Universidade da Beira Interior

Engenharia Informática

Assinaturas Digitais e TimeStamps
Nº 52

Projecto Elaborado Por:
Ricardo Miguel Lopes de Andrade Simões

Orientador:
Professor Doutor Paul Andrew Crocker

Covilhã - 2012

Agradecimentos

Durante os meus estudos na Licenciatura em Engenharia Informática sem dúvida que este foi o meu maior desafio, e sem a ajuda de diversas pessoas, dificilmente conseguiria obter os resultados finais que atingi.

Em primeiro lugar, desejo agrader sem dúvida à minha mãe, Fátima Lopes, que ao longo destes anos sempre me deu energia para continuar, mesmo quando as coisas corriam mal não só na universidade como fora dela, e por isso mãe agradeço-te do fundo do meu coração.

Agradeço também à minha namorada, Tânia Vieira, que sem dúvida a seguir à minha mãe foi a pessoa que mais lutou por mim, ajudando me sempre que possível.

Outra pessoa que eu estou especialmente grato é, sem dúvida, o meu orientador de projecto, Doutor Paul Andrew Crocker, pela sua impressionante capacidade de orientação e ajuda na obtenção deste projecto.

Um agradecimento especial aos meus amigos, Ana Ventura, Guilherme Carvalho, João Lanzinha, Pedro Sequeira , Miguel Antunes e Tiago Pereira, pelo tempo que tiveram comigo para contribuír para um melhor projecto.

Um obrigado também para o resto da minha família que sempre me apoiou e para os meus restantes colegas de curso que sempre me foram ajudando com várias dicas.

Por último, e não menos importante, agradeço a todas as pessoas envolvidas no (rel)ease, onde pude encontrar pessoas fantásticas, sempre disponíveis a ajudar criando também um ambiente de trabalho agradável.

A todos voçês um enorme obrigado,

Muito Obrigado

Ricardo Miguel Lopes de Andrade Simões

Conteúdo

Lista de Figuras

Acrónimos e Siglas

API: Application Programming Interface.

BD: Base de Dados.

BI: Bilhete de Identidade.

CC: Cartão do Cidadão

CSP: Cryptographic Service Provider

C#: C-Sharp

DB: DataBase

DS: Digital Signature

ECC: European Citizen Card

ISO: International Organization for Standarization

PIN: Personnel Identification Number

PKCS: Public Key Criptography Standards

PKI: Public-Key Infrastructure

RFC: Request for Comments

SC: SmartCard

SDK: Software Development Kit

SHA: Secure Hash Algorith

SQL: Structured Query Language

SVN: Subversion

TS: TimeStamp

TSA: TimeStamping Authority

TTP: Trusted Third Party

TTS: Trusted TimeStamping

UBI: Universidade da Beira Interior

VS: Visual Studio

Capítulo 1

Introdução

Este relatório foi elaborado no âmbito da cadeira de Projecto do 3º ano da Licenciatura em Engenharia Informática da Beira Interior, tendo como objectivo descrever todo o trabalho realizado sobre o projecto de Assinaturas Digitais e TimeStamps.

Com a evolução que tem ocorrido no mundo da segurança informática, hoje em dia torna-se essencial aumentar a segurança das pessoas no mundo da informática e para tal a autenticação é um requisito essencial. Como tal, a introdução das assinaturas digitais é um meio muito importante para a segurança das pessoas nesta área, quer isto dizer, que com o uso das assinaturas digitais nós vamos permitir que uma pessoa possa autenticar um documento, provando que aquele documento é seu. E é aqui que vai entrar o CC, visto este trazer um PIN de Assinatura Digital.

1.1 Motivação Pessoal

Actualmente, são cada vez mais necessários Engenheiros Informáticos no mercado de trabalho e como tal, estes devem ser pessoas capazes de construir soluções adequadas à resolução de certos problemas, com a ajuda de todas as tecnologias já existentes. E é neste aspecto que podemos introduzir o CC.

As oportunidades que podem surgir com a introdução do CC são enormes, o que pode ajudar na relação entre as entidades privadas ou mesmo

o Estado. O surgimento do CC, no fundo pretende ajudar as pessoa, como por exemplo a burocracia. Todos nós sabemos que em Portugal para se fazer qualquer coisa é sempre preciso um documento. Assim, com o CC podemos começar a deixar de usar tantos documentos e começar a fazer tudo ao nível da Informática.

Ciente desta realidade, é com enorme prazer e sem dúvida motivador poder fazer parte desta mudança que o CC pretende trazer e poder contribuir para a Universidade da Beira Interior seja uma instituição ainda mais respeitada e especialmente nesta àrea relacionada com todos os cidadãos de Portugal.

1.2 Objectivos do Projecto

Com este projecto, pretende-se implementar uma aplicação relacionada com o CC e que possa apresentar uma interface agradável e de fácil utilização .

Assim, este projecto tem como objectivos:

- Construir uma aplicação para criar assinaturas digitais e timestamps.
- A aplicação terá de guardar e gerir estes tokens de segurança, usando uma base de dados de sqlite.

1.3 Organização do Relatório

Este relatório encontra-se dividido em 4 capítulos:

- **Capítulo 1** - Introdução ao trabalho realizado, bem como a motivação pessoal e objectivos do projecto.
- **Capítulo 2** -Abordagem ao desenvolvimento do projecto, que consiste em descrever todas as ferramentas necessárias para a sua elaboração, toda a pesquisa cientifica realizada, tal como as bibliotecas utilizadas na elaboraçao do projecto.

- **Capítulo 3** - Descrição da aplicação e ilustração da interface.
- **Capítulo 4** - Conclusões do trabalho desenvolvido e trabalho futuro.

Capítulo 2

Ferramentas

2.1 Ferramentas Externas

A escolha das ferramentas a utilizar ao longo do projecto é sem dúvida importante. Assim, nesta secção será feita uma descrição das ferramentas usadas para o desenvolvimento deste projecto.

2.1.1 Visual Studio 2010

O Visual Studio é uma ferramenta da Microsoft para desenvolvimento de software especialmente dedicado ao .NET Framework e a linguagens como Visual Basic(VB), C, C++, C# e J#. As linguagens com maior frequência são o VB.NET(Visual Basic.NET) e o C#. O VS 2010 foi lançado em Abril de 2010 com o objectivo de ser a IDE mais completa do mercado. Esta ferramenta foi usada para o código implementado em C#.

2.1.2 SQLite

O SQLite é uma biblioteca em linguagem C que implementa uma BD SQL embutido. Programas que usam a biblioteca SQLite podem ter acesso a BD SQL sem executar um processo SGBD seprado. O SQLite não é uma biblioteca cliente usada para conectar com um grande servidor de BD, mas sim com o próprio servidor. A biblioteca SQLite lê e escreve directamente para e do arquvo da BD no disco.

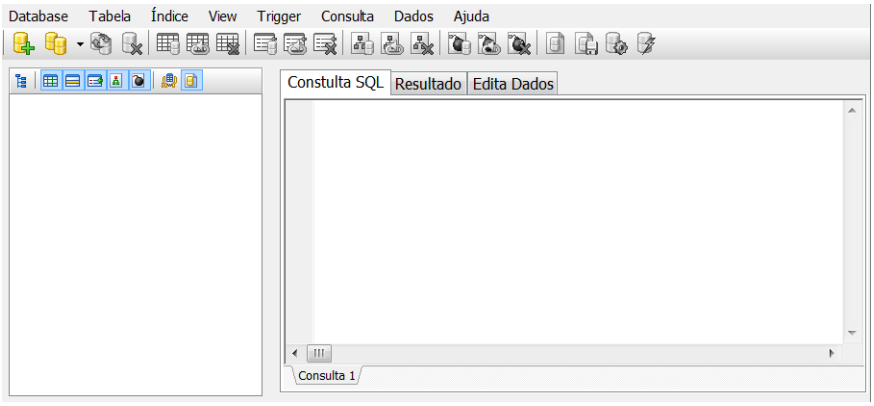

Figura 2.1: *SQLite*

2.1.3 Dropbox

A Dropbox é um serviço para armazenamento de arquivos. É baseado no conceito de "computação em nuvem".

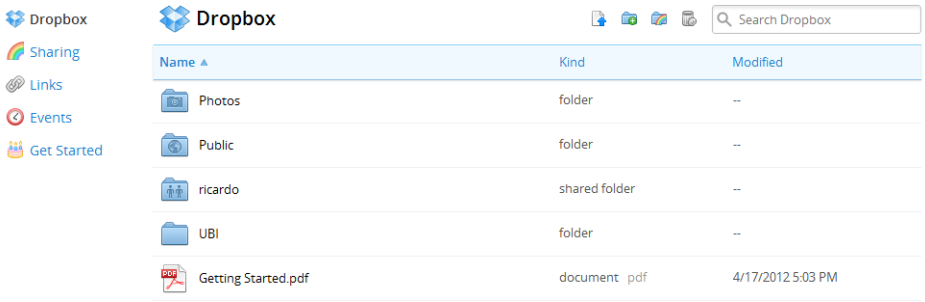

Figura 2.2: *Dropbox*

Computação em nuvem

Refere-se à utilização da memória e das capacidades de armazenamento e cálculo de computadores e servidores compartilhados e interligados por meio da internet. O armazenamento de dados é feito em serviçoes que poderão ser acessados de qualquer lugar do mundo, a qualquer hora, sem necessidade da instalação de programas.

2.1.4 VisualSVN

O VisualSVN nao é nada mais do que uma ferramenta visual para instalação e manutenção simplificada do Subversion para o Windows, implementado como um pacote de extensão para o Visual Studio.

2.2 Criptografia

A criptografica é o estudo dos princípios e técnicas pelas quais a informação pode ser transformada das sua forma original para outra ilegível, de forma a que possa ser conhecida apenas pelo seu destinatário. É uma ramo da Matemática, parte da Criptografia, responsável pela encriptação e desencriptação de informação. O seu uso é visível no armazenamento de informações sensíveis ou na transmissão de dados através de redes inseguras.

2.2.1 Encriptação e Desencriptação

Em criptografia, a informação que pode ser lida sem qualquer tipo de transformação é designada por texto original. Como tal, a encriptação não é mais do que o processo de transformar essa informação, usando um algoritmo, de modo a impossibilitar a sua leitura a todos, excepto aqueles que possuam a chave referente a essa encriptação. A desencriptação é o processo inverso, isto é, obter o texto original a partir de um texto encriptado

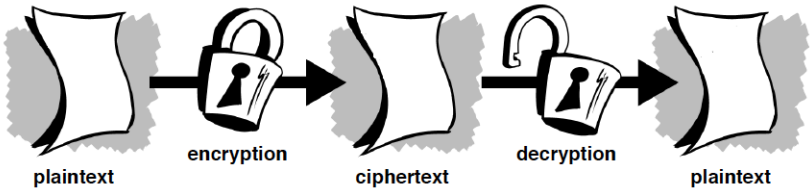

Figura 2.3: *Encriptar e Desencriptar*

2.2.2 Assinaturas Digitais

A Assinatura Digital é um método de autenticação de informação digital análoga à assinatura física em papel. A utilização da Assinatura Digital providencia a prova de que uma mensagem veio do emissor. Para tal, uma assinatura deve ter as seguintes propriedades:

- Autenticação - o receptor deve poder confirmar a assinatura do emissor;

- Integridade - a assinatura não pode ser falsificável;

- Não repúdio - o emissor não pode negar a sua autenticidade.

A assinatura digital previne que alguém possa alterar o conteúdo da informação, quer os dados do verdadeiro emissor, pois qualquer mudança que ocorra, será detectada quando essa assinatura for verificada.

Na figura 2.4 podemos ver como funciona o processo de Assinatura Digital e a sua verificação.

Tal como se pode ver na figura, o primeiro passo da assinatura é a criação de uma sequência de caracteres(hash) através de um algorimo de disperão(*hash functions*). De seguida, essa sequência é então encriptada através de um algoritmo de assinatura com a chave privada do autora da mensagem sendo depois o resultado desta encriptação anexada ao documento original, dando assim origem à mensagem assinada digitalmente.

A verificação da assinatura é feita em dois processos distintos, onde um deles passa por criar um nova sequência de caracteres com o mesmo algoritmo de dispersão a partir do texto original. O outro processo é onde se usa um algorimo de desencriptação(inverso ao de encriptação), que através da chave pública do autor da mensagem, permite recriar a sequência de caracteres gerada pela função de dispersao na fase de assinatura. No fim dos dois processos é feita a compração entre as duas cadeias de caracteres e se ambas forem iguais a verificação é considerada válida, senão é uma assinatura inválida.

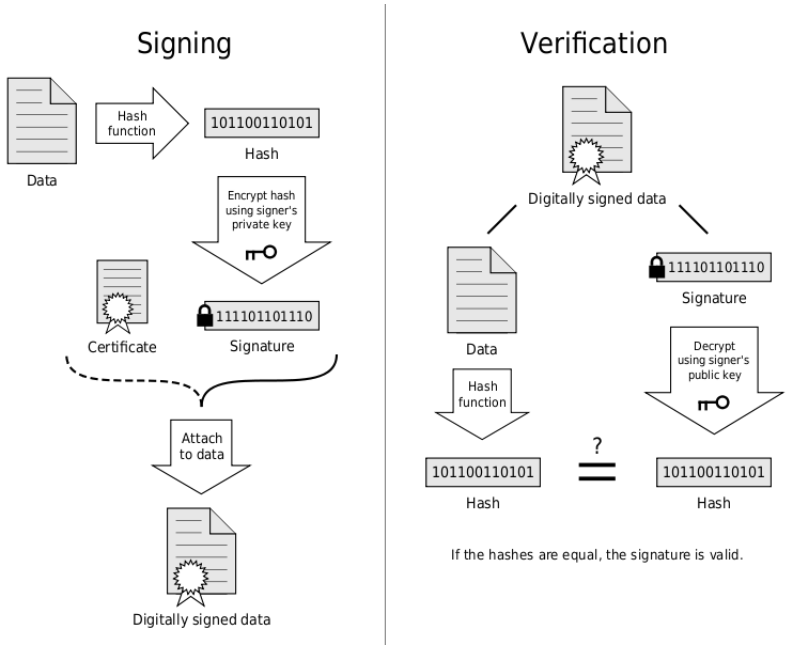

Figura 2.4: *Assinatura Digital e Verificação*

De referir que para um Assinatura Digital ser considerada válida há que ter em conta também a validade do certificado digital. É importante voltar a mencionar que uma pequena alteração no corpo da mensagem invalida o processo de verificação da Assinatura Digital.

2.2.3 Timestamp

Um *TimeStamp* é uma cadeira de caracteres denotando a hora ou data que certo evento ocorreu. Segundo o standard RFC 3161, um *TimeStamp* de confiança é um *TimeStamp* emitido por uma terceira parte de confiança, actuando como uma Autoridade *TimeStamping*(TSA). Múltiplos TSA's podem ser usados para aumentar a confiança e diminuir a vulnerabilidade. No fundo um *TimeStamp* usa-se para provar a existência de certos dados, antes de um determinado ponto.

O processo de inserção de um *TimeStamp* válido, conhecido por *trusted time-stamping*, está exemplificado na figura 2.5. Como tal para se criar um *TimeStamp* em primeiro lugar calcula-se o hash do ficheiro, sendo depois esse hash enviado para o TSA. Este concatena o *TimeStamp* com o hash e calcula então um novo hash dessa concatenação sendo depois este hash assinado com a chave privada do TSA. Finalmente o hash assinado e o *TimeStamp* são enviados de volta ao seu solicitante, ou seja este vai guarda o ficheiro com o *TimeStamp* mais o ficheiro original. Visto que os dados originais não podem ser calculdados a partir do hash o TSA nunca vai descobrir os dados originais, o que permite utilizar este método como um método de confiança.

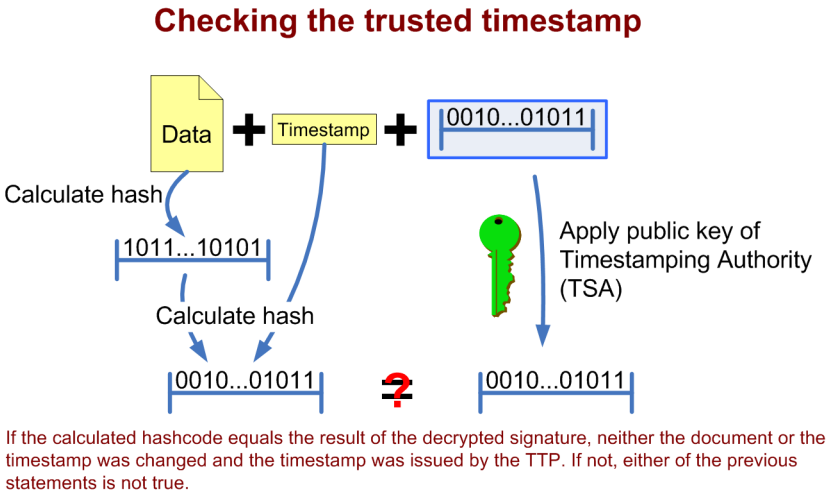

Figura 2.5: *Criação de um TimeStamp*

Relativamente ao processo de verificação, ilustrado na figura 2.6, como primeiro ponto temos de calcular o hash dos dados originais, anexando-se depois o *TimeStamp* enviado pelo TSA ao hash, calculando-se assim um novo hash a partir desta concatenação. Depois é necessário validar a Assinatura Digital do TSA, usando a sua respectiva chave pública, para se confirmar se realmente o hash fornecido pelo TSA foi de facto assinado com a sua chave privada através da verificação da Assinatura Digital. Por fim, temos assim dois hash's, fazendo então uma comparação entre ambos. Se forem iguais então prova-se que o *TimeStamp*, que a mensagem

esta inalterada e que foi enviada pelo TSA. Senão forem iguais, então cada *TimeStamp* foi alterada ou então este não voi enviado pelo TSA.

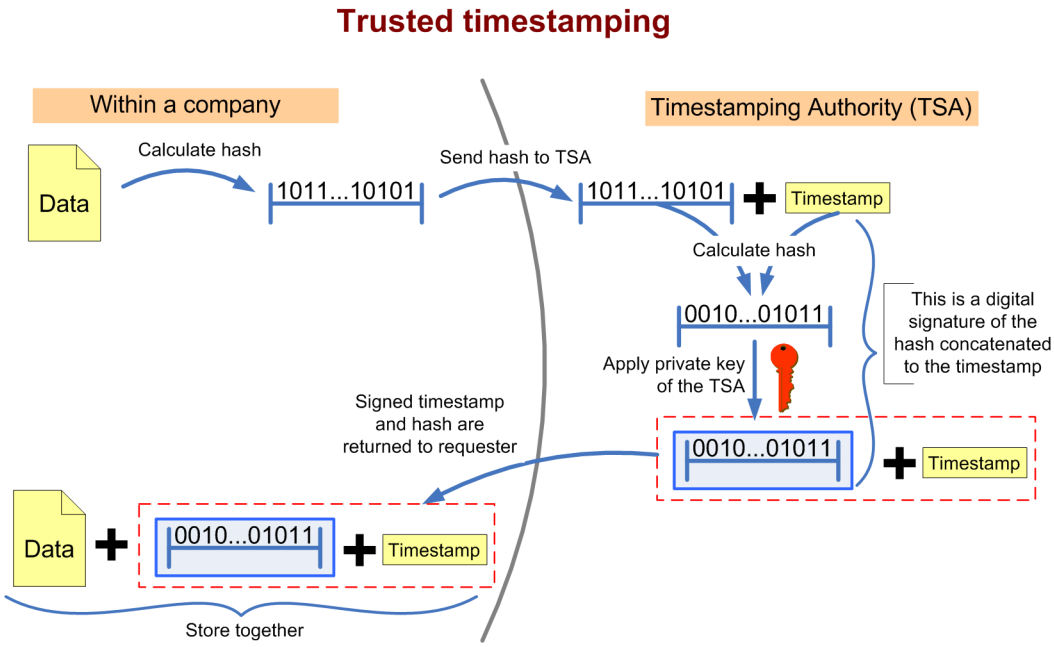

Figura 2.6: *Verificação de um TimeStamp*

2.3 Cartão do Cidadão

O Cartão do Cidadão(CC) é o documento de identifcação dos cidadãos portugueses. Este foi desenvolvido durante o Governo do ex Primeiro-Ministro José Sócrates e começou a ser emitido em meados de 2006-2007. O CC substitui não só o bilhete de identidade, como também o cartão de beneficiário da Segurança Social, o cartão de utente do Serviço Nacional de Saúde, o cartão de contribuinte e o cartão de eleitor, sendo que o CC se distingue destes outros cartões devido às suas capacidades electrónicas associadas.

De referir que neste projecto do CC várias entidades de renome internacional estiveram envolvidas, destacando a Microsoft, Siemes, Multicert entreoutras entidades. É importante perceber quais os pontos fortes e fracos do CC, não só na área tecnológica como tambem à legislação associada ao CC.

2.3.1 Objectivos

O objectivo principal deste documento foi reduzir o número de cartões de identificação necessários para cada cidadão se apresentar perante as instituições do estado. Como documento de identificação, o CC permite não só a identificação visual e presencial quando requerida, como tambem permite a sua identificação electrónica e para isto ser possível foi necessário equipar o CC com um chip capaz de guardar dados pessoais cifrados, que permite garantir privacidade desses dados.

Sendo este um projecto de modernização, o CC deve garantir ao cidadão uma maior segurança na identificação. Este deve tambem facilitar a vida do cidadão devido à necessidade de menos cartões e pela melhoria da relação entre este e os serviços públicos a nível de qualidade e rapidez de execução de processos. Ainda se deve falar do CC ao nível tecnológico, pois este sendo moderno deve procurar acompanhar os tempos actuais e para tal este deve promover a sua utilização nos serviços electrónicos, como por exemplo a Assinatura Digital de documentos.

Por fim referir que relativamente à recolha e tratamento de dados e à informação transmitida nas ligações aos diferentes serviços, o CC não

poderá conter dados sobre cada organismo, sendo que cada organismo deve armazenar essa mesma informação na sua respectiva BD, para se evitar correr o risco de, por exemplo, ir-mos às finanças e saberem o nosso estado de saúde.

2.3.2 Características

Podemos dividir o CC em três características distintas:

- Físico
- Visual
- Electrónico

Ao nível físico sabemos que o CC tem um formato Smart Card substituindo vários documentos. Do ponto de vista visual, como se pode ver na figura 2.7, este mostra a informação que é apresentada no CC. Por último, o aspecto electrónico, em que o CC possui um chip de contacto com dois certificados digitais, um para autenticação e outro para Assinaturas Digitais. A informação visual também se encontra no interior do chip, bem como a morada do cidadão.

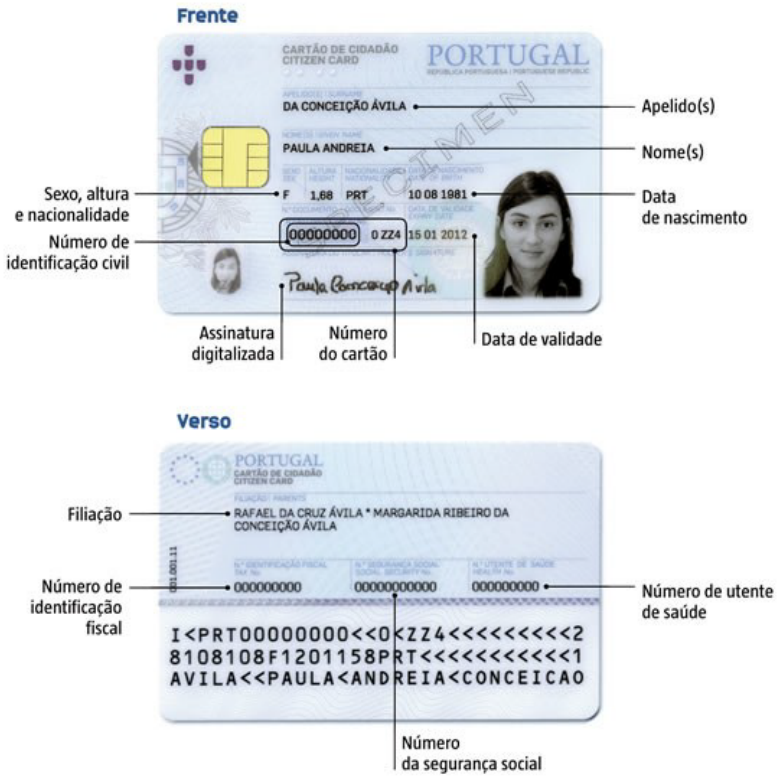

Figura 2.7: *Cartão do Cidadão*

2.4 C#

O C# é uma linguagem de programação orientada a objectos, fortemente tipada, foi desenvolvida pela Microsoft como parte da plataforma .NET. A sua sintaxe orientada a objectos foi baseada no C++ embora inclua muitas influências de outras linguagens de programação, como Java.

A sintaxe C# é altamente expressiva, mas também é simples e fácil de aprender. Qualquer pessoa que saiba programar em C, C++ ou Java vai perceber rápidamente como trabalhar com C#, devido às suas muitas semelhanças. O C# simplifica muitas das complexidades do C++ e oferece outros recursos poderosos, como por exemplo as expressões lambda e acesso directo à memória, que não são encontrados em Java.

Em c# não existe herança múltipla, quer isto dizer, que cada classe só pode herdar apenas uma outra classe e não mais do que uma, no entanto e possivel simular herança múltipla utilizando interfaces. Assim, através da herança reduz-se código atraves da sua reutilização.

2.5 DLL's

DLL, proveniente do inglês *Dynamic-link library*, é a implementação feita pela Microsoft para o conceito de bibliotecas compartilhadas nos sistemas operacionais Microsoft Windows e OS/2. Os formatos de arquivos para DLL são os mesmos dos arquivos executáveis para Windows. Assim como executáveis, as DLL podem conter códigos, dados, e recursos em qualquer combinação.

Após uma breve descriçao de DLL referir que para a realização deste projecto foi necessário o uso especial de algumas DLL's.

2.5.1 pteidlib_dotnet

A pteidlib_dotnet é a biblioteca oficial do CC que permite ver e gerir a informação no cartão. Como tal é possível ler e mostrar a informação sobre o cidadão e morada. Lê também os certificados do governo e do cidadão, permite a gestão de códigos PIN entre outras funções.

2.5.2 CSJ2K

Esta DLL é a DLL que nos vai permitir ler a fotografia do cidadão, visto esta encontrar-se no formato JPEG2000.

2.5.3 BouncyCastle.Crypto

DLL que pertence ao *Bouncy Castle* para a plataforma .NET usada em criptografia. Neste projecto vai permitir efectuar as operações necessárias com o *Timestamp*.

2.5.4 System.Data.SQLite

É o motor de BD do SQLite e um completo provedor do ADO.NET 2.0/3.5m tudo em um único conjunto que nos vai então permitir fazer a ligação à BD.

Capítulo 3

Middleware

Para começar podemos dividir o projecto em três partes:

- 1ª - Criação do *middleware* que permite ler os dados do CC;
- 2ª - Estudo sobre Assinaturas Digitais e *TimeStamps* e desenvolvimento destes dois assuntos relativamente ao*middleware*;
- 3ª - Conhecimento do funcionamento do SQLite em C# e introdução na nossa aplicação de modo a gerir os tokens relacionados com as Assinaturas Digitais e *TimeStamps*.

3.1 Utilidade do Cartão do Cidadão

Apesar das alterações a nível electrónico, o CC continua a ser o principal meio de identificação perante as entidades públicas, ou seja, tendo o CC o mesmo valor legal que o antigo BI ou cartão de contribuínte, dependendo da entidade interessada. Sendo assim umas das principais inovações foi sem dúvida as potencialidades digitais, como a autenticação. Outro ponto forte foi sem dúvida a introdução do certificado digital que vai permitir aos cidadãos selarem um determinado documento com a sua respectiva assinatura digital.

Resumidamente a introdução da parte electrónica no CC, vai sem dúvida ser muito útil não só ao nível da autenticação como das assinaturas digitais.

3.1.1 Autenticação

Num mundo onde cada vez mais as pessoas têm de gerir as diversas chaves que se tem espalhadas pela Internet, tem vindo a ser um pesadelo, pois aliado a um fenómeno de insegurança cada vez mais acentuado ao nível da informática, obriga as pessoas a terem várias chaves para variados sites. Com a introdução do CC, a sua autenticação é sem dúvida um passo enorme para re-estabelecer a confiança dos cidadãos ao nível das informações que têm na Internet.

3.1.2 Assinatura de Documentos

Se tivermos, por exemplo, em conta o ambiente académico onde existe uma constante troca de informações, é importante ter em conta se essas informações sao de confianças ou não. Imaginemos que os académicos recebem as notas finais da cadeira de Projecto, é importante que estes tenham a certeza que as notas foram entregues pelo regente da cadeira ou por alguém com confiança para isso. E é aqui que entram as assinaturas digitais, pois o regente da cadeira ao assinar o documento com as respectivas notas, nunca vai poder negar que aquelas foram as notas finais.

3.2 Aplicação

Primeiro foi necessário a criação do *middleware* para a leitura dos dados do CC, tal como mostra a figura 3.1.

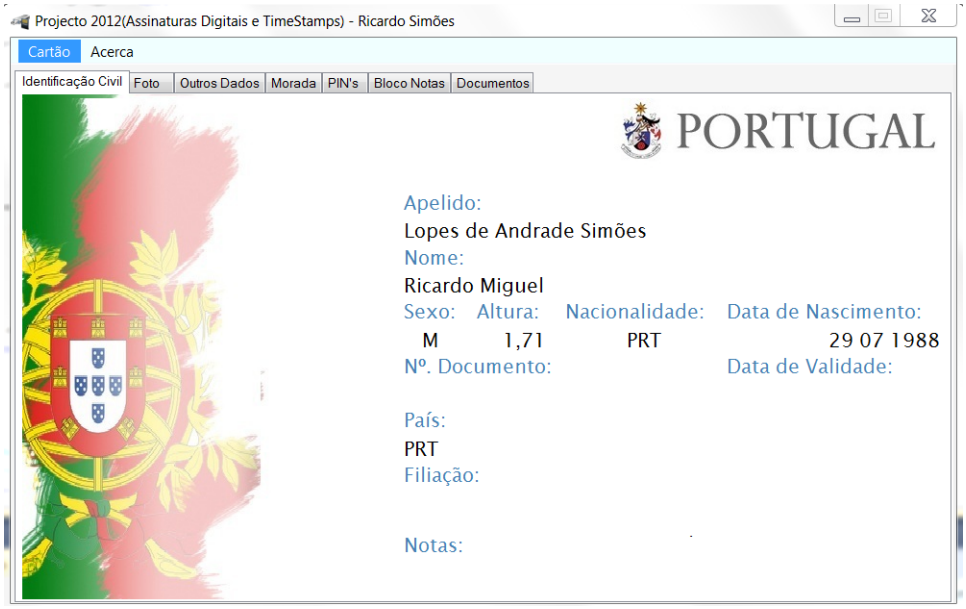

Figura 3.1: *Middleware*

Como já referido anteriormente foi utilizado a linguagem C# para a programção da aplicação, sendo também necessária a DLL pteidlib_dotned, pois sem esta DLL não seria possível aceder aos dados guardados no CC. Esta DLL é disponibilizada pelo Middleware do CC, que é possível encontrar no respectivo site do CC. No fundo a SDK disponibilizada pelo Estado está destinada a organizações que pretendem desenvolver aplicações em que se utilize o CC, sendo o caso deste projecto.

Sendo assim pode-se dizer que a API é organizada em em quatro categorias:

- Função de inicialização;
- Funções de identificação;

- Funções para propósitos gerais de alto nível;
- Função de terminação.

3.2.1 Função de inicialização

Função que inicializa a utilização do toolkit.

```
Pteid.Init("");
```

Figura 3.2: *Inicialização*

3.2.2 Funções de identificação

Funções usadas para obter dados identificativos do cartão(nome, morada, etc).

```
PteidId dadosIdentificacao = Pteid.GetID();
```

Figura 3.3: *Obtenção da Identificação*

3.2.3 Funções para propósitos gerais de alto nível

Funções gerais que são usadas para aplicações que necessitem de efectuar outras acções para além do acesso aos dados identificativos do cidadão.

```
PteidPin pin = (PteidPin)treeView1.SelectedNode.Tag;
try
{

    pin.triesLeft = Pteid.ChangePIN(pin.id, null, null);
    textBox24.Text = "Resta(m) " + pin.triesLeft + " tentativa(s)";
    treeView1.SelectedNode.Tag = pin;
}
```

Figura 3.4: *Alteração do PIN*

3.2.4 Função de terminação

Função que termina a utilização do toolkit.

```
Pteid.Exit(0);
```

Figura 3.5: *Terminação*

3.2.5 Fotografia

Devido ao facto da fotografia estar no formato JPEG2000 foi necessário criar uma uma função própria para fazer a sua respectiva leitura, tal como usar a DLL CSJ2K, embora esteja inserida na secção das funções de identificação, porque para ir buscar a imagem vai-se buscar através das respectivas funções de identificação como ilustrada em cima na figura 3.3.

```
PteidPic picData = Pteid.GetPic();

if (picData != null)
{
    MemoryStream ms = new MemoryStream(picData.picture);

    Bitmap image = (Bitmap)CSJ2K.J2kImage.FromStream(ms);
    System.Drawing.Size size = new System.Drawing.Size(230, 307);
    Bitmap bitmap = new Bitmap(image, size);

    pictureBox1.Image = bitmap;

}
```

Figura 3.6: *Fotografia*

3.3 Assinaturas Digitais e TimeStamps

Nesta aplicação tanto as assinaturas digitais como os *timestamps* podem-se encontrar na aba dos Documentos, tal como ilustra a figura 3.7.

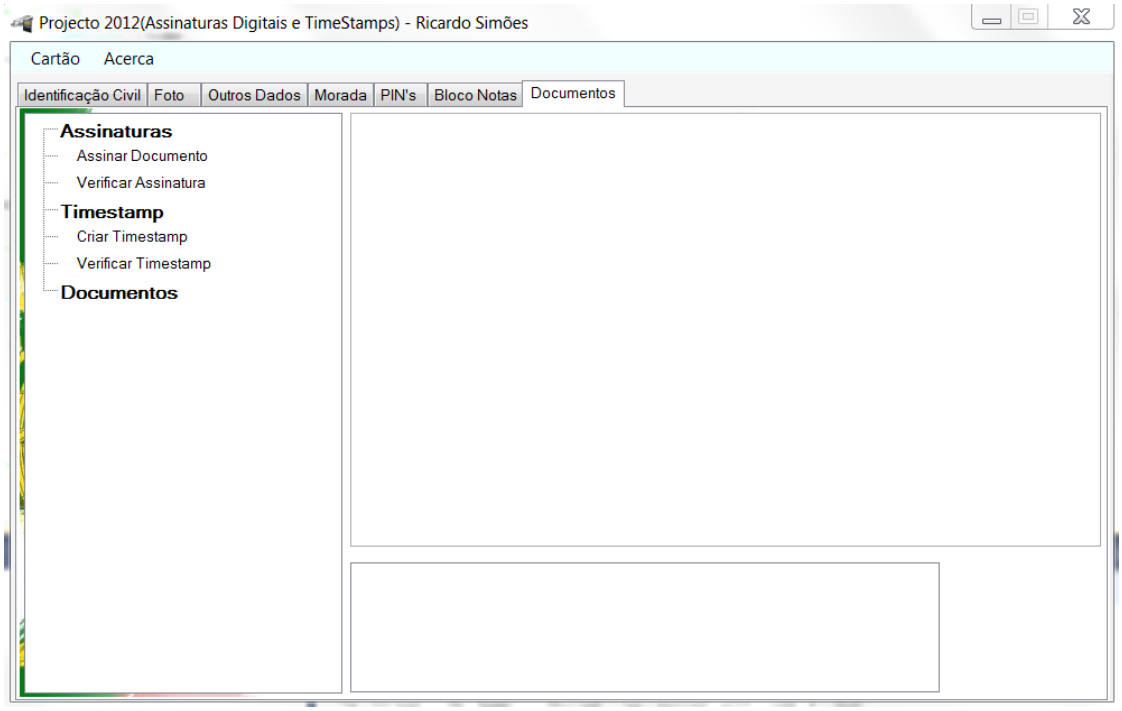

Figura 3.7: *Aba referente a Assinaturas Digitais e TimeStamps*

3.3.1 Assinaturas Digitais

Tal como já referido em capítulos anteriores as assinaturas digitais não são mais do que uma prova de que um documento pertence a certo cidadão, significa isto que uma assinatura digital não pode ser alvo de não repúdio, é integra e autentica, é ÚNICA. Nesta secção vamos então estruturar através de dois pontos, um que é a parte da assinatura e outro que é a parte da verificação.

Assinar

Quanto à assinatura de referir que esta já tem à partida a extensão *.sig* para todos os documentos que forem assinados. O funcionamente é muito simples. Tal como ilustrado na figura 3.7 nós podemos verificar que existem um nó referente às assinaturas, sendo que quando o utilizador clicar onde diz "Assinar Documento"irá abrir uma nova janela para a pessoa seleccionar o ficheiro que pretende assinar, sendo que a directoria por defeito neste projecto é os Documentos. Após a escolha do ficheiro irá aparecer um caixa que irá pedir para se inserir o PIN de assinatura digital. Após isto o programa irá então pedir o nome do novo ficheiro, guardando de seguida esse novo ficheiro.

Para tal é necessário o cálculo do hash SHA1 para os dados de entrada. Após isto é necessário ir buscar o certificado do cartão inserido que está na função CC_Certificados(); através do certificado X509Certificate2. Também é necessário correr criptografia assimétrica através do algoritmos RSA fornecido pelo provedor de serviços criptográficos(CSP).

```
try
{
    if (myString != "")
    {

        SHA1Managed sha1 = new SHA1Managed();
        byte[] data = Encoding.Unicode.GetBytes(myString);
        byte[] hash = sha1.ComputeHash(data);

        X509Certificate2 cc = CC_Certificados();
        RSACryptoServiceProvider csp = (RSACryptoServiceProvider)cc.PrivateKey;
        byte[] assinatura = csp.SignHash(hash, CryptoConfig.MapNameToOID("SHA1"));

        SaveFileDialog saveFileDialog1 = new SaveFileDialog();
        saveFileDialog1.FileName = Path.ChangeExtension(filename, ".sig");
        saveFileDialog1.Filter = "sig files (*.sig)|*.sig";
        saveFileDialog1.FilterIndex = 2;
        saveFileDialog1.RestoreDirectory = true;

        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
            //Escreve assinatura num ficheiro
            FileStream file = new FileStream(saveFileDialog1.FileName, FileMode.Create);
            BinaryWriter br = new BinaryWriter(file);
            br.Write(assinatura);
            br.Close();

            MessageBox.Show("Ficheiro assinado com sucesso", "Assinar Ficheiro com CC", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
}
```

Figura 3.8: *Função assinar documento*

Verificar

Tal como para assinar, quando fazemos verificar irá abrir uma janela para escolhermos o ficheiro original e de seguida o ficheiro com a assinatura. Após a escolhe dos ficheiros irá ser pedido ao utilizar que escolha um dos certificados que se encontram no KeyStore do Windows, pois nós ao inserirmos uma assinatura o certificado será guardado no computador onde estiver a ser usada a aplicação.

Tal como para assinar, na verificação é necessário o cálculo do SHA1, o uso do certificado X509Certificate2 e o os serviços RSA fornecidos pelo CSP.

```
X509Certificate2 cc = StoreCertificados(listCertificates.Items[listCertificates.SelectedIndex].ToString());
SHA1Managed sha1 = new SHA1Managed();
RSACryptoServiceProvider csp = new RSACryptoServiceProvider();
csp = (RSACryptoServiceProvider)cc.PublicKey.Key;

byte[] data = Encoding.Unicode.GetBytes(novaString);
byte[] hash = sha1.ComputeHash(data);
```

Figura 3.9: *Função verificar assinatura*

De referir também que para se mostrar os certificados foi necessário recorrer a uma função que neste caso foi definida como *MostraStoreCertificados()*.

3.3.2 TimeStamps

Os *TimeStamps* não são mais do que assinaturas só que com a hora e a data a que tal acontecimento ocorreu, ou seja ninguém pode negar que certos dados não existiam até ao ponto em que foi criado o *TimeStamp*.

Tal com com as assinaturas, neste projecto subdividimos o *TimeStamp* em dois prontos, o criar e o verificar.

Criar

A criação do *TimeStamp* é em tudo idêntico à assinatura, isto em termos de funcionalidade com a aplicação, à excepção que não pede o PIN de assinatura. O *TimeStamp* também tem já uma extensão definida para os ficheiros que é *.tsr*.

Como já referido na secção 2.2.3 é necessário o cálculo do hash do ficheiro sendo que depois se envia esse hash ao TSA, aguardo por último pela resposta deste.

```
SHA1 sha = SHA1CryptoServiceProvider.Create();
byte[] hash1 = sha.ComputeHash(Encoding.ASCII.GetBytes(file));

TimeStampResponse response = getTsrFromServer2(hash1);
TimeStampToken tk = response.TimeStampToken;
SignerID signer_id = tk.SignerID;
BigInteger cert_serial_number = signer_id.SerialNumber;

SaveFileDialog saveFileDialog1 = new SaveFileDialog();
saveFileDialog1.FileName = Path.ChangeExtension(filename, ".tsr");
saveFileDialog1.Filter = "tsr files (*.tsr)|*.tsr";
saveFileDialog1.FilterIndex = 2;
saveFileDialog1.RestoreDirectory = true;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
    //Escreve assinatura num ficheiro
    FileStream ficheiro = new FileStream(saveFileDialog1.FileName, FileMode.Create);
    writeTsrToFile2(response, filename);

    MessageBox.Show("TimeStamp criada com sucesso", "Assinar Ficheiro com CC", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Figura 3.10: *Função criar TimeStamp*

Verificar

Também já referido na secção 2.2.3 para a verificação do *TimeStamp* será necessário calcular o hash dos dados originais e anexa-lo ao *TimeStamp* enviado pelo TSA, calculando-se depois um novo hash. Após isto é preciso confirmar se o hash fornecido pelo TSA foi de facto assinado com a sua chave privada. Por último verificamos então se os dois hash's são iguais ou não.

Na aplicação o funcionamento é exactamente igual à verificação da assinatura.

```
SHA1 sha = SHA1CryptoServiceProvider.Create();
byte[] hash1 = sha.ComputeHash(Encoding.ASCII.GetBytes(file));

TimeStampResponse response = getTsrFromServer2(hash1);
TimeStampToken tk = response.TimeStampToken;
SignerID signer_id = tk.SignerID;
BigInteger cert_serial_number = signer_id.SerialNumber;

SaveFileDialog saveFileDialog1 = new SaveFileDialog();
saveFileDialog1.FileName = Path.ChangeExtension(filename, ".tsr");
saveFileDialog1.Filter = "tsr files (*.tsr)|*.tsr";
saveFileDialog1.FilterIndex = 2;
saveFileDialog1.RestoreDirectory = true;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
    //Escreve assinatura num ficheiro
    FileStream ficheiro = new FileStream(saveFileDialog1.FileName, FileMode.Create);
    writeTsrToFile2(response, filename);

    MessageBox.Show("TimeStamp criada com sucesso", "Assinar Ficheiro com CC", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Figura 3.11: *Função verificar TimeStamp*

3.4 Base de Dados SQLite

Tal como proposta de trabalho para a realização deste projecto era a gestão dos tokens relacionados com as assinaturas digitais e com os *timestamps* e para isso acontecer seria necessário inserir na nossa base de dados SQLite esses tokens. Como tal foi então necessário criar uma tabela na nossa BD cujo o nome é documentos, tabela essa que se encontra visível na figura que se segue.

```
CREATE TABLE [documentos] (
[ID] INTEGER  NOT NULL PRIMARY KEY AUTOINCREMENT,
[Tipo] NVARCHAR(20)  NOT NULL,
[Documento] NVARCHAR(100)  NOT NULL,
[Ficheiro] NVARCHAR(30)  NOT NULL,
[Tempo] NVARCHAR(30)  NOT NULL
)
```

Figura 3.12: *Tabela documentos da base de dados*

Tal como é visível na figura existem cincos parâmetros, sendo esses o ID que é a nossa chave primária sendo que nunca se pode repetir, o tipo que identifica se é uma assinatura ou um *timestamp*, o documento refere-se à directoria do ficheiro original, sendo que o campo ficheiro indica só o nome do respectivo ficheiro que se cria, ou *.sig* ou *.tsr* e por último é guardado o tempo em que foi inserido na BD.

3.4.1 Inserção na Base de Dados

A inserção na BD é efectuada automaticamente após a criação de uma assinatura ou de um *timestamp* sendo em ambos os casos exactamente igual o modo de inserção. Na figura 3.13 é possível ver como foi feita a inserção na BD.

```
try
{
    //base de dados
    String strConn = @"Data Source=C:\Users\wrt\Documents\Visual Studio 2010\Projects\oooo\proj.s3db";

    SQLiteConnection conn = new SQLiteConnection(strConn);

    conn.Open();

    SQLiteCommand cmd = new SQLiteCommand("Insert into documentos(Tipo, Documento, Ficheiro, Tempo) values (@Tipo, @Documento, @Ficheiro, @Tempo)", conn);

    cmd.Parameters.AddWithValue("@Tipo", assi);
    cmd.Parameters.AddWithValue("@Documento", nome);
    cmd.Parameters.AddWithValue("@Ficheiro", b);
    cmd.Parameters.AddWithValue("@Tempo", DateTime.Now.ToString());

    cmd.ExecuteNonQuery();
    conn.Close();
}
catch
{
    MessageBox.Show("erro na inserção na bd", "inserção na bd - erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

Figura 3.13: *Inserção na Base de Dados*

Ter em conta que o ID não é preciso inserir na tabela, pois este já tem um valor que auto-incrementa. Referir também a facilidade com que se consegue obter a data e hora em C#, sendo únicamente necessário a linha de código *DateTime.Now.ToString()*.

3.4.2 Leitura da Base de Dados

Relativamente à leitura dos dados que se encontram na BD, basta clicar na aba Documentos e depois no nó que também está designado com o nome Documentos. Ao carregar a BD esta irá aparecer, com os dados exactamente iguais à BD, numa *DataGridView*, sendo isto uma de muitas toolbox's disponibilizadas pelo VS.

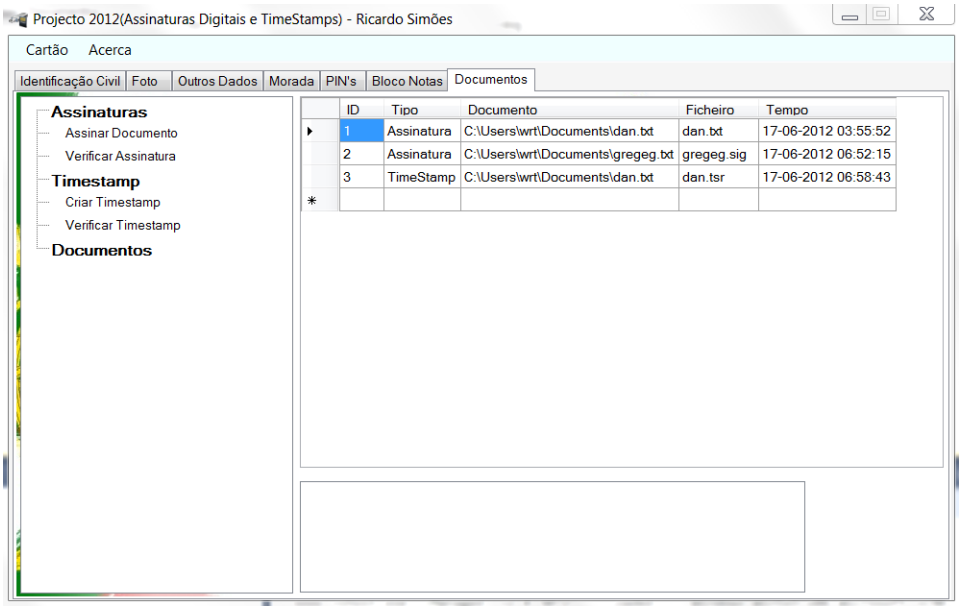

Figura 3.14: *Leitura da Base de Dados*

Capítulo 4

Conclusão e Trabalho Futuro

4.1 Trabalho Futuro

Após a realização deste projecto, sei que ainda há muito trabalho pela frente no que ao CC diz respeito. E um ponto muito importante é a segurança do CC, pois nós ao trabalharmos com este, nunca nos podemos esquecer que os dados dos cidadãos devem sempre ser protegidos. Actualmente ninguém nos pode assegurar que quando estamos a inserir o CC no respectivo leitor, este não esteja a obter os dados do cidadão.

Assim, uma grande batalha que se terá de enfrentar no que ao CC diz respeito, é a protecção dos dados e o seu facil acesso. Nunca poderá haver um motivo, que por maior que ele seja, se sobreponha aos direitos de cada cidadão.

Outro aspecto importante tem haver com a gestão da BD, onde deverá ser efectuada uma remoção da BD quando se efectuar uma alteração no documento. E mesmo a interface ao nível da BD seria algo que podia ser mais trabalhado.

4.2 Conclusão

Apesar de poder ter-se implementado mais funcionalidades neste projecto, posso dizer que o ponto chave deste, foi sem dúvida concretizado. O principal objectivo seria as Assinaturas Digitais e os TimeStamps, guardando depois estes tokens numa BD. Durante a realização deste projecto posso dizer que não foram só estes pontos que me chamaram a atenção para a realização do memso, isto porque fui-me deparando com vários assuntos relacionados com a segurança que o CC transmite, tal como já referi na secção anterior. Sem dúvida que esse é um ponto que tem de ser muito bem estudado para nos mantermos seguros e assim facilitar o uso do CC noutras ocasiões que não só Assinaturas Digitais e TimeStamps.

A maior prova de que o CC tem falhas relativamente à sua segurança, é a facilidade com que se pode recriar um middleware idêntico ao do Estado e acabar por manipular os dados de cada cidadão.

Por último, desejo agradecer novamente ao meu Orientador Paul Andrew Crocker por este projecto, pois isto para mim foi sem dúvida uma mais valia ao nível da minha formação académica e não só. Mesmo já tendo certas noções do CC e de Segurança Informática, consegui aprender muito mais sobre estes assuntos e dando-me como um *update* na minha visão sobre estes assuntos.

Bibliografia

[1] http://www.cartaodecidadao.pt/.

[2] http://www.di.ubi.pt/~crocker.

[3] http://www.di.ubi.pt/~release.

[4] http://www.portugal-a-programar.pt.

[5] http://social.msdn.microsoft.com/Forums/pt/vscsharppt/thread/52255833-a691-4fed-9db3-832d68a00ff8.

[6] http://msdn.microsoft.com.

[7] http://cartaodecidadao.codeplex.com.

[8] http://www.bouncycastle.org/.

[9] http://www.dotnetperls.com/windows.

[10] http://tsp.iaik.at/tsp/.

[11] http://en.wikipedia.org/.

[12] http://zetcode.com/db/sqlitecsharp/.

[13] www.sqlite.org/

[14] http://www.macoratti.net/10/03/c_sqlt1.htm