UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Marcelo Carlomagno Carlos, Rodrigo Muller Pons

Formulários Eletrônicos Seguros

Trabalho de conclusão submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciências da Computação.

Douglas Nascimento Rechia
Orientador

Prof. Ricardo Felipe Custódio, Dr.
Co-Orientador

Florianópolis, Novembro de 2004

Formulários Eletrônicos Seguros

Marcelo Carlomagno Carlos, Rodrigo Muller Pons

Este Trabalho de conclusão foi aprovado em sua forma final pelo Curso de Ciências da Computação da Universidade Federal de Santa Catarina

---

Prof. José Mazzuco Júnior, Dr.

Coordenador do Curso

Banca Examinadora

Douglas Nascimento Rechia

Orientador

Prof. Ricardo Felipe Custódio, Dr.

Co-Orientador

Jean Everson Martina

Prof. Júlio da Silva Dias, Dr.

”A morte do homem começa no instante em que ele desiste de aprender”. Albino Teixeira

Aos nossos pais que tornaram possível a realização deste trabalho.

Agradecimentos

Primeiramente ao orientador Douglas Rechia por nos auxiliar, tirar nossas dúvidas e pela paciência durante o desenvolvimento deste projeto.

Ao amigo Rodrigo Muller Pons, pelo esforço, dedicação e companheirismo durante a realização deste trabalho.

Ao professor Ricardo Felipe Custódio, pela amizade e orientação no projeto.

À minha namorada Vania, por seu companheirismo, amor, amizade e compreensão nos momentos de ausência.

Aos meus amigos que sempre estiveram ao meu lado durante mais essa jornada.

Ao amigo Luciano Ignaczak que sempre me incentivou e com quem muito aprendi na área de segurança em computação.

Aos meus familiares por todo apoio durante todo o curso.

(Marcelo Carlomagno Carlos)

Ao meu amigo Marcelo Carlomagno Carlos que esteve sempre a disposição para a realização deste trabalho. Você é um grande amigo e foi um prazer muito grande fazer este trabalho com você.

Ao orientador Douglas Nascimento Rechia por dedicar horas nos orientando e auxiliando, mostrando sempre o melhor caminho para que a gente desenvolvesse o nosso trabalho.

Ao professor Ricardo Felipe Custódio pela oportunidade de desenvolver este trabalho e poder conhecer melhor este grande professor.

A minha namorada Giselle por estar sempre ao meu lado me compreendendo e me apoiando em todos os momentos de minha vida.

Aos meus amigos por sempre me ajudarem a me manter alegre e disposto, mesmo nos momentos mais difíceis de minha vida.

Aos meus pais pelo imenso carinho que eles sempre tiveram comigo e pelos ensinamentos que eu levarei para o resto da minha vida.

Aos meus familiares por me apoiarem durante todo o curso.

(Rodrigo Muller Pons)

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

PKCS#7	Public-Key Cryptography Standards #7
PKCS#12	Public-Key Cryptography Standards #12
XFDL	Extensible Forms Description Language
XML	Extensible Markup Language
ASN.1	Abstract Syntax Notation One
HTML	HyperText Markup Language
SHA-1	Secure Hash Algorithm
W3C	World Wide Web Consortium
ASCII	American Standard Code for Information Interchange
SSL	Secure Socket Layer
TLS	Transport Layer Security
VPN	Virtual Private Network

Resumo

Os formulários eletrônicos devem apresentar características equivalentes aos formulários em papel para que a informação digital não venha a ser facilmente alterada e nem obtida por pessoas não autorizadas. Para solucionar tal problema escolheu-se a linguagem XFDL para a definição dos formulários, e a partir da aplicação de conceitos de segurança, como criptografia e assinatura digital, e utilizando a linguagem java, foi implementado um visualizador de formulários com funções que permitem garantir características como confidencialidade, integridade e autenticidade a um formulário eletrônico.

Palavras chave: Formulários Eletrônicos, XFDL, Segurança

Abstract

In order to protect electronic data against non-authorized people, electronic forms must have the same security features as paper-based forms. To achieve that goal, XFDL language is being used in this work to specify and define electronic forms. Applying security concepts, like cryptography and digital signatures, a XFDL-form viewer has been implemented in Java. So, confidentiality, integrity and autenticity are accomplished by the XFDL-forms tool developed in this work.

Keywords: Electronic Forms, XFDL, Security

Capítulo 1

Introdução

A popularização da informática, e o crescente uso da internet pelas mais variadas pessoas e empresas, faz com que o uso dos formulários eletrônicos aumente.

Os formulários eletrônicos são utilizados tanto para processos organizacionais, quanto para executivos, onde funcionários de empresas diferentes que possuem seus próprios sistemas informatizados, possam trocar informações. Entretanto, estas informações sigilosas não devem trafegar pela rede de modo que possam ser facilmente obtidas por pessoas não autorizadas. Portanto, não há como utilizar formulários eletrônicos sem questionar dois assuntos de fundamental importância, a segurança de informação e a validade jurídica.

Diante disso, deve-se adaptar tanto o conceito de documento, quanto a apresentação do documento digital para que este possua validade jurídica e sejam garantidos seus requisitos de segurança.

Nos capítulos subsequentes serão apresentados de forma mais clara e precisa os conceitos de Documentos Eletrôicos e formulários Eletrônicos

1.1 Objetivos

O objetivo deste trabalho é apresentar uma forma de dar a um formulário eletrônico características que façam com que este possua valor jurídico além de garan-

tir a confidencialidade dos dados contidos nele.

1.2 Objetivos Específicos

Desenvolver uma aplicação que garanta a um formulário eletrônico as seguintes características:

- Confidencialidade - Cifrar e decifrar um formulário, já que podem estar envolvidos neste formulários dados empresariais sigilosos, dados financeiros, processos de licitação, etc

- Autenticidade e integridade - a assinatura digital garante a autenticidade e a integridade do documento. É preciso haver confiança na identidade do autor do formulário e da pessoa que preencheu o formulário. Além disso, o formulário deve estar livre de erros de transmissão (caso venha a ser transmitido pela rede) e de alterações por qualquer pessoa que não seja aquele que assinou o documento.

- Envelopamento - para cifrar e decifrar um formulário eletrônico, realizar uma assinatura em grupo (co-assinatura) e detectar uma possível violação.

1.3 Justificativa

O compartilhamento e a troca de dados entre organizações e pessoas comumente não é realizado de forma segura. Isto leva as pessoas a não utilizarem o seu computador para realizar este tipo de operaçôes, fazendo isto então sempre pelo meio de formulários impressos em papel. Para que todos confiem na troca de dados pelo computador deve-se garantir as características já citadas anteriormente: confidencialidade, autenticidade, integridade, tempestividade, envelopamento.

A partir da aplicação dos conceitos de segurança aos formulários eletrônicos para garantir tais características, o seu uso tende a crescer muito, se tornando um

meio seguro, confiável e principalmente muito rápido e dinâmico para a troca de informações.

Segundo Júlio Dias (2004): ”O uso de documentos eletrônicos apresenta uma série de vantagens, como a agilização de processos e a facilidade de armazenamento e transmissão, levando a uma substancial redução de custos. Com isto, o uso do papel poderá ser reduzido, diminuindo gastos e sem diminuição da confiança por parte das organizações e pessoas envolvidas.”

1.4 Metodologia

O presente trabalho apresentará a implementação de visualizador de formulários, onde será possível a cifragem e assinatura dos dados garantindo as características desejadas para que o formulário eletrônico apresente as características de autenticidade, confidencialidade, integridade e não repúdio.Para isto seja possível, serão utilizados algoritmos de criptografia sobre o formulário e os dados que o preenchem.

1.5 Conteúdo do Trabalho

O capítulo 2 descreve os conceitos de documentos e formulários eletrônicos, tais como suas definições e quais características devem ser apresentadas em um formulário para que este apresente a validade jurídica de um documento. O capítulo 3 descreve os conceitos de segurança necessários para a criação de um documento xfdl seguro. O capítulo 4 descreve algumas ferramentas já disponíveis para resolver o problema. Finalmente, o capítulo 5 apresenta a solução para o problema abordado e sua implementação.

Capítulo 2

Documentos e Formulários Eletrônicos

Este capítulo descreve os conceitos de documentos e formulários eletrônicos. No decorrer do capítulo será discutido o que é necessário a um formulário eletrônico para que possua a validade de um documento eletrônico.

A seção 2.1 conceitua os documentos eletrônicos. A seção 2.2 conceitua os formulários eletrônicos. A seção 2.3 faz um comparativo entre os dados das seções 2.1 e 2.2. E finalmente a seção 2.4 demonstra a aplicação dos conceitos de segurança sobre os formulários eletrônicos

2.1 Documentos Eletrônicos

Segundo Costa (2003); ”o conceito de documento sempre esteve relacionado com a idéia de um escrito oficial, de uma informação fixada sobre um meio material. Isto porque por muito tempo o papel foi utilizado como o principal meio onde o conhecimento era registrado.”

Com o advento da informática, surge a necessidade de mudança no conceito de documento, já que este não é mais associado com um suporte material.[COS 03] A partir daí, surge o questionamento sobre quais características um documento eletrônico deve possuir para ter validade jurídica. Estas características, no entanto, são as mesmas necessárias para garantir a validade jurídica dos documentos em papel, como:[dSD 04]

- Integridade - certeza de que o conteúdo não foi alterado
- Autenticidade - confiança na identidade do autor
- Tempestividade - comprovação da extistência do documento em um determinada instante no tempo
- Não Repúdio - o autor não pode negar a autoria, nem o conhecimento do conteúdo do documento.
- Disponibilidade - possibilidade de verificar e acessar o documento

A forma de apresentação de um documento eletrônico é uma das maiores barreiras para que o documento eletrônico atenda tais características, pois este é formando por uma sequência de bits que necessita de auxílio de uma plataforma computacional para que seja exibido, desta forma, ocorre a separação entre o meio físico e o conteúdo do documento.[dSD 04]

Como exemplo de um documento pode-se citar uma certidão de nascimento. Uma certidão de nascimento para ser válida deve conter todos os dados necessários preenchidos, como nome da criança, nome dos pais e dos avós paternos e maternos, assinatura das pessoas envolvidas, a data de nascimento e a data em que esta sendo feita a certidão, nome e endereço do cartório, etc.

Somente com todos os dados devidamente preenchidos este documento poderá ter uma validade jurídica. Para que um documento eletrônico tenha esta mesma validade ele deve possuir todos os dados que possui uma certidão de nascimento em papel e garantir que este documento eletrônico não será modificado por ninguém depois de preenchido.

A grande dificuldade em se produzir documentos eletrônicos esta no fato de ter que garantir que o documento não será, em hipótese alguma, modificado posteriormente, coisa que em um documento em papel aparenta ser muito mais fácil pelo fato de se estar manipulando com papéis que quando modificados apresentam falhas que são visíveis ao olho humano. Para que seja possível a verificação da integridade de

um documento eletrônico, faz-se uso das funções resumo onde é calculado o resumo criptográfico do documento original, e após isso qualquer alteração no documento original pode ser dectectada a partir de uma nova aplicação desta função no documento e a comparação deste resultado com o resultado anterior. Mais detalhes sobre a função resumo serão descritos na seção 3.2.1

2.2 Formulários Eletrônicos

Tradicionalmente, formulários em papel são usados para estabelecer contratos, registros financeiros, transações comerciais, etc. Entretanto, com o aumento do uso da Internet, os formulários eletrônicos passaram a se difundir cada vez mais. Através de um simples navegador de internet, tornou-se possível preencher formulários através de campos de texto, caixas de verificação, consultar listas de opções, além de outras facilidades.

Estes formulários eletrônicos acabam deixando de lado a necessidade do uso do papel para armazenar e coletar todo o tipo de informação. Além disso, os formulários eletrônicos oferecem grandes vantagens em relação ao papel, como a validação instantânea dos dados, campos obrigatórios, consulta a lista de opções, etc. Outra vantagem é a facilidade de envio das informações, sendo possível o envio instantâneo a qualquer lugar do mundo.

Com o objetivo de proporcionar validade jurídica aos formulários eletrônicos diversos tipos de linguagens estão sendo desenvolvidas. A grande meta destas linguagens é se tornarem um padrão mundial para a criação de formulários eletrônicos.

Para desenvolver um formulário eletrônico deve-se utilizar uma linguagem que possibilite incluir todos os elementos fundamentais para a construção deste. Dentre estes principais elementos destacam-se:

- Campos de texto - onde será informada textualmente alguma informação, como nome ou endereço, por exemplo;

- Campos de múltipla seleção - onde serão selecionadas as informações necessárias, como preferências do usuário;
- Campos de seleção única - onde será informada apenas uma das opções apresentadas pelo formulário, como o sexo da pessoa;
- Lista de seleção - onde é selecionada uma das informações que estão contidas na lista, etc.

Um formulário eletrônico deve garantir a confidencialidade dos dados que foram inseridos nele. O documento eletrônico, isto é, o formulário, deverá ser acessado somente por pessoas autorizadas, especialmente depois de preenchido. Isto porque os dados que são inseridos em um formulário ,especialmente informações pessoais como número de identidade, CPF, número do cartão de crédito, se forem interceptados por pessoas mal intencionadas podem causar um grande prejuízo para a pessoa envolvida.

Algumas soluções existentes para prover a confidencialidade das informações que são enviadas de um simples formulário HTML para um servidor são Secure Socket Layer (SSL), Transport Layer Security (TLS) ou Virtual Private Network (VPNs). Porém estes mecanismos apenas provém segurança no canal de comunicação. Depois de armazenada a informação ficará desprotegida.

A confidencialidade de formulários eletrônicos (assim como a confidencialidade de documentos eletrônicos em geral) envolve não apenas evitar que a informação não seja acessada por pessoas não autorizadas. Os seguintes requisitos em específicos foram identificados para os formulários eletrônicos:

Cifrar dados Indispensável para que somente as pessoas autorizadas acessem os dados do formulário. Trata-se de uma característica muito importante de um documento, já que frequentemente a troca de informações e dados deve ser sigilosa.

Enviar dados para o futuro Em alguns casos, onde é necessário que o forumlário seja aberto somente em uma data futura, esta característica se torna muito importante. Um exemplo de seu uso é em aplicações de licitação onde são enviadas

propostas de orçamento para a execução de uma obra, e é essencial que o formulário cifrado seja aberto ,isto é, decifrado, somente em uma data futura.

Abertura do formulário em grupo Outra característica bastante importante em algumas situações onde é necessário que todos os participantes de um processo estejam presentes no momento da leitura dos documentos. Utilizando os documentos em papel, o conteúdo é colocado em envelopes lacrados e abertos somente na presença de todos. Nos formulários eletrônicos deve ocorrer da mesma maneira, eles devem ser decifrados somente na presença de todos os participantes.

Assinar dados Muito importante para garantir a autenticidade do documento e também sua integridade. Um documento assinado garante o não repúdio por parte do assinante, desta forma, a assinatura pode ser usada com sucesso para preencher este requisito do documento eletrônico válido juridicamente.

2.3 Comparação

Para que um formulário eletrônico seja considerado um documento eletrônico, basta que ele possua as características de integridade, autenticidade, tempestividade, não repúdio e disponibilidade que são as mesmas caracteríticas apresentadas pelo documentos.

Neste trabalho, adicionaremos a característica de confidencialidade aos formulários eletrônicos, tendo em vista que as informações contidas em um formulário são frequentemente sigilosas, aumentando assim a confiança em seu uso.

Após estes esclarecimentos verifica-se que um formulário se torna um documento quando ele está devidamente preenchido e garante todas as características necessárias para que se possa ter validade jurídica.

Voltando ao exemplo da certidão de nascimento pode-se verificar que uma certidão não preenchida não passa de um formulário e que somente após estar devidamente preenchido este formulário se tornará um documento, no caso específico, uma

certidão.

2.4 Conclusão

A partir das definições de documentos e formulários eletrônicos, conclui-se que aplicando técnicas conhecidas de segurança da informação, pode se garantir as características necessárias para validade de um formulário.

A questão da confidencialidade, é tratada a partir do uso de criptografia (cifrar/decifrar dados), que será vista com mais detalhes no capítulo 5. Já a autenticidade e a integridade, são tratadas com o uso de assinatura digital, que também serã vista com mais detalhes no capítulo 3.

Capítulo 3

Segurança

A troca de informações sigilosas no meio digital tem se tornado um dos problemas principais no mundo atual. Varios sistemas, como a web por exemplo, tinham como objetivo inicial apenas manter conectado um grupo de pessoas com as quais todos pudessem trocar informações entre si, não foi projetado nenhum sistema de segurança que permitisse um envio seguro de informações.

Hoje em dia circulam, principalemente através da internet, informações extremamente sigilosas, como dados bancários por exemplo, e com isso tornou-se necessária a criação de mecanismos que garantam a segurança e a integridade destes dados.

Este capítulo aborda as técnicas de criptografia e os serviços por ela oferecidos. A criptografia é uma das ferramentas mais adequadas para garantir a segurança de dados e comunicações.[IGN 02] Pode-se definir a criptografia como uma técnica de tornar uma informação inteligível em algo ilegível e incompreensível para qualquer pessoa que não esteja autorizada a ler esta informação.

A seção 3.1 conceitua a criptografia e seus tipos existentes. A seção 3.2 conceitua a assinatura digital mostrando como ela é feita e quais formatos utilizados nos formulários eletrônicos. A seção 3.3 conceitua os certificados digitais. E finalmente a seção 3.4 apresenta as técnicas de segurança que serão utilizadas no formulário eletrônico.

3.1 Criptografia

Criptografia é a arte de escrever em cifras ou em códigos, a fim de transformar uma informação inteligível numa forma aparentemente ilegível, através de um processo chamado cifragem, fazendo com que apenas o destinatário desejado consiga decodificar e ler a mensagem com clareza, no processo inverso chamado decifragem.[TRI 98]

A criptografia é um mecanismo de segurança que permite a implementação de diversos serviços, como a autenticação, o não-repúdio, a integridade e a confidencialidade. Ela pode ser classificada em duas categorias, de acordo com o tipo de chave utilizada:

- Criptografia simétrica;
- Criptografia assimétrica.

3.1.1 Criptografia Simétrica

A criptografia simétrica foi a primeira forma conhecida para ocultação de dados. Uma técnica de criptografia simétrica foi utilizada por Julio César, na Roma antiga, para enviar mensagens seguras para seus exércitos.[STA 99]

A criptografia simétrica tem como principal característica à utilização de somente uma chave para autenticar e garantir a confidencialidade de uma mensagem. Esta chave é denominada de chave simétrica ou chave secreta, e é através dela que se torna possível o ato de cifrar ou decifrar um texto que depois será transmitido para uma ou mais pessoas.

A figura 3.1 apresenta o processo de cifragem e decifragem de um documento.

A grande vantagem desta técnica é o fato de ela ser geralmente mais rápida que a criptografia por chave assimétrica ou chave pública. Dentre as principais desvantagens destacam-se:

- O envio desta chave entre as pessoas deve ocorrer em algum meio extremamente seguro, o que nem sempre é possível de se garantir;

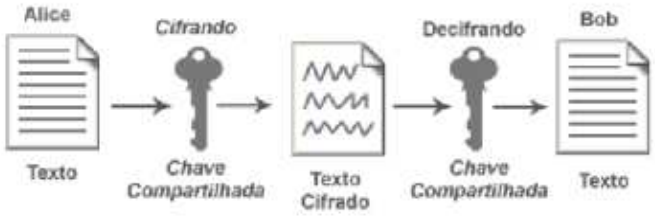

Figura 3.1: Figura que representa um processo de cifragem de dados de forma simétrica

- Este sistema não permite que se garanta a autenticidade de uma mensagem, pelo fato de que qualquer pessoa que detiver a chave simétrica poderá ter cifrado a mensagem em questão.

3.1.2 Criptografia Assimétrica

A criptografia assimétrica ou de chave pública caracteriza-se por utilizar duas chaves diferentes, sendo uma privada e outra pública. Um texto cifrado com uma das chaves pode somente ser decifrado utilizando a outra chave correspondente.[TRI 98]

A garantia da confiança está no fato que cada usuário deverá gerar seu próprio par de chaves. A chave privada deve ser mantida em segredo e não deve ser utilizada por ninguém, exceto pela pessoa a qual ela pertence. A chave pública, ao contrário, deve ser disponibilizada para ser utilizada por qualquer aplicação ou indivíduo.[IGN 02]

A grande vantagem deste sistema é permitir que qualquer um possa enviar uma mensagem secreta a uma pessoa apenas utilizando a chave pública, correspondente a chave privada de quem irá recebê-la. Como a chave pública está amplamente disponível, não há necessidade do envio de chaves em meios seguros como é feito no modelo simétrico. A confidencialidade da mensagem é garantida, enquanto a chave privada estiver segura.

A criptografia assimétrica permite a utilização de dois tipos de serviços.

- A assinatura digital, onde após cifrada uma mensagem com uma chave privada o autor da mensagem não poderá repudiar a mesma;

- E o sigilo, onde se utiliza a chave pública do destinatário para cifrar uma mensagem e o receptor deve utilizar sua chave privada para a decifrá-la.

A figura 3.2 apresenta o processo de cifragem e decifragem de um documento.

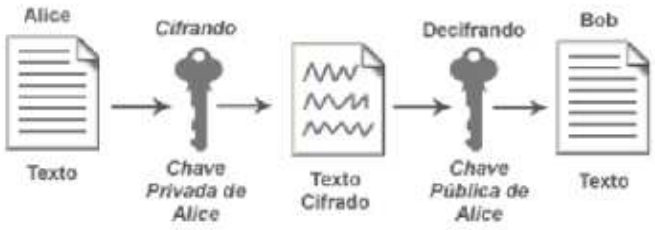

Figura 3.2: Figura que representa um processo de cifragem de dados de forma assimétrica

3.2 Assinatura Digital

A assinatura digital é utilizada para verificar a autenticidade e a integridade de dados além de garantir no não repúdio por parte do assinante. Para a criação da assinatura digital de uma mensagem, o indivíduo deve utilizar sua chave privada juntamente ao resultado da função resumo da mensagem. No decorrer desta seção este processo será descrito mais detalhadamente.

3.2.1 Função Resumo

Função Resumo é uma função (*Hash*) que recebe como entrada uma mensagem de qualquer tamanho e produz um resumo de tamanho fixo. O tamanho da saída

varia de acordo com o algoritmo usado. O propósito de uma função resumo é produzir uma ”impressão digital” da mensagem[STA 99].

Uma das características das funções resumo é o fato de as funções serem de sentido único, ou seja, a mensagem não pode ser determinada através do seu resumo.

Outra característica é a possibilidade de se garantir a integridade de uma mensagem, pois se alguma parte dela for alterada, quando a Função Resumo for aplicada novamente na mensagem seu resultado será muito diferente, detectando dessa forma a alteração em seu conteúdo.

3.2.2 Gerando uma assinatura digital

Para gerar uma assinatura digital inicialmente é executado o processo para criação do resumo da mensagem. Após isso, o resumo é cifrado utilizando a chave privada do assinante. O resultado é desta operação, ou seja, o resumo cifrado da mensagem é a assinatura digital.

O processo para verificação da assinatura digital é realizado pelo usuário que recebeu a mensagem criando o resumo da mensagem. Depois de criado o resumo da mensagem, ele decifra o resumo que foi enviado juntamente com a mensagem utilizando a chave pública do emissor. Se o resultado da nova função resumo for idêntico ao resumo decifrado, é possível garantir a integridade da mensagem e assegurar que os dados foram assinados pela chave privada correspondente aquela chave pública.[IGN 02]

É importante destacar que a assinatura digital, como descrita no exemplo anterior, não garante a confidencialidade da mensagem;

3.3 Certificados Digitais

Um certificado digital pode ser definido como um documento eletrônico, assinado digitalmente por uma terceira parte confiável, que associa o nome (e atributos) de uma pessoa ou instituição a uma chave criptográfica pública.[FER 01]

São os certificados digitais que se responsabilizam por tornarem possíveis

as transações e comunicações sem que haja riscos entre as partes envolvidas.

Um certificado digital é formado, basicamente, pelos seguintes campos: versão do certificado, número serial, algoritmo de assinatura, dados do emissor, validade, dados do sujeito, informações da chave pública, identificador do emissor, identificador do sujeito.

Os certificados digitais possuem um tempo de vida formado pela data de início da validade do certificado e a data que o certificado deixará de ser válido. Um certificado necessita ter um prazo de validade devido a evolução dos dispositivos de processamento. O certificado possui uma chave com um tamanho definido, determinado por um número de bits. Com o passar do tempo, o tamanho da chave contida no certificado pode ser considerado fraco, ficando sujeita a ser quebrada por computadores com alto poder de processamento.[NOT 02]

3.4 Conclusão

Este capítulo apresentou as técnicas de segurança necessárias para o desenvolvimento do projeto. Tais técnicas são imprescindíveis para o restante do projeto, já que seu uso torna possível garantir confidencialidade, autenticidade e integridade dos formulários eletrônicos, o que é o escopo principal deste documento.

Capítulo 4

Linguagens para definição de formulários

Este capítulo apresenta algumas linguagens mais utilizadas para a definição de formulários eletrônicos. Escolheu-se como objeto de estudo tais linguagens a partir de uma seleção baseada em critérios como a sua utilização, seu suporte a cifragem de dados e assinatura digital. Para cada linguagem apresentada no decorrer do capítulo, será apresentada uma visão geral de sua estrutura e logo após serão apresentadas formas de aplicação dos conceitos de segurança sobre elas.

A seção 4.1 apresenta a mais tradicional forma de apresentação de formulários eletrônicos, o formato html. A seção 4.2 apresenta a definição dos XForms. A seção 4.3 conceitua o XFDL. E finalmente a seção 4.4 apresenta a solução escolhida.

4.1 HTML

O HTML é a linguagem utilizada para publicar hipertexto na Internet. É um formato não proprietário e pode ser criado e processado por uma grande variedade de ferramentas, desde simples editores de texto, até as mais avançadas ferramentas de desenvolvimento. A linguagem baseia-se no conceito de tags, que organizadas de forma estruturada, definem o conteúdo do documento.[W3C]

Os formulários HTML são uma das primeiras representações de formulários eletrônicos, e ainda hoje é o formato mais utilizado. Basicamente, formulários HTML eram usados como formas de permitir que os visitantes de um website se comunicassem com seus desenvolvedores, mas com o passar dos anos, os formularios, aliados a bancos de dados, passaram a ter também funções de cadastro de clientes, troca de informações, serviços de atendimento e até mesmo compra e venda de produtos. A figura 4.1 apresenta um exemplo bastante simplificado de um formulário HTML.

```
01: <html>
02: <head><title>Busca</title></head>
03: <body>
04: <form action="http://www.exemplo.com/busca" method="get">
05: Buscar: <br>
06: <input type="text" name="q">
07: <input type="submit" value="Ir">
08: </form>
09: </body>
10: </html>
```

Figura 4.1: Um simples formulário html.

A questão segurança é bastante enfatizada, principalmente, quando imagina-se a possibilidade de se ter suas informações expostas a atacantes ou intrusos da Internet, que surgem com meios cada vez mais sofisticados para violar a privacidade e a segurança das comunicações[TRI 98]. Com o aumento do uso dos formulários HTML, cada vez mais dados sigilosos são expostos a este meio, e por não prover muitas características desejadas a um documento seguro, o formulário HTML inspira muita desconfiança à seus usuários.

Atualmente a cifragem de dados de documentos html está restrita ao canal

de comunicação que pode ser SSL. Um formulário armazenado em disco, ou fora deste meio de comunicação não apresenta nenhuma proteção em relação a segurança das informações contidas nele. A assinatura digital também é bastante crítica, já que o HTML não foi originalmente projetado para prover tal caracteristica.

4.2 XForms

O XForms é uma ”linguagem” integrada ao XML originalmente criada para projetar formulários para web, mas já é encontrada em outras áreas também, como no processamento de documentos.

A grande vantagem do uso de documentos baseados em XML é que através de um processo totalmente eletrônico, muitas informações incorretas são eliminadas, fazendo com que os dados dos formulários sejam tratados e verificados antes de serem armazenados em um banco de dados.[O'R 03]

Um importante conceito dos XForms é que eles coletam dados no formato XML, dessa forma, esses formulários representam uma forma estruturada de troca de dados. Através dele é possível verificar quais campos do formulário foram preenchidos, verificar se uma data é anterior a uma outra, etc.

Apesar de à uma primeira vista ser mais complicado do que um formulário html, um XForm é tão simples quanto, tanto para se implementar quanto em sua visualização. A figura 4.2 mostra um exemplo de como o formulário HTML apresentado na figura 4.1 é representado XForm.

A grandes vantagens dos XForms em relação ao HTML são:[DUB 03b]

- Verifica os valores dos campos de dados no momento em que o usuário está preenchendo o campo.
- Não permite que o formulário seja submetido se os campos obrigatórios não tiverem sido preenchidos
- Submeter dados dos formulários como um XML.

```
01: <h:html xmlns:h="http://www.w3.org/1999/xhtml"
02:         xmlns="http://www.w3.org/2002/xforms">
03: <h:head>
04: <h:title>Busca</h:title>
05: <model>
06: <submission action="http://www.examplo.com/busca"
07:         method="get" id="s"/>
08: </model>
09: </h:head>
10: <h:body>
11: <h:p><input ref="q"><label>Buscar</label></input>
12: <submit submission="s"><label>Ir</label></submit>
13: </h:p>
14: </h:body>
15: </h:html>
```

Figura 4.2: O mesmo formulário em XForms.

- Submeter o mesmo formulário para diferentes servidores.
- Usar o resultado de um formulário e submeter o resultado como uma entrada para um outro formulário.
- A apresentação e os dados são apresentados de forma separada.

Apesar de todas as vantagens apresentadas pelos XForms, ele não foi originalmente projetado para suportar assinatura digital, embora seja possível que sejam adicionadas extensões ao XForms para prover esta característica. Mesmo sendo possível extender o XForms para prover assinatura digital, algumas desvantagens e características indesejadas podem ser verificadas:

- A facilidade de criar extensões muitas vezes pode ser considerada uma vantagem, mas também possui um lado negativo. Tal facilidade permite que existam duas diferentes implementações para a mesma funcionalidade, fazendo com que não se possam especificar padrões para exbição e verificação dessas extensões.[DUB 03a]

- Aplicações XML normalmente ignoram elementos e atributos desconhecidos, o que em alguns casos podem causar grandes problemas.[DUB 03a]

- A separação dos dados e da apresentação nos XForms é outro problema a ser tratado, já que uma característica das assinaturas digitais é de se ”assinar o que se vê”, e tal separação permite diferentes apresentações dos dados de acordo com a apresentação. Assinar a apresentação e os dados é possível, mas exige uma série de adaptações ao modelo original.

- O formato dos formulário exibido na tela e o conteúdo enviado ao servidor após a submissão do formulário é diferente, apenas os dados são enviados.

Quanto a confidencialidade, por se tratar de um arquivo xml, um XForm pode ser cifrado seguindo os padrões definidos pela W3C.[REA 02]. Outra forma de prover a confidencialidade é através do canal de comunicação SSL, que funciona de maneira semelhante a transmissão dos dados de um HTML no mesmo canal.

4.3 XFDL

O XFDL(Extensible Forms Description Language) é uma linguagem que descreve uma sintaxe de XML adequada para desenvolver formulários eletrônicos. A finalidade do XFDL é permitir a expressão de formulários poderosos e complexos em uma sintaxe que promova a interoperabilidade da aplicação e a aderência aos padrões da Internet.

Com o objetivo de proporcionar validade jurídica aos formulários eletrônicos, o *World Wide Web Consortium* (W3C) publicou em 1998 o *draft* da especificação *Ex-*

tended Forms Definition Language (XFDL). Segundo este *draft*, o XFDL é uma linguagem projetada para criar formulários com os seguintes objetivos:[JB 98]

- Um formulário deve ser um objeto único, sem depender de nenhum recurso externo.
- A linguagem XFDL deve ser legível aos olhos humanos.
- O XFDL deve ser um padrão aberto e disponível publicamente.
- Deve prover uma sintaxe capaz de suportar o cálculo de expressões matemáticas e condicionais.
- Dados binários poderão ser encapsulados no formulário desde que sejam codificados em base-64.
- A disposição dos componentes que compõem o formulário (texto, caixas de texto, etc) deve ser precisamente definida de forma que a visualizaçãoo e impressão do documento seja fiel.
- A validação e formatação dos dados na máquina cliente deve reduzir o processamento e a carga de trabalho do servidor.
- Elementos definidos pelo usuário poderão ser adicionados a linguagem, como o código de funções externas, itens e opçõees personalizadas.
- Oferecer suporte à assinatura digital.

Formulários criados com a linguagem XFDL são documentos com validade jurídica, pois possuem as seguintes características: segurança[1], não-repúdio e auditabilidade. [BLA 99].

[1]Segundo o autor, a "segurança"é conseguida através de autorização do assinante, autenticação do documento, e autenticação do assinante.

Apesar de serem válidos no sentido legal, formulários escritos em XFDL não possuem confidencialidade. Isto porque os dados contidos em um formulário desenvolvido em XFDL não são criptografados e podem ser lidos por qualquer tipo de usuário.

Outra característica interessante do XFDL é que ele não é voltado apenas para a web como o HTML e o XForms. Um formulário XFDL pode ser utilizado, preenchido e visualizado localmente, sem a necessidade de um servidor web para esta função. Além disso ele pode sem mantido em disco sem maiores problemas.

4.3.1 Fundamentos da linguagem XFDL

Segundo John Boyer (1998), um formulário escrito em XFDL é um documento XML que possui um elemento raiz que o caracteriza como do tipo XFDL. Um formulário escrito em XFDL apresenta dois atributos característicos que são o version e o sid. O atributo version especifica a versão do XFDL que está sendo usada no documento atual. O atributo sid é um identificador de escopo (scope identifier). Apenas o atributo version é obrigatório. A figura abaixo mostra um exemplo de um formulário XFDL

O elemento XFDL é sucedido por elementos de opções globais de página. Na sequência, haverá pelo menos um elemento page que conterá os itens de interface que constituirão o formulário. A linguagem XFDL permite mais de um elemento page por arquivo, mas no entanto, apenas uma página é mostrada por vez.

4.3.2 Suporte do XFDL a segurança

Entre as linguagens apresentadas neste capítulo, a linguagem XFDL é a única que possui nativamente suporte à assinatura digital. Através do elemento signature, um elemento contido no próprio documento pode-se adicionar a ssinatura do documento e também as informações do assinante. Com isso, um formulário XFDL já possui uma maneira de prover não-repúdio, autenticidade e integridade da informação apresentada por ele.

```
01: <?xml version="1.0"?>
02:  <XFDL version="4.1.0">
03:    <bgcolor content="array">
04:        <ae>128</ae> <ae>128</ae> <ae>128</ae>
05:    </bgcolor>
06:    <page sid="Pythagorean_Theorem">
07:        <bgcolor content="array">
08:            <ae>192</ae> <ae>192</ae><ae>192</ae>
09:        </bgcolor>
10: ....
11:    </page>
12:  </XFDL>
```

Figura 4.3: Um formulário simples em XFDL.

Uma assinatura contém além da própria assinatura, todos os dados necessários para a verificação da autenticidade do formulário assinado. A assinatura deverá ser criada pelo programa visualizador do formulário. O usuário assinará o formulário utilizando um botão para este propósito.

Algumas das opções disponíveis ao elemento signature são:

- signformat - especifica o tipo de codificação que o visualizador do formulário deverá usar para criar a assinatura na opção mimedata;
- signer - identifica quem assinou o formulário;
- signature - atribui um nome a assinatura; este nome será usado para referenciar esta assinatura;
- signitemrefs - especifica quais elementos do formulário serão assinados; esta opção é útil para que o usuário possa escolher quais partes do formulário ele

deseja assinar;

- mimedata - contém o resultado da função hash cifrado com a chave privada do assinante; o mimedata é codificado usando base-64.

Um formulário XFDL pode conter zero ou mais assinaturas, sendo que a cada nova assinatura, é inserido um novo elemente signature com as informações do novo assinante.

Apesar de ser o único apresentado a prover assinatura em sua especificação, o XFDL não possui nenhum mecanismo para prover a confidencialidade da informação. Todos os dados contidos em um formulário desenvolvido em XFDL não são cifrados e estão disponíveis para qualquer pessoa. Entretanto, por se tratar de um documento XML pode-se utilizar os padrões definidos pela W3C para cifragem de documentos XML.

4.4 Conclusão

O conteúdo deste capítulo é de fundamental importância para a implementação do projeto. A linguagem escolhida a partir da análise feita no decorrer do capítulo deve ser a que melhor apresenta as características de prover validade do formulário e também a confidencialidade.

A tabela 4.4 apresenta um quadro comparativo entre as linguagens e suas respectivas funcionalidades relevantes ao projeto.

Ao analisar todas as características das linguagem apresentadas, escolheu-se o XFDL para a definição dos formulários deste trabalho pelo fato dela suportar assinaturas em sua especificação, sem necessidade de criar extensões e modificações em sua estrutura. Além disso, por se apresentar no formato XML, o que facilita sua manipulação, exibição e validação.

Apesar de serem válidos no sentido legal, formulários escritos em XFDL não possuem confidencialidade. Para que se possa prover tal característica, fez-se uso

	Assinatura Digital	Cifragem dos dados
HTML	Não possui suporte em sua especificação	Apenas no canal de comunicação
XForms	Não possui suporte em sua especificação	No canal de comunicação e pode também ser criada um extensão para esta finalidade
XFDL	Possui suporte	No canal de comunicação e pode também ser criada um extensão para esta finalidade

Tabela 4.1: Tabela comparativa entre as linguagens

do fato do XFDL ser um documento XML, dessa forma, serão seguidos os padrões definidos pela W3C para o xml para a cifragem dos dados XML.

Capítulo 5

Solução e Implementação

A partir dos conceitos de documentos eletrônicos, segurança, das linguagens em uso e das necessidades apresentadas para garantir a validade jurídica e confidencialidade de um formulário eletrônico decidiu-se realizar a implementação de um visualizador de formulários eletrônicos, através do qual será possível seu preenchimento, assinatura e cifragem dos dados.

5.1 Tecnologias Utilizadas

5.1.1 Java

A linguagem escolhida foi o Java, por apresentar uma série de características desejáveis à aplicação:

- É multiplataforma, fazendo com que a aplicação seja portável à diferentes sistemas operacionais.
- É totalmente orientada a objetos, fazendo com que as etapas de implementação, atualização e manutenção sejam mais eficientes e organizadas.
- Possui classes bastante utilizadas e testadas no pacote java.security o que torna a aplicação mais confiável e de tamanho reduzido por usar várias classes nativas.

- O grande número de desenvolvedores nessa linguagem, facilitando a discussão sobre dúvidas e conceitos nas mais variadas listas.

A versão do Java utilizada foi a 1.4.2, pois apresenta melhorias principalmente no pacote java.security em relação as versões anteriores.

5.1.1.1 Bibliotecas

Para a implementação de algumas funcionalidades da aplicação, houve a necessidade de utilização de biblioteca auxiliares. Abaixo segue uma breve descrição dessas bibliotecas além de sua finalidade para o projeto.

- **BouncyCastle** - biblioteca que fornece um conjunto de classes muito bem organizadas e estruturadas para prover segurança a aplicações, permitir codificação e decodificação de dados no formato ASN.1 entre outras funcionalidades. . Sua principal finalidade neste trabalho, foi utilizá-la para a criação da assinatura digital no formato PKCS#7, e sua respectiva verificação. [BOU]

- **Apache XML-Security** - biblioteca que fornece classes para a criação de documentos XML cifrados de acordo com as normas definidas pela W3C. Foi muito útil neste projeto para a implementação da cifragem dos dados do formulário XFDL. [APA]

- **DOM4J** - biblioteca que fornece classes para a manipulacao de arquivos xml.[DOM]

5.1.2 XML

Extensible Markup Language (XML) é linguagem de marcação de dados (meta-markup language) que provê um formato para descrever dados estruturados. Isso facilita declarações mais precisas do conteúdo e resultados mais significativos em sua aplicação em múltiplas plataformas.

O XML é um elemento fundamental no desenvolvimento deste projeto, já o XFDL, utilizado para a definição dos formulários, é estruturado no formato XML.

Toda a parte de manipulação do conteúdo dos dos formulários, sua exibição, e todas as funções relacionadas com criptografia e assinatura digital se dão pela utilização deste formato.

5.1.3 XFDL

Para a definição do formulário será utilizado o XFDL (Extensible Forms Description Language) [JB 98] (mais detalhes são descritos no capítulo 4.3), pois é apresentado no formato XML, o que o torna bastante portável, além de já prover suporte à assinaturas digitais.

Outra vantagem da utilizacçao desta linguagem, é que por ser no formato XML, questão a confidencialidade fica mais simples de ser criada, pois o conteúdo será cifrado de acordo com as especificações da W3C para cifrar XML[REA 02].

5.2 Projeto

A partir da análise de todos os requisitos necessários para a elaboração do projeto, foi feito um levantamento sobre o que a aplicação deve conter. Nesta seção será descrita detalhadamente a aplicação, bem como suas funcionalidades.

5.2.1 Requisitos

O sistema deve consistir em um programa com as opções de abrir e salvar formulários, cifrar e decifrar, além de assinar e verificar assinatura. Todas essas opções ficarão disponíveis em ítens de menu acessíveis pelo usuário.

Os requisitos para que a aplicação possua as caracteríticas desejadas são:

- Visualizador de documentos XFDL: deve permitir a exibição do conteúdo do formulário de forma amigável ao usuário, além de permitir o preenchimento deste formulário.

- Assinador de documentos XFDL: deve permitir a assinatura digital de forma bastante fácil, simples e eficiente, além de que a partir de um documento já assinado, permita a verificação da assinatura, e novas assinaturas (co-assinatura).

- Cifrador de documentos XFDL: deve permitir a cifragem dos dados do documento, permitindo que nem os dados contidos no formulário, nem sua estrutura seja visualizada por terceiros. Além disso, deve permitir decifrar o conteúdo cifrado.

Tais requisitos deve sem obrigatóriamente preenchidos pela aplicação, desta forma, a ferramenta será bastante completa e funcional para as finalidades desejadas.

5.2.2 Comportamento do sistema

Para definir o comportamento do sistema foram definidos casos de uso (*Use Cases*), que podem ser vistos mais detalhadamente no Apêndice A. Com base nestes casos de uso, este capítulo descreverá como deve se comportar o sistema.

5.2.2.1 Visão Geral

O sistema consistirá em um programa com as opções descritas anteriormente. Dessa forma, serão alcançados os objetivos essenciais deste trabalho.

Para manter as características do documento XFDL, convencionou-se que deve ser obedecida uma certa ordem em relação aos processos de assinatura e cifragem de dados. Um documento XFDL não pode ser assinado depois de ser cifrado, pois quando cifrado, a estrutura do formulário XFDL não é apresentada em sua forma original, ou seja, o XFDL cifrado não apresenta a tag signature onde deve ser colocada a assinatura. Para se obter um documento assinado e cifrado, deve-se assinar e depois cifrar o documento. Já para a verificação da assinatura, deve-se primeiramente decifrar o documento (se cifrado) e depois verificar a assinatura.

A figura abaixo exemplifica as ações possíveis em um XFDL Seguro.

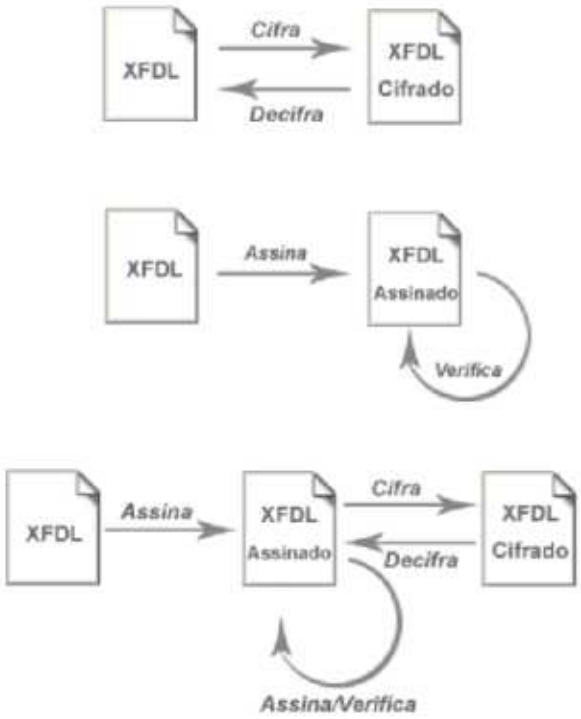

Figura 5.1: Diagrama que representa as possíveis ”transições” de um documento XFDL seguro

5.2.2.2 Visualizador

O visualizador de documentos XFDL será uma apliação com interface gráfica, onde os compomentes integrantes de um documento xfdl serão exibidos em modo gráfico.

Os campos do formulário, como caixas de texto, lista de opções, botões, etc, serão editáveis pelo usuário, ou seja, com o documento aberto no visualizador, tais campos podem ser preenchidos de acordo com os dados inseridos pelo usuário. Caso o formulário já tenha sido preenchido anteriormente, tais campos aparecem com os dados já preenchidos, e os valores dos campos podem ser alterados. Caso ocorra a alteração dos valores preenchidos em um documento já assinado, as assinaturas ante-

riores a modificação serão descartadas.

Ao clicar no botão salvar, a aplicação salva o estado atual[1] do formulário, permitindo que o preenchimento do formulário fique salvo.

5.2.2.3 Assinatura Digital

Ao abrir um documento XFDL a primeira ação que ocorre, antes mesmo da própria exibição do formulário, é a verificação se o documento está assinado ou não. Em caso positivo, todas as assinaturas são verificadas e em caso de alguma ser inválida, ocorre uma indicação na tela.

Na janela principal da aplicação, com o formulário aberto, a opção de verificar fica habilitada. Tal funcionalidade será utilizada para permitir a verificação da assinatura digital do documento.

Para assinar, o botão de assinatura deve estar especificado no conteúdo do documento XFDL. Caso esteja especificado, quando o usuário escolhe a opção Assinar, ou seja, clica no botão, abre-se uma janela de escolha de arquivo, onde deve ser escolhido um arquivo no formato PKCS#12 (.pfx), que conterá o certificado a ser utilizado para assinatura e também sua chave privada. Após escolhido o arquivo .pfx, é requisitada a senha do arquivo e da-se inicio ao processo de assinatura.

O processo de assinatura funciona da seguinte maneira:[BOY]

1. Verifica-se se já existem assinaturas anteriores à essa, ou seja busca-se por tags signature

2. Caso exista, as tags sao removidas temporariamente.

3. Gera-se a assinatura do documento no formato PKCS#7 a partir do resumo criptográfico do documento e a chave privada contida no arquivo .pfx.

4. As assinaturas antigas são recolocadas do documento e verificadas para ver se houve algum comprometimento durante o processo, ou se o documento foi alte-

[1] entende-se por estado atual as condições do preenchimento do formulário, como caixas de texto preenchidas, opções selecionadas, etc

rado. Caso alguma assinatura esteja comprometida, ela é descartada e é mostrado um aviso ao usuário.

5. A nova assinatura é codificada no formato BASE64[2] e inserida em uma nova tag signature no documento juntamente com alguns dados do assinante.

Com o processo de assinatura finalizado, o documento estará pronto para novas ações, como cifrar e verificar.

Quando o usuário escolhe a opção verificar, todas as assinaturas são verificadas,e ao final da verificação, fica disponível a ação de visualização dos dados dos certificados dos assinantes, bem como a data e hora da assinatura, algoritmo utilizado, etc.

O processo de verificação funciona da seguinte maneira:

1. Busca-se por todas as assinaturas do documento.

2. Para cada assinatura encontrada, pega-se o conteúdo do codificado em BASE64 da tag signature. A partir desse conteúdo, obtem-se o certificado do assinante, o hash do arquivo cifrado com a chave privada do assinante.

3. Pega-se todo o conteúdo do formulário (exceto as tags signature) e gera-se o resumo criptografico.

4. A partir dos dados extraidos da assinatura, extrai-se o hash, o qual deve ser decifrado com a chave pública contida no certificado do assinante.

5. Compara-se o hash gerado a partir do conteúdo do formlário, com o hash decifrado extraído de cada assinatura.

6. Ao final das comparações, se todas forem positivamente verificadas, a integridade do documento está correta. Caso ocorra erro em alguma verificação, mostra-se um aviso, indicando que alguma das assinaturas está inválida.

[2]O formato BASE64 é uma codificação para transferência de conteúdo que representa dados codificados na forma binárias em caracteres ASCII

Após a verificação, ficam também disponíveis todas as informações sobre os certificados dos assinantes, permitindo ao usuário visualizar esses dados.

A figura 5.2 apresenta o conteúdo da tag signature de um documento XFDL assinado.

```
01: <signature>
02: <signformat>application/x-xfdl</signformat>
03: <signer>Marcelo Carlomagno Carlos,
04: mcc\@inf.ufsc.br</signer>
05: <fullname>CN=Marcelo Carlomagno Carlos,
06: OU=LabSEC,O=UFSC,L=Florianopolis,ST=SC,C=BR,
07: EMAILADDRESS=mcc\@inf.ufsc.br</fullname>
08: <mimedata>
09: MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgU
10: ggPooAMCAQICChYWoVQAEQAABt8wDQYJKoZIhvcNAQE
11: ...
12: YSInGdH3ezk9C3FNJEVVwzzC4mMqyaMNZPRi5Kzzf9a
13: ggPooAMCAQICChYWoVQAEQAABt8wDQYJKoZIhvcNAQE
14: </mimedata>
15: </signature>
```

Figura 5.2: Documento XFDL Assinado

5.2.2.4 Confidencialidade

Para garantir a confidencialidade do conteúdo do formulário, serão utilizados algoritmos de criptografia simétrica. O XFDL inicialmente não oferece suporte a cifragem de dados, mas por se tratar de um arquivo XML será proposta e utilizada neste trabalho uma solução para garantir a cifragem dos dados.

Seguindo os padrões definidos pela W3C de cifragem de dados no XML (*XML Encryption*) [REA 02], o XFDL será cifrado como um arquivo XML qualquer, sendo gerada inicialmente uma chave para cifrar os dados.

Para decifrar os dados, basta seguir o processo inverso de cifragem do XML, além de fornecer a chave para que os dados do XFDL voltem a ser legiveis (decifrados).

Por se tratar de algoritmos de criptografia modernos e ser definida uma chave de tamanho suficiente para prover a segurança necessária, pode-se prover confidencialidade ao formulário sem maiores problemas.

Para realizar a cifragem do documento XFDL, propõe-se que seja cifrado o elemento raiz do documento. Desta forma, serão cifrados também todo seu subconjunto de elementos e seus dados. Como resultado, haverá dois principais elementos no documento, um com os dados cifrados da chave utilizada para cifrar o XFDL e outro com o conteúdo do formulário cifrado. Recomenda-se a cifragem do elemento raiz para que não fique visível nem a estrutura do formulário cifrado, maximizando a privacidade das informações.

Na janela principal da aplicação, com o formulário aberto, a opção de cifrar fica habilitada. Quando o usuário escolhe a opção Cifrar, dá-se inicio ao processo de cifragem dos dados. Tal processo de dá da seguinte forma:

1. É gerada uma chave de forma aleatória para cifrar os dados.
2. Escolhe-se o elemento raiz do documento XFDL para ser cifrado.
3. Cifra-se a o elemento e todo seu subconjunto de elementos a partir desta chave.
4. A chave é cifrada com a chave pública da pessoa a quem se deseja enviar o documento cifrado.
5. A chave cifrada, e o conteúdo são armazenados novamente no arquivo no formato XML, de acordo com o padrão definido pela W3C[REA 02], onde existem indicações do tipo de algoritmo utilzado, e a estrutura que indica onde está a chave cifrada e onde estão os dados.

6. O é armazenado em disco, com o nome do documento xfdl adicionado da extensão ".enc"

A figura 5.3 apresenta o conteúdo de um documento XFDL cifrado.

Para abrir um documento XFDL cifrado, basta selecionar a opção abrir da aplicação e escolher o arquivo cifrado. Neste momento será também solicitado o arquivo que contém a chave privada necessária para decifrar a chave do documento. Após escolhidos o arquivo cifrado e o arquivo que contém a chave privada (.pfx), a verificação se dá na forma inversa da cifragem, sendo salvo o arquivo original em disco com o mesmo nome do arquivo cifrado, mas com a extensão .xfdl

5.3 Implementação

Após a análise dos requisitos necessários, deu-se inicio ao processo de implementação da aplicação. A implementação foi divida em etapas, durante a primeira etapa, desenvolveu-se o cifrador de arquivos xfdl e em uma segunda etapa, foi desenvolvido paralelamente o visualizador de documentos e o assinador.

Para desenvolver o cifrador de formulários, foram implementadas classes para tratar do processo de cifragem dos dados e outras para decifrar. Tais classes utilizaram as bibliotecas Apache XML-Security e DOM4J, e seguiram os padroes definidos na etapa de projeto da aplicação.

Já para o assinador de formulários, foram implementadas classes genéricas para prover assinatura e verificação de dados, e a partir dessas classes implementou-se outras para realizar o processo de assinatura e verificação especificamente em arquivos XFDL. Nesta etapa, foi implementada também uma classe para acessar os dados de um Certificado Digital, facilitando dessa forma a obtenção dos dados do(s) assinate(s) do formulário. A biblioteca utilizada na implementação dessas classes foi a BouncyCastle, para a realização de assinatura e verificação, e a DOM4J para a manipulação do documento XFDL para que se possa inserir os dados da assinatura no documento e remover os dados referentes a assinaturas anteriores. Todos os dados que se apresenta-

vam no formato binário, foram codificados em BASE64 através do uso de uma classe específica para este propósito.

O visualizador de documentos XFDL foi um elemento da implementação que apresentou alto grau de complexidade para se desenvolver, pois trata-se da tradução dos elementos do arquivo XFDL para uma interface gráfica.

A partir de classes do pacote swing do java, além de classes adicionais desenvolvidas para caracterizar graficamente os elementos de um arquivo XFDL, desenvolveu-se o visualizador, onde os compomentes do formulários XFDL são exibidos.

Durante a etapa de implementação, buscou-se seguir fielmente o conteúdo descrito na etapa de projeto, e como produto final, obteve-se uma aplicação que preenche os requisitos desejados.

```
01: <XFDL sid="FORM1" version="4.1.0" xmlns="">
02: <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Content"
03: xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
04: <xenc:EncryptionMethod
05: Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
06: xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"/>
07: <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
08: <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
09: <xenc:EncryptionMethod
10: Algorithm="http://www.w3.org/2001/04/xmlenc#kw-tripledes"
11: xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"/>
12: <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
13: <xenc:CipherValuexmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
14: w6NJf4Fzdi944ZokNu0DSNMv6Ug0yd6CG5G+px9SQsM=
15: </xenc:CipherValue>
16: </xenc:CipherData>
17: </xenc:EncryptedKey>
18: </ds:KeyInfo>
19: <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
20: <xenc:CipherValue xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
21: lLYcuxcU4F/PYIsTTyw53UqwmwAiwnsXdWtyU0LlrGRL4DMsRZqf4Rg8bXaJDqqTr
22: 01ezoPidTXjUxGC32mgl78y1Ngaa6oh1sFErv2RaVvUsd5ChkjXahk1HkWqRlFv+V
23: ...
24: wVoAl3KKUovsieVGUqSAMXyNt4RINSKSwYWGmj+j8bvH8n+SRcfV95JgaSjKf6Grm
25: rhppR9+949BpolebkQr9g3/PNwZg6LC826ZbHYLg0bo0s0pLXA4G6vhqYtXVtnwdS
26: </xenc:CipherValue>
27: </xenc:CipherData>
28: </xenc:EncryptedData>
29: </XFDL>
```

Figura 5.3: Documento XFDL cifrado

Capítulo 6

Considerações Finais

Para garantir que um formulário eletrônico seja viável ao uso por parte de pessoas e empresas é necessário que este possua características que o torne válido juridicamente. Com isso, a segurança envolvida neste processo se tornará cada vez mais importante além de ser um fator de muito peso no momento de escolha de qual ferramenta utilizar para processos onde se aplicam o uso de formulários.

A proposta de formularios no formato XFDL, vem a ser um importante passo rumo a popularização dos formulários eletrônicos. A garantia de integridade, autenticididade e confidencialidade são características de fundamental importância neste trabalho. Além disso tais caracteristicas serão altamente relevantes durante a adoção de um sistema de formulários seguros.

Este trabalho apresenta uma solução para tais necessidades, onde as caracteristicas citadas podem ser encontradas. Além disso, a aplicação desenvolvida pode ser facilmente utilizada tanto para se obter confidencialidade, quanto para integridade e autenticidade. A experiência de implementação permitiu verificar que a aplicação de segurança a um fomulário eletrônico é algo plenamente viavel e de baixo custo. No caso apresentado não houve nenhum custo com licensas de software, tudo o que foi utilizado é gratuito, o único custo de implantação seria da compra dos equipamentos.

A única característica necessária a um documento eletrônico que a aplicação não fornece é a tempestividade, ou seja, o envio do formulário e uma autoridade

de datação. Tal funcionalidade teve sua implementação impossibilitada por não haver tal entidade que permitisse livre acesso aos protocolos de envio e recebimento de requisições.

6.1 Trabalhos futuros

Esta seção apresenta sugestões de funcionalidades a serem implementadas a aplicação desenvolvida neste trabalho.

- Prover Tempestividade - para garantir que o documento existiu em um determinado momento, deve-se implementar a comunicacao da aplicação com uma PDDE. Como a assinatura do formulário é no formato PKCS#7 sugerimos que seja adicionada uma extensão a assinatura contendo o recibo devolvido pela PDDE.

- Gerador de formulários - permitir que o usuário crie novos formulários xfdl, a partir de um documento em branco e/ou modelos pré-definidos.

- Utilização de Smart Cards e Token - para prover uma maior segurança em relação ao acesso da chave privada durante a realização da assinatura.

Referências

[APA] APACHE. **Apache XML Security**. Disponível em <http://xml.apache.org/security/Java/>. Acesso em 21 de Julho de 2004.

[BLA 99] BLAIR, B. T.; BOYER, J. Xfdl: Creating electronic commerce transaction records using xml. In: THE EIGHTH INTERNATIONAL WORLD WIDE WEB CONFERENCE, 1999. [s.n.], 1999.

[BOU] BOUNCYCASTLE. **Bouncy Castle Crypto APIs**. Disponível em <http://www.bouncycastle.org/>. Acesso em 15 de Julho de 2004.

[BOY] BOYER, J. Signed xml: Experiences from the creation of xfdl. Canada.

[COS 03] COSTA, V. **Análise Da Confiança Do Sistema de Protocolação Digital de Documentos Eletrônicos**. Florianópolis: Universidade Federal de Santa Catarina, Outubro, 2003. Dissertação de Mestrado.

[DOM] DOM4J. **DOM4J - The flexible XML framework for Java**. Disponível em <http://www.dom4j.org//>. Acesso em 03 de Setembro de 2004.

[dSD 04] DA SILVA DIAS, J. **Confiânça no Documento Eletrônico.** Florianópolis: Universidade Federal de Santa Catarina, 2004. Tese de Doutorado.

[DUB 03a] DUBINKO, M. **XForms Essentials**. 1. ed. O'Reilly, 2003.

[DUB 03b] DUBINKO, M.; KLOTZ, L. L.; MERRICK, R. **XForms 1.0**. Disponível em <http://www.w3.org/TR/xforms/>. Acesso em 02 de Junho de 2004.

[FER 01] FERNANDES, A. Risking trust in public key infrastructure: Old techniques of managing risk applied to new technology. In: ELSEVIER SCIENCE, 2001. [s.n.], 2001.

[IGN 02] IGNACZAK, L. **Um novo modelo de Infra-estrutura de Chaves Públicas para uso no Brasil utilizando aplicativos com o código fonte aberto**. Florianópolis: Universidade Federal de Santa Catarina, Maio, 2002. Dissertação de Mestrado.

[JB 98] JOHN BOYER, TIM BRAY, M. G. **Extensible Forms Description Language (XFDL) 4.0**. Disponível em <http://www.w3.org/TR/1998/NOTE-XFDL-19980902>. Acesso em 20 de Maio de 2004.

[NOT 02] NOTOYA, A. E. **IARSDE- Infra-Estrutura de Armazenamento e Recuperação Segura de Documentos Eletrônicos.** Florianópolis: Universidade Federal de Santa Catarina, 2002. Dissertação de Mestrado.

[O'R 03] O'REILLY. **XForms Essentials**. Disponível em <http://www.webreference.com/programming/xforms/>. Acesso em 20 de Junho de 2004.

[REA 02] REAGLE, J. **XML Encryption Requirements**. DisponDisponível em <http://www.w3.org/TR/xml-encryption-req>. Acesso em 12 de Junho de 2004.

[STA 99] STALLINGS, W. **Cryptography and Network Security - Principles and Practice**. 2. ed. Prentice-Hall, 1999.

[TRI 98] TRINTA, F. A. M.; DE MACêDO, R. C. Um estudo sobre criptografia e assinatura digital. Recife, Setembro, 1998.

[W3C] W3C. **HyperText Markup Language Home Page**. Disponível em <http://www.w3.org/MarkUp/>. Acesso em 12 de agosto de 2004.

Apêndice A

Casos de Uso

Use case: Abrir documento

1. O usuário seleciona a opção Abrir no menu

2. O programa apresenta uma janela de seleção de arquivos para que o usuário escolha o arquivo desejado.

3. Ao selecionar o arquivo desejado e abri-lo, o programa lê seu conteúdo e o exibe na tela.

3.1 Caso o documento possua assinaturas, estas devem ser verificadas durante o processo de abertura, e caso algum problema com alguma das assinaturas ocorra, deve-se notificar o usuário.

Use case: Salvar documento

Pré condições: Use case: Abrir Documento

1. O usuário seleciona a opção Salvar no menu

2. O programa verifica o estado atual do formulário e salva seu conteúdo em disco.

2.1 Caso o documento possua assinaturas, a aplicação deve verificar se o conteúdo do documento foi alterado durante sua exibição, e em caso positivo, deve-se remover as assinaturas anteriores a modificação pois estas estarão invalidadas pela modificação.

Use case: Cifrar dados

Pré condições: Use case: Abrir Documento

1. O usuário seleciona no menu a opção cifrar.

2. O programa gera uma chave de forma aleatória para cifrar os dados.

3. A chave é armazenada em disco, com o nome do documento xfdl adicionado da extensão ".key"

4. O conteúdo do documento é cifrado de acordo com as normas definias pela W3C

5. O formulário cifrado é armazenado em disco, com o nome do documento xfdl adicionado da extensão ".enc"

Use case: Decifrar dados

1. O usuário seleciona no menu a opção decifrar.

2. O programa apresenta uma janela de seleção de arquivos para que o usuário escolha o arquivo desejado.

3. Ao selecionar o arquivo desejado e abri-lo, o programa procura no mesmo diretorio do arquivo escolhido por sua chave.

4. O programa decifra seu conteúdo e o exibe na tela.

3.1 Caso a chave não seja encontrada, o usuário deve ser notificado e a operação cancelada. 4.1 Caso o arquivo após ser decifrado esteja assinado, as assinaturas devem ser verificadas durante o processo de abertura, e caso algum problema com alguma das assinaturas ocorra, deve-se notificar o usuário.

Use case: Assinar documento *Pré condições*: Use case: Abrir Documento

1. O clica no botão assinar.

2. O programa apresenta uma janela de seleção de arquivos para que o usuário escolha o arquivo que contem a chave privada.

3. Gera-se a assinatura do documento

4. As assinaturas antigas são são verificadas para ver se houve algum comprometimento durante o processo, ou o documento foi alterado.

4.1 Caso alguma assinatura esteja comprometida, ela é descartada e é o usuário é notificado.

Anexo A

Artigo

Formulários Eletrônicos Seguros

Marcelo Carlomagno Carlos, Rodrigo Muller Pons

INE - Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC), Brasil
mcc@inf.ufsc.br , rmpons@inf.ufsc.br

Resumo

Este artigo apresenta uma solução para que os formulários digitais tenham as mesmas características legais encontradas nos formulários em papel, além de um característica adicional, a cifragem dos dados. Escolheu-se a linguagem XFDL para a definição dos formulários e a partir de aplicação de conceitos de segurança, como criptografia e assinatura digital pode-se criar uma aplicação que apresente tais características a um formulário XFDL.

Palavras chave: Formulários Eletrônicos, XFDL, Segurança.

Abstract

This paper presents a way to provide features found in paper-based forms to electronic forms. Moreover, cryptography algorithms are applied in electronic forms in order to make its contents unreadable to non-authorized people. XFDL language has been chosen to specify such forms. Security concepts have also been applied to engage those security requirements to electronic forms.

Keywords: Eletronic forms, XFDL, Security.

1 Introdução

A popularização da informática e o crescente uso da internet fazem com que o uso dos formulários eletrônicos se torne uma tarefa comum a todos.

Os formulários eletrônicos vêm sendo utilizados tanto para processos organizacionais, quanto para executivos, onde funcionários de empresas diferentes que possuem seus próprios sistemas informatizados possam trocar informações. Entretanto, estas informações sigilosas não devem trafegar pela rede de modo que possam ser facilmente obtidas por pessoas não autorizadas. Portanto, não há como utilizar formulários eletrônicos sem questionar dois assuntos de fundamental importância, a segurança de informação e a validade jurídica.

2 Documentos e Formulários Eletrônicos

Segundo Costa (2003); "o conceito de documento sempre esteve relacionado com a idéia de um escrito oficial, de uma informação fixada sobre um meio material. Isto porque por muito tempo o papel foi utilizado como o principal meio onde o conhecimento era registrado".

A forma de apresentação de um documento eletrônico é uma das maiores barreiras para que o documento eletrônico atenda tais características, pois este é formando por uma sequência de bits que necessita de auxílio de uma plataforma computacional para que seja exibido, desta forma, ocorre a separação entre o meio físico e o conteúdo do documento. [dSD 03]

Outra dificuldade está na verificação da integridade do documento, ou seja, verificar se ele foi modificado ou não. Tal verificação

pode ser realizada através da comparação dos resumos criptográficos do documento original e o documento que se deseja verificar a integridade.

A partir daí surge o questionamento sobre quais características um documento eletrônico deve possuir para ter validade jurídica. Estas características são as mesmas necessárias para garantir a validade jurídica dos documentos em papel, ou seja, a integridade, a autenticidade, tempestividade e não repúdio. [dSD 03]

Para um formulário eletrônico possuir a validade, ele deve estar devidamente preenchido e possuir as mesmas características que um documento eletrônico.

3 Segurança

Para que seja possível a adição de integridade, autenticidade, tempestividade e não repúdio a um formulário eletrônico, serão apresentados inicialmente alguns conceitos básicos de segurança.

3.1 Criptografia

Criptografia é a arte de escrever em cifras ou em códigos, a fim de transformar uma informação inteligível numa forma aparentemente ilegível, através de um processo chamado cifragem, fazendo com que apenas o destinatário desejado consiga decodificar e ler a mensagem com clareza, no processo inverso chamado decifragem. [TRI 98]

A criptografia permite a implementação de diversos serviços, como a autenticação, o não-repúdio, a integridade e a confidencialidade e pode ser classificada em duas categorias, de acordo com o tipo de chave utilizada: criptografia simétrica e a criptografia assimétrica.

A criptografia simétrica foi a primeira forma conhecida para ocultação dos dados e tem como principal característica a utilização de apenas uma chave para cifrar e decifrar os dados. A grande vantagem desta técnica é sua velocidade, que é muito maior do que a criptografia de chave assimétrica.

A criptografia assimétrica ou de chave pública, caracteriza-se por utilizar duas chaves diferentes, sendo uma privada e outra pública. Um texto cifrado com uma das chaves só pode ser decifrado a partir de sua chave correspondente. A grande vantagem deste sistema é permitir que qualquer um possa enviar uma mensagem secreta a uma pessoa apenas utilizando a chave pública, correspondente a chave privada de quem irá recebê-la. Como a chave pública está amplamente disponível, não há necessidade do envio de chaves em meios seguros como é feito no modelo simétrico. A confidencialidade da mensagem é garantida, enquanto a chave privada estiver segura.

3.2 Assinatura Digital

A assinatura digital é utilizada para verificar a autenticidade e a integridade de dados além de garantir o não repúdio por parte do assinante.

Para gerar uma assinatura digital inicialmente deve-se executar o processo para criação do resumo da mensagem (*hash*). Após isso, o resumo é cifrado utilizando a chave privada do assinante. O resultado é a assinatura digital.

O processo para verificação da assinatura digital é realizado pelo usuário que recebeu a mensagem criando o resumo da mensagem. Depois de criado o resumo da mensagem, ele decifra o resumo que foi enviado juntamente com a mensagem utilizando a chave pública do emissor. Se o resultado da nova função resumo for idêntico ao resumo decifrado, é possível garantir a integridade da mensagem e assegurar que os dados foram assinados pela chave privada correspondente aquela chave pública. [IGN 02]

3.3 Certificados Digitais

Um certificado digital pode ser definido como um documento eletrônico assinado digitalmente por uma terceira parte confiável, que associa o nome (e atributos) de uma pessoa ou instituição a uma chave criptográfica pública. [FER 01]

Um certificado digital é formado, basicamente, pelos seguintes campos: versão do certificado, número serial, algoritmo de assinatura, dados do emissor, validade, dados do sujeito, informações da chave pública, identificador do emissor, identificador do sujeito.

4 Linguagens para definição de formulários

Escolheu-se como objeto de estudo algumas linguagens baseadas em critérios como: sua utilização, seu suporte a cifragem

de dados e assinatura digital. Para cada linguagem apresentada será apresentada uma visão geral de sua estrutura e logo após serão apresentadas formas de aplicação dos conceitos de segurança sobre elas.

4.1 HTML

O HTML é a linguagem utilizada para publicar hipertexto na Internet. Trata-se de um formato não proprietário e pode ser criado e processado por uma grande variedade de ferramentas, desde simples editores de texto, até as mais avançadas ferramentas de desenvolvimento. A linguagem baseia-se no conceito de *tags*, que organizadas de forma estruturada, definem o conteúdo do documento. [W3C]

Os formulários HTML são uma das primeiras representações de formulários eletrônicos, e ainda hoje é o formato mais utilizado.

Um dos problemas considerados mais críticos do uso do HTML hoje em dia é sua segurança. Atualmente a cifragem de dados de documentos HTML está restrita ao canal de comunicação (SSL). Um formulário armazenado em disco, fora deste meio de comunicação, ou até mesmo os dados interpretados pelo servidor web, não apresentam nenhuma proteção em relação à segurança das informações contidas nele. A assinatura digital também é um fator crítico, já que o HTML não foi originalmente projetado para prover tal característica.

4.2 XForms

O XForms é uma linguagem integrada ao XML originalmente criada para projetar formulários para web. Como os XForms coletam os dados de um arquivo XML, estes formulários apresentam uma forma estruturada de troca de dados sendo possível verificar quais campos do formulário foram preenchidos, verificar se uma data é anterior a uma outra, etc.

Os XForms foram desenvolvidos como uma alternativa ao HTML, e possivelmente até como um sucessor deste, já que também é voltado para web.

Quanto a segurança dos dados, por se tratar de um arquivo XML, a cifragem pode ser efetuada com base nos padrões definidos pela W3C para cifragem de documentos XML. [REA 02] Em relação a assinatura digital os XForms não foram originalmente concebidos para prover esta característica, embora possam ser extendidos e desta forma conter assinaturas. Mesmo com esta possibilidade, algumas desvantagens de seu uso são claramente verificadas:

- A facilidade de criar extensões muitas vezes pode ser considerada uma vantagem, mas também possui um lado negativo. Tal facilidade permite que existam duas diferentes implementações para a mesma funcionalidade, fazendo com que não se possam especificar padrões para exibição e verificação dessas extensões.[DUB 03a]
- Aplicações XML normalmente ignoram elementos e atributos desconhecidos, o que em alguns casos podem causar grandes problemas.[DUB 03a]
- A separação dos dados e da apresentação nos XForms é outro problema a ser tratado, já que uma característica das assinaturas digitais é de se "assinar o que se vê", e tal separação permite diferentes apresentações dos dados. Assinar a apresentação e os dados é possível, mas exige uma série de adaptações ao modelo original.
- O formato dos formulários exibido na tela e o conteúdo enviado ao servidor após a submissão do formulário é diferente. Apenas os dados são enviados.

4.3 XFDL

O XFDL é uma linguagem que descreve uma sintaxe de XML adequada para desenvolver formulários eletrônicos. A finalidade do XFDL é permitir a expressão de formulários poderosos e complexos em uma sintaxe que promova a interoperabilidade da aplicação e a aderência aos padrões da Internet.

Com o objetivo de proporcionar validade jurídica aos formulários eletrônicos, o *World Wide Web Consortium* (W3C) publicou em 1998 o *draft* da especificação *Extended Forms Definition Language* (XFDL). Segundo este *draft*, o XFDL é uma linguagem projetada para criar formulários com os seguintes objetivos: [JB 98]

- Um formulário deve ser um objeto único, sem depender de nenhum recurso externo.

- A linguagem XFDL deve ser legível aos olhos humanos.
- O XFDL deve ser um padrão aberto e disponível publicamente.
- Deve prover uma sintaxe capaz de suportar o cálculo de expressões matemáticas e condicionais.
- Dados binários poderão ser encapsulados no formulário desde que sejam codificados em base-64.
- A disposição dos componentes que compõem o formulário (texto, caixas de texto, etc.) deve ser precisamente definida de forma que a visualização e impressão do documento seja fiel.
- A validação e formatação dos dados na máquina cliente deve reduzir o processamento e a carga de trabalho do servidor.
- Elementos definidos pelo usuário poderão ser adicionados a linguagem, como o código de funções externas, itens e opções personalizadas.
- Oferecer suporte à assinatura digital.

Formulários criados com a linguagem XFDL são documentos com validade jurídica, pois possuem as seguintes características: segurança, não-repúdio e auditabilidade. [BLA 99].

Outra característica interessante do XFDL é que ele não é voltado apenas para a web como o HTML e o XForms. Um formulário XFDL pode ser utilizado, preenchido e visualizado localmente, sem a necessidade de um servidor web.

Entre as linguagens apresentadas, a linguagem XFDL é a única que possui nativamente suporte à assinatura digital. Através do elemento signature, um elemento contido no próprio documento pode-se adicionar assinatura do documento e também as informações do assinante. Com isso, um formulário XFDL já possui uma maneira de prover não-repúdio, autenticidade e integridade da informação apresentada por ele.

Uma assinatura contém além da própria assinatura, todos os dados necessários para a verificação da autenticidade do formulário assinado. A assinatura deverá ser criada pelo programa visualizador do formulário. O usuário assinará o formulário utilizando um botão para este propósito.

Apesar de ser o único apresentado a prover assinatura em sua especificação, o XFDL não possui nenhum mecanismo para prover a confidencialidade da informação. Todos os dados contidos em um formulário desenvolvido em XFDL não são cifrados e estão disponíveis para qualquer pessoa. Entretanto, por se tratar de um documento XML pode-se utilizar os padrões definidos pela W3C para cifragem de documentos XML.

4.4 Comparação

Após a apresentação das linguagens estudadas, faz-se necessária a comparação entre elas para que seja possível escolher a que melhor se adapta as necessidades de um ambiente de formulários eletrônicos seguros.

A tabela abaixo apresenta um quadro comparativo entre as linguagens, destacando os aspectos referentes a segurança de seus dados.

	Assinatura Digital	**Cifragem dos Dados**
HTML	Não possui suporte em sua especificação	Apenas no canal de comunicação
XForms	Não possui suporte em sua especificação	No canal de comunicação e pode ser criada uma extensão para esta finalidade.
XFDL	Possui suporte	No canal de comunicação e pode ser criada uma extensão para esta finalidade.

Ao analisar todas as características das linguagem apresentadas, escolheu-se o XFDL para a definição dos formulários neste trabalho pelo fato de suportar assinaturas em sua especificação, sem necessidade de criar extensões e modificações em sua estrutura, evitando os problemas descritos durante a apresentação dos XForms. Além disso, foi escolhido por se apresentar no formato XML, o que facilita sua manipulação, exibição e validação dos dados.

5. Implementação a partir do XFDL

Após a apresentação de todos os requisitos necessários para que obtenha

validade jurídica e confidencialidade de um formulário eletrônico, e a definição da linguagem para implementação dos formulários, serão abordados nesta seção detalhes referentes a implementação do formulário eletrônico seguro.

A implementação basicamente divide-se em três partes: o visualizador, o assinador, e o cifrador de formulários.

5.1 Visualizador

Deve permitir a exibição do formulário de forma clara e fiel a especificação auto-contida no documento XFDL, além de permitir o preenchimento dos dados.

No visualizador também deve-se encontrar o botão de assinatura (caso o formulário defina um botão para este propósito), o qual é responsável por especificar detalhes referentes a assinatura do documento.

Outras opções do visualizador são a de cifrar, decifrar e exibir os dados das assinaturas do formulário.

5.2 Assinador

Deve permitir a assinatura do conteúdo do formulário, de forma fácil e eficiente. O assinador deve ser capaz de realizar múltiplas assinaturas e também verificar a validade das mesmas.

A assinatura do formulário deve ser no formato PKCS#7 para permitir que a partir do conteúdo encontrado no campo signature seja possível a verificação dos dados e conhecer informações do assinante através de seu certificado digital.

Para que seja possível prover a tempestividade do documento, no processo de assinatura deve-se também enviar o formulário, ou seu resumo, a uma entidade de datação, e o retorno deste processo, adicionado a assinatura.

5.3 Cifrador

A partir do padrão estabelecido pelo W3C para cifrar documentos XML, deve-se cifrar todo o arquivo XFDL. Este padrão especifica que pode-se escolher tanto o elemento XML como também apenas seus dados.

Para realizar a cifragem do documento XFDL, propõe-se que seja cifrado o elemento raiz do documento. Desta forma, serão cifrados também todo seu subconjunto de elementos e seus dados. Como resultado, haverá dois principais elementos no documento, um com os dados da chave utilizada para cifrar o XFDL e outro com o conteúdo cifrado.

Recomenda-se a cifragem do elemento raiz para que não fique visível nem a estrutura do formulário cifrado, maximizando a privacidade das informações.

6 Conclusão

Para garantir que um formulário eletrônico seja viável ao uso por parte de pessoas e empresas é necessário que este possua características que o torne válido juridicamente. Com isso, a segurança envolvida neste processo se tornará cada vez mais importante, além de ser um fator de muito peso no momento de escolha de qual ferramenta utilizar para processos onde se aplicam o uso de formulários.

A proposta de uso de formulários no formato XFDL, vem a ser um importante passo rumo à popularização dos formulários eletrônicos. A garantia de integridade, autenticidade e confidencialidade são características de fundamental importância neste trabalho. Além disso, tais características serão altamente relevantes durante a adoção de um sistema de formulários seguros.

Referências

[BLA 99] BLAIR, B. T.; BOYER, J. **XFDL: Creating electronic commerce transaction records using xml**. In: THE EIGHTH INTERNATIONALWORLD WIDE WEB CONFERENCE, 1999. [s.n.], 1999.

[IGN 02] IGNACZAK, L. **Um novo modelo de Infra-estrutura de Chaves Públicas para uso no Brasil utilizando aplicativos com o código fonte aberto.** Florianópolis: Universidade Federal de Santa Catarina, Maio, 2002. Dissertação de Mestrado.

[TRI 98] TRINTA, F. A. M.; DE MACÊDO, R. C. **Um estudo sobre criptografia e assinatura digital.** Recife, Setembro, 1998.

[COS 03] COSTA, V. **Análise Da Confiança Do Sistema de Protocolação**

Digital de Documentos Eletrônicos. Florianópolis: Universidade Federal de Santa Catarina, Outubro, 2003. Dissertação de Mestrado.

[FER 01] FERNANDES, A. **Risking trust in public key infrastructure: Old techniques of managing risk applied to new technology**. In: ELSEVIER SCIENCE, 2001. [s.n.], 2001.

[DUB 03] DUBINKO, M.; KLOTZ, L. L.; MERRICK, R. **XForms 1.0**. Disponível em <http://www.w3.org/TR/xforms/>. Acesso em 02 de Junho de 2004.

[DUB 03a] DUBINKO, M. **XForms Essentials**. 1. ed. O'Reilly, 2003.

[W3C] W3C. **HyperText Markup Language Home Page**. Disponível em

[dSD 03] DA SILVA DIAS, J. **Confiança no Documento Eletrônico.** Florianópolis: Universidade Federal de Santa Catarina, 2003. Tese de Doutorado.

[JB 98] JOHN BOYER, TIM BRAY, M. G. **Extensible Forms Description Language (XFDL) 4.0**. Disponível em <http://www.w3.org/TR/1998/NOTE-XFDL-19980902>. Acesso em 20 de Maio de 2004.

[REA 02] REAGLE, J. **XML Encryption Requirements**. Disponível em <http://www.w3.org/TR/xml-encryption-req>. Acesso em 12 de Junho de 2004.

Anexo B

Código Fonte

Arquivo Visualizador.java

```
/*
 * Visualizador.java
 *
 * Created on 5 de Novembro de 2004, 00:24
 */

package br.ufsc.ine.gui;

import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.JTextPane;
import javax.swing.text.StyledDocument;

import org.dom4j.Document;
import org.dom4j.DocumentException;

import br.ufsc.ine.gui.xfdl.XFDL;
import br.ufsc.ine.xfdl.crypto.XFDLCifrador;
import br.ufsc.ine.xfdl.crypto.XFDLCifradorException;
import br.ufsc.ine.xfdl.crypto.XFDLDecifrador;
import br.ufsc.ine.xfdl.signer.XFDLAssinador;
import br.ufsc.ine.xfdl.signer.XFDLVerificador;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JButton;

import java.awt.BorderLayout;
import javax.swing.JDialog;

/**
 *
 * @author mcc
 */
public class Visualizador extends javax.swing.JFrame {

        private XFDL xfdl;

        private String nomeArquivo = "";

        private String nomeArquivoPFX = "";

        private String senhaPFX = "";

        private ArrayList dadosAssinatura;

        private int opPfx = 1; // 0 - decifra, 1 - assina

        /** Creates new form Visualizador */
        public Visualizador() {
                dadosAssinatura = new ArrayList();
```

```
        initComponents();
        this.setContentPane(jScrollPane1);
    }

    private void initComponents() {//GEN-BEGIN:initComponents
        jDialog1 = new javax.swing.JDialog();
        jFileChooser1 = new javax.swing.JFileChooser();
        jScrollPane1 = new javax.swing.JScrollPane();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();
        jMenuItem2 = new javax.swing.JMenuItem();
        jSeparator1 = new javax.swing.JSeparator();
        jMenuItem3 = new javax.swing.JMenuItem();
        jMenu2 = new javax.swing.JMenu();
        jMenuItem4 = new javax.swing.JMenuItem();
        jMenuItem5 = new javax.swing.JMenuItem();

        initJTextPane();

        jDialog1.setSize(391, 175);
        jDialog1.setModal(true);
        jDialog1.setContentPane(getJPanel());
        jDialog1.setTitle("Selecione o certificado");

        getContentPane().setLayout(new java.awt.GridLayout());
        this.setSize(538, 415);

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Visualizador de Formul\u00e1rios XFDL");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        getContentPane().add(jScrollPane1);

        jMenu1.setText("Arquivo");
        jMenuItem1.setText("Abrir");
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });

        jMenu1.add(jMenuItem1);

        jMenuItem2.setText("Salvar");
        jMenu1.add(jMenuItem2);
        jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem2ActionPerformed(evt);
            }
        });

        jMenu1.add(jSeparator1);

        jMenuItem3.setText("Sair");
        jMenu1.add(jMenuItem3);
        jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);
            }
        });

        jMenuBar1.add(jMenu1);

        jMenu2.setText("Seguran\u00e7a");
        jMenuItem4.setText("Cifrar");
        jMenuItem4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
                        jMenuItem4ActionPerformed(evt);
                    }
                });
                jMenuItem4.setEnabled(false);

                jMenu2.add(jMenuItem4);

                jMenuItem5.setText("Verificar Assinatura");
                jMenu2.add(jMenuItem5);
                jMenuItem5.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        jMenuItem5ActionPerformed(evt);
                    }
                });
                jMenuItem5.setEnabled(false);

                jMenuBar1.add(jMenu2);

                setJMenuBar(jMenuBar1);


                pack();

                this.setSize(600, 500);
        }//GEN-END:initComponents

        public void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
                System.exit(0);
        }

        private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jMenuItem1ActionPerformed
                // TODO add your handling code here:
                int returnVal = jFileChooser1.showOpenDialog(this);
                if (returnVal == jFileChooser1.APPROVE_OPTION) {
                        initJTextPane();
                        StyledDocument doc = (StyledDocument) jTextPane1.getDocument();
                        try {
                                File f = new File(jFileChooser1.getSelectedFile()
                                                .getAbsolutePath());
                                if (f.getName().substring(f.getName().length() - 4)
                                                .equalsIgnoreCase("xfdl")) {
                                        nomeArquivo = f.getAbsolutePath();
                                        xfdl = new XFDL(f);
                                        xfdl.parse(doc);
                                        setButtonAction();
                                        XFDLVerificador xv = new XFDLVerificador(xfdl.getDocument());
                                        //System.out.println(xv.verifica());
                                        if (!xv.verifica())
                                                JOptionPane
                                                                .showMessageDialog(
                                                                                this,
                                                                                "Este documento possui assinatura
(s) inválidas",
                                                                                "Aviso",
JOptionPane.WARNING_MESSAGE);
                                        dadosAssinatura = (xv.getListDadosAssinaturas());
                                        jMenuItem4.setEnabled(true);
                                        jMenuItem5.setEnabled(true);
                                } else if (f.getName().substring(f.getName().length() - 4)
                                                .equalsIgnoreCase(".enc")) {
                                        opPfx = 0;
                                        nomeArquivo = f.getAbsolutePath();
                                        jDialog = getJDialog();
                                        jDialog.show();
                                } else {
                                        jTextPane1.setText("Arquivo inválido ...");
                                }
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
```

```
	}//GEN-LAST:event_jMenuItem1ActionPerformed

	private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jMenuItem1ActionPerformed
		// TODO add your handling code here:
		xfdl.salva();
	}

	private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jMenuItem1ActionPerformed
		// TODO add your handling code here:
		jDialog1.show();
	}


	private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jMenuItem1ActionPerformed
		// TODO add your handling code here:
		XFDLVerificador xv = new XFDLVerificador(xfdl.getDocument());
		//System.out.println(xv.verifica());
		xv.verifica();
		dadosAssinatura = (xv.getListDadosAssinaturas());

		DadosAssinatura da = new DadosAssinatura(dadosAssinatura);
		da.show();
	}

	/** Exit the Application */
	private void exitForm(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_exitForm
		System.exit(0);
	}//GEN-LAST:event_exitForm

	/**
	 * This method initializes jPanel
	 *
	 * @return javax.swing.JPanel
	 */
	private JPanel getJPanel() {
		if (jPanel == null) {
			jLabel1 = new JLabel();
			jLabel = new JLabel();
			jPanel = new JPanel();
			jPanel.setLayout(null);
			jLabel.setBounds(17, 16, 286, 23);
			jLabel.setText("Selecione o certificado do destinatário:");
			jLabel1.setBounds(19, 59, 62, 18);
			jLabel1.setText("Arquivo:");
			jPanel.add(jLabel, null);
			jPanel.add(jLabel1, null);
			jPanel.add(getJTextField(), null);
			jPanel.add(getJButton(), null);
			jPanel.add(getJButton1(), null);
			jPanel.add(getJButton2(), null);
		}
		return jPanel;
	}

	/**
	 * This method initializes jTextField
	 *
	 * @return javax.swing.JTextField
	 */
	private JTextField getJTextField() {
		if (jTextField == null) {
			jTextField = new JTextField();
			jTextField.setBounds(85, 59, 161, 18);
		}
		return jTextField;
	}

	/**
	 * This method initializes jButton
```

```
	 *
	 * @return javax.swing.JButton
	 */
	private JButton getJButton() {
		if (jButton == null) {
			jButton = new JButton();
			jButton.setBounds(56, 110, 106, 20);
			jButton.setText("Ok");
			jButton.addActionListener(new java.awt.event.ActionListener() {
				public void actionPerformed(java.awt.event.ActionEvent e) {
					XFDLCifrador xc = new XFDLCifrador(nomeArquivo);
					try {
						xc.cifra(nomeArquivo + ".enc", jTextField.getText());
						JOptionPane
								.showMessageDialog(jDialog1,
										"Arquivo cifrado e armazaendo em
disco com as extensao .enc");
						jDialog1.hide();
					} catch (XFDLCifradorException ex) {
						// TODO Auto-generated catch block
						ex.printStackTrace();
						JOptionPane.showMessageDialog(jDialog1,
								"Erro ao cifrar dados", "Erro",
								JOptionPane.ERROR_MESSAGE);
					}
				}
			});
		}
		return jButton;
	}

	/**
	 * This method initializes jButton1
	 *
	 * @return javax.swing.JButton
	 */
	private JButton getJButton1() {
		if (jButton1 == null) {
			jButton1 = new JButton();
			jButton1.setBounds(218, 110, 106, 20);
			jButton1.setText("Cancelar");
			jButton1.addActionListener(new java.awt.event.ActionListener() {
				public void actionPerformed(java.awt.event.ActionEvent e) {
					jDialog1.hide();
				}
			});
		}
		return jButton1;
	}

	private void setButtonAction() {
		xfdl.setButtonAct(new java.awt.event.ActionListener() {
			public void actionPerformed(java.awt.event.ActionEvent evt) {
				jDialog = getJDialog();
				jDialog.show();
				//System.out.println("aqui");
				//XFDLAssinador xa = new XFDLAssinador(xfdl.getDocument());
				//doc = xa.assina(caminhoPfx,senhaPfx);
				//salva();
			}
		});
	}

	/**
	 * This method initializes jButton2
	 *
	 * @return javax.swing.JButton
	 */
	private JButton getJButton2() {
		if (jButton2 == null) {
			jButton2 = new JButton();
			jButton2.setBounds(255, 59, 113, 18);
			jButton2.setText("Selecione ...");
```

```
                        jButton2.addActionListener(new java.awt.event.ActionListener() {
                                public void actionPerformed(java.awt.event.ActionEvent e) {
                                        int ret = jFileChooser1.showOpenDialog(jDialog1);
                                        if (ret == JFileChooser.APPROVE_OPTION) {
                                                jTextField.setText(jFileChooser1.getSelectedFile()
                                                                .getAbsolutePath());
                                        }
                                }
                        });
        }
        return jButton2;
}

/**
 * This method initializes jContentPane
 *
 * @return javax.swing.JPanel
 */
private JPanel getJContentPane() {
        if (jContentPane == null) {
                jLabel4 = new JLabel();
                jLabel3 = new JLabel();
                jLabel2 = new JLabel();
                jContentPane = new JPanel();
                jContentPane.setLayout(null);
                jLabel2.setBounds(22, 60, 53, 18);
                jLabel2.setText("Arquivo:");
                jLabel3.setBounds(22, 95, 48, 14);
                jLabel3.setText("Senha:");
                jLabel4.setBounds(22, 20, 234, 18);
                jLabel4.setText("Selecione o arquivo pfx (PKCS#12)");
                jContentPane.add(getJTextField1(), null);
                jContentPane.add(getJTextField2(), null);
                jContentPane.add(getJButton3(), null);
                jContentPane.add(getJButton4(), null);
                jContentPane.add(jLabel2, null);
                jContentPane.add(jLabel3, null);
                jContentPane.add(getJButton5(), null);
                jContentPane.add(jLabel4, null);
        }
        return jContentPane;
}

/**
 * This method initializes jDialog
 *
 * @return javax.swing.JDialog
 */
private JDialog getJDialog() {
        if (jDialog == null) {
                jDialog = new JDialog();
                jDialog.setContentPane(getJContentPane());
                jDialog.setTitle("Selecione o arquivo pfx");
                jDialog.setSize(397, 216);
                jDialog.setModal(true);
        }
        return jDialog;
}

/**
 * This method initializes jTextField1
 *
 * @return javax.swing.JTextField
 */
private JTextField getJTextField1() {
        if (jTextField1 == null) {
                jTextField1 = new JTextField();
                jTextField1.setBounds(85, 60, 165, 18);
        }
        return jTextField1;
}

/**
```

```
	 * This method initializes jTextField2
	 *
	 * @return javax.swing.JTextField
	 */
	private JTextField getJTextField2() {
		if (jTextField2 == null) {
			jTextField2 = new JTextField();
			jTextField2.setBounds(85, 92, 165, 19);
		}
		return jTextField2;
	}

	/**
	 * This method initializes jButton3
	 *
	 * @return javax.swing.JButton
	 */
	private JButton getJButton3() {
		if (jButton3 == null) {
			jButton3 = new JButton();
			jButton3.setBounds(58, 144, 106, 23);
			jButton3.setText("Ok");
			jButton3.addActionListener(new java.awt.event.ActionListener() {
				public void actionPerformed(java.awt.event.ActionEvent e) {
					jDialog1.hide();
					try {
						if (opPfx == 0) {
							// OK DECIFRA
							XFDLDecifrador dec = new XFDLDecifrador(
									nomeArquivo, jTextField1.getText(),
									jTextField2.getText());
							String caminhoDecifrado = nomeArquivo.substring(0,
									nomeArquivo.length() - 9);
							File f = new File(caminhoDecifrado + ".xfdl");
							boolean gravar = true;
							if (f.exists()) {
								if (JOptionPane
										.showConfirmDialog(
getContentPane(),
"O arquivo "
	+ f.getName()
	+ " já existe, deseja sobreescrever ?") != JOptionPane.OK_OPTION) {
									gravar = false;
								}
							}
							if (gravar) {
								dec.decifra(caminhoDecifrado + ".xfdl");
								xfdl = new XFDL(new File(caminhoDecifrado
										+ ".xfdl"));
								nomeArquivo = caminhoDecifrado + ".xfdl";
								JOptionPane.showMessageDialog(getContentPane(),
										"Arquivo decifrado com sucesso");
								jDialog.hide();
								StyledDocument doc = (StyledDocument) jTextPane1
										.getDocument();
								xfdl.parse(doc);
								xfdl.setArquivoPFX(nomeArquivoPFX, senhaPFX);
								XFDLVerificador xv = new XFDLVerificador(xfdl
										.getDocument());
								//System.out.println(xv.verifica());
								if (!xv.verifica())
									JOptionPane
											.showMessageDialog(
getContentPane(),
"Este documento possui assinatura(s) inválidas",
"Aviso",
```

```
JOptionPane.WARNING_MESSAGE);
                                                                dadosAssinatura = (xv.getListDadosAssinaturas());
                                                                jMenuItem4.setEnabled(true);
                                                                jMenuItem5.setEnabled(true);
                                                        }
                                                } else {
                                                        // OK Assina

                                                        Document d = xfdl.getDocument();
                                                        XFDLAssinador xa = new XFDLAssinador(d);
                                                        d = xa.assina(jTextField1.getText(),
                                                                        jTextField2.getText());
                                                        xfdl.salva();
                                                }
                                        } catch (Exception ex) {
                                                ex.printStackTrace();
                                                JOptionPane.showMessageDialog(getContentPane(),
                                                                "Não foi possível decifrar o arquivo", "Erro",
                                                                JOptionPane.ERROR_MESSAGE);
                                        }

                                }
                        });
                }
                return jButton3;
        }

        /**
         * This method initializes jButton4
         *
         * @return javax.swing.JButton
         */
        private JButton getJButton4() {
                if (jButton4 == null) {
                        jButton4 = new JButton();
                        jButton4.setBounds(222, 144, 106, 23);
                        jButton4.setText("Cancelar");
                        jButton4.addActionListener(new java.awt.event.ActionListener() {
                                public void actionPerformed(java.awt.event.ActionEvent e) {
                                        jDialog.hide();
                                }
                        });
                }
                return jButton4;
        }

        /**
         * This method initializes jButton5
         *
         * @return javax.swing.JButton
         */
        private JButton getJButton5() {
                if (jButton5 == null) {
                        jButton5 = new JButton();
                        jButton5.setBounds(261, 60, 113, 18);
                        jButton5.setText("Selecione ...");
                        jButton5.addActionListener(new java.awt.event.ActionListener() {
                                public void actionPerformed(java.awt.event.ActionEvent e) {
                                        int ret = jFileChooser1.showOpenDialog(jDialog);
                                        if (ret == JFileChooser.APPROVE_OPTION) {
                                                jTextField1.setText(jFileChooser1.getSelectedFile()
                                                                .getAbsolutePath());
                                        }
                                }
                        });
                }
                return jButton5;
        }

        /**
         * @param args
         *            the command line arguments
```

```
	 */
	public static void main(String args[]) {
		new Visualizador().show();
	}

	private void initJTextPane() {
		jTextPane1 = new javax.swing.JTextPane();
		jTextPane1.setEditable(false);
		jScrollPane1.setViewportView(jTextPane1);
	}

	// Variables declaration - do not modify//GEN-BEGIN:variables
	private javax.swing.JDialog jDialog1; //  @jve:decl-index=0:visual-constraint="10,822"

	private javax.swing.JFileChooser jFileChooser1;

	private javax.swing.JMenu jMenu1;

	private javax.swing.JMenu jMenu2;

	private javax.swing.JMenuBar jMenuBar1;

	private javax.swing.JMenuItem jMenuItem1;

	private javax.swing.JMenuItem jMenuItem2;

	private javax.swing.JMenuItem jMenuItem3;

	private javax.swing.JMenuItem jMenuItem4;

	private javax.swing.JMenuItem jMenuItem5;

	private javax.swing.JScrollPane jScrollPane1;

	private javax.swing.JSeparator jSeparator1;

	private javax.swing.JTextPane jTextPane1;

	private JPanel jPanel = null;

	private JLabel jLabel = null;

	private JLabel jLabel1 = null;

	private JTextField jTextField = null;

	private JButton jButton = null;

	private JButton jButton1 = null;

	private JButton jButton2 = null;

	private JPanel jContentPane = null;

	private JDialog jDialog = null; //  @jve:decl-index=0:visual-constraint="27,274"

	private JTextField jTextField1 = null;

	private JTextField jTextField2 = null;

	private JButton jButton3 = null;

	private JButton jButton4 = null;

	private JLabel jLabel2 = null;

	private JLabel jLabel3 = null;

	private JButton jButton5 = null;

	private JLabel jLabel4 = null;
}
```

Arquivo DadosAssinatura.java

```
/*
 * DadosAssinatura.java
 *
 * Created on 5 de Novembro de 2004, 09:58
 */

package br.ufsc.ine.gui;

import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.text.Style;
import javax.swing.text.StyleConstants;
import javax.swing.text.StyledDocument;

import br.ufsc.ine.seguranca.Certificado;

/**
 *
 * @author  mcc
 */
public class DadosAssinatura extends javax.swing.JFrame {

        static private final String newline = "\n";

        private StyledDocument doc;
        private ArrayList dados;

    /** Creates new form DadosAssinatura */
    public DadosAssinatura(ArrayList da) {
        dados = da;
       initComponents();
       doc = (StyledDocument)jTextPane1.getDocument();
       mostraDados();
    }

    public void mostraDados() {
        insereTextoBold("Resultado da verificação"+newline+newline);
        Iterator i = dados.iterator();
        int c = 1;
        while (i.hasNext()) {
            br.ufsc.ine.xfdl.signer.DadosAssinatura da = ((br.ufsc.ine.xfdl.signer.DadosAssinatura)i.next());
            Certificado cert = da.getCert();
            int status = da.getStatus();
            insereTextoBold("Assinatura "+c+": "+getMessage(status)+newline);
            insereTextoBoldIt("Assinatura relizada em: ");
            insereTexto(da.getDataAssinatura()+newline);
            insereTextoBoldIt("Assinador por: ");
            insereTexto(cert.getAssunto()+newline);
            insereTextoBoldIt("Emissor: ");
            insereTexto(cert.getEmissor()+newline+newline);
                c++;
        }
    }

    private String getMessage(int id) {
        switch (id) {
            case 1:
                return "ok";
            case -200:
                return "Erro: Assinatura inválida";
            case -201:
                return "Erro: Algoritmo desconhecido";
            case -202:
                return "Erro: Provider desconhecido";
            case -100:
```

```
                        return "Erro: Assinatura não encontrada";
                case -101:
                        return "Erro: Certificado expirado";
                case -102:
                        return "Erro: Certificado ainda não válido";
                default:
                        return "Erro desconhecido";
        }
}

public void insereTexto(String texto) {
   try {
      // Create a style object and then set the style attributes
      Style style = doc.addStyle("StyleName", null);
      // Append to document
      doc.insertString(doc.getLength(), texto, style);
   } catch (Exception e) {
      e.printStackTrace();
   }
}

public void insereTextoBold(String texto) {
   try {
      // Create a style object and then set the style attributes
      Style style = doc.addStyle("StyleName", null);
      // Italic
      //StyleConstants.setItalic(style, true);
      StyleConstants.setBold(style, true);
      // Foreground color
      //StyleConstants.setForeground(style, Color.GREEN);
      // Append to document
      doc.insertString(doc.getLength(), texto, style);
   } catch (Exception e) {
      e.printStackTrace();
   }
}

public void insereTextoIt(String texto) {
   try {
      // Create a style object and then set the style attributes
      Style style = doc.addStyle("StyleName", null);
      // Italic
      StyleConstants.setItalic(style, true);
      //StyleConstants.setBold(style, true);
      // Foreground color
      //StyleConstants.setForeground(style, Color.GREEN);
      // Append to document
      doc.insertString(doc.getLength(), texto, style);
   } catch (Exception e) {
      e.printStackTrace();
   }
}

public void insereTextoBoldIt(String texto) {
   try {
      // Create a style object and then set the style attributes
      Style style = doc.addStyle("StyleName", null);
      // Italic
      StyleConstants.setItalic(style, true);
      StyleConstants.setBold(style, true);
      // Foreground color
      //StyleConstants.setForeground(style, Color.GREEN);
      // Append to document
      doc.insertString(doc.getLength(), texto, style);
   } catch (Exception e) {
      e.printStackTrace();
   }
}


/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
```

```
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
            this.setTitle("Verificação das assinaturas");
        jTextPane1 = new javax.swing.JTextPane();
        jScrollPane1 = new javax.swing.JScrollPane();

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });
        jTextPane1.setEditable(false);
        jScrollPane1.setViewportView(jTextPane1);

        getContentPane().add(jScrollPane1);


        pack();

        this.setSize(400,300);
    }//GEN-END:initComponents

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_exitForm
        this.hide();
    }//GEN-LAST:event_exitForm

    /**
     * @param args the command line arguments
     */
 /*   public static void main(String args[]) {
        new DadosAssinatura().show();
    }*/


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JTextPane jTextPane1;
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration//GEN-END:variables

}
```

Arquivo XFDL.java

```
/*
 * Created on 04/11/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.gui.xfdl;

import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Iterator;

import javax.swing.JFrame;
import javax.swing.text.StyledDocument;

import org.dom4j.Document;
import org.dom4j.DocumentException;
```

```
import org.dom4j.Element;
import org.dom4j.Node;
import org.dom4j.io.OutputFormat;
import org.dom4j.io.SAXReader;
import org.dom4j.io.XMLWriter;

import br.ufsc.ine.gui.Visualizador;
import br.ufsc.ine.gui.xfdl.tags.Button;
import br.ufsc.ine.gui.xfdl.tags.Buttongroup;
import br.ufsc.ine.gui.xfdl.tags.Cell;
import br.ufsc.ine.gui.xfdl.tags.Check;
import br.ufsc.ine.gui.xfdl.tags.Combobox;
import br.ufsc.ine.gui.xfdl.tags.Field;
import br.ufsc.ine.gui.xfdl.tags.GenericTag;
import br.ufsc.ine.gui.xfdl.tags.Label;
import br.ufsc.ine.gui.xfdl.tags.Radio;
import br.ufsc.ine.xfdl.signer.XFDLAssinador;


/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class XFDL {

	private ArrayList lista = new ArrayList();
	private ArrayList listaBtnSig = new ArrayList();
	private Document doc;
	private Element raiz;
	private StyledDocument stDoc;
	private File arquivo;
	private Hashtable ht;
	private Hashtable htRadio;
	private Buttongroup grupo;

	private String caminhoPfx;
	private String senhaPfx;

	public Document getDocument() {
		return doc;
	}

	public XFDL(File arquivo) throws DocumentException, FileNotFoundException {
		ht = new Hashtable();
		htRadio = new Hashtable();
		SAXReader reader = new SAXReader();
		//Le o arquivo do disco
		FileInputStream fis = new FileInputStream(arquivo);
		doc = reader.read(fis);
		raiz = doc.getRootElement();
		this.arquivo = arquivo;
		grupo = new Buttongroup();
		try {
			fis.close();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

	public void parse(StyledDocument d) {
		stDoc = d;
		Iterator i = raiz.elementIterator();

		while (i.hasNext()) {
			Node nodo = (Node)i.next();
			//Checkbox
			if (nodo.getName().equalsIgnoreCase(Check.TAGNAME)) {
				Check c = new Check(nodo);
				c.parseItem(lista,stDoc);
			} else if (nodo.getName().equalsIgnoreCase(Label.TAGNAME)) {
```

```
                        // LABEL
                                Label l = new Label(nodo);
                                l.parseItem(lista,stDoc);
                        } else if (nodo.getName().equalsIgnoreCase(Field.TAGNAME)) {
                                Field f = new Field(nodo);
                                f.parseItem(lista,stDoc);
                        } else if (nodo.getName().equalsIgnoreCase(Button.TAGNAME)) {
                                Button b = new Button(nodo);
                                if (b.getTipo().equals("signature")) {
                                        listaBtnSig.add(b);
                                        /*b.addActionListener(new java.awt.event.ActionListener() {
                                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                                      //System.out.println("aqui");
                                        XFDLAssinador xa = new XFDLAssinador(doc);

                                        doc = xa.assina(caminhoPfx,senhaPfx);
                                        salva();
                                    }
                                });*/
                                }
    b.parseItem(lista,stDoc);
                        } else if (nodo.getName().equalsIgnoreCase(Combobox.TAGNAME)) {
    Combobox combo = new Combobox(nodo);
    combo.parseItem(lista,stDoc);
    ht.put(combo.getGroup(), combo);
  //Cell
  } else if (nodo.getName().equalsIgnoreCase(Cell.TAGNAME)) {
    Cell cell = new Cell(nodo);
    cell.parseItem(stDoc);
    Combobox combo = (Combobox) ht.get(cell.getGroup());
    combo.addItemJCombo(cell.getValue());
  } else if (nodo.getName().equalsIgnoreCase(Radio.TAGNAME)) {
                                Radio radio = new Radio(nodo);
                                radio.parseItem(lista, stDoc);
                                if (htRadio.size() == 0) {
                                        htRadio.put(radio.getGroup(), radio);
                                        grupo.add(radio.getComponent());
                                } else {
                                        if (htRadio.contains(radio.getGroup())) {
                                                radio = (Radio) htRadio.get(radio.getGroup());
          grupo.add(radio.getComponent());
                                        } else {
                                                htRadio.put(radio.getGroup(), radio);
                                                grupo.add(radio.getComponent());
                                        }
                                }
                                //htRadio.put(radio.getGroup(), radio);
                        }
                }
    }

    public void setButtonAct(ActionListener e) {
                Iterator i = listaBtnSig.iterator();
                while (i.hasNext()) {
                        Button b = (Button)i.next();
                        b.addActionListener(e);
                }
    }

    public void setArquivoPFX(String caminho, String senha) {
                caminhoPfx = caminho;
                senhaPfx = senha;
    }

    public void salva() {
                Iterator i = lista.iterator();
                while (i.hasNext()) {
                        GenericTag tag = (GenericTag) i.next();
                        if(tag instanceof Field) {
                                Field f = (Field)tag;
                                f.setValue(f.getComponent().getText());
                        } else if (tag instanceof Check) {
                                Check c = (Check)tag;
```

```
                                        if (c.getComponent().isSelected())
                                                c.setValue("on");
                                        else
                                                c.setValue("off");
                                } else if (tag instanceof Combobox) {
                                        Combobox c = (Combobox)tag;
                                        c.setValue((String)c.getSelected());
                                } else if (tag instanceof Radio) {
                                        Radio r = (Radio)tag;
                                        if (r.getComponent().isSelected()) {
                                                r.setValue("on");
                                        } else {
                                                r.setValue("off");
                                        }
                                }
                        }
                        try {
                                OutputFormat format = new OutputFormat("", false, "iso-8859-1");
                                FileOutputStream saida = new FileOutputStream(arquivo);
                                XMLWriter writer = null;
                                writer = new XMLWriter(saida, format);
                                writer.write(doc);
                                writer.flush();
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
}
```

Arquivo Button.java

```
/*
 * Created on 04/11/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.gui.xfdl.tags;

import java.awt.Font;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.swing.JButton;
import javax.swing.text.StyledDocument;

import org.dom4j.Element;
import org.dom4j.Node;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class Button extends GenericTag {
        public static final String TAGNAME = "button";
        private JButton jButton1;
        private Font font;
        private String fontName;
        private int fontSize;
        private int fontStyle;
        private String tipo;

        public Button (Node n) {
                nodo = n;
                jButton1 = new JButton();
        }
```

```
	public JButton getComponent() {
		return jButton1;
	}

	public void setFontName(String name) {
		fontName = name;
		font = new Font(fontName,fontStyle,fontSize);
		jButton1.setFont(font);
	}

	public void setFontSize(int size) {
		fontSize = size;
		font = new Font(fontName,fontStyle,fontSize);
		jButton1.setFont(font);
	}

	public void setFontStyle(int style) {
		fontStyle = style;
		font = new Font(fontName,fontStyle,fontSize);
		jButton1.setFont(font);
	}

	public String getTipo() {
		List l = ((Element)nodo).elements("type");
		if(l.size()>0)
			return ((Node)l.get(0)).getText();
		else
			return "";
	}

	public void parseItem(ArrayList lista, StyledDocument doc) {
		this.stDoc = doc;
		Element ele = (Element)nodo;
		Iterator iter = ele.elementIterator();
		while (iter.hasNext()) {
			Node este = (Node) iter.next();
// ve se eh o value
			if(este.getName().equalsIgnoreCase("value")) {
				jButton1.setText(este.getText());
			// tipo
			} else if(este.getName().equalsIgnoreCase("type")) {
				tipo = este.getText();
			} else if (este.getName().equalsIgnoreCase("fontinfo")) {
				// dados da fonte
				Element elFontInfo = (Element) este;
				List l = elFontInfo.elements();
				int tam = l.size();
				if(tam>0)
					// nome da fonte
					this.setFontName(((Node)l.get(0)).getText());

				if (tam>1) {
					// tamanho
					this.setFontSize(Integer.parseInt(((Node)l.get(1)).getText()));
				}
				if (tam>2) {
					// weigth
					String estilo = ((Node)l.get(2)).getText();

					if(estilo.equalsIgnoreCase("bold"))
						this.setFontStyle(Font.BOLD);
					else if(estilo.equalsIgnoreCase("italic"))
						this.setFontStyle(Font.ITALIC);
					else
						this.setFontStyle(Font.PLAIN);
				}
			}

		}
//		 coloca no StyledDoc
		setStyleComponent(jButton1);
		this.addNewLine();
```

```
                // coloca na lista
                lista.add(this);
        }

        public void addActionListener(ActionListener act) {
                jButton1.addActionListener(act);
        }
}
```

Arquivo Buttongroup.java

```
/*
 * Created on 05/11/2004
 *
 * To change the template for this generated file go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
package br.ufsc.ine.gui.xfdl.tags;

import javax.swing.ButtonGroup;
import javax.swing.JRadioButton;

/**
 * @author Administrador
 *
 * To change the template for this generated type comment go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
public class Buttongroup extends GenericTag {
        private ButtonGroup grupo;

        public Buttongroup() {
                grupo = new ButtonGroup();
        }

        public ButtonGroup getComponent() {
                return grupo;
        }

   public void add(JRadioButton radio){
      grupo.add(radio);
   }

}
```

Arquivo Cell.java

```
/*
 * Created on 05/11/2004
 *
 * To change the template for this generated file go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
package br.ufsc.ine.gui.xfdl.tags;

import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.text.StyledDocument;

import org.dom4j.Element;
import org.dom4j.Node;

/**
 * @author Administrador
 *
 * To change the template for this generated type comment go to
```

```
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
public class Cell extends GenericTag {
   public static final String TAGNAME = "cell";
   private String group;
   private String value;

   public Cell(Node n) {
      nodo = n;
      sid = ((Element) n).attributeValue("sid");
   }

   public void parseItem(StyledDocument doc) {
      this.stDoc = doc;
      Element ele = (Element) nodo;
      Iterator iter = ele.elementIterator();
      while (iter.hasNext()) {
         Node este = (Node) iter.next();

         if (este.getName().equalsIgnoreCase("group")) {
            this.setGroup(este.getStringValue());

         } else if (este.getName().equalsIgnoreCase("value")) {
            this.setValue(este.getText());
         }
      }
   }

            public String getGroup() {
                     return group;
            }

            public String getValue() {
                     return value;
            }

            public void setGroup(String string) {
                     group = string;
            }

            public void setValue(String string) {
                     value = string;
            }

}
```

Arquivo Check.java

```
/*
 * Created on 04/11/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.gui.xfdl.tags;

import java.awt.Color;
import java.awt.Font;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.swing.JCheckBox;
import javax.swing.text.StyledDocument;

import org.dom4j.Element;
import org.dom4j.Node;

import br.ufsc.ine.gui.xfdl.tags.Label;
```

```
/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class Check extends GenericTag {
	public static final String TAGNAME = "check";

	private Label label;
	private JCheckBox jCheck;

	public Check(Node n) {
		nodo = n;
		sid = ((Element)n).attributeValue("sid");
		label = new Label();
		jCheck = new JCheckBox();
		jCheck.setBackground(Color.WHITE);
	}

	public void setValue(String valor) {
		Node n = ((Element)nodo).element("value");
		n.setText(valor);
	}

	public void parseItem(ArrayList lista, StyledDocument doc) {
		this.stDoc = doc;
		Element ele = (Element)nodo;
		Iterator iter = ele.elementIterator();
		while (iter.hasNext()) {
			Node este = (Node) iter.next();
	//adiciona o conteudo de label para um texto ao lado dos checkbox
			if (este.getName().equalsIgnoreCase(Label.TAGNAME)) {
				// define texto do label
				label.setJLabelText(este.getText());
	// pega informacoes sobre a font
			} else if (este.getName().equalsIgnoreCase("labelfontinfo")) {
				Element elFontInfo = (Element) este;
				List l = elFontInfo.elements();
				int tam = l.size();
				if(tam>0)
					// nome da fonte
					label.setFontName(((Node)l.get(0)).getText());

				if (tam>1) {
					// tamanho
					label.setFontSize(Integer.parseInt(((Node)l.get(1)).getText()));
				}
				if (tam>2) {
					// weigth
					String estilo = ((Node)l.get(2)).getText();

					if(estilo.equalsIgnoreCase("bold"))
						label.setFontStyle(Font.BOLD);
					else if(estilo.equalsIgnoreCase("italic"))
						label.setFontStyle(Font.ITALIC);
					else
						label.setFontStyle(Font.PLAIN);
				}
			} else if(este.getName().equalsIgnoreCase("labelfontcolor")) {
				Element elFontInfo = (Element) este;
				List l = elFontInfo.elements();
				int tam = l.size();
				if(tam>0) {
					try {
						label.setFontColor(Color.getColor(((Node)l.get(0)).getText()));

					} catch (Exception e) { }
				}
			} else if(este.getName().equalsIgnoreCase("value")) {
				if (este.getText().equalsIgnoreCase("on"))
					jCheck.setSelected(true);
```

```
                        }
                    }
                    // coloca no StyledDoc
                    setStyleComponent(label.getComponent());
//                   coloca no StyledDoc
                    setStyleComponent(jCheck);
                    this.addNewLine();
                    // coloca na lista
                    lista.add(this);
        }

        public JCheckBox getComponent() {
                return jCheck;
        }

        public void setSelected(boolean selected) {
                jCheck.setSelected(selected);
        }

        public boolean getSelected() {
                return jCheck.isSelected();
        }

}
```

Arquivo Field.Java

```
/*
 * Created on 04/11/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.gui.xfdl.tags;

import java.awt.Color;
import java.awt.Font;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.swing.JLabel;
import javax.swing.JTextArea;
import javax.swing.border.BevelBorder;
import javax.swing.border.Border;
import javax.swing.plaf.basic.BasicBorders;
import javax.swing.text.StyledDocument;

import org.dom4j.Element;
import org.dom4j.Node;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class Field extends GenericTag {
        public static final String TAGNAME = "field";
        private JLabel jLabel;
        private JTextArea jTextArea;
        private Font font;
        private String fontName;
        private int fontSize;
        private int fontStyle;
        private Label label;

        protected static Border dropBorder = new BasicBorders.FieldBorder
```

```
(Color.BLACK,Color.WHITE,Color.BLACK,Color.WHITE);

	public Field(Node n) {
		label = new Label();
		jTextArea = new JTextArea();
		nodo = n;
		jTextArea.setBorder(dropBorder);
	}

	public void setValue(String valor) {
		Node n = ((Element)nodo).element("value");
		n.setText(valor);
	}

	public void setFontName(String name) {
		fontName = name;
		font = new Font(fontName,fontStyle,fontSize);
		jTextArea.setFont(font);
	}

	public void setFontSize(int size) {
		fontSize = size;
		font = new Font(fontName,fontStyle,fontSize);
		jTextArea.setFont(font);
	}

	public void setFontStyle(int style) {
		fontStyle = style;
		font = new Font(fontName,fontStyle,fontSize);
		jTextArea.setFont(font);
	}

	public void parseItem(ArrayList lista, StyledDocument doc) {
		this.stDoc = doc;
		Element ele = (Element)nodo;
		Iterator iter = ele.elementIterator();
		while (iter.hasNext()) {
			Node este = (Node) iter.next();
	//adiciona o conteudo de label para um texto ao lado dos checkbox
			if (este.getName().equalsIgnoreCase(Label.TAGNAME)) {
				// define texto do label
				label.setJLabelText(este.getText());
	// pega informacoes sobre a font do label
			} else if (este.getName().equalsIgnoreCase("labelfontinfo")) {
				Element elFontInfo = (Element) este;
				List l = elFontInfo.elements();
				int tam = l.size();
				if(tam>0)
					// nome da fonte
					label.setFontName(((Node)l.get(0)).getText());

				if (tam>1) {
					// tamanho
					label.setFontSize(Integer.parseInt(((Node)l.get(1)).getText()));
				}
				if (tam>2) {
					// weigth
					String estilo = ((Node)l.get(2)).getText();

					if(estilo.equalsIgnoreCase("bold"))
						label.setFontStyle(Font.BOLD);
					else if(estilo.equalsIgnoreCase("italic"))
						label.setFontStyle(Font.ITALIC);
					else
						label.setFontStyle(Font.PLAIN);
				}
			} else if(este.getName().equalsIgnoreCase("labelfontcolor")) {
				Element elFontInfo = (Element) este;
				List l = elFontInfo.elements();
				int tam = l.size();
				if(tam>0) {
					// cor da fonte
					try {
```

```
                                        label.setFontColor(Color.getColor(((Node)l.get(0)).getText()));

                                    } catch (Exception e) { }
                                }
                            // Valor inicial
                            } else if(este.getName().equalsIgnoreCase("value")) {
                                jTextArea.setText(este.getText());
                            // TAMANHO
                            } else if(este.getName().equalsIgnoreCase("size")) {
                                Element elTam = (Element) este;
                                List l = elTam.elements();
                                int tam = l.size();
                                if(tam>0) {
                                    jTextArea.setColumns(Integer.parseInt(((Node)l.get(0)).getText()));
                                }
                                if(tam>1) {
                                    jTextArea.setRows(Integer.parseInt(((Node)l.get(1)).getText()));
                                }
                            }else if (este.getName().equalsIgnoreCase("fontinfo")) {
                                // dados da fonte
                                Element elFontInfo = (Element) este;
                                List l = elFontInfo.elements();
                                int tam = l.size();
                                if(tam>0)
                                    // nome da fonte
                                    this.setFontName(((Node)l.get(0)).getText());

                                if (tam>1) {
                                    // tamanho
                                    this.setFontSize(Integer.parseInt(((Node)l.get(1)).getText()));
                                }
                                if (tam>2) {
                                    // weigth
                                    String estilo = ((Node)l.get(2)).getText();

                                    if(estilo.equalsIgnoreCase("bold"))
                                        this.setFontStyle(Font.BOLD);
                                    else if(estilo.equalsIgnoreCase("italic"))
                                        this.setFontStyle(Font.ITALIC);
                                    else
                                        this.setFontStyle(Font.PLAIN);
                                }
                            }

                        }
                        // coloca no StyledDoc
                        setStyleComponent(label.getComponent());
//                      coloca no StyledDoc
                        setStyleComponent(jTextArea);
                        this.addNewLine();
                        // coloca na lista
                        lista.add(this);
                }

                public JTextArea getComponent() {
                        return jTextArea;
                }
}
```

Arquivo GenericTag.Java

```
/*
 * Created on 05/11/2004
 *
 * To change the template for this generated file go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
package br.ufsc.ine.gui.xfdl.tags;

import java.awt.Color;
```

```
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.JComboBox;
import javax.swing.text.StyledDocument;

import org.dom4j.Element;
import org.dom4j.Node;

/**
 * @author mcc, rmpons
 *
 * To change the template for this generated type comment go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
public class Combobox extends GenericTag {
	public static final String TAGNAME = "combobox";

	private Label label;
	private JComboBox jCombo = null;
   private String group;

	public Combobox(Node n) {
		nodo = n;
		sid = ((Element) n).attributeValue("sid");
		label = new Label();
		jCombo = new JComboBox();
		jCombo.setBackground(Color.WHITE);
	}

	public void setValue(String valor) {
		if(((Element)nodo).element("value")==null) {
			Element e = ((Element)nodo).addElement("value");
			e.setText(valor);
		} else {
			Node n = ((Element)nodo).element("value");
			n.setText(valor);
		}
	}

	public void parseItem(ArrayList lista, StyledDocument doc) {
		this.stDoc = doc;
		Element ele = (Element) nodo;
		Iterator iter = ele.elementIterator();
		while (iter.hasNext()) {
			Node este = (Node) iter.next();
			//adiciona o conteudo de label para um texto ao lado dos checkbox
			if (este.getName().equalsIgnoreCase("label")) {
				// define texto do label
				label.setJLabelText(este.getText());
			} else if (este.getName().equalsIgnoreCase("group")) {
				this.setGroup(este.getStringValue());
			} else if (este.getName().equalsIgnoreCase("value")) {
				jCombo.addItem(este.getText());
				jCombo.setSelectedItem(este.getText());
			}
		}
		//coloca no StyledDoc
		setStyleComponent(label.getComponent());

		//coloca no StyledDoc
		setStyleComponent(jCombo);
		this.addNewLine();
		// coloca na lista
		lista.add(this);
	}

	public JComboBox getComponent() {
		return jCombo;
	}

	public void setSelectedItem(Object selected) {
		jCombo.setSelectedItem(selected);
```

```
	}

	public Object getSelected() {
		return jCombo.getSelectedItem();
	}

	public void addItemJCombo(String value) {
		int tam = jCombo.getItemCount();
		int i = 0;
		boolean add = true;
		while ((i<tam)) {
			if (((String)jCombo.getItemAt(i)).equalsIgnoreCase(value))
				add = false;
			i++;
		}
		if(add)
			jCombo.addItem(value);
	}

	public String getGroup() {
		return group;
	}

	public void setGroup(String string) {
		group = string;
	}

}
```

Arquivo Radio.Java

```
/*
 * Created on 05/11/2004
 *
 * To change the template for this generated file go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
package br.ufsc.ine.gui.xfdl.tags;

import java.awt.Color;
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.JRadioButton;
import javax.swing.text.StyledDocument;

import org.dom4j.Element;
import org.dom4j.Node;

/**
 * @author Administrador
 *
 * To change the template for this generated type comment go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
public class Radio extends GenericTag {
	public static final String TAGNAME = "radio";

	private Label label;
	private JRadioButton jRadio = null;
	private String group;

	public Radio(Node n) {
		nodo = n;
		sid = ((Element) n).attributeValue("sid");
		label = new Label();
		jRadio = new JRadioButton();
		jRadio.setBackground(Color.WHITE);
	}

	public void setValue(String valor) {
```

```
                if(((Element)nodo).element("value")==null) {
                        Element e = ((Element)nodo).addElement("value");
                        e.setText(valor);
                } else {
                        Node n = ((Element)nodo).element("value");
                        n.setText(valor);
                }
        }

        public void parseItem(ArrayList lista, StyledDocument doc) {
                this.stDoc = doc;
                Element ele = (Element) nodo;
                Iterator iter = ele.elementIterator();
                while (iter.hasNext()) {
                        Node este = (Node) iter.next();
                        //adiciona o conteudo de label para um texto ao lado dos checkbox
                        if (este.getName().equalsIgnoreCase("label")) {
                                // define texto do label
                                label.setJLabelText(este.getText());
                        } else if (este.getName().equalsIgnoreCase("group")) {
                                this.setGroup(este.getStringValue());

                        } else if (este.getName().equalsIgnoreCase("value")) {
         if(este.getText().equalsIgnoreCase("on")){
            this.setSelectedJRadio(true);
         }else{
            this.setSelectedJRadio(false);
         }
                        }
                }
                //coloca no StyledDoc
                setStyleComponent(label.getComponent());

                //coloca no StyledDoc
                setStyleComponent(jRadio);
                // coloca na lista
                this.addNewLine();
                lista.add(this);
        }

        public String getGroup() {
                return group;
        }

        public void setGroup(String string) {
                group = string;
        }

        public JRadioButton getComponent() {
                return jRadio;
        }

        public void setTextJRadio(String value) {
                jRadio.setText(value);
        }

   public void setSelectedJRadio(boolean bool){
      jRadio.setSelected(bool);
   }

}
```

Arquivo Label.Java

```
/*
 * Created on 04/11/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.gui.xfdl.tags;
```

```
import java.awt.Color;
import java.awt.Font;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.swing.JLabel;
import javax.swing.text.StyledDocument;

import org.dom4j.Element;
import org.dom4j.Node;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class Label extends GenericTag {
	public static final String TAGNAME = "label";
	private JLabel jLabel;
	private Font font;
	private String fontName;
	private int fontSize;
	private int fontStyle;

	public Label() {
		jLabel = new JLabel();
		font = new Font(null,Font.PLAIN,12);
		jLabel.setFont(font);
	}

	public Label(Node n) {
		jLabel = new JLabel();
		font = new Font(null,Font.PLAIN,12);
		jLabel.setFont(font);
		nodo = n;
	}

	public void setJLabelText(String texto) {
		jLabel.setText(texto);
	}

	public JLabel getComponent() {
		return jLabel;
	}

	public void setFontName(String name) {
		fontName = name;
		font = new Font(fontName,fontStyle,fontSize);
		jLabel.setFont(font);
	}

	public void setFontSize(int size) {
		fontSize = size;
		font = new Font(fontName,fontStyle,fontSize);
		jLabel.setFont(font);
	}

	public void setFontStyle(int style) {
		fontStyle = style;
		font = new Font(fontName,fontStyle,fontSize);
		jLabel.setFont(font);
	}

	public void setFontColor(Color cor) {
		jLabel.setForeground(cor);
	}

	public void parseItem(ArrayList lista, StyledDocument doc) {
		this.stDoc = doc;
		Element ele = (Element)nodo;
```

```
                Iterator iter = ele.elementIterator();
                while (iter.hasNext()) {
                        Node este = (Node) iter.next();
                        // texto do label
                        if(este.getName().equalsIgnoreCase("value")) {

                                jLabel.setText(este.getText());
                        } else if (este.getName().equalsIgnoreCase("fontinfo")) {
                                // dados da fonte
                                Element elFontInfo = (Element) este;
                                List l = elFontInfo.elements();
                                int tam = l.size();
                                if(tam>0)
                                        // nome da fonte
                                        this.setFontName(((Node)l.get(0)).getText());

                                if (tam>1) {
                                        // tamanho
                                        this.setFontSize(Integer.parseInt(((Node)l.get(1)).getText()));
                                }
                                if (tam>2) {
                                        // weigth
                                        String estilo = ((Node)l.get(2)).getText();

                                        if(estilo.equalsIgnoreCase("bold"))
                                                this.setFontStyle(Font.BOLD);
                                        else if(estilo.equalsIgnoreCase("italic"))
                                                this.setFontStyle(Font.ITALIC);
                                        else
                                                this.setFontStyle(Font.PLAIN);
                                }
                        } else if(este.getName().equalsIgnoreCase("fontcolor")) {
                                Element elFontInfo = (Element) este;
                                List l = elFontInfo.elements();
                                int tam = l.size();
                                if(tam>0) {
                                        // cor da fonte
                                        try {
                                                this.setFontColor(Color.getColor(((Node)l.get(0)).getText()));
                                        } catch (Exception e) {         }

                                }
                        }
                }
                // coloca no StyledDoc
                setStyleComponent(this.getComponent());
                this.addNewLine();
                // coloca na lista
                lista.add(this);
        }
}
```

Arquivo Combobox.Java

```
/*
 * Created on 05/11/2004
 *
 * To change the template for this generated file go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
package br.ufsc.ine.gui.xfdl.tags;

import java.awt.Color;
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.JComboBox;
import javax.swing.text.StyledDocument;

import org.dom4j.Element;
import org.dom4j.Node;
```

```
/**
 * @author mcc, rmpons
 *
 * To change the template for this generated type comment go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
public class Combobox extends GenericTag {
	public static final String TAGNAME = "combobox";

	private Label label;
	private JComboBox jCombo = null;
   private String group;

	public Combobox(Node n) {
		nodo = n;
		sid = ((Element) n).attributeValue("sid");
		label = new Label();
		jCombo = new JComboBox();
		jCombo.setBackground(Color.WHITE);
	}

	public void setValue(String valor) {
		if(((Element)nodo).element("value")==null) {
			Element e = ((Element)nodo).addElement("value");
			e.setText(valor);
		} else {
			Node n = ((Element)nodo).element("value");
			n.setText(valor);
		}
	}

	public void parseItem(ArrayList lista, StyledDocument doc) {
		this.stDoc = doc;
		Element ele = (Element) nodo;
		Iterator iter = ele.elementIterator();
		while (iter.hasNext()) {
			Node este = (Node) iter.next();
			//adiciona o conteudo de label para um texto ao lado dos checkbox
			if (este.getName().equalsIgnoreCase("label")) {
				// define texto do label
				label.setJLabelText(este.getText());
			} else if (este.getName().equalsIgnoreCase("group")) {
				this.setGroup(este.getStringValue());
			} else if (este.getName().equalsIgnoreCase("value")) {
				jCombo.addItem(este.getText());
				jCombo.setSelectedItem(este.getText());
			}
		}
		//coloca no StyledDoc
		setStyleComponent(label.getComponent());

		//coloca no StyledDoc
		setStyleComponent(jCombo);
		this.addNewLine();
		// coloca na lista
		lista.add(this);
	}

	public JComboBox getComponent() {
		return jCombo;
	}

	public void setSelectedItem(Object selected) {
		jCombo.setSelectedItem(selected);
	}

	public Object getSelected() {
		return jCombo.getSelectedItem();
	}

	public void addItemJCombo(String value) {
		int tam = jCombo.getItemCount();
```

```
                int i = 0;
                boolean add = true;
                while ((i<tam)) {
                        if (((String)jCombo.getItemAt(i)).equalsIgnoreCase(value))
                                add = false;
                        i++;
                }
                if(add)
                        jCombo.addItem(value);
        }

        public String getGroup() {
                return group;
        }

        public void setGroup(String string) {
                group = string;
        }

}
```

Arquivo Assinador.Java

```
/*
 * Created on 09/10/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.seguranca;

import java.security.PrivateKey;
import java.security.KeyStore;
import java.security.PublicKey;
import java.security.Security;
import java.security.cert.*;
import org.bouncycastle.cms.*;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

import java.util.*;

import java.io.*;
/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class Assinador {

        Certificado certificado;

        public Assinador() {

        }

        static {
                Security.addProvider(new BouncyCastleProvider());
        }

        public byte[] assina(byte[] __conteudo, String arquivoPfx, String senha) {
                try {
            // abre o arquivo pfx
            KeyStore keystore = KeyStore.getInstance("PKCS12");
            keystore.load(new FileInputStream(arquivoPfx), senha.toCharArray());
            String alias = null;

            // pega o alias do certificado
            for(Enumeration e = keystore.aliases() ; e.hasMoreElements() ;) {
               alias = e.nextElement().toString();
            }
```

```
            // pega o caminho de certificacao
            Certificate[] certChain = keystore.getCertificateChain(alias);

            // inicia variaveis
            ArrayList certList = new ArrayList();
            // adiciona certificados do caminho em uma lista
            for ( int i = 0; i < certChain.length;i++)
               certList.add(certChain[i]);
            // pega a CertStore
            CertStore certS = CertStore.getInstance("Collection", new CollectionCertStoreParameters(certList), "SUN");
            // pega a chave privada
            PrivateKey priv = (PrivateKey)(keystore.getKey(alias, senha.toCharArray()));
            // pega o certificado
            Certificate cert = keystore.getCertificate(alias);
            certificado = new Certificado(cert);
            // pega a chave publica
            PublicKey pub = keystore.getCertificate(alias).getPublicKey();

            //--- Gerando assinatura ---
            // inicializa gerador
            CMSSignedDataGenerator signGen = new CMSSignedDataGenerator();
            // adiciona certificado, chave privada e o algoritmo
            signGen.addSigner(priv, (X509Certificate)cert, CMSSignedDataGenerator.DIGEST_SHA1);
            // adiciona caminho de certificacao
            signGen.addCertificatesAndCRLs(certS);
            CMSProcessable content = new CMSProcessableByteArray(__conteudo);
            // gerando assinatura
            CMSSignedData signedData = signGen.generate(content,"BC");
            return signedData.getEncoded();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
            }

            public Certificado getCertificado() {
                    return certificado;
            }

}
```

Arquivo Certificado.Java

```
/*
 * Created on 01/11/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.seguranca;

import java.security.cert.Certificate;
import java.security.cert.CertificateExpiredException;
import java.security.cert.CertificateNotYetValidException;
import java.security.cert.X509Certificate;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to Window -
 * Preferences - Java - Code Style - Code Templates
 */
public class Certificado {

            private Certificate cert;

            private X509Certificate x509cert;

            public Certificado(Certificate certificado) {
```

```
		// TODO Auto-generated constructor stub
		cert = certificado;
		x509cert = (X509Certificate) cert;
	}

	public boolean ehValido() {
		try {
			x509cert.checkValidity();
			return true;
		} catch (CertificateExpiredException e) {
			return false;
		} catch (CertificateNotYetValidException e) {
			return false;
		}
	}

	public String getAssunto() {
		return x509cert.getSubjectDN().getName();
	}

	public String getAssuntoCN() {
		return getCampoAssunto("CN=");
	}

	public String getAssuntoO() {
		return getCampoAssunto("O=");
	}

	public String getAssuntoOU() {
		return getCampoAssunto("OU=");
	}

	public String getAssuntoL() {
		return getCampoAssunto("L=");
	}

	public String getAssuntoS() {
		return getCampoAssunto("ST=");
	}

	public String getAssuntoC() {
		return getCampoAssunto("C=");
	}

	public String getAssuntoE() {
		return getCampoAssunto("EMAILADDRESS=");
	}

	public String getEmissor() {
		return x509cert.getIssuerDN().getName();
	}

	public String getEmissorCN() {
		return getCampoEmissor("CN=");
	}

	public String getEmissorO() {
		return getCampoEmissor("O=");
	}

	public String getEmissorOU() {
		return getCampoEmissor("OU=");
	}

	public String getEmissorL() {
		return getCampoEmissor("L=");
	}

	public String getEmissorS() {
		return getCampoEmissor("S=");
	}

	public String getEmissorC() {
```

```
                return getCampoEmissor("C=");
        }

        public String getEmissorE() {
                return getCampoEmissor("EMAILADDRESS=");
        }

        public String getPontosCRL() {
                try {
          byte[] b = x509cert.getExtensionValue("2.5.29.31");
          String ponto = new String(b);
          return ponto.substring(ponto.indexOf("http"),ponto.lastIndexOf(".crl")+4);
      } catch (NullPointerException e) {
          return null;
      }

        }

        private String getCampoAssunto(String Field) {
                String cn = x509cert.getSubjectDN().getName();
                String[] aux = cn.split(Field);
                aux = aux[1].split(",");
                return aux[0];
        }

        private String getCampoEmissor(String Field) {
                String cn = x509cert.getIssuerDN().getName();
                String[] aux = cn.split(Field);
                aux = aux[1].split(",");
                return aux[0];
        }

}
```

Arquivo Verificador.Java

```
/*
 * Created on 09/10/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.seguranca;

import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Security;
import java.security.cert.CertStore;
import java.security.cert.CertStoreException;
import java.security.cert.Certificate;
import java.security.cert.CertificateExpiredException;
import java.security.cert.CertificateNotYetValidException;
import java.security.cert.X509Certificate;
import java.util.Collection;
import java.util.Iterator;

import org.bouncycastle.asn1.DERObjectIdentifier;
import org.bouncycastle.asn1.DERUTCTime;
import org.bouncycastle.asn1.cms.Attribute;
import org.bouncycastle.asn1.cms.AttributeTable;
import org.bouncycastle.cms.CMSException;
import org.bouncycastle.cms.CMSProcessableByteArray;
import org.bouncycastle.cms.CMSSignedData;
import org.bouncycastle.cms.SignerInformation;
import org.bouncycastle.cms.SignerInformationStore;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to Window -
```

```
 * Preferences - Java - Code Style - Code Templates
 */
public class Verificador {

	static {
		Security.addProvider(new BouncyCastleProvider());
	}

	private Certificado certificado;

	private SignerInformation sinfo;

	public Verificador() {

	}

	public int verifica(byte[] signedData, byte[] original) {
		try {
			// pega dados da assinatura
			CMSProcessableByteArray cba = new CMSProcessableByteArray(original);
			CMSSignedData sd = new CMSSignedData(cba,signedData);
        CertStore certs = sd.getCertificatesAndCRLs ("Collection","SUN");
        SignerInformationStore sis = sd.getSignerInfos();
        Collection signerColl = sis.getSigners();
        // para cada assinatura
        for (Iterator signerIt = signerColl.iterator(); signerIt.hasNext(); ) {
           sinfo = (SignerInformation) signerIt.next();
           Collection certColl = certs.getCertificates (sinfo.getSID());
           // para cada certificado
           for (Iterator certIt = certColl.iterator(); certIt.hasNext(); ) {
              Certificate cert = (Certificate)certIt.next();
              X509Certificate certv = (X509Certificate)cert;
              certificado = new Certificado(cert);
              try {
                 if (sinfo.verify(certv,"BC")) {
                    return 1;
                 } else {
                    // hashs diferentes
                    return -200;
                 }
              } catch (CertificateExpiredException e) {
                 return -101;
              } catch (CertificateNotYetValidException e) {
                 return -102;
              }
           }
        }
			return -100;
		} catch (NoSuchAlgorithmException e) {
			return -201;
		} catch (NoSuchProviderException e) {
			return -202;
		} catch (CMSException e) {
			return -200;
		} catch (CertStoreException e) {
			return -200;
		}

	}

	public Certificado getCertificado() {
		return certificado;
	}

	public String getDataUTC() {
		AttributeTable attribTable = sinfo.getSignedAttributes();
		Attribute att = attribTable.get(new DERObjectIdentifier(
				"1.2.840.113549.1.9.5"));
		DERUTCTime dutc = (DERUTCTime) att.getAttrValues().getObjectAt(0);
		String aux = dutc.getTime();

		String dia, mes, ano, hora, minutos, segundos;
		ano = aux.substring(0, 2);
```

```
                mes = aux.substring(2, 4);
                dia = aux.substring(4, 6);
                hora = aux.substring(6, 8);
                minutos = aux.substring(8, 10);
                segundos = aux.substring(10, 12);

                return (dia + "/" + mes + "/" + ano + " " + hora + ":" + minutos + ":" + segundos);
        }

}
```

Arquivo XFDLCifrador.java

```
/*
 * Created on 11/10/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.xfdl.crypto;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import java.security.Key;
import java.security.PublicKey;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

import javax.crypto.SecretKey;
import javax.crypto.KeyGenerator;

import org.apache.xml.security.keys.KeyInfo;
import org.apache.xml.security.encryption.XMLCipher;
import org.apache.xml.security.encryption.EncryptedData;
import org.apache.xml.security.encryption.EncryptedKey;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.OutputKeys;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to Window -
 * Preferences - Java - Code Style - Code Templates
 */
public class XFDLCifrador {

        private Document document;

        private Element raiz;

        static {
                org.apache.xml.security.Init.init();
        }

        public XFDLCifrador(String nomeArquivo) {
                try {
```

```
                    DocumentBuilderFactory _factory = DocumentBuilderFactory.newInstance();
                    DocumentBuilder _builder = _factory.newDocumentBuilder();
                    document = _builder.parse(new File(nomeArquivo));
                    raiz = document.getDocumentElement();
            } catch (SAXParseException spe) {
                    spe.printStackTrace();
            } catch (ParserConfigurationException pce) {
                    pce.printStackTrace();
            } catch (IOException ioe) {
                    ioe.printStackTrace();
            } catch (SAXException sxe) {
                    sxe.printStackTrace();
            }
}

public void cifra(String __arqSaida, String __arqCert) throws XFDLCifradorException {
        try {
                    // gera chave para cifragem dos dados para o algoritmo AES
                    Key symmetricKey = geraChaveCifragemDados();

                    // gera chave para cifrar a chave (Triple DES) e armazena seu conteudo em disco

    FileInputStream is = new FileInputStream(__arqCert);

    CertificateFactory cf = CertificateFactory.getInstance("X.509");
    java.security.cert.Certificate cert = cf.generateCertificate(is);
    PublicKey kek = cert.getPublicKey();

                    // define algoritmo para cifrar chave
                    String algorithmURI = XMLCipher.RSA_v1dot5;
                    XMLCipher keyCipher = XMLCipher.getInstance(algorithmURI);
                    // inicia processo de cifragem
                    keyCipher.init(XMLCipher.WRAP_MODE, kek);
                    // cifra a chave
                    EncryptedKey encryptedKey = keyCipher.encryptKey(document, symmetricKey);

                    // pega o elemento raiz
                    Element rootElement = document.getDocumentElement();
                    // define algoritmo
                    algorithmURI = XMLCipher.AES_128;
                    XMLCipher xmlCipher = XMLCipher.getInstance(algorithmURI);
                    // inicia processo de cifragem dos dados com a chave gerada
                    xmlCipher.init(XMLCipher.ENCRYPT_MODE, symmetricKey);

                    // define keyinfo para os dados cifrados
                    EncryptedData encryptedData = xmlCipher.getEncryptedData();
                    KeyInfo keyInfo = new KeyInfo(document);
                    keyInfo.add(encryptedKey);
                    encryptedData.setKeyInfo(keyInfo);

                    // cifra o conteudo do documento
                    xmlCipher.doFinal(document, rootElement, true);

                    // grava em disco o documento cifrado
                    gravaArquivo(document, __arqSaida);
            } catch (Exception e) {
                    e.printStackTrace();
                    throw new XFDLCifradorException("Erro cifrando arquivo");
            }
}

private static SecretKey geraChaveCifragemDados() throws Exception {
            String jceAlgorithmName = "AES";
            KeyGenerator keyGenerator = KeyGenerator.getInstance(jceAlgorithmName);
            keyGenerator.init(128);
            return keyGenerator.generateKey();
}

private static void gravaArquivo(Document doc, String fileName)
                    throws Exception {
            File encryptionFile = new File(fileName);
            FileOutputStream f = new FileOutputStream(encryptionFile);
```

```
                TransformerFactory factory = TransformerFactory.newInstance();
                Transformer transformer = factory.newTransformer();
                transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");
                DOMSource source = new DOMSource(doc);
                f.write("<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\n".getBytes());
                StreamResult result = new StreamResult(f);
                transformer.transform(source, result);
                f.close();
        }
}
```

Arquivo XFDLCifradorException.Java

```
/*
 * Created on 01/11/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.xfdl.crypto;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class XFDLCifradorException extends Exception {


        public XFDLCifradorException(String arg0) {
                super(arg0);
                // TODO Auto-generated constructor stub
        }


}
```

Arquivo XFDLDecifradorInvalidKeyException.java

```
/*
 * Created on 01/11/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.xfdl.crypto;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class XFDLDecifradorInvalidKeyException extends Exception {

        /**
         * @param arg0
         */
        public XFDLDecifradorInvalidKeyException(String arg0) {
                super(arg0);
                // TODO Auto-generated constructor stub
        }

}
```

Arquivo XFDLDecifradorException.java

```
/*
 * Created on 01/11/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.xfdl.crypto;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class XFDLDecifradorException extends Exception {

	public XFDLDecifradorException(String arg0) {
		super(arg0);
		// TODO Auto-generated constructor stub
	}


}
```

Arquivo XFDLDecifrador.java

```
/*
 * Created on 11/10/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.xfdl.crypto;

import org.w3c.dom.Document;
import org.w3c.dom.Element;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

import java.security.Key;
import java.security.KeyStore;
import java.security.PrivateKey;
import java.security.Security;
import java.security.cert.CertStore;
import java.security.cert.Certificate;
import java.security.cert.CollectionCertStoreParameters;
import java.util.ArrayList;
import java.util.Enumeration;

import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESedeKeySpec;

import org.apache.xml.security.encryption.XMLCipher;
import org.apache.xml.security.utils.JavaUtils;
import org.apache.xml.security.utils.EncryptionConstants;


import javax.xml.transform.TransformerFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.TransformerException;

/**
 * @author mcc
```

```
 *
 * TODO To change the template for this generated type comment go to Window -
 * Preferences - Java - Code Style - Code Templates
 */
public class XFDLDecifrador {

	private File arquivoCifrado;
	private File arquivoPfx;
	private String senha;

	static {
		org.apache.xml.security.Init.init();
	}

	public XFDLDecifrador(String __arquivoCifrado, String __arquivoPfx, String __senha)
		throws FileNotFoundException {

		arquivoCifrado = new File(__arquivoCifrado);
		if (!arquivoCifrado.exists())
			throw new FileNotFoundException();

		arquivoPfx = new File(__arquivoPfx);
		if (!arquivoPfx.exists())
			throw new FileNotFoundException();
		senha = __senha;
	}

	public void decifra(String __arquivoSaida) throws XFDLDecifradorException, XFDLDecifradorInvalidKeyException {
		Document document;
		try {
			// carrega o documento original
			document = carregaDocumento();
		} catch (Exception e) {
			throw new XFDLDecifradorException("Erro carregando documento cifrado");
		}
		// pega o conteudo cifrado
		Element encryptedDataElement = (Element) document
			.getElementsByTagNameNS(EncryptionConstants.EncryptionSpecNS,
					EncryptionConstants._TAG_ENCRYPTEDDATA).item(0);

		PrivateKey kek=null;
		try {
		// carrega chave utilizada para cifrar a chave
	// abre o arquivo pfx
	KeyStore keystore = KeyStore.getInstance("PKCS12");
	keystore.load(new FileInputStream(arquivoPfx), senha.toCharArray());
	String alias = null;

	// pega o alias do certificado
	for(Enumeration e = keystore.aliases() ; e.hasMoreElements() ;) {
	   alias = e.nextElement().toString();
	}
	// pega a chave privada
	kek = (PrivateKey)(keystore.getKey(alias, "teste".toCharArray()));;

		} catch (Exception e) {
			e.printStackTrace();
		}

		// define o provider
		String providerName = "BC";
		// incia o objeto xmlCipher
		try {
			XMLCipher xmlCipher = XMLCipher.getInstance();
			//		define a chav utilzada para cifra a chave de cifragem dos dados
			xmlCipher.init(XMLCipher.DECRYPT_MODE, null);
			xmlCipher.setKEK(kek);
			//		subistitui os dados cifrados pelos decifrados
			xmlCipher.doFinal(document, encryptedDataElement);
		} catch (Exception e) {
			e.printStackTrace();
			throw new XFDLDecifradorException("Erro decifrando arquivo");
		}
```

```
                try {
                        gravaArquivo(document, __arquivoSaida);
                } catch (Exception e) {
                        e.printStackTrace();
                        throw new XFDLDecifradorException("Erro gravando arquivo");
                }
        }

        private Document carregaDocumento() throws Exception {
                javax.xml.parsers.DocumentBuilderFactory dbf = javax.xml.parsers.DocumentBuilderFactory
                                .newInstance();
                dbf.setNamespaceAware(true);
                javax.xml.parsers.DocumentBuilder db = dbf.newDocumentBuilder();
                Document document = db.parse(arquivoCifrado);
                return document;
        }

        private void gravaArquivo(Document doc, String fileName)
                        throws IOException, TransformerException {
                File encryptionFile = new File(fileName);
                FileOutputStream f = new FileOutputStream(encryptionFile);

                TransformerFactory factory = TransformerFactory.newInstance();
                Transformer transformer = factory.newTransformer();
                transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");
                DOMSource source = new DOMSource(doc);
                f.write("<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\n".getBytes());
                StreamResult result = new StreamResult(f);
                transformer.transform(source, result);

                f.close();
        }

}
```

Arquivo DadosAssinatura.java

```
/*
 * Created on 05/11/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.xfdl.signer;

import br.ufsc.ine.seguranca.Certificado;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class DadosAssinatura {

                private Certificado cert;
                private int status;
                private byte[] data;
                private String dia;

                public DadosAssinatura(Certificado c, String dia, byte[] d, int s) {
                        cert = c;
                        data = d;
                        status = s;
                        this.dia = dia;
                }

                public Certificado getCert() {
                        return cert;
                }
```

```
		public byte[] getData() {
			return data;
		}

		public int getStatus() {
			return status;
		}

		public String getDataAssinatura() {
			return dia;
		}

}
```

Arquivo XFDLVerificador.java

```
/*
 * Created on 11/10/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.xfdl.signer;

import java.io.ByteArrayOutputStream;
import java.util.ArrayList;
import java.util.Iterator;

import org.dom4j.Document;
import org.dom4j.Element;
import org.dom4j.io.OutputFormat;
import org.dom4j.io.XMLWriter;

import br.ufsc.ine.seguranca.Certificado;
import br.ufsc.ine.seguranca.Verificador;
import br.ufsc.ine.util.Base64;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class XFDLVerificador {

	private Document doc;
	private Document docClone;

	private Element raiz;

	private ArrayList listaAssinaturas;
	private ArrayList listaDadosAssinaturas;

	public XFDLVerificador(Document __xfdl) {
		doc = __xfdl;
		docClone = (Document)doc.clone();
		raiz = docClone.getRootElement();
		listaAssinaturas = new ArrayList();
		listaDadosAssinaturas = new ArrayList();
	}

	public ArrayList getListDadosAssinaturas() {
		return listaDadosAssinaturas;
	}

	/**
	 * Verifica todas as assinaturas do documento.
	 * Caso alguma das assinaturas seja invalida retorna falso.
	 * Os dados das assinaturas podem ser recuperados atraves dos metodos getCertificado(), getStatusAssinatura
	 */
	public boolean verifica() {
```

```
		boolean retorno = true;
		removeAssinaturas();
		byte[] b = getBytes();
		Verificador v = new Verificador();
		Iterator i = listaAssinaturas.iterator();
		while (i.hasNext()) {
			Element assinatura = (Element)i.next();
			Element mime = assinatura.element("mimedata");
			byte[] mimedata = Base64.decode(mime.getText());
			int status = v.verifica(mimedata,b);
			System.out.println(status);
			if(status<0)
				retorno = false;
			listaDadosAssinaturas.add(new DadosAssinatura(v.getCertificado(),v.getDataUTC(),mimedata,status));
		}
		return retorno;
	}

	public Certificado getCertificado(int i) {
		if (i>=listaDadosAssinaturas.size())
			return null;
		else {
			return ((DadosAssinatura)listaDadosAssinaturas.get(i)).getCert();
		}
	}

	public int getCountCertificados() {
		return listaDadosAssinaturas.size();
	}

	/**
	 * Pega o status da assinatura indice i na lista de assinaturas
	 * @param i indice da lista de assinaturas
	 * @return status na posicao i, se a posicao nao existir retorna -1
	 */
	public int getStatus(int i) {
		if (i>=listaDadosAssinaturas.size())
			return -1;
		else {
			return ((DadosAssinatura)listaDadosAssinaturas.get(i)).getStatus();
		}
	}

	private byte[] getBytes() {
		OutputFormat format = new OutputFormat("", false, "iso-8859-1");
		ByteArrayOutputStream saida = new ByteArrayOutputStream();
		XMLWriter writer = null;
		try {
			writer = new XMLWriter(saida, format);
			writer.write(docClone);
			writer.flush();
		} catch (Exception e) {
			e.printStackTrace();
		}
		return removeDuploBarraN(saida.toByteArray());
	}

	private void removeAssinaturas() {
		Iterator i = raiz.elementIterator("signature");
		while (i.hasNext()) {
			Element e = (Element) i.next();
			listaAssinaturas.add(e);
			raiz.remove(e);
		}
	}

	private byte[] removeDuploBarraN(byte[] b) {
		try {
			ByteArrayOutputStream saida = new ByteArrayOutputStream();
			int l = b.length;
			for (int i = 0; i < l; i++) {
				if (b[i] == 10) {
					if (b[i + 1] != 10) {
```

```
                                        saida.write(10);
                                    }
                                } else {
                                    saida.write(b[i]);
                                }
                            }
                            byte[] retorno = saida.toByteArray();
                            saida.close();
                            return retorno;
                        } catch (Exception e) {
                            e.printStackTrace();
                            return null;
                        }
                    }
}
```

Arquivo XFDLAssinador.java

```
/*
 * Created on 11/10/2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package br.ufsc.ine.xfdl.signer;

import java.io.ByteArrayOutputStream;
import java.util.Iterator;

import org.dom4j.Document;
import org.dom4j.Element;
import org.dom4j.io.OutputFormat;
import org.dom4j.io.XMLWriter;

import br.ufsc.ine.seguranca.Assinador;
import br.ufsc.ine.seguranca.Certificado;
import br.ufsc.ine.util.Base64;

/**
 * @author mcc
 *
 * TODO To change the template for this generated type comment go to Window -
 * Preferences - Java - Code Style - Code Templates
 */
public class XFDLAssinador {

    private Document doc;
    private Document docClone;

    private Element raiz;

    public XFDLAssinador(Document __xfdl) {
        doc = __xfdl;
        docClone = (Document)doc.clone();
        raiz = docClone.getRootElement();
    }

    public Document assina(String arqPfx, String senha) {
        removeAssinaturas();
        byte[] b = getBytes();
        Assinador assinador = new Assinador();
        byte[] assinatura = assinador.assina(b,arqPfx,senha);
        String mimedata = Base64.encodeBytes(assinatura);
        // cria o elemento signature
        Element sig = doc.getRootElement().addElement("signature");
        // cria o elemento signformat
        Element format = sig.addElement("signformat");
        format.setText("application/x-xfdl");
        // cria o elemento signer
```

```
                Certificado cer = assinador.getCertificado();
                Element signer = sig.addElement("signer");
                signer.setText(cer.getAssuntoCN()+", "+cer.getAssuntoE());
                // cria o elemento fullname
                Element fullname = sig.addElement("fullname");
                fullname.setText(cer.getAssunto());
                // cria o mimedata
                Element mime = sig.addElement("mimedata");
                mime.setText(mimedata);
                return doc;
        }


        private byte[] getBytes() {
                OutputFormat format = new OutputFormat("", false, "iso-8859-1");
                ByteArrayOutputStream saida = new ByteArrayOutputStream();
                XMLWriter writer = null;
                try {
                        writer = new XMLWriter(saida, format);
                        writer.write(docClone);
                        writer.flush();
                } catch (Exception e) {
                        e.printStackTrace();
                }
                return removeDuploBarraN(saida.toByteArray());
        }

        private void removeAssinaturas() {
                Iterator i = raiz.elementIterator("signature");
                while (i.hasNext()) {
                        Element e = (Element) i.next();
                        raiz.remove(e);
                }
        }

        private byte[] removeDuploBarraN(byte[] b) {
                try {
                        ByteArrayOutputStream saida = new ByteArrayOutputStream();
                        int l = b.length;
                        for (int i = 0; i < l; i++) {
                                if (b[i] == 10) {
                                        if (b[i + 1] != 10) {
                                                saida.write(10);
                                        }
                                } else {
                                        saida.write(b[i]);
                                }
                        }
                        byte[] retorno = saida.toByteArray();
                        saida.close();
                        return retorno;
                } catch (Exception e) {
                        e.printStackTrace();
                        return null;
                }
        }

}
```