefa. Toda vez que um programador terminava a implementação de um *ticket*, este enviava a sua implementação para o gerente de qualidade via funcionalidades do software Trac.

O gerente de qualidade era responsável por avaliar as implementações dos *tickets* e dar uma resposta ao programador com relação a qualidade e correção do mesmo. Quando o gerente de qualidade encontrava problemas na implementação, ele encaminhava este *ticket* novamente ao programador, com notas especificando as correções a serem tomadas. Após o programador corrigir os erros ele enviava de volta o *ticket* ao gerente de qualidade que podia fechar, ou encaminhar o *ticket* novamente devido a algum problema. No caso, se o gerente de qualidade avaliasse a implementação adequada, ele fechava o *ticket* sem precisar encaminhá-lo ao desenvolvedor.

Reuniões diárias foram realizadas para que cada componente da equipe pudesse apresentar, rapidamente, o que estava fazendo e suas possíveis dúvidas. Essas reuniões, em geral, ajudaram a equipe a desenvolver a aplicação de forma bastante integrada, possibilitando a todos os integrantes terem conhecimento do que cada um estava fazendo. Outro ponto interessante era que as dúvidas apresentadas muitas vezes eram sanadas rapidamente por algum integrante da equipe.

4 ESTRUTURA DE INTEGRAÇÃO CADES E XADES

A estrutura da biblioteca de referência foi projetada com o objetivo de obter-se uma integração entre assinaturas no formato CAdES e XAdES. Essa integração foi responsável por uma funcionalidade interessante da biblioteca, que é a possibilidade de gerar e verificar assinaturas sem precisar informar diretamente o formato a ser utilizado. No momento de criação da assinatura o usuário precisa somente selecionar uma PA. Assim, a biblioteca identifica o formato e gera a assinatura. Na verificação, a biblioteca consegue reconhecer o formato da assinatura e verificá-la automaticamente.

Essa integração também ajudou a padronizar as implementações entre os formatos. As seções a seguir apresentam todo o processo de desenvolvimento dessa estrutura.

4.1 IMPLEMENTAÇÃO PERFIL RB E RT

No início da implementação da biblioteca ainda não se pensava em desenvolver uma estrutura de integração entre os formatos CAdES e XAdES. Com isso, foi iniciada a implementação da biblioteca de referência sem se importar com esta integração, dividindo a equipe em dois grupos, cada um responsável pela implementação de um formato.

O projeto tinha como objetivo representar cada perfil de assinatura através de uma classe. Desta forma, seriam desenvolvidas uma classe para cada um dos dez perfis existentes no PBAD, cinco em cada formato. Também foram elaboradas uma classe geradora de assinatura e uma verificadora em cada formato.

Primeiro, foram implementadas as classes geradoras, verificadoras e as classes representantes do perfil Referência Básica (RB-CAdES e RB- XAdES). Com o perfil RB implementado, o próximo passo foi implementar o perfil de Referência de Tempo (RT). Devido ao fato do perfil RT ter todos os atributos presentes no perfil RB e a mais o atributo de carimbo do tempo de assinatura, as classes que representavam este perfil apenas estendiam as classes RB e implementavam este atributo.

Ao fim da implementação do perfil RT, CAdES e XAdES, percebeu-se a duplicação de código em métodos genéricos, tais como assinar e verificar uma assinatura. Isso ocorreu pelo fato da equipe estar dividida em grupos, conforme previamente dito. Ainda pelo motivo dessa divisão, classes com a mesma função tinham nomes muito distintos, e por se tratar da implementação de uma biblioteca isso poderia trazer desconforto aos seus usuários, que

em alguns momentos teriam que utilizar métodos e classes com nomes diferentes para fazer a mesma operação. A partir disso, resolveu-se integrar a implementação dos formatos para diminuir ao máximo esses problemas.

4.2 INTEGRAÇÃO CADES E XADES

Pelos motivos apresentados na seção anterior, foi decidido remodelar a biblioteca. O objetivo desta nova estrutura de integração era abstrair as diferenças de implementações entre os formatos, apresentando uma camada única para o usuário, independente de formato.

O primeiro modelo de integração implementado é apresentado na subseção 4.2.1, posteriormente esse modelo foi substituído por outro, apresentado na seção 4.3.

4.2.1 Implementação integrada não dinâmica

A implementação integrada não dinâmica foi o primeiro modelo desenvolvido para fazer a integração entre os formatos de assinatura CAdES e XAdES. Este modelo se baseou na implementação de interfaces java que definiam basicamente as classes de geração de assinatura e de perfis de assinatura.

O ponto de maior avanço neste modelo foi o desenvolvimento do diagrama de classes representado pela Figura 5 (ver página 45). Esse diagrama representa a estrutura principal de interfaces e classes que deveriam ser implementadas.

Como os perfis RB e RT já estavam implementados, o processo de integração foi simples. Basicamente foi feito uma reestruturação das classes já implementadas, alterando apenas nomes de métodos e de classes, mantendo suas funcionalidades.

Além disso, foram implementadas as interfaces definidas no diagrama. A criação dessas interfaces garantiu que os métodos teriam a mesma assinatura, possibilitando uma camada de abstração para o usuário final, independente do formato.

Com toda a implementação da biblioteca baseada nesse modelo, percebeu-se uma grande melhora na qualidade do software que estava sendo desenvolvido. Durante o desenvolvimento notou-se que esse modelo poderia ser aprimorado e substituído por um outro de maior qualidade.

O maior problema encontrado nesse modelo foi o fato de ser baseado na implementação de classes que representavam cada perfil de assinatura

existente no PBAD.

Um fator importante sobre as PAs é o seu ciclo de vida, elas têm prazo de validade. Portanto, existe a necessidade de renovar, abandonar ou criar novas PAs.

Com isso, a cada nova PA criada uma nova classe precisaria ser inserida à biblioteca, e a para cada PA revogada sua classe representante deveria sofrer a remoção das funcionalidades de criação da assinatura. Para resolver esse problema surgiu o modelo que ficou conhecido como Implementação Integrada Dinâmica, apresentado na seção 4.3.

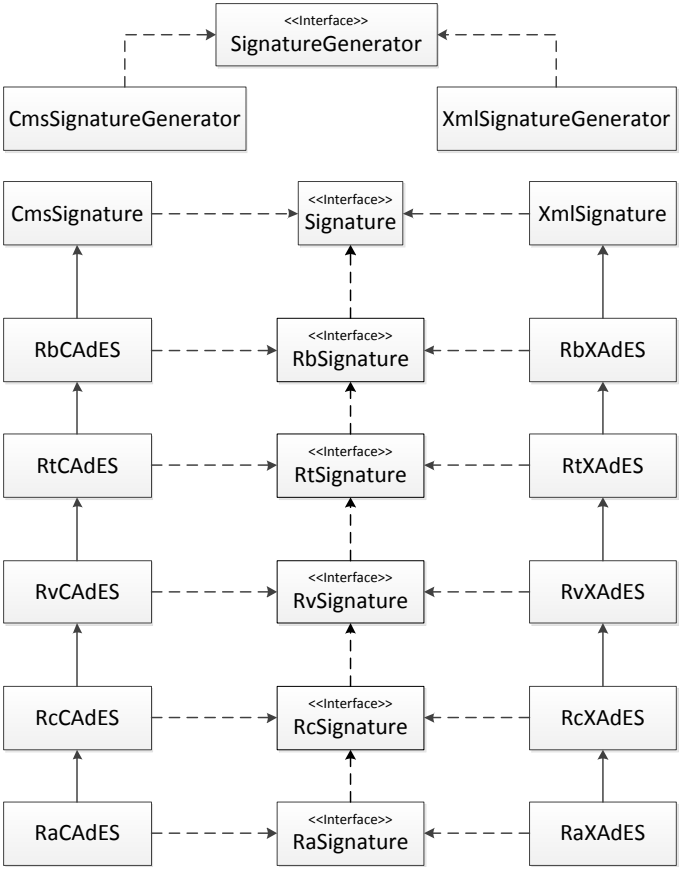

Figura 5: Diagrama de classes - Estrutura integrada não dinâmica

4.3 IMPLEMENTAÇÃO INTEGRADA DINÂMICA

A implementação integrada dinâmica foi o segundo e último modelo desenvolvido para a estrutura integrada entre CAdES e XAdES. Manteve-se a idéia da utilização de interfaces, mas com um grande diferencial do modelo anterior, implementado agora de forma dinâmica, sem precisar de classes para representar perfis de assinatura. Desta forma, a própria biblioteca passaria a reconhecer e montar os perfis de assinatura de acordo com a política desejada.

Com a criação do modelo dinâmico foi possível extinguir o problema apresentado na seção acima, de tal maneira que a implementação das classes que representavam perfis de PAs foram removidas da biblioteca. Uma nova estrutura responsável por reconhecer os perfis de assinatura foi montada, agora com a capacidade de interpretar qualquer perfil de assinatura. Assim, mesmo com a criação de uma nova PA a biblioteca não necessita sofrer nenhuma alteração. Ela consegue interpretar informações de uma nova PA e montar o perfil correspondente.

A estrutura dinâmica está preparada para reconhecer qualquer PA que utilizem os atributos/propriedades de assinatura definidos no DOC-ICP-15.02 (BRASIL, 2010a). Se for criado algum atributo/propriedade de assinatura novo que ainda não está definido nesse documento, a aplicação que faz uso da biblioteca deverá criar uma classe que implemente a criação e validação deste atributo. Para isso é preciso implementar a interface *SignatureAttribute* da biblioteca de referência.

O detalhamento de como foi feita e como funciona esta estrutura é apresentado nas subseções seguintes.

4.3.1 Leitura dinâmica de Políticas de Assinatura

O primeiro passo para utilização da estrutura dinâmica foi fazer a leitura dos artefados de PA (PA em linguagem de máquina ASN.1 e XML) através do Gerenciador de Políticas (seção 3.3). Essa leitura permitiu, em tempo de execução, conhecer as informações relevantes no momento de criar e verificar uma assinatura que utilize PA, como é o caso no PBAD.

Essa leitura consiste basicamente em obter as seguintes informações:

- Prazo de validade da PA;
- Atributos obrigatórios impostos pela PA (assinados e não assinados, do signatário e do verificador);
- Informações de referência de validação da assinatura;
- Certificado da âncora de confiança inserido na PA;
- Algoritmo de resumo criptográfico a ser utilizado na assinatura;

• Algoritmo de criptografia assimétrica a ser utilizado na assinatura;
• Tamanho mínimo da chave criptográfica a ser utilizada.

Através destas informações é possível saber o que deve ser inserido e o que pode ser utilizado no momento de geração de uma assinatura de acordo com a PA desejada, e também tudo que deve ser verificado para garantir a validade da assinatura.

Outra questão que a leitura dinâmica agregou ao projeto foi o fato de que para gerar uma assinatura ou verificar não é necessário informar o formato a ser utilizado. O usuário precisa, apenas, selecionar uma PA no momento de geração e a aplicação consegue reconhecer o formato a ser utilizado. A verificação é feita sem nenhuma indicação de formato.

4.3.2 Uso de interfaces

A utilização de interfaces da integração entre os formatos CAdES e XAdES foi mantida. No entanto, devido a leitura de políticas de assinatura ser de forma dinâmica não foi mais necessário o uso das interfaces que definiam os perfis de assinatura.

As pricipais interfaces da biblioteca de referência são listadas abaixo:

• *Signature*: Define métodos comuns entre uma assinatura CAdES e XAdES, possibilitando a manipulação da assinatura sem precisar saber necessariamente qual formato será usado.

• *SignatureAttribute*: Define todas as funções que um atributo de assinatura deve ter. Faz com que exista um padrão de desenvolvimento entre todos os atributos implementados na biblioteca de referência.

• *ContainerGenerator*: Define os métodos para geração de uma assinatura, independente de formato.

• *SignatureContainer*: Define os métodos utilizados dentro de um contêiner de assinaturas, independente de formato. Na biblioteca de referência, um contêiner de assinatura é uma estrutura que pode armazenar uma ou mais assinaturas de mesmo formato.

4.3.3 GenericEncoding

A classe *GenericEncoding* possui uma função essencial. A partir da implementação desta classe foi possível abstrair a codificação dos atributos/-propriedades de assinatura em ASN.1 e XML, tornando viável o tratamento dos mesmos como se fossem iguais.

Essa classe foi implementada utilizando dois atributos de classe, um

para codificação de atributos em ASN.1 e outro das propriedades XML. Ela possui dois construtores, um recebe o objeto codificado em ASN.1 e outro em XML, dependendo o construtor utilizado um dos atributos será instanciado.

Para obter estes atributos existem dois métodos *get*, um retornando o objeto de acordo com a codificação em ASN.1 e outro em XML, possibilitando a biblioteca manipular os atributos da mesma forma, independente de formato.

4.3.4 Uso de Reflexão Java

A reflexão em Java é normalmente utilizada em aplicações que necessitam examinar ou modificar, em tempo de execução, o comportamento de aplicativos executados através da máquina virtual Java. A reflexão é uma técnica poderosa e permite às aplicações executarem funções que de outra forma seriam inviáveis. Esta é uma funcionalidade avançada e deveria ser utilizada apenas por usuários com grande conhecimento da linguagem. (Oracle Technology Network, 2011b)

A implementação dinâmica só foi possível graças a tecnologia de reflexão Java. Ela permitiu instanciar, em tempo de execução, cada atributo de assinatura (assinado e não assinado) que foi reconhecido na etapa de leitura de política. Através desta instanciação dinâmica foi possível gerar os atributos e validá-los no momento da verificação da assinatura.

Uma vez que todos os atributos de assinatura eram lidos, instanciados, e validados de forma dinâmica, não se fez mais necessário a utilização de classes para representar perfis de assinatura.

Exemplo do uso de reflexão na biblioteca de referência:

```
1 // Instanciando um atributo de assinatura e validando
2 Constructor<?> constructor = attributeClass.getConstructor(new Class<?>[] {
3                     AbstractVerifier.class, Integer.class });
4 SignatureAttribute attributeInstance = (SignatureAttribute) constructor
5                     .newInstance(this, attributeIdentifier);
6 attributeInstance.validate();
```

Neste exemplo é apresentado o uso de reflexão para instanciar e validar um atributo na biblioteca de referência. Na linha número 2 é definido quais são os tipos dos parâmetros esperados no construtor da classe que será instanciada. Na linha número 4 é instanciado um atributo de assinatura passando como argumento os tipos definidos na segunda linha. Note que o argumento *attributeIdentifier* identifica qual atributo será instanciado, por meio de seu identificador único. Na linha 6 é invocado o método de validação do atributo instanciado no passo anterior.

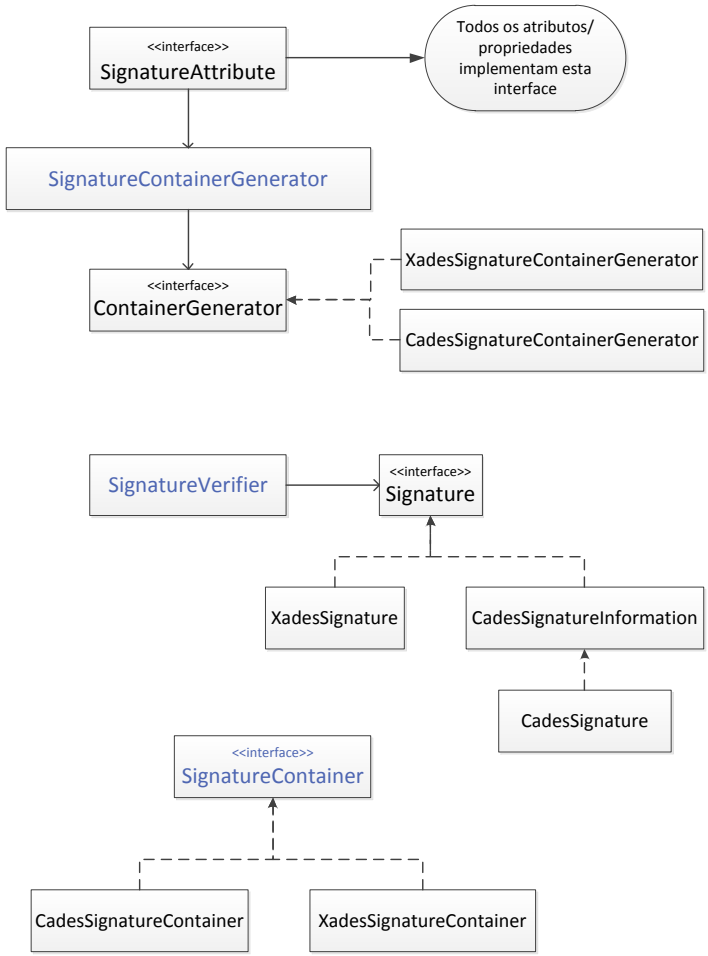

Figura 6: Principais classes da estrutura integrada dinâmica

5 FUNCIONALIDADES FUNDAMENTAIS

Neste capítulo é destacada a implementação de outras funcionalidades que foram fundamentais para o desenvolvimento do projeto.

5.1 GERAÇÃO DE ASSINATURA

A geração da assinatura na biblioteca de referência foi implementada de uma forma a trazer maior comodidade para o usuário que deseja gerar tanto assinaturas no formato CAdES quanto XAdES. Para isso trabalhou-se no desenvolvimento de uma única classe geradora de assinatura para ambos os formatos.

A classe desenvolvida recebeu o nome de *SignatureContainerGenerator* tendo como seu método principal o *sign()*. O construtor dessa classe recebe apenas o identificador da política de assinatura, ou então a própria política em linguagem de máquina.

O método *sign()* faz a leitura do campo que representa o formato da assinatura, identificando se a mesma deve ser gerada em CAdES ou XAdES. O método declara uma variável do tipo *ContainerGenerator*, instanciando a classe *CadesContainerGenerator* caso o formato lido seja CAdES, e a classe *XadesContainerGenerator*, caso o formato seja XAdES. Ambas as classe implementam a interface *ContainerGenerator*.

Esse método retorna um objeto do tipo *SignatureContainer*, que é uma interface implementada pelas classes *CadesSignatureContainer* e *XadesSignatureContainer*. Assim, é possível gerar contêineres de assinaturas em ambos os formatos sem precisar indicá-los explicitamente, apenas é selecionado uma política de assinatura.

Até esta etapa temos um contêiner de assinatura representado por um objeto *SignatureContainer*. Para obter a assinatura, utiliza-se o método *getSignatureAt(int index)* que retorna um objeto do tipo *Signature* na posição do índice passado como parâmetro. A interface *Signature* é implementada pelas classes CadesSignature e XadesSignature, possibilitando assim a utilização de algumas funcionalidades específicas de cada formato.

Exemplo de geração de uma assinatura:

```
1 SignatureContainerGenerator signatureContainerGenerator
2             = new SignatureContainerGenerator(signaturePolicyIdentifier);
3 signatureContainerGenerator.addContentToBeSigned(contentToBeSigned);
4 signatureContainerGenerator.setSigner(signer);
5 signatureContainerGenerator.addAttribute(signingCertificate);
6 signatureContainerGenerator.addAttribute(dataObjectFormat);
```

```
7  SignatureContainer container = signatureContainerGenerator.sign();
8  Signature signature = container.getSignatureAt(0);
```

Abaixo segue o diagrama de atividades que representa essa funcionalidade:

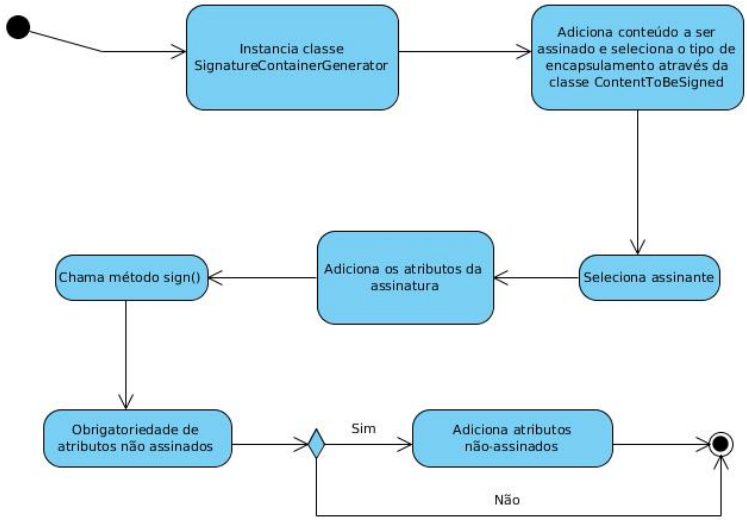

Figura 7: Geração de Assinatura

5.2 VERIFICAÇÃO DE ASSINATURA

A verificação da assinatura, assim como sua geração, foi projetada para ser independente de formato. A classe *SignatureVerifier* é responsável por esta verificação. Ela estende a classe *AbstractVerifier*, que contém métodos usados na verificação de carimbos do tempo e de assinaturas.

Para verificar uma assinatura através da biblioteca é necessário instanciar a classe *SignatureVerifier* passando como parâmetro um objeto *Signature* e passar as informações de revogação juntamente com o caminho de certificação do assinante através de um objeto *CertStore*. Outras maneiras de verificação através desta classe também são possíveis, mas estão fora do escopo desse trabalho.

Exemplo de como essa verificação é realizada:

```
1  Signature signature = ...;
2  SignatureVerifier signatureVerifier = new SignatureVerifier(signature);
```

```
3  signatureVerifier.setCertStore(this.certStore);
4  boolean isValid = signatureVerifier.verify();
```

Abaixo segue o diagrama de atividades que representa essa funcionalidade:

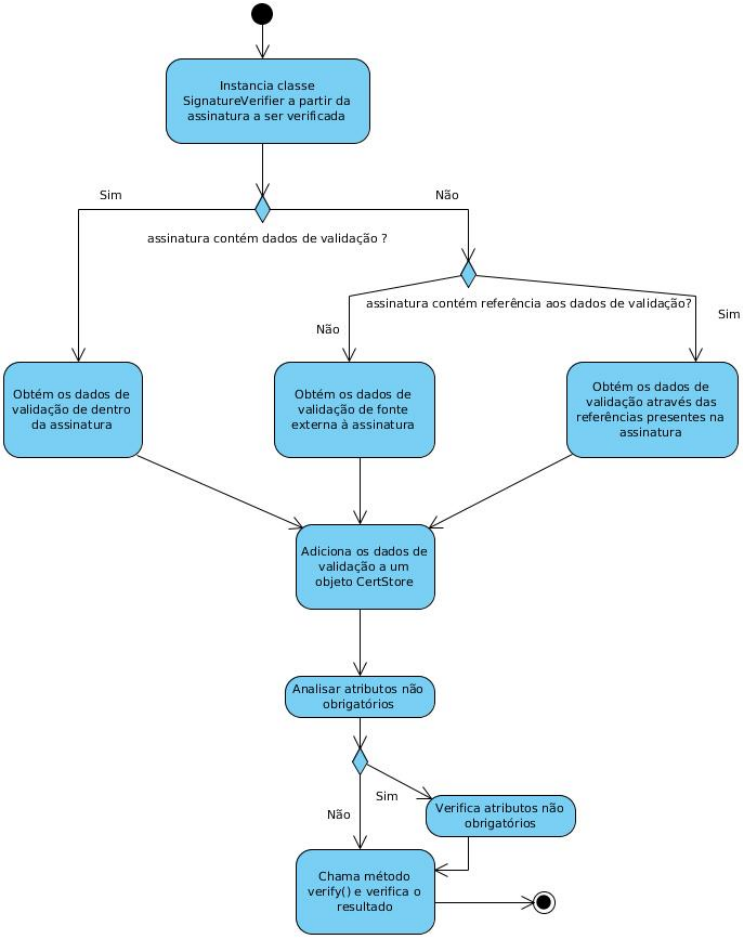

Figura 8: Verificação de Assinatura

5.3 ATRIBUTOS DE ASSINATURA

A biblioteca de referência tem suporte aos atributos/propriedades de assinatura apresentados no DOC-ICP-15.02 (BRASIL, 2010a), assinados e não-assinados.

A implementação desses atributos/propriedades foi realizada através de uma interface Java que definiu o que um atributo de assinatura deveria ter. Essa interface recebeu o nome de *SignatureAttribute*.

O desenvolvimento dessa interface contou basicamente com a definição de um método de validação, a identificação se o atributo é assinado ou não-assinado, se é único (pode aparecer apenas uma vez dentro da estrutura de uma assinatura), seu identificador e um método de retorno do atributo codificado conforme sua especificação.

Após a definição da interface e seus respectivos métodos, iniciou-se a implementação dos atributos. Para todos os atributos implementados foram criados três tipos diferentes de construtores, os quais seguem abaixo:

• *Tipo 1*: Recebe os parâmetros necessários para sua construção. Esse construtor é utilizado no momento de criação da assinatura, quando se gera os atributos para adicioná-los à assinatura.

Exemplo de uso:

```
1 Signature signature = ...;
2 byte[] digestValue = ...;
3 SignatureAttribute idMessageDigest = new IdMessageDigest(digestValue);
4 signature.addAttribute(idMessageDigest);
```

• *Tipo 2*: Recebe um objeto GenericEncoding. Serve para quando a assinatura já existe e se pretende obter algum atributo que está inserido nela. Normalmente essa operação é realizada para resgatar informações inseridas dentro dos atributos.

Exemplo de uso:

```
1 Signature signature = ...;
2 GenericEncoding genericEncoding = signature.
3                                   getEncodedAttribute(attributeId);
4 IdAaEtsCertValues certValues = new IdAaEtsCertValues(genericEncoding);
5 List<X509Certificate> certs = certValues.getCertValues();
```

• *Tipo 3*: Recebe um objeto AbstractVerifier e um índice. Esse construtor é utilizado pela própria biblioteca no momento de verificação da assinatura. Devido ao fato da validação dos atributos ser realizada através da instanciação dos mesmos por uso de reflexão (subseção 4.3.4) foi necessário ter um construtor idêntico para todos os atributos. Com esses parâmetros a aplicação é capaz de reconstruir qualquer atributo de assinatura e validá-lo.

Exemplo de uso:

```
1  // Instanciando um atributo de assinatura a partir do construtor Tipo 3
2  Constructor<?> constructor = attributeClass.getConstructor(new Class<?>[]
3                  AbstractVerifier.class, Integer.class });
4  SignatureAttribute attributeInstance = (SignatureAttribute) constructor
5                  .newInstance(this, attributeIdentifier);
6  // validando o atributo
7  attributeInstance.validate();
```

Cada atributo possui um método de validação particular, conforme sua especificação. A validação do atributo pode estar diretamente ligada ao resultado final da verificação da assinatura. A biblioteca é capaz de reconhecer essa situação e informar se a falha de validação de algum atributo torna a assinatura inválida ou não. Além de informar todos os atributos que tiveram problemas em suas respectivas validações.

5.4 CONTRA-ASSINATURA

A contra-assinatura é o único atributo/propriedade de assinatura que tem um tratamento diferente na sua geração e verificação. Isso se deve ao fato desse atributo/propriedade ter um comportamento muito semelhante a uma assinatura comum.

Para integração entre os formatos CAdES e XAdES das contra-assinaturas, criou-se uma interface chamada *CounterSignatureInterface*. Ela permite que os usuários utilizem os mesmos métodos seja qual for o formato utilizado.

Para geração da contra-assinatura implementou-se uma classe chamada *CounterSignatureGenerator* que estende a classe responsável pela geração das assinaturas (*SignatureContainerGenerator*), adicionando apenas a funcionalidade de contra-assinar.

O processo de geração de uma contra-assinatura é muito semelhante ao processo de geração de uma assinatura. O que muda é que a classe a ser instanciada é sempre a *CounterSignatureGenerator*, ao invés da classe *SignatureContainerGenerator*, o conteúdo a ser assinado sempre deve ser o valor, em *bytes*, da assinatura que está sendo contra-assinada, e o método a ser chamado é o *counterSign()* e não mais *sign()* .

Na biblioteca o conteúdo a ser contra-assinado é representado pelas classes *CadesSignatureToBeSigned* e *XadesSignatureToBeSigned*, que estendem a classe *ContentToBeSigned*. Todo o resto do processo de geração da contra-assinatura é exatamente igual ao da assinatura.

É interessante deixar claro que é possível gerar contra-assinaturas de outras contra-assinaturas. Para se obter a contra assinatura utiliza-se o método

getCounterSignatures(), definido pela interface *Signature*, que retorna uma lista contendo todas as contra-assinaturas, ou o *getCounterSignature(X509Certificate signerCertificate)* que retorna a contra-assinatura específica do contra-signatário proprietário do certificado passado como parâmetro.

5.5 CO-ASSINATURA

No formato CAdES foi preciso ter um tratamento diferenciado para as co-assinaturas. No formato XAdES é preciso apenas assinar um documento que já estava assinado e a co-assinatura é gerada e adicionada ao documento automaticamente. No caso do CAdES essa co-assinatura deve ser gerada e adicionada manualmente à estrutura da assinatura a qual foi co-assinada.

Para isso, desenvolveu-se na biblioteca uma estrutura que automatiza esse processo de adição da co-assinatura a estrutura de uma assinatura. Esse desenvolvimento contou com a criação de uma classe chamada *CadesCoSignatureGenerator*, que estende a classe *SignatureContainerGenerator*, da mesma forma que foi implementado nas contra-assinaturas.

Esta classe possui um método chamado *coSign()*, que é responsável tanto por criar a co-assinatura quanto por adicioná-la a estrutura de uma assinatura. Se a primeira assinatura foi feita de forma anexada, ou seja, o conteúdo do documento está inserido na assinatura, a geração da co-assinatura é feita sem precisar selecionar o conteúdo a ser assinado, caso contrário, o conteúdo precisa ser indicado.

Todo o processo é igual ao de uma assinatura comum, alterando-se a instanciação da classe *SignatureContainerGenerator* pela classe *CadesCoSignatureGenerator* e a invocação do método *coSign()* ao fim do processo no lugar do método *sign()*.

6 CONSIDERAÇÕES FINAIS

Os resultados obtidos com o trabalho desenvolvido ocorreram dentro do esperado. A maioria dos objetivos traçados no início do projeto foram cumpridos, com exceção da implementação, por necessitar entregar o projeto no prazo definido, de alguns atributos/propriedades de assinatura opcionais. São esses:

- *CommitmentTypeIndication*
- *AllDataObjectsTimeStamp*
- *ContentTimeStamp*
- *IndividualDataObjectsTimeStamp*

A definição dos normativos DOC-ICP-15 (BRASIL, 2010d) forneceu a estrutura para a padronização da assinatura digital no Brasil. O que faltava era uma implementação do padrão descrito nesses documentos para mostrar que era possível desenvolvê-lo e mantê-lo em funcionamento. Através da implementação dos códigos de referência isso tornou-se realidade.

A biblioteca de referência é um software do qual entidades poderão se beneficiar, pois toda a implementação pesada do PBAD é abstraída por ela. As vantagens trazidas pela biblioteca de referência a desenvolvedores que pretendem implementar um assinador digital conforme o PBAD são muitas. Entre elas destacam-se:

- Suporte a todas as políticas de assinaturas do PBAD;
- Geração de assinatura no formato CAdES e XAdES;
- Verificação de assinatura no formato CAdES e XAdES;
- Verificação do caminho de certificação;
- Implementação e validação dos atributos de assinatura do PBAD.

Além do desenvolvimento dessa biblioteca, teve-se também o desenvolvimento do assinador minimalista de referência. O assinador utiliza todos os recursos que a biblioteca de referência tem para oferecer. Ele é um grande exemplo de utilização da biblioteca para aqueles desenvolvedores que desejam ter uma base mais sólida para desenvolver sua aplicação de assinatura digital dentro do contexto do PBAD.

O maior ganho com a implementação do PBAD é interoperabilidade entre os sistemas de assinatura digital no País. Nesse contexto, aplicações diferentes poderão gerar assinaturas interoperáveis sem nenhum problema.

Por se tratar de um projeto de software livre, outros poderão se be-

neficiar dessas implementações, podendo utilizar essas estruturas para fins comerciais, acadêmicos, e/ou pessoais.

6.1 TRABALHOS FUTUROS

As propostas de trabalhos futuros que visam o aprimoramento dos códigos de referência são:

- reformular o tratamento de exceções da biblioteca de referência e do assinador minimalista de referência;
- criar um padrão para o uso de provedores criptográficos, permitindo que usuário da biblioteca tenha a opção de escolher entre o provedor padrão ou algum outro;
- Implementar os atributos/propriedades de assinatura opcionais que ainda não foram implementados (citados nas considerações finais);
- implementar suporte para geração e validação de assinaturas digitais ICP-Brasil de documentos eletrônicos maiores que a memória da máquina virtual Java;
- aprimoramento do assinador minimalista de referência e inclusão de novas funcionalidades;
- implementar a validação da Lista de Políticas Aprovadas (LPA).

REFERÊNCIAS BIBLIOGRÁFICAS

ADAMS, C. et al. *Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)*. IETF, ago. 2001. RFC 3161 (Proposed Standard). (Request for Comments, 3161). Disponível em: <http://www.ietf.org/rfc/rfc3161.txt>.

Apache Subversion. *Enterprise-class centralized version control for the masses*. 2011. Disponível em: <http://subversion.apache.org/>. Acesso em: 18 jul 2011.

Bouncy Castle. *The Legion of the Bouncy Castle*. 2011. Disponível em: <http://www.bouncycastle.org/>. Acesso em: 18 jul 2011.

BRASIL. *MP nº 2200-2*. 2001. Disponível em: <http://www.planalto.gov.br/ccivil_03/mpv/Antigas_2001/2200-2.htm>. Acesso em: 30 jun 2011.

BRASIL. *PERFIL DE USO GERAL PARA ASSINATURAS DIGITAIS NA ICP-BRASIL - DOC-ICP-15.02*. Instituto Nacional de Tecnologia da Informação, abril 2010a. Disponível em: <www.iti.gov.br/twiki/pub/Certificacao/DocIcp/DOC-ICP-15.02.pdf>. Acesso em: 05 jun 2011.

BRASIL. *REQUISITOS DAS POLÍTICAS DE ASSINATURA DIGITAL NA ICP-BRASIL - DOC-ICP-15.03*. Instituto Nacional de Tecnologia da Informação, abril 2010b. Disponível em: <www.iti.gov.br/twiki/pub/Certificacao/DocIcp/DOC-ICP-15.03.pdf>. Acesso em: 05 jun 2011.

BRASIL. *Requisitos Mínimos para Geração e Verificação de Assinaturas Digitais na ICP-Brasil - DOC-ICP-15.01*. Instituto Nacional de Tecnologia da Informação, abril 2010c. Disponível em: <www.iti.gov.br/twiki/pub/Certificacao/DocIcp/DOC-ICP-15.01.pdf>. Acesso em: 05 jun 2011.

BRASIL. *Visão Geral sobre Assinaturas Digitais na ICP-Brasil - DOC-ICP-15*. Instituto Nacional de Tecnologia da Informação, abril 2010d. Disponível em: <www.iti.gov.br/twiki/pub/Certificacao/DocIcp/DOC-ICP-15.pdf>. Acesso em: 05 jun 2011.

Donald Eastlake, J. R. D. S. *XML-Signature Syntax and Processing*. Fevereiro 2002. Disponível em: <http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/>. Acesso em: 25 jun 2011.

FERGUSON, N.; SCHNEIER, B.; KOHNO, T. *Cryptography Engineering: Design Principles and Practical Applications*. [S.l.]: Wiley Publishing, 2010.

HARRIS, S. *CISSP All-in-One Exam Guide*. 5. ed. [S.l.]: Mc Graw Hill, 2010.

HOFFMAN, P. *Enhanced Security Services for S/MIME*. IETF, jun. 1999. RFC 2634 (Proposed Standard). (Request for Comments, 2634). Updated by RFC 5035. Disponível em: <http://www.ietf.org/rfc/rfc2634.txt>.

HOUSLEY, R. *Cryptographic Message Syntax (CMS)*. IETF, jul. 2004. RFC 3852 (Proposed Standard). (Request for Comments, 3852). Updated by RFCs 4853, 5083. Disponível em: <http://www.ietf.org/rfc/rfc3852.txt>.

MENEZES, A. J.; OORSCHOT, P. C. van; VANSTONE, S. A. *Handbook of Applied Cryptography*. 5. ed. [S.l.]: CRC Press, 2001.

Oracle Technology Network. *Moving Java Forward*. 2011a. Disponível em: <http://www.oracle.com/br/technologies/java/index.html>. Acesso em: 18 jul 2011.

Oracle Technology Network. *Trail: The Reflection API*. 2011b. Disponível em: <http://download.oracle.com/javase/tutorial/reflect/>. Acesso em: 18 jul 2011.

PAAR, C.; PELZL, J.; PRENEEL, B. *Understanding Cryptography: A Textbook for Students and Practitioners*. [S.l.]: Springer, 2010.

PINKAS, D.; POPE, N.; ROSS, J. *CMS Advanced Electronic Signatures (CAdES)*. IETF, mar. 2008. RFC 5126 (Informational). (Request for Comments, 5126). Disponível em: <http://www.ietf.org/rfc/rfc5126.txt>.

ROSS, J.; PINKAS, D.; POPE, N. *Electronic Signature Policies*. IETF, set. 2001. RFC 3125 (Experimental). (Request for Comments, 3125). Disponível em: <http://www.ietf.org/rfc/rfc3125.txt>.

The Trac Project. *Welcome to the Trac Open Source Project*. 2011. Disponível em: <http://trac.edgewall.org/>. Acesso em: 18 jul 2011.