**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

**CURSO DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO**

**Gabriel de Carvalho Nogueira Reis**

**Victor Hugo Germano**

# INAFF – Um Framework para Suporte a Bases de Dados Seguras na Web

**Trabalho de Conclusão de Curso**

Prof. Luiz Carlos Zancanella, Dr.
Orientador

Florianópolis, Junho de 2004.

# INAFF – Um Framework para Suporte a Bases de Dados Seguras na Web

Gabriel de Carvalho Nogueira Reis

Victor Hugo Germano

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

_____

Prof. José Mazzuco Jr., Dr.
Coordenador do Curso

Banca Examinadora

_____

Prof. Luiz Carlos Zancanella, Dr.
Orientador

_____

Prof. Ricardo Felipe Custódio, Dr.
Co-orientador

_____

Iuri Campana

_____

Sérgio Roberto de Lima e Silva Filho

*“Nossas dúvidas são traidoras e nos fazem perder o quê,*
*com freqüência, poderíamos ganhar,*
*pelo simples medo de arriscar”.*
*(William J. Shakespeare)*

*Aos nossos pais.*

# Agradecimentos

Gostaria de agradecer especialmente os meus pais, Ademir e Ana Helena, por todo o amor, carinho, incentivo e apoio que me deram desde o meu primeiro dia como ser humano, até o dia de hoje, no qual concluo este trabalho como sendo mais um passo dado a um futuro profissional que apenas está começando. Eles nunca deixaram de acreditar em mim, e foram muito importantes para que eu pudesse crescer saudavelmente ao longo destes quatro anos e meio que se passaram desde o dia que eu, ainda com dezesseis anos, recebera a notícia de que tinha passado no vestibular naquela distante e maravilhosa Ilha da Magia, e dois meses mais tarde estaria vindo morar aqui, sozinho, porém sempre com a certeza de que seja lá o que acontecesse eu poderia contar com eles.

À minha irmã, Laura, que sempre se preocupou comigo, mesmo estando longe. Apesar da diferença da idade, sinto muita saudade dela e gostaria que pudéssemos passar mais tempo juntos.

À minha melhor amiga e namorada, Larissa, por sua compreensão nos momentos mais difíceis que passei na reta final da minha faculdade. Obrigado por sempre ter feito com que eu acreditasse mais em mim, e me esforçasse mais para atingir meus objetivos. Do seu lado os sonhos mais impossíveis parecem tornarem-se fáceis de realizar.

À Silvinha, minha prima daqui, que eu tanto adoro. Ela sempre me ajudou quando eu precisei. Além disso, me deu o prazer de ser padrinho da sua filhinha caçula, a doce Giovanna.

Aos meus amigos que conheci aqui em Floripa, especialmente o Animalzinho e Túlio, que eu tenho muita felicidade em dizer que eu tive sorte de conhecer as melhores figuras por aqui. Vamos sempre aproveitar os bons momentos, pois são esses que serão lembrados com aquele gostinho de saudade para sempre, não é?

Aos meus amigos e amigas de Marília, que mesmo cada um tendo seguido seu caminho diferente, ainda fazem parte da minha vida, e adoro todos vocês. Espero sempre ansiosamente pela chegada de um feriadão para que a gente possa se divertir em nossa querida cidade. Valeu Rafael, Marco, Rodrigo e Daniel.

Àqueles que ajudaram na realização deste trabalho, principalmente o meu professor e orientador Luiz Carlos Zancanella, pela atenção dispensada durante a realização

deste trabalhado e por todos os seus ensinamentos e confiança, e meu amigo Victor Hugo Germano.

Aos diretores da Directa Automação, pela oportunidade a mim confiada. Aproveito também para agradecer os meus colegas de trabalho, em especial o Iuri Campana, que se tornou meu amigo.

Ao Criador, por permitir que eu tenha energia para poder desfrutar de um prazer imensurável que é viver.

Finalmente gostaria de agradecer a todos aqueles que, embora não tenham sido citados, são meus amigos e contribuíram de alguma forma para o meu amadurecimento como pessoa.

***Gabriel de Carvalho Nogueira Reis***

# Sumário

# Lista de Figuras

## Capítulo 2



## Capítulo 4



## Capítulo 6

# Lista de Siglas

| | |
|---|---|
| AC | Autoridade Certificadora |
| AES | Advanced Encryption Standard |
| ANSI | American National Standards Institute |
| API | Application Program Interface |
| ASP | Active Server Pages |
| CGI | Common Gateway Interface |
| COM | Component Object Model |
| CRL | Certification Revocation List |
| DES | Data Encryption Standard |
| GNU | GNU's Not Unix |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IDE | Integrated Development Environment |
| IIS | Internet Information Services |
| ISO | International Organization for Standardization |
| ITU-T | International Telecommunications Unions |
| MD5 | Message Digest 5 |
| MIT | Massachussets Institute of Technology |
| MSDN | Microsoft Developer Network |
| NCSA | National Center for Supercomputing Applications |
| PDF | Portable Document Format |
| PHP | PHP: Hypertext Processor |
| RSA | Rivest Shamir Adler |
| SGBD | Sistema de Gerenciamento de Banco de Dados |
| SHA | Secure Hash Algorithm |
| SQL | Structured Query Language |
| UFSC | Universidade Federal de Santa Catarina |

# Resumo

Este trabalho apresenta o desenvolvimento de uma possível solução para dar suporte à segurança em base de dados na Web.

Serão apresentados os conceitos relacionados ao desenvolvimento de *sites* Web que interagem com algum Banco de Dados através de formulários, os fundamentos de criptografia necessários para a compreensão e implementação de assinaturas digitais, e como isso resolve o problema de autenticidade de dados armazenados em uma base de dados na Web.

O resultado prático deste projeto de pesquisa é uma ferramenta que tem como principal objetivo minimizar os esforços do programador na tarefa de buscar uma solução para prover segurança aos dados que são exibidos em sua página Web. Ao final deste trabalho também será abordado um exemplo prático de uso real desta ferramenta.

Palavras-chave: criptografia, assinaturas digitais, segurança.

# Abstract

This work presents the development of a possible solution to support security into databases in Web *sites*.

It will be presented related concepts about Web *sites* development which interact with some Data Base through web forms, cryptography fundaments needed to comprehend and implement digital signatures, and how it solves the data integrity problem in a database under Web.

The practical result of this research project is a tool, the main function of which is to minimize the programmer work in order to do the job of finding a solution to provide security for information that will be supposed to be shown in your Web page. Finally, a practical example of the real use of this tool will be approached.

Keywords: cryptography, digital signatures, security.

# Capítulo 1

# Introdução

Neste capítulo iremos destacar a motivação do desenvolvimento deste projeto, contextualizando-o e apresentando os objetivos gerais e específicos, a metodologia utilizada e como se encontra organizado este texto.

## 1.1 Justificativa

Hoje a humanidade atravessa a Era da Informação. Com o recente *boom* da Internet, milhões de pessoas se conectam diariamente na rede e nela descarregam uma quantidade enorme de dados. Atualmente uma das maiores preocupações dos profissionais da área de Segurança da Informação é fornecer credibilidade a um conjunto de dados que você está enviando ou recebendo.

O que existe hoje é um grande universo de páginas na Internet que contém um formulário para entrada de dados e outras páginas que exibem resultados de consultas e pesquisas feitas em uma base de dados. Também é importante saber que poucos *web sites* oferecem algum mecanismo de segurança para esses dados.

O fato de um *site* ser seguro ou não pode ser determinante para medir o nível de confiança de uma empresa no mundo digital, o que pode ser crucial para o crescimento da mesma no mercado financeiro. Portanto, é do interesse de analistas, desenvolvedores e *webmasters* que esforços não sejam poupados para se oferecer segurança para seus clientes e usuários.

Uma das tecnologias que vem sendo empregada freqüentemente para prover um certo nível de segurança em sistemas computacionais na Web é a assinatura digital, pois oferece: integridade, autenticidade e não-repúdio. Muitos estudos e pesquisas vêm sendo realizados para desenvolver e aprimorar esta tecnologia.

Porém, o problema que se observa é que o trabalho que um programador tem para adicionar recursos de assinatura digital em um sistema na Web é um tanto quanto penoso.

Primeiramente, o analista deve ter conhecimentos básicos de segurança computacional como, por exemplo, técnicas de criptografia. Se não o tiver, terá que dispensar algum tempo para adquirir este conhecimento. Isso, então, significa que será necessário mais tempo para colocar o sistema seguro em funcionamento, o que por sua vez implica diretamente em um maior custo do projeto.

A situação ideal seria aquela em que o programador já dispondo de um sistema pronto pudesse, fazendo o mínimo possível de alterações, adicionar recursos de assinatura digital a um conjunto de dados que interesse a ele. Um mecanismo que fosse fácil de usar e, portanto, não consumisse muito do precioso tempo que dispõe o desenvolvedor.

## 1.2 Objetivos

### 1.2.1 Objetivo geral

O objetivo geral deste trabalho é procurar e implementar uma solução para o problema da dificuldade em se implementar assinatura digital em cima de dados que trafegam a partir de páginas da Web para base de dados, e vice-versa. Para tal, pretendemos construir um aplicativo, inicialmente batizado de INAFF, na forma de um *framework*, que poderá ser utilizado pelos programadores que desenvolvem aplicativos Web para fornecer recursos de segurança de dados em suas aplicações, com poucas linhas de código, de maneira muito rápida e que seja fácil de usar por qualquer um que não conheça profundamente criptografia, mas que deseje oferecer tais recursos para seus usuários.

### 1.2.2 Objetivos específicos

- Desenvolver inicialmente o *framework* em PHP. Com o conhecimento adquirido migrar a aplicação para outras linguagens;

- Utilizar a biblioteca de criptografia da Microsoft, CAPICOM, no código gerado pelo *framework*, para utilizar os recursos de assinaturas digitais, principalmente funções de gerar e verificar assinaturas;
- Entender o funcionamento das assinaturas digitais e como esta tecnologia pode ser empregada para prover segurança em formulários e base de dados na Web;
- Executar uma bateria de testes que comprovem a eficiência e eficácia do INAFF, apresentado os resultados obtidos;
- Criar uma aplicação robusta e de fácil utilização e aprendizado pela parte dos *webmasters*, tornando-se uma referência no desenvolvimento de aplicações Web seguras.

## 1.3 Materiais e métodos

O desenvolvimento deste trabalho foi feito em etapas:

1. Inicialmente foi realizada uma coleta de artigos, tutoriais, documentação sobre criptografia e assinatura digital. Também reunimos material sobre a biblioteca CAPICOM, incluindo documentação e exemplos práticos de uso da mesma;
2. Em segundo lugar, foi dado início à etapa de análise, quando foi feito um levantamento de requisitos funcionais do nosso *framework*. Nesta etapa, basicamente foram buscadas respostas para perguntas como “O que nosso software deve fazer?”, “Como ele deve ser?”, “O que ele deve ter?”, “Como será a interação com o usuário?”, “Que linguagens e tecnologias serão empregadas?”, entre outras.
3. A terceira etapa consistiu-se da implementação do *framework*. Ao final desta etapa, tínhamos condições de verificar se os objetivos planejados no início do projeto foram atingidos ou não.

4. Elaboração de um estudo de caso para o uso do *framework*. Neste passo, foram feitos vários testes que possibilitaram que melhorias e correções fossem feitas antes de entregar a versão final do trabalho.
5. Finalmente, foi escrita essa monografia e elaborado um pequeno manual de utilização da ferramenta, produto de todo este trabalho de conclusão de curso.

Ao longo de todas essas etapas o uso da Internet foi indispensável, pois nela se encontra a maior parte da fonte de estudo que utilizamos para adquirir os conhecimentos e tecnologias necessários para a elaboração do projeto.

Também foram feitas diversas reuniões com o Prof. Luiz Carlos Zancanella, para que este pudesse tirar nossas dúvidas e ajudar na especificação dos requisitos do *framework*.

Foram utilizados os recursos bibliográficos disponíveis na biblioteca da Universidade Federal de Santa Catarina, principalmente de dissertações de mestrado e de trabalhos finais de graduação submetidos a avaliação no Departamento de Informática e Estatística da UFSC.

## 1.4 Organização do texto

O restante deste trabalho encontra-se assim estruturado: no capítulo 2 são apresentados conceitos básicos e fundamentos de Criptografia, e suas aplicações na área de Segurança da Informação. O capítulo 3 apresenta as ferramentas e tecnologias que foram utilizadas no desenvolvimento deste projeto e na implementação do INAFF. O capítulo 4 introduz os problemas na geração de uma página Web que forneça segurança aos dados transmitidos e recebidos de um Banco de Dados. O capítulo 5 apresenta o INAFF, produto final deste projeto, abordando todo o processo de seu desenvolvimento, e também uma discussão sobre seu uso e principais funcionalidades. No capítulo 6 encontram-se as conclusões obtidas ao final do estudo realizado. Finalmente, foram incluídos como anexos à esta dissertação um estudo de caso real, mostrando como incorporar o INAFF em uma página na *Web*, e um breve manual do usuário.

# Capítulo 2

# Conceitos de Criptografia

## 2.1 Introdução

O objetivo aqui não é, de maneira alguma, se aprofundar em criptografia, estudar técnicas de criptoanálise, muito menos destrinchar algoritmos de criptografia. Mas, por se tratar de um trabalho realizado no campo da Segurança da Informação, devem ser apresentados aqui alguns conceitos básicos sobre Criptografia, que é a base de qualquer sistema seguro.

Neste capítulo serão apresentados fundamentos em Criptografia, como criptografia de chaves simétricas, criptografia de chaves assimétricas, funções resumo ou funções hash e o funcionamento das assinaturas digitais.

### 2.1.1 Fluxo de informações

O fluxo normal de informação de uma fonte A para um destino B, mostrada na figura 2.1, sem solução de continuidade, e sem qualquer interferência, é a situação ideal que se pretende alcançar em uma comunicação eletrônica segura.

Segundo Stallings [STA 99] existem várias formas de ataques em que um agente malicioso inimigo I pode impedir, interferir ou detectar as informações transmitidas de A para B.

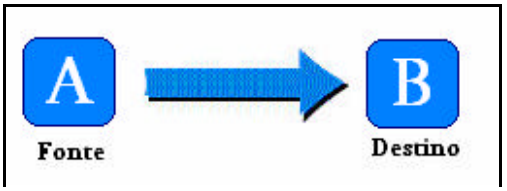



**Figura 2.1 – Fluxo normal de informação**

A seguir serão apresentadas as quatros formas de deturpação no fluxo de informações: interrupção, modificação, interceptação e fabricação.

### Interrupção

A informação parte da fonte A, porém não chega ao chega ao destino B, pois o fluxo é interrompido no meio do caminho, por algum motivo como, por exemplo, queda de energia ou problemas no meio físico de transmissão, conforme mostra a figura 2.2.

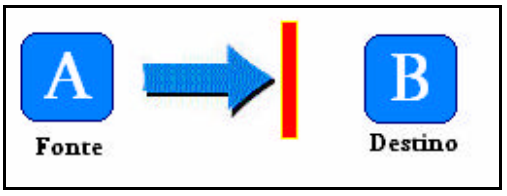



**Figura 2.2 – Esquema de uma interrupção no fluxo de informação**

### Modificação

O fluxo parte da fonte A, mas é interceptado por um inimigo I, que modifica a informação e envia para o destino B, conforme mostra a figura 2.3. Este tipo de ataque visa afetar a *integridade* dos dados.

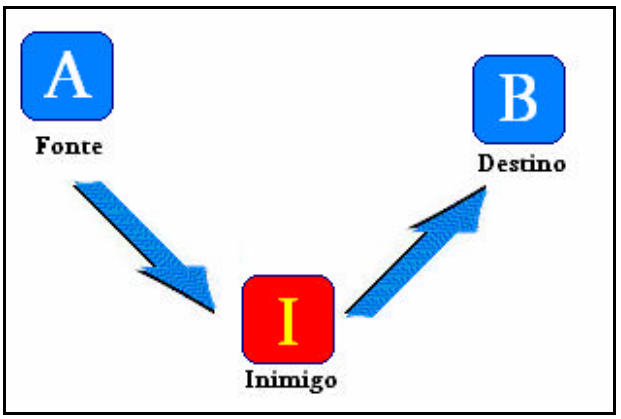



**Figura 2.3 – Esquema de modificação no fluxo de informação**

### Fabricação

O inimigo entra no circuito de comunicação e envia uma mensagem para o destino B, se passando pela fonte A, ou seja, a informação é fabricada por I, enviada para B, e este a recebe como sendo de autoria de A. A figura 2.4 mostra o esquema deste ataque, que afeta a *autenticidade* dos dados.

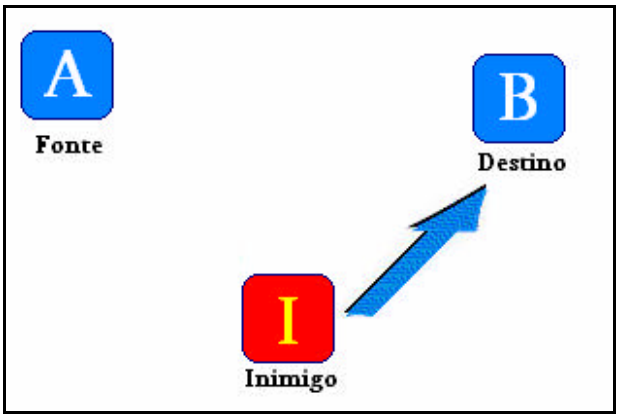



**Figura 2.4 – Esquema de fabricação no fluxo de informação**

### Interceptação

A informação gerada na fonte A chega normalmente até o destino B. O fluxo de informação aparentemente é normal, exceto pelo fato de haver um inimigo I monitorando o fluxo e que também conseguiu ter acesso à informação, sem que A ou B percebam. Este tipo de ataque afeta a *confidencialidade* dos dados, e é esquematizado na figura 2.5.

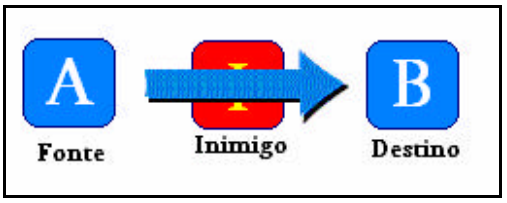



**Figura 2.5 – Esquema de uma interceptação no fluxo de informação**

### 2.1.2 Terminologia

Para que um sistema ofereça um fluxo seguro da informação a partir de A para B, ele deve atender principalmente os seguintes requisitos:

- **Confidencialidade** – ou sigilo, é a garantia de que somente as partes envolvidas na comunicação possam ler e utilizar as informações transmitidas eletronicamente pela rede. Isto é, caso um inimigo consiga interceptar de alguma forma o fluxo de informação, a mensagem transmitida deve ser indecifrável;
- **Integridade** – garantia de que o conteúdo de uma mensagem não será alterado durante seu tráfego, sem sofrer alterações, isto é, os dados que foram gerados em A são os dados que foram recebidos por B;
- **Autenticação** – garantia da identificação das partes envolvidas na comunicação, tendo certeza de que a comunicação esteja realmente sendo feita entre A e B, ou seja, que não há nenhum agente malicioso se fazendo passar por A para gerar uma informação falsa, ou se passando por B para simular a recepção da mensagem enviada por A;
- **Não repúdio** – garantia de que o emissor de uma mensagem não poderá, posteriormente, negar sua autoria.

## 2.2 Criptografia

[BUR03] "A Criptografia converte dados legíveis em algo sem sentido, ilegível. Estes dados poderão ser recuperados posteriormente a partir desses dados sem sentido".

Criptografia é uma palavra originada no grego, *kriptos* = escondido, oculto e *grifo* = grafia ou escrita, e é a arte ou ciência de escrever em cifras ou códigos. A criptografia é uma das ferramentas mais importantes para proteção dos dados, e a idéia principal é transformar um conjunto de dados legível em algo ilegível, garantindo que o segredo ali codificado permanecerá confidencial Apenas a pessoa que possuir a chave para

decifrar poderá reverter essa operação, ou seja, tornar esses dados ilegíveis em algo que faça sentido novamente.

Desde a Antigüidade, tão logo o homem aprendeu a escrever, ele começou a pensar em alguma forma de esconder o que ele havia escrito. Foi assim que começou a se desenvolver a Criptografia, base tecnológica para problemas de segurança em comunicações e em computação. Hoje em dia, esta ciência utiliza conceitos e fundamentos matemáticos para a construção de seus algoritmos. Um estudo sobre aritmética modular e teoria dos números primos é indispensável para quem deseja conhecer mais profundamente o assunto.

Porém, é fato que não existe alguém que diga que não precisa de segurança ou que não tenha algum segredo para ocultar. Todos nós temos informações que queremos manter em segredo, seja pelo simples desejo de privacidade ou até por autoproteção. Então, nenhuma pessoa ou empresa, quer ver seus dados sigilosos caindo em mãos erradas e maliciosas.

A Criptografia, de modo algum, é a única tecnologia necessária para assegurar os dados, nem resolverá todos os problemas relacionados à segurança. É apenas um instrumento, dentre outros existentes. Infelizmente, além disso, a Criptografia não é à prova de falhas, o que significa que por mais ininteligível que uma cifra seja, ainda assim existe a possibilidade dela ser quebrada.

### 2.2.1 Criptografia Simétrica

É a forma mais convencional de criptografia que existe. Por isso, também é conhecida por Criptografia Clássica.

Os algoritmos de criptografia de chaves simétricas são assim denominados por apresentarem como principal característica o fato de utilizarem apenas uma chave. Isto significa que a mesma chave que é utilizada para cifrar é também usada para decifrar. Os algoritmos para cifrar e decifrar também são os mesmos. O que muda é apenas a forma como são utilizadas as chaves. É tido como pré-requisito que os usuários envolvidos compartilhem esta chave entre si. A troca de chaves é geralmente feita através do

estabelecimento de um canal de comunicação seguro, através de tunelamento, entre as duas partes envolvidas.

Abaixo está representado graficamente o modelo simplificado da criptografia convencional


**Figura 2.6 – [ZIM98] Criptografia Simétrica: a mesma chave é usada para cifrar e decifrar.**

Existem vários algoritmos de criptografia simétrica. Porém os exemplares mais conhecidos e utilizados desta classe atualmente são:

- **AES** - *Advanced Encryption Standard*, um algoritmo concebido em outubro de 2000 para substituir o famoso, e cada vez mais vulnerável, algoritmo *Data Encryption Standard* (DES) e 3DES.
- **Blowfish**- cifrador de bloco simétrico desenvolvido por Bruce Schneier, em 1993, e permanece inquebrável até hoje. Possui as características de ser rápido, compacto, simples e possuir o tamanho de sua chave variando entre 32 e 448 bits. Opera com blocos de 64 bits.

O principal problema encontrado ao utilizar a criptografia simétrica é a questão do gerenciamento das chaves. Para cada par de usuários que desejam se comunicar é necessário uma chave, o que torna o gerenciamento muito complexo. Em uma rede com n usuários, serão necessárias n(n-1)/2 chaves para que todos os usuários possam trocar informações de maneira sigilosa entre si.

### 2.2.2 Criptografia Assimétrica

Este tipo de criptografia utiliza duas chaves diferentes: uma usada para cifrar o texto na origem e a outra usada para decifrar o texto quando este chegar no destino. Este sistema gera um par de chaves que são, na verdade, números primos muito grandes relacionados entre si, através de propriedades da aritmética modular.

Uma das chaves será chamada de chave privada e deverá ser mantida em sigilo. A outra chave será chamada de chave pública e deverá ser compartilhada com outras pessoas.

Se uma pessoa A cifra uma informação com sua chave privada, outra pessoa B somente poderá decifrar a mensagem com a chave pública de A, tendo a certeza de que esta mensagem realmente teve sua autoria em A, pois somente A possui a chave privada. Por outro lado, se A cifrar uma mensagem com a chave pública de B, somente B poderá decifrar a mensagem com sua chave privada, garantindo desta forma o sigilo ou confidencialidade da mensagem.

A grande vantagem da criptografia assimétrica é a eficiência no sigilo. Porém, como os algoritmos de criptografia assimétrica são um pouco lento quando, em sua entrada, apresentam volume muito grande de dados, para cifragem de mensagens grandes ela é ineficiente se for levado em conta o parâmetro tempo que tal processo leva. O que muito se faz é empregar a criptografia assimétrica para realizar a troca de chave de criptografia simétrica, de modo que esta última é cifrada com chaves assimétricas, podendo ser transmitida eletronicamente pela rede de maneira de segura.

Na página seguinte é apresentada uma figura que mostra o funcionamento da criptografia de chaves assimétricas.

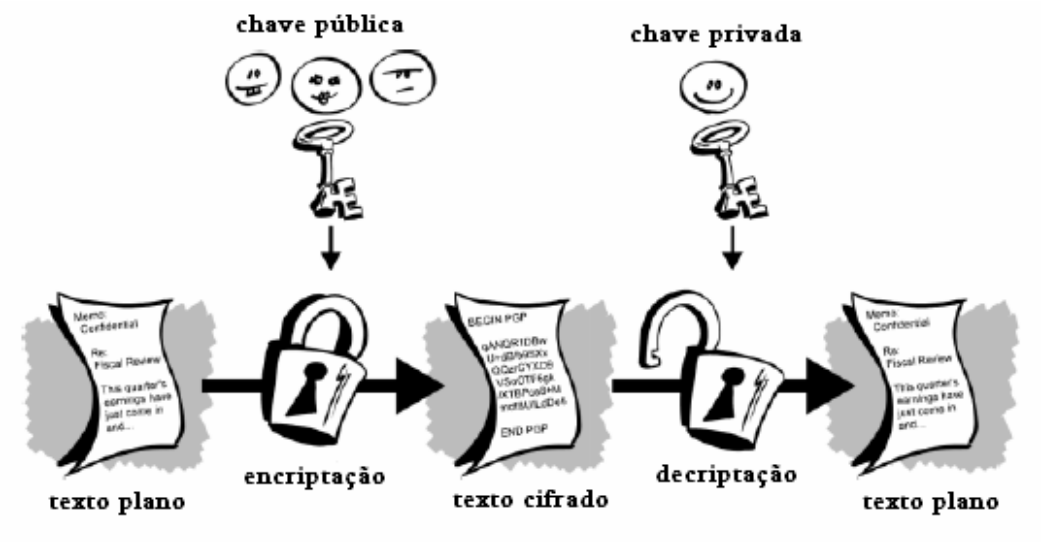



**Figura 2.7 – [ZIM98] Criptografia Assimétrica: o que uma chave cifra a outra decifra.**

## 2.3 Funções de Resumo

Como os algoritmos de criptografia assimétrica são lentos, não é uma boa idéia cifrar um grande volume de dados, como um texto inteiro.

Na prática, o que geralmente se faz é gerar um resumo dos dados e depois cifrar este resumo utilizado a chave privada do autor dos dados. Estes resumos são gerados através das chamadas funções de *hash*, e constituem uma representação da mensagem. Isto significa que, se o resumo possui essencialmente o mesmo valor de identidade da mensagem, é muito mais fácil trabalhar com um resumo criptográfico cujo tamanho é de alguns bytes (geralmente 16 ou 24 bytes), do que manipular mensagens demasiadamente grandes.

Tais funções são um algoritmo que recebe qualquer comprimento de entrada e mescla a entrada para produzir uma saída pseudo-aleatória de tamanho fixo.

A palavra "hash" pode significar desordem ou confusão, o que descreve eficientemente o resultado de uma mensagem resumida.

Uma propriedade verificável nestas funções é que elas são irreversíveis, isto é, não é possível reconstruir o conjunto original de dados a partir do resumo. Outra

particularidade que é importante saber é que até hoje não foram verificadas colisões – termo técnico empregado para descrever uma situação em que duas mensagens produzem um mesmo resumo. Elas existem, mas ninguém consegue encontrar uma colisão por demanda.

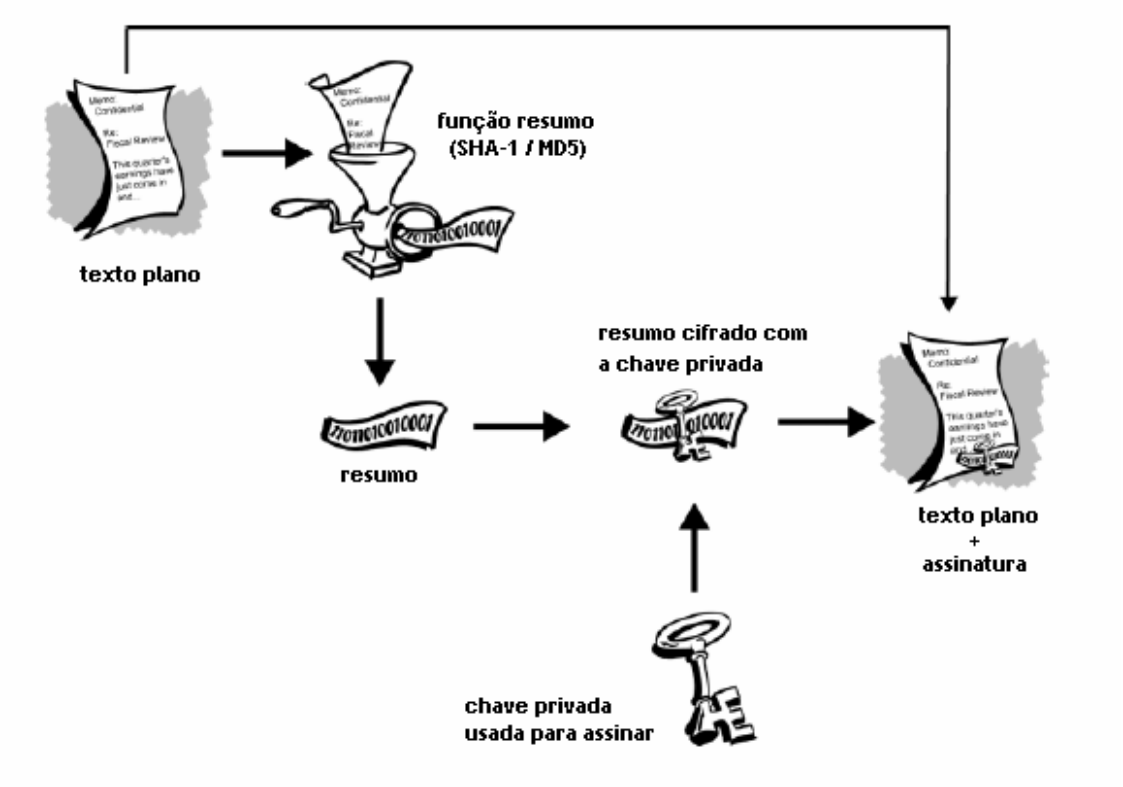



**Figura 2.8 – [ZIM98] Aplicação de funções de resumo para gerar assinaturas digitais seguras**

Os dois algoritmos mais importantes e conhecidos de resumo, dentre os vários existentes, são:

**MD5 - Message Digest 5**

Criado por Ron Rivest, do Massachussets Institute of Technology – MIT, é o sucessor mais bem-sucedido do MD2. Produz um resumo de 128 bits, porém é mais robusto e rápido que seu antecessor. Sabe-se de potenciais vulnerabilidades que podem ser exploradas e atualmente está o seu uso tem sido desencorajado por profissionais da área de

Criptografia, pois já foram encontradas colisões. Mesmo assim, por ter sido muito utilizado e ainda estar muito presente nos sistemas, vale a pena ser lembrado.

**SHA-1 - Secure Hash Algorithm**

Parecido com o MD5, porém mais forte. Sua saída produz um resumo de 160 bits, o que significa em uma probabilidade menor de ocorrência de colisões, pois apresenta um universo de $2^{160}$ resumos possíveis.

É altamente recomendado pela comunidade de criptografia, e é o resumo mais utilizado hoje em dia em certificados digitais e bibliotecas criptográficas.

A grande vantagem de utilizar resumos criptográficos é que mensagens muito grandes podem ser reduzidas a poucos bytes.

Com resumos é possível garantir a integridade dos dados, isto é, é possível saber se um conjunto de dados foi alterado a partir da comparação de resumos criptográficos.

Se você está preocupado com o fato de que as informações podem ser alteradas, envie seus dados juntamente com um resumo. Se os dados forem alterados, o resumo de verificação também será diferente, e você saberá que alguma coisa aconteceu.

Obviamente é preciso se assegurar de que o valor do resumo não pode ser alterado para corresponder com quaisquer alterações na mensagem. É neste ponto, portanto, que entra a assinatura digital, principal alicerce do nosso projeto, que consiste basicamente no procedimento de cifrar o resumo com a chave privada do assinante.

## 2.4 Assinatura Digital

Verificar a integridade dos dados muitas vezes não basta. É preciso também poder garantir a autoria destes.

Baseadas na aplicação de técnicas de criptografia assimétrica, as assinaturas digitais foram feitas para que uma entidade possa, digitalmente, "assinar" um documento eletrônico como, por exemplo, um arquivo texto ou um e-mail, comprovando de forma

única e exclusiva a autoria dos dados. Espera-se que uma assinatura digital possua as mesmas características de uma assinatura qualquer no mundo real.

Estas características desejáveis são:

1. A assinatura digital deve ser fácil de produzir por quem assina
2. Deve ser fácil de ser verificada por qualquer pessoa
3. Deve ser muito difícil de ser falsificada
4. Quem assine não possa negar que assinou (não-repúdio)

Além das quatro características citadas acima, é requisito fundamental que a assinatura digital dependa do conteúdo assinado. Isto é, a assinatura deve sempre ser diferente para diferentes informações assinadas. É nesta propriedade que se encontra o grande poder das assinaturas digitais. Senão, é possível imaginar como seria fácil de forjar a assinatura: bastaria o inimigo pegar a assinatura em um outro documento assinado e utilizá-la no documento o qual ele deseja falsificar.

É importante também saber que todo conjunto de dados cifrados com uma chave privada é uma assinatura digital, pois só pode ter sido gerado pela pessoa que possui a chave privada.

Forjar uma assinatura digital significa que alguém é capaz de criar uma massa de dados, por um outro meio, que fosse idêntica à assinatura. Isso significaria que o forjador quebrou o algoritmo RSA, o que é altamente improvável. É nisso que são baseadas as propriedades do não-repúdio e não-falsificação.

É até possível alegar que a chave privada foi roubada, o que tornaria possível uma outra pessoa gerar uma assinatura digital autêntica. Porém, existem meios de revogar um certificado digital (o que inclui invalidar a chave privada a ele associado) e também é possível proteger a chave privada.

As assinaturas digitais têm sido implementadas basicamente utilizando conceito de chaves públicas com o padrão RSA, para cifrar resumos gerados a partir de funções de hash como MD5 e SHA-1. Existem também o DSS - *Digital Signature Standard* – que é

um algoritmo muito utilizado para criação de assinaturas digitais e existe um esforço para torná-lo um padrão para tal finalidade.


**Figura 2.9 – [ZIM98] Esquema simplificado do funcionamento de uma assinatura digital.**

A chave privada é utilizada pelo assinante para assinar, enquanto a chave pública é utilizada para verificar a autenticidade da assinatura.

É muito importante esclarecer que uma assinatura digital não é feita para proteger uma mensagem contra um suposto inimigo. Ela tem como principal finalidade garantir que uma determinada pessoa, a autora da assinatura, realmente tenha assinado tal mensagem.

Todos que quiserem verificar a assinatura terão que ter acesso à assinatura e à mensagem. O inimigo, então, poderá verificar a assinatura. A idéia é evitar apenas que se falsifique a assinatura.

Para fins de confidencialidade e segurança das informações devem ser utilizadas técnicas de criptografia simétrica.

Uma maneira de criar assinaturas verificáveis é cifrar o resumo criptográfico com a chave privada RSA do assinante.

## 2.5 Certificados Digitais

Pergunta: Como alguém pode verdadeiramente saber se uma chave pública pertence a uma determinada pessoa em questão?

A maneira mais comum de verificar isso é por meio de um certificado digital, que associa um nome à uma chave pública. Uma analogia para compreender isso melhor é o passaporte, que associa uma foto, a um número e a um nome.

Um certificado digital é produzido de tal maneira que torna perceptível se um impostor pegou um certificado existente e substituiu a chave pública ou o nome ali contido. Qualquer pessoa ao examinar esse certificado fraudado saberá que algo está errado e, portanto, não confiará na combinação desse par de chaves e nome.

### 2.5.1 Funcionamento

Pegue um nome e uma chave pública, concatene esses dois dados e assine-os. Isto é um certificado digital. Geralmente a assinatura é feita por uma Autoridade Certificadora (AC), que é responsável por gerenciar as chaves públicas e os certificados digitais de terceiros.


**Figura 2.10 – Esquema de um certificado digital**

As ACs servem como terceiros confiáveis para associar uma identidade de uma pessoa a sua chave pública. A AC tem como principal tarefa a autenticação de seus usuários finais. Para isso, é preciso que a AC forneça sua própria chave pública para todos os usuários finais certificados, bem como para todas as partes verificadoras que podem utilizar as informações certificadas, pois para poder validar um certificado digital é necessário ter a chave pública da AC.

Os certificados digitais constituem um meio seguro de distribuir a chave pública para quem quiser verificar sua assinatura digital. O formato de certificado mais amplamente aceito é o X.509 v3 da *International Telecommunications Unions* (ITU-T), e apresenta alguns campos como: versão, número serial, identificador do algoritmo de assinatura, nome do emissor (AC), validade, nome do sujeito, chave pública, identificação do algoritmo de chave pública, etc.

### 2.5.2 Revogando um certificado digital

Dentre os dados que são armazenados dentro de um certificado digital está a data de validade. Uma assinatura digital gerada com uma chave privada expirada não será considerada válida no momento de sua verificação.

Os certificados são criados acreditando-se que estes serão válidos e usáveis durante todo o tempo de vida estimado no campo de Validade. Entretanto, em alguns casos, um certificado ainda em vigor poderá não ser mais utilizado.

Por exemplo, a chave privada pode ter sido comprometida de alguma forma. Caso o certificado digital de uma pessoa for extraviado, isto é, sua chave privada for roubada, ou ainda se for o caso de alguém perder o seu certificado digital, é possível entrar com um pedido de revogação do mesmo.

O método mais comum de se invalidar certificados é a *Certification Revocation List* – CRL – que é uma lista dos certificados revogados em cada Autoridade Certificadora.

Desta forma é possível constatar junto com a AC se o certificado é válido ou não, antes de gerar ou validar uma assinatura digital e verificar se o certificado digital se encontra ou não na CRL.

Similarmente, também é possível suspender por um tempo determinado a validade de um certificado digital.

## 2.6 Resumo

Neste capítulo foi apresentada uma série de conceitos da área de Segurança da Informação, tendo como principal objetivo abordar as técnicas mais utilizadas atualmente de Criptografia.

Tendo o foco do presente trabalho apresentado e, ainda, tendo em vista o conteúdo aqui exposto, acredita-se que para o problema de garantir a autenticidade dos dados que trafegam em uma aplicação Web o conceito de assinatura digital é fundamental para o sucesso da aplicação implementada.

Foram introduzidos fundamentos como confidencialidade, autenticação, integridade, não-repúdio, criptografia simétrica, criptografia assimétrica, assinaturas digitais, funções de hash e certificados digitais.

Dentre os tipos de ataques ou quebras do fluxo normal de informação durante uma comunicação, pode-se perceber que a solução que será apresentada neste documento impossibilita o “Ataque do homem do meio”, ou modificação, uma vez que mesmo que com os dados alterados, pode-se verificar sua autenticidade, encontrando qualquer problema.

# Capítulo 3

# Frameworks

## 3.1 Definição

De acordo com [GAM95], um *framework* pode ser definido como sendo um conjunto de classes que cooperam entre si e compõem um conjunto reutilizável para uma categoria específica de *software*. Um *framework* fornece direcionamento arquitetural do *software*, através do particionamento do projeto em classes abstratas e da definição de suas responsabilidades e colaborações. Um desenvolvedor customiza o *framework*, para uma aplicação particular, através da especialização e da composição de instâncias de classes do mesmo.

Outra definição amplamente aceita é a de [JOH93] que diz que "um *framework* é um conjunto de classes que incorpora um projeto abstrato para soluções destinadas a uma família de problemas relacionados".

Enfim, o grande objetivo de se construir um *framework* é promover a reusabilidade, de forma a obter ganhos significativos na diminuição do tempo de desenvolvimento.

As partes comuns de um *framework* são as partes estáveis, chamadas de *cold spots*. As partes flexíveis são chamadas de *hot spots*, podendo ser incompletas. O usuário de um *framework* irá estender as funcionalidades já implementadas através destas partes flexíveis, produzindo a aplicação completa.

## 3.2 Indivíduos envolvidos

Segundo [SIL00], existem dois tipos de indivíduos envolvidos no processo tradicional de desenvolvimento de software. O primeiro é aquele que desenvolve a aplicação, e o segundo é aquele que a utiliza.

Porém, ao se utilizar um *framework* surge um terceiro indivíduo: o desenvolvedor do *framework*, que faz com que o papel do desenvolvedor de aplicações se modifique, de forma que este passe a utilizar o *framework* para desenvolver suas aplicações.

É da responsabilidade do desenvolvedor de *framework*s também determinar, de algum modo, uma forma de ensino para utilizá-los na produção de aplicações.

## 3.3 Classificação dos frameworks

Independentemente do escopo, [FAY99] classifica os *framework*s pelas técnicas utilizadas para estendê-los, conforme segue:

- **Caixa-branca**: são extensíveis através da utilização de herança. Para estender *framework*s caixa-branca é necessário que os usuários do *framework* tenham um profundo conhecimento de sua estrutura interna;
- **Caixa-preta**: são extensíveis pela definição de interfaces para componentes que podem ser conectados no *framework* através da composição de objeto. Os objetos compostos não revelam detalhes internos de implementação, semelhantemente a uma "caixa-preta". Os *framework*s caixa-preta são geralmente mais fáceis de usar e estender do que os caixa-branca, porém são mais difíceis de serem desenvolvidos, já que o desenvolvedor do *framework* deve prever uma maior variação das funcionalidades a serem implementadas na aplicação que usa o *framework*;
- **Caixa-cinza**: é uma técnica híbrida que torna o *framework* estensível tanto através de herança de classe como pela composição de objetos.

Um *framework* caixa-branca é mais fácil de construir, mais flexível, porém mais difícil de usar. Já o *framework* caixa-preta é mais fácil de estender para uma nova aplicação, porém é mais difícil de construir. Por isso, a maioria dos *framework* vão provavelmente estar entre esses dois extremos, como um *framework* caixa-cinza.

## 3.4 Desenvolvendo um framework

Segundo [EXT03], um *framework* é concebido como qualquer outro projeto de software que enfoque uma solução para um determinado negócio. A principal diferença é que enquanto um software normal é resultado da necessidade de um negócio individual e específico, o *framework* é resultado de uma necessidade que pode ser observada repetidamente em vários outros projetos.

Isto quer dizer que o objetivo principal ao se projetar um *framework* é que este seja utilizado mais de uma vez, em situações diferentes, permitindo que futuras aplicações sejam desenvolvidas muito mais facilmente.

É por isso que o *framework* deve apresentar flexibilidade, extensibilidade e facilidade para configuração.

Muitas vezes acontecem situações onde o usuário emprega o *framework* para uma determinada tarefa a qual o desenvolvedor do *framework* não tinha pensado antes.

O *framework* visa explorar as implementações que são necessárias em vários e diferentes sistemas. Quando isso ocorre há a conseqüência do surgimento de um padrão, que deve ser identificado como uma excelente oportunidade para o reuso.

Identificar pontos reusáveis é fundamental para a criação do processo de desenvolvimento da solução que poderá ser reutilizada, que neste caso é o *framework*.

## 3.5 Vantagens do uso de frameworks

As principais vantagens da utilização de *framework*s são [FAY99]:

- **Modularidade**: através do encapsulamento dos detalhes de implementação e de interfaces estáveis, os *framework*s possibilitam maior modularidade, melhorando a qualidade do produto final;
- **Reusabilidade**: as interfaces estáveis providas pelos *framework*s facilitam a reutilização de componentes genéricos em novas aplicações;

- **Capacidade de extensão**: *framework*s aumentam a capacidade de extensão pelo fornecimento explícito de métodos *hook*, que permitem às aplicações estenderem suas interfaces estáveis;
- **Inversão de controle**: possibilita implementar o fluxo de controle da aplicação. Segundo o princípio de Hollywood, que diz *"Don't call us, we'll call you"*, é o *framework* que deve chamar a aplicação, e não o contrário.

## 3.6 Pontos fracos dos frameworks

Para obter o sucesso prometido com a utilização de *framework*s, é preciso reconhecer e resolver alguns obstáculos como [FAY99]:

- **Esforço de desenvolvimento**: desenvolver *framework*s de alta qualidade, extensíveis e reutilizáveis para aplicações complexas exige muito esforço, experiência e trabalho dos desenvolvedores;
- **Curva de aprendizado**: o esforço demandado dos usuários de um *framework* é considerável. São necessários documentações e exemplos de aplicações implementadas. Em alguns casos, é preciso ainda cursos e treinamentos para ensinar os usuários do *framework* como utilizá-lo eficientemente;
- **Integrabilidade**: é preciso ainda levar em conta a integração de vários *framework*s, bibliotecas e componentes no desenvolvimento de uma aplicação;
- **Manutenibilidade**: é difícil manter a compatibilidade entre os *framework*s e as aplicações sobre ele desenvolvidas, pois ambos deverão evoluir juntamente ao longo do tempo;
- **Validação e remoção de erros**: componentes genéricos são mais difíceis de validar.
- **Eficiência**: A generalidade e flexibilidade providas pelos *framework*s reduzem a eficiência;

- **Falta de padronização**: atualmente, não existe padronização amplamente aceita para projeto, implementação, documentação a adaptação de frameworks.

## 3.7 Resumo

Apesar do projeto e implementação de um *framework* ser geralmente cara, pode-se reduzir este custo no desenvolvimento de várias aplicações, através da reusabilidade existente com os frameworks.

Um framework deve incorporar as abstrações de um domínio de problemas relacionados, e é sempre resultado de uma analise de exemplos concretos deste domínio.

Os frameworks devem ser suficientemente simples para que possam ser aprendidos e, ao mesmo tempo, devem implementar funcionalidades que tornem o desenvolvimento de aplicações mais rápido, e pontos adaptáveis para que possam ser extensíveis.

# Capítulo 4

# Ferramentas utilizadas

## 4.1 Introdução

Este capítulo apresenta as ferramentas e tecnologias que foram utilizadas para o desenvolvimento deste projeto, incluindo linguagens de programação, softwares, servidores Web e Banco de Dados. Foi feito um estudo detalhado em cima das principais tecnologias e ferramentas empregadas, abrangendo tópicos como história, vantagens e desvantagens, comparações com outras tecnologias do mesmo âmbito e seu estado da arte.

## 4.2 Apache

Toda aplicação que se encontra na *Web* é executada em cima de um servidor http. Como o INAFF foi desenvolvido utilizando PHP, é necessário rodá-lo em um servidor http que ofereça suporte para PHP. E finalmente, para simular o uso real de nossa ferramenta foi preciso rodar um servidor HTTP para que pudéssemos acessar e simular a execução das páginas HTML do nosso exemplo.

O servidor escolhido foi o Apache, principalmente por ser um servidor gratuito e multiplataforma. Diferentemente do Microsoft IIS - *Internet Information Services* - o Apache possui versões para inúmeras plataformas, dentre as quais se encontram o Linux e o Microsoft Windows.

Neste projeto, por se utilizar a biblioteca CAPICOM, da Microsoft, o servidor Apache foi instalado somente na plataforma Windows. Porém, para sustentar uma proposta de extensão deste projeto, existe a intenção de transformá-lo em um *framework* multiplataforma, optamos pelo Apache, e também o instalamos no Linux, para podermos saber como o servidor se comporta nas duas plataformas.

O servidor HTTP Apache é resultado de um projeto cujos objetivos eram criar um servidor que fosse robusto, de nível comercial, com diversas funcionalidades e com o

código fonte aberto e livre. Este projeto é conjuntamente gerenciado por um grupo de voluntários ao redor do mundo, que usam a Internet e a Web para se comunicarem, planejar, e desenvolver o servidor e sua documentação. Este grupo de voluntários é conhecido como Grupo Apache. Além disso, centenas de usuários têm contribuído com idéias, códigos e documentação para o projeto [APA04].

Em fevereiro de 1995, o servidor mais popular na Web era o de domínio público HTTP daemon (httpd) desenvolvido por Rob McCool, no NCSA - National Center for Supercomputing Applications - da Universidade de Illinois, nos Estados Unidos. Entretanto, o desenvolvimento do HTTPd estagnou com a saída de Rob McCool do NCSA, em meados de 1994, e muitos webmasters escreveram suas próprias extensões e correções para o servidor. Um pequeno grupo destes webmasters, via e-mail, se reuniram com o propósito de organizar as suas alterações, em forma de *patches*. Oito membros deste grupo inicial fundaram o Apache Group (Grupo Apache), que tem como principal função gerenciar os projetos e trabalhos que estão sendo desenvolvidos pelos voluntários espalhos pelo mundo todo.

Finalmente, desde abril de 1996 o servidor HTTP Apache tem sido o mais popular servidor Web na Internet. Em outubro de 2003, a Netcraft Web Server Survey divulgou uma pesquisa onde mais de 64% do *sites* da Web na Internet estão rodando em cima de um servidor Apache. Por ser um projeto que conta com a ajuda de desenvolvedores de todo o mundo, está sendo constantemente atualizado e as falhas são corrigidas tão logo estas são descobertas, o que torna o objetivo do projeto Apache de criar um servidor seguro, eficiente e extensível uma realidade.

## 4.3 PHP

Alguns dizem que PHP significa "Personal Home Page", ou "Home Page Pessoal". Mas, segundo o *site* oficial - http://www.php.net/ - PHP é um acrônimo recursivo para PHP: Hypertext Preprocessor, e é uma linguagem interpretada, escrita na forma de *scripts*, muito utilizada hoje em dia para criar páginas dinâmicas na Web. Diferentemente de outras linguagens de script, como Perl, o código em PHP pode ser incorporado junto ao código HTML.

```
1  <HTML>
2    <HEAD>
3      <TITLE>Exemplo do uso do PHP</TITLE>
4    </HEAD>
5    <BODY>
6
7      <?php
8
9        echo "Olá, este é um script escrito em PHP!";
10
11     ?>
12
13   </BODY>
14 </HTML>
```

**Figura 3.1 – Programa escrito em PHP embutido em uma página HTML**

PHP é extremamente simples para os iniciantes, porém oferece muitas funcionalidades avançadas para os programadores profissionais. Uma vez que PHP roda do lado do servidor, você pode usufruir de qualquer função de qualquer programa em CGI como, por exemplo, coletar informações vindas de um formulário na Web, gerar conteúdo de páginas dinamicamente, enviar ou receber *cookies*, gerar arquivos em PDF, gráficos, e muito mais.

O PHP te oferece a liberdade de escolher o sistema operacional e servidor Web que será utilizado, pois o PHP pode ser usado na maioria dos sistemas operacionais existentes hoje em dia (Microsoft Windows, Linux, Unix, MAC OS X, etc) e também é suportado pela maioria dos servidores HTTP (Web Servers), como Apache e Microsoft IIS.

Quanto ao paradigma de programação, com PHP você pode programar tanto de maneira estruturada / procedural como também pode programar orientado a objetos ou, ainda, mesclar ambos estilos.

Finalmente, uma das mais fortes e significantes características do PHP é o fato deste oferecer suporte para uma enorme variedade de Banco de Dados. Escrever uma página Web que acesse uma base de dados, em PHP, é incrivelmente simples.

Além das vantagens citadas acima como, por exemplo, ser multiplataforma, compatível com os principais Bancos de Dados do mercado, fácil de usar, outro motivo que pesou muito na escolha do PHP para ser usado na implementação deste projeto foi o fato de

termos uma certa intimidade com PHP, sendo que programamos com esta linguagem desde 2000 e já tivemos experiências profissionais com a mesma.

## 4.4 CAPICOM

CAPICOM é um ActiveX da Microsoft que provê uma interface COM - Component Object Model - com a biblioteca de criptografia CryptoAPI, também da Microsoft. Ela faz a interface com algumas funções da CryptoAPI que permitem que os desenvolvedores adicionem facilmente os recursos de assinatura digital e funcionalidades de cifragem de dados em suas aplicações.

Por usar COM, os programadores podem acessar essas funcionalidades em um grande número de ambientes de programação como, por exemplo, Microsoft Visual Basic, VBScript, JavaScript, ASP, Microsoft JScript, C++, Delphi e outros.

Também por ser um pacote ActiveX, a CAPICOM permite que os desenvolvedores escrevam aplicações para Web.

A CAPICOM pode ser usada para executar as seguintes tarefas:

1. Criar assinaturas digitais em dados com um smart card ou uma chave;
2. Verificar assinaturas digitais;
3. Exibir graficamente informações de certificados digitais;
4. Obter informações e propriedades de um certificado digital como, por exemplo, nome do sujeito e data de expiração;
5. Incluir e excluir um certificado digital do repositório de certificados;
6. Cifrar e decifrar dados com uma senha (Criptografia simétrica);
7. Cifrar e decifrar dados utilizando chaves públicas e certificados digitais (Criptografia assimétrica).

Infelizmente, a CAPICOM somente é suportada pela plataforma Windows (95, 98, ME, 2000, XP, 2003), com navegador *Internet Explorer* versão 5 ou superior.

Para uma aplicação que usa a CAPICOM funcionar corretamente o usuário terá que obrigatoriamente tê-la instalada na máquina. É possível fazer o download gratuitamente da biblioteca CAPICOM diretamente do *site* do MSDN.

O trecho de código a seguir mostra como disponibilizar o download da CAPICOM em aplicações Web:

```
1  <html>
2  <head>
3          <title>Exemplo INAFF</title>
4
5  <OBJECT
6    ID="oCapicom"
7    classid="clsid:E38FD381-6404-4041-B5E9-B2739258941F"
8    codebase="http://www.inf.ufsc.br/~bill/capicom.cab#version=2,0,0,0">
9  </OBJECT>
10
11 </head>
12
13 <body>
14
15
16
17 </body>
18 </html>
```

**Figura 4.1 - Referenciando objeto da CAPICOM em uma página HTML**

No parâmetro *codebase* da tag **object** é possível determinar uma URL para que, caso a CAPICOM não estiver instalada na máquina do cliente, seja possível fazer o download do arquivo da biblioteca.

Não faz parte dos objetivos deste trabalho explicar detalhadamente as funções, métodos e propriedades dos objetos da CAPICOM. Portanto, a seguir serão brevemente comentados os objetos que foram utilizados na implementação do INAFF.

Para obter acesso à documentação detalhada e maiores informações deve-se visitar a referência da CAPICOM na Web, disponível no *site* http://msdn.microsoft.com/library/en-us/security/Security/capicom_reference.asp

### 4.4.1 Objeto *Signed Data*

O objeto *SignedData* oferece propriedades e métodos para estabelecer o conteúdo que será assinado com uma assinatura digital, para assinar e co-assinar dados digitalmente e, também, verificar a assinatura digital de um conjunto de dados assinado. A assinatura será gerada no formato PKCS #7.

Conforme visto na sessão de conceitos básicos, uma assinatura, quando verificada, provê a associação entre um indivíduo assinante e os dados, e comprova que estes não foram alterados de maneira alguma depois que a assinatura foi criada.

#### Propriedades

***SignedData.Content***

Tipo: string
Acesso: escrita e leitura

Esta propriedade armazena os dados que serão assinados, e deve obrigatoriamente ser definida antes de chamar o método Sign.

É a propriedade padrão do objeto (Em Visual Basic todo objeto possui uma propriedade padrão, de maneira que é possível invocar esta propriedade simplesmente omitindo o seu nome na instrução. Por exemplo, ObjetoSignedData = "UFSC" é o mesmo que digitar ObjetoSignedData.Content = "UFSC").

#### Métodos

***SignedData.Sign***

Cria uma assinatura digital em cima do conteúdo a ser assinado.

Conforme foi escrito acima, este método só pode ser invocado após a propriedade Content ter sido inicializada. Então, é gerada a assinatura digital, que nada

mais é que o resumo (hash) dos dados contido em *Content* cifrados com a chave privada do assinante.

A assinatura é gerada usando o algoritmo de resumo SHA-1.

Os parâmetros para essa função são todos opcionais. Porém é possível, e recomendável, passar como parâmetro um objeto do tipo Signer, que contem o certificado digital do assinante, caso contrário o método pode não funcionar conforme esperado.

A função Sign retorna uma string contendo a assinatura digital.

IMPORTANTE: Este método utiliza a chave privada do assinante. Isto implica em um nível de risco para a segurança. Uma caixa de diálogo que pergunta se o Web *site* pode utilizar sua chave privada aparece quando esse método é chamado. É muito importante para a segurança do usuário que este nunca desabilite a opção de exibir sempre essa caixa de diálogo, caso contrário o usuário corre o risco de outros *sites* na Web utilizarem sua chave privada para criar assinaturas digitais em nome dele.

Importante também frisar que o presente trabalho parte do axioma que indica total confiabilidade na plataforma utilizada pelo usuário, isto é, uma vez que o código adicionado pelo INAFF é colocado em execução diretamente na máquina cliente, todos os componentes ali instalados garantem sua segurança. A geração do próprio certificado não é, de maneira alguma, tratada pela aplicação criada e apresentada neste documento. Os níveis de segurança para utilização de navegadores de rede, gerenciamento de certificados digitais é definido apenas pelo usuário, e cabe a ele garantir que seu sistema esteja inviolável durante um processo de envio de dados.

***SignedData.Verify***

Determina se uma assinatura é válida ou não.

Para verificar uma assinatura, o resumo cifrado dos dados é decifrado usando a chave pública obtida no certificado digital do assinante. O resumo decifrado é comparado com um novo resumo dos dados contidos na propriedade *Content* do objeto *SignedData*.

Isto depende se no momento da geração da assinatura o modo utilizado foi *bDetached* ou não. Então, a assinatura é dita válida se os resumos forem idênticos.

Além disso, também é possível verificar a validade do certificado utilizado para criar a assinatura.

Este método recebe como parâmetro obrigatório uma string contendo a assinatura que deve ser verificada.

O método *Verify* não retorna nenhum valor.

### 4.4.2 Objeto *Store*

Objeto usado para escolher, gerenciar e usar os repositórios de certificados (na máquina do usuário) e os certificados neles contidos.

A propriedade *Certificates* é a padrão, e é utilizada para buscar os certificados contidos no repositório.

### 4.4.3 Objeto *Certificate*

Este objeto é utilizado para representar um certificado digital, e é geralmente utilizado quando se deseja:

- Recuperar dados do certificado, incluindo a chave privada que é associada a ele (apenas lembrando que o certificado digital não contem a chave privada. O que o objeto *Certificate* faz é permitir que através de um certificado seja localizada sua chave privada no Windows);
- Obter informações básicas do certificado;
- Exibir o certificado;
- Importar e exportar o certificado;
- Determinar a validade de um certificado.

Ele é usado para inicializar a propriedade *Certificate* do objeto *Signer*. Sem um certificado, não é possível gerar assinaturas digitais.

### 4.4.4 Objeto *Certificates*

Coleção de objetos do tipo *Certificate*, sendo que cada um deste representa um certificado digital.

É utilizado para buscar um certificado específico dentro de uma coleção.

O método Certificates.Select exibe uma caixa de diálogo para selecionar certificados e retorna uma coleção dos certificados selecionados (*Certificates*).

### 4.4.5 Objeto *Signer*

O objeto *Signer* estabelece o assinante para o objeto *SignedData*, inicializando a propriedade *Signer* deste último.

A propriedade padrão do objeto Signer é Certificate, que corresponde ao certificado digital do assinante e que deve conter uma chave privada disponível, de modo que esta possa ser utilizada para assinar dados.

## 4.5 Banco de Dados

### 4.5.1 O que são?

Para armazenarmos dados de um sistema qualquer poderíamos utilizar um simples sistema de arquivos que guarda os dados que são importantes e que devem poder ser consultados ou alterados em um momento futuro. Porém, a utilização de arquivos texto para este propósito pode trazer uma série de problemas como [SIL99]:

- Inconsistência e redundância de dados;
- Dificuldade no acesso aos dados;
- Isolamento de dados – múltiplos arquivos e formatos;
- Problemas de integridade;
- Atomicidade de atualizações;
- Acesso concorrente por múltiplos usuários;
- Problemas de segurança.

Para resolver estes problemas é necessário utilizar-se de sistemas de gerenciamento de banco de dados, ou SGBDs, que nada mais são do que um conjunto de dados associados a um conjunto de programas e ferramentas para acesso a estes dados. Eles são projetados para gerenciar de forma conveniente e eficiente grandes volumes de dados, possibilitando a estruturação das informações a serem armazenadas, assim como sua segurança.

Os SGBDs executam suas funções respeitando um modelo de três níveis: nível físico, nível lógico e nível de visão, este último seria a interface com o usuário, abstraindo, assim, a necessidade deste conhecer detalhes de implementação, armazenamento e inter-relacionamento dos dados.

### 4.5.2 A Linguagem SQL

A SQL – *Structured Query Language* – é a linguagem padrão mais comum usada para acessar banco de dados, e é definida pelo padrão ANSI/ISO SQL. O padrão SQL é uma linguagem de alto nível para manipulação de dados dentro do modelo relacional e se tornou o mecanismo mais popular de acesso aos grandes bancos de dados cliente/servidor.

Quando o primeiro SGBD relacional foi desenvolvido, no início da década de 70, o primeiro mecanismo de acesso àqueles dados foi uma forma primitiva de SQL. A partir dali o Instituo Americano de Padrões – ANSI – padronizou as implementações da

linguagem. As pequenas variações da linguagem não afetam o padrão global e costumam ser incorporadas para complementar as capacidades da linguagem.

Ainda que tais variações possam aumentar a complexidade da migração de um ambiente para outro, elas não afetam a estratégia global de portabilidade entre plataformas. Se for o caso o desejo de se desenvolver aplicações portáveis, mesmo que se perca em desempenho e facilidade, o padrão ANSI deve ser seguido rigorosamente, pois é suportado por todos os produtos, de uma maneira geral.

### 4.5.3 MySQL

O MySQL é um dos Sistemas de Gerenciamento de Banco de Dados – SGBD – disponíveis, que são da família SQL. É um SGBD relacional que tem como principal característica a rapidez em executar as transações e consultas. Pode ser utilizado inclusive em sistemas críticos, onde são armazenados muitos dados (da ordem de milhões de registros).

Este SGBD possui dois tipos de licença de uso:

- *Open Source* – sob os termos da GPL – *General Public License* - que o categoriza como sendo um software livre;
- Comercial – para compra, geralmente utilizada para quem deseja embutir o MySQL em seu *software* e vendê-lo.

Atualmente, o MySQL é um dos mais populares SGBDs SQL. Tal popularidade se deve ao fato dele ser extremamente rápido para fazer consultas, confiável e fácil de usar.

O banco MySQL é um sistema cliente/servidor que consiste de um servidor *multithread* que suporta diversos acessos simultâneos provenientes de diversos programas clientes. O sistema também possui ferramentas administrativas, bibliotecas e API para programação.

Como é um software *Open Source*, atualmente existe no mundo centenas de pessoas trabalhando para melhorar o código do MySQL e desenvolvendo um conjunto de recursos muito práticos.

Suas principais características são:

- Performance;
- Velocidade das consultas;
- Conectividade;
- Segurança.

## 4.6 phpMyAdmin

O uso do phpMyAdmin na implementação do INAFF foi importante principalmente na fase de testes, quando foi necessário o uso de um banco de dados para fazer nosso estudo de caso real do *framework*.

O phpMyAdmin é uma ferramenta escrita em PHP para manipular o gerenciamento de um banco de dados MySQL através da Web.

Atualmente é possível criar e excluir bases de dados, criar/excluir/alterar tabelas, executar qualquer comando SQL, gerenciar chaves, privilégios, além de poder exportar dados para diversos formatos.

O software está disponível em 47 idiomas, incluindo o Português. A licença de uso é de acordo com a GPL - *General Public License* - o que o caracteriza como sendo um software livre.

Até a data de conclusão deste trabalho, o phpMyAdmin estava em sua versão 2.6.0-alpha3, podendo ser obtido através da seção de downloads do *site* oficial do projeto phpMyAdmin - http://www.phpmyadmin.net/ - onde também é possível encontrar a documentação completa da ferramenta e alguns tutoriais.

## 4.7 Editores de Texto

### 4.7.1 Bloco de Notas

Todo mundo alguma vez já usou o bloco de notas. É o editor de textos mais simples que tem no ambiente Windows.

Foi muito utilizado para editar e visualizar rapidamente códigos fonte de páginas HTML e PHP.

### 4.7.2 PhpED

PhpED é uma IDE - *Integrated Development Environment* ou, traduzindo, Ambiente de Desenvolvimento Integrado - para desenvolver programas em PHP. Possui recursos muito úteis para facilitar nas tarefas de projeto, criação, desenvolvimento, depuração e testes.

Todo o código fonte do INAFF foi feito utilizando esta ferramenta.

## 4.8 ERWin

O ERwin é uma solução para modelagem de dados líder de mercado. Permite criar e manter modelos relacionais de bancos de dados.

Os modelos do ERwin visualizam estruturas de dados de forma a ajudá-lo a organizar, gerenciar e mesmo aliviar a complexidade de dados, de tecnologias de banco de dados e do ambiente de distribuição.

Os bancos de dados não somente são desenvolvidos mais rapidamente, como também sua qualidade e capacidade de manutenção são melhoradas significativamente. O ERwin gera automaticamente tabelas, *stored procedures* e *triggers* para os diversos bancos de dados existentes.

Sua tecnologia *complete-compare* permite o desenvolvimento interativo de forma que seu modelo está sempre sincronizado com o banco de dados.

## 4.9 Microsoft Paintbrush

É o editor gráfico de imagens mais popular do mundo. Tal condição deve se ao fato de o Paintbrush sempre ter feito parte do pacote de aplicativos que vem com a distribuição do Microsoft Windows.

Foi utilizado para criar e editar as figuras que estão presentes nesta monografia.

# Capítulo 5

# Implementação do INAFF

## 5.1 Por que INAFF?

Primeiramente, você deve estar curioso para saber de onde veio esse nome, INAFF, não é?

Pois bem, quando optamos por dar um nome fantasia ao produto final do nosso trabalho, passamos a procurar algum nome que sugerisse o que realmente o *framework* é. Então, veio a idéia de dar o nome de INAFF, que é mais ou menos como se pronuncia a palavra *"enough"* que, em inglês, significa "suficiente"

Então, o que sugerimos com esse nome é que o INAFF é um *framework* que gera o mínimo de código necessário e **suficiente** para se ter implementado um mecanismo que gera uma assinatura digital em cima de dados provenientes de um formulário na Web e, mais tarde, verifica a validade de uma assinatura comparando com uma assinatura proveniente de uma base de dados.

## 5.2 Projeto do INAFF

Quando decidimos construir um *framework* para dar suporte à base de dados seguras na Web, sabíamos que a primeira etapa seria levantar quais são os pontos em comum que todas as aplicações nesse escopo apresentam em comum. É assim que começa a nascer um *framework*. A primeira pergunta a ser feita é "Quais são os padrões que essas aplicações têm em comum?".

Como resposta à pergunta feita acima, podemos concluir que todas as aplicações que implementam um mecanismo de segurança que se baseia em criar uma assinatura digital em cima dos dados provenientes de um formulário de uma página da Web seguem, basicamente, os seguintes passos:

1. Pegam as informações passadas para o formulário;
2. Concatena esses dados;
3. Solicita o certificado digital para o usuário;
4. Assina digitalmente os dados;
5. Armazena a assinatura gerada em uma variável;
6. Armazenam os dados e a assinatura em um Banco de Dados.

E no caminho de volta:

1. Pegam os dados e assinatura do Banco de Dados;
2. Decifra a assinatura com a chave pública do assinante, resgatada a partir do certificado digital do mesmo;
3. Compara o resumo criptográfico obtido no passo anterior, e gera um novo resumo (ou *hash*) em cima dos dados que desejamos verificar a autencidade;
4. Tenta verificar ou validar a assinatura através da comparação dos resumos criptográficos;
5. Exibe algum tipo de resposta para o usuário.

Esses passos são feitos repetidamente em vários sistemas diferentes. O projeto do INAFF teve como foco principal explorar esses pontos para que pudesse ser implementado o *framework*.

Os usuários do INAFF não terão que se preocupar mais em ir atrás de concatenar dados, em estudar profundamente os conceitos de Criptografia, destrinchar as funções da CAPICOM, e outras tarefas que, geralmente, ocupariam seu tempo precioso.

Ao se usar o INAFF, o *framework* cuidará da geração do código necessário para fazer os passos descritos anteriormente de maneira automática. Além disso, o aprendizado e utilização do INAFF podem ser feitos de maneira muito fácil. O requisito necessário é que o usuário do *framework* tenha conhecimentos da linguagem de programação PHP.

## 5.3 Métodos

A seguir serão mostrados os principais métodos que foram implementados, abordando principalmente uma descrição detalhada de cada um e apresentando também trechos do código-fonte.

### 5.3.1 CarregaConf

Este método deve ser chamado logo após a inclusão do INAFF em um arquivo PHP. Isto é, somente depois que o usuário adicionou a seguinte linha dentro do seu script PHP será possível chamar o método CarregaConf, conforme é mostrado logo abaixo:

```
include("path/to/INAFF.php");
CarregaConf("idPagina");
```

**Código-fonte**

```
---------------------------------------8<-------------------------------------------------
function carregaConf($nomePagina) {
  global $parametros, $configFile;
  $linhas = file($configFile);
  $linhaAtual = "";
  $achou = false;

  for ($i=0; $i<count($linhas); $i++)
  {
    $linhaAtual = $linhas[$i];
    if (($linhaAtual[0] != "#") && (trim($linhaAtual)!="")) {
        $parametros = explode(',', $linhaAtual);
        if ( $parametros[0] == $nomePagina ) {
          $achou = true;
          break;//sai do for
        }
    }
  }
  if (!$achou) {
    die ("$fontErro PÁGINA NÃO ENCONTRADA NO ARQUIVO DE CONFIGURAÇÕES.
CANCELANDO...</FONT>");
  }
}
---------------------------------------->8-------------------------------------------------
```

Esse método abre o arquivo de configurações do INAFF em modo de leitura e faz uma varredura completa no arquivo, obtendo linha por linha do mesmo. Cada linha do arquivo que não estiver em branco ou não se iniciar com o caractere '#', que indica um comentário, será processada e analisada.

Os parâmetros de configurações são separados por vírgula. É feita então uma divisão desses parâmetros, e eles são jogados em uma variável global $parametros, para que possam ser utilizados posteriormente em outros métodos do *framework*.

O primeiro parâmetro é o identificador de página, que serve como chave para o acesso aos outros parâmetros. Este procedimento foi adotado como solução para que páginas diferentes pudessem compartilhar do mesmo *framework* instalado no servidor.

Então, o que o método **CarregaConf** basicamente faz é ler cada linha do arquivo até encontrar uma em que o primeiro parâmetro seja igual ao que foi passado como argumento para este método. Quando for encontrada a ocorrência, o método "explode" a linha, obtendo todo os parâmetros que ali estão, separados por vírgulas, jogando-os em um vetor de parâmetros.

Se for passado como argumento um identificador que não existe no arquivo de configurações, então será exibida uma mensagem de erro e a execução do script PHP será abortada.

OBSERVAÇÃO: se acontecer um caso onde existam duas linhas indexadas com o mesmo identificador, prevalecerão os parâmetros da primeira ocorrência.

### 5.3.2 MontaFuncaoAssina

Essa função deve ser utilizada na página que contem o formulário para entrada dos dados, e tem como principal função gerar automaticamente o código em VBScript para criar uma assinatura digital e escrevê-lo juntamente com o código HTML.

Para que ela seja invocada é preciso que o INAFF tenha sido antes incluído na página, através do **include( )** e que a função **CarregaConf()** tenha sido chamada corretamente com o devido identificador da página, passado como parâmetro.

## Código fonte

```
-------------------------------------8<-------------------------------------------------
function montaFuncaoAssina() {
  global $parametros, $strEchoAssina, $nomeFormulario;
  $nomeFormulario = $parametros[1];
  $strEchoAssina = "dadosParaAssinar = ";

  for ($i=2; $i<count($parametros); $i++)
  {
    $campo = explode('=',$parametros[$i]);
    $strEchoAssina = $strEchoAssina."document.".$nomeFormulario.".".$campo[0].".value";

    if ($i == (count($parametros) - 1)) {
      $strEchoAssina = $strEchoAssina."\n";
    } else {
      $strEchoAssina = $strEchoAssina." & ";
    }
  }

    // Escrevendo na página HTML
    echo "

    <SCRIPT LANGUAGE=\"VBSCRIPT\">
    function Assina()
      on error resume next

      $strEchoAssina

      Set oStore = CreateObject(\"CAPICOM.Store\")
      Set oCertificate = CreateObject(\"CAPICOM.Certificate\")
      Set oCertificates = CreateObject(\"CAPICOM.Certificates\")
      Set oCertificates1 = CreateObject( \"CAPICOM.Certificates\")
      oStore.open
      for each oCertificate in oStore.certificates
        oCertificates.add oCertificate
      next
      set oCertificates1 = oCertificates.select(\"Selecione um certificado
digital\",\"\",false)
      Set oSigner = CreateObject(\"CAPICOM.Signer\")
      oSigner.certificate = oCertificates1(1)

      Set oAssinador = CreateObject(\"CAPICOM.SignedData\")
      oAssinador.Content = dadosParaAssinar
      Assinatura = oAssinador.Sign(oSigner,true)
```

```
      document.$nomeFormulario.assinatura.value = Assinatura

      set oStore = nothing
      set oCertificate = nothing
      set oCertificates = nothing
      set oCertificate1 = nothing
      set oSigner = nothing
      set oAssinador = nothing

    end function

    </script>";

}// MontaFuncaoAssina
----------------------------------------->8-----------------------------------------------
```

A partir dos parâmetros obtidos na função **CarregaConf**, este método basicamente concatena os valores de cada campo que se deseja assinar no formulário. Finalmente, ele gera a função **Assina()**, em VBScript, que deverá ser invocada para que a assinatura digital possa ser criada.

A função **Assina** é constituída completamente por métodos da biblioteca CAPICOM. Os dados concatenados são passados para a propriedade **Content** do objeto SignedData, utilizado para gerar uma assinatura digital através do método **Sign**. A assinatura digital gerada é armazenada no campo **assinatura** (hidden) do formulário.

O método **MontaFuncaoAssina** não requer passagem de nenhuma variável como parâmetro ou argumento.

### 5.3.3 MontaFuncaoVerifica

Ao contrário do método anterior, este método exige que seja passado como parâmetro uma variável contendo as informações concatenadas. É em cima desses dados que será gerado um resumo criptográfico para ser comparado com o resumo criptográfico cifrado e armazenado no Banco de Dados na forma de uma assinatura digital.

A ação desse método consiste basicamente na geração de um código em VBScript, contendo um método denominado **Verifica**, que chama outros métodos da biblioteca CAPICOM, para verificar a validade de uma assinatura digital. Para o método

Verifica() do VBScript,deverá ser passado como parâmetro a assinatura digital que veio do Banco de Dados.

## Código fonte

```
---------------------------------------8<-------------------------------------------------
function MontaFuncaoVerifica($camposConcatenados) {

echo "

<script language=\"VBScript\">

    function Verify(campoAssinatura)

            on error resume next
            Set oSign = CreateObject(\"CAPICOM.SignedData\")
            oSign.Content = \"$camposConcatenados\"

            teste = oSign.Verify(campoAssinatura.value, true, 1)

            if err = 0 then
              Verify = \"OK\"
            else
              Verify = err.description
            end if

    end function
</script>";

} // MontaFuncaoVerifica
----------------------------------------->8-------------------------------------------------
```

# Capítulo 6

# Utilizando o INAFF

## 6.1 Instalação

O INAFF pode ser instalado em qualquer diretório ou pasta. O único requisito é que o servidor Web possua permissão de leitura neste diretório.

## 6.2 Requisitos de Sistema

### *6.2.1 Server side*

Uma vez que este foi escrito em PHP, o INAFF pode ser instalado e executado a partir do Windows ou de um Linux. Os únicos requisitos são:

- Ter instalado e rodando na máquina um servidor http que suporte PHP, por exemplo, Apache;
- PHP;
- MySQL.

### *6.2.2 Client side*

Infelizmente, somente usuários do Windows poderão desfrutar dos recursos oferecidos pelo INAFF, uma vez que o *framework* gera código que invoca objetos e métodos da biblioteca CAPICOM, que só funciona em ambientes do Microsoft Windows. Ainda assim é obrigatório o atendimento dos seguintes requisitos:

- Sistema operacional Microsoft Windows 98/ME/2000/XP/2003;
- Navegador Internet Explorer versão 5.0 ou superior;

- Ter instalado na máquina o certificado digital do usuário que utilizará a aplicação gerada pelo *framework*.

## 6.3 O arquivo de configurações

O INAFF pode ser utilizado em diferentes páginas da Web. Porém, é obrigatório que cada página carregue o INAFF juntamente com alguns parâmetros necessários para o funcionamento do mesmo.

Para acoplar o framework INAFF em sua aplicação Web basta inserir a seguinte linha no início do código PHP:

```
include("path/to/INAFF.php");
```

Optamos por centralizar os parâmetros necessários para a inicialização do INAFF em um único arquivo de configurações, que será usado por todas páginas que chamarem o *framework*.

O arquivo de configuração é bem simples, podendo ser editado em qualquer editor de textos como, por exemplo, o Bloco de Notas do Windows.

Tão logo é carregado o INAFF, o arquivo de configurações é lido e analisado linha por linha. As linhas iniciadas pelo caractere # são utilizadas com a finalidade de comentário, dessa forma sendo ignoradas pelo analisador.

Cada linha que não for um comentário ou não estiver em branco será analisada e tratada como uma linha válida do arquivo de configurações. O formato válido de uma instrução no arquivo de configurações é:

```
idPage,FormName,Campo1Form=Campo1BD,...,CampoNForm=CampoNBD
```

Onde cada campo é separado por vírgula e significam:

- *idPage* – identificador que deve ser único e exclusivo. É a partir desse identificador que será possível resgatar os valores dos outros parâmetros. Uma boa prática de programação é geralmente usar o nome da página que chama o *framework* (página chamadora) como valor para este campo.

- *FormName* – nome do formulário que deve estar declarado na página chamadora, onde se encontram os campos que serão assinados e armazenados na base de dados.
- *CampoNForm* – nome de um campo no formulário. Os valores contidos nesses campos serão concatenados e assinados, no momento da geração da assinatura.
- *CampoNBD* – nome de uma coluna no Banco de Dados. Os valores contidos nesses campos serão concatenados e assinados, sendo comparados com a assinatura também resgatada do Banco de Dados, no momento da verificação/validação da assinatura.

Por exemplo, um programador que deseja, em uma certa página *pag1.php* que possui um formulário que tem como nome "*FormCadastroPessoas*" e alguns campos como "*Nome*", "*Idade*", assinar digitalmente esses valores para garantir a segurança dos mesmos quando estes forem armazenados em uma tabela no Banco de Dados nas colunas DSC_NOME e VLR_IDADE, respectivamente, deverá adicionar a seguinte linha no arquivo de configurações:

```
pag1.php,FormCadastroPessoas,Nome=DSC_NOME,Idade=VLR_IDADE
```

É muito importante saber que a ordem dos campos é significativa. Lembre-se que a assinatura digital depende do conteúdo que está sendo assinado. Portanto, ao assinar "AB" e "BA" percebe-se que serão geradas assinaturas completamente diferentes.

Considere a situação onde, no Banco de Dados, já existem valores assinados de acordo com a configuração acima e, por algum motivo, o programador resolva trocar a ordem dos campos na linha do arquivo de configurações, ficando assim:

```
pag1.php,FormCadastroPessoas, Idade=VLR_IDADE, Nome=DSC_NOME
```

No momento em que for feita a verificação da assinatura, o resultado será inesperado. Pois, por mais íntegros que sejam os dados que ali se encontram, ao se concatenar novamente os valores para gerar uma nova assinatura a ser comparada, a ordem dos campos não será a mesma em que foi feita a primeira assinatura. Por conseguinte, a

nova assinatura será diferente da primeira, e o *framework* acusará que os dados ou a assinatura não são confiáveis.

## 6.4 Exemplo prático: Notas da Graduação

O objetivo deste capítulo é apresentar um estudo de caso realizado para testar o funcionamento prático do INAFF e que possa servir como um tutorial para os usuários do *framework* para que esses criem novas aplicações sobre o INAFF.

Para isto, vamos considerar como exemplo o problema do armazenamento de notas no sistema acadêmico da graduação, em uma universidade. No final de todo semestre, o professor deve acessar o sistema e cadastrar as médias finais obtidas por todos alunos, de cada disciplina que ele ministra na universidade. O sistema consiste em uma aplicação que roda na Web e os dados são entrados através de um formulário contido em uma página HTML, conforme mostra a figura 6.1 da página seguinte.

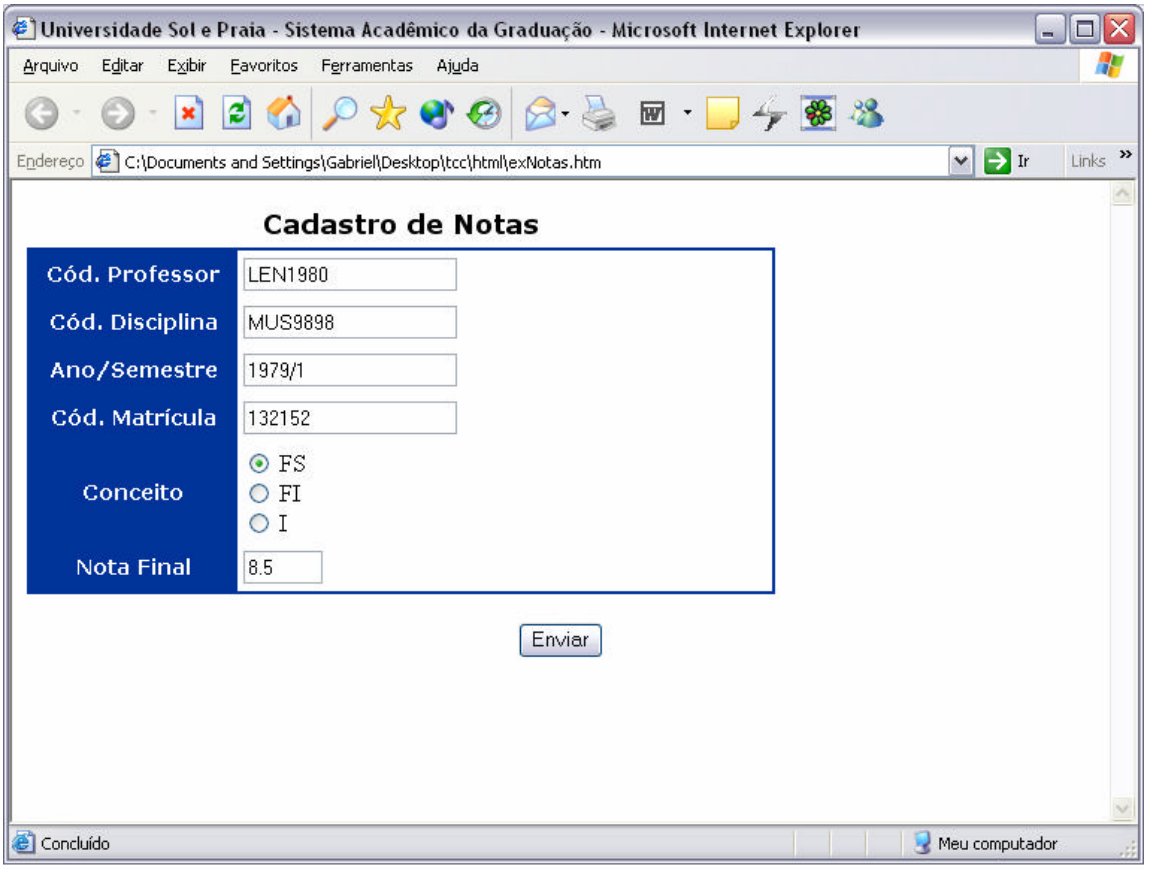



**Figura 6.1 – Estudo de caso: formulário para entrada de dados**

Os campos contidos nos formulário são:

- Código do Professor;
- Código da Disciplina;
- Ano/Semestre – por exemplo, 2004/1;
- Código de Matrícula do aluno;
- Conceito – Freqüência suficiente / Freqüência insuficiente / Indefinido;
- Nota Final – valor compreendido entre 0 e 10, podendo ser representado em frações de 0,5.

O professor preenche esse formulário e, então, clica no botão *Enviar* para transferir as informações ali postadas para uma base de dados. No Banco de Dados relacional utilizado pelo sistema existirá uma tabela pronta para receber o registro, de acordo com a seguinte relação:

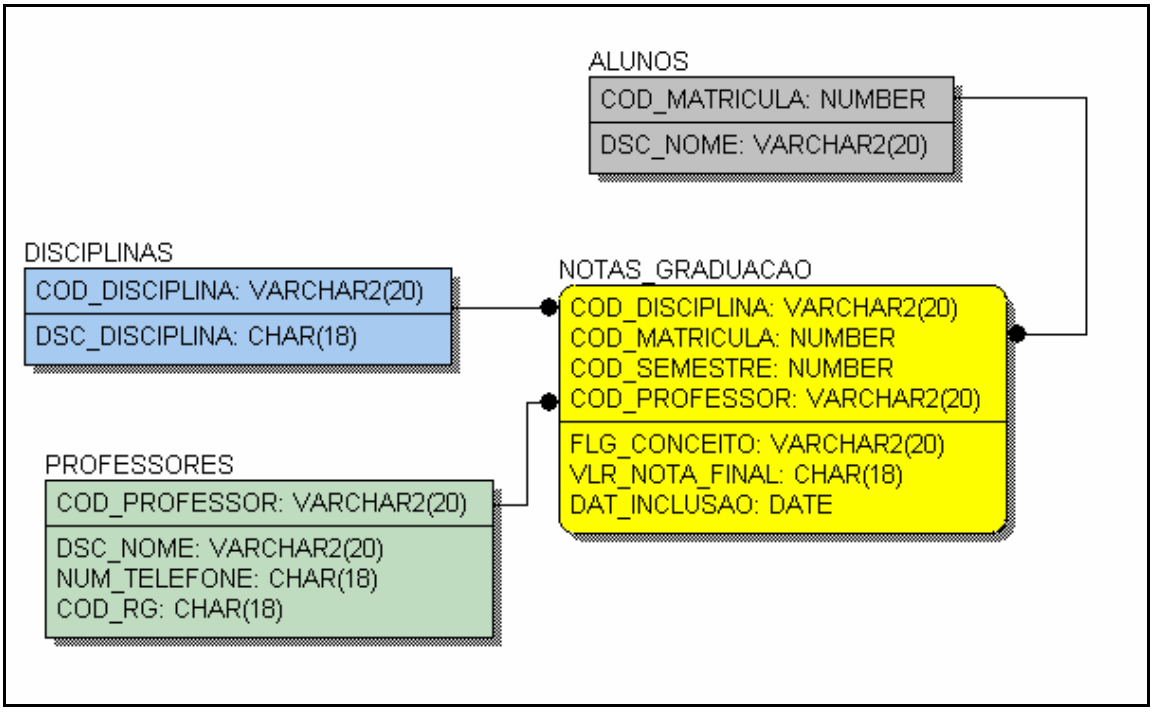



**Figura 6.2 – Parte do modelo relacional do sistema acadêmico**

Os dados então são armazenados na tabela NOTAS_GRADUACAO, cujos campos do código da disciplina, código da matrícula do ano, código do professor e código do semestre formam a chave primária desta relação.

Posteriormente, o aluno ou qualquer pessoa com acesso ao sistema poderá fazer consultas nesta base de dados. Por exemplo, um aluno que deseja saber seu desempenho em uma determinada disciplina, poderá acessar o sistema e pesquisar sua nota, informando os dados da chave (código do semestre, código da disciplina, código da sua matrícula).

Um sistema como esses, do jeito que está apresentado aqui, é muito comum de se encontrar na Internet hoje em dia, sem ter nenhum recurso de segurança implementado para garantir a autenticidade, integridade e confidencialidade dos dados. Esta situação é um prato cheio para os *hackers*, que dispõe de tempo e recursos para explorarem facilmente as vulnerabilidades desses sistemas.

Agora imagine que um inimigo deste aluno, cheio de intenções maliciosas, tenha conseguido, de alguma forma, acesso ao Banco de Dados. Uma vez com acesso, ele poderá alterar quaisquer dados contidos na base de dados. Então, o inimigo simplesmente troca a nota do nosso bom aluno de 6,5 para 5,5. Considerando que nesta universidade a média para aprovação em uma disciplina é 6,0, logo, o aluno estaria reprovado. E agora? Pense em todo o trabalho que será preciso para provar que esta nota foi fraudada, e pergunte-se sobre como esta fraude poderia ser facilmente detectada.

É neste tipo de situação que as assinaturas digitais são muito úteis. Se a nota final do aluno estivesse assinada digitalmente, ficaria fácil de afirmar que aquela nota fora alterada. Tal afirmação pode ser sustentada argumentando como prova que a assinatura digital depende do conteúdo assinado, o que significa que, para notas diferentes, serão geradas assinaturas diferentes. E na hora em que os resumos criptográficos forem comparados, a fraude ficará exposta.

Porém o trabalho de implementar assinatura digital em um sistema como esse não é trivial. Mas, como solução fácil para este problema, agora existe o INAFF.

Inicialmente o arquivo HTML tinha um código parecido com o que está apresentado na figura 6.3 da próxima página.

```
1  <html>
2    <head>
3      <title>Sistema Academico da Graduacao</title>
4    </head>
5    <body>
6    <form name="formNotas" action="insereNota.php">
7      <table>
8        <tr>
9          <td>Cod. Professor:</td>
10         <td><input name="codProfessor" type="text"></td>
11       </tr>
12       <tr>
13         <td>Cod. Disciplina:</td>
14         <td><input name="codDisciplina" type="text"></td>
15       </tr>
16       <tr>
17         <td>Ano/Semestre:</td>
18         <td><input name="codSemestre" type="text"></td>
19       </tr>
20       <tr>
21         <td>Cod. Matricula:</td>
22         <td><input name="codMatricula" type="text"></td>
23       </tr>
24       <tr>
25         <td>Nota Final:</td>
26         <td><input name="vlrNota" type="text"></td>
27       </tr>
28     </table>
29     <br>
30     <center>
31       <input name="btnEnviar" type="button" value="Enviar">
32     </center>
33   </form>
34   </body>
35 </html>
```

**Figura 6.3 – Arquivo inicial (sem sofrer alterações do INAFF)**

Suponha que para este nosso exemplo o programador deseja garantir a integridade e a autoria dos valores contidos nos campos *codMatricula* e *vlrNota* assinando digitalmente essas duas informações, possibilitando, assim, que alguém mais tarde que acesse esses dados tenha a certeza de que os valores não foram alterados e foram gerados por quem assinou.

Se fosse utilizar-se de outros meios para fazê-lo, provavelmente o programador levaria algum tempo até ter essa segurança funcionando em sua aplicação. Porém, ele sabe da existência do INAFF, um *framework* que faz exatamente o que ele quer: pega os valores relacionados a determinados campos, concatena-os e, automaticamente, gera o código necessário para criar uma assinatura digital em cima desses dados.

As alterações que serão necessárias fazer são poucas. O programador terá que, basicamente, fazer:

1. Editar o arquivo de configurações do INAFF e lá incluir uma linha que referencie a página, o formulário e os campos que ele deseja assinar;
2. Incluir no formulário um campo para receber o valor da assinatura. Recomenda-se que o campo seja do tipo *hidden* (oculto);
3. Adicionar um campo na tabela, NOTAS_GRADUCAO neste caso, para armazenar a assinatura digital;
4. Incluir o INAFF em sua aplicação, isto é feito através da instrução **include( )** do PHP;
5. Inicializar o *framework* INAFF;
6. Chamar a função desejada: assinar ou verificar.

Seguir os seis passos acima é suficiente para que o INAFF funcione da maneira esperada, e gere as chamadas para a biblioteca CAPICOM, necessárias para utilizar os recursos da assinatura digital.

Como resultado do passo 1, o usuário do *framework* terá que editar o arquivo de configurações e incluir a seguinte linha:

```
sistemaNotas,formNotas,codMatricula=COD_MATRICULA,vlrNota=VLR_NOTA
```

É importante verificar que o primeiro parâmetro, “sistemaNotas”, deve ser único em todo o arquivo de configurações, pois ele é o identificador utilizado para acessar os outros parâmetros, e serão prevalecidos os valores obtidos na primeira ocorrência encontrada do identificador.

Feito isso, agora é a hora de editar o arquivo onde está o código-fonte em HTML do formulário. Antes do fechamento da tag do formulário, </form>, o desenvolvedor terá que adicionar a seguinte linha:

```
<input name=”assinatura” type=”hidden”>
```

Isso é necessário para que a função **Assina( )**, gerada em VBScript pelo *framework*, armazene o seu resultado, que é a assinatura digital. A assinatura digital é uma string muito grande, e é possível armazená-la em qualquer campo ***input***, de qualquer tipo. Porém, geralmente não se deseja exibir a assinatura digital gerada na tela, por isso utilizamos neste exemplo um campo input com o tipo igual a ***hidden***, que é escondido.

Em algum momento mais tarde esta string contida no campo *assinatura* será automaticamente passada adiante pelo *browser*, como parâmetro, para a página em PHP responsável pela inserção das informações postadas no Banco de Dados, podendo, desta forma, ser utilizada como variável no código PHP.

**NOTA DE PHP:** Todos os campos de um formulário são passados como parâmetros para a página PHP que recebe a ação do formulário. Esta página é definida na declaração do formulário, dentro da tag form no código HTML da página, como a seguir: <form name=formNotas method=POST **action=insert.php**>. Então, assim que um botão com o tipo igual a *submit* for acionado na página, todos os valores contidos nos campos serão passados para a página insert.php, e poderão ser utilizados ali como sendo variáveis globais. Por exemplo, um campo input com nome "codMatricula" se tornará a variável $codMatricula para o PHP.

Finalmente, resta ao usuário do *framework* incluir um código PHP no ínicio do arquivo, dentro dos delimitadores do PHP. O código deverá inicializar o INAFF, passando como parâmetro o identificador da página e, logo em seguida, chamar a função ***MontaFuncaoAssina***, conforme mostrado na ilustração abaixo:

```
1  <HTML>
2  <HEAD>
3
4  <?
5
6    include("v0.0.DES/INAFF.php");
7
8    carregaConf("teste");
9
10   montaFuncaoAssina();
11
12
13 ?>
14
15 </HEAD>
16
17 <body>
```

**Ilustração 6-1 – Inicialização do INAFF e chamada do método montaFuncaoAssina()**

Abaixo é apresentado o novo código-fonte da página Web, após as alterações feitas pelo desenvolvedor, que agora utiliza-se do INAFF para implementar automaticamente a segurança nos dados de sua página:

```
-------------------------------------8<-------------------------------------------------
<HTML>
<HEAD>
<?
  include("v0.0.DES/INAFF.php");
  carregaConf("teste");
  montaFuncaoAssina();
?>
</HEAD>
<body>
 <div align="center"><span class="titulo1">Exemplo Prático - Notas </span> </div>
 <form name="FrmTest" method="POST" action="insere.php">
  <table width="413" border="1">
    <tr>
      <th width="135" scope="row">Semestre</th>
      <td width="262"><input type=text name=vlrSemestre></td>
    </tr>
    <tr>
      <th scope="row">Cod. Disciplina </th>
      <td><input type=text name=codDisciplina></td>
    </tr>
    <tr>
      <th scope="row">Matrícula</th>
      <td><input type=text name=codMatricula></td>
    </tr>
    <tr>
      <th scope="row">Nota</th>
      <td><input type=text name=vlrNota assinar=true></td>
    </tr>
    <tr>
      <th scope="row"> </th>
      <td><input name="button" type=button onClick="Assina()" value="ASSINA">
      <input name="submit" type=submit value="Salvar"></td>
    </tr>
  </table>
  <input type=hidden name=assinatura>
</form>
</body>
</HTML>
----------------------------------------->8-------------------------------------------------
```

Perceba que são pouquíssimas as alterações necessárias!

Assim, quando for feita uma solicitação para acessar a página ao servidor Web, o código-fonte será processado pelo PHP e, como resultado, será gerado o seguinte trecho de código:

```
---------------------------------------8<-------------------------------------------------
    <SCRIPT LANGUAGE="VBSCRIPT">
    function Assina()
      on error resume next

      dadosParaAssinar = document.frmTest.codMatricula.value &
document.frmTest.vlrNota.value

      Set oStore = CreateObject("CAPICOM.Store")
      Set oCertificate = CreateObject("CAPICOM.Certificate")
      Set oCertificates = CreateObject("CAPICOM.Certificates")
      Set oCertificates1 = CreateObject("CAPICOM.Certificates")
      oStore.open

      for each oCertificate in oStore.certificates
        oCertificates.add oCertificate
      next

      set oCertificates1 = oCertificates.select("Selecione um certificado digital","",false)
      Set oSigner = CreateObject("CAPICOM.Signer")
      oSigner.certificate = oCertificates1(1)
      Set oAssinador = CreateObject("CAPICOM.SignedData")
      oAssinador.Content = dadosParaAssinar
      Assinatura = oAssinador.Sign(oSigner,true)
      document.frmTest.assinatura.value = Assinatura

      set oStore = nothing
      set oCertificate = nothing
      set oCertificates = nothing
      set oCertificate1 = nothing
      set oSigner = nothing
      set oAssinador = nothing

    end function
    </script>
----------------------------------------->8-------------------------------------------------
```

Essa parte do código é responsável pela criação da assinatura digital em cima dos dados que foram passados para o formulário na página Web. Este código está escrito na

linguagem VBScript, e necessita da biblioteca CAPICOM instalada na máquina do cliente que estiver acessando a página.

Perceba que o código acima foi gerado automaticamente pelo *framework*. O usuário não terá que programar absolutamente nada em VBScript, muito menos terá que criar ou manipular objetos da CAPICOM.

Por isso que o INAFF é classificado como sendo um *framework* caixa-preta. O usuário não precisa saber como isto está implementado dentro do *framework*. O que ele precisa saber é que existe um método pronto que gera automaticamente aquele trecho de código que é responsável pela criação de assinaturas digitais. Então, ele pode adicionar funcionalidades externas a esse código, utilizando o próprio PHP, por exemplo.

É muito importante também fazer-se notar que o trecho de código gerado pelo *framework* e mostrado acima, foi gerado a partir dos parâmetros que se encontravam configurados no arquivo de configurações.

Agora, o programador precisa associar a função **Assina( )** mostrada acima com o evento *OnClick* de um botão no formulário. Então, quando o usuário da aplicação acionar esse botão, a função **Assina( )** será invocada, pegará os valores contidos nos campos do formulário e os concatenará.

Supondo que anteriormente tinha o seguinte código:

```
<input type=”submit” name=”btnEnviar”>
```

A simples alteração que será necessária fazer nessa linha é a seguinte:

```
<input type=”submit” name=”btnEnviar” OnClick=Assina()>
```

Porém, conforme foi introduzido no Capítulo 2, para que seja possível gerar uma assinatura digital é necessário utilizar a chave privada do assinante. Então, a CAPICOM solicitará para que o usuário da aplicação selecione seu certificado digital dentre uma lista de certificados que será exibida na tela, para que, então, seja possível assinar os dados concatenados com a chave privada do usuário.

A figura da página seguinte mostra o diálogo exibido na tela para a escolha do certificado:

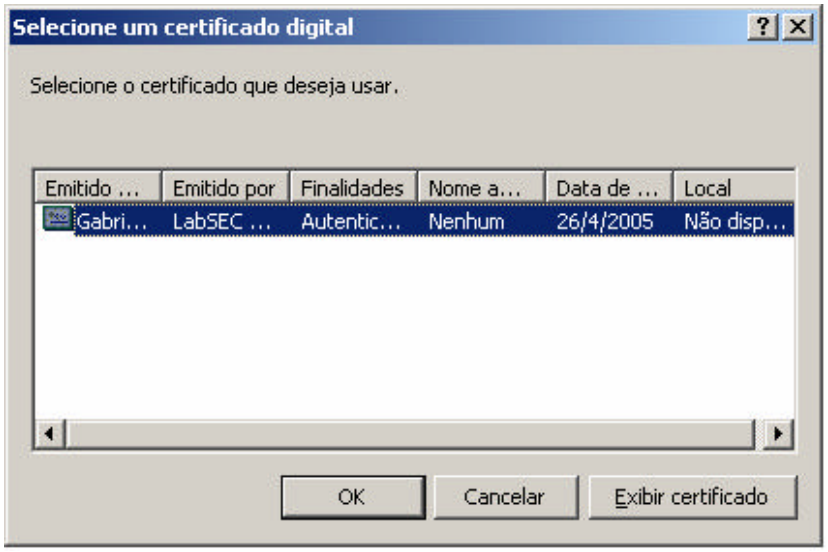



**Figura 6.4 – Diálogo que permite usuário selecionar um certificado digital**

O usuário deverá, então, selecionar o seu certificado digital na lista e clicar em OK para confirmar a escolha. A seguir, o script gerado pelo INAFF invocará o método **Sign** do objeto SignedData da CAPICOM. Este método requer o acesso à chave privada do certificado digital selecionado anteriormente. Portanto, por questões óbvias de segurança e privacidade, será exibido um diálogo de confirmação para que o usuário permita que esse procedimento seja executado legalmente, conforme é mostrado na figura abaixo:

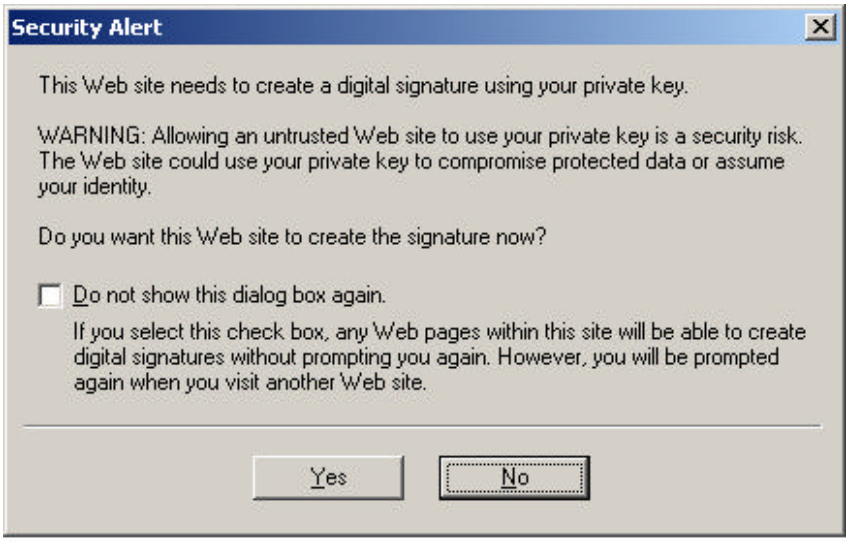



**Figura 6.5 – Diálogo de confirmação para uso da chave privada do usuário**

Na tela mostrada na página anterior o usuário tem a opção de marcar um *checkbox* para que esta mensagem de confirmação não seja mais exibida para ele futuramente. Porém, se isso for feito, das próximas vezes que algum *site* tentar utilizar o certificado digital dele para gerar uma assinatura digital ele não será notificado. Isto significa que poderão ser geradas quaisquer assinaturas com a chave privada do usuário, sem ele saber. Portanto, se você não quiser passar por experiências desagradáveis, não é uma boa idéia que essa opção seja marcada. Mas, finalmente, para dar continuidade com o procedimento será preciso que o usuário autorize o script a utilizar sua chave privada, clicando em "Yes".

Neste ponto, após todo esse processo que foi disparado com o clique no botão "Enviar", será criada a assinatura digital, a qual será armazenada no valor do campo **assinatura**, que anteriormente foi declarado como sendo um campo oculto no formulário.

Agora, provavelmente o programador vai querer armazenar no Banco de Dados as informações postadas através do formulário, juntamente com a assinatura digital criada pelo script gerado pelo INAFF.

Porém, a parte que trata da conexão com o Banco de Dados, bem como as operações de consulta, inserção, alteração, exclusão, dentre outras, fica como responsabilidade do programador, que é o usuário do *framework*. Isto significa que o INAFF não gerará nenhum código que faça interface com o Banco de Dados, pois não é o objetivo do nosso *framework* automatizar esta tarefa.

Neste exemplo utilizamos o MySQL como SGBD. Para fazer com que a aplicação escrita em PHP conecte-se ao banco de dados MySQL adotamos a criação de um script em PHP para tratar exclusivamente da conexão com o banco

Isto é uma prática muito comum entre os programadores PHP, pois uma vez criado o arquivo será possível incluí-lo nas páginas que necessitam de conexão com o banco de dados através da função **include( )** do PHP.

Continuando com o nosso estudo de caso, mostraremos agora como criar um arquivo para conectar no MySQL através do PHP.

Para isto, considere o seguinte arquivo **banco.inc**, como é mostrado na Figura 6.6 abaixo :

```
1  <?php
2
3    $host = "localhost";
4    $usuario = "root";
5    $senha = "fr4m3w0rk";
6
7    $c = mysql_connect("$host", "$usuario", "$senha") ||
8            die ( "Erro ao tentar conexão com o banco de dados!" );
9
10   $db = "tcc";
11
12 ?>
```

**Figura 6.6 – Script PHP para conexão com o Banco de Dados**

Este arquivo deverá ser criado da maneira conforme foi mostrada acima e colocado no diretório onde estão sendo salvos os arquivos desta aplicação-exemplo.

A função **mysql_connect( )** é a responsável por tentar abrir uma conexão com o banco de dados MySQL. Em caso de falhas, a função retornará *false* e, neste caso, será exibida na tela uma mensagem de erro e a execução do programa PHP terminará, devido o uso da função **die( )**.

Os parâmetros que devem ser passados para a função **mysql_connect( )** são:

- **host**: *hostname* da máquina onde está rodando o servidor MySQL;
- **usuário**: nome do usuário ou username para realizar um login no servidor MySQL;
- **senha**: senha referente ao usuário acima, necessária para autenticar-se perante o servidor MySQL;
- **db**: nome da base de dados que será utilizada.

Na figura 6.3 foi mostrado o código-fonte em HTML da página que contém o formulário para a entrada dos dados. A ação desse formulário foi definida como sendo em cima do arquivo **insereNotas.php**. Este arquivo deverá conter um script PHP pronto para receber os dados vindos do formulário, passados como parâmetro pelo navegador e, depois, dar algum tratamento e fim nele.

Para nosso exemplo, armazenaremos os dados em um banco de dados, na tabela NOTAS_GRADUACAO, criada e mostrada anteriormente.

O trecho de código mostrado a seguir insere um registro na tabela NOTAS_GRADUACAO, existente na base de dados "tcc". Perceba que a primeira coisa que foi feita no arquivo cima foi carregar o arquivo **banco.inc**, para que fosse possível enviar comandos para o servidor de banco de dados MySQL. Se houver algum erro na tentativa de conectar-se com o banco de dados, uma mensagem de erro será exibida e a execução do programa será terminada.

```
1  <?
2    include("banco.inc");
3
4    $result = mysql($db, "INSERT INTO NOTAS_GRADUACAO
5                          (COD_DISCIPLINA, COD_MATRICULA,
6                           COD_SEMESTRE, VLR_NOTA, SIG_ASSINATURA)
7                           VALUES
8                          ('$codDisciplina', '$codMatricula',
9                           '$vlrSemestre', 'vlrNota',
10                          '$assinatura')");
11
12
13   if (!$result) {
14       echo "Erro ao inserir registro! <BR>";
15   }
16 ?>
```

**Figura 6.7 – Script PHP para inserir os dados no Banco de Dados**

A função **mysql( )** é responsável por enviar comandos para o servidor MySQL. Para que ela funcione corretamente, é necessária que exista uma conexão aberta com o banco. Tal conexão é realizada no arquivo **banco.inc**. Perceba que a primeira coisa que foi feita no arquivo mostrado acima foi o carregamento do arquivo **banco.inc**. Neste arquivo

também contém o valor da variável **$db**, que é a base de dados onde está armazenada a tabela NOTAS_GRADUACAO.

Agora, o mais importante de se perceber no trecho de código acima é que um dos campos que estamos inserindo é o campo SIG_ASSINATURA, que receberá como valor o conteúdo da variável **$assinatura**. Mas, de onde vem esse valor? Conforme foi dito anteriormente, esse valor é passado pelo *browser*, como parâmetro para a pagina **insereNotas.php**, quando a página recebe a ação do formulário. Essa variável contém a string referente à assinatura digital gerada pela função **Assina( )**.

A seguir é mostrado o formato de uma string contendo uma assinatura digital gerada pela CAPICOM. A string tem esse tamanho todo porque nela é incluída, além da assinatura em si e o certificado digital. Isto é feito para facilitar o processo de validação desta assinatura. Se não fosse assim, dependeria de uma AC para obter a chave pública do assinante para, então, poder verificar a assinatura.

A assinatura apresentada na Figura 6.8 está codificada em BASE64.

A inclusão do certificado na assinatura é opcional. É possível, ainda, também anexar os dados que estão sendo assinados na própria assinatura, através de passar o valor *true* para o parâmetro *bDetached*.


MIINTwYJKoZIhvcNAQcCoIINQDCCDTwCAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3DQEHAaCCC5cwggUJMIID8aADAgECAgohfUmGABEAAAXkMA0GCSqGSIb3DQEBBQUA
MIHKMRkwFwYDVQQDExBMYWJTRUMgLSBDbGFzc2UxMQswCQYDVQQIEwJTQzELMAkGA1UEBhMCQlIxITAfBgkqhkiG9w0BCQEWEmxhYnNlY0BpbmYudWZzYy5icjE2MDQG
A1UEChMtVW5pdmVyc2lkYWRlIEZlZGVyYWwgZGUgU2FudGEgQ2F0YXJpbmEgLSBVRlNDMTgwNgYDVQQLEy9MYWJvcmF0b3JpbyBkZSBTZWd1cmFuY2EgZW0gQ29tcHV0
YWNhbyAtIExhYlNFQzAeFw0wNDA0MjcwMDIwMThaFw0wNTA0MjcwMDMwMThaMIGfMR8wHQYJKoZIhvcNAQkBFhBiaWxsQGluZi51ZnNjLmJyMQswCQYDVQQGEwJCUjEL
MAkGA1UECBMCU0MxFjAUBgNVBAcTDUZsb3JpYW5vcG9saXMxDTALBgNVBAoTBFVGU0MxDzANBgNVBAsTBkxhYlNFQzEqMCgGA1UEAxMhR2FicmllbCBkZSBDYXJ2Ywxo
byBOb2d1ZWlyYSBSZWlzMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDXy+hbh4F0BH7kTYSyztg60ptuN6WE7gyD1jEfBo0u9Iat9ZN8RXbFDAE45p887Pxjb/Re
jvfguAmzdRVhLE4b7ayfi62hg9qvuyy0LhsRqCpryPhidL2UVKsXgx3bYf+mIzFQD5E2ZS5oMjBnzkt35S9y74KTh9fZfpOLK3NGIQIDAQABo4IBnDCCAZgwDgYDVR0P
AQH/BAQDAgTwMB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQcDBDAdBgNVHQ4EFgQUN2gi9WM+hOuJOxOylOc9WaUxSqgwgaQGA1UdIwSBnDCBmYAUxdFtlQ7O6iCx
wtUK5KptW0xyl7qhfqR8MHoxGDAWBgNVBAMTD0FDIFJhaXogZGEgVUZTQzELMAkGA1UECBMCU0MxCzAJBgNVBAYTAkJSMSEwHwYJKoZIhvcNAQkBFhJsYWJzZWNAaW5m
LnVmc2MuYnIxEDAOBgNVBAoTB0FDLVVGU0MxDzANBgNVBAsTBkxhYlNFQ4IBATBKBgNVHR8EQzBBMD+gPaA7hjlodHRwOi8vd3d3LmluZi51ZnNjLmJyL35sYWJzZWMv
YWMvY3JsL2NsYXNzZTEvY2xhc3NlMS5jcmwwVQYIKwYBBQUHAQEESTBHMEUGCCsGAQUFBzAChjlodHRwOi8vd3d3LmluZi51ZnNjLmJyL35sYWJzZWMvYWMvY3J0L2Ns
YXNzZTEvY2xhc3NlMS5jcnQwDQYJKoZIhvcNAQEFBQADggEBAK/Vcs+VbigQWNVsw+70i8MchdrqRXM3T3kgRltrHRGYU86EdudNP7gwaqMrz64vsmyyubyIlz4VjcuY
Rb219/W+t3FiXuMhObi4xG+Z7jRHJtbVf99wcftl8Ewz9r7ndZk8u4Tah1rxCPArZS/zj+mTUq0i67BFyNxZygDn15hfW0pkIaCiPOgGXtKq5e8aPTTrPc3/2IbkYVh3
mPKSShS9I9h4Aceb/M2FptISbCQjHiWWb2KZ+tcIXznqliYg1PjWNzOXNr0ncK7ZrfekHGTgyPwUVQuFYCD4o9J9NhAm9lwYRFfXlTy1HkkloR3nPpzpG3z+L0T4yxRP
s/YJYPcwggaGMIIFbqADAgECAgEBMA0GCSqGSIb3DQEBBAUAMHoxGDAWBgNVBAMTD0FDIFJhaXogZGEgVUZTQzELMAkGA1UECBMCU0MxCzAJBgNVBAYTAkJSMSEwHwYJ
KoZIhvcNAQkBFhJsYWJzZWNAaW5mLnVmc2MuYnIxEDAOBgNVBAoTB0FDLVVGU0MxDzANBgNVBAsTBkxhYlNFQzAeFw0wMzEwMTQxMjI1MDBaFw0xMzEwMTExMjI1MDBa
MIHKMRkwFwYDVQQDExBMYWJTRUMgLSBDbGFzc2UxMQswCQYDVQQIEwJTQzELMAkGA1UEBhMCQlIxITAfBgkqhkiG9w0BCQEWEmxhYnNlY0BpbmYudWZzYy5icjE2MDQG
A1UEChMtVW5pdmVyc2lkYWRlIEZlZGVyYWwgZGUgU2FudGEgQ2F0YXJpbmEgLSBVRlNDMTgwNgYDVQQLEy9MYWJvcmF0b3JpbyBkZSBTZWd1cmFuY2EgZW0gQ29tcHV0
YWNhbyAtIExhYlNFQzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAL0atnTVVGQyD8ALMYfKjSUZagO0+etE1yfVOFL2QdqjpY9BwYRGvT/bb5dnElQhjDk5
1eJVkrKYfHjM7eDtvwjtZ8LGIoSqrkylZ14Zb21wPbQS/EdrQBSzLICQwMV0rzeUq91sEkAzvRCsc17y9uMHO6gdbyEUr/LwgHsmSKz8QtpbFBteUld9SJCcct6BpQYS
/KdeHdG7ExtpKtb3BZDpkyMJelVpBaZiXuElOUB/09BcmCrOA8JX/KUCAnX+/tAY0geygwa9vFsQNNpCL5IpjGWsQUKfS3Gn/FnVcPBPq5WcR1Df7Ycn+H/xNi4JfsOm
VdOLwtR4BzObczv8pvkCAwEAAaOCAsQwggLAMIGkBgNVHSMEgZwwgZmAFMKV+WsDxZgZoKA5n8UfYX97hYLVoX6kfDB6MRgwFgYDVQQDEw9BQyBSYWl6IGRhIFVGU0Mx
CzAJBgNVBAgTAlNDMQswCQYDVQQGEwJCUjEhMB8GCSqGSIb3DQEJARYSbGFic2VjQGluZi51ZnNjLmJyMRAwDgYDVQQKEwdBQy1VRlNDMQ8wDQYDVQQLEwZMYWJTRUOC
AQAwEgYDVR0TAQH/BagwBgEB/wIBADBEBgNVHR8EPTA7MDmgN6A1hjNodHRwOi8vd3d3LmluZi51ZnNjLmJyL35sYWJzZWMvYWMvY3JsL3JhaXovcmFpei5jcmwwggGr
BgNVHSAEggGiMIIBnjCCAZoGBWBMAQEAMIIBjzBEBggrBgEFBQcCARY4aHR0cDovL3d3dy5pbmYudWZzYy5ici9+bGFic2VjL2FjL3BvbGl0aWNhcy9kcGMtdWZzYy5w

ZGYwggFFBggrBgEFBQcCAjCCATcaggEzRXN0ZSBDZXJ0aWZpY2FkbyDpIGV4Y2x1c2l2byBwYXJhIGF1dGVudGljYefjbyB2aWEgbmF2ZWdhZG9yIFdFQiBlIHBhcmEg
YXNzaW5hdHVyYSBlIHNpZ2lsbyBlbSBjbGllbnRlcyBkZSBjb3JyZWlvIGVsZXRy9G5pY28gdXNhbmRvIFMvTWltZSBkYSBVbml2ZXJzaWRhZGUgRmVkZXJhbCBkZSBT
YW50YSBDYXRhcmluYS4gRXN0ZSBjZXJ0aWZpY2FkbyBz8yBnYXJhbnRlIHF1ZSB1bWEgY29udGEgZGUgY29ycmVpbyBlbGV0cvRuaWNvIGRvIGRvbe1uaW8gdWZzYy5i
ciBleGlzdGlhIHF1YW5kbyBvIGNlcnRpZmljYWRvIGZvaSBlbWl0aWRvLjAOBgNVHQ8Baf8EBAMCAQYwDQYJKoZIhvcNAQEEBQADggEBAEOa5r56eFAYnr2v6n0nZIQE
12JJ/4nNZ5TZEXBo1GF0EgJWvptEOsL5teJw1C0MZS69JjdkfraWx/7/wDxe9+BLDt47uZXGqfEWEcHw82ML7S8LiFZV/pM/8+Pqp7p4np/LXj1zDqv5adjRqkF41uJi
fCs97FPcKXBymmLiF/WctYyQYie9DCqj7lZMGmP01U/OIGlN9v5qDzCW+jp3iMniQ1oAGAHLL1GP7RA8qtnXBiTmEQQZGhLxbHzc6dGG7IlXfMsQguy8MvpOwaS1401V
81E2sTZbhZUTV2Ebyza4eMtv0z2R1vE/JYSOAL6YLogwUWwyIVTa6YNsz4OdQVwxggGAMIIBfAIBATCB2TCByjEZMBcGA1UEAxMQTGFiU0VDIC0gQ2xhc3NlMTELMAkG
A1UECBMCU0MxCzAJBgNVBAYTAkJSMSEwHwYJKoZIhvcNAQkBFhJsYWJzZWNAaW5mLnVmc2MuYnIxNjA0BgNVBAoTLVVuaXZlcnNpZGFkZSBGZWRlcmFsIGRlIFNhbnRh
IENhdGFyaW5hIC0gVUZTQzE4MDYGA1UECxMvTGFib3JhdG9yaW8gZGUgU2VndXJhbmNhIGVtIENvbXB1dGFjYW8gLSBMYWJTRUMCCiF9SYYAEQAABeQwCQYFKw4DahoF
ADANBgkqhkiG9w0BAQEFAASBgLa7/uotQDfkF1TNAMIyGmWNGJsmuwHA8UWQWpDO8YD225c5eeksY05VXx2IQTSl0+gi++W4z2JoVf5SZs6A5Kuhv5Mztq7enwJI1iF+
/xzcs2CABjggot9vQ6eMOiuWYLcMCJTfVk8U4Qxqf/2HnnvHGbrxvRh+QysriWhw2IuN

**Figura 6.8 – String de uma assinatura digital gerada pela CAPICOM**

Desta forma tem-se implementado um mecanismo de assinatura digital que oferece segurança aos dados trafegados nos caminhos formulário Web para base de dados (ida) e base de dados para página da Web (volta). Simples, e sem dificuldades, não é mesmo?

Vimos, com este exemplo, que se o programador que estivesse desenvolvendo este sistema de gerenciamento de notas da graduação optasse por utilizar o INAFF para adicionar esta segurança à aplicação, o mesmo não teria de gastar muito tempo para fazê-lo, pois com apenas alguns passos e com o mínimo de alterações o INAFF gera o código suficiente para atender as necessidades do usuário do *framework*.

Vamos supor que alguém, um inimigo qualquer, tenha conseguido acesso ao Banco de Dados onde está a tabela NOTAS_GRADUACAO. Conseqüentemente esta pessoa tem acesso aos dados ali armazenados, podendo inclusive alterá-los.

Para que seja possível que o usuário se certifique de que os dados que estão sendo mostrados ali na tela para ele não foram alterados, o INAFF possui a funcionalidade de gerar código para verificação ou validação de assinaturas digitais.

De maneira similar ao processo de geração de assinaturas, aqui também é chamada apenas um método do framework. O nome do método é **MontaFuncaoVerifica().**

Uma vez chamando esse método, será gerado automaticamente o código para verificação de assinaturas, utilizando a biblioteca CAPICOM.

Lembre-se de que será preciso passar os dados concatenados a serem verificados para esta função, de modo que seja possível comparar os resumos.

# Capítulo 7

# Considerações finais

## 7.1 Conclusão

O objetivo geral deste trabalho foi atingido com o projeto e implementação do INAFF, apresentados nos capítulos 4 e 5. O INAFF é um *framework* simples, porém muito robusto, que têm como principal funcionalidade automatizar a geração de código para criação e validação de assinaturas digitais.

## 7.2 Trabalhos futuros

Como trabalhos futuros, poderão ser cogitadas questões como:

- Desenvolver uma IDE para criar aplicações que utilizam o *framework*. Dessa forma, a interação do usuário do *framework* com este se daria através de interface gráfica, e o processo do desenvolvimento se tornaria ainda mais fácil, pois seriam implementados mais recursos para automatizar e ajudar o usuário do *framework* a desenvolver aplicações;
- Permitir que o INAFF gere códigos para as mais diferentes linguagens de programações existentes hoje em dia para desenvolver aplicações para Web, de modo com que a aplicação final possa ser executada em várias plataformas diferentes, a gosto do desenvolvedor usuário do *framework*;
- Verificação de CRL e validação de certificado antes que a assinatura seja criada;
- Permitir que o *framework* possa trabalhar com dados provenientes de múltiplos formulários e tabelas diferentes;

- Permitir que o usuário do INAFF possa parametrizar valores como nome do campo de assinatura, nomes das funções para assinar e verificar, dentre outros, no arquivo de configurações;
- Adicionar novos recursos que permitam interoperabilidade entre Banco de Dados. Atualmente, o INAFF não tem a responsabilidade de tratar das transações com o Banco de Dados. Porém, futuramente, seria muito interessante que isso se tornasse possível;
- Explorar melhor os recursos oferecidos pela CAPICOM para poder melhorar o código final gerado pelo *framework*, principalmente nas funções de Assinar e Verificar;

# Referências Bibliográficas

[APA04] APACHE. The Apache Software Foundation. **About the Apache HTTP Project**. Disponível em: http://httpd.apache.org/ABOUT_APACHE.html. Acesso em: Abril de 2004.

[BUR02] BURNETT, S.; PAINE, S. **Criptografia e segurança - O guia oficial RSA**. Rio de Janeiro: Editora Campus, 2002.

[CAM03] CAMPANA, I.; SILVA, V. **Sistema seguro de compras**. Florianópolis: Universidade Federal de Santa Catarina, Fevereiro, 2003. Trabalho de conclusão de curso de graduação.

[CAR01] BALPARDA, D. **Segurança de dados com criptografia - Métodos e algoritmos**. 2 ed. Rio de Janeiro: Editora Book Express, 2001.

[EXT03] EXTERKOETTER, F. **Blendwork: Framework Orientado a Objetos para Desenvolvimento Rápido de Aplicações Comerciais Cliente/Servidor**. Florianópolis: Universidade Federal de Santa Catarina, Maio, 2003. Dissertação de Mestrado.

[FAY99] FAYAD, M.; SCHMIDT, D.; JOHNSON, R. **Building Application Frameworks: Object-Oriented Foundations of Framework Design**. New York, NY: John Wiley and Sons, 1999.

[GAM95] GAMMA, E.; et al., 1995. Tradução de Luiz A. Meireles Salgado. **Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos**. Porto Alegre: Bookman, 2000.

[JOH93] JOHNSON, R. **How to Design Frameworks**, Tutorial Notes, OOPSLA 93, Washington, 1993. Disponível em http://www.cse.msu.edu/~cse870/Materials/Frameworks/how-to-design-fw-tutorial.ps. Acesso em: Maio de 2004.

[MEN96] MENEZES, A.; OORSCHOT, P.; VANSTONE, S. **Handbook of applied cryptography**. CRC Press, 1996.

[SCH95] SCHNEIER, B. **Applied cryptography - Protocols, algorithms and source code in C**. 2 ed. Editora John Wiley & Sons, 1995.

[SIL00] SILVA, R. **Suporte ao Desenvolvimento e Uso de Frameworks e Componentes**. 262 f. Teste (Doutorado em Ciência da Computação) – Programa de Pós-graduação em Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre. 2000.

[SIL99] SILBERSCHATZ, A.; KORTH, H.; SUDARSH. **Sistema de Banco de Dados**. Makron Books, São Paulo, 1999.

[STA99] STALLINGS, W. **Cryptography and Network Security: Principles and Practice**. 1999.

[ZIM98] ZIMMERMANN, P. **An Introduction to Cryptography**. PGP: Pretty Good Privacy Documents. 1998.

# Anexos

### Código-fonte

```
<?
#########################################################################
# INAFF Framework para suporte a base de dados seguras na Web     #
# Trabalho de Conclusão de Curso 2004.1                           #
# Universidade Federal de Santa Catarina                          #
# Curso de Ciências da Computação                                 #
# Alunos: Gabriel de Carvalho Nogueira Reis - 0013215-2           #
#                   <bill@inf.ufsc.br>                            #
#         Victor Hugo Germano - 0023248-3                         #
#                   <victorhg@inf.ufsc.br>                        #
#                                                                 #
# Orientação: Prof. Dr. Luiz Carlos Zancanella                    #
#                   <zancanella@inf.ufsc.br>                      #
#########################################################################



 $configFile = "v0.0.DES/include/INAFF.conf";
 $constFile = "v0.0.DES/include/INAFF.const";


/*
*******************************************************
**** ATENÇÃO: Não edite mais nada a partir daqui   ****
**** WARNING: Do not edit anything below this line ****
*******************************************************
                                                      */

 if (file_exists($configFile)) {
 } else {
   die("<FONT FACE=VERDANA SIZE=2 COLOR=RED><B>ARQUIVO DE CONFIGURAÇÕES
\"$configFile\" NÃO ENCONTRADO...</B></FONT>");
 }

 if (file_exists($constFile)) {
   include($constFile);
 } else {
   die("<FONT FACE=VERDANA SIZE=2 COLOR=RED><B>ARQUIVO DE CONSTANTES
\"$constFile\" NÃO ENCONTRADO...</B></FONT>");
 }




function carregaConf($nomePagina) {

  global $parametros, $configFile;
```

```
  $linhas = file($configFile);

  $linhaAtual = "";


  $achou = false;

  for ($i=0; $i<count($linhas); $i++)
  {
    $linhaAtual = $linhas[$i];
    // se não for comentário ou linha em branco, então analisa a linha
    if (($linhaAtual[0] != "#") && (trim($linhaAtual)!="")) {
        $parametros = explode(',', $linhaAtual);
        if ( $parametros[0] == $nomePagina ) {
          $achou = true;
          break;//sai do for
        }//if
    }//if
  }//for

  if (!$achou) {
    die ("$fontErro PÁGINA NÃO ENCONTRADA NO ARQUIVO DE CONFIGURAÇÕES.
CANCELANDO...</FONT>");
  }

}//carregaConf







function montaFuncaoAssina() {
  global $parametros, $strEchoAssina, $nomeFormulario;

  $nomeFormulario = $parametros[1];

  $strEchoAssina = "dadosParaAssinar = ";

  for ($i=2; $i<count($parametros); $i++)
  {
    $campo = explode('=',$parametros[$i]);

//    print_r($campo);

    $strEchoAssina =
$strEchoAssina."document.".$nomeFormulario.".".$campo[0].".value";

    if ($i == (count($parametros) - 1)) {
      $strEchoAssina = $strEchoAssina."\n";

    } else {
      $strEchoAssina = $strEchoAssina." & ";
```

```
    }//if-else

  }//for

    // Escrevendo na página HTML
    echo "

    <SCRIPT LANGUAGE=\"VBSCRIPT\">
    function Assina()
      on error resume next

      $strEchoAssina

      Set oStore = CreateObject(\"CAPICOM.Store\")
      Set oCertificate = CreateObject(\"CAPICOM.Certificate\")
      Set oCertificates = CreateObject(\"CAPICOM.Certificates\")
      Set oCertificates1 = CreateObject(\"CAPICOM.Certificates\")

      oStore.open

      for each oCertificate in oStore.certificates
        oCertificates.add oCertificate
      next

      set oCertificates1 = oCertificates.select(\"Selecione um
certificado digital\",\"\",false)
      Set oSigner = CreateObject(\"CAPICOM.Signer\")
      oSigner.certificate = oCertificates1(1)

    '  MsgBox dadosParaAssinar remover depois
      Set oAssinador = CreateObject(\"CAPICOM.SignedData\")
      oAssinador.Content = dadosParaAssinar
      Assinatura = oAssinador.Sign(oSigner,true)




      document.$nomeFormulario.assinatura.value = Assinatura  'MEXER



      set oStore = nothing
      set oCertificate = nothing
      set oCertificates = nothing
      set oCertificate1 = nothing
      set oSigner = nothing
      set oAssinador = nothing


    end function


    </script>";


}// MontaFuncaoAssina
```

```
# Esta função recebe como parâmetro o resultado de uma consulta SQL
(query)
# Então o que a função faz é pegar os campos BD do arquivo de
configurações
# e montar a string com os dados a serem verificados
function MontaStringDadosConcatenados($SQL_result) {
  global $parametros, $assinaturaParaVerificar, $dadosParaVerificar;

  $dadosConcatenados = "";
  if ($SQL_result) {

        $linha = mysql_fetch_array($SQL_result, MYSQL_ASSOC);
        $nomeCampoBD = "";

        $assinaturaParaVerificar = $linha["SIG_SIGN"];

        for ($i=2; $i<count($parametros); $i++)
        {
          $campo = explode('=',$parametros[$i]);
          $nomeCampoBD = trim($campo[1]);
          $dadosConcatenados = $dadosConcatenados .
$linha["$nomeCampoBD"];
        }//for

        $dadosParaVerificar = $dadosConcatenados;

  }//if

  return $dadosConcatenados;
}// MontaStringDadosConcatenados




# Esta função monta o conteúdo em VBScript que utiliza-se da CAPICOM
# para comparar a assinatura dos dados passados como parâmetro com
# uma assinatura já existente

function MontaFuncaoVerifica($camposConcatenados) {

echo "

<script language=\"VBScript\">

    function Verify(campoAssinatura)

            on error resume next
            Set oSign = CreateObject(\"CAPICOM.SignedData\")
            oSign.Content = \"$camposConcatenados\"

            teste = oSign.Verify(campoAssinatura.value, true, 1)

            if err = 0 then
```

```
              Verify = \"OK\"
            else
              Verify = err.description
            end if

    end function
</script>";

} // MontaFuncaoVerifica



function INAFF_Verifica($SQL_result) {
  $dadosParaVerificar = MontaStringDadosConcatenados($SQL_result);
  MontaFuncaoVerifica($dadosParaVerificar);
}



?>
```

# INAFF: UM FRAMEWORK PARA SUPORTE A BASES DE DADOS SEGURAS NA WEB

Gabriel Carvalho Nogueira Reis (bill@inf.ufsc.br), Victor Hugo Germano (victorhg@inf.ufsc.br)
Departamento de Informática e Estatística, Universidade Federal de Santa Catarina

**Resumo**

O presente artigo visa apresentar a implementação de um framework para oferecer suporte à segurança em base de dados na Web. Baseada nas assinaturas digitais, a ferramenta INAFF tem como principal objetivo tornar mais fácil a tarefa do programador que precisa implementar recursos para garantir a integridade, autenticidade e autoria dos dados trafegados em uma aplicação Web.
Palavras-chave: segurança, criptografia, assinatura digital, framework, aplicações Web.

**Abstract**

This article presents a framework implementation to offers security support to Web databases. Based on digital signatures, the INAFF tool has as the main target make the programmer's job easier, who has to implement resources to guarantee integrity, authenticity and authorship of transmitted data through a Web application.
Keywords: security, cryptography, digital signature, framework, Web applications.

## 1. Introdução

Com a disponibilização cada vez maior de funcionalidades ligadas à área comercial dentro da Internet, isto é: facilidades de compra e venda de qualquer tipo de produto, possibilidades infinitas de análise de conteúdos cadastrados em *sites* de pesquisa ou ainda a capacidade do acesso a dados compartilhados em qualquer parte do planeta, cria-se também um grande problema para gestores da informação.

Sendo a informação algo valioso, deve-se prezar por sua confiabilidade e seu sigilo. Assim, técnicas de controle de acesso a dados, e garantias de confiabilidade na extração de informação vêm sendo estudadas e testadas à exaustão há muito tempo.

Mecanismos de comércio eletrônico, por exemplo, devem garantir que uma entidade que está disponibilizando um produto seu possa ter certeza de que, em nenhum momento, será lograda por um comprador, ou ainda garantir que nenhum agente malicioso possa se passar por um vendedor, destruindo a reputação de tal entidade.

Existem então, dois processos a serem analisados: garantir a identidade do lado que disponibiliza o serviço pelo agente utilizador, e garantir ao lado servidor de uma funcionalidade a identificação confiável do utilizador e da informação enviada por tal.

O presente trabalho trata do segundo caso de análise indicando como solução uma aplicação que provê funcionalidades com o objetivo de garantir a confiabilidade dos dados enviados.

## 2. Conceitos

### 2.1 Fluxo da informação

O fluxo normal da informação que trafega através de um meio pode ser definido como sendo o caminho percorrido, sem interrupções entre duas entidades: A e B.

Entretanto, existe, obviamente a possibilidade que, de alguma maneira, o meio seja pouco confiável, possibilitando alguns problemas nesta comunicação. Assim, definem-se quatro tipos de deturpação do fluxo normal da informação: modificação, interrupção, interceptação e fabricação.

Modificação é também chamado de "ataque do homem do meio", indicando o processo de, durante o tráfego da informação, algum agente receber a informação transeunte, alterá-la, e repassar tal informação ao seu destino, sem que possa ser percebido qualquer erro na comunicação entre A e B.

Interrupção ocorre no momento em que o fluxo da informação é perdido. As causas deste tipo de problema podem ser, por exemplo, saída do destino (B) da rede de

comunicações, problemas no meio ou ainda bloqueadores de acesso ao destino (*firewalls*).

Interceptação é o ataque mais difícil de ser detectado, uma vez que o fluxo de informação em nada é modificado, mas toda a informação que trafega no meio é também visualizada por um terceiro agente malicioso. Assim, este agente poderá ter todo o acesso à informação, o que pode ser um enorme problema.

Fabricação é o problema relacionado à falsidade de identificação da entidade emissora da informação. Assim, um agente malicioso se faz passar por A e envia para B a informação que quiser.

### 2.2 Terminologia

Para que um sistema ofereça um fluxo seguro da informação a partir de A para B, ele deve atender principalmente os seguintes requisitos:

**Confidencialidade** – ou sigilo, é a garantia de que somente as partes envolvidas na comunicação possam ler e utilizar as informações transmitidas eletronicamente pela rede. Isto é, caso um inimigo consiga interceptar de alguma forma o fluxo de informação, a mensagem transmitida deve ser indecifrável;

**Integridade** – garantia de que o conteúdo de uma mensagem não será alterado durante seu tráfego, sem sofrer alterações, isto é, os dados que foram gerados em A são os dados que foram recebidos por B;

**Autenticação** – garantia da identificação das partes envolvidas na comunicação, tendo certeza de que a comunicação esteja realmente sendo feita entre A e B, ou seja, que não há nenhum agente malicioso se fazendo passar por A para gerar uma informação falsa, ou se passando por B para simular a recepção da mensagem enviada por A;

**Não repúdio** – garantia de que o emissor de uma mensagem não poderá, posteriormente, negar sua autoria.

## 3. Criptografia

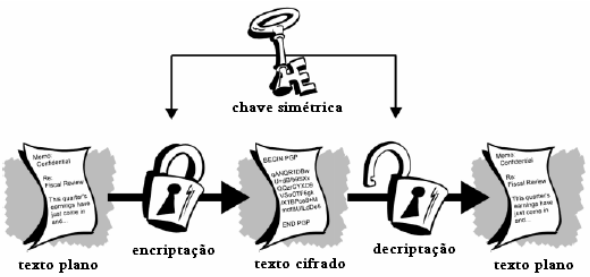
[BUR03] "A Criptografia converte dados legíveis em algo sem sentido, ilegível. Estes dados poderão ser recuperados posteriormente a partir desses dados sem sentido".

Desde a Antigüidade, tão logo o homem aprendeu a escrever, ele começou a pensar em alguma forma de esconder o que ele havia escrito. Foi assim que começou a se desenvolver a Criptografia, base tecnológica para problemas de segurança em comunicações e em computação. Hoje em dia, esta ciência utiliza conceitos e fundamentos matemáticos para a construção de seus algoritmos. Um estudo sobre aritmética modular e teoria dos números primos é indispensável para quem deseja conhecer mais profundamente o assunto.

### 2.1 Criptografia Assimétrica

É a forma mais convencional de criptografia que existe. Por isso, também é conhecida por Criptografia Clássica.

Os algoritmos de criptografia de chaves simétricas são assim denominados por apresentarem como principal característica o fato de utilizarem apenas uma chave. Isto significa que a mesma chave que é utilizada para cifrar é também usada para decifrar. Os algoritmos para cifrar e decifrar também são os mesmos. O que muda é apenas a forma como são utilizadas as chaves. É tido como pré-requisito que os usuários envolvidos compartilhem esta chave entre si. A troca de chaves é geralmente feita através do estabelecimento de um canal de comunicação seguro, através de tunelamento, entre as duas partes envolvidas.




Existem vários algoritmos de criptografia simétrica. Porém os exemplares mais conhecidos e utilizados desta classe atualmente são:

**AES** - *Advanced Encryption Standard*, um algoritmo concebido em outubro de 2000 para substituir o famoso, e cada vez mais vulnerável, algoritmo *Data Encryption Standard* (DES) e 3DES.

**Blowfish** - cifrador de bloco simétrico desenvolvido por Bruce Schneier, em 1993, e permanece inquebrável até hoje. Possui as características de ser rápido, compacto, simples e possuir o tamanho de sua chave variando entre 32 e 448 bits. Opera com blocos de 64 bits.

O principal problema encontrado ao utilizar a criptografia simétrica é a questão do gerenciamento das chaves. Para cada par de usuários que desejam se comunicar é necessário uma chave, o que torna o gerenciamento muito complexo. Em uma rede com n usuários, serão necessárias n(n-1)/2 chaves para que todos os usuários possam trocar informações de maneira sigilosa entre si.

### 2.2 Criptografia Assimétrica

Este tipo de criptografia utiliza duas chaves diferentes: uma usada para cifrar o texto na origem e a outra usada

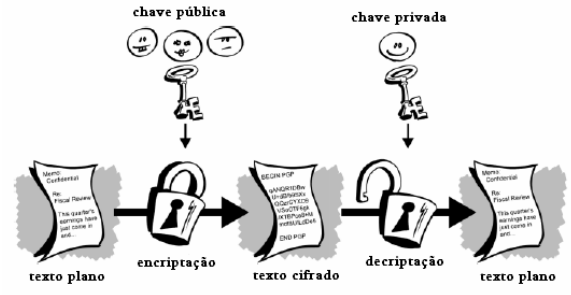
para decifrar o texto quando este chegar no destino. Este sistema gera um par de chaves que são, na verdade, números primos muito grandes relacionados entre si, através de propriedades da aritmética modular.

Uma das chaves será chamada de chave privada e deverá ser mantida em sigilo. A outra chave será chamada de chave pública e deverá ser compartilhada com outras pessoas.

Se uma pessoa A cifra uma informação com sua chave privada, outra pessoa B somente poderá decifrar a mensagem com a chave pública de A, tendo a certeza de que esta mensagem realmente teve sua autoria em A, pois somente A possui a chave privada. Por outro lado, se A cifrar uma mensagem com a chave pública de B, somente B poderá decifrar a mensagem com sua chave pública, garantindo desta forma o sigilo da mensagem.

A grande vantagem da criptografia assimétrica é a eficiência no sigilo. Porém, para cifragem de mensagens grandes ela é ineficiente se tido como parâmetro o tempo que tal processo leva. O que muito se faz é empregar a criptografia assimétrica para realizar a troca de chave de criptografia simétrica, de modo que esta última é cifrada com chaves assimétricas, podendo ser transmitida eletronicamente pela rede de maneira de segura.

A seguir é apresentada uma figura que mostra o funcionamento da criptografia de chaves assimétricas.




## 2.3 Funções de Resumo

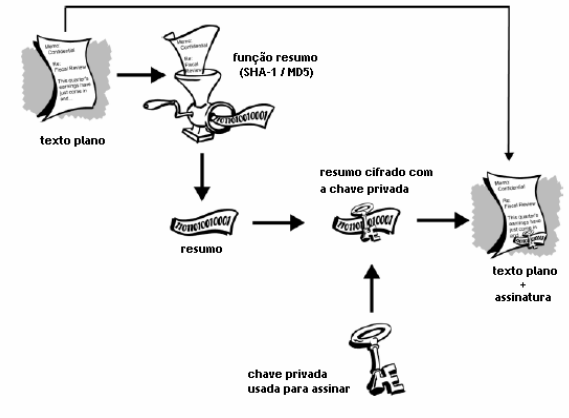
Como os algoritmos de criptografia assimétrica são lentos, não é uma boa idéia cifrar um grande volume de dados, como um texto inteiro.

Na prática, o que se faz é gerar um resumo dos dados e depois cifrar este resumo utilizado a chave privada do autor dos dados. Estes resumos são gerados através das chamadas funções de hash. Tais funções são um algoritmo que recebe qualquer comprimento de entrada e mescla a entrada para produzir uma saída pseudo-aleatória de tamanho fixo.

A palavra "hash" pode significar desordem ou confusão, o que descreve eficientemente o resultado de uma mensagem resumida.

Uma propriedade verificável nestas funções é que elas são irreversíveis, isto é, não é possível reconstruir o conjunto original de dados a partir do resumo. Outra particularidade que é importante saber é que até hoje não foram verificadas colisões – termo técnico empregado para descrever uma situação em que duas mensagens produzem um mesmo resumo. Elas existem, mas ninguém consegue encontrar uma colisão por demanda.

A figura abaixo esquematiza a utilização do resumo criptográfico nas assinaturas digitais:




## 2.4 Assinatura Digital

Verificar a integridade dos dados muitas vezes não basta. É preciso também poder garantir a autoria destes.

Baseadas na aplicação de técnicas de criptografia assimétrica, as assinaturas digitais foram feitas para que uma entidade possa, digitalmente, "assinar" um documento eletrônico como, por exemplo, um arquivo texto ou um e-mail, comprovando de forma única e exclusiva a autoria dos dados. Espera-se que uma assinatura digital possua as mesmas características de uma assinatura qualquer no mundo real.

Estas características desejáveis são:

- A assinatura digital deve ser fácil de produzir por quem assina
- Deve ser fácil de ser verificada por qualquer pessoa
- Deve ser muito difícil de ser falsificada
- Quem assine não possa negar que assinou (não-repúdio)

Além das quatro características citadas acima, é requisito fundamental que a assinatura digital dependa do conteúdo assinado. Isto é, a assinatura deve sempre ser diferente para diferentes informações assinadas. É nesta propriedade que se encontra o grande poder das assinaturas digitais. Senão, é possível imaginar como seria fácil de forjar a assinatura: bastaria o inimigo pegar

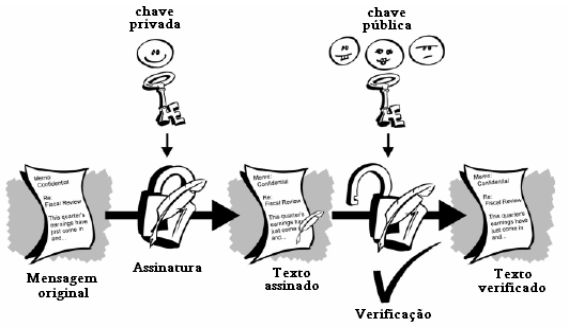
a assinatura em um outro documento assinado e utilizá-la no documento o qual ele deseja falsificar.

É importante também saber que todo conjunto de dados cifrados com uma chave privada é uma assinatura digital, pois só pode ter sido gerado pela pessoa que possui a chave privada.

Forjar uma assinatura digital significa que alguém é capaz de criar uma massa de dados, por um outro meio, que fosse idêntica à assinatura. Isso significaria que o forjador quebrou o algoritmo RSA, o que é altamente improvável. É nisso que são baseadas as propriedades do não-repúdio e não-falsificação.

As assinaturas digitais têm sido implementadas basicamente utilizando conceito de chaves públicas com o padrão RSA, para cifrar resumos gerados a partir de funções de hash como MD5 e SHA-1. Existem também o DSS - *Digital Signature Standard* – que é um algoritmo muito utilizado para criação de assinaturas digitais e existe um esforço para torná-lo um padrão para tal finalidade.

Abaixo é apresentado um esquema simplificado do funcionamento das assinaturas digitais em geral:




A chave privada é utilizada pelo assinante para assinar, enquanto a chave pública é utilizada para verificar a autenticidade da assinatura.

É muito importante esclarecer que uma assinatura digital não é feita para proteger uma mensagem contra um suposto inimigo. Ela tem como principal finalidade garantir que uma determinada pessoa, a autora da assinatura, realmente tenha assinado tal mensagem.

Todos que quiserem verificar a assinatura terão que ter acesso à assinatura e à mensagem. O inimigo, então, poderá verificar a assinatura. A idéia é evitar apenas que se falsifique a assinatura.

Para fins de confidencialidade e segurança das informações devem ser utilizadas técnicas de criptografia simétrica.

Uma maneira de criar assinaturas verificáveis é cifrar o resumo criptográfico com a chave privada RSA do assinante.

**2.5 Certificados Digitais**

Pergunta: Como alguém pode verdadeiramente saber se uma chave pública pertence a uma determinada pessoa em questão?

A maneira mais comum de verificar isso é por meio de um certificado digital, que associa um nome à uma chave pública. Uma analogia para compreender isso melhor é o passaporte, que associa uma foto, a um número e a um nome.

Um certificado digital é produzido de tal maneira que torna perceptível se um impostor pegou um certificado existente e substituiu a chave pública ou o nome ali contido. Qualquer pessoa ao examinar esse certificado fraudado saberá que algo está errado e, portanto, não confiará na combinação desse par de chaves e nome.

As Autoridades Certificadoras – AC – servem como terceiros confiáveis para associar uma identidade de uma pessoa a sua chave pública. A AC tem como principal tarefa a autenticação de seus usuários finais. Para isso, é preciso que a AC forneça sua própria chave pública para todos os usuários finais certificados, bem como para todas as partes verificadoras que podem utilizar as informações certificadas, pois para poder validar um certificado digital é necessário ter a chave pública da AC.

Os certificados digitais constituem um meio seguro de distribuir a chave pública para quem quiser verificar sua assinatura digital. O formato de certificado mais amplamente aceito é o X.509 v3 da International Telecommunications Unions (ITU-T), e apresenta alguns campos como: versão, número serial, identificador do algoritmo de assinatura, nome do emissor (AC), validade, nome do sujeito, chave pública, identificação do algoritmo de chave pública, etc.

A utilização de tal conceito deverá ser de uma maneira que possibilite a definição prévia de qual é a informação que necessita ser assinada, e possibilitar a verificação de autenticidade de dados assinados. Sabendo também dos problemas envolvidos com a cifragem de grande volume de dados utilizando o conceito de Criptografia Assimétrica, a aplicação deve restringir-se apenas a assinar campos dentro de um formulário, útil em casos de comércio eletrônico e notas para avaliações acadêmicas.

Foram introduzidos fundamentos como confidencialidade, autenticação, integridade, não-repúdio, criptografia simétrica, criptografia assimétrica, assinaturas digitais, funções de hash e certificados digitais.

Dentre os tipos de ataques ou quebras do fluxo normal de informação durante uma comunicação, pode-se perceber que a solução que será apresentada neste documento impossibilita o "Ataque do homem do meio", ou modificação, uma vez que mesmo que com os dados alterados, pode-se verificar sua autenticidade, encontrando qualquer problema.

## 3. Frameworks

[JOH93] diz que "um framework é um conjunto de classes que incorpora um projeto abstrato para soluções destinadas a uma família de problemas relacionados".

### 3.1 Classificação dos frameworks

Independentemente do escopo, [FAY99] classifica os frameworks pelas técnicas utilizadas para estendê-los, conforme segue:

- **Caixa-branca**: são extensíveis através da utilização de herança. Para estender frameworks caixa-branca é necessário que os usuários do framework tenham um profundo conhecimento de sua estrutura interna;
- **Caixa-preta**: são extensíveis pela definição de interfaces para componentes que podem ser conectados no framework através da composição de objeto. Os objetos compostos não revelam detalhes internos de implementação, semelhantemente a uma "caixa-preta". São geralmente mais fáceis de usar e estender do que os caixa-branca, porém são mais difíceis de serem desenvolvidos, já que o desenvolvedor do framework deve prever uma maior variação das funcionalidades a serem implementadas na aplicação que usa o framework;
- **Caixa-cinza**: é uma técnica híbrida que torna o framework estensível tanto através de herança de classe como pela composição de objetos.

Um framework caixa-branca é mais fácil de construir, mais flexível, porém mais difícil de usar. Já o framework caixa-preta é mais fácil de estender para uma nova aplicação, porém é mais difícil de construir. Por isso, a maioria dos framework vão provavelmente estar entre esses dois extremos, como um framework caixa-cinza.

### 3.2 Desenvolvendo um framework

Segundo [EXT03], um framework é concebido como qualquer outro projeto de software que enfoque uma solução para um determinado negócio. A principal diferença é que enquanto um software normal é resultado da necessidade de um negócio individual e específico, o framework é resultado de uma necessidade que pode ser observada repetidamente em vários outros projetos.

Isto quer dizer que o objetivo principal ao se projetar um framework é que este seja utilizado mais de uma vez, em situações diferentes, permitindo que futuras aplicações sejam desenvolvidas muito mais facilmente.

É por isso que o framework deve apresentar flexibilidade, extensibilidade e facilidade para configuração.

Muitas vezes acontecem situações onde o usuário emprega o framework para uma determinada tarefa a qual o desenvolvedor do framework não tinha pensado antes.

O framework visa explorar as implementações que são necessárias em vários e diferentes sistemas. Quando isso ocorre há a conseqüência do surgimento de um padrão, que deve ser identificado como uma excelente oportunidade para o reuso.

Identificar pontos reusáveis é fundamental para a criação do processo de desenvolvimento da solução que poderá ser reutilizada, que neste caso é o framework.

### 3.3 Vantagens do uso de frameworks

As principais vantagens da utilização de frameworks são [FAY99]:

**Modularidade**: através do encapsulamento dos detalhes de implementação e de interfaces estáveis, os *frameworks* possibilitam maior modularidade, melhorando a qualidade do produto final;

**Reusabilidade**: as interfaces estáveis providas pelos frameworks facilitam a reutilização de componentes genéricos em novas aplicações;

**Capacidade de extensão**: *frameworks* aumentam a capacidade de extensão pelo fornecimento explícito de métodos *hook*, que permitem às aplicações estenderem suas interfaces estáveis;

**Inversão de controle**: possibilita implementar o fluxo de controle da aplicação. Segundo o Princípio de Hollywood, que diz *"Don't call us, we'll call you"*, é o framework que deve chamar a aplicação, e não o contrário.

## 4. Tecnologia Utilizada

### 4.1 Capicom

CAPICOM é um ActiveX da Microsoft que provê uma interface COM - *Component Object Model* - com a biblioteca de criptografia CryptoAPI, também da Microsoft. Ela faz a interface com algumas funções da CryptoAPI que permitem que os desenvolvedores adicionem facilmente os recursos de assinatura digital e funcionalidades de cifragem de dados em suas aplicações.

Por usar COM, os programadores podem acessar essas funcionalidades em um grande número de ambientes de programação como, por exemplo, Microsoft Visual Basic, VBScript, JavaScript, ASP, Microsoft JScript, C++, Delphi e outros.

Também por ser um pacote ActiveX, a CAPICOM permite que os desenvolvedores escrevam aplicações para Web.

A CAPICOM pode ser usada para executar as seguintes tarefas:

- Criar assinaturas digitais em dados com um *smart card* ou uma chave;
- Verificar assinaturas digitais;
- Exibir graficamente informações de certificados digitais;
- Obter informações e propriedades de um certificado digital como, por exemplo, nome do sujeito e data de expiração;
- Incluir e excluir um certificado digital do repositório de certificados;
- Cifrar e decifrar dados com uma senha (Criptografia simétrica);
- Cifrar e decifrar dados utilizando chaves públicas e certificados digitais (Criptografia assimétrica).

Infelizmente, a CAPICOM somente é suportada pela plataforma Windows (95, 98, ME, 2000, XP, 2003), com navegador Internet Explorer versão 5 ou superior.

Para uma aplicação que usa a CAPICOM funcionar corretamente o usuário terá que obrigatoriamente tê-la instalada na máquina. É possível fazer o *download* gratuitamente da biblioteca CAPICOM diretamente do site do MSDN.

## 5. Conclusão

O INAFF é um framework simples, porém muito robusto, que têm como principal funcionalidade automatizar a geração de código para criação e validação de assinaturas digitais.

O resultado final do projeto atende os objetivos traçados no início, na etapa de planejamento, pois a ferramenta é fácil de ser utilizada, e com poucas alterações permite que o programador tenha implementado, em sua aplicação Web, os mecanismos de assinaturas digitais para prover segurança na transação de dados através desta aplicação.

Como trabalhos futuros, poderão ser cogitadas questões como:

- Desenvolver uma IDE para criar aplicações que utilizam o framework;
- Gerar códigos para as mais diferentes linguagens de programações existentes hoje em dia para desenvolver aplicações para Web, de modo com que a aplicação final possa ser executada em várias plataformas diferentes, a gosto do desenvolvedor usuário do *framework*;
- Verificação de CRL e validação de certificado antes que a assinatura seja criada.

## Referências

[BUR02] BURNETT, S.; PAINE, S. Criptografia e segurança - O guia oficial RSA. Rio de Janeiro: Editora Campus, 2002.

[CAR01] BALPARDA, D. Segurança de dados com criptografia - Métodos e algoritmos. 2 ed. Rio de Janeiro: Editora Book Express, 2001.

[EXT03] EXTERKOETTER, F. Blendwork: Framework Orientado a Objetos para Desenvolvimento Rápido de Aplicações Comerciais Cliente/Servidor. Florianópolis: Universidade Federal de Santa Catarina, Maio, 2003. Dissertação de Mestrado.

[FAY99] FAYAD, M.; SCHMIDT, D.; JOHNSON, R. Building Application Frameworks: Object-Oriented Foundations of Framework Design. New York, NY: John Wiley and Sons, 1999.

[JOH93] JOHNSON, R. How to Design Frameworks, Tutorial Notes, OOPSLA 93, Washington, 1993. Disponível em http://www.cse.msu.edu/~cse870/Materials/Frameworks/how-to-design-fw-tutorial.ps. Acesso em: Maio de 2004.

[MEN96] MENEZES, A.; OORSCHOT, P.; VANSTONE, S. Handbook of applied cryptography. CRC Press, 1996.

[SCH95] SCHNEIER, B. Applied cryptography - Protocols, algorithms and sourc e code in C. 2 ed. Editora John Wiley & Sons, 1995.

[SIL00] SILVA, R. Suporte ao Desenvolvimento e Uso de Frameworks e Componentes. 262 f. Teste (Doutorado em Ciência da Computação) – Programa de Pós-graduação em Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre. 2000.

[STA99] STALLINGS, W. Cryptography and Network Security: Principles and Practice. 1999.

[ZIM98]ZIMMERMANN, P. An Introduction to Cryptography. PGP: Pretty Good Privacy Documents. 1998.