

Universidade de Évora
Mestrado em Engenharia Informática

Simplificação de Processos com Cartão de Cidadão

Mário Jorge Costa Mourão

Orientador : Prof. Doutor Luís Miguel Rato

Outubro de 2009

Universidade de Évora
Mestrado em Engenharia Informática

Simplificação de Processos com Cartão de Cidadão

Mário Jorge Costa Mourão

Orientador : Prof. Doutor Luís Miguel Rato

Outubro de 2009

177962

Prefácio

Este documento contém uma dissertação intitulada “*Simplificação de processos com o cartão de cidadão*”, um trabalho do aluno Mário Jorge Costa Mourão¹, estudante de Mestrado em Engenharia Informática na Universidade de Évora.

O orientador deste trabalho é o Professor Doutor Luís Miguel Rato² do Departamento de Informática da Universidade de Évora.

O autor é licenciado em Engenharia Informática, pela Universidade de Évora. A presente dissertação foi entregue em Outubro 2009.

¹mario.mourao@saphety.com

²lmr@di.uevora.pt

Agradecimentos

Mãe e Pai, obrigado por todo o apoio, ajuda e carinho que me prestaram ao longo destes anos e que tornaram este trabalho numa realidade.

Por todos os momentos em que podíamos estar juntos e não tivemos, por todos os dias em que me sentia em baixo e sem motivação para continuar, sempre estiveste presente para me ajudar e dar todo o apoio. Um grande beijo e muito obrigado Mariana.

Muitos foram os dias ao longo deste projecto em que a expressão “partir pedra” era a mais ouvida. Obrigada Ruben nestes momentos de “desespero”, pela tua imprescindível ajuda, sem ela a concretização deste projecto seria muito mais complicada.

Queria agradecer ao professor Luis Rato por toda a disponibilidade e apoio que ofereceu ao longo desta dissertação. Também queria agradecer ao professor José Sais, Miguel Reis, Nuno Palma e Artur Romão que sempre mostraram disponibilidade e me ajudaram na definição deste trabalho.

Bruno Rodrigues, ao longo destes anos universitários muitos são os momentos que vão ficar como recordação. As grandes noitadas a trabalhar (é claro!) foram indispensáveis para conseguir chegar a esta recta final da vida universitária. Muito obrigada!

Muito Obrigada a todos!

Sumário

Na sociedade moderna, o uso de novas tecnologias e das correspondentes aplicações informáticas, levanta diversas questões, sendo, sem dúvida a mais importante, a segurança dos utilizadores e dos sistemas. A implementação de novos processos fazendo uso dos meios informáticos disponíveis permite o aumento da produtividade e a sua simplificação sem perder a fiabilidade, através da desmaterialização, desburocratização, acessibilidade, rapidez de execução, comodidade e segurança.

A introdução do cartão de cidadão com todas as suas potencialidades contribui para a implementação dos processos acima referidos, os quais acompanharam a evolução do quadro legislativo nacional e europeu. Contudo verificam-se algumas lacunas, devido à sua imaturidade, sendo o seu desenvolvimento um processo ainda em curso.

Com o presente trabalho pretende-se criar uma solução aberta a várias aplicações, que permite a optimização de processos através da sua simplificação, com recurso à assinatura digital, autenticação e uso de dados pessoais, tendo em atenção a legislação vigente, o actual cartão de cidadão e os requisitos de segurança.

Palavras-chave:cartão de cidadão, assinatura, XAdES, cifra

Abstract - Simplification of processes with citizen card

In modern society, the increasing application of information technologies have arisen several questions, being the private and sensitive data security the most important. Today new informatic processes have come to increase productivity using faster and simplified mechanisms, keeping reliability and security.

The introduction in Portugal, of the new citizen card is a great example of the above mentioned, in accordance with the National and European legislation. Nevertheless, being recently adopted, it stills vulnerable and therefore is in constant update and revision.

The purpose of this thesis is the creation of an open solution to other new applications, aiming a simplification and optimization of citizen card. Seeking the maximum security it is utilised digital signature, authentication and personal details, always according to the legislation in effect.

Keywords: citizen card, signature, XAdES, cipher

Conteúdo

Prefácio	i
Agradecimentos	iii
Sumário	v
Abstract	vii
Lista de Acrónimos	xv
Glossário	xix
1 Introdução	1
1.1 Enquadramento e Motivação	1
1.2 Objectivos e Contribuições	4
1.3 Estrutura da tese	5
2 Estado da arte	7
2.1 Cartões de identificação digitais	7
2.1.1 Extracção de dados	14
2.1.2 Estrutura para dados pessoais	15
2.2 Autenticação	16
2.3 Assinatura	21
3 Trabalho Relacionado	25
3.1 Conceitos	25

3.1.1	Autenticação	25
3.1.2	Assinatura	33
3.2	Cartão do Cidadão	51
3.2.1	O chip	53
3.2.2	Autenticação com cartão do cidadão	54
3.2.3	Assinatura com cartão de cidadão	56
4	Solução proposta	59
4.1	Arquitetura	60
4.2	Interface de cliente	62
4.2.1	Fluxos	63
4.2.2	Envio de informação	66
4.2.3	Instalador JCE	66
4.2.4	Parâmetros da interface	67
4.3	API	69
4.3.1	Preenchimento automático de documentos	69
4.3.2	Gerador de chaves	71
4.3.3	Assinaturas	72
4.3.4	Gestor de PKI	76
4.3.5	Validador de certificados	79
4.3.6	Cifra	79
4.3.7	Selos temporais	81
5	Caso de Estudo	83
5.1	Portal de Contratos	83
5.2	Sigpoa	85
5.3	BizGov	88
5.3.1	Assinatura dos documentos/ficheiros	89
5.3.2	Visualização da proposta submetida	89
5.3.3	Decifrar as propostas	90
6	Conclusão e trabalho futuro	97

CONTEÚDO

xi

Anexos

107

Lista de Figuras

2.1	Modernidade na Europa dos 27 em 2007 (extraído de [12]).	10
2.2	Registo criminal austríaco (extraído de [12]).	13
2.3	EMV-CAP (extraído de [14]).	19
2.4	Autenticação EMV-CAP (extraído de [14]).	20
3.1	<i>Somebody you know</i> (extraído de [9]).	27
3.2	Autenticação mutua (extraído de [20]).	32
3.3	Geração e verificação de uma assinatura digital (extraído de [38])	42
3.4	Cartão de cidadão	52
3.5	Chip (extraído de [20]).	54
3.6	Autenticação EMV-CAP com cartão de cidadão	55
4.1	Arquitectura da API e da Interface de cliente	59
4.2	Arquitectura	62
4.3	Interface gráfica	63
4.4	Fluxo	63
4.5	Intalação automática do JCE	67
4.6	Preenchimento automático de documentos	70
4.7	Validade de um cadeia de certificados	74
4.8	Lista de certificados instalados na <i>keystore</i> do Windows	77
4.9	PKCS12	78
4.10	PKCS11	79

5.1	Visão Geral do Sistema de Arquivo e Formalização de Contratos	84
5.2	Portal de contratos	85
5.3	Abertura de propostas - Envio de <i>password</i> para o júri	90
5.4	Abertura de propostas - Júris introduzem <i>password</i>	91
5.5	Abertura de propostas - Processo de decifração	91
5.6	BizGov - fluxo 1	92
5.7	BizGov - fluxo 2	93
5.8	BizGov - fluxo 3	94
5.9	Chaves por gerar	95
5.10	A gerar chaves	95
5.11	Chaves geradas	96

Lista de Acrónimos

AC	Autoridade de Certificação
ACK	Acknowledge
ADSL	Asymmetric Digital Subscriber Line
AES	Advanced Encryption Standard
ADN	Ácido Desoxirribonucleico
API	Application Programming Interface
CA	Certificate Authority
CAP	Chip Authentication Program
CPS	Certificate Practice Statement
CRL	Certificate Revocation List
DES	Data Encryption Standard
DOC	WinWord native file; Microsoft Word
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
EC	European Commission
EEPROM	Electrically-Erasable Programmable Read-Only Memory
eID	Electronic Identity Card
EMV	Europay, MasterCard and VISA
CAP	Chip Authentication Program
ETSI	European Telecommunications Standards Institute

FTP	File Transfer Protocol
GB	Gigabyte
HTTP	Hypertext Transfer Protocol
ISN	Initial Sequence Number
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union
JCE	Java Cryptographic Extensions
JNI	Java Native Interface
LDAP	Lightweight Directory Access Protocol
MAC	Message Authentication Codes
MCH	Modelo e Continente Hipermercados
MitM	Man In The Middle
MSS	Maximum Segment Size
MIST	National Institute of Standards and Technology
NNTP	Network News Transfer Protocol
OAL	Observatório Astronómico de Lisboa
OCSP	Online Certificate Status Protocol
ODT	OpenDocument Text
OTP	One Time Password
PDF	Adobe's Portable Document Format; Adobe Acrobat Reader
PKCS	Public-Key Cryptography Standards
PKI	Simple Public Key Infrastructure
PGP	Pretty Good Privacy
PIN	Personal Identification Number
POP3	Post Office Protocol
RSA	Ron Rivest, Adi Shamir e Len Adleman
SEMIC.EU	Semantic Interoperability Centre Europe
SFTP	SSH File Transfer Protocol
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SO	Operating System
SOAP	Simple Object Access Protocol
SPKI	Simple Public Key Infrastructure

SSL	Secure Sockets Layer
TIC	Tecnologias da Informação e Comunicação
TSA	Time-Stamping Authority
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
W3C	World Wide Web Consortium
WEB	World Wide Web
XAdES	XML Advanced Electronic Signatures
XML	Extensible Markup Language
XSD	XML Schema
ZIP	Archive; PKZIP/PKUNZIP

Glossário

Background	Processos que ocorre sem que exista interação com o utilizador.
Base64	Método para codificação de dados para transferência na Internet.
BizGov	Plataforma de contratação pública.
Browser	Navegador. Programa que permite aceder a páginas e a sítios da Internet e aos recursos neles disponibilizados.
Password	Palavra-chave.
Provider	Serviço.
Smart card	Semelhante a um cartão de credito tradicional. A diferença consiste na capacidade de processamento pois possui um microprocessador e memória. Pode ser usado para cartões bancário, de identificação pessoal e para telemóveis.
Timestamp	Selo temporal.
Web	Internet.
.NET	Plataforma desenvolvida pela Microsoft para desenvolvimento e execução de sistemas e aplicações.

Capítulo 1

Introdução

Este primeiro capítulo contém uma introdução às áreas abrangidas nesta dissertação. Na secção 1.1 é apresentado o enquadramento do trabalho proposto e descrita a motivação que levou à sua realização. Na secção 1.2 são descritos os objectivos definidos para o trabalho e na secção 1.3 é apresentada a estrutura da tese.

1.1 Enquadramento e Motivação

Com a evolução tecnológica que se tem vivido até ao presente, deram-se alterações em muitos dos processos realizados no dia a dia. Muitos deles já estão de tal forma interiorizados pela sociedade que faz parecer que sempre assim foram. Contudo pontos de mudança ou ruptura existem e sempre existirão. Tomemos por exemplo, o caso de como a simplificação e automatização de processos alterou o relacionamento das pessoas com o seu banco. Quando o sistema de multibanco se massificou, os cliente deixaram aos poucos de sentir necessidade de ir ao interior de um banco, tirar a sua senha, ir para uma fila e depois então realizar a operação pretendida. Gradualmente quebraram-se alguns tabus, tais como relacionamento homem-máquina e a confiança no serviço oferecido, fazendo com que esse tipo de serviços fosse utilizado por todos os escalões etários. No entanto

a evolução não termina quando algumas metas são alcançadas. Embora esta tecnologia esteja perfeitamente implementada e em funcionamento, a necessidade de evolução, juntamente com o constante aparecimento e aperfeiçoamento de novas tecnologias (i.e, cobertura de banda larga a chegar perto do 100% , computadores cada vez mais rápidos e mais baratos, aumento do número de utilizadores, ADSL mais rápida) contribuíram para a criação do *home banking*. Neste momento o cliente de um banco não necessita de sair de casa para realizar grande parte das acções que o banco oferece.

Com o aparecimento do Cartão de Cidadão, no início de 2008, houve também uma evolução tecnológica. Os documentos pessoais (i.e, BI, cartão de contribuinte, cartão de saúde) com “mera” informação visível, foram substituídos por documento com informação digital. Toda a informação que surgia visivelmente, está agora contida num chip. Além desta informação, o cartão de cidadão permite ainda registar a residência do seu proprietário, contendo também os seus certificados de assinatura e de autenticação. Cada um dos cartões referidos era emitido por uma entidade ou serviço independente, registando cada um na sua base de dados a informação de interesse, a qual na maioria das vezes era comum. Quem sabe ou se lembra do trabalho e do custo, quando era necessário pedir segundas vias dos documentos, em caso de perda ou de roubo? Com o cartão de cidadão estas questões serão extremamente simplificadas, dado que apenas é necessário contactar um serviço público (Conservatória de Registo Civil), o qual se encarrega do cruzamento de informações que é necessário estabelecer com as restantes entidades. Se houve toda esta evolução a nível de tecnologia não estamos perante um novo ponto de mudança? A resposta é sim. Em muitas áreas e de diversas formas.

Actualmente, os nossos dados pessoais são frequentemente requisitados, desde a compra de uma casa ou o aluguer de um carro, até á simples inscrição num clube de vídeo ou ginásio. Desde questões simples a algo de

maior complexidade, um serviço requisitado ou a realização de uma compra, exige o preenchimento de todos os nossos dados pessoais e geralmente de uma forma repetitiva.

Actualmente o processo comum realizado aquando da requisição de serviços é:

- O utente desloca-se a um serviço onde lhe são fornecidos determinados documentos.
- É feito o preenchimento dos campos pelo utente nos vários documentos.
- O utente assina os documentos necessários para garantir a autenticidade do pedido.
- Volta para o balcão para entregar os documentos e finalizar o processo.

Será possível simplificar este processo? Alguns passos já foram dados nesse sentido. Por exemplo, a universidade de Évora disponibiliza alguns documentos em formato PDF dos quais os alunos podem fazer *download*, imprimir, preencher todos os seus dados pessoais em casa e basta deslocar-se ao serviços para finalizar o processo. Mas será este um ponto de chegada? Não, é apenas mais um passo para a desmaterialização deste processo.

Na linha da simplificação foi criado o SIMPLEX que consiste num Programa de Simplificação Administrativa e Legislativa, que engloba um conjunto de iniciativas que visam a facilitação da vida dos Cidadãos e Empresas. A simplificação tem por objectivo melhorar a relação dos cidadãos com os serviços públicos, reduzir os custos de contexto das empresas no seu relacionamento com estes serviços tornar a Administração pública mais eficiente e, assim, tornar Portugal mais competitivo[21]. A estratégia de simplificação pode ser concretizada através de alguns objectivos genéricos:

- Diminuir o número de atendimentos presenciais;

- Reduzir tempos de espera
- Minimizar o número de interacções relacionadas com mesmo processo
- Prestar serviços na hora
- Dar mais e melhor acesso à informação

Perante este quadro seria possível aprofundar o estudo em várias linhas de orientação. Colocou-se o foco da dissertação nas áreas da simplificação e segurança.

1.2 Objectivos e Contribuições

Em sequência do que foi descrito anteriormente, o objectivo do presente trabalho consiste na simplificação e desburocratização e segurança de processos com o cartão de cidadão, tirando partido da actualização/criação de novas legislação e dos avanços tecnológicos (i.e.,cartão de cidadão).

A desburocratização de processos é conseguida pela sua transformação em plataformas tecnológicas acessíveis a todos os utilizadores via *web*. O cartão de cidadão é um documento de identificação pessoal que se tornará obrigatório dentro de poucos anos, permitindo ainda a sua utilização e vulgarização, desde que devidamente autorizada, em representação de determinada empresa ou organismo. Para obter os objectivos referidos será necessário contemplar as seguintes áreas:

- Acesso preferencial do cartão de cidadão;
- Preenchimento automático de documentos;
- Assinatura de documentos (PDF, DOC, etc);
- Assinatura XAdES;
- Cifra/Decifra;

- Aplicação e multi-plataforma.

O objectivo principal desta dissertação é a obtenção de uma solução que permita a desmaterialização de processos, rentabilizando as capacidade do cartão de cidadão. Esta solução será utilizada em produção. Para a implementação da mesma de frisar ainda as seguintes contribuições:

- desenvolvimento de um módulo de cifra de ficheiros grandes em XML segundo os standards existentes.
- desenvolvimento de um módulo assinatura fazendo uso do cartão de cidadão.

1.3 Estrutura da tese

A dissertação será desenvolvida de acordo com a seguinte estrutura. No capítulo 2 serão desenvolvido os temas relacionados com o estado da arte em Portugal e na Europa, com especial enfoque nos cartões digitais de identificação e nas aplicações que lhes oferecem suporte. O capítulo 3 contempla uma abordagem teórica sobre os principais conceitos envolvidos designadamente, sobre autenticação e assinatura, assim como um aprofundamento sobre as potencialidades e fundamentos do cartão de cidadão. No capítulo 4 será apresentada a solução proposta com especial atenção ao desenvolvimento de uma arquitectura baseada em dois componentes complementares (API e interface). O capítulo 5 apresenta casos de estudo onde a aplicação esquematizada é implementada. Por último será feito um balanço entre os objectivos que foram definidos no início do trabalho, os resultados finais apresentados, sendo também apresentado o trabalho futuro (capítulo 6).

Capítulo 2

Estado da arte

O presente capítulo consiste numa avaliação do estado da arte. O foco reside na inovação que envolve o aparecimento dos cartões digitais de identificação¹ e as aplicações que lhes oferecem suporte. Na secção 2.1 são apresentadas diferentes aplicações e utilizações que fazem uso do cartão de cidadão. Nesta mesma secção, ainda será feita uma abordagem sobre estruturação de dados pessoais. Na secção 2.2 é realizado uma avaliação do estado da arte no âmbito da autenticação. Por fim na secção 2.3 será apresentado um conjunto de bibliotecas que permitem a realização de assinaturas.

2.1 Cartões de identificação digitais

“ For citizens, the step by step implementation of eGovernment makes everyday life much easier. ”

Guia do *eGovernment* austríaco[12]

Portugal foi dos primeiros países a usufruir do cartão de identificação do cidadão electrónico. Mas não o único, existem outros países a usufruir desta mesma tecnologia. Todos eles com os mesmos objectivos, melhorar

¹Cartão de Cidadão Português e de outros países que também utilizam cartões digitais para identificação dos cidadãos.

a interoperabilidade entre os serviços administrativos públicos e privados e os cidadãos. Segundo o guia *eGovernment* austríaco [12] uma das vantagens relacionadas com esta alteração de processos está relacionada com a eliminação da necessidade de gastar tempo em salas de espera ou de ter de preencher e entregar um formulário de papel.

A Estónia, conta com o cartão de identificação deste 2002, ano em que foi introduzido, e em 2003 já sustentava uma cobertura de 25% de toda a população. Neste país existem algumas aplicações que fazem uso deste cartão. São disponibilizadas ferramentas para assinar documentos (DigiDoc Client² e o DigiDoc Portal³), para voto electrónico via browser web⁴, para compra e utilização do cartão identificação como bilhete para os transportes públicos⁵, para autenticação e para construção de web-services. Na Estónia foram adoptados como standard de assinatura o XML-DSIG. Mais tarde, com a publicação da extensão ETSI TS 101 903 ao XML-DSIG (conhecida por XAdES) pela ETSI este passou a ser o standard escolhido para a assinatura digital [3].

Em Espanha, o *Documento Nacional de Identidad electrónico*⁶ (DNI-e) foi introduzido na sociedade em Março de 2006. Este, segue os standards definidos na directiva 1999/93/EC do Parlamento Europeu para assinaturas digitais. Este cartão conta com vastas e diversificadas áreas de utilização:

- Administração Geral do Estado⁷;

²Aplicação desenvolvida para Windows.

³Permite assinatura digital de documentos através de um portal.

⁴Link: http://www.valimised.ee/linux_eng.html

⁵O revisor tem um terminal que permite validar se o cidadão possui o bilhete para a viagem em questão, através de uma ligação a uma base de dados central (<https://www.pilet.ee/pages.php/0403>)

⁶Cartão de identificação de cidadão espanhol

⁷A lista de serviços disponíveis pode ser consultada em, http://www.dnielectronico.es/servicios_disponibles/serv_disp_adm_loc.html

novas tecnologias vai-se estendendo a cada vez mais camadas da população, permitindo pela sua larga divulgação o acesso em igualdade a todos os cidadãos.

Em todos os países descritos e em muitos outros em que as novas tecnologias anunciadas acima já fazem parte do seu quotidiano existem alguns pontos em comuns. Um desses pontos, e o mais utilizado neste momento, é a utilização de certificados digitais para autenticação. As assinaturas digitais com valor legal, fazem já parte de realidade de alguns países. A utilização de dados contidos nos cartões de identificação electrónica em geral é usada para o pré-preenchimento de formulários.

2.1.1 Extracção de dados

A maioria dos cartões de identificação, têm uma API que fornece acesso aos dados de identificação do cidadão. O cartão de cidadão Português não é uma excepção, e possui uma API que é direccionada para as funcionalidades não criptográficas designadas de eID lib[19].

A eID lib (SDK¹⁴) está destinada a organizações cujo objectivo seja desenvolver aplicações que utilizam o Cartão de Cidadão. Este kit de desenvolvimento lida apenas com os dados identificativos do cidadão e não com operações criptográficas. O kit de desenvolvimento é fornecido como:

- Uma interface C/C++, como biblioteca dinâmica;
- Uma biblioteca Java wrapper (JNI) sobre uma interface C/C++;
- Uma biblioteca C# wrapper para .NET sobre uma interface C/C++.

A API está organizada da seguinte forma:

- Funções de inicialização e término, obrigatórias para iniciar e terminar a utilização do *toolkit*;

¹⁴Software Development Kit

Bundespolizeidirektion Wien
Strafregisteramt

Wasagasse 22
1090 Wien

Gebühr entrichtet

BEZUG: SB INTERNET SRB2008072509213901
(REFERENCE NUMBER)

S T R A F R E G I S T E R B E S C H E I N I G U N G
(C R I M I N A L R E C O R D C E R T I F I C A T E)

Familienname(n): Mustermann
(Family Name)

Geschlecht: männlich
(Gender: MALE)

Vorname(n): Wendelin
(First Name)

Akad. Grad: Dr.
(Academic Degree)

Geboren am: 29.02.1956
(Date of Birth: DD.MM.YYYY)

Geburtsort: Wien
(Place of Birth)

In Strafregister der Republik Österreich - geführt von der
Bundespolizeidirektion Wien - scheint keine Verurteilung auf.
(No convictions are listed in the criminal records database of the
Republic of Austria, kept by the Federal Police Directorate of Vienna.)

DVR: 000356

Tagesdatum: 25.07.2008
(Date)

Uhrzeit: 09:40:08
(Time)

	Verfahren	orn:publicid:gv.et.sdm-srb-1.1
	Datum	2008-07-25T09:40:08
	Aussteller	CS-a-sign-corporate-light-02,00-a-sign-corporate-light-02,0-a-Trust Ges. f. Sicherheitssysteme in elektr. Datenverkehr Gmbh, C-AT
	Seriennummer	82210
Signaturwert	K2cspv3h720j5bGLTc2ncv8D+afqIn1XnqY1y1E2hm+}rgn8pe20Ds w3/8est7ytT0WskVtpejltLOT4b9nd/#2X0HA-CR0q/Dnr0FV4Pccxyj l6K0899v672000ct10ascqO1y8vrc88M 0uQ8E17a1Q998E1+we7db+	
Prüfinformation	Informationen zur Prüfung der elektronischen Signatur finden Sie unter: https://apps.sgis.gv.at/srb-wart	
Hinweis	Dieses Dokument wurde zertifiziert. Auch ein Ausdruck dieses Dokuments hat gemäß §20 e-Government-Gesetz die Beweiskraft einer öffentlichen Urkunde.	

Figura 2.2: Registo criminal austríaco (extraído de [12]).

aos objectivos de simplificação, agregação, segurança e desmaterialização que prossegue. A potencialidade deste instrumento poderá ainda ser ampliada de futuro, quer pela introdução de novas funcionalidades como pelo desenvolvimento de novas aplicações, tendo sempre presente a sua garantia em termos de segurança e polivalência. Por outro lado o acesso às

cidadãos declarar o seu estado de desemprego com antecedência, bem como cancelá-lo novamente através da Internet. Ao poder declarar o estado de desemprego, mesmo antes de o estar, aumenta as hipóteses de encontrar um novo emprego mais rapidamente. Sendo positivo para o cidadão, que passa menos tempo desempregado, como também para o estado que minimiza os gastos com desempregados. Através deste serviço é possível accionar os pedidos de subsídio de desemprego. As entidades empregadoras têm uma maior facilidade na procura de novos funcionários.

- As autoridades públicas possuem uma assinatura electrónica, tal como os cidadãos, que pode ser usada para assinar documentos digitais.
- Esta assinatura é anexa aos documentos.
- Esta assinatura oficial tem a particularidade de que, juntamente com alguns atributos, permitem verificar a assinatura e confirmar a sua validade, mesmo quando o documento é impresso em papel.
- Essa assinatura colocada no papel está presente na figura 2.2.

Nem todos os países, contam com cartões de identificação electrónicos, por exemplo, a Alemanha é um pouco mais prudente e só receberá o cartão electrónico em 1 de Novembro de 2010, sendo este denominado de *elektronischer Personalausweis*(ePA).

Em Portugal o cartão de cidadão nasceu com o intuito de permitir a identificação visual e presencial de um cidadão e a identificação e a autenticação electrónica do cidadão nos actos informatizados em que intervenha. Este cartão agrega e substitui os actuais cartões de contribuinte, de utente do serviço nacional de saúde, de beneficiário da segurança social e de eleitor. O projecto Cartão de Cidadão integra-se na política de modernização da Administração pública constante do Programa do Governo. É também um dos principais catalisadores da estratégia de modernização, atendendo

- Transparência;
- Acessibilidade;
- Usabilidade;
- Segurança dos dados;
- Cooperação;
- Sustentabilidade;
- Interoperabilidade;
- Neutralidade Tecnológica;

Este último ponto considera que os sistemas, soluções e dispositivos são desenvolvidos no ramo da informação e comunicação, com maior rapidez do que em qualquer outra área. Um produto que hoje é novo, amanhã já está desactualizado. Então, segundo [12], o eGovernment deve ser aberto a novos desenvolvimentos e não insistir em usar apenas uma tecnologia em particular. Não deve ficar dependente de um software específico ou do um único hardware. A Áustria, apresenta um conjunto de soluções que fazem uso do *e-card*¹², que vão desde a assinatura, autenticação, passando pela utilização dos dados contidos no cartão. Estas soluções, visam reduzir tempos e custos, assim como dispender de dias de férias apenas para resolver assuntos pessoais. Surgem a seguir alguns exemplos de soluções desenvolvidas:

- Possibilidade de requerer o registo criminal utilizando o *e-card*. Este, é assinado digitalmente e possui total valor legal. Exemplo, Figura 2.2.
- “AMS Next Job” - Assim que um cidadão recebe um aviso prévio de despedimento ou pretende deixar o emprego para encontrar um novo que vá de encontro as suas expectativas, deseja que a mudança seja rápida e com a menor burocracia possível. Para que este processo seja agilizado foi desenvolvido o projecto “AMS”¹³, que permite aos

¹²Cartão de identificação electrónico Austríaco

¹³Arbeitsmarktservice - Serviço de Mercado de trabalho

organizações realizem os seus negócios com o governo, com mais facilidade, mais rapidez e menor custo [1].

Em 2001, o governo Austríaco delineou uma estratégia, que passava por colocar este país no topo na área de *eGovernment*. Esse objectivo foi atingido em 2006 e reforçado em 2007, como podemos constatar na Figura 2.1. A Áustria, conseguiu alcançar o sucesso concentrando os seus

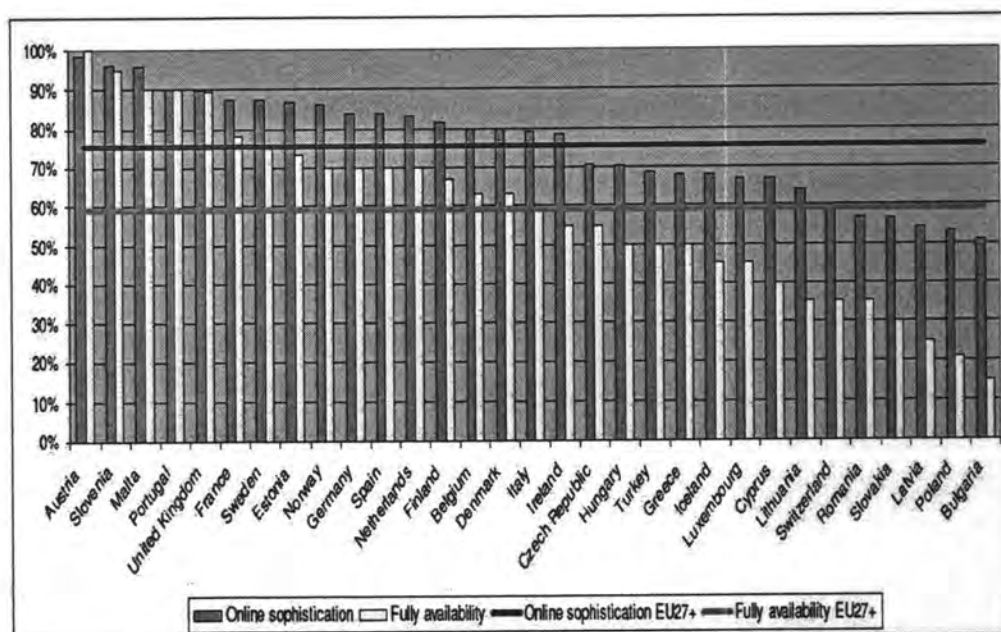


Figura 2.1: Modernidade na Europa dos 27 em 2007 (extraído de [12]).

esforços na construção de uma infra-estrutura aberta e escalável que pode ser expandida, e ao mesmo tempo, segura e sustentável a longo prazo. Os pilares deste sucesso assentam no seguimento dos seguintes princípios:

- Proximidade com os cidadãos;
- Conveniência através da eficiência;
- Confiança e segurança;

- Nas comunidades Autónomas ⁸;
- Na Administração local ⁹;
- No sector privado ¹⁰.

De salientar, que da lista de serviços disponibilizados no sector privado uma boa parte é oferecida pela banca, em muitos casos como meio de autenticação. Este sector vai mais além, e segundo [25] a *Caja Madrid* ¹¹ começou a adaptar, em Setembro de 2007, as suas caixas multibanco, de forma a permitir operações com o novo cartão de identificação electrónica. Para a *Caja Madrid* estas máquinas estão sub-utilizadas e é necessário explorar todo o seu potencial. Fazendo uso da identificação electrónica, é possível efectuar levantamentos e realizar transferências bancárias, por exemplo. A *Caja Madrid* não acredita que esta tecnologia esteja totalmente difundida até 2011, mas a entidade quer reforçar e ser pioneira em aplicações comerciais para esta nova forma de identificação digital.

O *eGovernment* consiste no uso de ferramentas e sistemas que tornam possível, através das tecnologias de informação e comunicação (TICs), melhorar os serviços públicos para os cidadãos e para as empresas. As TICs já se encontram disseminadas pelos órgãos governamentais, assim como nas empresas, mas o *eGovernment* envolve muito mais que apenas ferramentas. Um *eGovernment* eficaz, implica também repensar as organizações, processos e mudanças de comportamentos, para que os serviços públicos respondam mais eficientemente aos cidadãos que necessitam de os utilizar. Bem aplicado o *eGovernment* permite que todos os cidadãos, empresas e

⁸A lista de serviços disponibilizados pelas comunidades autónomas pode ser consultada em, http://www.dnielectronico.es/servicios_disponibles/serv_disp_ccaa.html

⁹A lista de serviços disponibilizados pelas administrações locais podem ser consultada em, http://www.dnielectronico.es/servicios_disponibles/serv_disp_ccaa.html

¹⁰A lista de serviços disponibilizados pelo sector privado está disponível em http://www.dnielectronico.es/servicios_disponibles/serv_disp_priv.html

¹¹Banco Espanhol

- Funções de identidade, usadas para obter dados identificativos (nome, morada, etc.) do cartão;
- Funções de uso geral de alto nível, usadas para aceder a dados de uma forma genérica (ficheiros, PIN), principalmente noutras aplicações que não as de identidade. Não há necessidade de usar estas funções para aceder a dados identificativos.
- Funções de segurança. Existem funções para garantir a integridade dos dados do cartão, assim como funções para estabelecer sessões seguras, com garantia de autenticação e confidencialidade.
- Funções de alteração do cartão, são funções que permitem alterar a morada escrita no cartão e o PIN da applet EMV-CAP.

2.1.2 Estrutura para dados pessoais

No seguimento do programa *eGovernment* do governo austríaco surgiu a definição de uma estrutura em XML para dados pessoais. Esse registo é usado exclusivamente para descrever um pessoa através da interligação dos seguintes blocos de informação: “Pessoa”, “Morada” e “Assinatura”. Esta estrutura é utilizada em todos os processos de *eGovernment* em que sejam usados dados pessoais [12].

As aplicações que são construídas sobre esta estrutura XML, podem ainda derivá-la, fazer restrições ou expandi-la, na medida do necessário para se adaptar às suas próprias necessidade.

O nível de topo para o objecto genérico “Pessoa“ define elementos para as pessoas singulares e colectivas. Os elementos que definem uma pessoa singular incluem aspectos tais como, nomes, estado civil, sexo, naturalidade, data de nascimento, nacionalidade, etc. Os elementos que descrevem uma pessoa colectiva incluem nome completo, nomes alternativos,

etc.

O esquema descreve também um objecto endereço onde é especificado número de telefone, *site* ou endereço postal, juntamente com elementos específicos para cada tipo de atributo. Este esquema foi apresentado à SEMIC.EU¹⁵ (Projecto Europeu com vista a suportar a troca de informação para Serviços de e-Government Pan-Europeus, constituindo uma plataforma central para a colaboração e troca de activos em termos de interoperabilidade semântica).

Segundo o relatório [40] da SEMIC.EU a qualidade geral do projecto apresentado é elevada. A documentação é extensa, mas, infelizmente, disponível apenas em alemão. Por outro lado, contém exemplos e informação adicionais para implementadores. O modelo está estruturado de forma clara e fácil de entender. O conteúdo que está muito específico para o governo austríaco e deveria ser descrito de forma mais clara, de modo a que seja compreensível num contexto pan-europeu. Segundo o relatório [40] da SEMIC.EU foi atribuído o estado de “maduro” a este projecto.

Foi solicitado um esclarecimento à AMA (Agência para a Modernização Administrativa) sobre existência de uma estrutura para dados pessoais. A resposta enviada por *e-mail*[11] foi no sentido de não existir nenhuma estrutura para este efeito, uma vez que se assume que em cada projecto e de acordo com as necessidades se defirá o formato.

2.2 Autenticação

A autenticação é cada vez mais um requisito indispensável para obter bons níveis de segurança, contra as ameaças que surgem no dia a dia. Em muitos dos crimes descritos na secção 3.1.1 (*phishing*, *pharming*, *vising*, *scam*,

¹⁵Centro de Interoperabilidade Semântica Europeu

entre outras) não é trivial saber quem são os verdadeiros culpados. E para valores baixos de fraude não compensa muitas das vezes participar, devido aos valores jurídicos associados principalmente quando o crime é cometido através de outro país. Por isso os bancos tentam apostar em dois factores no alerta dos seus clientes e investimentos nas medidas de autenticação.

Schneier argumentava em 2005 [39] que autenticação de dois factores não era a nossa salvação. E que esta não nos protege contra o *phishing* e outros tipo de ataques apresentados anteriormente. Schneier defendia que esse tipo de autenticação resolvia os problema de segurança de há 10 anos atrás. Facilmente se perde o controlo das palavras-chave, o medo de as perder leva as pessoas a escrevê-las em papel, ou a passá-las a outras pessoas, enviam-nas por e-mail que fácilmente são interceptados. Segundo ele a solução passa pela utilização de T-FA com a utilização de *token*, com geradores de novas senha a cada minuto que passe ou através de troca de senhas por cada nova utilização, ou seja, quando existe uma nova transacção com um banco é usada uma senha e o banco responde devolvendo a nova senha que será usada na próxima comunicação. Com o aparecimento da autenticação por diferentes canais (T-CA¹⁶) via *password* e um código enviado via SMS contribui para o fortalecimento da T-FA. Embora estes não ofereçam garantias que não haja qualquer tipo de fraude, Schneier acredita que os bancos invistam bastante dinheiro em investigação para que os danos sejam os menores possíveis.

Nestes parágrafos foi demonstrado a grande importância e confiança que é necessário ter durante a autenticação. Os bancos são muitas vezes tomados como exemplo já que este é um sistema totalmente implementado na sociedade e ao mesmo tempo um sistema crítico em relação à segurança, devido aos fluxos financeiros em jogo.

¹⁶Two-Chanel Authentication

Na actualidade uma das formas de garantir a segurança dos dados através da *web* é com o SSL. O SSL como um canal seguro, que é, implementa vários mecanismos de autenticação. Um dos que oferece maiores níveis de segurança é denominado por *handshake*. Este processo está definido na secção 3.1.1. Descrição simplificada para configuração de uma ligação cliente / servidor:

- **Configurar o servidor para ligações SSL** - será necessário obter um certificado para o site em questão e proceder à sua instalação no servidor web, de modo a ser possível estabelecer comunicações seguras entre o servidor e as aplicações clientes;[17]
- **Configurar o servidor para aceitar certificados de clientes** - neste passo configura-se o servidor de modo a este efectuar o pedido ao cliente de um certificado digital;[17]
- **Configurar o servidor para pedir e aceitar o certificado do cartão de cidadão** - os servidores estão normalmente configurados para pedir e aceitar certificados clientes que sejam emitidos pelo seu LDAP. Neste passo configura-se o servidor de modo a que aceite igualmente certificados emitidos pela *Certification Authority* emissora dos certificados presentes no cartão de cidadão;[17]
- **Validação aplicacional do certificado** - de forma a garantir que só são aceites certificados presentes nos cartões de cidadão, o código desenvolvido para autenticação, deve validar um conjunto de parâmetros presentes no certificado, de forma a garantir a origem do certificado;[17]
- **Validação de validade do certificado** - a última validação é efectuada pela entidade emissora do certificado, de modo a garantir que o certificado não foi revogado, sendo que poderão ser usados dois métodos, CRL[29] (Certificate Revocation List) ou OCSP[31] (Online Certificate Status Protocol).[17]

Em Portugal a SIBS¹⁷ prevê implementar este sistema para realizar uma autenticação forte online. Partindo das várias alternativas apresentadas na figura 2.3, chegaram às seguintes conclusões [14]:



Figura 2.3: EMV-CAP (extraído de [14]).

- **Matriz OTP** - O cartão matriz tem como vantagem: a portabilidade, a não exigência de dispositivos externos e o baixo custo de implementação. Mas apresenta como desvantagens: o risco de cópia da matriz, dificuldade na gestão de credenciais usadas e disponíveis e apresenta fragilidades contra ataques *MITM*¹⁸.
- **SMS** - O envio de mensagens escritas como forma de um segunda via de autenticação tem como vantagem: a portabilidade, utilização de um dispositivo já existente, os baixos custos de implementação e rapidez de implementação. Mas como desvantagem apresenta: estar dependente da infraestrutura da rede móvel (i.e, falta de rede) e o custo associado ao envio de mensagem.

¹⁷Sociedade Interbancaria de Servicos

¹⁸Man in the middle

- Cartão EMV – Esta solução está em alinhamento com as tendências do mercado mundial e aponta como vantagens: conjugação de segurança no eBanking e no comércio electrónico e como este segue as normas EMV-CAP a independência com o fornecedor de cartões e de leitores apenas têm de seguir a implementação. Esta solução, tal como as anteriores, também apresenta pontos desfavoráveis, tais como: exigência de um dispositivo externo adicional.

A SIBS aponta o EMV-CAP como melhor solução. Tal como está ilustrado na Figura 2.4 a realização da autenticação segundo esta tecnologia passaria pelos seguintes passos:

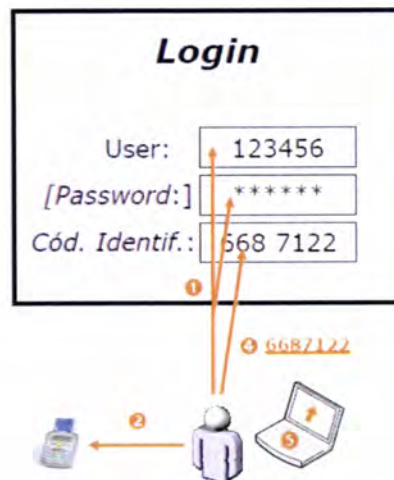


Figura 2.4: Autenticação EMV-CAP (extraído de [14]).

1. Introduzir utilizador e a *password* (1º factor);
2. Escolher a função e introduzir cartão e o PIN no dispositivo;
3. O dispositivo gera código OTP (cartão EMV-CAP);
4. Introduzir código OTP gerado pelo cartão EMV-CAP;
5. Prosseguir operações.

2.3 Assinatura

A assinatura XAdES assenta nos standards definidos pela ETSI¹⁹, e as suas especificações técnicas serão abordadas na secção 3.1.2. Esta assinatura reúne todas aspectos técnicos que lhe garantam toda a validade legal. Também o Parlamento e Conselho Europeu definiram esta assinatura com standard a seguir, na directiva 1999/93/EC [22]. Na presente secção serão apresentadas bibliotecas existentes que permitem a realização e validação de assinaturas XAdES. De seguida, serão apresentadas quatro bibliotecas que cumprem estes requisitos, sendo elas:

- *OpenXAdES*;
- *DContract*;
- *EldoS*;
- *IAIK*;

O OpenXAdES surgiu com o intuito de demonstrar que as assinaturas digitais não são para mera demonstração tecnológica, mas efectivamente para facilitar a vida das pessoas.

As bibliotecas fornecidas pelo OpenXAdES estão disponíveis em C e em Java para a criação de documentos, assinatura e verificação. A criação de documentos usa um formato específico para armazenar os documentos originais assinados (seja qual for o formato dos mesmos), as assinaturas e as informações técnicas associadas. A estrutura destes documentos é baseada nas normas técnicas definidas pelo ETSI. Com base nas bibliotecas e na estrutura referida foram desenvolvidas duas aplicações. Uma das aplicações corre no lado do cliente em ambiente Windows (DigDoc Client), a segunda é um portal que se encontra disponível *online* mas que, ao contrário do anterior não necessita de nenhuma instalação de software no lado do cliente.

¹⁹European Telecommunications Standards Institute

Embora esta implementação siga o *standard* XAdES, este perfil não corresponde exactamente à sua definição. Deste, não fazem parte os selos temporais, sendo as datas validadas junto das respostas OCSP. Segundo os seus criadores, este perfil poderia ser denominado de “XAdES-C-L”[42]. Deve-se realçar que este é um projecto *open-source*.

O *Dcontract* é uma biblioteca disponível para o Java. É uma ferramenta que permite gerir assinaturas digitais. Este oferece a possibilidade de realizar todos os níveis da assinatura XAdES até à XAdES-X-L a partir da versão 1.3.2. Algumas características chave são apresentadas pelo *Dcontract*:

- Assinatura, co-assinatura e contador de assinaturas de um documento digital;
- Robustez na verificação de assinaturas digitais;
- Possibilidade de configuração de alguns aspectos da assinatura;
- Suporte de XAdES-X-L;
- Fornecimento de um nível de abstracção para a fácil utilização de assinaturas com valor legal;
- Repositório de documentos para obter/guardar documentos XML DOM de / para diversas fontes;
- Presença de um documento esquema que permite a validação de forma simples;
- Geração de certificados X509 *self-sign*, dando suporte às extensões mais comuns;
- Possibilidade de gerir repositórios PKCS12;
- Suporte para *smart card*;

As bibliotecas desenvolvidas pela *EldoS* oferecem um conjunto de ferramentas que garantem a segurança dos dados, quer seja através de assinaturas, cifra ou protecção do canal de comunicação. Estas ferramentas correm em diversas plataformas Windows, Windows Mobile e Linux. Estas permitem também, a integração em diferentes linguagens de programação *Visual Studio*, *Delphi*, *Kylix and FreePascal*, *Visual C++* bem como outras ferramentas que suportem .NET ou *ActiveX*.

A *SecureBlackbox* (solução da *EldoS*) apresenta as seguintes características como principais referências:

- Utilização de protocolos standards da Internet, para transferência de dados. Nomeadamente, o FTP, HTTP, SMTP, POP3 e o NNTP;
- Transferência de ficheiros de forma segura através de um canal seguro SFTP. A SFTP API possui funções que permitem criar, ler e escrever ficheiros; criar, apagar e enumerar directorias, e alterar atributos de um ficheiro;
- Suporte a chaves *OpenPGP*. Possibilidade de estender a aplicações próprias de segurança, utilizando funções que vão desde a assinatura, cifra, decifra e verificação dos dados, usando chaves OpenPGP;
- Possibilidade de assinatura e cifrar de PDF, dando ainda a oportunidade de proteger os mesmo com *password*;
- Possibilidade de assinatura e cifrar de XML dando ainda a capacidade de proteger os mesmo com *password*. A função de assinatura e cifra são de acordo com as especificações XMLDSig, XAdES e XMLenc. Para a geração de uma assinatura XAdES só é dada a possibilidade de implementação da assinatura XAdES-BES, XAdES-EPES ou XAdES-T;
- Permissão para adicionar as funcionalidades de PKI facilmente a uma aplicação.

O SecureBlackbox não é uma solução, nem *open-source*, nem gratuita. Tem como ponto favorável a quantidade de informação existente no site (documentação, exemplo de utilização, wiki, fórum).

O IAIK oferece uma ferramenta de segurança para XML denominada de XSECT²⁰. Esta ferramenta, oferece métodos de assinatura digital, criptografia e manipulação de documentos em XML. Fazem parte do XSECT duas APIs, uma de assinatura e outra de cifra denominadas de *XML Digital Signatures* e *XML Digital Encryption* respectivamente. Ambas as APIs foram desenhadas de forma a serem utilizadas para correr em plataformas Java. A IAIK-XAdES é uma *add-on* para a biblioteca XSECT. Esta vem permitir a criação de assinaturas electrónicas avançadas, que continuam a ser válidas por longos períodos de tempo e são compatíveis com a directiva da União Europeia relativamente às assinaturas electrónicas. Esta biblioteca pode ser integrada em versões acima do Java 2 (JDK 1.2). A IAIK-XAdES possui independência relativamente ao provedor de criptografia, uma vez que pode ser utilizada com qualquer provedor normalizado JCA/JCE 1.2 desde que o IAIK's JCE se encontre no classpath. Fornece uma integração simples com *smartcards* ou outros módulos de segurança sobre *hardware* e é fornecida juntamente com os produtos IAIK-JCE e IAIK-XSECT.

²⁰XML Security Toolkit

Capítulo 3

Trabalho Relacionado

3.1 Conceitos

Nas próximas secções serão apresentadas as tecnologias que servem de base ao cartão de cidadão e às suas aplicações. Tais como :

- Autenticação;
- Assinatura;

3.1.1 Autenticação

A autenticação consiste em provar a identidade de um utilizador perante terceiros. Até alguns anos atrás esta autenticação era quase sempre feita de um modo presencial. Nos dias de hoje, caminhamos no sentido da realização de uma autenticação electrónica.

A autenticação electrónica pode ser realizada num ambiente *web* ou por telefone. Esta pode ser realizada de diversas maneiras, a mais simples e a mais comum, nos dias de hoje, é através do nome de utilizador e palavra-chave. Este meio pouca segurança oferece, apenas usa um factor de autenticação. Factores de autenticação, são factores de ordem distinta que podem ser usados para provar a identidade de uma pessoa de modo



seguro. Existem factores de várias ordens, factores de propriedade, onde é requerido algo que se tem (i.e, token, cartão de identificação¹, telefone ou telemóvel), factores de conhecimento que abrangem algo que o utilizador sabe (i.e, palavra-chave, frase chave, PIN, Número de identificação pessoal), outro factor é o de inerência e conta com algo que se é (i.e, impressão digital, leitura da retina, ADN, assinatura, e outros meios biométricos).

Quando por algum motivo, um utilizador esquece uma palavra chave o modo de recuperação desta é através do reenvio das chaves para o email ou através de uma pergunta acerca da vida pessoal do utilizador (i.e, “Como se chamou o teu primeiro animal de estimação?”). É perceptível que este método tem falhas do ponto de vista da segurança. O quarto factor será *somebody you know*² que vem colmatar este tipo de falhas no sistema actual. Neste modelo, que foi concebido para do caso o bloqueio de um token, o utilizador pede a alguém conhecido e em quem confia que lhe gere um *vouch*. Este entra em contacto com o servidor faz a sua autenticação usando um ou mais factores descritos anteriormente, o servidor devolve um *vouch* que o utilizador devolve a quem pediu ajuda. Este com o *vouch* efectua novamente o pedido de autenticação junto do servidor, e prossegue a comunicação [9]. Para este processo correr como esperado, é importante ter alguém que se conheça, se confie plenamente e use o mesmo sistema. A figura 3.1 ilustra este processo.

Os factores de autenticação atrás descritos são os principais, mas a esta lista ainda podem ser acrescentados factores baseados no tempo e na localização. Apresentados os factores de autenticação será abordado a autenticação em si. Como já foi referido a autenticação pode ser realizada em ambientes *web* mas também por canais alternativos. Começemos por abordar em ambientes *web*.

¹Cartão do cidadão ou cartão Multibanco

²Alguém que tu conheces

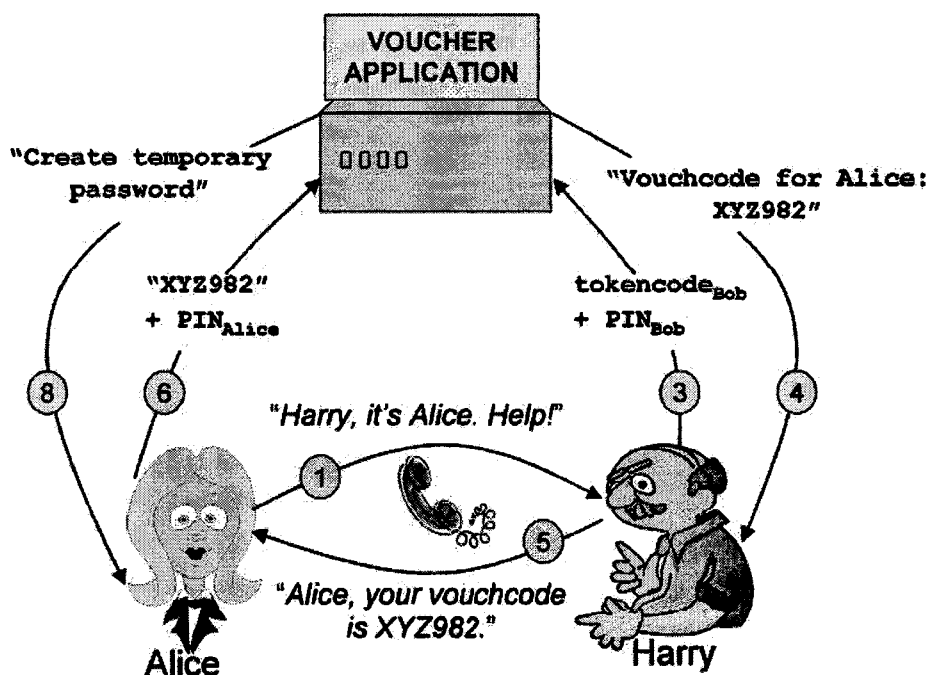


Figura 3.1: *Somebody you know* (extraído de [9]).

Em ambientes *web*, não é possível isolar o utilizador e os seus respectivos dados. Diferentes pessoas, com diferentes funções podem aceder a estes dados, com a condicionante de que nem toda a informação está disponível para qualquer utilizador. Por exemplo, dentro de uma empresa, os clientes, os empregados ou os directores têm acesso a dados comuns, no entanto, nem toda a informação está disponível para os três de igual forma e nem todos têm acesso á mesma quantidade de informação. É necessário e indispensável garantir que estes dados estão guardados de forma perfeitamente segura. Um determinado utilizador tem de realizar a sua identificação perante um servidor. Carece então de algo mais que apenas a autenticação. Surge assim, a necessidade de diferenciar autenticação de autorização.

Enquanto que, a autenticação consiste no acto de alguém comprovar a sua identidade perante uma pessoa ou entidade, a autorização define níveis de acesso. Estes níveis de acesso podem ser físicos (i.e., área reservada de uma empresa, ao cofre de um banco) ou virtuais (i.e., não ter acesso a todos os conteúdos de um site; ter privilégios restritos numa determinada “máquina”).

Um utilizador pode realizar a sua autenticação com sucesso, no entanto, isto não significa que tenha acesso a todos os dados existentes, mas apenas aqueles que foram previamente disponibilizados para esse determinado utilizador, ou grupo de utilizadores pelo administrador do sistema. O nível de acesso só tem relevância se a autenticação for realizada com sucesso.

Aquando da realização da autenticação, é possível observar que um método A é mais seguro que um método B, ou que um método B é menos seguro que um método C. No entanto, se se juntar A e B para efectuar esta autenticação, é mais seguro que um método C. Se for escolhido dois ou mais métodos para a realização da mesma está-se perante uma autenticação forte. Para uma autenticação multi-factor ou autenticação de dois factores (T-FA³) é necessário que sejam utilizados dois ou mais métodos de pelo menos 2 factores descritos anteriormente.

Quando se pensa em sistemas críticos logo se pensa em segurança. Um óptimo exemplo de um sistema crítico é o utilizado pelas entidades bancária. Os dados dos bancos são confidenciais. Além da obrigatoriedade dos níveis de acesso estarem bem definidos, é indispensável que a autenticação ocorra sem falhas. Se um utilizador se conseguir autenticar usando a identificação e dados alheios, pode visualizar todos os detalhes da respectiva conta bem como movimentar os seus bens. Quando um utilizador se está a autenticar com os dados de outro está a praticar um crime, que em

³Two-Factor Authentication

Portugal e em muitos outros países é punido por lei. Este tipo de crimes será equiparado a falsificar a assinatura de outra pessoa. Os bancos têm estado sujeitos a muitos outros tipos fraudes nos últimos anos tais como:

- *phishing* - Consiste em simular ou copiar um site, e enviar email a todos os cliente possíveis sobre um qualquer pretexto para que actualizem os seus dados. Os autores do phishing fornecem também link para uma para uma página que simula ser a original do banco;
- *pharming* - Consiste na manipulação das direcções DNS, levando a que os endereços escritos no *browser* não encaminhem ao site da entidade bancária procurado, mas sim a um idêntico que pede os dados ao cliente e os envia para onde o responsável pela fraude desejar. Para que este tipo de fraude ocorra é necessário utilizar algum tipo de aplicação maliciosa (vírus, trojan, ...);
- *vising* - A vítima recebe um email em nome da sua entidade bancária, com um número de telefone que deve contactar. Do outro lado a vítima ouve uma gravação que pede os dados do cartão e os códigos do mesmo. Este tipo fraude também se pode verificar via mensagem escrita, em que o utilizador recebe um sms no telemóvel onde foi efectuado um depósito de X euros e um número de telefone para onde pode ligar em caso de dúvida. Mais uma vez do outro lado espera um atendedor que pede e regista os dados de utilizador;
- *scam* - Este tipo de fraude é pouco diferente das outras apresentadas e consta de anúncios de trabalhos na internet em que publicitam que é possível ganhar dinheiro facilmente e a partir de casa. A única coisa necessária é possuir um computador, acesso à internet e um conta bancária. A função desta pessoa é receber os dinheiro das pessoas vítimas de *phishing*, encaminhando posteriormente para outras contas e ficando com um parte do dinheiro. Esta pessoa, que em princípio pode nem saber que está num negócio fraudulento, ao ficar com um percentagem do dinheiro está a agir como segundo cúmplice;

Como foi descrito ao longo desta secção existe uma grande variedade de formas que podem ser usadas para a autenticação. Nos próximos parágrafos serão apresentadas duas formas distintas de fazer autenticação via *web*. A primeira diz respeito a autenticação via *browser* a segunda a protocolos de autenticação via *web-services*.

A autenticação através do *browser* é sobre SSL[28], pode ou não existir autenticação mútua. No caso que irá ser descrito existe autenticação mútua, ou seja, o servidor autentica-se perante o cliente e o cliente também necessita de se autenticar perante o servidor.

Para o estabelecimento de uma autenticação com sucesso via SSL, entre cliente e servidor, é necessário que os passos abaixo descritos ocorram todos com sucesso:

1. O cliente faz um pedido de sincronismo ao servidor com o qual quer comunicar. É gerado no cliente o ISN⁴(Numero de sequência inicial). A este valor é acrescentado uma unidade (i.e, se o ISN gerado for 1356735 a sequência terá inicio em 1356736). Nessa mensagem poderá também ser definido o MSS⁵(Tamanho máximo do Segmento). Se este valor for 5 então a sequência enviada será 1356736-1356740. Outro valor importante que será enviado será o ACK⁶. Neste caso será enviado a zero, mas apenas nesta primeira de três fases tem este valor. Além de mais alguma informação complementar a mensagem enviada para o servidor inclui (i.e., Len = 4, seq = 1356736-1356740, ACK = 0).
2. Na fase dois, o servidor recebe a mensagem do cliente e tenta sincronizar-se com este. Ao mesmo tempo o servidor envia para o cliente o seu pedido de sincronização com a sequência. A forma do pedido é igual ao pedido criado no passo anterior pelo cliente, com a

⁴Initial Sequence Number

⁵Maximum Segment Size

⁶Acknowledge

diferença de que o valor do ACK já não será 0 mas o primeiro valor da sequência recebida pelo cliente com o incremento de uma unidade. O pedido de sincronismo é então enviado para o cliente (i.e, len = 4, seq = 1109870-1109873, ACK=1356737).

3. O cliente recebe o pedido e valida se o ACK recebido corresponde ao número de sequência após o incremento de uma unidade. É validado o certificado do servidor, certifica-se se este se encontra expirado e se o mesmo tem a confiança do cliente. É também escolhido o certificado utilizado para o cliente se autenticar perante o servidor. O cliente envia então a sua última mensagem antes de estabelecida a chave de sessão, envia a confirmação de sincronismo com o servidor (i.e, len = 0, seq = 1356737-135673, ACK = 11098701).
4. O servidor confirma o sincronismo com o cliente através de ACK, tal como foi feito do cliente na fase anterior. É então validado o certificado do cliente por parte do servidor. É comprovado se o certificado está registado no sistema, o que pode ser feito de duas formas: a primeira através da impressão digital do certificado⁷; a segunda através do número de série do certificado e nome da Autoridade de Certificação (AC). É verificado também se o certificado não se encontra expirado ou revogado. Para comprovar se um certificado se encontra ou não revogado, é necessário validar junto da AC. Existem dois serviços distintos usados para a validação da revogação de um certificado, apenas é necessário a AC oferecer um deles. Um dos serviços é apresentado através de uma Lista de Certificados Revogados[29] (CRL⁸). Esta lista contém o número de série dos certificados revogados, a data em que foram revogados. Para garantir a autenticidade da lista, esta é assinada pela AC. O outro serviço é o OCSP[31] (Online Certificate Status Protocol) e consiste num pedido que é realizado à AC e que tem como argumento o número de série do certificado. Na resposta vem o estado

⁷fingerprint

⁸Certificate Revocation List

em que se encontra o certificado, sendo esta resposta assinada pela AC para garantir a sua veracidade. A figura 3.2 exemplifica o processo anterior.

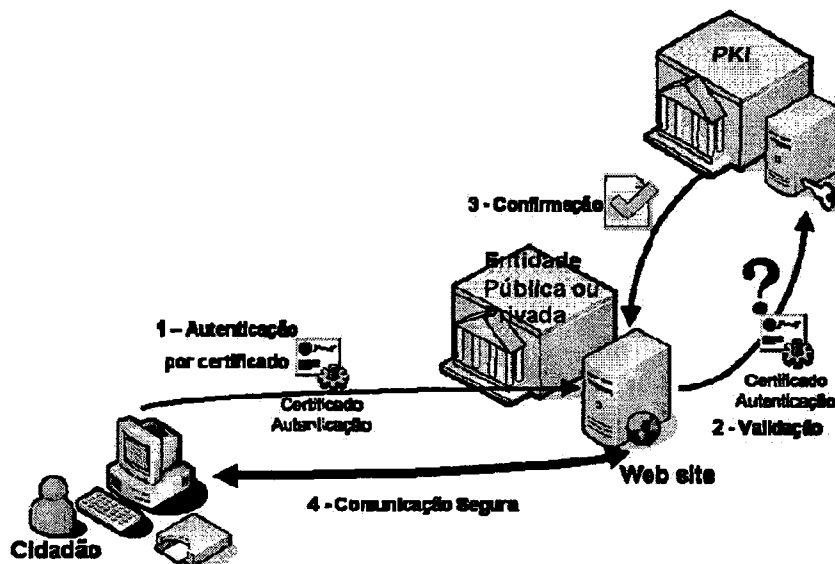


Figura 3.2: Autenticação mútua (extraído de [20]).

É então gerada e trocada a chave de sessão que vai passar a ser utilizada para comunicação. Até este ponto as mensagens trocadas terão sido assinadas com as chaves do servidor e do cliente conforme o destinatário da mensagem. O protocolo de comunicação descrito em cima, de estabelecimento de uma autenticação e troca da chave de sessão designa-se por “handshake”.

A utilização da autenticação em *web-services* é similar ao processo descrito anteriormente. Este processo tem como base o SSL. Tanto o cliente, como o canal em si pode ser configurado de maneira a obter o melhor resultado velocidade/segurança. Se se escolher uma autenticação básica do cliente (i.e. nome de utilizador, palavra-chave) e um canal em que a mensagem não tenha qualquer tipo de codificação, esta conexão será rápida mas apresentará fragilidades a nível de segurança. Por outro lado, se se

definir uma autenticação mútua com certificados de cliente e servidor e definir que toda a troca de mensagem será cifrada, o risco de quebra de sigilo da mensagem e da autenticidade dos intervenientes será menor. No entanto, o aumento da segurança, aumenta também o tempo de passagem dessa mesma informação e requer ainda, maior capacidade de processamento.

O Sistema EMV-CAP é responsável pela geração de palavras-chave únicas também conhecidas por OTP⁹. Este sistema é usado normalmente por canais alternativos, como por exemplo o telefone. A EMV-CAP tem como base as normas desenvolvidas pela Europay, Mastercard e pela Visa. São estas que lhe dão o nome (EMV[27]) e são responsáveis pela interoperabilidade entre os *token* e os leitores que permitem a leitura dos mesmos, através dos padrões nível 1 (requisitos de Hardware) e EMV nível 2 (requisitos de software). O CAP¹⁰ define as especificações dos leitores.

3.1.2 Assinatura

Numa assinatura tradicional (assinatura autógrafa) pretende-se garantir que uma determinada entidade é responsável pela autoria ou aprovação de um documento em papel. Nas assinaturas digitais pretende-se manter o mesmo significado e o mesmo grau de importância. A diferença reside no modo de assinatura, enquanto a assinatura tradicional é realizada com a caneta a assinatura digital faz uso das chaves digitais e da criptografia assimétrica¹¹. Embora construída de maneiras diversas, ambas têm um único fim, ligar univocamente uma entidade a um documento.

Segundo [2] a assinatura digital é um processo de assinatura electrónica baseado em sistema criptográfico assimétrico composto por um algoritmo ou série de algoritmos, mediante o qual é gerado um par de chaves assimétricas

⁹One Time Password

¹⁰Chip Authentication Program

¹¹Ver secção 3.1.2

exclusivas e interdependentes, uma privada e outra pública, e que permite ao titular usar a chave privada para declarar a autoria do documento electrónico no qual a assinatura é construída em concordância com o seu conteúdo. E ao declaratório usar a chave pública para verificar se a assinatura foi criada mediante o uso da correspondente chave privada ou se o documento electrónico foi alterado depois de construída a assinatura.

Com assinatura digital garante-se um nível de segurança superior ao oferecido pela assinatura tradicional, já que esta última não garante o controle de alterações do documento. Mas uma assinatura digital não é impossível de forjar, basta para isso perder o controlo da chave privada. Se alguém conseguir obter a chave privada de outrem, pode assinar documentos em vez deste sem que as outras entidades se apercebam da falsificação.

Para melhor entender o que está na base da assinatura digital é necessário compreender alguns conceitos de criptografia. As próximas secções focarão este ponto. O tema da assinatura digital volta a ser abordado nas secções de assinatura digital em XML e assinatura XAdES.

Criptografia

A palavra criptografia surgiu da junção de duas palavras gregas “*Kryptós*” e “*gráphein*” e que significam “*secreto*” e “*escrita*” respectivamente. Tal como diz o seu significado quando se faz uso da criptografia pretende-se manter o segredo da escrita, de algo, para que só entidades autorizadas a consigam interpretar. A criptografia é usada desde os tempos dos antigos Egípcios.

Nos dias hoje, a criptografia tem um papel fundamental nas mais diversificadas áreas, que vão desde áreas militares às áreas financeiras e comerciais. Em todas elas, com o mesmo objectivo, garantir a confidencialidade de dados e das comunicações. Nos últimos anos, temos assistido a

grande evoluções nesta área, empolgadas em grande parte pelas necessidades de aplicações comerciais.

Ao longo das próximas secções será usada a terminologia de “emissor” para designar a entidade que cria a informação e “receptor” ao responsável pelo destino da mesma. Para a informação, será utilizado o termo “texto original”, e ao texto que cifrado e para o qual se pretenda manter o segredo será designado por “texto cifrado”, ou simplesmente “cifra”¹².

Para transformar o texto original no texto cifrado é utilizado um algoritmo de cifra, que faz uso de uma chave para transformar o texto, quer seja para cifrar como para decifrar. Existem dois grandes grupos de algoritmos, os algoritmos de chave simétrica e os de chave assimétrica e serão ser descritos nas próximas secções.

Criptografia de chave simétrica

Na criptografia de chave simétrica, são usados algoritmos onde a chave (o segredo) utilizada para reverter o texto cifrado no texto original poder ser calculada a partir da chave utilizada para cifrar o mesmo (e vice-versa). A grande especificidade deste tipo de algoritmos, é a utilização de uma única chave para a execução de todo o processo (a cifra e a posterior decifra). A segurança deste tipo de algoritmos reside na privacidade da chave usada pelo o emissor e receptor. O extravio desta pode colocar em causa a segurança do mesmo. Uma terceira entidade que não seja nem o emissor nem o receptor e que tenha acesso à chave vai conseguir decifrar e cifrar dados, como sendo um deles.

A utilização deste tipo de algoritmos obriga a um conhecimento prévio da chave a ser usada antes de os dados serem cifrados. Uma das maneiras mais seguras de distribuição de chave é através do contacto directo entre o

¹²Também pode ser usado “encriptação” e “encriptar”

receptor e emissor, antes de se iniciar qualquer troca de dados cifrados. Nem sempre esta possibilidade existe, os intervenientes podem estar distantes geograficamente ou podem nem ser pessoas (i.e., programa de computadores a comunicar entre si). O tipo de problema apresentado pode ser resolvido através da utilização de um protocolo de troca de chave[13]. Ainda existe outro tipo de problema que poderá surgir e está relacionado com a garantia de que a entidade a quem pertence a chave é a entidade à qual é suposto pertencer. A solução deste problema pode ser encontrada nos certificados digitais.

O uso deste tipo de algoritmos é normalmente muito eficiente em termos computacionais, sendo este usado para cifrar grande volume de dados.

Um dos algoritmos mais antigo e mais usado, e entre os algoritmos de chave simétrica, é o DES¹³[33]. Este foi desenvolvido na década de 70 pela IBM, tendo sido adoptado pelo governo americano como norma para criptografia (não militar) com chave secreta. Contudo, com o passar do tempo este apresentou algumas fragilidades, desde então a preferência tem decaído a favor de outro algoritmo.

Esse algoritmo é designado por AES¹⁴[34] e foi desenvolvido em 2001 por dois belgas. Recentemente a sua segurança começou a ser questionada no campo científico e levantaram-se questões de natureza puramente teórica, não se esperando, no entanto, que brevemente sejam descobertas formas práticas de atacar a segurança deste algoritmo. Este algoritmo faz uso de chaves de 128, 192 e 256 bits. As operações realizadas sobre o conjunto de dados a cifrar são simples operações matriciais, nas quais, cada passo pode ser facilmente reversível, desde que se possua a chave. Este algoritmo apresenta também como vantagem os baixos requisitos computacionais para

¹³Data Encryption Standard

¹⁴Advanced Encrypted Standard

a sua utilização.

Actualmente, NIST¹⁵[35] desaconselha a utilização do algoritmo DES. Este ser usado, mas apenas como componente do Triple DES¹⁶[8] e mesmo a utilização deste deve ser gradualmente substituída pela a utilização do AES.

Criptografia de chave pública

A criptografia de chave pública [16] faz uso de duas chaves complementares, sendo uma privada e outra pública. A chave pública pode ser de conhecimento público, sem que com isso exista perda de segurança. Esta é usada para cifrar e validar a autenticidade de dados. Já a chave privada deve ser mantida secreta, protegida e só a usar quem lhe tem direito. Esta é usada para decifrar e autenticar dados. O par de chaves geradas simultâneamente tem como principal característica a complementaridade. Para a geração da mesma recorre-se a métodos matemáticos complexos.

Existem vários métodos que respondem a este tipo de algoritmo, como o RSA, ElGamal e DSS. O mais usado e mais estudado em todo o mundo é o RSA[30]. Apareceu nos finais da década de 70 e foi inventado por Ronald Rives, Adi Shamir e Lenny Addleman. Segundo os seus criadores [37] a segurança deste tipo de algoritmos está na realização de operações pertencentes ao grupo da exponenciação e nos logaritmos discretos.

Certificados digitais

Um certificado digital é um documento electrónico que liga uma chave privada a uma entidade. Este tem uma estrutura predefinida e é assinado pela entidade emissora, entidades estas que são designadas por Autoridades Certificadoras (CA¹⁷). A assinatura sobre o certificado tem como objectivo

¹⁵National Institute of Standards and Technology

¹⁶ou 3DES

¹⁷Certification Authority

garantir a integridade da chave pública assim como a sua autenticidade. Os certificados podem ser distribuídos livremente (seja por canais seguros ou inseguros), sem correr qualquer risco do ponto de vista de segurança.

Existem alguns tipos de certificados digitais. Os mais conhecidos e divulgados são o X.509 [29] o SPKI [24, 23] e o openPGP [10]¹⁸.

O X509 é sem dúvida o mais utilizado, surgiu em 1988 desenvolvido pela ITU-T e pela ISO. Este derivava das recomendações X.500, e ficou registado como versão 1¹⁹. Em 1996 o formato V1 foi substituído pela segunda versão²⁰ e esta veio adicionar dois elementos ao certificado, um identificador único da entidade detentora do certificado e um identificador único da CA responsável pela emissão do mesmo. A evolução mostrou que este dois campos não preenchiam todas as necessidades. Então em 1996 surge a terceira versão²¹, que tem como novidade a incorporação de extensões em relação ao formato base. Actualmente, existem inúmeras definições padrão definidas. Além destas, o certificado é constituído, como já foi referido, por uma informação base, da qual fazem parte as seguintes características:

1. Versão do certificado (V1, V2, V3);
2. Número de série;
3. Algoritmo usado pela CA para assinar digitalmente o certificado;
4. Nome identificativo da CA responsável pela emissão;
5. Validade do certificado (data de início e data do fim);
6. Nome da entidade detentora do certificado;

¹⁸standard aberto baseado no PGP(Pretty Good Privacy)

¹⁹V1

²⁰V2

²¹V3

7. Chave pública;
8. Assinatura do certificado pela chave privada da CA;

O certificado SPKI²², tinha como objectivo substituir o X.509 devido aos seus problemas, relacionados com a complexidade e escalabilidade. No entanto, esta sobreposição nunca se veio a constatar, sendo os certificados X509 os mais procurados do mercado.

Os certificados openPGP seguem standards abertos e são baseados no PGP²³ que tem como alvo o uso do correio electrónico. Estes não são gerados por CA's mas sim pelos próprios indivíduos que os usam. Este tipo de certificado tem como vantagem o baixo custo, já que qualquer entidade os pode gerar. No entanto, quando entramos no campo empresarial, estes certificados não oferecem as garantias mínimas necessárias para o seu uso.

Para ser qualificado, o certificado digital tem de ser emitido por uma entidade certificadora credenciada. Este tipo de certificado quando utilizado para assinar um documento electrónico equivale, para efeitos legais, a uma assinatura manuscrita.

Autoridade de certificação

A autoridade de certificação é responsável por todos os estados de um certificado, desde a sua emissão, passando pela suspensão, revogação ou re-emissão. As CA são também responsáveis por definir as políticas de criação e gestão dos certificados, bem como, manter acessível esta mesma informação (CPS²⁴). Outra das funções é manter actualizadas as listas de revogação, e oferecer um serviço para que essa informação se torne acessível o que pode ser feito através de CRL ou OCSP ²⁵.

²²Simple Public Key Infrastructure

²³Pretty Good Privacy

²⁴Certificate Practice Statement

²⁵O funcionamento das CRL's e OCSP's já foi descritas anteriormente

As CA estão organizadas numa estrutura hierárquica. A hierarquia possui um certificado de topo, que é assinado por ele próprio, segue-se depois para diferentes níveis em que cada certificado está assinado por o de nível acima. Normalmente, os certificados de topo correspondem às CA de “confiança”, confiança essa atribuída por exemplo, pelos browsers , Java, SO e pretendem estabelecer uma primeira filtragem, sendo que a decisão final deve ser sempre feita pelo utilizador.

Autoridade de selos temporais

A autoridade de selos temporais (TSA²⁶) é responsável pela geração de selos temporais (estes podem também ser denominados de estampilha temporal, carimbo temporal ou timestamp). A validação cronológica de dados é feita através da associação de informação temporal aos dados que se pretendem validar, de forma inequívoca através de uma assinatura digital de uma entidade especializada, a TSA. De frisar, que a TSA não contém a totalidade da informação, apenas é enviada para esta um sumário²⁷ e é sobre este que é realizado a assinatura com a chave privada do TSA.

É importante realçar qual a fonte temporal que é usada para obter o selo temporal. A TSA pode fazer uso do seu próprio relógio ou recorrer a uma autoridade para a hora legal da área geográfica onde actua. A hora legal de Portugal é mantida pelo Observatório Astronómico de Lisboa (OAL) por meio de um conjunto de relógio atómicos. Estes encontram-se inseridos numa rede mundial gerida pelo *Bureau International des Poids et*

²⁶Time-Stamping Authority

²⁷Também pode ser designado por Message Digest ou por hash. O algoritmo mais usado hoje em dia é SHA1[32] e tem como principais características: o resultado do hash terá sempre o mesmo número de byte seja qual for o tamanho do documento; para dois documentos em que apenas muda um bit o resultado final terá de ser diferente e o resultado final terá de ser fácil de calcular já o seu inverso deverá ser impossível.

Mesures para definição do padrão mundial UTC ²⁸. Se uma TSA obter o valor temporal através desta rede seguindo todos os padrões, oferece um serviço com melhor qualidade e maior independência.

Assinatura digital XML

Após uma abordagem teórica, sobre algoritmos simétricos e assimétricos, certificados digitais entre outros conceitos, serão agora abordadas algumas características que as assinaturas digitais garantem e como estas se realizam.

Uma assinatura digital básica é realizada em duas fases. Numa primeira fase, é utilizado um algoritmo de sumário para obter um sumário da mensagem. Posteriormente, o sumário é cifrado com a chave privada do emissor, sendo em seguida enviada para o receptor o documento original juntamente com o sumário cifrado. Este dois juntos designam-se por assinatura digital. O receptor recebe a assinatura digital e valida a mesma. Com a chave pública decifra o sumário do documento. Por último, usando o mesmo algoritmo de sumário que foi utilizado pelo emissor obtém-se um novo sumário. Se ambos coincidirem, o receptor conclui que a assinatura digital é válida. A figura 3.3 ilustra este processo.

As garantias de segurança dadas pela assinatura acima descrita são as seguintes:

1. **Integridade** - Como é utilizado um sumário, que é baseado numa função matemática invertível e que para dois documentos idênticos o sumário gerado é sempre diferente, a comparação final do sumário garante que o documento não sofreu quaisquer alterações;
2. **Autenticação** - Com a integridade da mensagem não é obrigatório que se conheça a entidade emissora. Nestes casos, recorre-se à autenticação para obter garantias fortes de que os dados provêm da entidade emissora pretendida. Como já foi visto na secção 3.1.2, este ponto é

²⁸Universal Coordinated Time

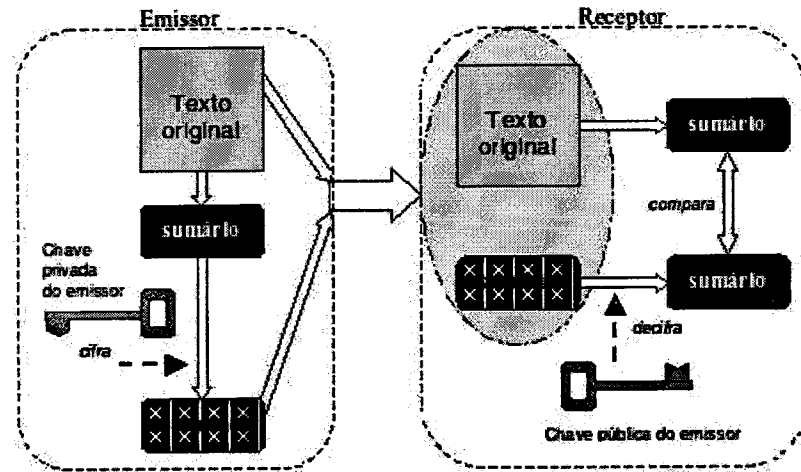


Figura 3.3: Geração e verificação de uma assinatura digital (extraído de [38])

facilmente resolvido através da utilização de certificados digitais emitidos por CA's competentes e reconhecidas. Do certificado é retirada a chave pública e decifrado o conteúdo do sumário, se a comparação for realizada com sucesso. Logo, só a chave privada correspondente à chave pública pode ter sido usada para cifrar o sumário do emissor. Pode-se então afirmar que foi este o responsável pela assinatura.

3. **Não-Repúdio** - O não-repúdio consiste em garantir que o emissor não pode negar a sua participação na assinatura do documento. Partindo do princípio que só o emissor tem acesso à sua chave privada, quando se consegue estabelecer a autenticação do emissor através da relação da chave pública com a privada não pode ser negado o seu conhecimento no momento de assinar. É então de realçar que é necessário ter a certeza que a chave pública pertence realmente ao emissor, sendo por isso necessário ter toda a confiança na CA.

O XML oferece um conjunto de propriedades que leva a que este seja considerado standard devido ao grande número de formatos e tecnologias que oferece a uma estrutura de suporte. Um bom exemplo, será, o uso do

protocolo SOAP²⁹ usado nos *web-services*, que fazendo uso de standards XML garante a interoperabilidade entre diferentes tecnologias (i.e., Java e .NET).

Segundo [41], uma assinatura digital resume-se ao acto de adicionar um valor calculado computacionalmente, através de um algoritmo de cifra, ao objecto de dados, de maneira a que os mesmos possam ser verificados, tanto a nível de originalidade como da sua integridade. As assinaturas digitais XML (XMLDSIG³⁰) seguem esta filosofia e surgiram com o intuito de garantir a segurança nas transacções que utilizam o XML, como num conjunto repleto de outros formatos de assinaturas digitais, que garantem a integridade, autenticação e não-repúdio. De seguida, será apresentada a estrutura de uma assinatura digital em XML em que o símbolo “?” denota o uso de zero ou uma ocorrência, o “+” representa o uso de uma ou mais ocorrências e o “*” denota o aparecimento de zero ou mais ocorrências.

```
<Signature ID?>
----<SignedInfo>
-----<CanonicalizationMethod/>
-----<SignatureMethod/>
-----(<Reference URI? >
-----(<Transforms>)?
-----<DigestMethod>
-----<DigestValue>
-----</Reference>)+
----</SignedInfo>
----<SignatureValue>
-----(<KeyInfo>)?
-----(<Object ID?>)*
</Signature>
```

Será agora feita uma breve descrição sobre os elementos que fazem parte da estrutura acima apresentada:

1. *Signature* - É o elemento que representa a assinatura digitalizada. A sua ocorrência pode surgir por diversas vezes num documento. Uma

²⁹Simple Object Access Protocol

³⁰XML Digital Signatures

assinatura não necessita de ser realizada a todo um documento pode ser apenas a uma parte deste. Estas várias assinaturas podem ser realizadas em momentos distintos e por diferentes entidades.

2. *SignedInfo* - Neste elemento é apresentada a informação sobre a qual a assinatura é realizada, bem como os métodos e protocolos usados para alcançar o valor da assinatura final. Deste fazem parte os elementos *CanonicalizationMethod*, *SignatureMethod* e *Reference*.
3. *CanonicalizationMethod* - Especifica o algoritmo usado no elemento *SignedInfo* antes de ser efectuado o cálculo da assinatura, para que os dados possam ser processados de forma independente. O método de canonização *default* é denominado XML canónico. O método de canonização consiste no processo de conversão de dados que são apresentados em mais do que uma representação possível para uma representação canónica padrão. São exemplos de algoritmos de canonização, o XML-C14N, XML-C14N11, CRLF entre outros.
4. *SignatureMethod* - É um elemento obrigatório. Define, qual foi o algoritmo usado na realização da assinatura e que será utilizado para a validação da mesma, por exemplo, algoritmos de chave pública, *padding*, algoritmos de sumário ou códigos de autenticação de mensagens (MAC³¹). Existe um algoritmo que todas as aplicações deverão reconhecer: DSA com SHA1. O método de codificação de mensagem estabelecida na especificação do W3C é o Base64. Contudo, também é permitido, a utilização de algoritmos de codificação de mensagem com as suas próprias codificações.
5. *Reference* - Pode ocorrer uma ou mais vezes. Neste elemento, é especificado qual o valor e o algoritmo de sumário. Além destes dados, existe uma lista de transformações que podem ser realizadas antes da obtenção do sumário. O algoritmo de transformação mostra como é

³¹Message Authentication Codes

possível chegar ao sumário criado. Fazem parte deste elemento três atributos: Id, URI e Type. O URI identifica o objecto sobre o qual está a ser realizado a assinatura. Pode ser URLfootnote³² de um ficheiro externo ou a localização de uma porção de dados localizados dentro do próprio ficheiro de XML. Se o atributo descrito atrás for uma String vazia (URI="") está a ser a raiz do documento XML. Se esta referência for omitida a aplicação deverá ter conhecimento prévio de quais os dados sobre o qual se realizará a assinatura. Fazem parte de Reference os elementos *Transforms*, *DigestMethod* e *DigestValue*.

6. *Transforms* - Este elemento é opcional e pode ser repetido, formando assim uma lista de transformações. Estas transformações respeitam uma ordem e o resultado de uma transformação será o *input* da próxima transformação. O resultado da última transformação é o *input* para a sumarização. Alguns exemplos de algoritmos de transformação: *canonicalization*, *Base64*, *Xpath Filtering*, *Enveloped Signature Transform* e *XSLT Transform*.
7. *DigestMethod* - É um elemento obrigatório e é responsável por indicar qual o algoritmo de sumário aplicado sobre o objecto a assinar.
8. *DigestValue* - É também um elemento obrigatório que contém o valor de sumário codificado. A codificação usada é sempre Base64.
9. *Signature Value* - Contém o valor da assinatura de acordo com a codificação especificada no elemento *SignatureMethod*.
10. *KeyInfo* - Este é um elemento opcional e permite que o receptor tenha acesso à(s) chave(s) para a validação da assinatura. Este elemento pode conter chaves, certificados, nomes, e outra informações relacionadas com chave pública. Se este elemento não fizer parte da assinatura parte-se do princípio que o receptor tenha o conhecimento prévio da(s) chave(s) a usar para validar a assinatura.

³²Uniform Resource Locater

11. *Object* -Se o tipo de assinatura a realizar for do tipo *enveloping*³³, este elemento estará presente e representa o objecto sobre o qual recai a assinatura. Este elemento é opcional contudo pode ocorrer uma ou mais vezes.

Existem três possibilidades de representar um documento numa assinatura digital: *Enveloping*, *Enveloped* e *Detached*. Estamos perante uma assinatura *enveloping* quando a assinatura engloba os dados assinados (neste tipo de assinatura o elemento *object* necessita de estar presente). Na *enveloped* acontece precisamente o contrário, o documento é que contém as assinaturas. Estas são realizadas sobre porções de dados. Pode aparecer mais que uma assinatura ao longo de um mesmo documento. Por fim, a assinatura *detached* que se refere a dados externos ao ficheiro onde se encontra a assinatura. Essas referências normalmente apontam para documentos acessíveis através da *web*.

Assinatura Electrónica Avançada XML

A directiva 1999/93/EC [22] do Parlamento e Conselho Europeu define uma base legal para assinaturas electrónicas. Nesta directiva é feita a distinção entre dois tipos de assinaturas, as assinaturas electrónicas normais e as assinaturas electrónicas avançadas. As assinaturas electrónicas normais, têm uma validade legal reduzida, enquanto que nas assinaturas electrónicas avançadas é exigido que satisfaçam os mesmos requisitos legais em relação à informação electrónica, que uma assinatura escrita satisfaz em relação à informação no papel.

³³os dados assinados encontram-se dentro de um elemento *Object*, ou seja, os dados estão incluídos dentro da própria assinatura

XAdES

Com o constante crescimento da utilização do XML como *standard* o ETSI³⁴ tem vindo a desenvolver as especificações de assinaturas XML-XAdES³⁵. A XAdES além de garantir os requisitos exigidos pela directiva da União Europeia no que toca a assinaturas electrónicas avançadas, ainda garante os requisitos para o não-repúdio e validade de longo prazo. Com esta assinatura, será possível provar a validade da assinatura, mesmo que o emissor tente mais tarde repudiar a validade da mesma. A XAdES consegue provar eventuais disputas entre o assinante e a entidade verificadora, mesmo que esta ocorra vários anos depois da data de assinatura.

A XAdES é constituída por vários níveis. O mais básico é denominado por XAdES – BES³⁶ e é construído a partir de uma assinatura XML (XMLDSIG) adicionando informação sobre a assinatura e os dados assinados. Esta informação adicional é acrescentada à XMLDSIG como um elemento *Object*, continuando a XAdES a ser uma XMLDSIG válida. Para transformar uma assinatura XML numa XAdES-BES é necessário que seja adicionada a data na qual foi criada a assinatura, a referência para o certificado a que corresponde a chave privada, que foi utilizada na assinatura (no elemento *SigningCertificate* ou no *keyInfo*) e a referência para a política de assinatura que se aplica à assinatura.

Neste nível ainda podem fazer parte da assinatura os seguintes elementos : *SigningTime*, *DataObjectFormat*, *CommitmentTypeIndication*, *SignerRole*, *SignatureProductionPlace*, um ou mais *IndividualDataObjectTimeStamp* ou um *AllDataObjectTimeStamp*, *SignaturePolicyIdentifier* e/ou um *CounterSignature*. Se da assinatura XAdES-BES fizer parte o elemento *SignaturePolicyIdentifier* esta passa a ser denominada de XAdES-EPES³⁷ Ao

³⁴European Telecommunications Standard Institute

³⁵XAdES- XML Advanced Electronic Signature

³⁶BES – Basic Electronic Signature

³⁷EPES - *Explicit Policy based Electronic Signature*

nível mais básico da XAdES pode ser incrementada informação que aumenta o nível de segurança. Estão especificados nos standards os seguintes níveis adicionais: XAdES-T, XAdES-C, XAdES-X, XAdES-X-L e XAdES-A. Estes níveis serão abordados nas próximas secções.

XAdES-T

Uma propriedade importante de uma assinatura de longo prazo, é que, uma vez considerada válida, a assinatura permaneça válida durante meses ou anos. Como a informação criptográfica que serve de base à construção da assinatura poderá mais tarde ser comprometida, a este nível é adicionado o elemento *SignatureTimeStamp* que contém um selo temporal produzido por uma entidade externa, provando assim, o não-repúdio da existência da assinatura numa determinada data.

XAdES-C

Este nível adiciona ao anterior XAdES-T 3.1.2 referências para toda a informação necessária de modo a que seja possível validar a assinatura. Estas referências incluem informação acerca dos certificados necessários para validar a assinatura, incluindo referências para as cadeias de certificação e para a informação de revogação³⁸ correspondente. Esta informação será adicionada nos seguintes elementos *CompleteCertificateRefs* com referências para a cadeia de certificação e *CompleteRevocationRefs* com referências para a informação de revogação correspondente.

XAdES-X

Este nível garante segurança na validação a longo prazo sobre a informação do nível anterior 3.1.2. Com a adição de um selo temporal sobre toda a assinatura, com a adição do elemento *SigAndRefsTimeStamp* ou apenas sobre o nível XAdES-C com a adição do elemento *RefsOnlyTimeStamp* é garantido

³⁸ CRL ou OCSP

que o estado das cadeias de certificação e informação de revogação era válido na data do selo temporal. No caso da quebra do segredo da CA emissora de um certificado pertencente à cadeia de certificação, a validade da assinatura mantém-se, desde que a data desse evento seja posterior à data do selo temporal.

XAdES-X-L

Como no futuro poderá não ser possível obter a informação de validação referenciada no nível XAdES-X (caso por exemplo as referências especifiquem endereços de internet desactualizados), este nível adiciona à assinatura toda a informação de validação referenciada pelo nível XAdES-X. Os elementos onde se encontram os certificados correspondentes à cadeia de certificação e a informação de revogação são *CertificatesValues* e *RevocationValues* respectivamente.

XAdES-A

Este nível adiciona finalmente à assinatura um selo temporal sobre toda a informação de validação adicionada no XAdES-X-L, protegendo assim a assinatura a longo prazo contra algoritmos de segurança que se venham a tornar fracos e contra pares de chaves criptográficas que se tornem conhecidas. Este nível pode adicionar sucessivos selos temporais no futuro sempre que se torne necessário (nomeadamente, para protecção dos selos temporais anteriores relativos ao mesmo nível). O elemento onde se integra o selo temporal descrito atrás é o *ArchiveTimeStam*.

A estrutura abaixo mostra quais os elementos que fazem parte de cada nível e como estão organizados, é mostrado também quais os elementos que fazem parte da XMLDISG. É importante realçar que os elementos que fazem parte do elemento *SignedProperties* são propriedades incluídas nas referências da assinatura e, conseqüentemente, assinadas pelo algoritmo de assinatura. Os elementos que fazem parte do elemento *UnsignedProperties* não entram para

o cálculo da assinatura e podem ser adicionados posteriormente.

```

                                XMLDISG
                                |
<ds:Signature ID?>- - - - - + - - - - + + + + + + + +
  <ds:SignedInfo>                | | | | | | |
    <ds:CanonicalizationMethod/> | | | | | | |
    <ds:SignatureMethod/>        | | | | | | |
    (<ds:Reference (URI=? >    | | | | | | |
      (<ds:Transforms>)?       | | | | | | |
      <ds:DigestMethod>        | | | | | | |
      <ds:DigestValue>         | | | | | | |
    </ds:Reference>)+          | | | | | | |
  </ds:SignedInfo>              | | | | | | |
  <ds:SignatureValue>           | | | | | | |
  (<ds:KeyInfo>)? - - - - - + | | | | | | |
  <ds:Object>                   | | | | | | |
                                  | | | | | | |
  <QualifyingProperties>        | | | | | | |
                                  | | | | | | |
  <SignedProperties>            | | | | | | |
                                  | | | | | | |
    <SignedSignatureProperties>  | | | | | | |
      (SigningTime)             | | | | | | |
      (SigningCertificate)      | | | | | | |
      (SignaturePolicyIdentifier) | | | | | | |
      (SignatureProductionPlace)? | | | | | | |
      (SignerRole)?             | | | | | | |
    </SignedSignatureProperties> | | | | | | |
                                  | | | | | | |
    <SignedDataObjectProperties> | | | | | | |
      (DataObjectFormat)*       | | | | | | |
      (CommitmentTypeIndication)* | | | | | | |
      (AllDataObjectsTimeStamp)* | | | | | | |
      (IndividualDataObjectsTimeStamp)* | | | | | | |
    </SignedDataObjectPropertiesSigned> | | | | | | |
                                  | | | | | | |
  </SignedProperties>           | | | | | | |
                                  | | | | | | |
  <UnsignedProperties>          | | | | | | |
                                  | | | | | | |
    <UnsignedSignatureProperties> | | | | | | |
      (CounterSignature)*- - - - - + | | | | | | |
      (SignatureTimeStamp)+- - - - - + | | | | | | |
      (CompleteCertificateRefs) | | | | | | |
      (CompleteRevocationRefs)- - - - - + | | | | | | |
      ((SigAndRefsTimeStamp)* | | | | | | |

```



```

    (RefsOnlyTimeStamp)*- - - - - + | |
    (CertificatesValues)          | |
    (RevocationValues)- - - - - + | |
    (ArchiveTimeStamp)+          | |
    </UnsignedSignatureProperties>- - - + + + + + | |
                                     | | | | | |
</UnsignedProperties>              | | | | | |
                                     | | | | | |
</QualifyingProperties>           | | | | | |
                                     | | | | | |
</ds:Object>                       | | | | | |
                                     | | | | | |
</ds:Signature>- - - - - + + + + + + +
                                     | | | | | |
                                     XAdES | | | | |
                                     | | | | |
                                     XAdES-T | | | | |
                                     | | | | |
                                     XAdES-C | | | | |
                                     | | | | |
                                     XAdES-X | | | | |
                                     | | | | |
                                     XAdES-X-L | | | | |
                                     | | | | |
                                     XAdES-A

```

Os níveis podem ser construídos de forma incremental pela ordem pela qual foram apresentados. No entanto, a existência de uma assinatura de um determinado nível não implica que todos os níveis abaixo tenham de ser implementados. Por exemplo, a XAdES-X não necessita da propriedade específica de XAdES-T.

3.2 Cartão do Cidadão

“ O Cartão de Cidadão vem revolucionar a forma como o cidadão se pode relacionar com diversas Entidades (Públicas ou Privadas). ”

A 19 de Janeiro de 2007 é promulgada o Decreto Lei 7/2007 que torna o cartão de cidadão (figura 3.4) uma realidade em Portugal.

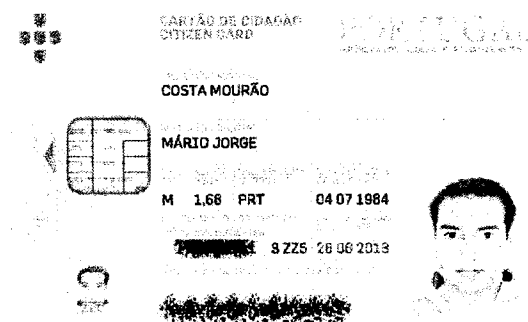


Figura 3.4: Cartão de cidadão

Com o aparecimento do cartão de cidadão surgiram oportunidades para desenvolvimento de novas aplicações e formação de diferentes áreas de estudo. A tabela 3.2 ilustra as diferenças entre o cartão de cidadão e Bilhete de Identidade. As principais diferenças estão relacionadas com a passagem de toda a informação visível para um único chip. Além de conter informação que identifica inequivocamente um cidadão português, o chip ainda se distingue pela presença de:

- certificado qualificado para assinatura electrónica qualificada;
- certificado para autenticação segura;
- impressões digitais;
- morada;
- cartão de contribuinte;
- cartão de utente;
- carta de eleitor;

- entre outras.

De notar que toda a informação visível no cartão de cidadão está contida no chip à excepção da imagem da assinatura digitalizada.

3.2.1 O chip

Toda a informação e aplicações que fazem parte do cartão de cidadão estão contidas no chip inserido neste. O chip é um Chip JavaCard, Samsung S3CC9TC, com 73Kb de EEPROM para aplicações e dados. Este está protegido com vários mecanismos de segurança, tanto a nível de algoritmo e cifras, como na protecção contra ataques. Conta com a certificação FIPS140-2 level 3, com aplicações IAS certificadas SSCD³⁹. O chip cumpre também a norma EMV.

O chip está dividido em duas áreas, uma destas é responsável pelo armazenamento dos dados do cidadão, o qual foi apresentada no capítulo anterior, a outra área contém aplicações que permitem a execução das seguintes funcionalidades:

- **IAS** - aplicação responsável pelas operações de autenticação e assinatura electrónica[18];
- **EMV-CAP** - aplicação responsável pela geração de palavras-chave únicas por canais alternativos (i.e, telefone)[18];
- **Match-on-card** - aplicação responsável pela verificação biométrica de impressões digitais.

A área responsável pelo armazenamento dos dados do cidadão contém :

- Templates biométricos de Impressão digitais. Estes dados não estão acessíveis;

³⁹Secure Signature-Creation Device

- Dados identificativos do cidadão (ver tabela 3.2);
- A fotografia;
- Uma área de uso pessoal⁴⁰.
- O certificado digital de autenticação e o certificado digital de assinatura são também de acesso público, contudo a sua utilização é protegida por PIN.
- O acesso à morada também se encontra protegida por PIN.

A figura 3.5 demonstra a organização do chip. Nas próximas secções será feita uma análise mais aprofundada sobre estas aplicações.

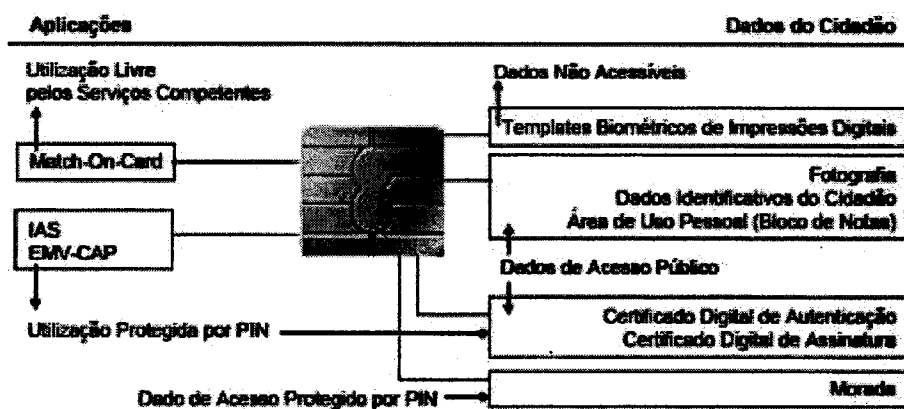


Figura 3.5: Chip (extraído de[20]).

3.2.2 Autenticação com cartão do cidadão

O cartão de cidadão oferece duas soluções para a sua autenticação. Uma das soluções é pensada para ser utilizada em ambiente *web*, e a outra é vocacionada para ambientes alternativos. Para a autenticação on-line o cartão

⁴⁰Bloco de notas

de cidadão disponibiliza o certificado digital de autenticação, o qual identifica univocamente um cidadão e permite o acesso a serviços electrónicos de forma segura. Os modelos de acessos nesta vertente podem ser lidos no capítulo anterior e deles fazem parte a autenticação mútua via *web-service* e a autenticação via browser usando como base a especificações do SSL. A outra solução que pode ser encontrada no cartão permite uma autenticação por canais alternativos e é o EMV-CAP. O seu funcionamento é idêntico ao descrito na secção 3.1.1 com a variante de que os dados são transferidos via telefone. Um cidadão liga para um *contact center* de onde são solicitados os seus dados, o cidadão gera o *token* CAP (sequência de dígitos) gerado pela aplicação EMV-CAP presente no cartão e fornece os seus dados. Em seguida é feita a validação por parte do *contact center* junto do sistema de validação tokens-CAP, o qual devolve a resposta acerca da realização da autenticação, ou seja, se esta foi ou não realizada com sucesso. O esquema respectivo a esta informação, pode ser observado na figura 3.6.

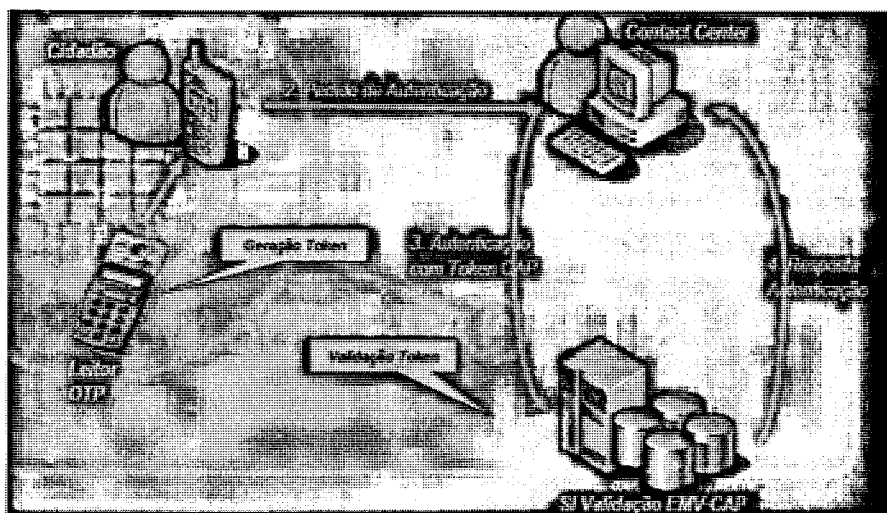


Figura 3.6: Autenticação EMV-CAP com cartão de cidadão

3.2.3 Assinatura com cartão de cidadão

Com a criação do Sistema de Certificação Electrónica do Estado (SCEE) e do cartão de cidadão começa a existir uma massificação de certificados digitais pessoais, que até hoje não se verificava. Com esta alteração, as assinaturas digitais ganham outra relevância. Existem cada vez mais sistemas baseados nesta tecnologia.

O certificado digital é um documento electrónico que liga os dados de verificação de assinatura ao seu titular e confirma a identidade desse titular. Para ser qualificado, o certificado digital tem de ser emitido por uma entidade certificadora credenciada. Este tipo de certificado quando utilizado para assinar um documento electrónico equivale, para efeitos legais, a uma assinatura manuscrita. Para emitir certificados digitais qualificados a entidade que emite o certificado terá de estar credenciada junto da entidade responsável pela credenciação de empresas certificadoras.

Ainda são poucas as empresas certificadoras que estejam credenciadas em Portugal. O cartão de cidadão vem garantir a todos os seus portadores certificados digitais qualificados que asseguram a identidade dos indivíduos e permitem a assinatura electrónica com efeitos legais [26].

As CA's credenciadas para a emissão de certificados digitais qualificados são as seguintes [15]:

1. ECCE⁴¹;
2. EC de Assinatura Digital Qualificada do Cartão de Cidadão 0001;
3. EC de Assinatura Digital Qualificada do Cartão de Cidadão 0002;
4. EC de Assinatura Digital Qualificada do Cartão de Cidadão 0003;

⁴¹Entidade de Certificação Electrónica do Estado

5. ECAR⁴²;
6. Justiça⁴³;
7. MULTICERT - Entidade de Certificação 001⁴⁴;
8. BT/DigitalSign Qualified CA⁴⁵;

Aos portadores de certificados digitais qualificados é recomendado que nunca se empreste ou ceda o seu cartão criptográfico a terceiros, nunca se partilhe o seu código PIN com ninguém e que se evite guardá-lo de forma pouco segura⁴⁶. Se por algum motivo houver uma suspeita de que alguém teve conhecimento do seu código PIN é aconselhável mudar na primeira oportunidade. Caso se perca o cartão ou se extravie é aconselhável comunicar ao ECCE. A importância de respeitar estas regras surge pelo facto do titular do certificado ser responsabilizado em caso de quebra de confiança.

⁴²Entidade Certificadora da Assembleia da República

⁴³Entidade Certificadora do Ministério da Justiça

⁴⁴Multicert - Serviços de Certificação Electrónica, S.A

⁴⁵British Telecommunications plc

⁴⁶post-it, papeis na carteira,etc

Conteúdo	Bilhete de Identidade	cartão de cidadão
Imagem Facial	Fotografia a cores	Imagem impressa P/B
Impressão Digital	Impressão directa (Impressões Digitais)	Só no Chip (minúcias de duas Impressão Digital)
Assinatura do titular	Manuscrita	Imagem da assinatura
Nº de identificação civil	X	X
Nº de documento	–	X
Data de emissão	X	–
Local de emissão	X	–
Nome	X X X	X (distinção entre nomes próprios e apelidos)
Filiação	X	X
Naturalidade	X	–
Residência	X (Freguesia – Concelho)	Morada não visível. Completa no Chip. (Necessita de autorização do titular para visualização)
Data de Nascimento	X	X
Estado Civil	X	–
Altura	X	X
Validade	X	X
Sexo	–	X
Nacionalidade	Fixa (Informação pré impressa no documento)	X (Variável)
Nº Identificação Fiscal	–	X
Nº Segurança Social	–	X
Nº Utente de Saúde	–	X
Zona de Leitura óptica	–	X
Bloco de Notas	–	X (Leitura livre, escrita limitada ao titular)

Tabela 3.1: Bi Vs cartão de cidadão [20]

Capítulo 4

Solução proposta

Neste capítulo é apresentada uma solução que dá resposta aos objectivos propostos. Na secção 4.1 será apresentada a arquitectura, constituída por dois módulos principais, a API e a Interface do cliente (figura 4.1). Estes

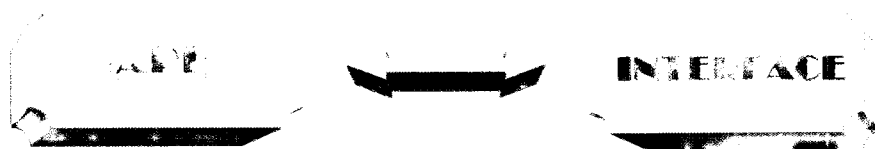


Figura 4.1: Arquitectura da API e da Interface de cliente

subdividem-se em pequenos módulos que vão ser aprofundados nas próximas secções, tais como:

- Módulo de envio de informação;
- Módulo verificador de JCE;
- Módulo de cifra;
- Módulo de assinatura;
- Módulo de TSA;

- Módulo de PKI gestor;
- Módulo de gerador de chaves;
- Módulo de extracção e estruturação de dados do cartão de cidadão;

4.1 Arquitectura

Nesta secção será apresentada a arquitectura da solução proposta. Esta pode ser apresentada (figura 4.1) em dois grandes módulos, sendo eles:

- API
- Interface de cliente

Muito sucintamente a API pode ser vista como o *core* da solução, e a Interface de cliente oferece a possibilidade de interacção entre um utilizador e a API. A API tem associado a si um conjunto de módulos, que usados de forma complementar permitem assegurar importantes funções a nível de segurança. Para garantir elevados níveis de segurança é necessário garantir a integridade, a autenticação, e o não-repúdio dos dados e comunicações, e estas garantias são dadas pelas assinaturas. Nalguns casos, além das garantias asseguradas pela assinatura, é necessário ainda garantir a confidencialidade dos dados, o que se obtém recorrendo à utilização de algoritmos de cifras.

Este sistema foi desenhado para responder à legislação portuguesa (Decreto-Lei n.º 290-D/99 [2], republicado pelo Decreto-Lei n.º 62/2003 [4]) no que respeita às assinaturas digitais. Esta legislação procede à transposição da Directiva do Parlamento Europeu e do Conselho n.º 1999/93/CE [36], de 13 de Dezembro, relativa a um quadro legal comunitário para as assinaturas electrónicas.

Os módulos que estão referenciados na figura 4.2 oferecem uma visão global, mas completa, sobre a arquitectura do projecto. Como se pode

verificar o módulo API interage com diversos módulos internos, a seguir mencionados:

- Módulo de assinatura;
- Módulo de cifra;
- Módulo de gestão de PKI;
- Módulo de timestamping;
- Módulo de preenchimento automático de documentos;
- Módulo de geração de certificados.

O módulo que constitui a Interface de cliente também contém um módulo interno que é responsável pelo envio de respostas para um determinado endereço, e um módulo responsável pela validação e instalação do JCE. Esta solução foi desenhada para ser integrada em plataformas *web*, mas a sua integração numa aplicação *standalone* também é possível.

Nas próximas secções serão descritas individualmente e pormenorizadamente os vários módulos da interface de cliente e da API que fazem parte da arquitectura apresentada.

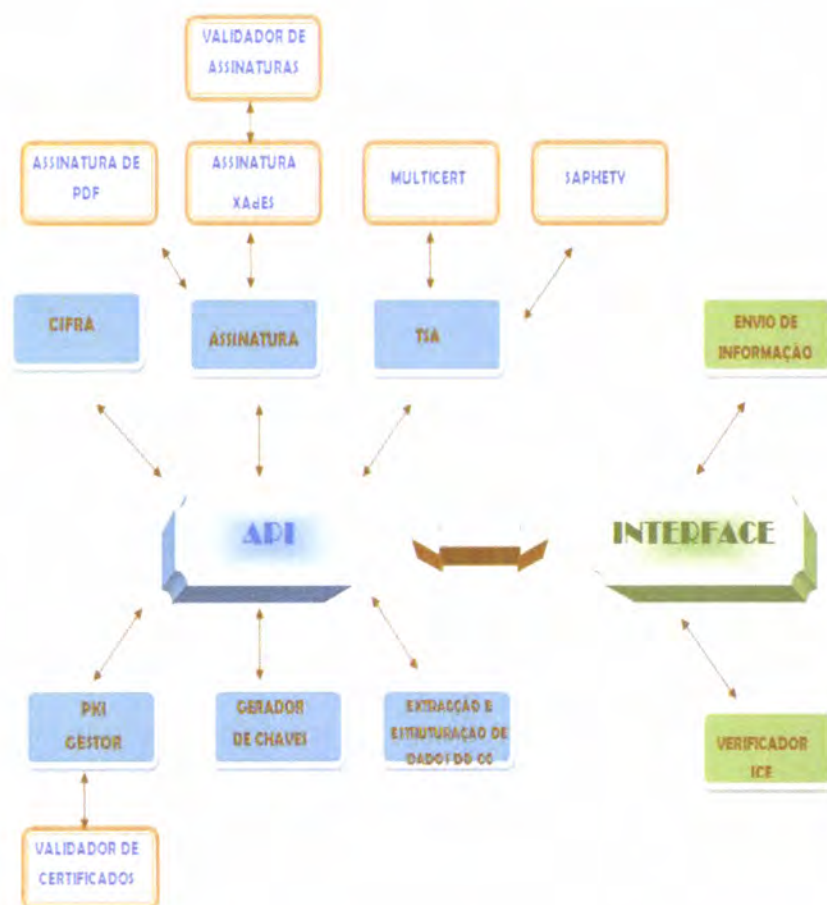


Figura 4.2: Arquitectura

4.2 Interface de cliente

A interface escolhida para esta aplicação foi uma *applet* desenvolvida em Java. Tirando partido de duas características desta tecnologia, para que a escolha recaísse sobre ela. Uma das características é a interoperabilidade das *applet*, que permite a sua execução em diversos sistemas operativos e correm nos mais diversos *browsers*. A outra está relacionada com o facto de as *applet* correrem do lado do cliente e não do lado do servidor. A grande vantagem desta segunda característica, está relacionada com o facto da chave

privada, aquando da realização de assinaturas, não sair do computador do cliente. O desenho deste módulo gráfico é o apresentado na figura 4.3. Ao longo deste capítulo, serão apresentados mais exemplos desta interface, em uso de casos específicos. O aspecto mais simples está relacionado com o facto de esta poder ser integrada em diferentes aplicações sem se apresentar descontextualizada.

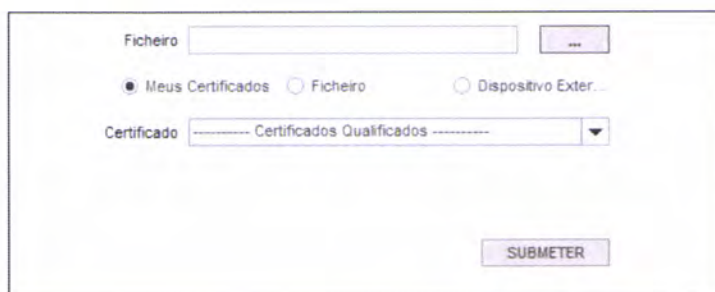


Figura 4.3: Interface gráfica

4.2.1 Fluxos

Fazem parte desta arquitectura 4 fluxos, dois de assinatura e cifra de documentos/ficheiros, um de decifra de documentos/ficheiros e um fluxo para gerar pares de chaves. Todos os fluxos podem ser descritos através da figura 4.4, mas cada um com as especificidades a seguir descritas.



Figura 4.4: Fluxo

Fluxo 1

1. O utilizador acede à plataforma;
2. A plataforma invoca a *applet*;
3. O utilizador escolhe o ficheiro a enviar;
4. O utilizador escolhe a chaves a usar;
5. São realizadas as operações pretendidas, sendo elas, assinatura, cifra do ficheiro seleccionado no ponto 3;
6. É enviada uma resposta dos resultados das operações obtidas do ponto anterior para a plataforma;
7. A plataforma processa esta informação e se necessário armazena essa informação em base de dados.

Fluxo 2

1. O utilizador acede à plataforma;
2. A plataforma invoca a *applet*;
3. Em vez do utilizador escolher o ficheiro a assinar, é a plataforma que ao invocar a *applet* define que ficheiros serão assinados. Podem ser mais que um, estes serão enviados dentro de um zip convertidos para Base64 e enviados como parâmetro na altura em que a *applet* é invocada;
4. O utilizador escolhe os certificados e as respectivas chaves a usar;
5. São realizadas as operações pretendidas, sendo elas, assinatura, cifra do ficheiro seleccionado no ponto 3;
6. É enviada uma resposta dos resultados das operações obtidas do ponto anterior para a plataforma;

7. A plataforma processa esta informação e se necessário armazena essa informação em base de dados.

Fluxo 3

1. O utilizador acede à plataforma;
2. A plataforma invoca a *applet*;
3. São enviados para a *applet* como parâmetros, os dados a serem decifrados;
4. O utilizador escolhe os certificados e as respectivas chaves a usar para este processo;
5. É invocada a API para decifrar os dados recebidos;
6. É enviada uma resposta dos resultados da operação, obtida no ponto anterior para a plataforma;
7. A plataforma processa esta informação e se necessário armazena essa informação em base de dados.

Fluxo 4

1. O utilizador acede à plataforma;
2. A plataforma invoca a *applet*;
3. São enviados para a *applet* como parâmetros, os dados necessários para a gerar o certificado;
4. O utilizador escolhe o local onde deseja gravar o certificado e a chave privada gerada e protegida por uma palavra-chave escolhida e confirmada pelo utilizador;
5. É invocada a API para gerar as chaves e o certificado;

6. É enviada uma resposta dos resultados da operação, obtida no ponto anterior para a plataforma;
7. A plataforma processa esta informação e se necessário armazena essa informação em base de dados.

4.2.2 Envio de informação

O envio da resposta para o servidor que foi descrito nos fluxos é realizado através de um *post* para um determinado URL, fazendo com que a resposta chegue à plataforma. Esse post inicialmente foi realizado usando o pacote *java.net.** que fornece as funcionalidades básicas de acesso via HTTP. Mas este não oferece a flexibilidade total ou as funcionalidades necessárias para muitas aplicações. Foi o que se verificou quando o volume de dados a transmitir aumentou. Quando o volume de dados se aproxima dos 50 *Megabytes*, esta solução deixa de ser viável e de funcionar correctamente.

Em alternativa foi utilizado um projecto da *apache*, o *Http component* que é um sub-projecto da *Jakarta Commons*. Este projecto visa preencher as lacunas da API do Java no que diz respeito ao HTTP. Este, oferece bibliotecas eficientes e actualizadas que implementam os mais recentes standards e recomendações do HTTP. Com a integração deste projecto na solução apresentada, esta ficou com capacidade de enviar para as plataformas documentos com alguns *gigabyte* de tamanho. Uma necessidade que se coloca na prática com cada vez maior frequência.

4.2.3 Instalador JCE

Para um cliente correr esta aplicação necessita de ter instalado o Java JRE 1.5, ou superior. Além deste, existe um outro requisito, que é a presença do JCE (*Java Cryptography Extension*), devido ao tamanho das chaves que

correntemente são utilizadas em Portugal (i.e., cartão do cidadão 1024 bits). A instalação do mesmo não é trivial para um utilizador comum, já que não existe um instalador para o mesmo, sendo necessário copiar ficheiros para uma determinada pasta. Foi então desenvolvida uma funcionalidade na *applet* que primeiro verifica se o JCE presente é o requerido. Se tal não acontecer o utilizador é questionado se pretende instalar este requisito (figura 4.5). Se o utilizador responder afirmativamente é então realizado o download do JCE e a sua instalação.

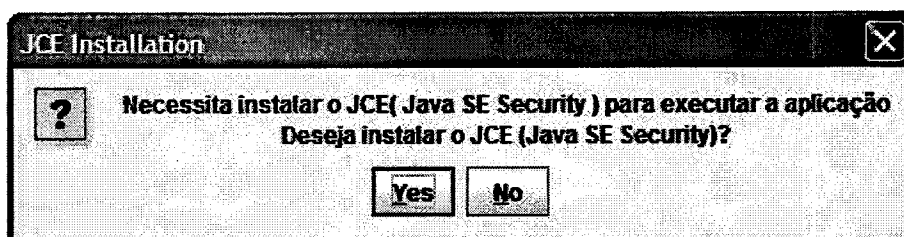


Figura 4.5: Instalação automática do JCE

4.2.4 Parâmetros da interface

Neste ponto serão abordados e descritos alguns dos principais parâmetros utilizados durante a evocação desta Interface de cliente. O exemplo abaixo exemplifica uma possível chamada à Interface de cliente.

```
<applet
  CODEBASE="."
  ARCHIVE="bcprov-jdk16-140.jar,bcprov-jdk14-138.jar,bcmail-jdk16-141.jar,bcmail-jdk14-138.jar,
    bctsp-jdk16-140.jar,commons-io-1.4.jar,commons-lang-2.2.jar,itext-2.1.3.jar,
    iaik-jce.jar-commercial-3.16-signed.jar,iaik_xsect-signed.jar,iaik-tsa-1.0.0-signed.jar,
    iaik-cms.jar-commercial-4.01-signed.jar,iaik_tsp-signed.jar,iaik_javax_crypto-signed.jar,
    iaik_xades-signed.jar,bizgov-pki-API-1.0.3-signed.jar,pki-applet-1.0.0-signed.jar"
  CODE="net.saphety.cipher.gui.Bizgov"
  WIDTH=657 HEIGHT=266 >
<param name="procedure_id" value="Concurso Teste">
<param name="public_key" value="-----BEGIN CERTIFICATE-----
MIID1zCCAr+gAwIBAgIQeq3Jr6sLv9y71eJUQ0xyMzANBgkqhkiG9wOBAQFADBB
MQswCQYDVQQGEwJQVDEQMA4GA1UEChMHU2FwaGV0eTEgMB4GA1UEAxMXU2FwaGV0
eSBQcm9mZXNzaW9uYWwgMDEwHhcNMDgxMjA1MTUwOTE3WbcNMDkxMjA1MTUwOTE3
WjB5MURUwEwYDVQDDAxNXXJpbyBNb3VyYW8xCzAJBgNVBAYTA1BUMSowKAYDVQQK
```

```

DCFTYXBoZXR5IExlDmVsIFRydXNOZWQgU2VydmJjZXMgU0ExJzA1BkqkqhkIG9wOB
CQEWGG1hcmVlLm1vdXJhb0BzYXBoZXR5Lm51dDCBnzANBkqkqhkIG9wOBAQEFAA0B
jQAwgYkCgYEA8+j+veoekpQcPTAR/ueAVY6g7Kigjrill8taPOFy9ZGZAYESk0i9
3wXajK0y98P54IJGLGlnWq1hPjqSpHOYGurArnpru/9f2W1p5VxUuVJ/c0/WfpnP
iy2Q+W0jCe9ge9ai5jggF/9ZCjAzu6FBdaSJoUR2tVm/vM8xK9S/ceECAwEAAaOC
ARUwggERMA8GA1UdEwEB/wQFMAMCAQAwYAYDVROgBFkwVzBVBgsrBgEEAfz6FEYK
DDBGMEQGCCsGAQUFBwIBFjhdHRwOi8vZG9jLnNhcGhldHkubmVOL2NwLONQX1Nh
cGhldHl1fUUhJvZmVzc2lvmFzXzAxLnBkZjBkZjBkZjBkZjBkZjBkZjBkZjBkZjBkZj
0i8vZG9jLnNhcGhldHkubmVOL2NybC9DUkxfU2FwaGV0eV9Qcm9mZnZlbnZaW9uYXxf
MDEuY3JsmA8GA1UdDwEB/wQFAwMHsAAwHwYDVROjBBgwFoAUGECav423URK1Xspf
pIpFR/ah6RcwhQYDVROBBYEFONFMFAEco15yJrckhv09usyPlc0MA0GCSqGSIb3
DQEBBQUAA4IBAQCfJZNdQ9wrikY5LNV/wlvzGXd0k9nPK71tVChVEOLyX0sJPOU
genovQAGVEFcY9B2Zziz0/PPbkoEK95/or9Wr8tnsU+ew1Dvaqv66qRDum2FxxWD
11lQ/SZqvsQuM2pxIhEi/X9JnWcwTJ9UaHswV30FPk8XR5/xj1Ym5wla3ncrGMtc
olnuMKRB1a15PEcZBUk/R/vzPh3NKwNo61MUE4Cq001ZnQGMVobsorbgnfEQ0z2L
MAMPP8FYEiuelQok4tdj5ZztQguYmoy7ftocK1fI7sX4DCnCPHuZmfsV7FWK/ij5
U/R69mjA0UFaFAuyocj0gq8IP4qSKIr2uKo4
-----END CERTIFICATE-----

```

">

```

<param name="post_url" value="http://10.192.60.32:8080/bizgov-pki-server/teste">
<param name="extra_args" value="type,id,id_request_doc,id_doc_type">
<param name="type" value="XXXXX">
<param name="id" value="XXXXX">
<param name="id_request_doc" value="XXXXX">
<param name="id_doc_type" value="XXXXX">
<param name="sign_only" value="true">
<param name="applet_format" value="0">
</applet>

```

Serão agora descritos os principais parâmetros:

- **procedure_id** ID que será reenviado para o servidor para identificação do pedido;
- **sign_only** Se o seu valor for *true* os documentos apenas serão assinados. Se o valor deste campo for *false* os documentos além de serem assinados também serão cifrados;
- **public_key** No caso da propriedade *sign_only* ser *false*, este é a chave a utilizar para cifrar os dados. O valor da propriedade será um Base64 do certificado (*.pem);
- **post_url** URL para onde será enviada a resposta;

- *applet_format* Existem interfaces diferentes correspondendo aos fluxos anteriormente apresentados;
- *extra_args* Esta propriedade identifica uma lista de propriedades que são recebidas pela *applet*, e reencaminhadas na resposta para a plataforma tal como a propriedade *procedure_id*.

4.3 API

A API é o *core* da solução apresentada. Esta pode ser invocada através da Interface de cliente já descrita, contudo pode ser integrada num outro sistema de forma independente. Esta API foi desenvolvida em Java e é compatível com o Java 1.5 ou superior. Os métodos disponibilizado por esta API podem ser consultados em anexo 6. Nas próximas secções serão apresentados detalhes sobre os módulos que fazem parte desta API.

4.3.1 Preenchimento automático de documentos

Este módulo foi desenhado com o intuito de simplificar um determinado grupo de processos muito comuns na nossa sociedade, fazendo uso do novo cartão de identificação português. Os nossos dados pessoais (nome, morada, número de contribuinte) são-nos solicitados repetitivamente, seja para realização de algo complexo, como a compra de uma casa ou de um carro, mas também para coisas comuns e usuais no nosso dia, como para passar uma factura, ou em qualquer tipo de inscrição. A apresentação deste módulo visa melhorar este processo.

Este módulo pode ser dividido em duas fases, com está exemplificado na figura 4.6. A primeira fase passa pela extracção e estruturação da informação. A extracção é realizada através de um *middleware* denominado eID, que consta numa API que é direccionada às funcionalidades não-criptográficas do cartão de cidadão e apenas lida com os dados

identificativos do cidadão. O *kit* de desenvolvimento é fornecido como uma biblioteca *Java wrapper* (JNI) sobre uma interface C/C++. Com a informação retirada do cartão de cidadão é construída uma estrutura XML com base na estrutura XSD, definido pelo governo austríaco para os seus cidadãos e submetido para standard europeu (para mais detalhes consultar a secção 2.1.2). Com esta informação guardada num XML conseguimos obter toda a flexibilidade que esta tecnologia nos oferece. Quer seja no envio dos dados para outra entidade, quer seja para o seu armazenamento em base de dados ou seja apenas para uso local. De realçar, que dados mais sensíveis como a morada só podem ser extraídos com o consentimento do proprietário do cartão e só após a introdução PIN de morada. Numa

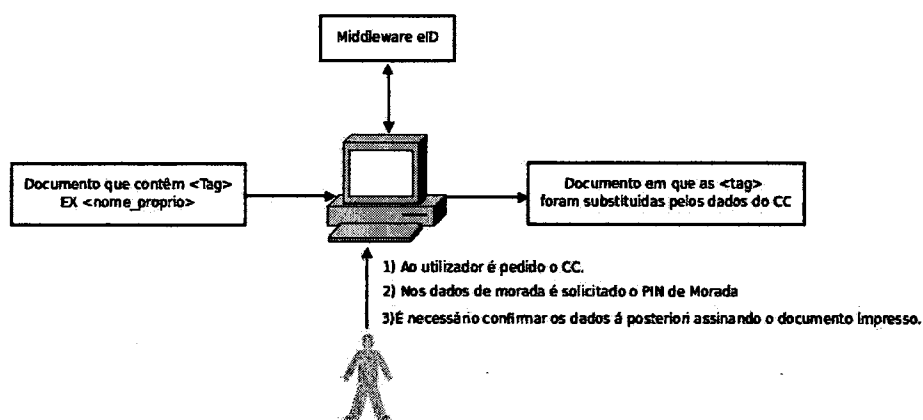


Figura 4.6: Preenchimento automático de documentos

segunda fase, é feito um *parser* sobre um documento indicado, seja este um PDF, um ODT ou qualquer outro, desde que a implementação do *parser* seja realizada. Este documento possui algumas características próprias que consta na presença TAG especiais ao longo do documento. Estas TAG correspondem a um elemento do XML, que por sua vez correspondem a um elemento identificativo do cidadão. Essas TAG serão então substituídas pelos respectivos dados que estão contidos no cartão de cidadão. O ficheiro é então devolvido ao cidadão com todos os seus elementos de identificação

automaticamente preenchido, sendo apenas necessário fazer uma validação sobre os dados.

4.3.2 Gerador de chaves

Este módulo é responsável pela geração de chaves. Uma chave privada e uma chave pública, são geradas em simultâneo, de modo a serem complementares. A criação e o desenvolvimento deste módulo surgiu da necessidade de serem gerados pares de chaves (entenda-se chave pública e privada) para o utilizador, de modo seguro. A solução mais segura, foi a criação destas na máquina do utilizador, para que o segredo da chave privada não seja comprometido. Estes requisitos são realizados através do fluxo 4, apresentado na secção 4.2.1.

Este módulo permite ainda a geração de certificados. Com este é possível realizar uma associação entre a chave privada e uma entidade. Os dados dos certificados são obtidos de duas formas:

1. Os dados do emissor (*issuer*), são informação que identifica a entidade responsável pela verificação e validade de um certificado. Esta informação está contida no ficheiro de propriedade que faz parte da API.
2. Por sua vez, os dados do assunto (*subject*), dados de identificação da pessoa, ou entidade são recebidos como argumentos na chamada directa da API ou como parâmetros na chamada realizada através da interface.

Em relação ao primeiro ponto, as variáveis que podem ser definidas em propriedades são as seguintes:

- *issuer.country* - É indicado o país da entidade emissora (i.e., Portugal);
- *issuer.organization* - É indicado o nome da organização na qual a entidade emissora está inserida (i.e., Saphety level trusted services SA);

- *issuer.commonName* - É designado o nome comum definido pela entidade emissora (i.e., Bizgov - Plataforma Electrónica de Contratação Pública).

Para os dados do assunto (segundo caso apresentado) os parâmetros a definir são os seguintes:

- *dn.cn* - Nome comum da pessoa ou entidade à qual o certificado pertence (i.e., Mário Mourão);
- *dn.c* - Nome do país;
- *dn.o* - Organização na qual a pessoa ou a entidade estão inseridos;
- *dn.e* - Email;
- *dn.ou* - É definido em que unidade da organização o titular do certificado está associado.

Nos casos apresentados apenas os campos *issuer.commonName* e *dn.ou* são de preenchimento obrigatório, todos os outros são de preenchimento facultativo. Este certificado não tem associado a si uma cadeia de certificação, para maior segurança da sua utilização, logo após a geração do certificado, uma cópia do mesmo é enviada para a plataforma onde a Interface de cliente está inserida, através do módulo de envio de documentos.

4.3.3 Assinaturas

Nesta secção será apresentado o módulo da assinatura. A desmaterialização de processos é cada vez mais uma prática corrente dos nossos dias. Esta desmaterialização muito simplificadamente visa levar os processos do “mundo” material para um “mundo” virtual, sem que com isso haja qualquer tipo de perdas, quer processuais, quer legais. Uma das formas de garantir a legalidade, é através da assinatura digital. O módulo da assinatura, está organizado da seguinte forma:

- Assinaturas genéricas sobre qualquer tipo de documentos ou dados (Assinatura XAdES);
- Assinaturas sobre alguns tipos de documentos específicos (Assinatura de PDF);

Assinaturas XAdES

Os detalhes de implementação deste tipo de assinaturas foram descritos na secção 3.1.2. Na secção 2.3, foram apresentadas as seguintes API no que respeita a implementação da assinatura XAdES:

- OpenXAdES;
- Dcontract;
- EldoS;
- IAIK.

A escolhida para a implementação da assinatura XAdES foi a API oferecida pela IAIK. A escolha recaiu sobre esta pelo facto de ser a única que oferece todos os níveis de implementação, contendo ainda, boa documentação de suporte, bem como muitos exemplos de implementação. De realçar também o bom serviço de suporte técnico, que permite em caso de dúvidas de implementação ou qualquer dificuldade em ultrapassar obstáculos obter uma resposta útil e rápida. Serão apresentados a seguir, certos detalhes de implementação, que estão ligados a algumas das decisões que foram tomadas durante a implementação.

O nível de implementação actual é o XAdES-X-L. A escolha sobre um alto nível esteve relacionada com as garantias que este oferece a uma assinatura. Este tipo de assinatura contém todos os certificados da cadeia de certificação. Associado a cada um destes é sempre guardada a informação que permite validar o respectivo certificado, seja através de OCSP ou de

CRL. Numa primeira fase apenas se guarda as CRL's correspondente a cada certificado. Esta passou a ser inviável devido ao tamanho de algumas CRL's. Existem CRL's associados à cadeia de certificação do cartão do cidadão que são superiores a 14 MB. Devido a casos como este deixou de ser viável utilizarmos exclusivamente CRL. Como nem todas as CA oferecem serviço de OCSP é necessário que:

1. Seja verificado através da extensão *Authority Information Access* se o certificado tem associado algum serviço de OCSP;
2. Se possuir este serviço, é efectuado o pedido de validação do certificado, e é assim guardado na XAdES a resposta do OCSP;
3. Se não possuir este serviço, seja guardada na XAdES a CRL correspondente.

A figura 4.7 exemplifica um possível caso de como fica armazenada esta informação na XAdES. A primeira lista contém todos os certificados pertencentes à cadeia de certificação. A segunda lista contém a resposta do OCSP em relação ao certificado correspondente. Por sua vez a terceira é constituída pelas CRL's.

Cert.	Cert. 1	Cert. 2	Cert. 3	CERTIFICADOS
				RESPOSTA DO OCSP
				CRL

Figura 4.7: Validade de um cadeia de certificados

Existem três possibilidades de representar um documento numa assinatura digital: *enveloping*, *enveloped* e *dettached*. A representação escolhida para a XAdES foi *enveloping*. Mas à medida que os testes se tornaram mais

exigentes, este tipo de abordagem não permitiu superar os objectivos. O problema surgia quando se pretendia assinar documentos volumosos que podiam ir até alguns GB's. Como os outros dois tipos de representação (*enveloped e dettached*), também não respondiam a todos os requisitos, procedeu-se a uma alteração na representação *enveloping*, de modo a que conseguisse responder aos requisitos sem que com isso fosse posta em causa a sua legalidade. A alteração passa por, em vez de colocar o documento em Base64 no elemento *object* da XAdES, é colocado o sumário (i.e., SHA1) do mesmo. Surgem então dois ficheiros: o documento original e um XML com a assinatura XAdES correspondente ao ficheiro original.

Validador de assinaturas XAdES

A assinatura XAdES contém toda a informação necessária para se tornar numa assinatura com valor legal. Paralelamente ao desenvolvimento da geração da assinatura foi desenvolvido um sub-módulo que permite a validação da mesma. As validações *core* são processadas através da biblioteca IAIK_XAdES. No entanto, devido a algumas especificações próprias aquando da geração da assinatura XAdES, surgiu a necessidade de poder validar os dados provenientes dessas especificações. Nomeadamente a validade do certificado e a respectiva cadeia de certificação (entenda-se por validade se um certificado se encontra ou não revogado à data da assinatura e se o mesmo não se encontrava expirado). Além deste ponto, é necessário também validar se o sumário (i.e., SHA1) corresponde ao ficheiro original.

Durante a validação, em caso de erro ou falhas vão sendo adicionados os códigos correspondentes a várias listas dependendo do erro a que se refere. Existem três listas para mais facilmente distinguir o tipo de erro ou falha, sendo elas:

- **Lista de validação *core*** - Nesta lista, surgem os erros que ocorrem durante a validação da estrutura da assinatura, da validação da assinatura em si e da análise do selo temporal.

- **Lista de certificados** - Nesta lista, encontram-se os erros relativos a verificação dos certificados da cadeia de certificação, tendo em conta a sua validade.
- **Lista de validação de referências** - Como foi referido na secção 3.1.2, as referências dos certificados que pertencem à cadeia de certificação, bem como as referências para as respectivas CRL's ou respostas do serviço de OCSP, encontram-se assinadas e possuem um selo temporal. A validação desta informação pertence à validação *core*, contudo é necessário assegurar que tanto os certificados como a informação que os permite validar, correspondem às referências assinadas, e que não houve alterações nestes elementos da XAdES já que esta informação não é assinada. Esta lista irá conter os erros relativos a esta validação.

Assinatura de documentos PDF

Além da assinatura sobre documentos genéricos, este módulo permite assinar documentos específicos, nomeadamente PDF's. Estes podem ser validados através do uso de aplicações externas, como é o caso do *Acrobat Reader*. Este módulo tem como base uma API *open-source* denominada *itext*.

Este módulo, permite ainda colocar selo de assinatura sobre o PDF. Pode ser definido onde será colocado este mesmo selo, sendo especificado através de argumentos as coordenadas do mesmo.

4.3.4 Gestor de PKI

Este módulo, é responsável por induzir transparência e versatilidade no modo de usar as chaves, sem que para o resto da aplicação tenha relevância se as chaves a usar são provenientes de um *token*, de um *smartcart*, de um PFX ou de um P12. Tal como é ilustrado na figura 4.8, no caso do sistema operativo Windows, ainda surge a possibilidade de obter as chaves através da *keystore* do mesmo, mantendo a transparência e versatilidade pretendida.

Para aceder a esta, foi utilizado o *provider* SunMSCAPI, que permite aceder às bibliotecas criptográficas que dão acesso à *keystore* do ambiente Microsoft Windows. Este *provider* não oferece funcionalidades criptográficas, serve apenas de canal entre ambientes Java e os serviços criptográficos nativos do Windows.

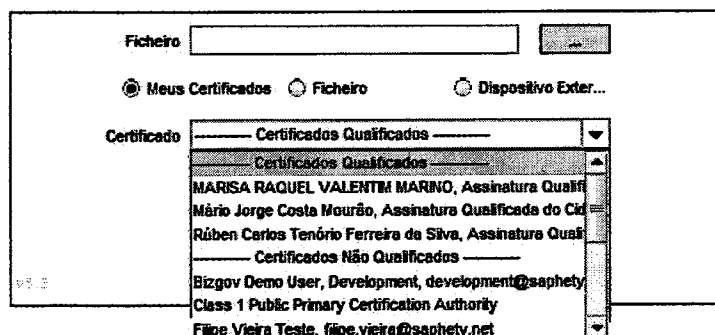


Figura 4.8: Lista de certificados instalados na *keystore* do Windows

A transparência e a versatilidade são conseguidas com uma classe que é designada de *KeyStoreSelected* (esta classe pode ser consultada em anexo 6), e que possui três variáveis:

- **privateKey** - representa a chave privada (*PrivateKey*);
- **certificate** - representa o certificado associado à chave privada (*X509Certificate*);
- **chain** - todos os certificados que fazem parte da cadeia de certificação podem ser guardados nesta variável (*X509Certificate[]*).

Esta classe é instanciada por uma das seguintes:

- **MyKeyStore** - Esta classe só faz sentido ser evocada em ambiente Windows. Esta classe lê e usa as chaves que estão instaladas na *keystore* do Windows. Tal como ilustra a figura 4.8 é dado ao utilizador uma lista de certificados a escolher, sendo feita a destinação entre certificados qualificados e não qualificados. Numa segunda fase, o utilizador

escolhe as chaves que pretende utilizar e nesta fase são então preenchidas as variáveis da classe *KeyStoreSelected* já instanciada. O método *getKeyStore* devolve o objecto *KeyStoreSelected* já instanciado e com as variáveis definidas. Embora esta funcionalidade seja apenas disponibilizada para o Windows a mesma é transversal ao *browsers* a usar.

- **PKCS12KeyStore** - Como é ilustrado na figura 4.9 o utilizador escolhe o ficheiro que contém as chaves e os respectivos certificados, sendo necessária a introdução de uma *password* que permite o acesso a esta informação. É então instanciada a classe *KeyStoreSelected* e as suas variáveis preenchidas. O método *getKeyStore* devolve este mesmo objecto. Esta implementação é transversal a diversos sistemas operativos e a diferentes *browser*.

The image shows a graphical user interface for selecting a PKCS12 key store. It features the following elements:

- A text input field labeled "Ficheiro" with a browse button (three dots) to its right.
- Three radio buttons: "Meus Certificados" (unselected), "Ficheiro" (selected), and "Dispositivo Exter..." (unselected).
- A text input field labeled "Certificado" containing the text "cliente de trabalho\shared\paraapagar\cliente.pfx" with a browse button (three dots) to its right.
- A text input field labeled "Password" with masked characters (asterisks).
- A "SUBMITER" button located at the bottom right of the dialog.

Figura 4.9: PKCS12

- **PKCS11KeyStore** - Esta classe serve para utilizar as chaves que se encontram guardadas em *tokens* ou *smartcard*. É solicitado ao utilizador que seleccione o driver para comunicação com dispositivos externos. São então mostradas informações dos certificados que têm a si associada uma chave privada nestes dispositivos (figura 4.10). O utilizador escolhe o que pretende usar e é então instanciada a classe *KeyStoreSelected* e as suas variáveis preenchidas. O método *getKeyStore* devolve este mesmo objecto. Esta implementação é transversal a diversos sistemas operativos e a diferentes *browser*.

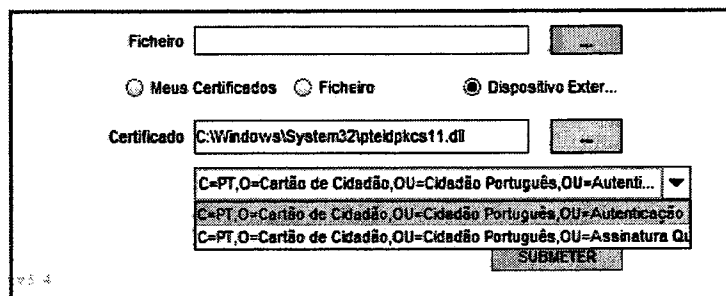


Figura 4.10: PKCS11

As classes *PKCS11KeyStore*, *PKCS12KeyStore*, *MyKeyStore* podem ser consultadas em anexo 6.

4.3.5 Validador de certificados

É feita uma avaliação sobre o objecto *KeyStoreSelected*. Em seguida é analisado o certificado correspondente à chave privada e a respectiva cadeia de certificação. Finalmente são analisados os seguintes pontos:

- Se a cadeia de certificação se encontra completa;
- Se nenhum certificado se encontra revogado;
- Se nenhum certificado se encontra expirado;
- Se o certificado base é considerado certificado digital qualificado.

4.3.6 Cifra

O módulo de cifra permite juntar à autenticação e integridade dos dados (assinatura) a confidencialidade. Devido ao tamanho dos ficheiros usados tornava-se impraticável cifrar os mesmos fazendo uso exclusivamente da criptografia assimétrica. O uso exclusivo de criptografia simétrica também não é viável, visto que os algoritmos simétricos são eficientes para cifrar

grandes volumes de dados, mas a distribuição das chaves é problemática. Assim sendo, é necessário encontrar um mecanismo que possa aproveitar as vantagens e evitar as desvantagens de cada tipo de algoritmo, para cifrar grandes volumes de dados.

Quando um emissor pretende enviar uma mensagem para um receptor do qual conhece a chave pública, começa por gerar uma chave secreta aleatória e usa-a para cifrar a mensagem. Em seguida, cifra esta chave com a chave pública do receptor. Finalmente, envia a mensagem e a chave secreta cifradas ao receptor. O conjunto destes dois elementos é habitualmente designado por envelope digital.

Ao receber o envelope digital, o receptor começa por decifrar a chave secreta com a sua chave privada (como a cifra foi produzida com a chave pública, apenas a chave privada a pode decifrar). Uma vez na posse desta chave, pode decifrar a mensagem.

Para a implementação desse envelope digital foi utilizada numa primeira fase a biblioteca IAIK_XSECT, com resultados aceitáveis a nível de cifra e decifra de mensagens. No entanto, é necessário mais do que cifrar e decifrar documentos. Este módulo necessitava de cifrar documentos que poderiam chegar a alguns GB's. Foi contactada a equipa técnica do IAIK mas as respostas em relação a este ponto não foram positivas. Analisaram-se então mais algumas soluções em Java mas sem nunca chegar a um resultado que cumprisse com todos os requisitos.

A solução passou pela implementação do standard definidos pela W3C para *XML Encryption Syntax and Processing*. Esta implementação pode ser consultada em anexo 6. A sua construção foi conseguida com o auxílio da API da *sun* denominada StAX. Esta API tem a particularidade de realizar o *parser* em blocos. Com esta abordagem não é realizada uma análise

estruturada ao XML, o que é prejudicial no que toca a despiste de erros, mas vantajoso quando se deseja o *parser* documentos com grandes dimensões. O elemento que contém o documento cifrado, com esta implementação pode ir até vários Gb's, sem qualquer problema, quando é necessário processá-lo. Com esta implementação é possível cifrar e decifrar documentos de qualquer volume cumprindo os standards definidos pela W3C numa qualquer máquina comum.

4.3.7 Selos temporais

Os selos temporais oferecem validade cronológica aos dados. As autoridades de selos temporais (TSA), são as responsáveis pela emissão destes selos. Os selos associam informação temporal aos dados através de uma assinatura. Este módulo é responsável pela integração com diversas TSA de modo a que a sua utilização seja transparente para a restante aplicação. Neste momento, a aplicação comunica com duas TSA são elas a Saphety e a Multicert, mas a integração com mais TSA é compatível, através de um nível de abstracção. Este nível de abstracção é conseguido devido à definição de uma interface (esta interface pode ser consultada em anexo 6). Os selos temporais são usados internamente pelos modos de assinatura e estão presentes na assinatura XAdES e na assinatura em PDF. Os selos temporais também podem ser usados externamente através de chamadas à API.

Capítulo 5

Caso de Estudo

Neste capítulo, serão apresentados alguns casos de estudo, em que foi implementada a aplicação referida no capítulo 4. A solução apresentada já foi integrada em três projectos de diferentes áreas e com diferentes funções, sendo eles:

- Portal de Contratos;
- Sigpoa;
- BizGov.

Nas próximas secções, serão abordados mais detalhadamente os projectos e a sua integração com a solução proposta.

5.1 Portal de Contratos

O projecto tem como objectivo, implementar para a Sonae Distribuição um sistema de arquivo e assinatura de contratos electrónicos no âmbito do “Programa Receitas” da mesma.

Os contratos são enviados, através do canal Biztalk já existente, pela Sonae Distribuição para a Saphety, sendo arquivados aquando da sua

recepção. Todos os contratos são disponibilizados aos utilizadores através de um Portal Saphety, que pode ser acedido directamente ou por um link no portal de fornecedores Sonae Distribuição e onde podem ser efectuadas as seguintes operações: assinatura digital por parte dos procuradores MCH, assinatura digital por parte dos representantes dos fornecedores e pesquisa sobre os contratos. Para os sistemas da Sonae Distribuição será sempre retornado um status da operação efectuada nos sistemas da Saphety. A figura 5.1 seguinte ilustra uma visão geral sobre o sistema. O sistema vai



Figura 5.1: Visão Geral do Sistema de Arquivo e Formalização de Contratos

permitir a todos os utilizadores autorizados, a funcionalidade de assinatura digital de contratos. Esta funcionalidade é disponibilizada pela componente de assinatura de contratos e obtida através do seguinte fluxo:

1. O utilizador acede ao portal Saphety;
2. O utilizador selecciona o contrato a assinae;
3. Os contratos são enviados via *applet* para o computador do utilizador;

4. O utilizador selecciona a chave privada associada ao seu certificado digital. O certificado a usar poderá ser o certificado digital qualificado presente no cartão de cidadão;
5. O utilizador insere a *password* da chave privada;
6. O contrato é assinado e enviado através de mecanismo descrito no capítulo 4 para os sistemas da Saphety.

A figura 5.2 mostra a integração da *interface* no portal de contratos.

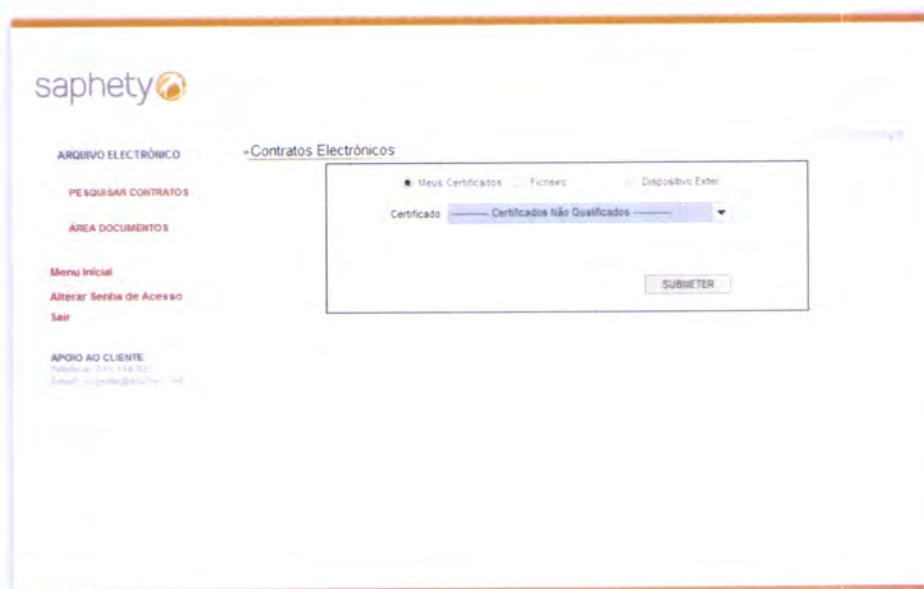


Figura 5.2: Portal de contratos

5.2 Sigpoa

A 28 de Julho de 2008 foi celebrado um protocolo entre a CCDR Alentejo e a Universidade de Évora, com vista ao desenvolvimento do sistema de informação definitivo, denominado SIGPOA – Sistema de Informação e Gestão do PO Alentejo. Em seguida, serão apresentadas as principais

características do sistema de informação em fase de desenvolvimento. O sistema assentará numa base de dados relacional em PostgreSQL, a qual além de oferecer garantias de segurança, fiabilidade e escalabilidade, não pressupõe quaisquer custos de licenciamento, já que se trata de uma solução *open-source*. No desenvolvimento do SIGPOA recorrer-se-á a tecnologia *open-source* baseada em PHP e metodologia MVC (Model View Controller) através da *framework* Cake. O sistema será disponibilizado através de interfaces *web*, garantindo-se a compatibilidade com os *browsers* mais comuns. A componente de arquivo documental deverá incorporar igualmente a capacidade de digitalização de documentos, funcionalidade que deverá ser prevista no âmbito do equipamento a adquirir para o efeito. Já a componente de repositório documental deverá assentar na tecnologia Alfresco (gestor de conteúdos *open-source*), garantido a possibilidade de integração com outros sistemas de gestão documental/conteúdos da CCDRA, a disponibilização de mecanismos de circulação electrónica e a partilha de documentos.

Este projecto assenta na integração da solução apresentada. Nesta primeira fase apenas integra a assinatura de um documento no formato PDF. Ainda assim, o grande objectivo é a assinatura digital qualificada de documentos. Estes PDF são ainda certificados quanto à data da assinatura através de um selo temporal emitido pela TSA Saphety.

Existem dois fluxos que fazem parte da solução proposta e que estão implementados no projecto SIGPOA. O primeiro fluxo pode ser apresentado nos seguintes passos:

1. O utilizador acede à plataforma;
2. A plataforma invoca a interface;
3. O utilizador selecciona o ficheiro a enviar, o qual deverá ser um ficheiro no formato PDF;

4. O utilizador escolhe as chaves a usar. O certificado a usar poderá ser o certificado digital qualificado presente no cartão de cidadão;
5. O documento é assinado digitalmente, sendo colocada uma imagem sobre o PDF representativa desta mesma assinatura;
6. É enviada uma resposta dos resultados das operações obtidas no ponto anterior para a plataforma;
7. A plataforma processa esta informação e se necessário armazena a mesma.

O segundo fluxo implementado é similar ao primeiro, residindo a diferença na forma como o ficheiro é carregado. Enquanto que, no primeiro fluxo apresentado o ficheiro é carregado pelo utilizador, neste segundo fluxo, o ficheiro é enviado para a interface (*applet*) como parâmetro. Pode ser enviado um ou mais documentos PDF num ficheiro ZIP, que é enviado por parâmetro. Antes de ser enviado este sofre uma transformação, que consiste na codificação através do algoritmo Base64. O segundo fluxo realiza-se então através dos seguintes passos:

1. O utilizador acede à plataforma;
2. A plataforma invoca a interface (*applet*);
3. É efectuada a leitura do ZIP através da descodificação do parâmetro *data_to_process*, fazendo uso do algoritmo Base64;
4. São lidos os documentos PDF que fazem parte do ZIP;
5. O utilizador escolhe os certificados a usar. O certificado a usar poderá ser o certificado digital qualificado presente no cartão de cidadão;
6. É assinado então cada documento separadamente;
7. À medida que os documentos vão sendo assinados, vão sendo novamente colocados num ficheiro ZIP. Ficheiro esse que no final é enviado novamente para a plataforma;

8. A plataforma processa esta informação e se necessário armazena essa informação em base de dados.

5.3 BizGov

O BizGov é a solução informática que, através da internet, permite a realização de procedimentos electrónicos públicos, bem como a aquisição electrónica de bens e serviços.

Esta solução, enquadra-se no âmbito do DL n.º 18/2008 de 29 de Janeiro [6], do DL n.º 143-A/2008[5] de 25 de Julho e da Portaria 701-G/2008[7] de 29 de Julho de 2008. O DL n.º 18/2008 regula a formação e execução dos contratos públicos, definindo desta forma todos os procedimentos que decorrem desde o momento em que é tomada a decisão de contratar uma entidade até à adjudicação, assim como a execução do contrato. O DL n.º 143-A/2008 estabelece os princípios e regras gerais a que devem obedecer as comunicações, trocas e arquivo de dados e informações, previstos no Código dos Contratos Públicos, aprovado pelo Decreto -Lei n.º 18/2008, de 29 de Janeiro, em particular, a disponibilização das peças do procedimento, bem como o envio e recepção dos documentos que constituem as candidaturas, as propostas e as soluções. Por sua vez a portaria define os requisitos e condições a que deve obedecer a utilização de plataformas electrónicas pelas entidades adjudicantes, na fase de formação de contratos públicos, e estabelece as regras de funcionamento daquelas plataformas.

A aplicação BizGov utiliza o processo de validação cronológica de acordo com o Standart RFC 3161 (Time-Stamp Protocol). A validação cronológica fica sempre associada aos documentos, sendo possível identificar qualquer alteração posterior ao selo temporal. A informação de validação cronológica é incluída no “envelope” Xades-X-L usado para o arquivo dos documentos, quer na base de dados, quer no *filesystem*, permitindo, ao longo do tempo, a

preservação dos selos temporais.

A adição do selo temporal nos documentos submetidos à plataforma é realizada pela solução proposta. A solução proposta recorre sempre ao servidor de *timestamping* que tem configurado e pré-definido pela plataforma, não utilizando nunca os serviços de data e hora do sistema local onde está em utilização.

5.3.1 Assinatura dos documentos/ficheiros

Todos os documentos/ficheiros enviados para a plataforma são assinados electronicamente utilizando uma assinatura do tipo XAdES-X-L (*eXtended long-term*). No momento da assinatura, não é obrigatório que o certificado utilizado para assinar o documento seja o mesmo que foi utilizado para a autenticação. A solução proposta verifica se o certificado do utilizador é emitido por uma entidade certificadora autorizada, onde se encaixa os certificados contidos no cartão de cidadão. A lista de entidades autorizadas será mantida numa pasta pública (mas só de leitura) no servidor do BizGov. A validação do certificado quanto à sua possível revogação e/ou data de expiração é feita pela solução proposta, utilizando as datas do certificado e a CRL ou através do serviço OCSP presente nesse mesmo certificado na data de assinatura do documento. O envelope XAdES-X-L criado contém a assinatura digital e a data na qual foi realizada essa mesma assinatura (através do elemento correspondente ao *timestamp*). Esta informação é guardada na base de dados da plataforma pelo período legal, pelo que em qualquer momento será possível realizar a operação de validação sobre um documento assinado digitalmente.

5.3.2 Visualização da proposta submetida

Enquanto a proposta está em composição o utilizador mantém os dados na página *web* que está disponível no seu browser. Em qualquer momento o utilizador pode finalizar a introdução dos dados. Esta acção irá gerar um

ficheiro estruturado que será enviado como parâmetro para a solução proposta para ser assinado, cifrado e enviado para a plataforma. Sempre que o utilizador pretender, pode guardar temporariamente os dados já introduzidos, sendo nesta acção gerado um ficheiro estruturado mantido na máquina do cliente. Posteriormente, este ficheiro pode ser carregado pelo cliente para finalizar a introdução dos dados. Embora estas acções façam com que os dados sejam sempre enviados para a plataforma, os mesmos nunca são registados em sistema, pelo que não existe a possibilidade de serem lidos e/ou analisados.

5.3.3 Decifrar as propostas

Chegada a data da abertura das propostas, são geradas *passwords* para os elementos do júri e enviadas por email (figura 5.3). Cada um deles (num



Figura 5.3: Abertura de propostas - Envio de *password* para o júri

mínimo de 3 elementos) deverá aceder à página de análise de propostas no detalhe de procedimento e ser-lhes-á pedido que insiram a respectiva *password*. O sistema aguarda até que todos os elementos do júri insiram a *password*, após o que disponibilizará os campos para que um dos elementos introduza, então, a chave privada e respectiva *password*, para o procedimento (figura 5.4). Após o que foi referido acima, é dado início ao processo, que deverá correr no sistema em *background*, e, enquanto estiver a decorrer será mostrada uma mensagem indicando essa situação. O processo de decifra dos documentos é feito do lado da plataforma, logo, do lado do servidor.

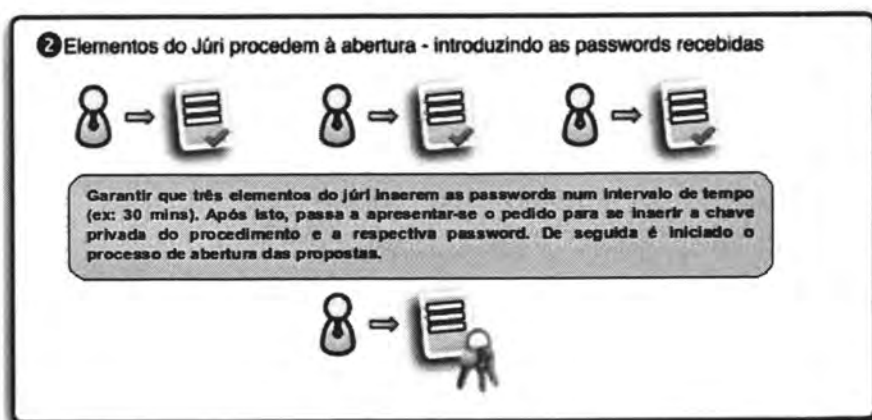


Figura 5.4: Abertura de propostas - Júris introduzem *password*

O processo percorre a lista de propostas do procedimento em causa e, recorrendo à API apresentada na solução proposta é decifrado cada um dos documentos, e os dados de cada proposta. Os ficheiros e os dados decifrados serão guardados na base de dados (figura 5.5). É um processo síncrono, pelo

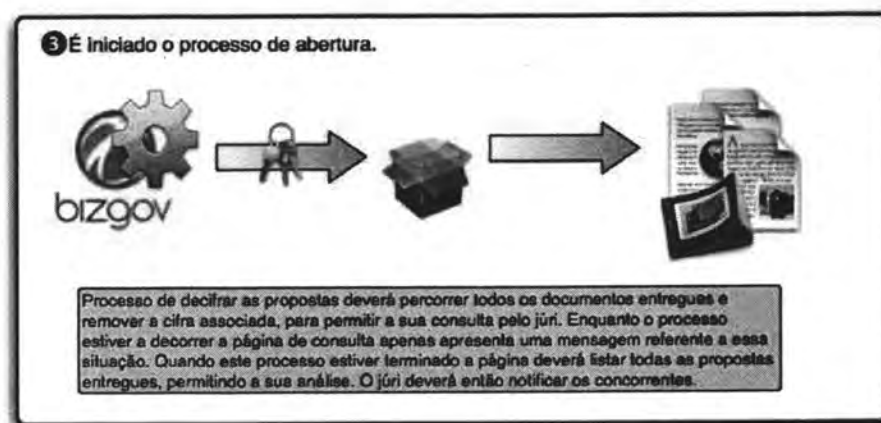


Figura 5.5: Abertura de propostas - Processo de decifração

que será importante dar *feedback* aos utilizadores de que está a decorrer uma acção. É importante também prever mecanismos de recuperação em caso de

falha do processo a meio. O utilizador deverá fornecer novamente a chave e a *password* e recomeça-se o processo onde suspendeu. No momento da abertura das propostas os elementos do júri têm a possibilidade de validar a informação de segurança (assinaturas e selos temporais) associada com cada documento.

Todos os fluxos apresentados na solução proposta (4.2.1) têm integração na aplicação BizGov. O fluxo 1, consiste em enviar para a plataforma documentos/ficheiros assinados, estes podem também ser cifrados. A figura 5.6 ilustra a integração da solução proposta com a plataforma BizGov neste mesmo fluxo. O fluxo 2, é algo similar ao fluxo 1, com a particularidade



Figura 5.6: BizGov - fluxo 1

que os ficheiros são enviados pela plataforma BizGov para a *applet* como argumento. Um exemplo deste fluxo será apresentado.

- Um fornecedor apresenta uma proposta (figura 5.7);
- Esta informação é guardada e estruturada num ficheiro XML;
- Essa informação é então enviada para a interface (*applet*) através do argumento `data_to_process`;

- Fazendo uso da solução apresentada o documento é assinado e cifrado, sendo então enviado novamente para o servidor e guardado na respectiva base de dados. A assinatura realizada deverá ser uma assinatura digital qualificada (i.e., cartão de cidadão).

05-10-2009 22:48:55 [Ver website: Oficina Reparação Auto 1](#) [IMC2](#) [SAR](#) [A. SENA](#)

Tipo: Licit. Directo - Regime Geral
Finalidade: Contrato Público
Procedimento Nº: 0007029_782
Designação do Procedimento: 0007029_782

Estado: Em apresentação de Propostas
Tempo restante para apresentação de Propostas: 4 dias e 16:13:42 horas
Tempo restante para apresentação de Esclarecimentos: 0 dias e 26:13:42 horas
Tempo restante para apresentação de Erros e Omissões: 4 dias e 08:42:42 horas

Proposta

Proposta Nº: 0 0 [ajuda](#)
Versão: 1
Fornecedor: Oficina Reparação Auto 1

Documentação

Documentos Requeridos			Documentos da Proposta	
Categoria de Documento	Descrição do Documento	Documento Modelo	Classificação	Ficheiro
Não existem documentos associados				

Aspectos da execução do contrato

Aspectos submetidos à concorrência

Factores	Unidade de medida	Ponderação	Resposta	Observações
Preço	EUR	100 %	110000	

Aspectos não submetidos à concorrência

Factores	Subfactor	Unidade de medida	Valor de referência	Resposta
Factores inexistentes				

Possível importar a matriz de quantidades em formato folha de cálculo e em XML.

A exportação em XML da matriz de quantidades permite a visualização e armazenamento dos dados de uma forma estruturada.

O formato folha de cálculo é uma forma rápida de visualizar e editar os valores da matriz de quantidades.

[IMPORTAR](#) [EXPORTAR XML](#) [EXPORTAR XLS](#)

[GUARDAR](#) [SUSPETER](#) [VOLTAR](#)

Figura 5.7: BizGov - fluxo 2

Quando é necessário decifrar dados, sem que com isso a chave “saia” do fornecedor é utilizado o fluxo 3. Este pode ser usado depois da acção segundo o fluxo anterior. É enviado para a *applet* o documento a decifrar, o fornecedor escolhe a chave privada correspondente à chave pública com a qual o documento foi cifrado (5.8). O documento é decifrado e enviado para a página *web* associada à sessão. Esta informação é mostrada ao fornecedor mas fica apenas em memória. Em terminando a sessão esta informação é

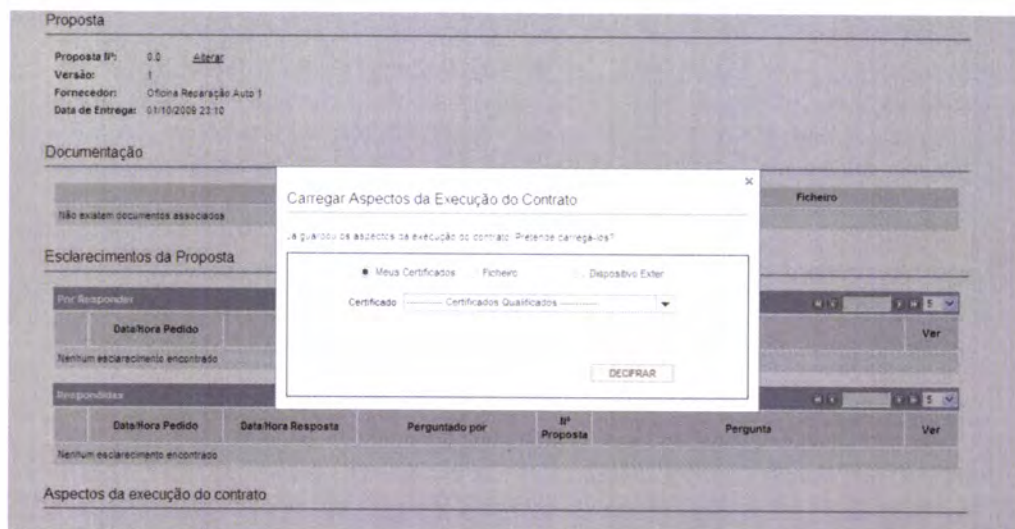


Figura 5.8: BizGov - fluxo 3

apagada, visto tratar-se de informação confidencial até à altura do acto público. Este fluxo serve para o fornecedor poder acompanhar as suas respostas enquanto o estado do procedimento o permite.

Para cada procedimento pode ser necessário gerar até três chaves, uma para cada fase. O número de fases está directamente relacionado com o tipo de procedimento. As fases para as quais são geradas chaves são as seguintes (figura 5.9):

- Fase de candidatura;
- Fase de propostas;
- Fase de soluções.

As chaves são geradas separadamente, através da solução apresentada neste trabalho (figura 5.10). O fluxo de geração de chaves corresponde ao fluxo 4 da solução apresentada. Depois de geradas as chaves e de ser enviado para

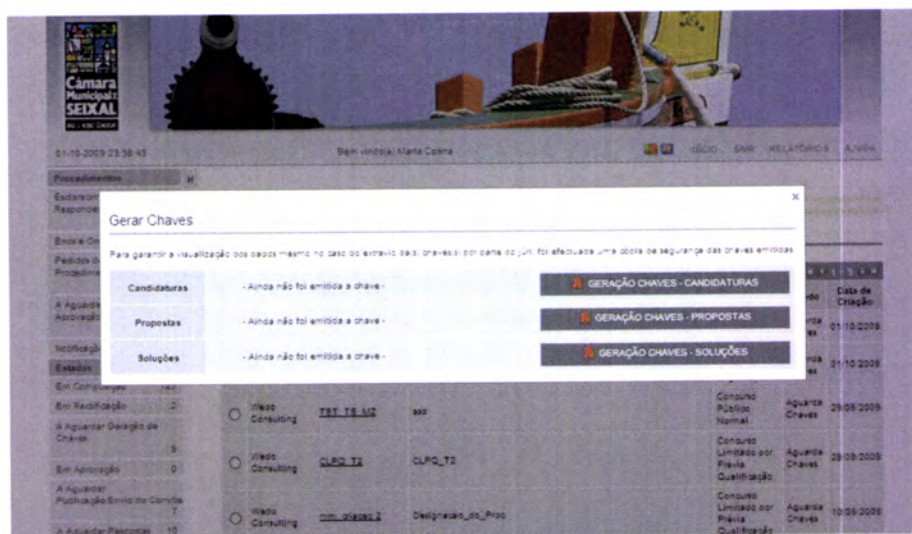


Figura 5.9: Chaves por gerar

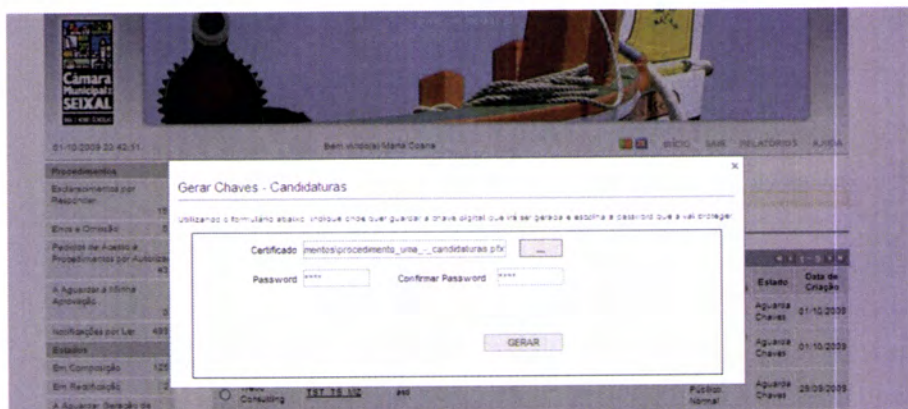


Figura 5.10: A gerar chaves

a plataforma o respectivo certificado, é então libertada a informação sobre o mesmo (figura 5.11).

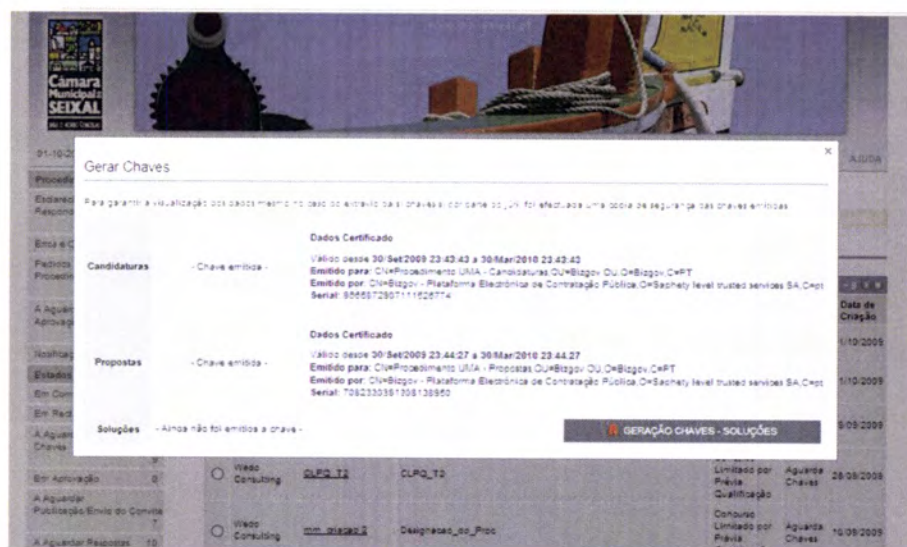


Figura 5.11: Chaves geradas

Capítulo 6

Conclusão e trabalho futuro

Neste capítulo será apresentada uma reflexão sobre o trabalho apresentado ao longo desta dissertação. Com a implementação da solução proposta, apresentada no capítulo 4, os objectivos previstos no capítulo 1.2, foram de um modo global atingidos. Esta solução faz uso do cartão de cidadão e através de tecnologias de autenticação, assinatura e cifra, permite juntamente com outras aplicações a simplificação e desmaterialização de processos.

Tratando-se de um trabalho de investigação foi necessário procurar vias de resolução dos problemas, com base nos conhecimentos e nas ferramentas disponíveis. Nem sempre a solução estudada foi a ideal, sendo necessário constantemente recuar e procurar novos caminhos. Como marcos fundamentais a assinalar, registo o desenvolvimento de alguns módulos da API, nomeadamente o módulo de assinatura em que a via seguida (XAdES-X-L) era das de maior grau de dificuldade de implementação. Também o módulo de cifra foi marcante por ser necessário implementar de raiz o standard necessário, sendo disponibilizado o respectivo código. Considero ainda relevante a construção do módulo de gestão PKI devido à sua interoperabilidade entre diversos sistemas operativos e *browsers*, sendo o seu funcionamento adaptado ao ambiente em utilização. Por outro lado é este que permite a integração com o cartão de cidadão.

Apenas o módulo de preenchimento automático de documentos (4.3.1) não foi implementado na prática. Todos os outros, apresentados no capítulo 4 foram implementados e estão integrados em diversas soluções como referenciado no capítulo 5.

Conforme referido no capítulo 5 a solução proposta foi integrada numa plataforma em produção e aberto a entidades externas (5.3), respondendo satisfatoriamente e em conformidade com os objectivos pretendidos. De salientar ainda que o desenvolvimento desta aplicação foi balizada pelo quadro normativo em vigor, o que permitiu a sua aprovação pelo CEGER (Centro de Gestão da Rede Informática do Governo) no âmbito da plataforma acima referida.

Existem diversos pontos que podem ser visto como oportunidade de melhoria, ou como trabalho futuro, designadamente:

- O desenvolvimento de um novo módulo para envio de ficheiros/documentos de grandes dimensões. Este novo módulo seria idêntico ao actual, mas complementado com alguma “inteligência” e suporte em caso de falhas externas. Este módulo passaria por dividir os ficheiros em pequenos blocos antes de os enviar. Numa primeira conexão o servidor seria informado do envio do ficheiro e algumas informações sobre o mesmo tais como: o sumário, o nome, o tamanho, o tipo e número de blocos em que seria dividido. Posteriormente o ficheiro seria enviado bloco a bloco. Se por algum motivo a conexão fosse quebrada, necessariamente o *upload* do ficheiro seria interrompido, mas o servidor guardaria a informação do número de blocos recebidos até esse momento. Numa nova conexão o upload do ficheiro poderia ser continuado num ponto onde tinha sido interrompido. O servidor seria informado da existência de um ficheiro para o qual se pretendia completar o *upload*. O servidor informará sobre o número

de blocos em que o *upload* foi realizado correctamente. Serão então enviados os restantes blocos. Por fim o servidor verifica a integridade do ficheiro através de algoritmos de sumário.

- Na listagem de certificados, como está exemplificado na figura 4.8, após a sua selecção, incluirá a possibilidade de consultar os detalhes correspondentes.
- Desenvolvimento de um módulo gráfico para validação de assinaturas, podendo este ser integrado na interface apresentada.
- Implementação do último nível da assinatura XAdES a XAdES-A, bem como a correspondente validação.
- Implementação do módulo de preenchimento automático de documentos.
- Desenvolvimentos de módulos de assinatura similar ao apresentado na secção de assinatura de documentos PDF (4.3.3), mas para outro tipo de documentos tais como: DOC, XLS, etc.
- Melhoramento das barras de progressão e apresentação de estimativas para a duração temporal do upload.

O futuro do desenvolvimento científico e tecnológico é imprevisível, embora construído sobre a pirâmide dos vários contributos. A aplicabilidade das soluções deverá ser dinâmica e poder incorporar novos elementos por forma a responder às solicitações permanentes. O presente trabalho foi desenvolvido tendo sempre presente este pressupostos, dando a sua contribuição para atingir um novo ponto de viragem na escalada do desenvolvimento.

Bibliografia

- [1] *ICT for Government and Public Services*. Obtido em 11 de Outubro, de http://ec.europa.eu/information_society/activities/egovernment/index_en.htm.
- [2] *Decreto-Lei n° 290-D/99*. 1999. Obtido em 11 de Outubro, de http://www.ancp.gov.pt/Legislacao/Documents/DL_290_D_992.pdf.
- [3] *The Estonian ID Card and Digital Signature Concept - Principles and Solutions*. 2003. Obtido em 11 de Outubro, de http://www.id.ee/public/The_Estonian_ID_Card_and_Digital_Signature_Concept.pdf.
- [4] *Decreto-Lei n° 62/2003*. 2007. Obtido em 11 de Outubro, de <http://dre.pt/pdf1sdip/2003/04/079A00/21702185.pd>.
- [5] *Decreto-Lei n.o 143-A/2008*. 2008. Obtido em 11 de Outubro, de <http://dre.pt/pdf1sdip/2008/07/14301/0000200006.PDF>.
- [6] *Decreto-Lei n° 62/2003*. 2008. Obtido em 11 de Outubro, de <http://dre.pt/pdf1s/2008/01/02000/0075300852.pdf>.
- [7] *Portaria 701-G/2008*. 2008. Obtido em 11 de Outubro, de <http://www.base.gov.pt/codigo/Documents/Portaria701G2008.pdf>.
- [8] ANSI. *ANSI X9.52-1998*. 1998. Obtido em 11 de Outubro, de <http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI+X9.52:1998>.
- [9] J. Brainard, A. Juels, R. Rivest, M. Szydlo, and M. Yung. *Fourth-Factor Authentication: Somebody You Know*. 2006. Obtido em 11 de Outubro, de <http://www.rsa.com/rsalabs/staff/bios/ajuels/publications/fourth-factor/ccs084-juels.pdf>.

- [10] J. Callas, L. Donnerhacke, IKS GmbH, H. Finney, D. Shaw, and R. Thayer. *OpenPGP Message Format*. Number 4880. 2007. Obtido em 11 de Outubro, de <http://www.ietf.org/rfc/rfc4880.txt>.
- [11] Fátima Carrão. *[Cartão de Cidadão] - Estrutura pessoal de dados. (Contacto Pessoal)*. 2009.
- [12] Austrian Federal Chancellery. *The ABC guide of eGovernment in Austria*. 2008. Obtido em 11 de Outubro, de http://www.inst-informatica.pt/servicos/informacao-e-documentacao/biblioteca-digital/areas-aplicacionais/administracao-publica-electronica/2008/administration-on-the-net-the-abc-guide-of/at_download/file.
- [13] Miguel Reis Calisto da Silva. *Um Sistema de Troca Segura de Mensagens com Garantias Fortes de Auditabilidade*. 2004.
- [14] Raúl Carvalho das Neves. *The ABC guide of eGovernment in Austria*. 2006. Obtido em 11 de Outubro, de <http://www.ife.pt/po/congres/smartca/docs/Ra%C3%BA1%20Neves%20-%20SIBS.pdf>.
- [15] A Autoridade Nacional de Segurança. *Relação das Entidades Certificadoras registadas na Autoridade Credenciadora*. 2009. Obtido em 11 de Outubro, de http://www.gns.gov.pt/NR/rdonlyres/B85421BE-F5B8-478F-9E23-5C312241F73C/0/Rela%C3%A7%C3%A3o_de_EC_registadas_na_AC.pdf.
- [16] Whitfield Diffie and Martin E. Hellman. *New Directions in Cryptography*, volume IT-22. 1976. Obtido em 11 de Outubro, de [cite-seer.ist.psu.edu/diffie76new.html](http://seer.ist.psu.edu/diffie76new.html).
- [17] Cartão do Cidadão. *Autenticação com o Cartão de Cidadão*. 2008. Obtido em 11 de Outubro, de http://www.cartaocidadao.pt/images/stories/Manual%20Autenticacao%20com%20Cartao%20de%20Cidadao_%20v1.7.pdf.
- [18] Cartão do Cidadão. *Manual de Utilização da Aplicação do Cartão de Cidadão*. 2009. Obtido em 11 de Outubro, de http://www.cartaocidadao.pt/media/Manual_Utilizacao_Cartao_Cidadao_v121.pdf.

- [19] Cartão do Cidadão. *Manual técnico do Middleware Cartão de Cidadão*. 2009. Obtido em 11 de Outubro, de http://www.cartaodecidadao.pt/images/cc_manual_technical_v1.21.pdf.
- [20] Cartão do Cidadão. *O novo documento de identificação dos cidadãos portugueses [nota informativa - 1]*. 2009. Obtido em 11 de Outubro, de http://www.cartaodecidadao.pt/images/stories/cc_nota_informativa.pdf.
- [21] Presidência do concelho de ministros. *Orientações para a simplificação*. 2009. Obtido em 11 de Outubro, de <http://www.simplex.pt/downloads/orientacoesideiasimplex.pdf>.
- [22] Parlamento e Conselho Europeu. *XML Advanced Electronic Signatures (XAdES)*. 1999. Obtido em 11 de Outubro, de http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=12532.
- [23] C. Ellison. *SPKI Requirements*. Number 2692. 1999. Obtido em 11 de Outubro, de <http://tools.ietf.org/html/rfc2692>.
- [24] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *SPKI Certificate Theory*. Number 2693. 1999. Obtido em 11 de Outubro, de <http://tools.ietf.org/html/rfc2693>.
- [25] ELPAIS.com. *Caja Madrid prepara sus cajeros para el DNI electrónico*. 2007. Obtido em 11 de Outubro, de http://www.elpais.com/articulo/internet/Caja/Madrid/prepara/cajeros/DNI/electronico/elpeputec/20070403elpepunet_6/Tes.
- [26] Portal empresa. *FAQ*. 2009. Obtido em 11 de Outubro, de http://www.portaldaempresa.pt/CVE/pt/Geral/faqs/EmpresaOnline/Criacao_da_Empresa_Online/#{68F9DB60-63F6-4F93-83AA-95BA0E27BB46}.
- [27] EMVCo. *EMV - Application Independent ICC to Terminal Interface Requirements*. EMVCo, fourth edition, 2008.
- [28] Alan O. Freier, Philip Karlton, and Paul C. Kocher. *The SSL Protocol Version 3.0*. November 1996. Obtido em 11 de Outubro, de <http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt>.

- [29] R. Housley, W. Polk, NIST, W. Ford, VeriSign, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. Number 3280. 1999. Obtido em 11 de Outubro, de <http://www.ietf.org/rfc/rfc3280.txt>.
- [30] RSA Laboratories. *RSA Cryptography Standard*. 2002. Obtido em 11 de Outubro, de <ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf>.
- [31] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. Number 2560. 1999. Obtido em 11 de Outubro, de <http://www.ietf.org/rfc/rfc2560.txt>.
- [32] NIST. *NIST Federal Information Processing Standard Publication 180-1: Secure Hash Standard*. May 1994.
- [33] National Institute of Standards and Technology. *DES MODES OF OPERATION*. 1980. Obtido em 11 de Outubro, de <http://www.itl.nist.gov/fipspubs/fip81.htm>.
- [34] National Institute of Standards and Technology. *ADVANCED ENCRYPTION STANDARD*. 2001. Obtido em 11 de Outubro, de <http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.
- [35] National Institute of Standards and Technology. *Proposed Withdrawal of FIPS for the Data Encryption Standard*. 2001. Obtido em 11 de Outubro, de <http://www.itl.nist.gov/fipspubs/FR%2007262004.pdf>.
- [36] European parliament. *Community framework for electronic signatures*. 1999. Obtido em 11 de Outubro, de http://eur-lex.europa.eu/smartapi/cgi/sga_doc?smartapi!celexapi!prod!CELEXnumdoc&numdoc=31999L0093&model=guichett&lg=en.
- [37] R.L. Rivest, A. Shamir, and L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. 2008. Obtido em 11 de Outubro, de <http://people.csail.mit.edu/rivest/Rsapaper.pdf>.

- [38] Artur Romão, Miguel Mira da Silva, Alberto Silva, and Nuno Conde. *Comércio Electrónico na Internet*. LIDEL, second edition, 2001.
- [39] Bruce Schneier. *Two-factor authentication: too little, too late*, volume 48. 2005.
- [40] SEMIC.EU. *Person Data Structure*. Obtido em 11 de Outubro, de <http://www.semic.eu/semic/view/Asset/downloadAssessmentReport.xhtml?releaseId=210&id=11>.
- [41] R. Shirey. *Internet Security Glossary*. Number 2828. 2000. Obtido em 11 de Outubro, de <http://www.ietf.org/rfc/rfc2828.txt>.
- [42] DigiDoc Format Specification. *openXAdES*. 2004. Obtido em 11 de Outubro, de <http://www.openxades.org/files/DigiDoc%20format%201.3.doc>.

Anexos

API

Interface

```
public interface Uma {

    public Xades removeSignature(InputStream xadesBA,OutputStream fileAttach)
        throws XadesException;
    public Xades removeSignature(InputStream xadesBA)
        throws XadesException;
    public OutputStream cipher(InputStream content, PublicKey publicKey)
        throws EncryptDocumentException;
    public OutputStream cipher(InputStream content, String pubKeyBase64)
        throws EncryptDocumentException;
    public OutputStream cipher(InputStream content, byte[] x509ByteArray)
        throws EncryptDocumentException;
    public InputStream decipher(InputStream cipherMessage, byte[] pkcs12,
        char[] password)
        throws DecipherException;
    public void decipher(InputStream cipherMessage, OutputStream decipherMessage,
        KeyStoreSelected keyStoreSelected)
        throws DecipherException;
    public InputStream sign(String filename, ByteArrayOutputStream attachment,
        byte[] pkcs12, char[] password)
        throws BuildXadesException;
    public InputStream separateSignAndFile(String filename, InputStream file,
        byte[] pkcs12, char[] password)
        throws BuildXadesException;
    public List<Integer> validateFilewithXades(InputStream xadesBA,InputStream file)
        throws XadesException;
    public boolean validatePassPFX(String certificate, String password)
        throws FileNotFoundException, PKCSException, IOException;
    public void singAndTimeStamp(InputStream input, OutputStream out,
        InputStream keyStorePFX, String passwordPFX,String reason,
        String location, String timestampServer, String sServer,
```

```

        String serverResponseEncondig, String timeStampTag,
        String timeStampProperty, String hashAlgorithm)
        throws SignPDFException, net.saphety.cipher.exception.TimeStampException ;
    public void signAndTimestamp(InputStream input, OutputStream out,
        InputStream keyStorePFX, String passwordPFX,String reason,
        String location, String timeStampServer)
        throws SignPDFException, TimeStampException ;
    public List<Integer> pfxValidator(KeyStoreSelected keyStoreSelected)
        throws PasswordException, TimeStampException, PfxValidatorException;
    public byte[] generatePFX(String country, String organization,
        String oraganizationUnit, String emailAddress, String commonName,
        String passWord, Map<String, String> issuerBy)
        throws GeneratePfxException;
    public void verifyJCE() throws JCEException;
    public byte[] timeStampRequestSaphety(byte[] request) throws TimeStampException;
    public byte[] timeStampRequest(byte[] request,TimeStampRequester timeStampRequest)
        throws TimeStampException;
    public void generateEnvelopeFile(Map<String, String> parameters,InputStream data,
        OutputStream envelopeOS)
        throws FileEnvelopeException;
    public void readDataEnvelopeFile(InputStream dataIS , OutputStream datatOS)
        throws FileEnvelopeException;
    public Map<String, String> readMetaDataEnvelopeFile(InputStream dataIS)
        throws FileEnvelopeException;
    public boolean isXmlEnvelope(InputStream dataIS );
    public void downloadAndInstall(String JCEBasePath) throws JCEInstallException;
}

```

Gestor PKI

Class KeyStoreSelected

```

public class KeyStoreSelected {

    private PrivateKey privateKey = null;
    private X509Certificate certificate = null;
    private X509Certificate[] chain = null;

    public PrivateKey getPrivateKey() {
        return privateKey;
    }
    public void setPrivateKey(PrivateKey privateKey) {
        this.privateKey = privateKey;
    }
    public X509Certificate getCertificate() {
        return certificate;
    }
}

```

```

public void setCertificate(X509Certificate certificate) {
    this.certificate = certificate;
}
public X509Certificate[] getChain() {
    return chain;
}
public void setChain(X509Certificate[] chain) {
    this.chain = chain;
}
}

```

Cifra

Class EncryptDocument

```

public class EncryptDocument {

    static DocumentBuilderFactory dbf = null;
    public String x509Certificate = null;
    public PublicKey publickey = null;
    private static KeyGenerator kg;

    public EncryptDocument(String cert) throws EncryptDocumentException {
        x509Certificate = cert;
        setPublickey(getX509Certificate().getPublicKey());
    }

    public EncryptDocument( PublicKey publicKey ){
        setPublickey(publicKey);
    }

    public EncryptDocument( byte[] x509ByteArray ) throws EncryptDocumentException {
        X509Certificate x509Certificate = null;
        try {
            x509Certificate = new X509Certificate(x509ByteArray);
            setPublickey(x509Certificate.getPublicKey());
        } catch (CertificateException e) {
            Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
            throw new EncryptDocumentException(e);
        }
    }

    public void cypher(InputStream xades, OutputStream outputStream) throws EncryptDocumentException {
        ToBeEncryptedUctetStream tbeData=null;
        try{

```

```

//does all provider registering for you

XSecProvider xsect = new XSecProvider();

Security.addProvider(xsect);

//First, we create an instance of an XMLEncryptionFactory.
//This factory is used to encrypt the desired xml and marshal the result
//using the DOM mechanism.
XMLEncryptionFactory xfac= null;

xfac = XMLEncryptionFactory.getInstance("DOM",xsect);

//The KeyInfoFactory needed for the Key retrieval process.
KeyInfoFactory kif = KeyInfoFactory.getInstance("DOM",xsect);

//Create a new document to insert the encrypted data into
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setNamespaceAware(true);

dbf.setNamespaceAware(true);

DocumentBuilder db = dbf.newDocumentBuilder();
Document plainDoc = db.newDocument();
Element envelope = plainDoc.createElementNS
("http://example.org/envelope", "Envelope");
envelope.setAttributeNS
("http://www.w3.org/2000/xmlns/", "xmlns", "http://example.org/envelope");
plainDoc.appendChild(envelope);
Element content = plainDoc.createElementNS
("http://exampl.org/content", "Content");
content.setAttributeNS
("http://www.w3.org/2000/xmlns/", "xmlns", "http://example.org/content");
content.setTextContent("Document cipher by Bizgov");
envelope.appendChild(content);

/**
 * Start encrypt Key
 */

SecretKey key = extracted(xfac, kif, content);

/**
 * Start encrypt Data

```

```

    */

    encryptDataStream(xades, outputStream, plainDoc, key);

} catch (ParserConfigurationException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (InvalidAlgorithmParameterException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (NoSuchAlgorithmException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (MarshalException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (XMLEncryptionException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (TransformerConfigurationException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (TransformerException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (IOException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (NoSuchPaddingException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (InvalidKeyException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (IllegalBlockSizeException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} catch (BadPaddingException e) {
    Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new EncryptDocumentException(e);
} finally{
    IOUtils.closeQuietly(xades);
    IOUtils.closeQuietly(outputStream);
}
}

private void encryptDataStream(InputStream xades,

```

```

        OutputStream outputStream, Document plainDoc, SecretKey key)
        throws TransformerException, IOException, NoSuchAlgorithmException,
        NoSuchPaddingException, InvalidKeyException, InvalidAlgorithmParameterException,
        IllegalBlockSizeException, BadPaddingException {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
    DOMUtils.serialize(plainDoc, byteArrayOutputStream);
    String message = new String(byteArrayOutputStream.toByteArray());
    String beginString = StringUtils.substringBeforeLast(message, "</Envelope>")+
    "<EncryptedData Id=\"Data\" xmlns=\"http://www.w3.org/2001/04/xmlenc#\"><EncryptionMethod Algorithm=
    \"http://www.w3.org/2001/04/xmlenc#aes128-cbc\"/><CipherData><CipherValue>";
    String afterString = "</CipherValue></CipherData></EncryptedData></Envelope>";
    NonClosableOutputStream nonClosableOutputStream = new NonClosableOutputStream(outputStream);
    nonClosableOutputStream.write(beginString.getBytes());

    Provider provider = Security.getProvider("IAIK");
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding", provider);
    cipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] iv = cipher.getIV();

    Base64.OutputStream base64OutputStream = new Base64.OutputStream(nonClosableOutputStream, Base64.ENCODING);
    CipherOutputStream cipherOutputStream = new CipherOutputStream(base64OutputStream, cipher);

    cipherOutputStream.write(iv);
    IOUtils.copyLarge(xades, cipherOutputStream);
    IOUtils.closeQuietly(xades);
    IOUtils.closeQuietly(cipherOutputStream);

    outputStream.write(afterString.getBytes());
    nonClosableOutputStream.finished();
    IOUtils.closeQuietly(outputStream);
}

private SecretKey extracted(XMLEncryptionFactory xfac, KeyInfoFactory kif,
    Element content) throws InvalidAlgorithmParameterException,
    NoSuchAlgorithmException, MarshalException, XMLEncryptionException {
    //We use Exclusive Canonicalization with comments.
    CanonicalizationMethod canM = xfac.newCanonicalizationMethod(
        CanonicalizationMethod.EXCLUSIVE_WITH_COMMENTS, null);

    //A KeyName element with the name "MedKey" is produced.
    KeyName medKeyName = kif.newKeyName("PayKey1");

    // Create an AES Key
    if (kg == null){
        kg = KeyGenerator.getInstance("AES");
    }
}

```

```

    }
    kg.init(128);
    SecretKey key = kg.generateKey();

    //Specify the data to be encrypted
    ToBeEncryptedKey tbeKey = new ToBeEncryptedKey(key);

    //Add the KeyName to the KeyInfo.
    KeyInfo ki = kif.newKeyInfo(Collections.nCopies(1, medKeyName));

    //Determine the first encryption method.
    EncryptionMethod em = xfac.newEncryptionMethod(EncryptionMethod.RSA_1_5, null, null);

    //The EncryptedData is built, using the above items.
    EncryptedKey ed = xfac.newEncryptedKey(tbeKey, em, null, null, null, "Key", null, null);

    //Set DOMEncryptContext using the empty document the encrypted data
    //should be included into
    //    DOMEncryptContext ctxt1 = new DOMtoBeEncryptContext(new MedInfoKeySelector(), plainDoc);
    DOMEncryptContext ctxt1 = new DOMEncryptContext(getPublicKey(), content);

    //encrypt
    ed.encrypt(ctxt1);
    return key;
}

/**
 * Gets the x509 certificate
 * @throws EncryptDocumentException
 * @throws AppletException
 * @throws IOException
 * @throws CertificateException
 * @throws CertificateException
 */
public X509Certificate getX509Certificate() throws EncryptDocumentException {

    if (x509Certificate.contains("BEGIN CERTIFICATE")) {
        x509Certificate.matches("(~)+(BEGIN CERTIFICATE)(~)+");
        x509Certificate = x509Certificate.replaceAll("(~)+(BEGIN CERTIFICATE)(~)+", "");
        x509Certificate = x509Certificate.replaceAll("(~)+(END CERTIFICATE)(~)+", "");
    }

    byte[] certBA;
    try {

        certBA = new sun.misc.BASE64Decoder().decodeBuffer(x509Certificate);
        X509Certificate x509Certificate = null;

```

```

        x509Certificate = new X509Certificate(certBA);
        return x509Certificate;

    } catch (IOException e) {
        Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
        throw new EncryptDocumentException(e);
    } catch (CertificateException e) {
        Logger.getLogger(EncryptDocument.class.getName()).log(Level.SEVERE, null, e);
        throw new EncryptDocumentException(e);
    }
}

public PublicKey getPublickey() {
    return publickey;
}

public void setPublickey(PublicKey publickey) {
    this.publickey = publickey;
}
}

```

Class DecryptDocument

```

public class DecryptDocument {

    private PrivateKey privateKey = null;

    public DecryptDocument() {
    }

    public void decrypt(InputStream cipherMessage, OutputStream out,
        KeyStoreSelected keyStoreSelected)
        throws DecipherException {

        XMLInputFactory xmlif = null;

        xmlif = XMLInputFactory.newInstance();
        Base64.OutputStream outputDecipher = null;
        CipherOutputStream cipherOutputStream = null;
        try {
            xmlif.setProperty(XMLInputFactory.IS_REPLACING_ENTITY_REFERENCES, Boolean.TRUE);
            xmlif.setProperty(XMLInputFactory.IS_SUPPORTING_EXTERNAL_ENTITIES, Boolean.FALSE);
            xmlif.setProperty(XMLInputFactory.IS_NAMESPACE_AWARE, Boolean.TRUE);

```



```

xmlif.setProperty(XMLInputFactory.IS_COALESCING , Boolean.TRUE);

Key key=null;

XMLInputFactory factory = XMLInputFactory.newInstance();
XMLStreamReader parser = factory.createXMLStreamReader(cipherMessage);

int eventType = parser.getEventType();
boolean haveKey = false;
String keyString = "";

Cipher cipherBody;

Security.getProvider("IAIK");
cipherBody = Cipher.getInstance("AES/CBC/PKCS5Padding");

while(parser.hasNext()) {
    eventType = parser.next();

    if(parser.getEventType()==1 && parser.getLocalName().equals("CipherValue") && !haveKey ){
        while(parser.hasNext()) {
            eventType = parser.next();
            if(eventType==2){

                Cipher deCipher;
                byte[] decodeBuffer = new sun.misc.BASE64Decoder().decodeBuffer(keyString);
                deCipher = Cipher.getInstance("RSA");
                deCipher.init(Cipher.DECRYPT_MODE, keyStoreSelected.getPrivateKey());
                byte[] keyBA = deCipher.doFinal(decodeBuffer);
                key = new javax.crypto.spec.SecretKeySpec(keyBA,"AES");
                haveKey= true;
                // System.arraycopy(key, 0, iv, 0, 16);
                // byte[] iv = Arrays.copyOf(keyBA, 16);
                /**
                 * initialize iv
                 */
                byte[] iv = new byte[16];
                AlgorithmParameterSpec paramSpec = new IvParameterSpec(iv);
                cipherBody.init(Cipher.DECRYPT_MODE, key,paramSpec);

                break;
            }else if(eventType==4){
                keyString += parser.getText();
            }
        }
    }
}
}

```

```

    if(parser.getEventType()==1 && parser.getLocalName().equals("CipherValue") && haveKey ){
        cipherOutputStream = new CipherOutputStream(out,cipherBody);
        outputDecipher = new Base64.OutputStream(cipherOutputStream,Base64.DECODE);
        while(parser.hasNext()) {
            eventType = parser.next();
            if(eventType==2){
                outputDecipher.suspendEncoding();
                break;
            }else if(eventType==4){
                String s = parser.getText();
                outputDecipher.write(s.getBytes());
            }
        }
    }
}
}
} catch (InvalidKeyException e) {
    Logger.getLogger(DecryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new DecipherException("Error decrypt document: " + e.getMessage(), e.getCause());
} catch (IllegalArgumentException e) {
    Logger.getLogger(DecryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new DecipherException("Error decrypt document: " + e.getMessage(), e.getCause());
} catch (NoSuchAlgorithmException e) {
    Logger.getLogger(DecryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new DecipherException("Error decrypt document: " + e.getMessage(), e.getCause());
} catch (NoSuchPaddingException e) {
    Logger.getLogger(DecryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new DecipherException("Error decrypt document: " + e.getMessage(), e.getCause());
} catch (IllegalBlockSizeException e) {
    Logger.getLogger(DecryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new DecipherException("Error decrypt document: " + e.getMessage(), e.getCause());
} catch (BadPaddingException e) {
    Logger.getLogger(DecryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new DecipherException("Error decrypt document: " + e.getMessage(), e.getCause());
} catch (InvalidAlgorithmParameterException e) {
    Logger.getLogger(DecryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new DecipherException("Error decrypt document: " + e.getMessage(), e.getCause());
} catch (FactoryConfigurationError e) {
    Logger.getLogger(DecryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new DecipherException("Error decrypt document: " + e.getMessage(), e.getCause());
} catch (XMLStreamException e) {
    Logger.getLogger(DecryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new DecipherException("Error decrypt document: " + e.getMessage(), e.getCause());
} catch (IOException e) {
    Logger.getLogger(DecryptDocument.class.getName()).log(Level.SEVERE, null, e);
    throw new DecipherException("Error decrypt document: " + e.getMessage(), e.getCause());
}finally{
    IOUtils.closeQuietly(cipherOutputStream);
}

```

```
        IOUtils.closeQuietly(outputDecipher);  
    }  
}  
  
}
```

Selos temporais

Class TimeStampRequester

```
public interface TimeStampRequester {  
    public byte[] getTimeStamp(byte[] hashed_message) throws TimeStampException;  
}
```