UNIVERSIDADE DO MINHO
Escola de Engenharia
Departamento de Informática

Marca Do Dia Electrónica em Smartphones

Implementação de um serviço para envio de
correio electrónico com marcas temporais

Ricardo Jorge Gomes Maia de Sousa

Dissertação de Mestrado em Engenharia Informática

Trabalho efectuado sob a orientação do
Professor Doutor Manuel Bernardo Barbosa

Janeiro de 2012

Agradecimentos

Em primeiro lugar queria agradecer ao meu orientador, o professor Manuel Bernardo Barbosa, pelas revisões cuidadosas e correcções sugeridas para a dissertação, bem como pela constante boa disposição e entusiasmo que sempre apresenta e que certamente foram decisivos para o sucesso deste trabalho.

Por todo o apoio demonstrado, pela sua companhia e pela enorme paciência, gostaria de agradecer à Filipa Costa, que nas mais complexas ocasiões foi capaz de me animar e fazer perceber todas as dificuldades podem ser ultrapassadas.

De igual forma, agradeço à minha família por sempre acreditarem em mim e por nunca se terem imposto no meu rumo pessoal e profissional, o que a meu ver é algo de extremo valor.

Também aos meus amigos devo um muito obrigado, em especial ao Francisco Ribeiro e ao Tiago Martins por já cerca de uma década de amizade, sempre prontos para me motivar em situações que requeriam maior concentração, como a escrita desta dissertação.

Pretendia ainda agradecer aos colegas de mestrado Arley Pinto e Miguel Lopez pela sua contribuição na implementação do módulo de segurança em software, assim como ao Stefan Selbitschka, director da empresa rundQuadrat OG, pela colaboração na solução para assinatura digital de *e-mails* em Android.

Por fim, agradeço à MULTICERT pela atenção e ajuda indispensável, com um especial obrigado ao João Cerdeira, Nuno Santos e ao Renato Portela, que sempre foram prestáveis para qualquer dúvida ao longo do desenvolvimento da solução.

Abstract

The *Marca Do Dia Electrónica* (MDDE) is a service which allows the use of e-mail with a high level of security and confidence, jointly developed by CTT and MULTICERT. This service, currently available for personal computers with the Windows operating system only, consists in assigning an electronic timestamp to the message being sent, in order to endorse the veracity of the time and date of dispatch. This way, it also provides evidence that the message was not changed, given the integrity and non-repudiation properties assured by the electronic timestamp.

Due to the impact of smartphones in today's society, which increasingly tend to be the dominant mean of communication through the Internet, the main purpose of this dissertation is to explore the prospects of using a mobile device as an ubiquitous solution for sending MDDE messages, taking advantage of a microSD memory card with cryptographic capabilities (i.e. a Smart Card) for storing and handling the necessary electronic credentials.

In order to specify this solution, we present a study on the major mobile platforms, with special emphasis on the Android operating system. We also describe the Java Card framework, as well as the Smart Cards' communication and data protocols. Finally, an implementation is presented as proof of concept that the proposed solution represents a viable alternative to the existing application, achieved through the development of two components — an Android application and a Java Card applet.

Keywords: Android, Cryptography, Java Card, MDDE, Smart Card, Smartphone, Timestamping

Resumo

A Marca Do Dia Electrónica (MDDE) trata-se de um serviço, desenvolvido em parceria pelos CTT e pela MULTICERT, que possibilita a utilização do correio electrónico com um elevado grau de segurança e fiabilidade. Este serviço, apenas disponível para computadores pessoais com o sistema operativo Windows, consiste na colocação de um selo temporal electrónico nas mensagens enviadas, de forma a certificar a veracidade da data e hora de envio das mesmas. Desta forma, é também assegurado que a mensagem enviada não foi alterada, pelas propriedades de integridade e não repúdio apresentadas pelo selo electrónico.

Dado o impacto dos *smartphones* na sociedade actual, que cada vez mais tendem para representar o meio dominante de comunicação através da Internet, o objectivo principal desta dissertação é explorar a possibilidade de utilizar um dispositivo móvel como solução ubíqua para o envio de mensagens com MDDE, recorrendo a um cartão de memória *microSD* com capacidades criptográficas (i.e. um *Smart Card*), para o armazenamento e utilização das credenciais electrónicas necessárias.

Para a especificação desta solução, são estudadas as principais plataformas móveis, com especial ênfase no sistema operativo Android. É também analisada a *framework* Java Card, bem como os protocolos de dados e de comunicação utilizados pelos *Smart Cards*. Por fim, é apresentada uma implementação como prova de conceito de que a solução proposta é uma alternativa viável à aplicação existente, concretizado através de duas componentes — uma aplicação para Android e uma *applet* para Java Card.

Palavras-chave: Android, Criptografia, Java Card, MDDE, *Smart Card*, *Smartphone*, *Timestamping*

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ADB *Android Debug Bridge*

AES *Advanced Encryption Standard*

AID *Application Identifier*

APDU *Application Protocol Data Unit*

API *Application Programming Interface*

BCC Cópia Oculta (do inglês *Blind Carbon Copy*)

CA Autoridade de Certificação (do inglês *Certification Authority*)

CC Com Conhecimento (do inglês *Carbon Copy*)

CMS *Cryptographic Message Syntax*

CPU Unidade Central de Processamento (do inglês *Central Processing Unit*)

CRL Lista de Revogação de Certificados (do inglês *Certificate Revocation List*)

CTT Correios de Portugal (da sigla Correios, Telégrafos e Telefones)

DNS Sistema de Nomes de Domínios (do inglês*Domain Name System*)

DTS *Digital Timestamping*

EPCM *Electronic Postal Certification Mark*

EEPROM *Electrical Erasable Programmable Read Only Memory*

GPS Sistema de Posicionamento Global (do inglês *Global Positioning System*)

GUI Interface Gráfica (do inglês *Graphical User Interface*)

HTTP *HyperText Transfer Protocol*

HTTPS *HyperText Transfer Protocol Secure*

IDE Ambiente Integrado de Desenvolvimento (do inglês *Integrated Development Environment*)

IMAP *Internet Message Access Protocol*

ITU-T Sector de Uniformização das Telecomunicações da *International Telecommunication Union*

JCDK *Java Card Development Kit*

JCRE *Java Card Runtime Environment*

JCVM *Java Card Virtual Machine*

JDK *Java Development Kit*

MAC Message Authentication Code

MDDE Marca Do Dia Electrónica

MIME *Multipurpose Internet Mail Extensions*

NPU Coprocessador Criptográfico (do inglês *Numeric Proccessing Unit*)

NTP *Network Time Protocol*

OAL Observatório Astronómico de Lisboa

OS Sistema Operativo (do inglês *Operating System*)

PDA Assistente Pessoal Digital (do inglês *Personal Digital Assistant*)

PKCS *Public-Key Cryptography Standards*

PKI Infraestrutura de Chaves Públicas (do inglês *Public Key Infrastructure*)

POP3 *Post Office Protocol*

RAM *Random Access Memory*

ROM *Read Only Memory*

RSA *Rivest-Shamir-Adleman*

SDK *Software Development Kit*

SIM *Subscriber Identity Module*

S/MIME *Secure/Multipurpose Internet Mail Extensions*

SMTP *Simple Mail Transport Protocol*

SSL *Secure Sockets Layer*

TLS *Transport Layer Security*

TSA Autoridade de *Timestamping*

TTP Terceira Entidade de Confiança (do inglês *Trusted Third Party*)

TTS *Trusted Timestamping*

UPU *Universal Postal Union*

URL *Uniform Resource Locator*

URI *Uniform Resource Identifier*

XML *Extensible Markup Language*

CAPÍTULO 1

Introdução

Contextualização

A comunicação é um dos pilares da nossa sociedade, e a eficiência da troca de informação no mundo empresarial um factor decisivo. Entre as inúmeras formas existentes para a troca de mensagens a longa distância, um dos meios mais frequentemente utilizados é o recurso a serviços postais. No entanto, a necessidade de comunicação à distância não é um problema actual, tendo surgido muito antes da criação dos primeiros sistemas de correio. Uma das raízes da troca organizada de documentos escritos remonta ao século 14 a.C., quando no Egipto os faraós decidiram usar mensageiros para a difusão de decretos-lei por todo o estado. Alguns anos depois, o imperador romano *Augustus Caesar* criou o "*Cursus Publicus*", agora reconhecido como tendo sido o primeiro serviço postal, que se distinguia pela ampla rede de estradas que suportavam o serviço[4].

Hoje em dia, os serviços postais estão presentes em quase todos os países do mundo, disponibilizando serviços de troca de correio — nacional e internacionalmente — com algumas garantias (quando solicitadas), como provas de expedição, acompanhamento de pedidos e avisos de recepção. Seja para uso pessoal ou comercial, praticamente toda a gente já enviou e recebeu cartas por este sistema, o que evidencia a eficiência e a conveniência de tais serviços.

Com a expansão da Internet, surgiram vários meios de comunicação inovadores, sendo a solução mais notável o correio electrónico (ou *e-mail*), introduzido em 1972 como o propósito de facilitar a coordenação entre a equipa de desenvolvimento da ARPANET[1][23]. Dado que,

[1] ARPANET — *Advanced Research Projects Agency Network*, a antecessora da Internet

em comparação ao correio postal, o *e-mail* é uma forma mais barata (geralmente gratuita) e mais rápida de enviar e receber correspondência, existe uma tendência a optar por esta opção, sempre que possível. A utilidade deste serviço e a facilidade de utilização do mesmo são tão notórias que não é difícil perceber porque é que, actualmente, todas as entidades com presença na Internet possuem pelo menos um endereço de correio electrónico.

Apesar da rápida difusão do *e-mail* como principal forma de troca de informação electrónica, algumas preocupações são levantadas quanto à autenticidade (e por vezes, privacidade) das mensagens trocadas — que garantias existem de que o correio electrónico não terá sido interceptado desde o seu remetente e, consequentemente, comprometido? Como demonstrado em [41], existe a possibilidade de um *e-mail* ser interceptado e adulterado, o que nalguns casos (principalmente em situações empresariais) justifica o recurso a soluções complementares de protecção da informação, tais como os serviços de *Electronic Postal Certification Mark* (EPCM).

Um serviço EPCM consiste na equivalente electrónica do serviço postal de correio registado, com as mesmas funcionalidades de prova de envio e identificação unívoca das mensagens trocadas por correio registado. Tem como base a aplicação de técnicas de *Digital Timestamping* (DTS) em documentos electrónicos, com o objectivo de permitir provar que uma certa informação existia a uma dada hora e data (i.e. a hora e data em que foi criado o *timestamp* do documento). Conforme descrito em [1], o *timestamp* da mensagem é construído com recurso a técnicas criptográficas (e.g. assinaturas digitais) e, para fornecer a garantia da hora e data de envio, são habitualmente utilizados servidores de hora e data externos. O uso de servidores de hora e data confiáveis na construção dos *timestamps* é crucial, pois grande parte da segurança destes serviços é baseada na disponibilidade e fiabilidade do relógio utilizado para a obtenção da data e hora [22].

A Marca Do Dia Electrónica (MDDE)[2] é um serviço EPCM desenvolvido pelos Correios de Portugal (CTT) e pela MULTICERT. O objectivo a que este serviço se propõe é disponibilizar a utilização do *e-mail* para suporte à troca de mensagens com provas de envio para o remetente e autenticidade (sobre quem envia) para o receptor, sendo que a hora e data, indicadas nas mensagens trocadas por este serviço, são obtidas através de uma terceira parte

[2] https://sce.ctt.pt/mdde/index.html

(autoridade) cuja confiabilidade está assente em bases legais — o Observatório Astronómico de Lisboa (OAL). A veracidade das mensagens trocadas por este serviço, com o comprovativo temporal da MDDE, é passível de ser confirmada *online* numa página disponibilizada para o efeito[3], onde pode ser confirmada a integridade da mensagem, a autenticidade do remetente e a legitimidade da hora e data indicada [6].

Actualmente, apesar de o serviço MDDE ter sido desenvolvido para utilização tanto a nível empresarial como por parte de individuais, este é maioritariamente utilizado por advogados inscritos na Ordem dos Advogados. Com a utilização do serviço por estas entidades, é assim possível o envio de peças processuais para tribunal via correio electrónico, por exemplo. Após o envio de uma mensagem por este serviço, o remetente recebe também um comprovativo de envio que, além de informação relativa à mensagem enviada, contém ainda um número único para a MDDE, de forma a permitir a identificação unívoca da mensagem. Deste modo, o serviço MDDE apresenta uma funcionalidade semelhante à do serviço postal de correio registado, sendo uma alternativa digital útil para a troca de informação prioritária com garantias de unicidade e autenticidade da mensagem enviada.

Motivação

Para o envio de mensagens através do serviço MDDE, um utilizador necessita de ter instalado o *plugin Desktop* MDDE, que de momento apenas existe para o Sistema Operativo (OS) Windows. No entanto, a mobilidade tem tomado conta do quotidiano empresarial como solução para o problema da conectividade ubíqua, de forma a facilitar e acelerar as comunicações (e, consequentemente, o processo de negócio), sendo que cada vez há mais soluções e lógicas de negócio a serem transportadas para dispositivos móveis. Uma vez que o serviço MDDE também pode usufruir destas vantagens acrescidas da mobilidade, e dada a competitividade e necessidade de constante actualização no universo tecnológico, torna-se imprescindível a existência de uma alternativa viável para o *plugin* MDDE existente, capaz de permitir o envio de mensagens com MDDE a partir de um dispositivo móvel.

Entre os tipos de dispositivos móveis existentes de momento, os que têm tomado maior projecção ultimamente são os *smartphones*. Estes dispositivos tratam-se basicamente de tele-

[3] https://sce.ctt.pt/registration/validate.html

fones móveis de última geração, resultantes da combinação de características de um Assistente Pessoal Digital (PDA) com um telemóvel, com elevadas capacidades de conectividade e facilmente adaptados a novas funcionalidades pela instalação de aplicações. O facto de os *smartphones* poderem ser transportados para todo o lado e utilizados diariamente como ferramenta de comunicação primária [24, 21] justificam a razão pela qual são usados cada vez mais no meio empresarial, facilitando o acesso quase instantâneo a operações de negócio anteriormente apenas passíveis de serem realizadas num computador pessoal. Por outro lado, o facto de estes dispositivos serem cada vez mais populares torna-os num dos principais alvos de ataques [48], uma questão que não pode ser ignorada quando se considera um serviço como o MDDE, onde a autenticidade das mensagens trocadas é um factor essencial.

O *plugin Desktop* MDDE, para aceder a credenciais necessárias ao envio de mensagens através deste serviço, está desenvolvido de forma a utilizar componentes do próprio OS onde é executado. Estas credenciais, mais especificamente certificados digitais e chaves privadas correspondentes, são por isso geridas pelo OS, o que permite a integração de aplicações que as utilizem com diferentes géneros de soluções de gestão de credenciais, quer em *software* como em *hardware*. Uma vez que a gestão de credenciais e a realização de operações criptográficas são realizadas exteriormente no processo de envio de mensagens pelo serviço MDDE, as garantias de segurança são determinadas pelas características da solução utilizada, sendo que habitualmente são utilizados *smart cards* pelas propriedades que apresentam. Porém, será possível alcançar uma solução equivalente a esta, capaz de fornecer as mesmas garantias em dispositivos móveis como os *smartphones*?

Nos últimos anos, diversas áreas de negócio têm recorrido a *smart cards* para colmatar várias falhas de segurança presentes noutras soluções. A capacidade de processar operações no *chip* destes cartões e a protecção física contra adulteração e cópia de dados armazenados são duas das principais propriedades que os distinguem de outros módulos de segurança físicos (e.g. cartões magnéticos). Apesar de esta tecnologia estar mais presente na área bancária e de telecomunicações (como por exemplo em cartões de crédito e cartões SIM), as vantagens apresentadas têm levado a que esta tecnologia seja utilizada para as mais variadas aplicações, desde chaves de entrada em edifícios a passaportes electrónicos.

Dos vários formatos de em que estes cartões se apresentam, o formato recente de *smart*

cards embutidos em cartões *microSD* tem demonstrado potencial, uma vez que permite a utilização destes cartões em dispositivos com entradas *microSD*, sem recurso a adaptadores ou outros leitores específicos. Isto facilita a utilização desta tecnologia em dispositivos móveis, uma vez que grande parte deste tipo de dispositivos possui entradas *microSD* e, desta forma, o próprio dispositivo móvel funciona como leitor de *smart cards*. Com o nível acrescido de protecção da informação armazenada nestes cartões e a capacidade de realizar operações criptográficas no próprio *chip*, surgem novas oportunidades de desenvolvimento de soluções móveis com garantias de segurança bastante superiores às possíveis de obter apenas com recurso a software.

Objectivos

Esta dissertação tem como principal objectivo estudar a possibilidade de utilização de dispositivos móveis — mais especificamente, de *smartphones* — como meio de envio de mensagens com MDDE. Uma vez que existe a preocupação com a segurança de uma possível solução nestes dispositivos e se pretende obter as mesmas funcionalidades do *plugin Desktop* já existente, é também necessário avaliar se será praticável o recurso a cartões *microSD* com capacidades criptográficas (*smart cards*) para o armazenamento e utilização de chaves criptográficas necessárias à construção de mensagens MDDE. Pretende-se portanto fazer um estudo do serviço MDDE e também analisar tanto os dispositivos móveis como os *smart cards*, com o propósito final de aplicar o estudo efectuado através da análise, concepção e implementação, com atenção a requisitos de segurança, de uma aplicação móvel de envio de MDDE.

A aproximação à questão da implementação é feita no OS móvel Android, conforme decidido em conjunto com a MULTICERT e estipulado o ambiente de desenvolvimento. No entanto, como objectivo de desenvolvimento, também está determinado que se pretende construir uma solução o mais genérica possível, de forma a garantir ou pelo menos facilitar a portabilidade da solução desenvolvida. Uma vez que se pretende obter uma solução portável, também foi estipulado que o *smart card* a utilizar deve suportar a personalização pela instalação de *applets*.

Contribuições

O resultado do estudo apresentado nesta dissertação é concretizado com uma aplicação em Android de envio de MDDE, totalmente funcional e completa, e uma *applet* para a plataforma JavaCard, para realização de assinaturas digitais em *smart cards*. É por isso demonstrado que o serviço pode ser transportado para o ambiente móvel, apesar das limitações impostas por este tipo de dispositivos (quando comparados com computadores pessoais).

A solução apresentada permite também alternar facilmente entre a utilização com módulo de segurança físico (com os *smart cards microSD*) ou com implementação em *software* (para o suporte às operações criptográficas necessárias). Pela comparação de características das alternativas e velocidades de operação das duas, é possível verificar que a solução com módulo de segurança físico, além de ser a opção preferencial em termos de protecção da informação privada, é ainda mais eficaz no cálculo da assinatura digital, o que demonstra a potencialidade destes cartões e motiva futuras aplicações com recurso a estes *smart cards microSD* em dispositivos móveis.

Estrutura da dissertação

No capítulo 2 é abordado o serviço MDDE, onde é descrito detalhadamente o funcionamento deste serviço e feita uma comparação ao serviço postal de correio registado. É dada especial atenção aos processos relativos ao envio e validação de mensagens trocadas pelo serviço MDDE, onde são também caracterizadas e verificadas as propriedades de segurança deste.

Ao longo do capítulo 3, é estudado o estado actual dos dispositivos móveis no mercado, com uma análise dos OS móveis disponíveis, e investigado o potencial de aplicações móveis como soluções para negócios já existentes. De entre estes tipo de dispositivos é dada especial atenção aos *smartphones*, por se tratarem dos equipamentos mais utilizados, comparativamente a outras soluções como *tablets*, por exemplo. Este capítulo termina com um estudo do OS Android, onde é descrito o ciclo de desenvolvimento de aplicações para este sistema e quais as principais características e componentes destas aplicações.

No capítulo 4 são apresentados os *smart cards* e explicado o porquê de esta tecnologia ser tão utilizada no âmbito de segurança informática. Após a contextualização do estado actual

destes cartões e principais características, são demonstrados os protocolos de comunicação dos mesmos e estudada a plataforma JavaCard.

O capítulo 5 representa a análise, estudo e concepção de uma possível solução para o envio de mensagens com MDDE a partir de um dispositivo móvel, onde é especificada a estrutura do sistema com foco numa implementação o mais genérica possível, de forma a poder ser reaproveitada para possíveis implementações em diferentes OS móveis. São assim apresentados os requisitos funcionais e requisitos dos dispositivos físicos (i.e. dos *smart cards* e dos dispositivos móveis) para o suporte à implementação e posterior utilização deste serviço. Este capítulo é concluído com uma análise de segurança à proposta apresentada, com consideração sobre as diferenças inerentes à diferente plataforma e ambiente de execução.

No capítulo 6 é demonstrado o processo implementação da solução especificada no capítulo anterior. São descritos os problemas decorridos durante a implementação e apresentados os resultados obtidos, com um estudo destes resultados e notas sobre possíveis optimizações.

Por fim, no capítulo 7 são expostas as conclusões sobre os resultados obtidos nesta dissertação e algumas considerações trabalho futuro neste tópico.

CAPÍTULO 2

A Marca Do Dia Electrónica (MDDE)

Neste capítulo, é apresentado o contexto do serviço MDDE, com descrição de como surgiu e com que propósito este foi criado. É também analisado em detalhe o seu funcionamento, bem como feito um estudo dos princípios de segurança que sustentam este serviço, pela análise do sistema de envio e validação de mensagens MDDE.

2.1 Contextualização do serviço

O MDDE é um serviço que permite a utilização do *e-mail* com um alto nível de segurança e confiança. Este serviço foi desenvolvido em conjunto pelos serviços postais CTT e pela empresa de certificação digital MULTICERT, com o propósito de fornecer uma solução digital equivalente à alternativa física de correio postal registado, estando disponível para utilização desde 2003. Como referido no capítulo 1, este serviço é baseado nas especificações de sistemas EPCM, definidas em [43] pela *Universal Postal Union* (UPU)[1]. Entre os cinco serviços postais com soluções deste género a serem utilizadas actualmente (nomeadamente, os serviços postais do Canadá, Estados Unidos, França, Itália e Portugal) [44],o MDDE destaca-se como sendo o primeiro serviço pago e reconhecido legalmente.

O objectivo a que o serviço MDDE se propõe é disponibilizar o envio de mensagens de correio electrónico com as seguintes garantias:

- Prova de envio das mensagens — é entregue ao remetente um comprovativo de envio, que permite a identificação unívoca da mensagem enviada e a confirmação (em qualquer

[1] *Universal Postal Union* — Organização internacional que coordena os serviços postais de cerca de 190 países

altura) de que a mensagem indicada foi enviada por este serviço;

- Integridade do assunto, conteúdo e endereços de *e-mail* (tanto remetente como destinatários) da mensagem — é passível de ser confirmado que o assunto, texto e possíveis anexos da mensagem não foram adulterados, e é garantida a autenticidade do remetente e a identificação dos destinatários da mensagem;

- Não-repúdio da data e hora de envio — é assegurada a veracidade da data e hora de envio da mensagem indicada.

Para cumprir tais objectivos, o serviço MDDE utiliza técnicas de *timestamping* sobre as mensagens enviadas. A associação de *timestamps* às mensagens, como descrito em [29], permite fornecer garantias temporais e de unicidade sobre as mensagens enviadas por e-mail, evitando assim ataques por repetição e intercalação de pedidos. As características dos *timestamps* deste serviço, mais frequentemente denominados de "selos MDDE", são descritas com maior detalhe no sub-capítulo 2.1.2.

2.1.1 A alternativa electrónica ao correio registado

Uma vez que o MDDE pretende servir como um meio de comunicação electrónico com as propriedades do serviço postal de correio registado, para melhor perceber o sistema utilizado pelo serviço MDDE é primeiro revisto o funcionamento do correio registado, representado na figura 2.1. Quando um utilizador pretende enviar uma carta por este sistema, são realizados os seguintes passos:

1. O utilizador entrega a correspondência que pretende enviar num posto de correio, onde tem de assinar o pedido de envio de correio registado;

2. É registado o envio, aplicado um código de barras com um número de identificação único à correspondência, e fornecido ao utilizador um recibo que serve como comprovativo de envio;

3. Dependendo da localização do destinatário e de opções de tratamento prioritário da correspondência, a carta registada é enviada pelo meio adequado;

4. Conforme a modalidade de entrega optada, a correspondência é depositada na caixa postal, entregue em mão na morada de destino (requer uma assinatura de quem recebe) ou entregue pessoalmente mediante a identificação exclusiva do destinatário.

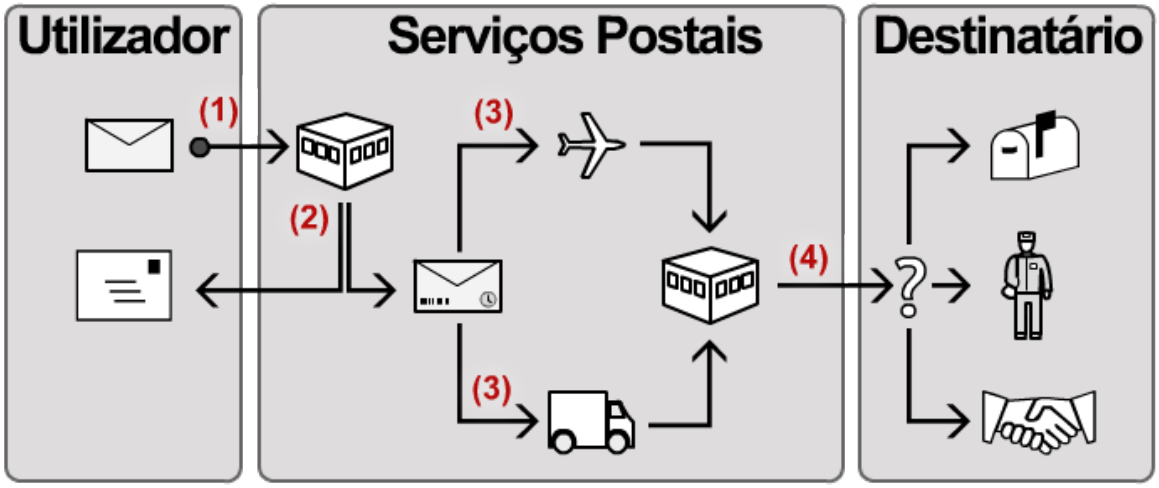

Figura 2.1: Funcionamento geral do correio registado

Comparativamente, o sistema de envio de mensagens MDDE, representado na figura 2.2, apresenta o seguinte fluxo:

1. O utilizador compõe a mensagem de correio electrónico num cliente de *e-mail* à sua escolha;

2. Ao ser enviada, a mensagem é interceptada por um *plugin* específico (sendo este processo descrito em maior pormenor no sub-capítulo 2.2.2) com o objectivo de gerar um pedido de envio MDDE, onde é realizada uma assinatura digital sobre o pedido de envio com as credenciais previamente especificadas pelo utilizador;

3. A mensagem e o pedido correspondente, seguem então para o serviço MDDE, onde é verificada a autenticidade e validado o conteúdo do pedido;

4. Neste processo interno e com o recurso a diferentes entidades (ver o sub-capítulo 2.2.1 para mais detalhes sobre este processo), é gerado um selo MDDE que contém um identificador único da mensagem a enviar, a hora e data do envio e um resumo digital do conteúdo da mensagem, sendo esta informação assinada digitalmente pelo serviço MDDE;

5. A mensagem original e o selo MDDE correspondente são entregues ao servidor de *Simple Mail Transport Protocol* (SMTP) do remetente, e o utilizador recebe um comprovativo do envio do seu *e-mail* com toda a informação que o caracteriza, bem como um número único para a mensagem MDDE, que permite identificá-la de forma unívoca.

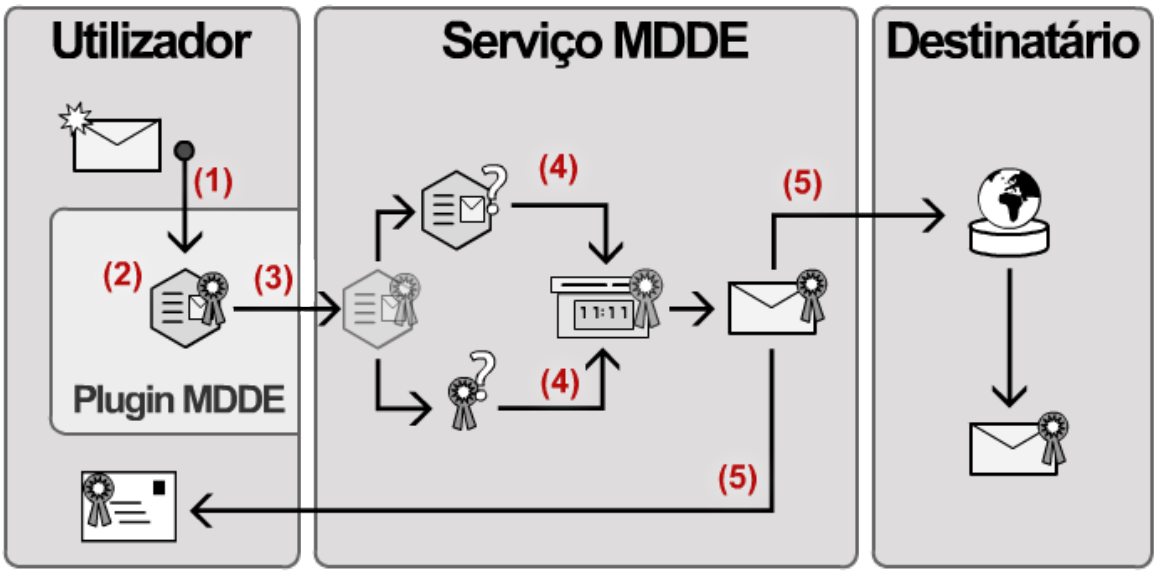

Figura 2.2: Esquema geral do serviço MDDE

Como se pode verificar, face ao serviço postal de correio registado, existem algumas semelhanças entre os dois processos. Mais especificamente, tanto no caso do correio registado como no caso de envio de mensagens MDDE é requirida a identificação remetente com o propósito de garantir a autenticidade do pedido de envio, e em ambos os serviços é disponibilizado ao utilizador um comprovativo de envio, que contém um número de identificação único e informação característica da mensagem enviada. Estas semelhanças facilitam a transição dum serviço para o outro, uma vez que os dados necessários e os recibos de envio são idênticos entre as soluções e os restantes processos internos que suportam os serviços são transparentes ao utilizador.

O MDDE, quando confrontado com a alternativa física, apresenta como vantagem o facto de ser mais barato por mensagem enviada — o preço actual por mensagem MDDE ronda os $0,38$ euros, enquanto que o preço de envio de correio registado oscila entre os $1,62$ e os $5,30$ euros — e ser mais rápido, uma vez que ao invés de um ou mais dias, as mensagens são entregues em apenas alguns segundos, dada a natureza electrónica deste serviço. Também por este motivo, a disponibilidade ininterrupta é um ponto forte do serviço MDDE, quando comparada ao horário habitual das estações de correio (dias úteis das $08:30$ às $18:00$, regra geral).

No entanto, o serviço MDDE também apresenta algumas desvantagens em relação ao correio registado, como o facto de requerer adesão prévia e de a sua utilização não estar disponível para o uso público (o público alvo do serviço é abordado no sub-capítulo 2.1.3). Apesar do custo por mensagem se mostrar ser mais acessível no caso do MDDE, existem também outros encargos acrescidos — como o valor mínimo mensal equivalente a 40 mensagens (i.e. caso envie menos de 40 mensagens, as não enviadas também são contabilizadas para efeitos de pagamento[2]) e a anuidade de 36, 90 euros, à data — o que pode ser um obstáculo à utilização do serviço quando apenas se considerar apenas a troca ocasional de mensagens.

Existem também algumas funcionalidades opcionais no envio de correio registado, como monitorização do estado de envio e prova de entrega das mensagens, não presentes neste serviço. Pelo facto de as transacções electrónicas serem quase instantâneas, a funcionalidade de monitorização não se justifica no caso do MDDE, porém, seria interessante a possibilidade de obter um recibo de entrega como funcionalidade complementar do serviço MDDE. A prova de entrega pode ser alcançada pelo recurso a outras técnicas sobre as mensagens enviadas (e.g. [2, 11, 38]), no entanto, visto que não é um objectivo da solução MDDE fornecer tal funcionalidade e como se trata de um serviço opcional no envio de correio registado, esta não pode ser considerada uma desvantagem significativa mas sim algo que poderia ser útil num serviço deste género.

2.1.2 O selo MDDE

O selo MDDE, que na prática representa o *timestamp* aplicado às mensagens trocadas por este sistema, é o elemento central do serviço MDDE. Este *timestamp* tem como objectivo certificar a veracidade da hora e data de envio da mensagem, para que seja possível confirmar a integridade da mensagem e assegurar o não-repúdio do conteúdo da mesma — ou seja, é o que fornece a prova de que a mensagem não foi alterada desde o seu remetente até o(s) destinatário(s) e de que a mensagem foi enviada na data indicada.

A emissão do selo MDDE é efectuada por uma Terceira Entidade de Confiança (TTP), e a hora e data inscritas são obtidas de uma fonte cuja confiabilidade está assente em bases

[2]Nota: apesar de existir um valor mínimo correspondente a 40 mensagens, as mensagens não utilizadas em cada conjunto de 40 são disponibilizadas no mês seguinte sem qualquer custo, analogamente a um serviço pré-pago com carregamentos obrigatórios.

legais, sendo que as partes envolvidas na construção do selo são estudadas com mais pormenor no sub-capítulo 2.2.1 e a sequência de construção durante o envio das mensagens é descrita no sub-capítulo 2.2.3. O *timestamp* resultante contém informação que caracteriza a transacção, representando a prova de envio de uma mensagem, permitindo que qualquer entidade (i.e. remetente, receptores ou terceiros na posse deste selo) seja capaz de verificar tanto a hora e data de envio como a integridade do assunto, endereço do remetente, endereços dos destinatários, corpo da mensagem e anexos.

Na ausência desta prova de data e hora, não há evidências do envio da mensagem por parte do remetente, caso o destinatário alegue não ter recebido a mensagem ou caso a mesma tenha sido extraviada, pelo que a posse do comprovativo de envio (ou, pelo menos, do número único de identificação da mensagem) é necessário para garantir o não-repúdio do envio e autenticidade do remetente. De igual modo, o selo MDDE presente na mensagem enviada é também necessário para o não-repúdio da mensagem recebida, caso o destinatário pretenda mais tarde provar que o remetente indicado foi responsável pelo envio, e que o conteúdo da mensagem é da autenticidade deste.

2.1.3 Público alvo do serviço

Inicialmente, a plataforma MDDE apenas disponibilizava a sua utilização a advogados inscritos na Ordem dos Advogados que possuíssem um certificado digital válido emitido por esta mesma entidade. Uma das principais aplicações do serviço MDDE neste âmbito é a utilização do serviço para o envio de peças processuais por *e-mail* em processos judiciais, o que enfatiza a confiança depositada neste serviço. Desta forma, o serviço tem sido utilizado pelos advogados para enviar correspondência para os tribunais por mensagens de correio electrónico registadas, em substituição do correio tradicional registado, tal como previsto na lei [7].

Apesar de os advogados serem o principal público do serviço MDDE, este foi concebido com o propósito de poder ser prestado em diferentes contextos. Por este motivo, posteriormente ao lançamento do serviço, o sistema MDDE foi adaptado de modo a permitir a sua utilização por parte de empresas, pela introdução de servidores generalistas no tratamento de pedidos de envio. Neste caso de uso, em alternativa ao processo requerido para a sua uti-

lização por advogados, são pedidos certificados digitais para os colaboradores das empresas, específicos para a finalidade de acesso ao serviço MDDE.

2.2 Detalhes de funcionamento

O funcionamento geral do MDDE e os objectivos a que este se propõe são em tudo semelhantes aos do serviço postal de correio registado. Porém, para cumprir tais propósitos e melhor compreender as operações desempenhadas por este sistema, existem varias componentes e entidades envolvidas nos processos e registo, envio e validação.

2.2.1 Entidades envolvidas

O serviço MDDE, como referido em 2.1, é um serviço prestado pelos CTT e desenvolvido em parceria com a MULTICERT, sendo esta última responsável pela manutenção de toda a infra-estrutura tecnológica que sustenta a componente transaccional do serviço.

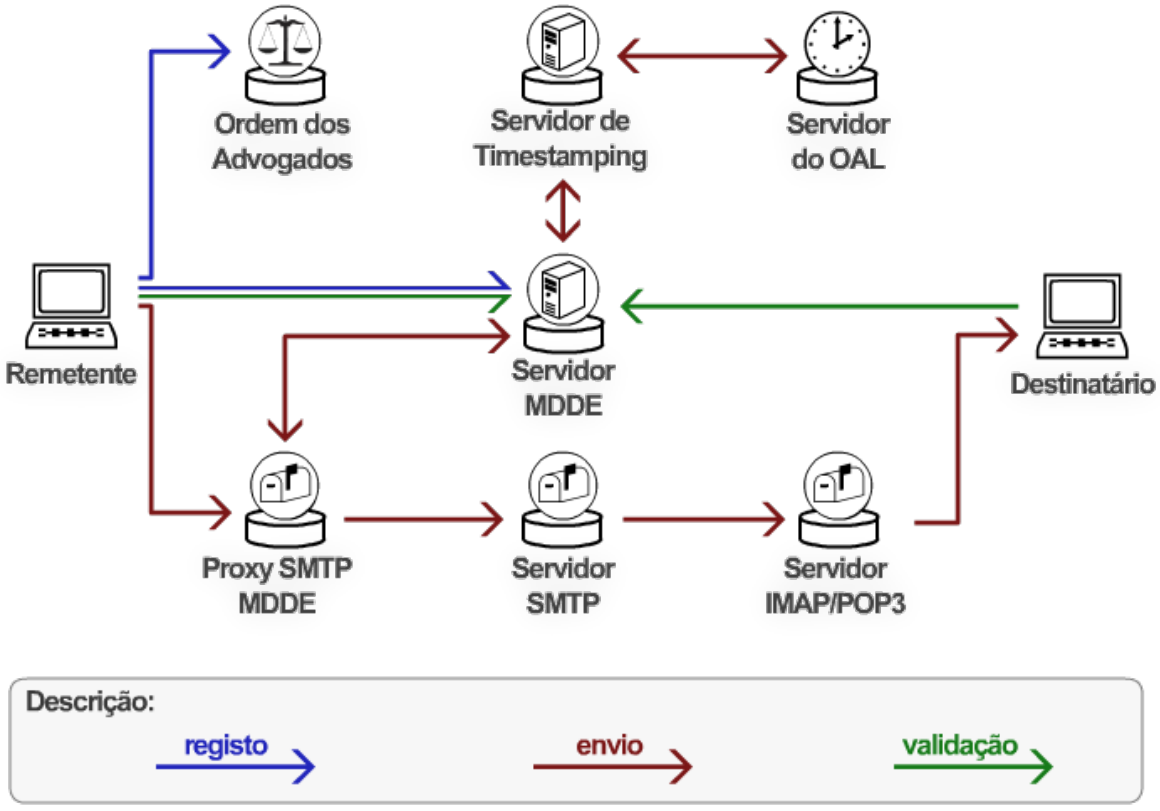

Figura 2.3: Entidades envolvidas no serviço MDDE

A estrutura de suporte do sistema é composta por um servidor MDDE, *proxys* SMTP MDDE, um servidor de *Timestamping* e o servidor SMTP de envio, conforme representado na figura 2.3. Além destes servidores, existem também outras entidades envolvidas no funcionamento do serviço MDDE, mas uma vez que são independentes e/ou não estão directamente relacionadas com o serviço, apenas são representadas para melhor compreensão da estrutura geral do sistema. Entre estas entidades externas, fazem parte o servidor de hora e data do Observatório Astronómico de Lisboa (OAL), o sistema informático da Ordem dos Advogados e os servidores IMAP/POP3 de recepção (dos destinatários). A participação de todas estas entidades nos processos de registo no serviço, envio de mensagens e validação dos selos MDDE é descrita de seguida.

***Proxys* SMTP MDDE** Os *proxys*[3] SMTP desempenham o papel de processamento e despacho dos pedidos MDDE. São o ponto de entrada deste serviço, uma vez que é com estes servidores que o *plugin* comunica quando é enviada uma mensagem MDDE. A comunicação entre o *plugin* e o *proxy* é realizada com o protocolo de transporte de mensagens SMTP, utilizado pelos servidores de envio de *e-mail*, sendo que o pedido de envio é composto por uma mensagem de correio electrónico que obedece ao formato *Secure/Multipurpose Internet Mail Extensions* (S/MIME) (a estrutura deste pedido é revista com maior atenção no subcapítulo 2.2.2). Como o serviço MDDE foi adaptado para diferentes usos, existem actualmente dois servidores *proxy* activos, nomeadamente o servidor utilizado para o envio por parte de advogados e o servidor *proxy* de uso generalista.

Servidor SMTP O servidor SMTP é responsável pelo transporte da mensagem já processada (i.e. o *e-mail* original e o selo MDDE) até ao servidor de correio electrónico do destinatário. Este é, portanto, o último passo do envio de mensagens MDDE no processo interno do serviço.

Servidor MDDE No centro de todos os processos deste sistema, está o servidor MDDE. Durante a adesão de novos utilizadores, este servidor é utilizado para registar os utilizadores no serviço. É também acedido no envio de mensagens, sendo que recebe os pedidos validados de um *proxy* SMTP MDDE e informação característica da mensagem original, de forma a pedir ao servidor de *Timestamping* a produção de um *timestamp* com o propósito de compor o selo MDDE para a mensagem. Por fim, este servidor também é utilizado na validação de

[3] *Proxy* — servidor que actua como intermediário na comunicação de um cliente com um serviço

mensagens, onde é consultado para confirmação da autenticidade do selo MDDE e avaliação da validade dos certificados digitais utilizados (que podem ter sido revogados ou ter expirado, por exemplo).

Servidor de *Timestamping* O servidor de *Timestamping* actua como intermediário na produção de selos MDDE. Para tal, esta Autoridade de *Timestamping* (TSA) recebe do servidor MDDE a informação necessária sobre a mensagem a enviar para a posterior associação do *timestamp*, cuja data e hora inscritas são obtidas do OAL pelo protocolo de sincronização *Network Time Protocol* (NTP).

Servidor de hora e data do OAL Para a obtenção da data e hora necessárias para composição de selos MDDE durante o processo de *timestamping*, é utilizado um servidor do Observatório Astronómico de Lisboa. O acesso ao OAL é justificado pelo facto de este observatório ser, desde 1911, a entidade oficialmente responsável pela determinação da hora legal portuguesa.

Sistema informático da Ordem dos Advogados No caso de uso mais frequente, isto é, a utilização do serviço por entidades judiciais, é necessário a posse de um certificado digital válido emitido pela Ordem dos Advogados. Deste modo, o sistema informático da Ordem actua como Autoridade de Certificação (CA) responsável pelo reconhecimento de advogados inscritos na mesma, para que estes possam efectuar o registo no serviço MDDE.

Servidores IMAP/POP3 de destino Os servidores *Internet Message Access Protocol* (IMAP) ou *Post Office Protocol* (POP3), conforme o protocolo utilizado no acesso aos serviço de correio electrónico do(s) destinatário(s), desempenham o passo final na transacção de envio, sendo responsáveis pela entrega da mensagem ao(s) receptor(es).

2.2.2 O *plugin* MDDE

O *plugin Desktop* MDDE é a componente de software, executada no computador do utilizador, responsável pela intercepção, processamento e envio de mensagens por este sistema. Este *plugin* está actualmente apenas disponível para o OS Windows, podendo ser executado na versão de Windows 98 ou superior. Para interceptar as mensagens de correio electrónico a enviar pelo serviço MDDE, o *plugin* executa e monitoriza um servidor local SMTP, para o qual as aplicações de envio de *e-mail* devem estar configuradas, caso pretendam utilizar este

Figura 2.4: O *plugin desktop* MDDE

serviço (i.e. servidor de envio com o endereço 127.0.0.1 e porta de destino definida para 25). Quando é recebido um *e-mail* no servidor SMTP do *plugin*, é apresentada uma mensagem de confirmação de envio, na qual o utilizador pode optar entre enviar pelo serviço MDDE ou enviar como mensagem de correio electrónico comum, como demonstrado na figura 2.4.

Em relação às configurações do *plugin* MDDE, este permite activar/desactivar o serviço (i.e. controlar a execução o servidor SMTP local), especificar qual o endereço do servidor SMTP (para o caso de envio de *e-mail* simples) e do servidor MDDE (para envio de mensagens pelo serviço, conforme se trate do servidor generalista ou o servidor específico para uso por advogados), definir o certificado digital que se pretende utilizar e ainda configurar o funcionamento de envio pela especificação individual por contacto de destinatário. Esta última configuração é efectuada através de uma lista de endereços (actualizada por cada envio de mensagem bem sucedido, podendo ser também gerida manualmente pelo utilizador), onde é explicitado o modo de operação para cada um desses destinatários: enviar sempre o correio electrónico com MDDE, nunca enviar com o serviço MDDE ou perguntar se é pretendido o envio com ou sem MDDE (opção por defeito).

Além do propósito de intercepção de *e-mails*, este *plugin* é também responsável pela com-

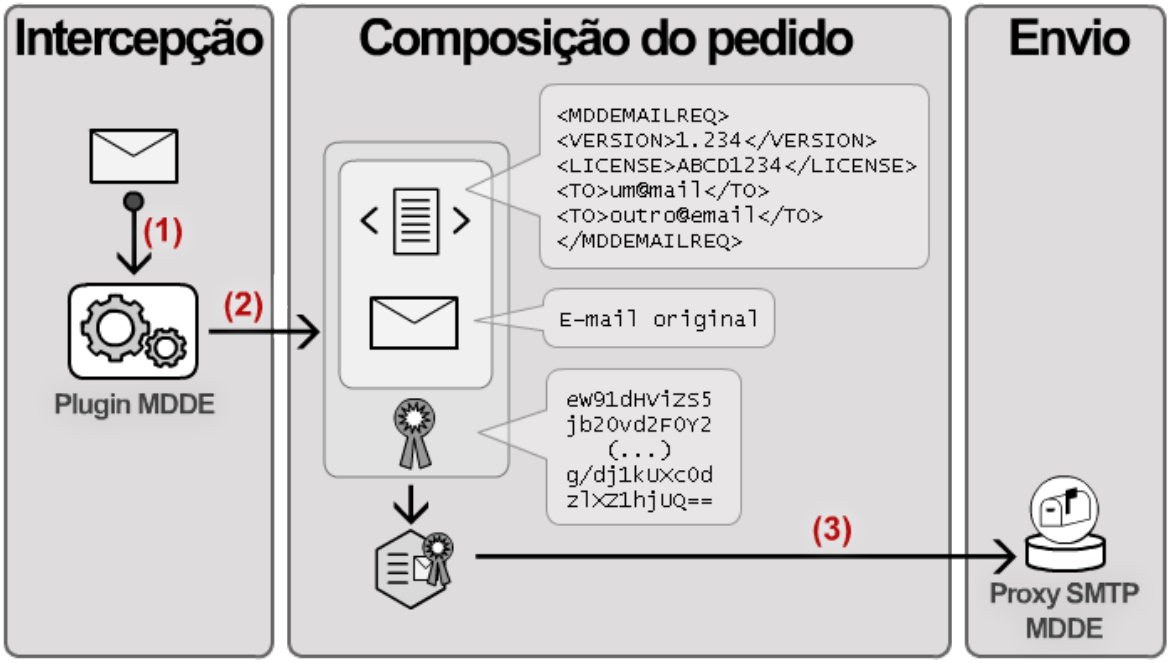

Figura 2.5: Esquema de funcionamento do *plugin* MDDE

posição do pedido MDDE e consequente envio da mensagem para o serviço. O pedido de envio MDDE consiste numa mensagem de correio electrónico no formato S/MIME, cujo corpo da mensagem contém uma estrutura em *Extensible Markup Language* (XML) com informações sobre os destinatários e sobre o *plugin* utilizado, e como anexo possui a mensagem de *e-mail* original, como representado na figura 2.5. O formato S/MIME utilizado é o padrão para esquemas que utilizam técnicas criptográficas de chave pública sobre dados no formato *Multipurpose Internet Mail Extensions* (MIME) [35], sendo que neste caso específico é utilizado para garantir a autenticidade do pedido MDDE, pela aplicação de uma assinatura digital ao mesmo.

A estrutura XML do pedido de envio, contém as seguintes marcações ("*tags*"):

- **MDDEMAILREQ** — delimita o pedido MDDE, contendo as restantes marcações (ou seja, um pedido de envio MDDE inicia com "<MDDEMAILREQ>" e termina com "</MDDEMAILREQ>");
- **VERSION** — representa a versão do *plugin* utilizado para compor o pedido;
- **LICENSE** — contém um valor hexadecimal gerado com o objectivo de identificar univocamente o pedido efectuado, de modo a evitar envios acidentalmente repetidos e

ataques por repetição;

- **TO** — indica o endereço de correio electrónico de um destinatário, sendo que para vários destinatários existem vários registos com marcações "TO". Estes endereços são obtidos da mensagem original[4] e devem estar em conformidade para que o pedido de envio seja correctamente processado.

Para efectuar a assinatura digital sobre o pedido, é utilizado o certificado digital $X.509$[5] (e correspondente chave privada) do cliente. Esta assinatura digital é então estruturada no formato *Cryptographic Message Syntax* (CMS) e codificada em *Base64*, de modo a poder ser adicionada à mensagem do pedido MDDE com o tipo "*pkcs7-signature*", conforme especificado em [16] para a composição de mensagens S/MIME. Uma vez realizados todos os passos necessários para a produção de um pedido de envio — interceptar o correio electrónico, compor a estrutura XML do corpo da mensagem do pedido, anexar o *e-mail* original e assinar esta mensagem — conforme demonstrado na figura 2.5, o *plugin* procede ao envio deste pedido para o *Proxy* SMTP especificado nas configurações, que dará continuação ao envio da mensagem MDDE.

2.2.3 Registo, envio, e validação de mensagens

O ciclo do serviço MDDE consiste em três processos distintos: o registo de novos utilizadores, o envio de mensagens por este serviço e a validação de mensagens MDDE. Para o registo no sistema, um utilizador precisa de possuir um certificado digital válido perante uma certa CA conforme o caso específico de uso, como está descrito no sub-capítulo 2.1.3. Em relação ao envio de mensagens pelo sistema, é requerido o registo prévio a instalação do *plugin* fornecido para esse propósito. Já a validação de mensagens, não requer qualquer registo ou identificação, pelo que qualquer pessoa na posse de uma mensagem MDDE pode proceder à verificação da autenticidade da mesma. De seguida, são descritos estes três processos em maior detalhe.

[4]Nota: quando existem destinatários Com Conhecimento (CC) ou Cópia Oculta (BCC), estes são de igual forma adicionados ao pedido com a marcação "TO"; no entanto, a mensagem MDDE resultante é posteriormente enviada com consideração sobre o modo de entrega especificado na mensagem original.

[5]$X.509$ — padrão definido pelo Sector de Uniformização das Telecomunicações da *International Telecommunication Union* (ITU-T) e utilizado em Infraestruturas de Chaves Públicas (PKI)

Registo de utilizadores

A adesão ao serviço MDDE é efectuada pelo registo *online* numa página disponibilizada para este efeito[6]. Este registo é concretizado através do preenchimento de um formulário, onde são especificados os dados pessoais do utilizador e condições de facturação. Em relação à submissão do registo, o formulário é assinado digitalmente (com as credenciais fornecidas pela CA correspondente para o acesso ao serviço), de modo a garantir a veracidade dos dados indicados e a aceitação dos termos de utilização do serviço. Para efectuar a assinatura digital, é requerida a instalação de um componente ActiveX, apenas disponível para o navegador Internet Explorer.

Após ter sido preenchido e assinado digitalmente o formulário, este é enviado para o servidor MDDE, onde é processado e efectuado o registo do utilizador. Uma vez confirmado este registo e efectuada a primeira facturação do serviço, o utilizador pode então instalar o *plugin* MDDE e enviar correio electrónico por este serviço, assim que tenha configurado a sua aplicação cliente de *e-mail*. Caso seja necessário actualizar os dados pessoais ou cancelar a subscrição do serviço, isto pode também ser realizado *online* na página *Web* do serviço[7].

Envio de mensagens

Dos três processos do serviço MDDE, o envio de mensagens representa a transacção de maior complexidade, sendo que envolve todas as entidades de que este serviço depende. Como representado na figura 2.6, durante o envio de mensagens são reproduzidos os seguintes passos:

1. O utilizador cria uma nova mensagem de correio electrónico, que é interceptada pelo *plugin* MDDE durante o seu envio;

2. Após gerado (e assinado digitalmente) o pedido de envio, este é enviado para o *Proxy* SMTP MDDE;

3. Uma vez verificada a estrutura do pedido e confirmada a autenticidade da assinatura digital, é redireccionado o pedido de envio para o Servidor MDDE;

[6] https://sce.ctt.pt/registration/new.html
[7] https://sce.ctt.pt/mdde/index.html

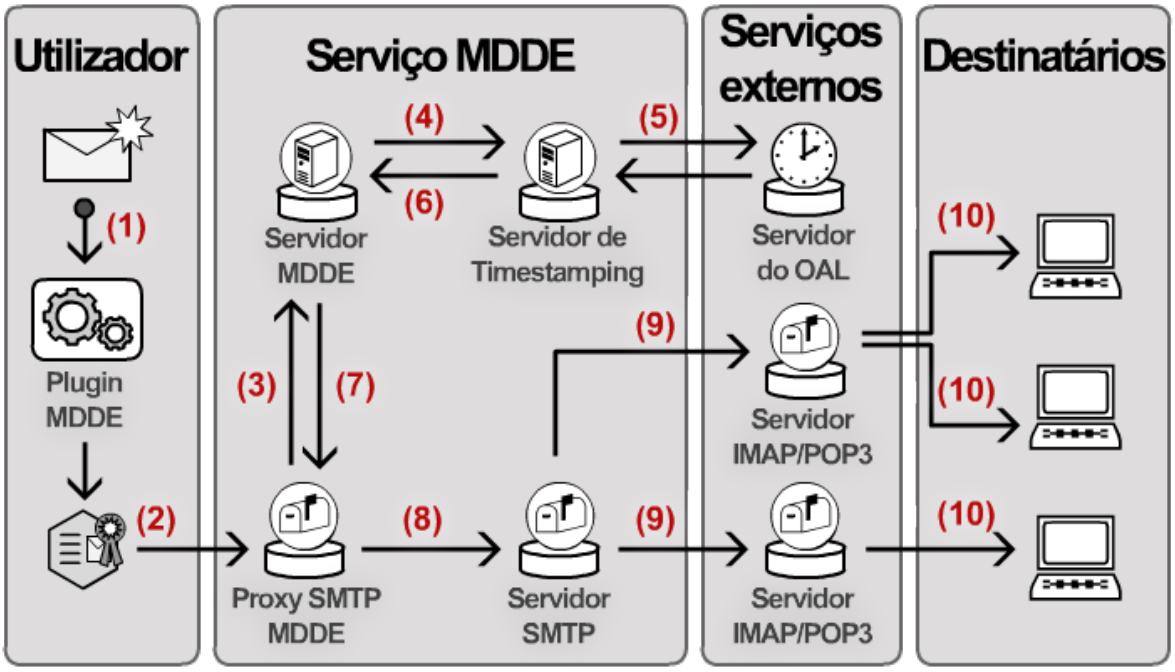

Figura 2.6: Diagrama de envio de mensagens MDDE

4. O Servidor MDDE requer ao Servidor de *Timestamping* que seja efectuado um *timestamp* sobre a mensagem;

5. A data e hora legal portuguesa é obtida do Servidor do OAL e é gerado um *timestamp*;

6. É devolvido ao Servidor MDDE o *timestamp* produzido, assinado digitalmente pelo Servidor de *Timestamping*;

7. O Servidor MDDE cria o selo MDDE, assina-o digitalmente e envia o mesmo para o *Proxy* SMTP MDDE;

8. Com o selo MDDE recebido e a mensagem original, o *Proxy* SMTP MDDE cria um novo *e-mail* MDDE, assina-o digitalmente e entrega-o ao Servidor SMTP;

9. O Servidor SMTP envia a correspondência para os servidores de *e-mail* do(s) destinatário(s);

10. Por fim, o(s) destinatário(s) recebe(m) o *e-mail* MDDE.

Validação de mensagens

Tal como o registo no serviço, a validação de mensagens MDDE é feita numa página *Web* específica[8] com recurso a um componente ActiveX, estando também apenas acessível para utilização no navegador Internet Explorer. Este processo permite verificar a autenticidade e integridade de uma mensagem enviada com MDDE sem que nenhuma informação sobre o conteúdo da mensagem seja transmitida ao longo do processo[9], uma vez que apenas o resumo digital da mensagem original ("*hash*") e o selo MDDE são enviados para o servidor onde é realizada a validação.

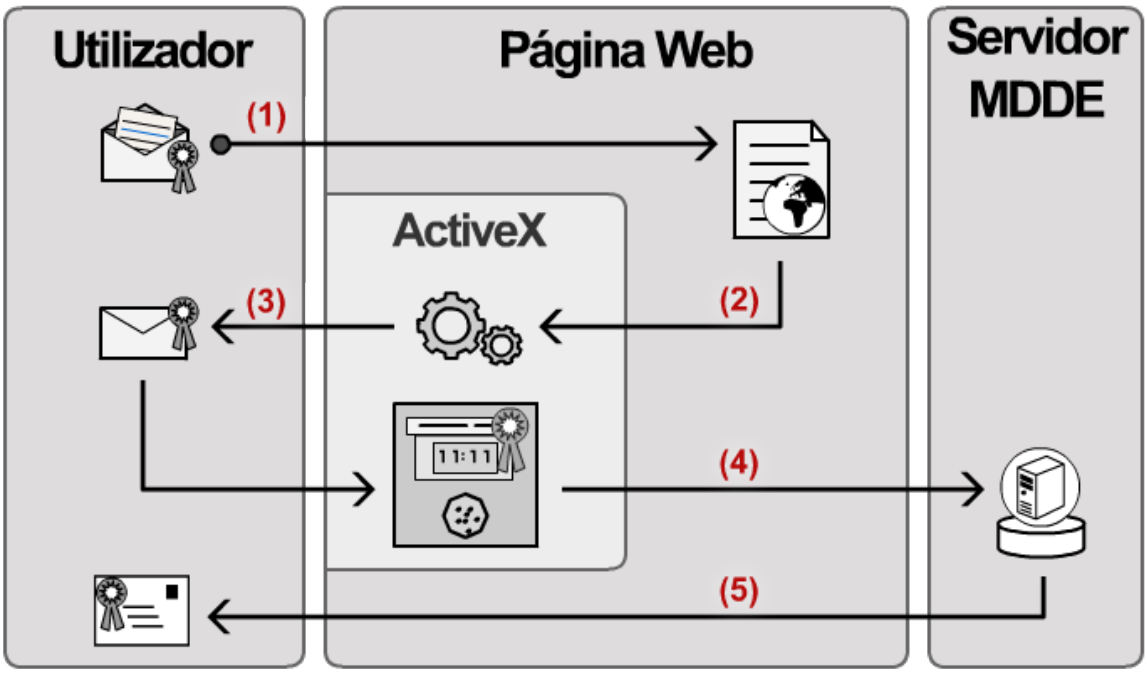

Figura 2.7: Diagrama de validação de mensagens MDDE

Quem tiver na sua posse um *e-mail* com MDDE, seja um destinatário, o remetente ou uma terceira parte que pretenda confirmar uma transacção, pode efectuar a sua validação da seguinte forma (como descrito na figura 2.7):

1. Aceder à página *Web* indicada no corpo da mensagem com MDDE;
2. Permitir a execução do componente ActiveX de validação de mensagens MDDE;
3. Seleccionar a mensagem que se pretende validar, previamente guardada em disco;

[8] https://sce.ctt.pt/registration/validate.html

[9] Para mais detalhes sobre esta propriedade, consultar o sub-capítulo 2.3.1.

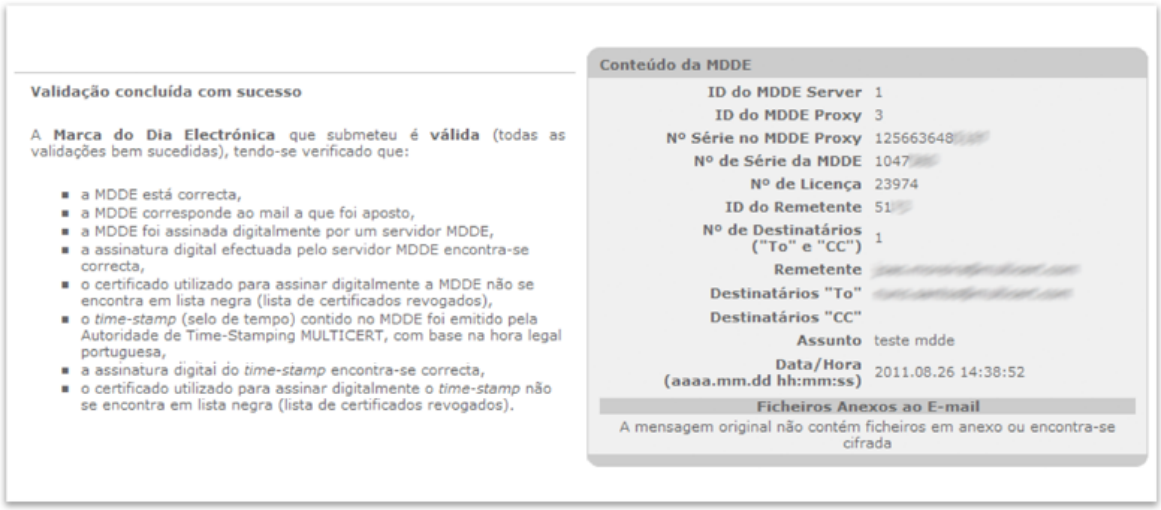

Figura 2.8: Exemplo de um resultado de validação

4. O componente ActiveX calcula um resumo digital da mensagem original e envia-o para o Servidor MDDE, juntamente com o selo MDDE da mensagem;

5. O resultado da validação é apresentado ao utilizador, como exemplificado na figura 2.8.

2.3 Análise de segurança

Quando se considera a utilidade e funcionalidade de um serviço como o MDDE, uma característica fundamental é a confiança incutida nos seus utilizadores. O que leva as pessoas a confiarem no correio registado é o facto de este serviço estar provado como sendo seguro e eficiente. No entanto, será que se pode dizer o mesmo do MDDE, e até que ponto este cumpre os objectivos propostos?

As garantias de segurança apresentadas pelo serviço MDDE, são as seguintes:

- Identificação do remetente;
- Autenticidade da data e hora de envio da mensagem;
- Protecção contra a adulteração — e consequentemente, garantia de integridade — do assunto, destinatário(s) e conteúdo da mensagem (incluindo anexos);
- Identificação unívoca da mensagem;

- Comprovativo de envio que garante o não repúdio das propriedades acima referidas;
- Garantia de que os CTT não têm acesso ao conteúdo da mensagem durante tanto o envio como a validação desta.

Relativamente a características indicadas como não presentes no serviço, é afirmado que o MDDE não garante a entrega e recepção da mensagem pelo destinatário, uma vez que isto está dependente de outros sistemas envolvidos (tais como os servidores de correio electrónico do destinatário). É também referido que não existe qualquer garantia de privacidade sobre o conteúdo das mensagens perante terceiros (que não os CTT), pois a mensagem é transmitida através da Internet, por entidades externas ao serviço, até ao seu destino. Caso a privacidade seja um factor imponente, é aconselhado pelos CTT a que a mensagem seja cifrada pelo seu remetente.

De modo a ser capaz de compreender as especificações de segurança de um sistema, é necessário ter um conhecimento geral dos conceitos envolvidos. De seguida, é apresentada uma breve descrição dos termos necessários para uma análise sobre se e como o serviço MDDE satisfaz os objectivos propostos.

Funções de *hash* Técnicas simples e eficientes que complementam outras técnicas criptográficas mais complexas. O propósito de uma função de *hash* é o de gerar as "impressões digitais" (ou resumo digital) de um documento electrónico (ou de qualquer outro fragmento de informação digital) [45]. Estas são funções determinísticas, habitualmente com valores de resultado que seguem um distribuição aparentemente uniforme. Com uma função criptográfica de *hash* ideal, é inviável encontrar a mensagem que originou um dado resumo digital (isto é, nenhuma informação sobre os valores de entrada da função pode ser obtida a partir dos valores de saída/resultado da mesma), ou até mesmo encontrar duas mensagens que originariam o mesmo resumo digital (i.e. é difícil de encontrar colisões). Desta forma, usando funções de *hash* pode-se obter um resumo digital relativamente curto (e de tamanho fixo) de uma mensagem arbitrária, sendo também impraticável modificar a mensagem original sem que o *hash* resultante seja alterado.

Assinaturas digitais Uma assinatura digital é um conjunto de dados em formato electrónico que demonstra a autenticidade de um documento digital, quando anexado ou logicamente associado a esse documento (de modo análogo assinatura física, em papel). As principais pro-

priedades de uma assinatura digital são a autenticidade, a integridade e o não-repúdio. Um documento assinado digitalmente é inalterável, ou seja, após ser efectuada a assinatura do documento este não pode sofrer alterações sem que seja invalidada a sua assinatura. Por este motivo, uma assinatura digital não é reutilizável e não é possível verificar apenas parte de um documento assinado digitalmente, pois a assinatura é válida unicamente para o documento completo. Devido às suas características de não-repúdio, o signatário não pode reclamar mais tarde que não assinou o documento (a menos que as suas credenciais tenham sido comprometidas).

Timestamps Como referido no sub-capítulo 2.1, um *timestamp* é uma representação digital da data e a hora de ocorrência de um determinado evento. Com o uso de técnicas de *Trusted Timestamping* (TTS) é possível atribuir criptograficamente a data e hora a um documento digital, com garantias de que essa data e hora é de confiança, sendo o *timestamp* emitido por uma Autoridade de *Timestamping* (TSA). O papel da TSA é actuar como uma TTP, responsável por criar *timestamps* para indicar que uma certa informação existia antes ou durante um determinado período de tempo [1]. Isto implica que a TSA utilize uma fonte confiável para a obtenção de data e hora, de modo a garantir a correcção e validade do *timestamp*.

Comunicação por HTTPS O *HyperText Transfer Protocol Secure* (HTTPS) é um protocolo de comunicação que resulta da combinação do *HyperText Transfer Protocol* (HTTP) com os protocolos SSL/TLS, de modo a permitir a transmissão de dados através de uma conexão cifrada. Assim sendo, o principal propósito do HTTPS é o de disponibilizar um canal de comunicação seguro, estabelecido no topo de uma ligação insegura (Internet). Além da privacidade obtida pela conexão cifrada, o HTTPS garante também a autenticação de pelo menos uma das partes envolvidas (o servidor) pelo uso de certificados digitais. Estas características que o protocolo apresenta, evitam que a comunicação seja observada por terceiros (ataques por *eavesdropping*) ou até interceptada e alterada (ataques de *man-in-the-middle*), justificando a sua vasta adopção por parte de serviços como *Online Banking* e compras pela Internet.

2.3.1 Análise ao protocolo MDDE

Embora o serviço MDDE seja baseado em padrões internacionalmente reconhecidos e adoptados (tais como as normas descritas em [1, 10, 16, 32, 33, 35, 43]), não se pode assumir

que a sua implementação está isenta de falhas, uma vez que a utilização de criptografia e outros conceitos de segurança podem até agravar ou introduzir problemas de segurança quando não são correctamente implementados. Tendo isto em conta, é apresentada uma análise sobre as especificações funcionais do serviço MDDE, para os processos de registo, envio e validação (descritos no sub-capítulo 2.2.3).

Registo no sistema

Durante a adesão ao serviço MDDE, o acesso à página *Web* onde o registo é efectuado é concretizado através de uma ligação HTTPS. Além da autenticação do Servidor MDDE, também o utilizador é identificado e autenticado durante a comunicação, uma vez que é requerido que este tenha instalado no navegador de Internet o certificado disponibilizado para efeitos de uso do serviço. Desta forma, é garantida a autenticidade e confidencialidade da informação trocada durante o registo no sistema, sendo por isso improvável que qualquer dado pessoal do utilizador seja comprometido durante este processo.

Envio de mensagens MDDE

Para o envio de mensagens, o *plugin* MDDE é responsável por gerar a informação característica do pedido de envio, que será posteriormente assinado digitalmente, com o certificado digital do utilizador e respectiva chave privada. Uma vez que é utilizado um módulo de segurança externo (disponibilizado pelo OS), a gestão de credenciais não está dependente do próprio *plugin* e podem ser utilizadas diferentes implementações para a manutenção e utilização das credenciais, em hardware ou software, sem interferência do *plugin*, pelo que não podem ser analisadas as propriedades deste módulo de segurança. Porém, considerando uma implementação correcta, uma vez que o pedido de envio e a mensagem original são assinados digitalmente no mesmo documento electrónico (no formato S/MIME), existe a confirmação de que o pedido é autêntico e corresponde à mensagem anexada.

Relativamente ao conteúdo do pedido de envio, no campo de licença (da estrutura XML) é gerado um número único com o intuito de precaver o uso indevido pela repetição de envios, e uma vez que o pedido é assinado digitalmente, qualquer alteração no código de licença resultaria na invalidação da assinatura do pedido. Desta forma, na eventualidade de serem efectuados dois pedidos com a mesma licença, o *Proxy* SMTP MDDE não procede ao envio, por se tratar de um pedido repetido. Também nos dados do pedido são listados os endereços

dos destinatários, o que pode ser considerado redundante uma vez que esta informação se encontra nos cabeçalhos da mensagem original. No entanto, esta opção pode ter sido motivada com o intuito de posteriormente facilitar a utilização do serviço para envio de documentos digitais que não apenas *e-mails* (apesar de que qualquer informação digital pode ser anexada a um documento de correio electrónico), ou até para o envio das mensagens MDDE para outros destinos como endereços de Fax, por exemplo.

Após o envio do pedido para o *Proxy* SMTP MDDE, apenas a *hash* da mensagem é transmitida ao Servidor MDDE, de forma a que nenhuma informação sobre o conteúdo da mensagem circule desnecessariamente. Durante este processo, são utilizados certificados digitais para estabelecer ligações seguras, de modo a assegurar a autenticidade e integridade da informação trocada na Infraestrutura de Chaves Públicas (PKI) do serviço. A hora e data inscritas no *timestamp* (aplicado ao resumo digital da mensagem original) são obtidas do OAL e verificadas em comparação com outras entidades internacionais (também responsáveis pela hora legal dos países correspondentes) para garantir uma tolerância máxima de 2 segundos, sendo que este processo de TTS segue o procedimento descrito no padrão definido em [1].

Validação de mensagens

Tal como processo de registo no sistema, também a validação de mensagens MDDE é feita através de uma ligação HTTPS. No entanto, a validação de mensagens não requer autenticação por parte do cliente (pois qualquer utilizador pode validar uma mensagem, seja o remetente, destinatário ou terceiros), sendo que neste caso apenas a identidade do Servidor MDDE é autenticada.

Em relação à informação transmitida, apenas é enviado para o servidor de validação o selo MDDE e um resumo digital da mensagem original calculado localmente pelo componente ActiveX, o que garante que nenhuma informação sobre o conteúdo específico desta (à parte dos destinatários, rementente, assunto, hora e data inscritos no selo MDDE) é enviada. A integridade da mensagem original é verificada pela comparação do resumo digital calculado com a *hash* presente no selo MDDE, cuja autenticidade é garantida por este ser digitalmente assinado pelo Servidor MDDE. Uma vez que neste selo estão contidos os endereços de remente e destinatários, é assegurado o não-repúdio dos mesmos caso a autenticidade do selo seja confirmada.

Durante este processo são também validados os certificados digitais utilizados, de modo a garantir que na eventualidade de ocorrer um envio fraudulento com certificados revogados este é identificado. Este é o principal motivo para a verificação ser concretizada *online*, uma vez que a maioria das restantes validações poderiam ser efectuadas *offline*. Assim sendo, caso um certificado digital tenha sido comprometido (e.g. o certificado do utilizador ou do *Proxy* MDDE) este é registado numa Lista de Revogação de Certificados (CRL) no Servidor MDDE para indicar que o certificado já não é de confiança e, consequentemente, torná-lo inválido.

2.3.2 Riscos e possíveis ataques ao serviço

Uma das preocupações de um utilizador de serviços como o MDDE é a privacidade da informação transmitida. No que toca a este assunto, é aconselhado pelos CTT que caso um utilizador pretenda assegurar a privacidade durante o envio da mensagem, este deve cifrar a mesma. No entanto, a grande maioria de utilizadores destas tecnologias não tem o conhecimento nem as ferramentas necessárias para tal efeito, pelo que esta questão pode ser vista como uma grande entrave à utilização do serviço, principalmente quando se considera o envio de informação confidencial para tribunal com MDDE, por exemplo. Como não é usada protecção dos canais durante o envio há o risco da comunicação poder ser observada, apesar de não haver hipótese de adulteração dos dados, uma vez que toda a informação transmitida é assinada digitalmente.

Quando considerado o risco de envios intencionalmente ou acidentalmente repetidos (e.g. ataques por repetição ou problemas de rede que levam ao reenvio de pacotes, respectivamente), estas ocorrências podem resultar em gastos elevados, uma vez que o MDDE se trata de um serviço pago. Para este serviço, não existe a possibilidade de isto acontecer visto que cada pedido contém um código de licença único, verificado por parte do *Proxy* SMTP MDDE durante o envio. Relativamente ao uso indevido do *plugin*, este tem uma protecção para evitar o envio de mensagens não solicitadas ("*spam*"), sendo que após 50 mensagens enviadas com MDDE durante um dia e sem confirmação do utilizador (i.e. com a configuração de enviar sempre com MDDE para os destinatários especificados no *plugin*) é apresentada uma mensagem de alerta a indicar tal facto e a pedir a confirmação do utilizador para prosseguir com o envio. No entanto, como este aviso apenas é apresentado ao fim de 50 envios bem sucedidos, isto pode resultar em custos bastante elevados para o utilizador, pelo que um limite

inferior (por exemplo, 10 mensagens) talvez fosse melhor prática para evitar gastos acrescidos e também a possível proliferação de vírus informáticos por este sistema.

Em todos os sistemas informáticos — mesmo que estes tenham sido desenvolvidos com altas especificações de segurança e implementados devidamente — a segurança da informação depende sempre, ultimamente, dos seus utilizadores. Esta dependência de factores humanos pode produzir problemas de segurança graves, particularmente quando se considera a utilização destes sistemas por utilizadores sem experiência suficiente com novas tecnologias.

Um exemplo de uma situação onde uma possível vulnerabilidade com base em falhas humanas podem ocorrer, é quando os utilizadores ignoram certos parâmetros de segurança, tais como a validação de assinaturas digitais em *e-mails* recebidos. Mesmo que a maioria dos actuais clientes de correio electrónico apresentem indicações gráficas (e.g. um cadeado ou um "visto" verde) e informação sobre a autenticidade dos *e-mails* (revelando se e por quem foram assinados), por vezes, os utilizadores simplesmente esquecem-se de verificar o estado da assinatura ou até ignoram os avisos sobre a autenticidade da mesma. Além disso, há ainda casos mais graves em que a assinatura não é sequer verificada, principalmente quando o *e-mail* é consultado através de um navegador de Internet, em que as assinaturas digitais são apresentadas como um ficheiro em anexo e em que, caso o utilizador deseje confirmar a autenticidade da mensagem, deve fazer a validação manualmente.

Esta situação poderia ser explorada através da criação de um *e-mail* falacioso, idêntico a uma mensagem MDDE legítima, com dois anexos: um documento falso (correspondente ao "*e-mail* original") com a mensagem se pretenderia transmitir, e um selo MDDE falso. Nesta mensagem fraudulenta, o *Uniform Resource Locator* (URL) para a validação do selo MDDE seria alterado de modo a iludir os utilizadores e persuadi-los a verificar o selo MDDE numa página falsa com um endereço similar, e.g. substituindo o URL $http://sce.ctt.pt/$ por $http://sce-ctt.pt/$ e efectuando o registo desse nome de domínio apenas para fins maliciosos (uma técnica conhecida como "URL spoofing"). Desta forma, o processo de validação poderia ser reproduzido para iludir um utilizador e levá-lo a acreditar que a mensagem MDDE que recebeu era válida. Outras técnicas mais complexas, tais como o envenenamento de cache do Sistema de Nomes de Domínios (DNS) (também conhecido como "*Pharming*"), podem ser usadas com as mesmas intenções, aproveitando-se da distracção ou falta de conhecimento de

um utilizador comum, no entanto nenhum destes problemas está directamente relacionado com as especificações do MDDE, como mencionado.

CAPÍTULO 3

Dispositivos móveis

Os dispositivos móveis estão presentes no dia-a-dia de praticamente toda a gente, por vários motivos específicos a cada tipo de dispositivo, mas todos com uma vantagem comum: a ubiquidade obtida pelo fácil transporte destes dispositivos, o que permite o acesso instantâneo e em qualquer lugar a certas funcionalidades (sejam estas de conectividade, entretenimento ou actividades profissionais) que antes apenas estavam disponíveis em dispositivos fixos. Este capítulo descreve a motivação e o uso actual dos dispositivos móveis, com uma comparação entre os tipos de dispositivos móveis que existem e respectivos Sistema Operativo (OS), sendo estudado com maior detalhe o caso dos *smartphones* e o impacto que estes têm na rotina diária das pessoas. Também neste capítulo, é feita uma introdução ao desenvolvimento de aplicações para o OS Android e explicado o ciclo de vida destas, concluindo com uma breve análise de segurança a este OS.

3.1 Contextualização

O mercado dos dispositivos móveis é provavelmente o mercado tecnológico com maior expansão actualmente, onde a inovação e a actualização de componentes e funcionalidades são uma constante. Estes dispositivos cada vez mais se tornam aparelhos integrados, com as mais diversas funcionalidades que necessitariam de vários dispositivos diferentes a serem desempenhadas num só, tais como Sistema de Posicionamento Global (GPS), acesso à Internet, televisão portátil, processadores de texto e leitores de multimédia — tudo isto num aparelho que muitas vezes cabe na palma da mão.

Um dos factores responsáveis pelo aumento de utilização deste tipo de dispositivos, foi

a adopção da Internet como meio suplementar (ou até, por vezes, principal) de comunicação e entretenimento. As propriedades de conectividade presentes nos dispositivos móveis permitem que um utilizador seja capaz de aceder a qualquer conteúdo na Internet, independentemente do local em que se encontre e sem a necessidade de uma ligação física à rede. Estas características reflectem-se na utilização de dispositivos móveis para os mais diferentes propósitos, que vão desde o uso em ambiente escolar/educacional ao uso empresarial e militar, sendo uma vantagem destes dispositivos a fácil adaptação e configuração para diversas funcionalidades.

A utilidade destes dispositivos móveis, contrariamente a muitos outros aparelhos electrónicos, não está dependente de uma configuração inicial, limitada a funcionalidades específicas definidas e implementadas por quem os desenvolve. Pelo contrário, habitualmente estas funcionalidades são acrescentadas após a compra dos dispositivos (desde que estes sejam capaz de as desempenhar com o *hardware* que possuem) e em função da opção do utilizador, determinadas pelo uso que este pretende dar ao aparelho e geridas pelo mesmo, o que torna a utilização dos dispositivos móveis num processo dinâmico. Esta característica é, na maioria dos casos, alcançada pela disponibilização de aplicações desenvolvidas por terceiros, que podem ser instaladas, configuradas, utilizadas e removidas do dispositivo.

Existem inúmeras aplicações móveis criadas para facilitar acções diárias, que se servem da conectividade à Internet para realizar certas operações, como por exemplo: comprar bilhetes de metro, realizar *check-ins* de viagens aéreas, auxiliar a procura de bombas de gasolina mais baratas na proximidade, consultar as últimas novidades e a bolsa de valores, actualizar o estado de redes sociais/*microblogging* e consultar a conta bancária ou realizar transferências bancárias. Há também outras aplicações para objectivos diferentes (e.g. entretenimento), mas todas partilham as mesmas vantagens: a acessibilidade instantânea e a comodidade de utilização.

Apesar da utilidade dos dispositivos móveis e das aplicações existentes para os mesmos, as dimensões limitadas destes aparelhos têm consequências negativas tanto para quem os produz como para quem desenvolve aplicações para os mesmos. Devido às dimensões reduzidas, a maioria dos dispositivos móveis incluem pouca memória (tanto persistente como volátil), possuem CPUs lentos e apresentam ecrãs pequenos — sendo geralmente ecrãs sensíveis ao toque,

ou "*touchscreens*", de modo a poupar o espaço e peso que seriam acrescidos pela presença de um teclado físico. Do ponto de vista de quem produz estes aparelhos, existe o problema de ser capaz de montar tais componentes num dispositivo tão pequeno, enquanto que para quem desenvolve aplicações o desafio principal é aproveitar ao máximo o potencial (bastante limitado) dos componentes, sem desvalorizar o consumo energético produzido quando estas aplicações são utilizadas, uma vez que também este é um recurso limitado (pela bateria do dispositivo). No entanto, os avanços tecnológicos nesta área são quase diários e, consequentemente, são desenvolvidos dispositivos móveis cada vez mais potentes e semelhantes a pequenos computadores móveis, chegando nalguns casos a substituir os computadores de secretária em rotinas diárias como a consulta de notícias, a visualização de filmes e a leitura e composição de correio electrónico.

3.1.1 Tipos de dispositivos móveis

De entre os vários tipos de dispositivos móveis existentes, os quatro mais conhecidos e utilizados actualmente — telemóveis, PDAs, *smartphones* e *tablets* — são apresentados de seguida.

Telemóveis Um telemóvel, tal como o nome indica, é um telefone móvel que pode ser utilizado para comunicação à distância sem fios. Surgiram há cerca de 38 anos [13] e, além da função básica de telefonia, a maior parte dos telemóveis actuais também disponibilizam outros serviços de comunicação como por exemplo mensagens escritas (SMS), mensagens de multimédia (MMS), vídeo-chamadas e acesso à Internet (para navegação e *e-mail*). Há também outras funcionalidades comuns à maioria destes dispositivos, como lista de contactos, alarme/relógio, jogos, calculadora, leitores de multimédia e captura de fotografias/vídeos. A maioria dos telemóveis, alem das teclas de função específicas (e.g. para seleccionar opções), têm um teclado físico composto pelos números de 0-9, as teclas asterisco ("*") e cardinal ("#"), ou então apresentam um mini-teclado "QWERTY" idêntico aos utilizados nos computadores pessoais.

PDAs Um Assistente Pessoal Digital (PDA) é um dispositivo móvel que tem como propósito principal disponibilizar funções de gestão de informação (pessoal ou profissional). Entre outras funcionalidades, a maioria dos PDAs permite organizar uma agenda digital com capacidades de gestão de contactos, calendário de eventos (e.g reuniões), lista de tarefas, notas

pessoais e, tipicamente, incluem também um cliente de *e-mail* e um navegador de Internet. Por este motivo, são dispositivos mais orientados ao uso profissional, uma vez que as características que apresentam facilitam a integração com o sistema informático de um negócio e a sincronização de dados entre os PDAs e computadores de secretária. No que toca a conectividade, estes aparelhos habitualmente disponibilizam ligações com e sem fios, sendo as ligações físicas por USB e as conexões sem fios por "Wi-Fi", "Bluetooth" e/ou Infra-vermelhos. Relativamente a características físicas, estes aparelhos costumam possuir um ecrã sensível ao toque, e por vezes incluem uma "caneta" (*stylus*) para facilitar selecções e operações nos dispositivos.

Smartphones Os *smartphones* são aparelhos que resultam da combinação entre funcionalidades de telemóveis com PDAs. Estes dispositivos versáteis apresentam as vantagens dos outros dois aparelhos, com as propriedades de produtividade e poder de processamento dos PDAs, em conjunto com as capacidades de comunicação e entretenimento dos telemóveis, tudo num só aparelho que pode ser transportado no bolso. Actualmente, estes dispositivos encontram-se tecnologicamente mais avançados que os seus predecessores, sendo que além das características habituais dos telemóveis e dos PDAs ainda costumam incluir câmaras fotográficas de alta resolução, navegação por GPS e outros sensores (e.g. giroscópios) que aumentam a utilidade destes aparelhos. Também os ecrãs destes dispositivos se destacam por serem, na sua maioria, ecrãs *multi-touch* (i.e. sensíveis a vários toques em simultâneo na mesma superfície) de alta resolução (que rondam os 480x800 *pixels*).

Tablets De entre estes quatro tipos de dispositivos móveis, os *tablets* são os que estão há menos tempo no mercado. Quando surgiram, eram reconhecidos pela forma de pequenos computadores com um ecrã sensível ao toque como método primário de operação, mas hoje em dia são mais frequentemente vistos como "*smartphones* grandes" com melhor autonomia, uma vez que são em quase tudo idênticos aos *smartphones* — desde os sistemas operativos que correm, às características de hardware e funções de conectividade que apresentam. Assim sendo, a principal diferença entre estes dois, além do tamanho de ecrã evidentemente superior nos *tablets*, é que os *smartphones* são mais orientados ao uso como dispositivo de comunicação e facilmente transportados para todo o lado, enquanto que os *tablets* são vocacionados para o entretenimento e navegação na Internet, e mais propensos ao uso prolongado em locais fixos (como em casa ou no local de trabalho, por exemplo), dadas as suas dimensões.

Figura 3.1: Exemplo de dispositivos móveis

Na figura 3.1 estão representados exemplos de cada um destes tipos de dispositivos móveis, mais especificamente: o telemóvel Nokia C5, o PDA Palm TX, o *smartphone* Apple iPhone 4S e o *tablet* Samsung Galaxy Tab 10.1. As características destes dispositivos móveis estão comparadas na tabela 3.1, onde são apresentadas as principais diferenças de desempenho e dimensões entre estes. No entanto, uma vez que os *smartphones* adoptaram as características dos PDAs e dos telemóveis (como sendo uma evolução lógica destes dois dispositivos), isso teve impacto no desenvolvimento tecnológico dos PDAs, que seguiram o mesmo caminho pela inclusão de antenas telefónicas nestes dispositivos. Por este motivo, sendo actualmente difícil distinguir os PDAs dos *smartphones* pelas suas características, é apresentado um PDA mais antigo (sem acesso à rede móvel) que os restantes dispositivos, apenas a título de exemplo.

3.1.2 Sistemas Operativos móveis

Os avanços tecnológicos na área dos dispositivos móveis têm levado a que estes dispositivos rumem todos na mesma direcção: apesar de apresentarem um exterior com aspecto diferente de aparelho para aparelho, no seu interior contêm componentes com características semelhantes (ou seja, mesma frequência de CPU, mesma quantidade de memória RAM, etc), dentro de cada gama de dispositivos. Para que cada aparelho seja capaz de desempenhar as funcionalidades para que foi desenvolvido, não é suficiente que apenas contenha os componentes físicos correspondentes — é preciso que exista *software* que disponibilize as operações a executar, suportadas pelo respectivo *hardware* do dispositivo. Apesar de praticamente todos

Características	Nokia C5	Palm TX	Apple iPhone 4S	Samsung Galaxy Tab 10.1
Dimensões (cm)	11,2 x 4,6 x 1,2	12,1 x 7,8 x 1,6	11,5 x 5,9 x 0,9	25,7 x 17,5 x 0,9
Peso (gramas)	89	149	140	565
Núcleos de CPU	1	1	2	2
Frequência por CPU	600MHz	312MHz	1GHz	1GHz
Memória volátil (RAM)	128MB	32MB	512MB	1GB
Memória persistente	50MB	128MB	16/32/64GB	16/32/64GB
Ecrã (polegadas)	2,2	3,9	3,5	10,1
Resolução (*pixels*)	240 x 320	320 x 480	640 x 960	1280 x 800
Bateria (em mAh)	1050	1250	1420	7000
Bateria em *standby*	até 25 dias	até 17 dias	até 8 dias	até 75 dias

Tabela 3.1: Comparação de dispositivos móveis (valores aproximados)

Fontes: http://www.gsmarena.com e http://www.pdadb.net

os dispositivos móveis actuais permitirem a sua personalização pela instalação de aplicações, que acrescem novas e diferentes funcionalidades aos mesmos, o que distingue estes dispositivos a nível de utilização é o Sistema Operativo (OS) que executam.

Existem vários OSs móveis, com características que os diferenciam, mas todos com o mesmo propósito: fornecer uma interface (gráfica) com que os utilizadores possam interagir de forma a desempenhar as funções para que estes foram feitos, tendo em conta o ambiente condicionado em que são executados, tais como a capacidade de processamento limitada, ecrãs de dimensões reduzidas e o efeito do consumo energético que se reflecte na autonomia dos mesmos (quando comparados aos sistemas operativos de computadores de secretária). Os principais OSs móveis da actualidade são o Google Android, o Samsung bada, o Blackberry OS da RIM, o Apple iOS, o Symbian OS da Nokia, o webOS da Hewlet Packard e o Windows Mobile/Windows Phone 7. Entre estes, são contextualizados de seguida os cinco sistemas que possuem maior quota de mercado [19].

Google Android O Android é um sistema operativo móvel, desenvolvido pela Google, para *smartphones* e *tablets*. Caracteriza-se por ser um OS livre e de distribuição gratuita, o que permite aos fabricantes personalizar a experiência de utilização da forma que acharem mais conveniente e que qualquer fabricante ou utilizador modifique e distribua o sistema sem

qualquer restrição. O OS Android não está associado a um hardware específico (i.e. não está preso a apenas uma marca ou um aparelho), o que justifica o facto de actualmente ser o sistema operativo móvel mais vendido do mundo, com aproximadamente 38,9% do mercado (como demonstrado na tabela 3.2).

Blackberry OS O sistema operativo BlackBerry, disponível para *smartphones* e *tablets*, foi desenvolvido pela multinacional Canadiana *Research In Motion* (RIM). Este, caracteriza-se por apresentar uma forma eficiente e simples de gestão de *e-mail* (baseada na tecnologia "*push*") com protecção criptográfica dos dados armazenados no dispositivo, o que permite uma constante, segura e rápida sincronização de mensagens de correio electrónico. O BlackBerry OS foi projectado para satisfazer as necessidades empresariais, permitindo um acesso total e em tempo real à informação da empresa e, de igual modo, melhorar o tempo de resposta entre clientes e fornecedores.

Apple iOS Desenvolvido pela Apple, este OS móvel de nome iOS está disponível exclusivamente para dispositivos da mesma marca (mais frequentemente referidos como "*iDevices*"). Distingue-se por ser o sistema operativo da linha de *smartphones* mais vendida do mundo (o iPhone), com uma quota de cerca de 19% do mercado do total de *smartphones* vendidos [17]. A interface gráfica intuitiva, a estabilidade do OS, a inovação apresentada (visto ser o primeiro sistema operativo móvel orientado à navegação exclusiva por toque) e o excelente desempenho são as principais características que distinguem esta plataforma.

Symbian OS O Symbian é um sistema operativo cuja origem remonta ao ano de 1998 [25], com um OS desde então bastante popular entre telemóveis e posteriormente actualizado para a utilização em *smartphones*. Surgiu de uma parceria entre a Nokia, a Ericsson, a Panasonic e a Samsung, apesar de estar principalmente associado à marca Finlandesa Nokia, sendo que este esforço de colaboração entre as companhias referidas justifica a elevada quota de mercado do Symbian. No entanto, a recente decisão de mudança de OS móvel por parte da Nokia (tendo optado pelo Windows Phone 7 para os seus dispositivos futuros [34]) e a acentuada fragmentação do OS — com diferentes versões, GUIs e funcionalidades por cada aparelho — explicam a queda percentual prevista para os próximos anos (apresentada na tabela 3.2).

Windows Mobile/Windows Phone 7 O Windows Phone 7, desenvolvido pela Microsoft, é o mais recente sistema operativo destinado a *smartphones*, sendo o sucessor do Windows Mobile (principalmente direccionado a PDAs). Este OS aposta na simplicidade de navegação

Sistema Operativo	Quota de Mercado 2011	Quota de Mercado 2015	CAGR[1] 2011-2015
Android	38,9%	43,8%	23,7%
Blackberry OS	14,2%	13,4%	18,3%
Symbian	20,6%	0,1%	-68,8%
iOS	18,2%	16,9%	17,9%
Windows Mobile/Phone 7	3,8%	20,3%	82,3%
Outros	4,3%	5,5%	27,6%

Tabela 3.2: Crescimento esperado dos Sistemas Operativos móveis

Fonte: IDC *Worldwide Quarterly Mobile Phone Tracker* [19]

como principal ponto forte, através da Interface Gráfica (GUI) de nome "Metro". Entre outras características que distinguem esta plataforma dos restantes sistemas operativos móveis estão, por exemplo, a integração com o Microsoft Office e a integração com o serviço de jogos "Xbox LIVE" da mesma empresa. Apesar de o Windows Phone 7 ainda ser bastante recente, estas propriedades do OS conjugadas com a decisão da Nokia (referida acima) fundamentam a estimativa de crescimento de aproximadamente 82%.

3.1.3 O impacto dos *smartphones*

Os *smartphones*, tal como referido no sub-capítulo 3.1.1, são dispositivos móveis bastantes versáteis, capazes de agradar aos mais variados grupos populacionais pelas funcionalidades de que dispõem. Da perspectiva de um utilizador casual, estes dispositivos servem de leitores de multimédia portáteis — capazes de reproduzir diferentes formatos de áudio e vídeos de alta resolução, quer localmente quer através da Internet —, máquinas fotográficas capazes de capturar fotografias e filmar com alta qualidade, aparelhos de navegação por GPS com comandos de voz e mapas sempre actualizados, agendas digitais que permitem organizar calendários, contactos, tarefas e notas pessoais, navegadores de Internet portáteis (seja por *Wi-Fi* ou por ligação à banda larga móvel), e tudo isto sem nunca esquecer um dos propósitos principais — o de fazer chamadas de voz. Já quanto ao espectro de um utilizador profissional, estes dispositivos também se demonstram ser bastante populares para quem requer a conveniência e utilidade de certos aspectos de um computador pessoal em situações em que não seria prá-

[1]Nota: CAGR — Taxa Composta de Crescimento Anual(do inglês *Compound Annual Growth Rate*): utilizada para medir a taxa de crescimento uniformizada durante um período de tempo específico

tico transportar um, tais como por exemplo consultar *e-mails* enquanto se está numa fila de espera, preencher relatórios e actualizar folhas de cálculo durante uma viagem de comboio ou fazer uma breve pesquisa na Internet sobre a localização de um certo escritório e utilizar a navegação por GPS para chegar a esse local.

A utilidade apresentada e a possibilidade de utilização destes dispositivos em situações tão diversificadas como as exemplificadas, justificaram a sua rápida adopção nos mercados dos países desenvolvidos, e o desenvolvimento acelerado das tecnologias de que estes dependem resulta na constante actualização dos modelos de gama alta e na introdução de modelos de gama baixa em países sub-desenvolvidos e noutras economias emergentes. Por estes motivos, é esperado que no fim de 2011 tenham sido vendidos cerca de 462 milhões de *smartphones* a nível mundial, o que representa aproximadamente 26% do total de vendas de dispositivos móveis, uma vez que se estimam alcançar 1790 milhões de unidades vendidas pelo final do mesmo ano [50]. Comparativamente aos anos anteriores, é possível constatar que esta continua a ser uma área em crescimento, visto que as vendas de *smartphones* têm aumentado 74% de ano para ano [5].

A prova de que estes dispositivos vieram para ficar é o impacto que tiveram pelo crescimento exponencial de utilizadores com *smartphones*, destacando-se não só entre os aparelhos móveis mas também no mercado dos restantes dispositivos electrónicos com capacidades de processamento, em geral. Esta adopção tem aumentado a um nível tão significativo, que apesar das previsões de 2010 terem apontado para que as vendas de *smartphones* ultrapassassem as de computadores pessoais [28] em 2012, a verdade é que no final do ano de 2011 isto já terá acontecido, uma vez que se esperam vender cerca de 400 milhões de computadores [9] (face ao valor de 462 milhões de *smartphones* já referido). Também em relação aos telemóveis estes superam as expectativas, tendo já ultrapassado na Europa Ocidental as vendas de telemóveis, durante o segundo trimestre do mesmo ano [18].

Mesmo com todas as funcionalidades que estes dispositivos apresentam, a comunicação (seja por voz, video chamada, sms, mms ou e-mail) é a mais usada [12, 49], sendo que o facto de os utilizadores terem um *smartphone* sempre à mão, disponível a toda a hora, não só permite mas também encoraja a que estes comuniquem entre si, mesmo em situações que isto provavelmente não ocorreria sem estes aparelhos [37]. O efeito da comunicação nestes

dispositivos móveis pode também ser observada pela elevada taxa de penetração das subscrições móveis nos países desenvolvidos, com uma média superior a 114 subscrições por cada 100 habitantes [20].

Em comparação aos restantes tipos de dispositivos móveis, é fácil constatar a principal vantagem que este apresenta: é possível obter, num *smartphone*, as mesmas funcionalidades e possibilidades de utilização num dispositivo de tamanho tão reduzido como os telemóveis e os PDAs, mas com um poder de processamento não muito distante dos *tablets*. Outra característica que os distingue dos telemóveis e PDAs é a velocidade de acesso à Internet. Actualmente, o sistema de rede mais frequente de um *smartphone* é o 3G (que representa a "Terceira Geração de redes móveis"), havendo já dispositivos com rede 4G, o que permite que estes aparelhos acedam à Internet com velocidades idênticas às apresentadas pelos serviços de banda larga fixos, para computadores de secretária. No entanto, apesar da utilidade de aceder à Internet com estas velocidades e da disponibilidade facilitada por se tratar de um dispositivo móvel, os preços praticados e as restrições de uso estipuladas são demasiado elevados, o que se pode demonstrar ser uma entrave à utilização de Internet nestes dispositivos. Por esta razão, o acesso à banda larga móvel é pouco comum em países em desenvolvimento, onde há cerca de 34% menos subscrições que nos países desenvolvidos [20].

Além dos motivos já referidos, houve outro factor responsável pelo crescimento acentuado de utilização dos *smartphones*: as aplicações desenvolvidas especificamente para estes sistemas (mais frequentemente chamadas de "*apps*"). Há inúmeras soluções para estes dispositivos, que são disponibilizadas aos utilizadores pela forma de aplicações móveis, desenvolvidas com APIs específicas (sendo estas fornecidas por quem desenvolve os OSs móveis). Estas APIs, permitem que as aplicações resultantes tenham um melhor nível de integração com o OS do *smartphone* e de acesso às componentes físicas (*hardware*) do mesmo, em relação às aplicações existentes para outros dispositivos móveis como os telemóveis.

A início, havia o problema de propagação deste tipo de soluções — quem pretendesse obter uma aplicação, tinha que a localizar na Internet e descarregar manualmente, o que se mostrava ser um processo moroso e pouco fiável, uma vez que não existiam indicadores da autenticidade nem avaliações de utilizadores sobre a aplicação. No entanto, isto tornou-se banal com a criação de "*appstores*" específicas a cada OS, responsáveis pela disponibilização

e manutenção de aplicações móveis para os dispositivos desse OS. Com estas plataformas de distribuição, hoje em dia cerca de 45% dos utilizadores de *smartphones* descarregam pelo menos uma aplicação por semana, sendo que na altura da escolha de uma aplicação, dão maior peso ao factor de portabilidade desta (i.e. preferem aplicações cuja utilidade assente no facto de fazerem uso da mobilidade do dispositivo) [8]. Com o sucesso evidente das *appstores*, é esperado que o lucro gerado pela venda de aplicações móveis no ano 2011 atinja os 15 mil milhões de dólares [31] (apesar de também existirem aplicações gratuitas).

Uma das vantagens do uso destas "lojas virtuais", quando conjugado com as capacidades de conectividade dos *smartphones*, é a gratificação imediata pela aquisição instantânea de aplicações — ao contrário do que acontece com produtos físicos e lojas reais, ao descarregar algo de uma *appstore* o conteúdo é instalado de imediato e fica pronto a usar em apenas alguns segundos. Em comparação às alternativas físicas, certas aplicações móveis (tais como jogos, aplicações para leitura de livros e para visualização de vídeos) permitem que o utilizador obtenha pela Internet o conteúdo que pretende consumir, sem que necessite de esperar em filas de espera nem que se tenha de deslocar a algum sítio específico para esse propósito.

Apesar de esta característica motivar o uso do *smartphone* como dispositivo de entretenimento, também existem aplicações direccionadas para a produtividade e uso profissional, como gestores de tarefas a fazer, aplicações de edição de documentos, calculadoras científicas e aplicações de acesso remoto a computadores (e.g. por VNC). Mais recentemente, surgiram soluções de comércio electrónico que fazem uso de componentes de *hardware* como antenas NFC (*Near Field Communication*) para efectuar pagamentos electrónicos e para propósitos de identificação, pelo que se espera que no futuro próximo os telemóveis acabem por desempenhar ainda outra função: a de uma carteira virtual.

3.2 O Sistema Operativo Android

O OS Android é um sistema operativo móvel de código aberto desenvolvido pela Open Handset Alliance e gerido pela Google. Um dos propósitos desta plataforma é facilitar a instalação do OS em *smartphones* e *tablets* sem que seja necessário a adaptação do sistema operativo para cada dispositivo. No entanto, é possível modificar o OS de modo a permitir a melhor integração com o dispositivo, pelo facto de o código fonte deste sistema estar aces-

sível publicamente e ser de distribuição livre, o que significa que o Sistema Operativo pode ser expandido para incorporar novas tecnologias ou para outros casos de uso específicos — havendo actualmente vários produtos diferentes com este OS, tais como Netbooks, leitores de música portáteis, telefones fixos e até sistemas de TV. Também por ser gratuito, optar por este sistema operativo reflecte-se em custos de desenvolvimento inferiores para quem produz os aparelhos e, consequentemente, em dispositivos mais baratos para o consumidor final.

Apesar de o sistema Android ser desenvolvido para ser executado em vários cenários diferentes, existem actualmente duas distribuições oficiais, geridas pela Google, para *smartphones* e para *tablets*, especificamente. As versões oficiais mais recentes deste OS e respectiva quota de utilização, estão representadas na figura 3.2, com as versões para *tablets* — 3.0 e superiores — comparadas à direita (por totalizarem apenas 1,8% da utilização).

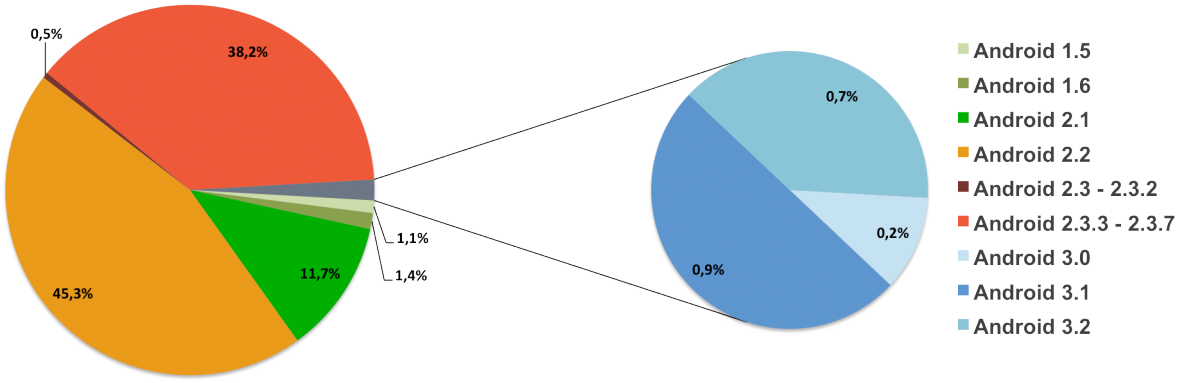

Figura 3.2: Distribuição actual de utilização por versões do OS Android

Fonte: http://developer.android.com/resources/dashboard/platform-versions.html[1]

Relativamente aos *smartphones*, a versão mais utilizada actualmente é a 2.2, que representa perto de metade do total de utilizações deste sistema operativo (mais precisamente, 45,3%). Já no caso dos *tablets*, a versão de Android mais comum é a 3.1 com praticamente metade da quota de utilização nestes dispositivos, não sendo porém um valor muito significativo quando comparado ao restante mercado deste OS, uma vez que representa menos de 1% do total.

A existência de duas distribuições do sistema operativo (conforme o dispositivo em que é

[1]Página acedida em 13 de Outubro de 2011

executado) é motivada pela diferença de dimensões dos ecrãs entre estes dispositivos — enquanto que um *tablet* contém um ecrã maior onde pode ser apresentada muita informação em simultâneo, os ecrãs dos *smartphones* são cerca de três vezes menores e por isso a disposição visual da informação é mais concentrada. A consequência desta decisão de separar os ambientes de *smartphones* e *tablets* é que apesar de uma aplicação desenvolvida para *smartphones* Android também funcionar em *tablets*, o contrário nem sempre se verifica, quando são usados componentes específicos a este último. No entanto, esta fragmentação de distribuições estará resolvida na próxima versão do Android, intitulada "*Ice Cream Sandwich*" (versão 4.0), em que as duas distribuições serão unificadas e não haverá a necessidade de distinção entre desenvolvimento para *smartphones* ou para *tablets*[2].

3.2.1 Arquitectura do Android

A plataforma Android está estruturalmente dividida em quatro camadas principais, representadas na figura 3.3, nomeadamente a camada das aplicações, a *framework* aplicacional, a camada com as bibliotecas nativas e o *runtime* do sistema e a camada base do *kernel* Linux.

Figura 3.3: A arquitectura do sistema Android

O que suporta esta estrutura é o *kernel* Linux, uma vez que o OS Android baseia-se em

[2]Conforme descrito em http://android-developers.blogspot.com/2011/09/preparing-for-handsets.html (acedido a 15 de Outubro de 2011)

Linux (mais especificamente, na versão 2.6 [3]) para fornecer serviços essenciais, como por exemplo a gestão de memória, processos e ligações de rede. Desta forma, o *kernel* também funciona como uma camada de abstracção que permite o acesso (controlado) ao *hardware* dos dispositivos pelos restantes componentes desta estrutura.

Num nível superior, estão as bibliotecas nativas e o *runtime* do OS. As bibliotecas nativas são bibliotecas usadas por vários componentes do sistema em camadas superiores, escritas em C/C++. O propósito destas é expor, às camadas acima, algumas funcionalidades específicas que de outro modo não estariam disponíveis, por dependerem de recursos do sistema cujos níveis superiores não têm permissões para aceder. A título de exemplo, entre estas bibliotecas estão incluídos o OpenGL ES (para processamento gráfico, quer por software quer por hardware) e o SQLite (para suporte a bases de dados relacionais). Quanto ao *runtime* do Android, este é composto por uma máquina virtual (a Dalvik VM), onde são executados os processos das aplicações, e por bibliotecas de suporte, que disponibilizam parte das funcionalidades existentes nas principais bibliotecas da linguagem Java.

A *framework* aplicacional, situada na camada acima do *runtime* e das bibliotecas nativas, é composta por vários componentes que devem ser usados por quem desenvolve aplicações, de forma a implementar a estrutura necessária para que a aplicação possa ser executada (esta estrutura é detalhada no sub-capítulo 3.2.3). Para tal, é disponibilizada uma *Application Programming Interface* (API) em Java, que estes devem utilizar para tomar partido das capacidades disponibilizadas pelas camadas inferiores.

No topo da estrutura do sistema Android, encontram-se então as aplicações deste OS. Ao contrário do que acontece na maioria das outras plataformas móveis, uma característica peculiar do Android é que todas as aplicações são desenvolvidas da mesma forma, ou seja, com recurso à API disponibilizada pela *framework* aplicacional. O que isto significa é que funcionalidades do OS como o despertador, a lista de contactos, o menu principal das aplicações e até os teclados virtuais são todas desenvolvidas segundo o mesmo princípio que as aplicações de terceiros — sendo que estas funcionalidades podem também ser substituídas por alternativas desenvolvidas por terceiros.

[3] http://developer.android.com/guide/basics/what-is-android.html (acedido em 17 de Outubro de 2011)

3.2.2 Ambiente de desenvolvimento

Para desenvolver aplicações para este OS, é necessário utilizar as APIs disponibilizadas, incluídas no Android *Software Development Kit* (SDK). A linguagem de programação utilizada para aceder a estas APIs e desenvolver aplicações para o Android é a linguagem Java, o que em parte cativa alguns programadores que já estão acostumados a desenvolver soluções para múltiplas plataformas. No entanto, o conceito de uma aplicação para Androide a estrutura que estas devem ter para funcionarem correctamente num dispositivo móvel é diferente das habituais aplicações para a máquina virtual Java presente nos computadores pessoais, como pode ser verificado no sub-capítulo 3.2.3.

Com o objectivo de facilitar o desenvolvimento de aplicações para esta plataforma, o SDK existente contém várias aplicações de exemplo, ferramentas específicas com a respectiva documentação e um *plugin* de integração com o IDE Eclipse (actualmente para a versão 3.5 ou superior), que permite a execução da maior parte destas ferramentas directamente a partir do IDE. De seguida são descritas algumas destas ferramentas de desenvolvimento.

ADB Um dos passos essenciais para testar o funcionamento das aplicações e corrigir possíveis erros é a depuração. Para este efeito, uma das ferramentas mais utilizadas deste SDK é o *Android Debug Bridge* (ADB), que é baseado num sistema cliente–servidor que permite a comunicação com um dispositivo conectado à máquina em que é executado (via USB). Além de facilitar a depuração (pela execução do comando "logcat" no ADB), esta ferramenta versátil dispõe também de outras funcionalidades como por exemplo o acesso a uma *shell* remota (executada no dispositivo móvel a que está conectado) e a transferência de dados de e para um dispositivo.

DX Esta ferramenta faz parte do processo de compilação, sendo responsável por converter código Java já compilado em *bytecode* para a Dalvik VM, de forma a poder ser executado em ambientes Android. Durante esta sequência de comandos, desde o código fonte em Java até o produto final de uma aplicação para Android, o procedimento habitual (que é transparente ao programador caso este utilize o *plugin* para o Eclipse referido) é a compilação do código Java, seguido da conversão para o formato ".dex" por esta ferramenta, a criação de um ficheiro ".apk" (um "*Android Package*") com o *bytecode* obtido e outros recursos compilados à parte (e.g. imagens e sons) com a ferramenta APKBuilder, que é posteriormente assinado digitalmente com recurso ao JarSigner e optimizado pela ferramenta ZipAlign (que alinha o

conteúdo do ficheiro ".apk" de modo a reduzir a quantidade de memória utilizada durante a execução da aplicação).

Emulator O Emulator, tal como o nome indica, é um emulador de dispositivos Android. Com esta ferramenta é possível simular um ambiente de execução com este sistema operativo, com um número arbitrário de máquinas virtuais e com versões diferentes do OS, conforme a necessidade do utilizador. Isto permite que sejam testadas diferentes configurações durante a depuração de uma aplicação, sejam estas configurações relativas a dimensões de ecrã, capacidades de processamento, memória disponível ou a própria versão do sistema. Caso não esteja disponível um dispositivo físico para os testes da aplicação, é também possível recorrer unicamente a esta solução para o desenvolvimento da aplicação.

SQLite3 Para a manutenção de bases de dados das aplicações, uma solução prática é recorrer à ferramenta SQLite3, fornecida com o SDK do Android. Como referido no subcapítulo 3.2.1, encontra-se uma versão do SQLite entre as bibliotecas nativas do OS, de forma a facilitar o armazenamento de dados por parte das aplicações (de um modo estruturado, com bases de dados relacionais). A ferramenta de desenvolvimento SQLite3 permite consultar, criar, modificar, apagar e avaliar remotamente (ou localmente, no caso de ser utilizado um emulador) bases de dados em dispositivos.

TraceView O TraceView pode ser utilizado para apresentar uma análise de execução de uma aplicação, pela representação gráfica de um painel cronológico, onde é apresentado o número de chamadas a métodos (com distinção de chamadas recursivas), a percentagem de uso do processador e o tempo de execução, sendo que estes dois últimos valores podem ser avaliados quer por processos quer por métodos da aplicação. Os eventos são registados com o auxílio da classe "*Debug*" da API fornecida, pelo uso de métodos que controlam os intervalos de tempo ou a sequência de comandos a registar.

3.2.3 Estrutura das aplicações

As aplicações desenvolvidas para este sistema são compostas por componentes modulares, escritos em Java, que obedecem aos padrões especificados pela Google e acessíveis pela API do SDK do Android. O facto de serem modulares, facilita a futura substituição de componentes, implementações diferentes ou até a ligação a componentes de outras aplicações. Existem quatro tipos de componentes destas aplicações, nomeadamente: *Activities* (Actividades),

Broadcast Receivers (Receptores de Difusões), *Content Providers* (Fornecedores de Conteúdo) e *Services* (Serviços).

Activities As *Activities* são componentes que representam tarefas distintas, sendo que quase todas interagem com o utilizador via GUI, através da projecção de uma ou mais *Views* (Vistas). Uma *View* é um elemento gráfico, definido em XML, que representa o conteúdo a apresentar no ecrã por uma actividade. Deste modo, a relação entre *Activities* e *Views* é idêntico ao apresentado em arquitecturas MVC (*Model-View-Controller*), pela separação da lógica de negócio (que incluem as *Activities*) da lógica da apresentação (representada pelas *Views*). As aplicações desenvolvidas para esta plataforma são habitualmente compostas de várias *Activities*, uma para cada "ecrã" apresentado ao utilizador. O ciclo de vida de uma *Activity* (ou Actividade), detalhado mais à frente, está representado na figura 3.4.

Broadcast Receivers Tal como o nome indica, a função desempenhada por estes componentes é a intercepção e processamento de notificações, pela recepção de mensagens difundidas pelo OS. Estas notificações podem ser originadas quer por eventos específicos do sistema (e.g. para indicar que o dispositivo foi reiniciado ou que está com pouca bateria), quer intencionalmente por outras aplicações (e.g. para avisar que foi recebido um *e-mail* ou que uma música está a ser reproduzida). Uma vez que os *Broadcast Receivers* não possuem uma interface gráfica com que o utilizador possa interagir, o uso destes componentes é frequentemente representado por uma indicação (ou notificação) na barra de estado do OS e/ou a execução de uma *Activity* da aplicação correspondente.

Content Providers Os *Content Providers* são módulos que permitem a consulta e actualização de dados, de um modo transparente e independentemente do meio de armazenamento utilizado (i.e. esteja a informação armazenada numa base de dados SQLite, num cartão externo ou na *Cloud*, por exemplo). Isto facilita a criação de repositórios robustos com a abstracção do meio de armazenamento, possivelmente partilhados por várias aplicações, podendo até existir vários suportes de armazenamento diferentes utilizados por um só *Content Provider*.

Services Para tarefas que sejam executadas em segundo plano e sem a necessidade de apresentar uma GUI, os componentes utilizados são os *Services*. Estes componentes são executados no mesmo processo de sistema que a aplicação de que fazem parte, pelo que para operações mais pesadas o processo habitual é a criação de uma nova *thread*. Neste sentido,

os serviços podem ser vistos como componentes que executam operações morosas (sejam elas contínuas ou não) que não precisam da interacção do utilizador.

Estes quatro tipos de componentes são desenvolvidos com recurso à API da *Framework* Aplicacional, havendo classes (em Java) para cada tipo de componente implementar, com métodos característicos de cada uma dessas classes, de modo a disponibilizar as funcionalidades necessárias do componente em causa. Esta modularidade obtida pela separação das diferentes lógicas — com componentes distintos para a apresentação (pelas *Views*), a lógica de negócio (*Activities*, *Broadcast Receivers* e *Services*) e a camada de dados (*Content Providers*) — permite também a interligação entre as aplicações e o reaproveitamento de funcionalidades entre estas: a título de exemplo, é possível que um evento no calendário despolete o envio de um *e-mail*, e que o cliente de *e-mail*, ao adicionar um anexo, utilize a componente de seleccionar imagem de uma aplicação (como a "galeria"), que por sua vez faz uso da aplicação da câmara fotográfica para capturar uma fotografia e disponibilizar essa mesma imagem. Neste exemplo, podem ser percebidos os quatro tipos de componentes em acção: o *Service* do calendário (que verifica periodicamente se tem eventos para notificar), o *Broadcast Receiver* do cliente de *e-mail* (que intercepta as notificações pretendidas), o *Content Provider* do calendário (onde o cliente de *e-mail*, com o identificador de um evento específico, poderia ir buscar detalhes como a localização e descrição do evento) e as *Activities* do cliente de *e-mail*, da galeria e da aplicação da câmara fotográfica.

Esta funcionalidade reflecte-se também no modo como as aplicações do sistema Android são executadas: não existe um "ponto de entrada" único para as aplicações, é criado um processo para que estas possam ser executadas assim que seja necessário aceder a um componente das aplicações. O que isto significa é que caso uma aplicação disponibilize o acesso por outras aplicações a um *Content Provider*, um *Service* ou uma *Activity*, ou caso tenha um *Broadcast Receiver* para interceptar certos eventos, quando estes componentes forem activados é criado um novo processo de sistema para a aplicação de que fazem parte (caso o processo correspondente ainda não exista). Deste modo, apesar de para o utilizador final as ligações entre aplicações serem parte de uma só operação, os ambientes de execução destas componentes estão separados por cada aplicação interveniente, com um processo na Dalvik VM para cada aplicação.

Com a excepção dos *Content Providers*, os restantes componentes de uma aplicação para Android são activados através de mensagens assíncronas chamadas *Intents*. Estas mensagens indicam o que pretendem que seja executado e, por vezes, fornecem também dados para o componente de destino (denominados de *Extras*), de modo a que este os utilize durante a sua execução. Quanto à definição do componente a que se pretende aceder, um *Intent*, quer seja direccionado para componentes da mesma aplicação ou para componentes de outra aplicação, pode ser definido implicitamente ou explicitamente. Os *Intents* explícitos, mais frequentemente trocados entre componentes da mesma aplicação, descrevem explicitamente qual o nome da classe Java do componente que pretendem activar. Já os *Intents* implícitos, podem ser definidos através dos tipos de dados, categorias e/ou acções genéricas que pretendem desempenhar. Tanto uns como os outros são interpretados pelo OS, que é responsável por decidir se o componente que fez o pedido tem permissões para executar o componente de destino, e no caso dos *Intents* implícitos tem também o dever de decidir qual o componente específico que deve activar —- sendo isto concretizado através de *Intent Filters*, onde os componentes podem definir quais os tipos de dados, categorias e/ou acções genéricas que são capazes de realizar (ou que pretendem interceptar, no caso dos *Broadcast Receivers*).

Em relação aos *Content Providers*, estes são acedidos através de um mecanismo específico com o recurso a *Content Resolvers*. Tal como no caso dos *Intents* explícitos, os *Content Resolvers* necessitam de um *Uniform Resource Identifier* (URI) com o nome completo da classe Java (i.e. o "caminho" para o componente) do *Content Provider* que pretendem utilizar, identificando assim univocamente o conjunto de dados a aceder. Uma vez obtido o *Content Provider* pretendido, podem ser realizadas as consultas necessárias pela composição de *queries* SQL, que retornam cursores de dados sobre os quais os componentes podem iterar.

No sub-capítulo seguinte é detalhado o ciclo de vida de uma *Activity*, a componente mais utilizada nas aplicações desta plataforma. Esta representa também a componente mais afectada pelas possíveis alterações de estado de execução de uma aplicação, também referidos no capítulo seguinte, pois as aplicações Androidapresentam um fluxo de execução bastante diferente das habituais aplicações de outras plataformas de computação (como o caso dos computadores pessoais, por exemplo), pelo que é necessário compreender estas diferenças.

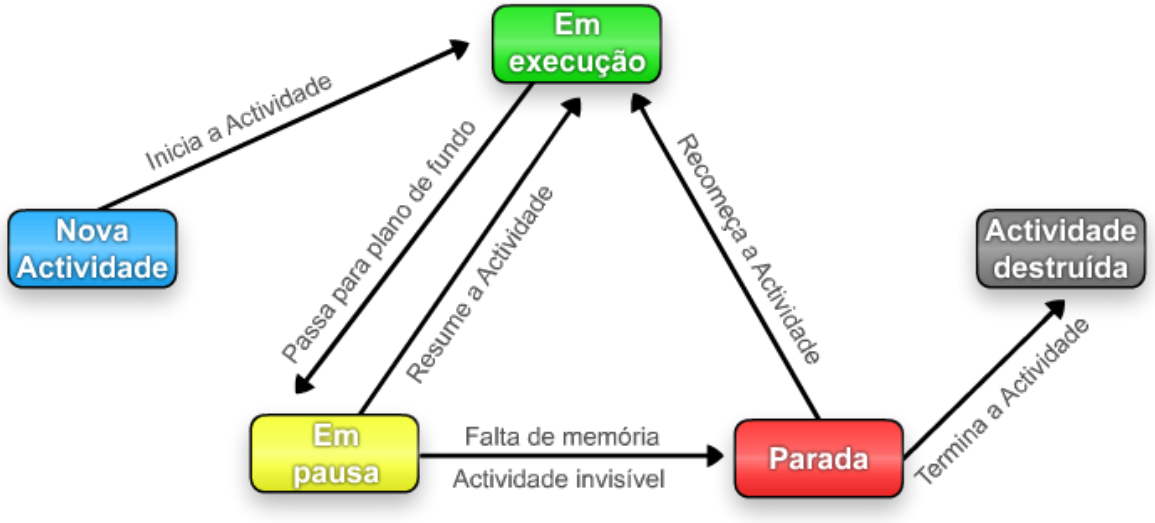

Figura 3.4: Ciclo de vida de uma Actividade

3.2.4 O ciclo de vida das actividades

Entre os quatro componentes de uma aplicação Android, o componente que tem maior impacto no ciclo de vida desta é a *Activity* (ou "Actividade"), uma vez que é o componente mais utilizado e essencial a qualquer aplicação, por apresentar a interface com que o utilizador pode interagir, focada a uma certa tarefa que este pretende realizar, e por habitualmente representar os possíveis "pontos de entrada" de uma aplicação. A gestão do estado das Actividades é automaticamente efectuada pelo OS, que conforme o estado de visibilidade e utilização destas, alterna entre os quatro estados essenciais (representados na figura 3.4) : em execução, em pausa, parada e destruída. Estes estados podem ser também "forçados" a alterar — pela própria Actividade ou por outro componente da aplicação — mas apenas para estados de menor prioridade, i.e. colocar em pausa uma Actividade que esteja em execução, parar uma Actividade em pausa ou terminar a Actividade parada. De seguida, é descrito como e quando o sistema muda o estado de uma Actividade:

1. Assim que uma Actividade é iniciada, o OS coloca-a no estado "em execução", com a qual pode ser interagida através da GUI;

2. Quando a Actividade perde o foco (i.e. quando passa para plano de fundo, por ser apresentada uma nova actividade no topo), esta passa para o estado "em pausa";

3. Caso o foco do utilizador volte à Actividade em pausa, esta é resumida (ou seja, retoma o estado de execução) com a disposição gráfica com que tinha sido deixada — se tiver

sido modificado o conteúdo ou disposição da *View* da Actividade (e.g. campos de texto preenchidos ou a deslocação vertical alterada), estas propriedades são retomadas;

4. Se a Actividade estiver invisível para o utilizador (com uma outra Actividade a ocupar totalmente o ecrã, por exemplo) ou em situações que seja necessário libertar memória para outros componentes poderem ser executados, a Actividade é parada;

5. Assim que o foco retome a uma Actividade parada, esta é recomeçada de forma idêntica a uma nova execução (apesar esta situação de poder ser diferenciada, como explicado mais à frente);

6. Quando a Actividade deixa de estar visível por ter terminado a sua tarefa, não sendo mais necessária, esta passa para o estado "destruída" (apesar de primeiro ser colocada em pausa e parada, antes deste estado final).

Durante a transição entre cada estado da Actividade, são invocados certos métodos na classe Java correspondente que podem ser implementados para que a Actividade possa actuar assim que o seu estado é alterado. Assim sendo, existem sete métodos para este efeito, nomeadamente: `onCreate()`, `onDestroy()`, `onPause()`, `onRestart()`, `onResume()`, `onStart()` e `onStop()`. No caso 1 da enumeração anterior, em que uma Actividade é iniciada, são invocados os métodos `onCreate()`, `onStart()` e `onResume()` (por esta mesma ordem). Já para o ponto 2, o método invocado é o `onPause()`, e quando a Actividade é resumida (caso 3), é executado o método `onResume()`. Assim que a Actividade é parada, conforme o caso do ponto 4, é invocado o método `onStop()` — sendo importante notar que uma Actividade nunca passa directamente do estado de execução para o estado "parada", pelo que é executado também o método `onPause()` (antes do método `onStop()`). Em relação ao ponto 5, caso a Actividade tenha sido parada por falta de memória, esta é recomeçada tal como uma nova actividade (i.e. igual ao caso 1). Porém, se esta tiver sido parada por se encontrar invisível, no lugar no método `onCreate()` é invocado o método `onRestart()` (seguidos dos métodos `onStart()` e `onResume()`), o que permite distinguir esta situação do caso de uma nova Actividade. Por fim, quando a Actividade é destruída, o método correspondente é o `onDestroy()`.

3.2.5 Segurança do Sistema Operativo

Hoje em dia, é comum o uso de dispositivos móveis como meio intermediário de autenticação e acesso a certos serviços, principalmente através da Internet, como por exemplo para

actualizar estados de redes sociais e aceder a contas bancárias *online* — tarefas que até então necessitavam de um computador pessoal para serem realizadas. Esta mudança no paradigma de disponibilização de serviços reflecte-se também na segurança com que estes podem ser utilizados, o que levanta questões como até que ponto é seguro recorrer dispositivos móveis para estas tarefas, comparativamente ao uso de computadores pessoais. Esta é também uma preocupação de quem produz os OS móveis, uma vez que o rápido crescimento deste tipo de dispositivos tem como consequência a existência de um novo alvo de ataques, ainda mais justificado pelo facto de dispositivos como os *smartphones* terem ultrapassado recentemente a venda de PCs.

Por ter como base um *kernel* Linux, o OS Android tira partido de uma plataforma sólida com vários anos de existência, aperfeiçoada para acomodar as funcionalidades de segurança necessárias durante a comunicação entre o *middleware* (as camadas superiores da arquitectura Android) e o *hardware* dos dispositivos. No entanto, o facto de este sistema tentar alcançar o equilíbrio com uma solução leve e portátil mas com as mais diversas funcionalidades, levanta questões quanto à correcção desta implementação e a segurança com que as aplicações são executadas.

Uma vez que as aplicações deste OS podem ser (e são maioritariamente) desenvolvidas por terceiros, a confiança depositada nas aplicações é fundamentada por dois factores principais: a autenticidade das mesmas (ou seja, até que ponto se pode afirmar que uma aplicação é autêntica e que foi desenvolvida pelo autor que lhe está associado) e as funções que estas desempenham (i.e. que recursos ou funcionalidades do sistema usam e se este uso pode ou não ser considerado de risco). Relativamente à autenticidade das aplicações, com vista a garantir a integridade das aplicações e a identificação (com não-repúdio de responsabilidade) dos seu autores, todas as aplicações de um mesmo autor têm de ser assinadas digitalmente pelo mesmo par de chave privada/certificado digital. Estas credenciais não necessitam de ser certificadas por uma CA, podendo o certificado ser auto-assinado, uma vez que servem unicamente para distinguir os autores das aplicações (e para permitir que duas aplicações de um mesmo autor sejam executadas no mesmo processo da Dalvik VM, caso este assim o deseje).

Como as aplicações são assinadas digitalmente, ao serem distribuídas pelo Android Mar-

ket (a loja virtual de aplicações oficial) há também a garantia de que futuras actualizações das aplicações partilham o mesmo autor [39]. Porém, os requisitos para publicar uma aplicação no Android Market são muito permissivos, sendo que basta pagar uma pequena quantia[4] e submeter o certificado digital (que não necessita de ser assinado por uma CA) com que as aplicações serão assinadas, e as aplicações podem ser logo distribuídas a partir desse momento. O facto de estas aplicações não serem verificadas por nenhuma entidade, conjugado com a facilidade com que qualquer pessoa coloca uma aplicação neste mercado, leva a que haja bastante *malware* no Android Market — algo que poderia ser evitado, mas que apenas é "corrigido" (com a remoção da aplicação do mercado) após as consequências serem detectadas e reportadas.

De modo a garantir que um utilizador toma conhecimento das funções de maior risco que uma aplicação pode executar, o OS Android possui um sistema extensivo de permissões, sendo que cada aplicação tem de indicar e requerer (durante a instalação da mesma) as permissões necessárias para tirar partido dessas funcionalidades específicas. Um problema frequente com o sistema de permissões utilizado é que muitas vezes os utilizadores, durante a instalação, simplesmente ignoram as permissões requisitadas por uma aplicação e aceitam a instalação sem as ler. Um autor mal intencionado pode aproveitar esta distracção dos utilizadores para aceder a funcionalidades que não deveria ter acesso, sem que o utilizador se aperceba — um caso comum quando se trata de jogos móveis em que, enquanto o utilizador está entretido a jogar, são comprometidas informações pessoais (como contactos e a localização do utilizador) sem que este tome nota de tal situação, e casos como permissões de mensagens mal utilizadas, em que podem ser enviadas mensagens de custo acrescentado ou de "SPAM" sem notificação [14].

Entre as principais permissões do Android, encontram-se por exemplo: aceder à Internet, aceder à lista de contactos, utilizar o GPS, consultar o estado da bateria, alterar configurações de rede, capturar fotografias, enviar SMS e realizar chamadas. As permissões das aplicações não são dinâmicas, uma vez que são definidas anteriormente à distribuição das aplicações, e não é apresentada nenhuma indicação visual caso uma aplicação tente utilizar uma funcionalidade para a qual não tem autorização (apesar de também nada ser executado) — o que pode ser problemático para programadores que não estejam familiarizados com a *framework*, pois é uma situação difícil de noticiar sem recorrer à depuração da aplicação.

[4] A taxa de inscrição é de 25 USD, à data actual.

Outra questão importante quando se considera a segurança de um sistema operativo é o nível de isolamento entre as aplicações. Neste contexto, as aplicações Android são executadas em processos diferentes da Dalvik VM, o que evita que uma aplicação aceda ao espaço de memória de outra aplicação ou que comprometa a gestão de recursos (e.g. o Unidade Central de Processamento (CPU)) por parte do OS com o propósito de prejudicar o funcionamento de outra aplicação. Além desta medida de prevenção, a cada aplicação instalada é atribuído um identificador de utilizador POSIX único (que apenas pode ser partilhado por aplicações de um mesmo autor e caso este o pretenda). Uma vez que cada aplicação é interpretada como sendo um utilizador diferente perante o sistema operativo, o acesso à memória física é também controlado, pelo que cada aplicação pode ter acesso de escrita e leitura apenas relativamente à sua pasta de instalação ou à memória física partilhada (i.e. onde são guardados os ficheiros multimédia, documentos, etc) — o que requer também a permissão de leitura e/ou escrita.

Apesar de este mecanismo de isolamento estar bem definido nas distribuições oficiais do OS, isto é contornável em dispositivos com privilégios de administração (mais conhecido como "acesso *root*"), o que pode ser perigoso quando são instaladas aplicações defeituosas ou mal intencionadas. Habitualmente motivados por funcionalidades adicionais, como a capacidade de personalizar extensivamente o OS e a possibilidade de executar aplicações que requerem permissões especiais (e.g. gestores de tarefas), muitos utilizadores optam por fazer o "*root*" aos seus dispositivos, o que inadvertidamente permite que os autores de *malware* aproveitem as falhas de segurança resultantes e, deste modo, subverter todos os mecanismos de segurança utilizados.

Outra técnica que pode ser utilizada para explorar o princípio de privilégio mínimo aplicado pelo Android é através de *escalation attacks* [46]. Este método consiste em realizar tarefas para as quais uma aplicação não tem permissões, e pode ser concretizada no OS Android caso uma aplicação com a permissão para uma certa tarefa tenha um "ponto de entrada" acessível através de outra aplicação sem essa permissão. Como exemplo, pode-se conceber uma aplicação X com a permissão de obter a localização do utilizador, que utiliza para construir uma lista com as coordenadas de restaurantes perto do dispositivo, acessível para consulta a outras aplicações através de um *Service*. Uma aplicação Y mal intencionada e sem a mesma permissão, pode recorrer a esse serviço para calcular a localização estimada

do utilizador e deste modo contornar o sistema de permissões. No entanto, a vulnerabilidade deve ser considerada da responsabilidade da aplicação e não da *framework* aplicacional do Android, pois esta situação pode ser prevenida com a definição de permissões específicas de utilização por cada componente acessível externamente (no caso exemplificado, o *Service* da aplicação *X* deveria requerer às aplicações externas a permissão respectiva para poder ser utilizado).

Em relação à correcção das funcionalidades do OS, apesar de esta plataforma ter por base um sistema sólido, parte das tecnologias utilizadas são muito recentes e, por isso, por vezes apresentam erros de implementação. Um problema detectado recentemente com a implementação da tecnologia NFC (*Near Field Communication*) em Android, tem como consequência o impedimento de utilização dessa funcionalidade, o que demonstra que esta pode ser um futuro alvo de ataques (e.g. durante transacções financeiras), como descrito em [30]. Outra questão relativa a problemas de implementação, prende-se com as APIs de segurança, mais especificamente sobre os *Security Providers* (utilizados pela arquitectura criptográfica da linguagem Java) — além dos *Security Providers* existentes estarem incompletos (por se tratarem de versões adaptadas especificamente para o ambiente móvel Android), estes não são frequentemente actualizados, o que pode levar a falhas de segurança por colmatar em versões desactualizadas e a problemas de funcionamento por se tratarem de versões adaptadas (não oficiais).

CAPÍTULO 4

Smart Cards

Um *Smart Card* pode ser definido como um cartão de dimensões reduzidas com circuitos integrados, sendo estes circuitos embutidos no cartão habitualmente sob a forma de um *chip*. Neste capítulo é introduzido o contexto em que esta tecnologia surgiu e a motivação para tal. São também apresentadas as principais características destes cartões, de hardware e software, e feita uma breve avaliação sobre as propriedades de segurança apresentadas pelos *Smart Cards*. Por fim, é referida a plataforma Java Card como ambiente de desenvolvimento e execução de aplicações nestes cartões, com detalhes sobre o processo de desenvolvimento e instalação destas aplicações.

4.1 Contextualização

Os *Smart Cards* estão entre os dispositivos electrónicos mais utilizados actualmente, apesar de na maioria dos casos o seu uso ser de tal modo transparente para os utilizadores que estes nem se apercebem da presença destes cartões. Seja em cartões de identidade, cartões para telecomunicações móveis ou até para pagamento de produtos e serviços através de cartões de crédito, o mais provável é que a maioria dos cidadãos tenha em sua posse vários *Smart Cards* e os usem com confiança no dia-a-dia, mesmo sem qualquer informação sobre os mecanismos que suportam tais sistemas.

A confiança depositada nos *Smart Cards* é proveniente do longo tempo de existência e da aceitação destes dispositivos no mercado internacional, motivado pela utilidade destes cartões e pelos elevados padrões de segurança a que este obedece (e.g. protecção contra cópia

de dados). Mas a história de cartões com o propósito de identificação e de cartões bancários não é recente, sendo que o primeiro cartão deste género surgiu há cerca de 60 anos, com um simples cartão de plástico a desempenhar a função de cartão de crédito destinado exclusivamente a membros de um clube de nome "*Diners Club*". Nesta altura, bastava apresentar estes cartões de plástico para se ter direito a refeições e estadias em hotéis (que apenas viriam a ser cobradas mais tarde) [15, 36].

Porém, estes cartões eram relativamente simples de copiar e a única informação que transportavam eram os dados directamente gravados na superfície do cartão, pelo que foram substituídos pelos cartões de banda magnética, capazes de armazenar informação em formato digital como suplemento dos restantes dados inscritos no cartão. Outra vantagem desta opção foi a redução da quantidade de papel necessária para pagamentos com estes cartões e outras transacções mais complexas, uma vez que anteriormente, além de apresentar os cartões necessários, era também preciso assinar certos documentos como forma de comprovar as operações realizadas [36]. Com a introdução de cartões de banda magnética, este passo deixou de ser estritamente necessário, pois as operações podem ficar então registadas tanto no cartão como no dispositivo de leitura. No entanto, também este tipo de cartões apresentam o mesmo grave problema: desde que se tenha o equipamento necessário, é possível ler e escrever dados na banda magnética [15].

Surgiu então, em 1968, a ideia de incorporar circuitos integrados num cartão de plástico, a que se viria a dar o nome de *Smart Card* [3]. Estes destacavam-se das restantes soluções por se apresentarem como dispositivos resistentes a adulterações e de uso restrito, protegidos por um certo código (e.g. palavra-passe) que apenas era conhecido por uma pessoa (o seu utilizador). Além de funcionarem como suporte de dados, os *Smart Cards* apresentavam ainda a vantagem de poderem executar operações lógicas e aritméticas nos próprios circuitos dos cartões, incluindo operações criptográficas (para facilitar a autenticidade e confidencialidade dos dados armazenados e/ou processados nos mesmos). Devido ao potencial destes cartões, que actualmente contêm maior poder de processamento e funcionalidades acrescidas através de sistemas operativos específicos (e.g. Java Card), tem-se observado a propagação desta tecnologia a nível mundial desde meados dos anos 90, fortemente motivado pela adopção de *Smart Cards* em áreas como as telecomunicações e pagamentos electrónicos [3, 15].

Figura 4.1: Exemplo de *Smart Card*: cartão de crédito

Uma questão crucial dos *Smart Cards* é que o principal propósito destes é serem percebidos como dispositivos de confiança, onde se pode armazenar dados privados e informações pessoais e onde é possível processar esta informação internamente com um elevado grau de fiabilidade, não tendo sido por isso desenvolvido como apenas mais uma plataforma de computação (até porque, quer a nível de poder de processamento como de capacidade de armazenamento, são inferiores à maioria dos restantes dispositivos electrónicos, como detalhado em 4.2.1). Para cumprir este objectivo e tornar possível a interoperabilidade de soluções desenvolvidas para estes cartões, os *Smart Cards* obedecem a padrões internacionais bem definidos, como por exemplo o `ISO 7816` (o padrão mais habitual, dividido em 14 partes por tópico de especificação[1]), o `EMV ICC` (padrão frequentemente utilizado quando estão envolvidas transacções financeiras) e/ou o `PC/SC` (utilizado como interface para ligação e utilização destes cartões em computadores pessoais), que especificam normas segundo as quais os leitores de cartões, os *Smart Cards* e as próprias aplicações se devem reger durante o desenvolvimento para estes cartões [15].

4.1.1 Utilidade dos *Smart Cards*

As principais características que determinam a utilidade de um *Smart Card*, são [27] :

[1] As principais partes do `ISO 7816` são as quatro primeiras, onde estão especificadas as características físicas, dimensões/localizações dos contactos eléctricos, sinais electrónicos e protocolos de transmissão e comandos interindustriais, respectivamente.

1. A capacidade de participar, de um modo transparente, numa transacção electrónica (e consequentemente a facilidade de utilização);

2. O facto de serem usados primariamente com o propósito de fornecer ou acrescer certas propriedades de segurança (tais como o armazenamento seguro de dados e a capacidade de executar operações criptográficas no cartão);

3. As propriedades físicas destes cartões, que evitam que estes sejam adulterados ou copiados com facilidade (descrito com maior detalhe em 4.2.1).

Uma vez que estes cartões não possuem qualquer mecanismo directo de interface de utilizador, o acesso a estes é sempre feito através de leitores específicos, o que permite que sejam usados *Smart Cards* sem que um utilizador se aperceba da sua presença, como indicado no ponto 1. Um exemplo notável onde pode ser observada esta propriedade trata-se do uso de cartões GSM (acrónimo de "*Global System for Mobile Communications*") de comunicações móveis, em que é inserido um *Smart Card* no dispositivo móvel (e.g. telemóvel) que é utilizado para autenticar o utilizador perante a sua rede móvel, sem que este tenha conhecimento dos passos reproduzidos para tal ou sequer da participação do cartão nessa transacção.

Em comparação com o sistema de cartões de banda magnética, os *Smart Cards* acrescem a propriedade de serem capazes de processar informação no próprio cartão, o que inclui operações criptográficas sobre dados armazenados e sobre dados transmitidos para o cartão, o que significa que é possível realizar operações como assinaturas digitais no próprio cartão, sem que as chaves criptográficas utilizadas sejam passadas para o exterior. Isto torna os *Smart Cards* em dispositivos capazes de serem utilizados como ambiente de processamento isolado, sem o risco de exteriorização de informação privada — já no caso dos cartões de banda magnética, uma vez que não possuem um processador e apenas funcionam como mecanismo de armazenamento de dados, isto não é possível sem que seja passada essa informação privada ao leitor de cartões. Quando conjugada esta capacidade de processamento referida no ponto 2 com a resistência contra cópia e adulteração de dados descrita no ponto 3, os *Smart Cards* podem ser utilizados como *tokens* de segurança fiáveis e de elevada mobilidade, com capacidade de personalização tanto de dados como de funcionalidades.

4.1.2 Aplicações actuais

Os *Smart Cards* são dispositivos com um papel cada vez mais importante no mundo actual, onde a expansão dos meios de comunicação digitais e a forte presença do dinheiro electrónico justificam ainda mais a adopção desta tecnologia, pela necessidade de segurança acrescida e portabilidade das soluções, com a vantagem de apresentarem conveniência de utilização e terem um baixo custo de produção.

Um dos usos mais comuns destes cartões é na área financeira, em que são usados *Smart Cards* como meio de autenticação em transacções electrónicas e/ou pela forma de carteiras electrónicas (com créditos que podem ser utilizados para pagar serviços, como por exemplo em transportes públicos). Alguns exemplos onde podem ser encontradas estas aplicações, são os cartões de crédito (e.g. figura 4.1), cartões de fidelização (que acrescem certas promoções ou acumulam pontos por compras), cartões de sistemas de televisão pagos e cartões de transportes públicos.

Também na indústria das telecomunicações são utilizados estes *Smart Cards*, com especial atenção nos países europeus, em que a abordagem de cartões com *chips* embutidos para o acesso à telefonia móvel foi largamente adoptado. Entre os principais sistemas em que estes cartões são utilizados, estão os telemóveis (com pelo menos um cartão associado à conta dos clientes, identificado univocamente através de um número de telefone atribuído) e os postos de telefone públicos (que funciona por créditos, tal como os cartões de transportes públicos, por exemplo). Mais recentemente, com a introdução do 3G e do 4G para acesso à Internet (serviços de banda larga móvel), também os computadores portáteis começaram a disponibilizar o uso destes cartões para o acesso à Internet.

Outro caso prático onde esta tecnologia é utilizada é em sistemas que necessitam de identificação e autenticação do utilizador, onde actualmente existem as mais diferentes soluções que recorrem a *Smart Cards* para este propósito. Há vários cartões deste tipo presentes no dia-a-dia das pessoas, tais como passaportes digitais/cartões de cidadão, cartões de acesso a edifícios (restringindo certas área a utilizadores válidos apenas, com o controlo de entradas e saídas), cartões de saúde, cartas de condução e sistemas de voto electrónico.

Além das funcionalidades referidas, que necessitam de leitores específicos, alguns *Smart*

Cards também podem ser utilizados como *tokens* de segurança em computadores pessoais (e outros dispositivos semelhantes), para autenticação por dois factores ou como alternativa para a inserção de códigos/palavras-passe — quer para operações locais como para operações pela Internet. Deste modo, é possível recorrer a esta tecnologia para controlar o acesso a PCs (sendo necessário apresentar o cartão para iniciar a sessão de utilizador, por exemplo) localmente ou remotamente, pela Internet. O facto de ser necessário um dispositivo físico para aceder a essas funcionalidades (i.e. o *Smart Card*), assegura que um possível atacante não é capaz de se autenticar perante o sistema sem a presença do cartão, o que aumenta consideravelmente o grau de segurança quando comparado à habitual protecção por palavra-passe, apenas. Estes cartões podem também ser utilizados para a troca de informação digital noutros sistemas de um modo genérico, com garantias de privacidade e integridade dos dados. Isto é concretizado através de técnicas criptográficas desempenhadas no próprio cartão, que podem ser conjugadas com protocolos de comunicação como o *Transport Layer Security* (TLS) para estabelecer comunicações seguras entre as partes intervenientes [15].

4.2 Principais características

Como referido na secção 4.1, há vários padrões disponíveis com especificações para *Smart Cards*, de forma a alcançar a uniformidade e interoperabilidade de soluções desenvolvidas para os mesmos. No entanto, e apesar de existir um esforço de adopção destas especificações, o facto de serem demasiado complexas (ou por serem demasiados dispendiosas) leva a que por vezes sejam adoptadas apenas parte das especificações de um ou vários padrões, com certas alterações e personalizações feitas pelos fabricantes, relativamente às operações e funcionalidades dos *Smart Cards* e no modo de interacção com os mesmos. Por este motivo e de forma a evitar qualquer confusão, os detalhes de funcionamento e as restantes características dos *Smart Cards* aqui descritas seguem o padrão `ISO 7816` (de nome "*Identification cards — Integrated circuit cards with contacts*"), podendo por isso certos detalhes serem diferentes em cartões que não sigam este padrão na sua totalidade.

4.2.1 Características físicas

As principais propriedades físicas que permitem classificar um *Smart Card* são o tipo de *chip* — com ou sem microprocessador — e o meio de transmissão de dados — com ou sem

contacto eléctrico.

Tipos de *Smart Cards*

Os cartões sem microprocessador apenas servem para o propósito de armazenar dados destinados a aplicações específicas, sendo o protocolo de comunicação gerido por um módulo (no cartão) responsável pela lógica de acesso, de forma a assegurar certas propriedades de segurança (podendo nalguns casos até cifrar os dados no cartão, por exemplo, dependendo das instruções pré-programadas nos circuitos do mesmo) [36]. Uma aplicação deste tipo de cartões é na área dos transportes públicos, mais especificamente em certos cartões de autocarro com esta tecnologia. Nestes sistemas, são lidos e actualizados os *tickets* de viagens com o processamento a ser realizado no exterior (i.e. incrementar ou decrescer as viagens disponíveis) mas a informação a ser protegida no cartão com acesso restringido aos leitores dos autocarros, de forma a prevenir contra leituras e escritas não autorizadas. No entanto, uma vez que os cartões sem microprocessador não podem ser realmente considerados "inteligentes" por não serem capazes de realizar operações no cartão, as restantes referências neste documento ao termo "*Smart Card*" são relativas ao caso de *Smart Cards* com microprocessadores.

No centro das atenções desta tecnologia, por serem os cartões mais utilizados actualmente e por apresentarem aplicações práticas mais notáveis, encontram-se os *Smart Cards* que incluem um *chip* com microprocessador. Este tipo de cartões permite executar operações lógicas e aritméticas no próprio cartão, e por vezes contêm coprocessadores criptográficos para acelerar técnicas que utilizem algoritmos criptográficos no *Smart Card*. Por este motivo, também apresentam a vantagem de serem dispositivos configuráveis para diferentes funcionalidades, podendo ser optimizados para uma ou mais aplicações de um modo flexível e posteriormente à sua produção — ao contrário dos cartões sem microprocessador, cujas funcionalidades são estáticas e dependentes da configuração inicial dos dispositivos. Em comparação aos cartões sem microprocessador, a desvantagem evidente é relativa aos custos de produção, uma vez que existe o custo adicional do *chip* embutido. No entanto, as possibilidades de soluções com estes *Smart Cards* e as restantes vantagens que apresentam levam a que estes dispositivos sejam produzidos em massa e, consequentemente, apresentem um custo de produção reduzido, e por esta razão são o tipo de *Smart Cards* mais frequentemente utilizados [3].

Relativamente aos meios de transmissão de dados entre os *Smart Cards* e os leitores de cartões, existem duas formas aceder e utilizar estes cartões: directamente — através de contactos eléctricos — ou por proximidade — via ondas de rádio. Em relação aos primeiros, estes possuem uma superfície dedicada a contactos físicos que é utilizada para fornecer energia ao cartão e transmitir os dados através de sinais eléctricos. Quanto aos *Smart Cards* que funcionam por proximidade, estes possuem uma antena e um modulador responsáveis por receber/transmitir os sinais via ondas de rádio e processar os sinais/gerar a voltagem necessária, respectivamente, representado na figura 4.2. Comparativamente, os cartões que funcionam por proximidade apresentam a vantagem de serem mais simples de utilizar, uma vez que não necessitam de ser introduzidos num leitor específico, com um certo formato e orientação. Por outro lado, apresentam a desvantagem de a fonte de alimentação destes *Smart Cards* ser gerada a partir de ondas electromagnéticas, o que limita a potência energética (e consequentemente as capacidades do microprocessador) [15]. Existem também cartões híbridos, que apresentam as duas interfaces, podendo ser utilizados tanto por contacto directo como por proximidade.

Estrutura interna dos *chips*

Internamente, os *chips* dos *Smart Cards* são constituídos por um microprocessador, um módulo de *Read Only Memory* (ROM), um módulo de *Random Access Memory* (RAM), um de módulo *Electrical Erasable Programmable Read Only Memory* (EEPROM) e, opcionalmente, um Coprocessador Criptográfico (NPU), como exemplificado na figura 4.2.

O microprocessador (CPU) é responsável por desempenhar as funções de cálculo definidas pelas aplicações no cartão, sendo que os *Smart Cards* actuais contêm processadores de 8, 16 ou 32 bits que determinam as operações passíveis de serem executadas — e.g. um cartão de 32 bits é capaz de fazer cálculos com valores do tipo "`Int`", enquanto que um de 16 bits no máximo faz cálculos com "`shorts`" e um de 8 bits apenas com "`bytes`".

O módulo ROM contém aplicações e dados estáticos, isto é, é um tipo de memória que não pode ser reescrita ou apagada posteriormente, sendo por isso a informação armazenada durante o processo de produção dos *chips*. Este é o suporte onde está armazenado o sistema

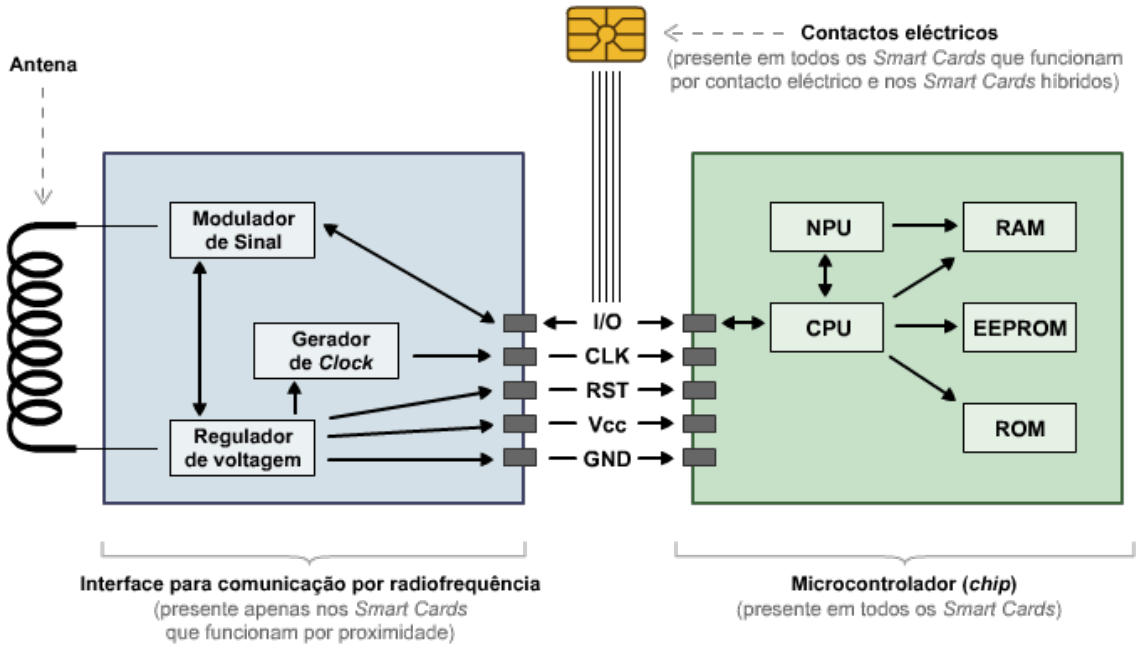

Figura 4.2:
Arquitectura típica de um *Smart Card*. Descrição das abreviaturas na imagem:

- **I/O** — Entrada e Saída de dados/instruções
- **CLK** — Sinal de *Clock* (frequência)
- **RST** — Sinal de *Reset*
- **Vcc** — Voltagem (fonte de alimentação)
- **GND** — Ligação à Terra

As restantes abreviaturas estão descritas em "Estrutura interna dos *chips*" (imagem adaptada de [36])

operativo do *Smart Card* e outras aplicações do fabricante, opcionalmente.

Quanto à memória RAM, o seu propósito e funcionamento é análogo ao da memória RAM presente nos PCs, ou seja, é um tipo de memória volátil que tem como objectivo armazenar informação de estado, como valores utilizados em operações das aplicações e o resultado destas operações, sendo estes valores perdidos quando a fonte de alimentação do cartão é interrompida (por falha de energia ou por terminar o uso do cartão).

Já a memória EEPROM pode ser utilizada para o armazenamento permanente de dados no cartão, pelo que os dados são mantidos em memória mesmo quando o cartão deixa de

receber energia. Caso o fabricante permita a personalização do cartão (por aplicações acrescentadas posteriormente ao fabrico do mesmo), este é o local onde as aplicações desenvolvidas são armazenadas (e respectivos dados persistentes). Por vezes este tipo de memória também é utilizada para dados voláteis, quando estes requerem certas propriedades de atomicidade, uma vez que os dados em memória RAM são susceptíveis a problemas relacionados com falhas de energia.

Além dos módulos descritos, como referido, por vezes existe também um NPU, capaz de realizar ou acelerar algoritmos criptográficos utilizados pelas aplicações dos *Smart Cards*. Dependendo das opções do fabricante, estes coprocessadores podem ser produzidos para acelerar o cálculo de certos algoritmos criptográficos simétricos e/ou assimétricos, como por exemplo o AES e o RSA, respectivamente, que na maioria dos casos seriam impraticáveis de alcançar através de implementações por *software* nos cartões.

4.2.2 Protocolos de comunicação

Tal como acontece na maior parte de arquitecturas que requerem a comunicação entre duas (ou mais) entidades — neste caso, entre o *Smart Card* e o dispositivo de leitura de cartões —, a troca de dados e instruções é feita com recurso a várias camadas de abstracção distintas. Relativamente à arquitectura dos *Smart Cards*, existem três camadas principais [15, 36]:

1. A camada de comunicação específica às aplicações, definida por quem desenvolve as aplicações no cartão e fora do cartão (para interagir com o *Smart Card*) e, por isso, particular a cada aplicação desenvolvida;

2. A camada relativa à estrutura dos dados da comunicação, que define o formato correcto no qual as instruções e dados devem estar organizados para poderem ser trocados entre as entidades participantes;

3. A camada do protocolo de transmissão de dados, responsável por enviar e receber dados entre o cartão e o leitor (que pode ser realizada por blocos ou *byte* a *byte*).

A primeira camada, correspondente à comunicação entre uma aplicação no cartão e a sua contraparte no exterior — i.e. uma aplicação executada num outro dispositivo (conectado a um leitor de *Smart Cards*) que utilize o *Smart Card* para desempenhar certas funções —,

é específica ao sistema em causa. Por este motivo, o protocolo desta camada não pode ser generalizado uma vez que é definido por quem desenvolve as aplicações, de forma a estruturar a comunicação do modo mais conveniente à situação específica. Assim sendo, esta camada tem como propósito modelar uma abstracção da lógica do protocolo de comunicação de dados subjacente, adaptada ao modelo de negócio do sistema.

Em relação às outras duas camadas — a da estrutura de dados e a de transmissão dos mesmos — são descritos de seguida os principais protocolos utilizados, tal como definidos nas partes 4 e 3 do `ISO 7816`, respectivamente.

APDU

A estrutura de dados definida para a comunicação com os *Smart Cards* é denominada de *Application Protocol Data Unit* (APDU). Uma vez que esta comunicação é realizada no sistema de cliente-servidor (i.e. são trocados pedidos e respostas entre um *Smart Card* e o dispositivo ligado ao mesmo), a especificação dos APDUs está dividida em duas categorias:

- **Comandos APDU**, que representam as operações enviadas para o *Smart Card* de forma a serem interpretadas e processadas;
- **Respostas APDU**, que dizem respeito aos resultados e/ou mensagens de estado provenientes do cartão, sendo sempre consequência de um comando APDU enviado anteriormente.

Relativamente aos comandos APDU, estes são compostos por duas partes: o cabeçalho e o corpo do comando (sendo este último de presença opcional), conforme representados na figura 4.3. No cabeçalho do comando, são especificadas a classe ("`CLA`"), a instrução ("`INS`"), o primeiro parâmetro da instrução ("`P1`") e o segundo parâmetro da instrução ("`P2`"). Estes valores são utilizados para definir qual o subconjunto de instruções a que corresponde o comando, qual a acção específica que se pretende realizar e quais os parâmetros específicos que se pretende passar a essa instrução. Já no corpo do comando, é definido o comprimento dos dados ("`Lc`"), os *bytes* correspondentes aos dados passados para o comando e, por fim, o comprimento esperado da resposta ("`Le`").

Quanto às respostas APDU, também estas possuem um corpo da resposta (composto unicamente pelos dados da resposta), seguido de dois *bytes* que representam o estado da

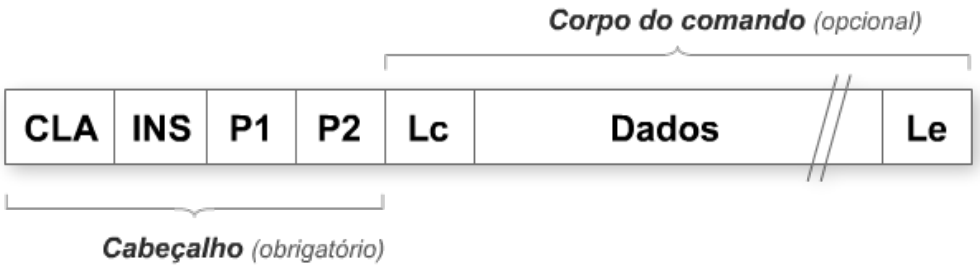

Figura 4.3: Estrutura de um comando APDU

mensagem ("`SW1`" e "`SW2`"), tal como apresentado na figura 4.4. O estado da mensagem é utilizado para indicar se uma operação foi realizada com sucesso ou para representar excepções que tenham ocorrido, podendo também fornecer alguma informação adicional sobre o estado interno de um valor (e.g. para indicar o número restante de tentativas de inserção de uma palavra-passe). Caso ocorram erros durante o processamento de uma instrução, ou caso o comando (a qual o APDU de resposta diz respeito) não espere dados em retorno, é opcional a presença do corpo da resposta, sendo apenas requirida a existência dos *bytes* de estado.

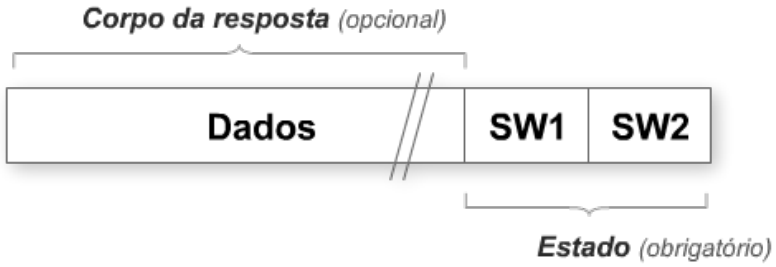

Figura 4.4: Estrutura de uma resposta APDU

À excepção dos dados passados para um comando e dos dados recebidos no corpo de uma resposta, cada uma das restantes variáveis indicadas é constituída por um só *byte*, habitualmente representado no formato hexadecimal (para simplificar a compreensão). Deste modo, o cabeçalho de um comando APDU tem sempre o comprimento fixo de 4 *bytes* e a "cauda"

da resposta tem sempre 2 *bytes*. Tanto num caso como no outro, o comprimento máximo para os dados transmitidos num APDU é de 255 *bytes* e, consequentemente, até 261 *bytes* para um comando APDU e 257 *bytes* para uma resposta [47].

Os valores específicos para estas variáveis dependem de quem desenvolve as aplicações dos *Smart Cards*, podendo estar conforme o especificado no `ISO 7816` (parte 4) ou então serem utilizados valores específicos das aplicações (ou ainda um conjunto das duas soluções). Caso uma aplicação disponha de comandos que estejam conforme as especificações do `ISO 7816`, esses comandos devem ser distinguidos pelo uso do valor `0x80` no campo `CLA` dos APDUs. Também para as respostas APDU, existem normas definidas no `ISO 7816` (parte 4) para os valores `SW1` e `SW2`, como por exemplo o valor `0x9000` (i.e. `SW1=0x90, SW2=0x00`) que representa uma operação bem sucedida, pelo que é encorajada a adopção deste padrão como forma de uniformização das aplicações para *Smart Cards*.

T=0 e T=1

Para a transmissão de dados durante a comunicação entre o *Smart Card* e o dispositivo conectado ao leitor de cartões, estão especificados no `ISO 7816` (parte 3) dois protocolos principais: o "`T=0`" e o "`T=1`". Em ambos os casos, tratam-se de protocolos *half-duplex* assíncronos, ou seja, em que a comunicação é realizada assincronamente e apenas numa direcção de cada vez, sendo a principal diferença entre os dois o modo como a informação é transmitida.

De um modo simples, pode-se afirmar que o protocolo `T=0` faz a transmissão dos dados *byte* a *byte*, enquanto que o protocolo `T=1` o faz por blocos de *bytes*. Uma vez que o primeiro preza pela simplicidade do protocolo e pela eficiência a nível de alocação de memória para a comunicação, tem como consequência uma desvantagem relativa à abstracção das camadas de comunicação: para obter o resultado de uma execução no cartão (isto é, a resposta APDU) é necessário enviar um comando adicional para o *Smart Card*, de forma a que este possa informar o dispositivo leitor quantos *bytes* deve esperar. Uma vez obtido o comprimento da resposta, o dispositivo envia então um novo comando (de nome "`GET RESPONSE`", conforme descrito no `ISO 7816`) com informação do tamanho esperado da resposta (i.e. comprimento indicado no *byte* `Le`) para obter o resultado efectivo. Apesar da desvantagem de ser necessária alguma atenção, por não haver uma perfeita separação das camadas de comunicação,

este protocolo é bastante utilizado, com o sistema de telecomunicações GSM a ser uma das soluções mais notáveis que usa `T=0` [15].

Já o protocolo `T=1`, como concretiza a transmissão por blocos, permite abstrair totalmente a camada de comunicação sem ser necessário qualquer modificação nas camadas superiores. Outra vantagem que esta alternativa apresenta é a possibilidade de utilizar, em certas ocasiões, um só bloco para o envio de um comando e a recepção de uma resposta — isto é, permite reduzir a alocação dinâmica de memória pelo uso de um só *buffer* na troca de APDUs, caso a resposta seja de tamanho igual ou inferior ao comando. Porém, esta solução tem também a inconveniência de ser mais complexa e requerer algum "*overhead*" nos blocos de transmissão, para poder controlar a sequência de comandos e corrigir erros de transmissão.

4.2.3 Propriedades de segurança

Como mencionado no sub-capítulo 4.1.1, o principal objectivo de um *Smart Card* é disponibilizar um meio seguro de armazenar e utilizar dados sensíveis ou confidenciais num dispositivo com a mesma portabilidade de um simples cartão de plástico. Por este motivo, e uma vez que os *Smart Cards* são capazes de executar operações internamente (i.e. sem transmitir os valores utilizados para o exterior do *chip*), a propriedade essencial de um dispositivo destes é a protecção física do *chip*, contra leituras e escritas não autorizadas.

Com este propósito em consideração, o facto de todos os componentes dos *Smart Cards* (i.e. processadores e memórias) estarem contidos num só *chip*, pelo que quaisquer sinais eléctricos trocados entre os componentes são trocados internamente. Esta arquitectura dificulta qualquer tentativa de análise de sinais, uma vez que seria necessário separar fisicamente os elementos do *chip* para poder estudar esta comunicação entre os componentes, e também essa ameaça é tida em conta nas especificações de construção dos cartões, que são produzidos com mecanismos específicos (descritos no `ISO 7816` e analisados em detalhe em [36]). Além da análise de sinais, os *Smart Cards* também são desenvolvidos com consideração sobre ameaças como manipulações de voltagem, leituras ópticas dos dados armazenados, ataques que utilizem raios X e temperatura extremas, entre outras, pelo que o nível de protecção física destes cartões é elevado.

Ainda em relação à análise física dos *Smart Cards*, existem algumas medidas adicionais que acrescentam protecção contra a leitura e adulteração de dados e instruções nos cartões, como por exemplo a presença de certas camadas que envolvem o *chip* dos *Smart Cards* e que destroem por completo a capacidade de operação dos cartões caso sejam modificadas ou removidas (o que evita que os componentes sejam separados e assim examinados em execução) e o uso de técnicas que visam evitar análises de potência (pela uniformização do consumo eléctrico do processador) [15].

Além das propriedades físicas descritas, certos *Smart Cards* facilitam também alguns mecanismos de autenticação bidireccional dos dispositivos e a troca segura de mensagens, através de protocolos criptográficos incluídos nos sistemas operativos dos cartões (sendo que estas funcionalidades dependem também de decisões do fabricante do *Smart Card* em causa). Deste modo, é possível autenticar o dispositivo conectado ao leitor perante o *Smart Card* e vice-versa, e é também possível trocar APDUs com garantias de integridade —através do uso de Message Authentication Codes (MACs) — e privacidade — com recurso a MACs e à cifra DES-CBC (ambos os processos detalhados em [15]).

4.3 A plataforma Java Card

Quando surgiram os primeiros *Smart Cards*, todas as funcionalidades que estes pudessem desempenhar teriam que ser definidas *a priori*, durante o processo de fabrico dos mesmos — i.e. não existia a possibilidade de serem desenvolvidas aplicações para estes cartões após a sua emissão. Para contornar esta questão, foram posteriormente desenvolvidos sistemas operativos para *Smart Cards*, que visam permitir o desenvolvimento de aplicações por terceiros e a personalização dos cartões por este meio.

A plataforma Java Card é constituída por um ambiente de desenvolvimento de aplicações para *Smart Cards* — denominado de *Java Card Development Kit* (JCDK) — e pelo OS Java Card capaz de executar essas mesmas aplicações — que inclui a *Java Card Virtual Machine* (JCVM) e o *Java Card Runtime Environment* (JCRE) (referidos com maior atenção em 4.3.1 e 4.3.2). Deste modo, é possível que um *Smart Card* com esta tecnologia seja personalizado para casos de uso diferentes com *applets*[2] desenvolvidas em Java. A propriedade

[2] *Applets* — nome frequentemente atribuído às aplicações para Java Card

de ser utilizado Java para o código fonte das *applets*, facilita a adopção por quem já tem experiência com a linguagem e permite que qualquer *applet* desenvolvida possa ser executada em *Smart Cards* diferentes (existe interoperabilidade desde que os cartões em causa possuam a mesma versão do OS e disponham das características necessárias, específicas à aplicação), sem a necessidade de alterar o código fonte da aplicação ou de sequer recompilar a mesma, uma vez que as *applets* são executadas na máquina virtual JCVM [15].

Além da facilitar o desenvolvimento de aplicações para estes cartões, a plataforma Java Card também simplifica o processo de comunicação entre os *Smart Cards* e os dispositivos externos, pela inclusão de APIs que auxiliam a leitura e escrita de comandos e respostas APDUs, de acordo com as especificações do `ISO 7816`. No entanto, apesar de o desenvolvimento de *applets* ser feito em Java e possuir APIs úteis para o ambiente em que é executado, o JCRE e a JCVM não possuem todas as funcionalidades e características que estão presentes nas versões habituais da *framework* Java, tais como as versões para computadores pessoais, por exemplo. Estes factores são a consequência lógica de se tratar de uma plataforma adaptada especificamente para a execução em sistemas embebidos de capacidades tão reduzidas, que não têm poder de processamento nem espaço de armazenamento sequer próximos das características dos computadores pessoais. Por este motivo, entre os principais componentes em falta neste subconjunto, ou que diferem da *framework* Java, estão:

- A limitação dos tipos de dados primitivos existentes — não estão disponíveis os tipos primitivos `char`, `long`, `float`, `double` e, nalguns casos, também o tipo `int`;
- A ausência de grande parte dos *packages* e classes da API original — como por exemplo a classe `String`, que não se encontra disponível;
- A falta de suporte para o *multi-threading* — uma vez que o OS Java Card é "monotarefa", apenas existe uma *thread* por cada processo em execução;
- A indisponibilidade de *arrays* multidimensionais — apenas é possível representar e aceder a arrays unidimensionais;
- A presença facultativa de um *garbage collector* — apesar de poder estar (opcionalmente, dependendo das opções do fabricante) implementado, não existe controlo sobre dados alocados em memória que deixem de ser necessários.

Mais recentemente foi especificada uma nova versão de Java Card, mais especificamente

a versão 3.0, que disponibiliza uma *framework* bastante completa, já com suporte para *threads*, inclusão de um *garbage collector*, mais tipos de dados (com todos os tipos primitivos à excepção dos tipos `float` e `double`) e a inclusão de API adicionais [3]. Porém, apesar de estar especificada já há algum tempo, a presença de cartões com esta versão do sistema operativo é praticamente inexistente, pelo que a maioria dos *Smart Cards* utilizam versões de Java Card com as limitações acima descritas.

4.3.1 Processo de desenvolvimento

A plataforma Java Card apresenta uma arquitectura distribuída, dividida em três partes (como representado na figura 4.5): o Java Card SDK (utilizado fora do cartão), a JCVM e o JCRE (ambos no *Smart Card*). O Java Card SDK é composto por ferramentas que facilitam a compilação e conversão do código fonte em Java para *applets*, e inclui a API definida para que as aplicações possam utilizar as funcionalidades dos *Smart Cards*, através de classes Java. A JCVM, tal como o nome indica, representa a máquina virtual onde é executado o código Java convertido, independentemente do cartão específico em que se encontra, sendo por isso uma abstracção lógica das componentes de *hardware* do *Smart Card*. Por fim, o JCRE é responsável por gerir as *applets* e os recursos do cartão, incluindo por exemplo alocações de memória por parte das aplicações e o isolamento dos ambientes de execução das *applets*, e contem as classes da *framework* Java Card com que as *applets* interagem (através da API).

Deste modo, a parte da plataforma relativa ao processo de desenvolvimento é o Java Card SDK, que permite auxiliar a programação, compilação, conversão e execução das *applets* num ambiente simulado, antes de estas serem instaladas e executadas no cartão (sendo este último processo descrito em 4.3.2). Além da API especificada, existem as seguintes ferramentas principais, fornecidas pelo SDK:

- O **JCWDE** ("*Java Card Workstation Development Environment*") — Simulador de Java Card, para execução e depuração de *applets* num ambiente simulado;
- O **CREF** ("*C Reference Implementation of Java Card*") — Emulador de Java Card, semelhante ao JCWDE mas com mais funcionalidades, sendo a principal vantagem o facto de guardar o estado da memória EEPROM entre execuções subsequentes (porém, ao contrário do JCWDE, não dispõe de um modo de depuração);

[3] Conforme descrito em http://java.sun.com/developer/technicalArticles/javacard/javacard3/

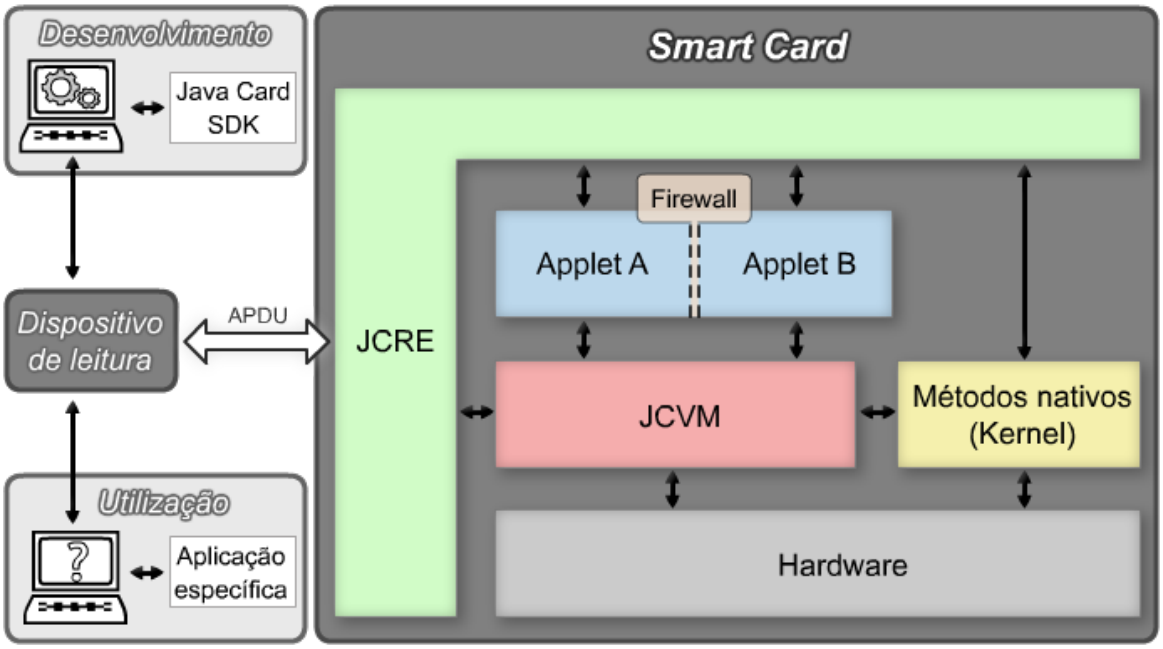

Figura 4.5: Arquitectura da plataforma Java Card

- O ***Converter*** — Após o código Java ter sido compilado com um compilador Java comum[4], os ficheiros resultantes têm de ser convertidos com esta ferramenta para um ficheiro `CAP` ("*Converted Applet*"), de modo a este poder ser instalado num cartão ou simulador;
- O ***ScriptGen*** — Ferramenta que gera APDUs para facilitar a instalação/execução das *applets* em ambientes de simulação;
- O ***APDUtool*** — Interface de comunicação com os emuladores/simuladores de Java Card, sendo que permite enviar e receber APDUs através duma linha de comandos.

Durante a escrita do código fonte das aplicações, é importante implementar as interfaces presentes na API do SDK de modo a que as *applets* resultantes obedeçam às normas do `ISO 7816` e estejam conforme o fluxo de execução definido pela *framework* do Java Card. Assim sendo, a estrutura básica de uma *applet* para Java Card contém obrigatoriamente os métodos `install()` e `proccess()`. Estes dois métodos são invocados pelo JCRE durante a instalação da *applet* e após ser enviado um comando APDU para a *applet*, respectivamente. Além destes, há também outros dois métodos de implementação opcional, mais especificamente o

[4]Apesar de existirem algumas restrições quanto à versão do *Java Development Kit* (JDK) utilizada, dependendo da versão de Java Card em causa.

método `select()` — invocado quando a *applet* é seleccionada e que pode ser utilizado para bloquear a selecção da mesma — e o método `deselect()` — que permite saber quando uma *applet* deixa de estar seleccionada e, caso necessário, efectuar certas operações de término de sessão (e.g. reinicializar um contador, como o de tentativas falhadas de autenticação).

Quanto às características do JCRE, estão ainda presentes as seguintes funcionalidades, disponíveis para uso através da API fornecida [3]:

- Alocação de objectos em memória persistente e em memória volátil — permite definir qual o tipo de memória a utilizar por cada objecto, se em RAM (memória transiente, sendo os dados perdidos em caso de falha de memória, ou no fim de uma sessão da aplicação) ou se em EEPROM (memória persistente, capaz de guardar estados entre sessões da *applet*);

- Execução atómica de operações e noção de transacção — existe a garantia de atomicidade durante a escrita de uma variável de um tipo de dados primitivo, i.e. há a certeza que uma alteração a um valor de um objecto ou classe tem como consequência um novo estado com esse valor guardado ou então é restaurado o valor anterior. Existe ainda a capacidade de definir transacções para escritas sucessivas, em que ou todas as escritas são bem sucedidas ou as variáveis retomam os valores do estado anterior;

- Controlo sobre comunicação/isolamento entre *applets* no cartão — Apesar de as *applets* estarem separadas através de uma "*firewall*" (que atribui, a cada *applet*, um espaço de execução específico), existem mecanismos de partilha de dados e serviços entre *applets* do cartão.

4.3.2 Instalação e configuração de Applets

Tal como referido em 4.2.1, existem vários tipos de memória nos *Smart Cards*, cada um com características e propósitos diferentes. Para o armazenamento de *applets* em cartões com Java Card, o tipo de memória utilizado é a *EEPROM*, enquanto que para o armazenamento do OS (e potencialmente para *applets* dos fabricantes, com a capacidade de aceder a métodos nativos [3]) o suporte utilizado é a memória ROM, que apenas pode ser gravada durante o processo de produção dos cartões.

O JCRE permite que várias *applets* de Java Card estejam instaladas num só cartão e

sejam acedidas arbitrariamente, o que torna estes cartões com Java Card em dispositivos "multifunções" personalizáveis, passíveis de serem usados em vários casos diferentes sem a necessidade de serem reprogramados ou de serem emitidos dispositivos novos (sendo o número permitido de *applets* no cartão dependente da quantidade de memória EEPROM disponível). A instalação de uma *applet* num cartão é transaccional e é feita através de um ficheiro `CAP`, resultante da conversão do código Java compilado, que é transferido para o cartão através de comandos APDU específicos para a instalação.

Uma vez em memória EEPROM, é invocado o método `install()` para criar uma instância da *applet* no cartão, método este que pode receber parâmetros de inicialização para a personalização da instância da *applet*, caso assim seja necessário. Visto que os cartões podem conter várias aplicações em simultâneo, cada *applet* tem de ser univocamente identificada, o que é feito através de um código denominado de *Application Identifier* (AID). Um AID é constituído por um *array* que contém entre 5 a 16 *bytes*[5], e o registo do AID de uma *applet* é também feito na instanciação da mesma, pela invocação do método *register()* que recebe como argumento o próprio AID (que pode ser definido quer estaticamente no código fonte, quer através dos parâmetros de instalação da *applet*).

Contrariamente ao que acontece com aplicações de outras plataformas (e.g. como os computadores pessoais), nestes dispositivos as *applets* não são executadas conforme são utilizadas — uma *applet* é instanciada uma única vez durante a instalação e fica pronta para utilização num estado "seleccionável" durante toda a vida da *applet* no cartão, independentemente de o cartão estar em utilização ou não e de a *applet* estar a ser efectivamente utilizada (i.e. no estado "activo") ou não, tal como representado na figura 4.6. Conforme seja posteriormente necessário, a *applet* é então activada através de um APDU de selecção, definido no `ISO 7816`, que recebe como parâmetro o AID da *applet*. A partir desse momento, todos os comandos APDU passados para o *Smart Card* passam a ser direccionados para a *applet* em questão, até que esta seja explicitamente terminada ou até que a fonte de alimentação do cartão seja interrompida, altura em que a *applet* volta para o estado "seleccionável".

Uma vez que a plataforma Java Card não implica a presença de um *garbage collector* e

[5]Mais especificamente, tal como definido na parte 5 do `ISO 7816`, os AID são constituídos por um *Registered Identifier* (RID) com tamanho fixo de 5 *bytes* e um *Proprietary Application Identifier Extension* (PIX) com tamanho entre 0 e 11 *bytes*.

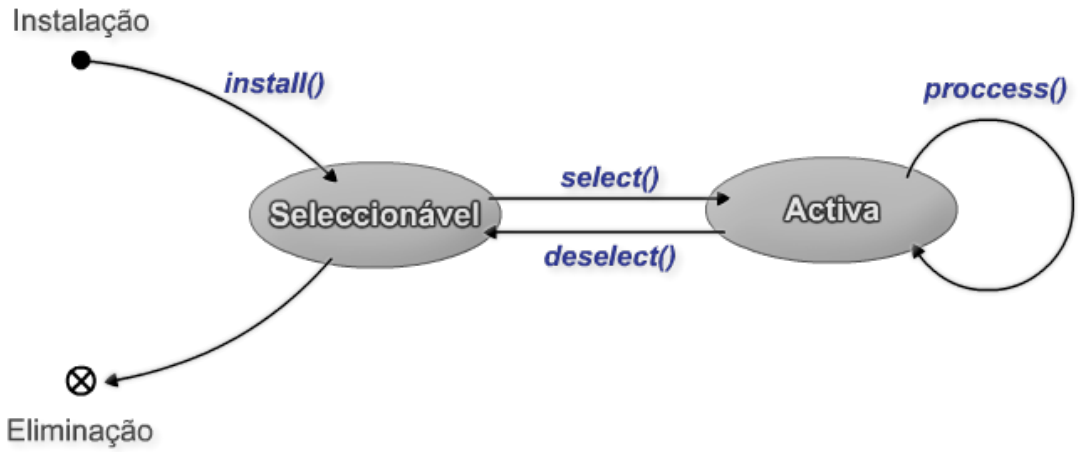

Figura 4.6: Ciclo de vida de uma *applet*

por a memória RAM destes dispositivos ser bastante limitada, o processo recomendado para a instanciação dos objectos persistentes necessários pelas *applets* é a definição dos mesmos durante o método `install()`. Desta forma, existe a garantia que irá haver sempre o espaço de memória alocado para o processamento desses objectos (pois caso contrário, ocorreria um erro durante a instalação e a *applet* não seria instanciada, visto que os objectos são criados durante a instalação).

Por este motivo também, certos objectos que requerem uma configuração inicial — tal como códigos PIN, dimensões de *arrays* (para armazenamento temporário de dados) e especificação/inicialização de algoritmos criptográficos — têm de receber esta configuração através dos parâmetros passados ao método `install()`. Como existe um limite no número de *bytes* transmitidos por um APDU, para objectos que precisem de ser configurados com dados que excedam esse limite é necessário recorrer a outros mecanismos adaptados, sendo criadas operações específicas (com os comandos APDU a serem interpretados no método `proccess()` da *applet*) definidas apenas para o efeito de inicialização, que recebem os dados por blocos (com indicações de comprimento e *offset* dos dados) e constroem o objecto iterativamente.

4.3.3 Considerações de Segurança

Implementar soluções de segurança para *Smart Cards* é diferente de desenvolver aplicações para outras plataformas. O poder de computação destes dispositivos é bastante limitado,

assim como a quantidade de memória disponível, e estes cartões dependem dos leitores de *Smart Cards* para o fornecimento de energia e do sinal de *clock*. Por estes motivos, qualquer aplicação desenvolvida para *Smart Cards* requer uma especial atenção às limitações impostas, visto que muitas vezes a solução passa pelo reaproveitamento de variáveis e objectos para armazenamento temporário de dados em memória persistente, mas isso pode resultar em erros de execução e até falhas de segurança caso a solução não seja implementada cautelosamente.

A plataforma Java Card apresenta várias funcionalidades que visam colmatar estes problemas, como referidas em 4.3.1:

- Isolação dos dados de uma *applet* através de uma *firewall* — objectos e variáveis das *applets* são geridos num ambiente isolado, evitando assim que outras *applets* no mesmo cartão tenham acesso a estes valores;
- Atomicidade e transacções — uma vez que estes dispositivos são alimentados por uma fonte externa (i.e. pelo leitor de cartões), é necessário garantir que o estado da memória persistente não é corrompido caso ocorram erros durante uma escrita ou caso a fonte de energia seja interrompida. Por este motivo, a API do Java Card disponibiliza métodos que garantem a atomicidade de operações de escrita individuais e sequenciais, através de transacções;
- Biblioteca criptográfica — de forma a utilizar o coprocessador criptográfico (quando presente) e simultaneamente evitar implementações erradas de algoritmos criptográficos, a API fornecida contém métodos que permitem utilizar vários algoritmos criptográficos (sendo os algoritmos específicos decididos pelos fabricantes dos cartões).

Além destas características, está igualmente presente uma colecção de classes na API do Java Card, que facilitam o uso de certas funcionalidades de segurança, como o caso do objecto "`OwnerPIN`" que é fornecido para permitir uma implementação correcta e eficiente de controlo de acesso a *applets* através de um código ou palavra-passe. O uso destes objectos é também encorajado de forma a evitar ataques por análise diferencial de consumo energético, visto que no caso de não ser utilizado um `OwnerPIN` poderia levar a uma implementação faltosa capaz de ser explorada através de ataques por força bruta, contornando o limite de tentativas de autenticação [26].

CAPÍTULO 5

A Marca Do Dia Electrónica em *Smartphones*

Tal como referido no capítulo 2, de momento, o serviço MDDE apenas está disponível para utilização em computadores com o OS Windows, uma vez que o envio de mensagens requer a instalação de um *plugin* desenvolvido unicamente para esta plataforma e a validação necessita de executar um componente ActiveX (também exclusivo a este OS). Com as mais recentes tendências de interacção social através da Internet em dispositivos móveis como os *smartphones* (e.g. correio electrónico e redes sociais), tornam-se comuns as soluções de comunicação electrónica para estes aparelhos. Também este serviço encontra vantagens numa possível solução móvel para troca de mensagens com MDDE, considerada a conveniência de estar acessível para qualquer utilizador, em qualquer altura e sem a necessidade de ter um computador por perto.

Assim sendo, neste capítulo é estudado um possível sistema de envio de mensagens MDDE através *smartphones*, com uma prévia análise de requisitos para os objectivos descritos. É proposta a estrutura de uma aplicação de envio com base no sistema operativo Android e de uma *applet* para a plataforma Java Card, com especial atenção para que o resultado seja transcrito numa solução o mais genérica possível (i.e. capaz de ser posteriormente implementado noutro sistema) e potencialmente modular — ou seja, que possam ser acrescentadas outras funcionalidades posteriormente, sem a necessidade de reestruturação ou grandes alterações nos módulos existentes. Por fim, o capítulo é concluído com uma breve análise de segurança, onde são tomados em consideração os potenciais novos riscos, por se tratar de uma aplicação móvel, e referidas quais as medidas adoptadas para combater tais problemas.

5.1 Objectivos e Arquitectura estipulada

Antes de analisar os requisitos e especificar as operações de uma alternativa móvel para o envio de mensagens com MDDE, é necessário especificar quais os objectivos para o sistema a desenvolver e detalhar a arquitectura estipulada para o mesmo, tal como definido inicialmente em conjunto com a equipa da MULTICERT.

5.1.1 Objectivos da solução

Para desenvolver uma solução que permita o envio de mensagens MDDE num ambiente móvel, com funcionalidades idênticas às que estão presentes no *plugin Desktop*, pretende-se alcançar os seguintes objectivos:

- Utilizar um *smartphone* como dispositivo de envio de mensagens com MDDE;
- A solução a desenvolver não deverá interferir com a estrutura actual dos servidores MDDE;
- A composição de mensagens deve ser possível a partir de outras aplicações residentes no dispositivo (mais especificamente, clientes de *e-mail*), não condicionando a escolha do utilizador a uma única interface;
- Armazenamento e utilização das credenciais electrónicas necessárias — i.e. certificados digitais para assinatura dos pedidos de envio — num *Smart Card* no formato *microSD*, de forma a poder ser utilizado num *smartphone*;
- A solução deve ser especificada e implementada com atenção à portabilidade para outras plataformas.

5.1.2 Arquitectura do sistema

Face aos objectivos propostos, estão identificadas quatro entidades físicas intervenientes no sistema: o dispositivo móvel (mais especificamente, o *smartphone*), o *Smart Card* (cartão *microSD*) e os dois servidores SMTP de envio — o *Proxy* MDDE e o servidor SMTP do utilizador, para *e-mails* sem MDDE. Uma vez que não estão previstas quaisquer alterações para o *Proxy* de envio e que o servidor SMTP do utilizador não faz parte do serviço MDDE, os dispositivos onde a solução será aplicada são então o *smartphone* e o *Smart Card*.

Com estes suportes físicos em consideração e para os objectivos indicados, foram identificados três componentes lógicos principais para a solução a desenvolver, representados na figura 5.1: o Serviço de reencaminhamento, o Núcleo MDDE e o *Smart Card*.

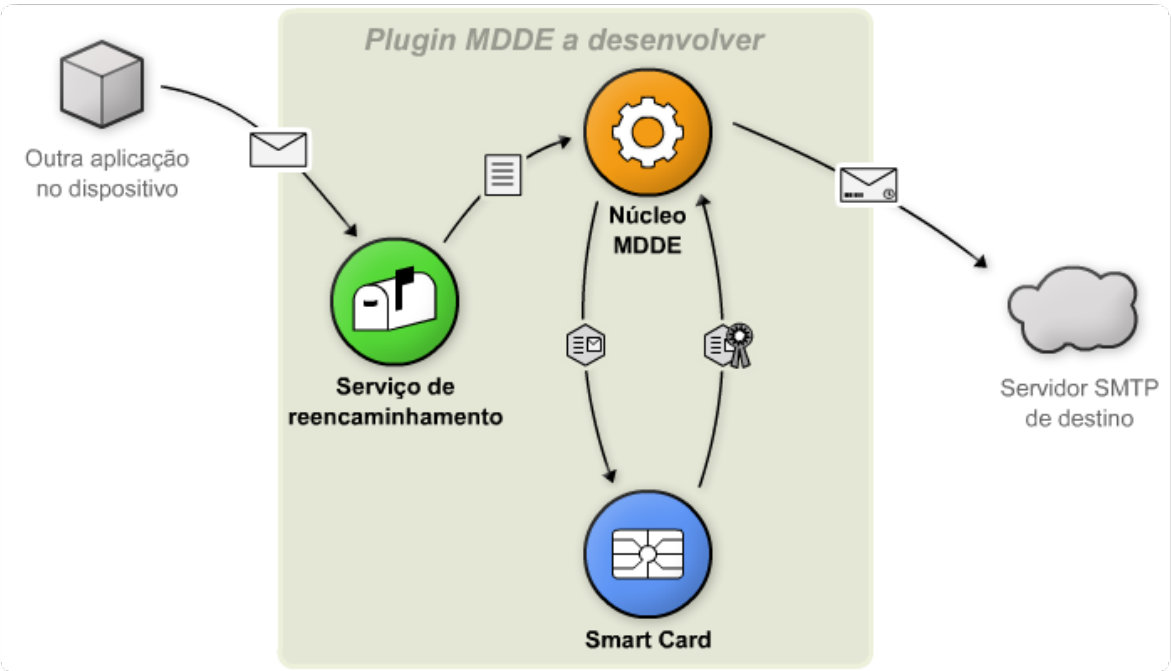

Figura 5.1: Componentes da solução

Serviço de reencaminhamento Módulo necessário para a intercepção de mensagens compostas por outras aplicações. Este componente recebe mensagens de *e-mail* que se pretendem enviar através serviço MDDE, e tem como dever reencaminhar as mesmas para o Núcleo MDDE. Uma vez que precisa de interceptar as mensagens compostas por aplicações do dispositivo, o destino físico para a implementação deste componente é o *smartphone*.

Núcleo MDDE Parte do *plugin* responsável pela estruturação dos pedidos de envio MDDE e pelo envio de mensagens para os servidores MDDE. Estas funções devem ser realizadas após a recepção de uma mensagem de *e-mail* redireccionada pelo Serviço de reencaminhamento, e comunica com o módulo do *Smart Card* durante este processo, para o cálculo da assinatura digital dos pedidos de envio. Após ter sido concluída a preparação de um pedido de envio, cabe a este componente o cargo de enviar o pedido

para o servidor SMTP de destino. Pelos motivos apresentados e tal como o módulo anterior, também o Núcleo MDDE será implementado no *smartphone*.

Smart Card Tendo sido determinado à partida pelo objectivo que define o suporte para as credenciais de envio, este componente é mapeado directamente no cartão *microSD*, sendo portanto o módulo responsável pelo armazenamento das credenciais necessárias e pela realização das assinaturas digitais sobre os pedidos MDDE.

5.2 Análise de requisitos

Um passo essencial na especificação de um qualquer sistema é a análise detalhada dos requisitos para a solução. De seguida, são apresentados os requisitos físicos, funcionais, operacionais, e de segurança da solução, de acordo com os objectivos propostos na secção anterior.

5.2.1 Características dos dispositivos

Para a implementação dos componentes da solução proposta, são necessários dois dispositivos electrónicos: o *smartphone* e o *Smart Card*.

Relativamente ao *smartphone* (com o sistema operativo Android), este não precisa de atributos físicos específicos à excepção do suporte para cartões *microSD* e um qualquer modo de acesso à Internet (como por exemplo, Wi-Fi ou 3G). A nível de espaço de armazenamento, também não são necessárias considerações especiais, uma vez que estes dispositivos são produzidos com memória suficiente para armazenar várias aplicações. Para efeitos de análise de segurança e posterior especificação do sistema, é tomado como pressuposto que o dispositivo móvel não dispõe de acesso *root*, visto que isto iria afectar as características de segurança do OS, como explicado no capítulo 3.

Quanto ao *Smart Card* com formato *microSD* (e com a plataforma Java Card), este tem de ser capaz de realizar as operações criptográficas essenciais ao serviço (mais especificamente, assinaturas digitais) através da API disponibilizada para o efeito, bem como disponibilizar memória suficiente para a chave privada, códigos de acesso e restantes configurações. Idealmente, este dispositivo deve possuir um co-processador criptográfico e a memória flash

(presente no cartão *microSD*, mas externa ao *chip* do *Smart Card*) disponível deve apresentar a capacidade de leitura e escrita de dados, para o armazenamento dos certificados digitais e possivelmente outros dados relativos às configurações da aplicação. Para efeitos de análise de segurança e especificação do sistema, são consideradas as propriedades de segurança — tanto a nível físico dos *chips* utilizados como a nível lógico da plataforma Java Card — detalhadas no capítulo 4.

5.2.2 Requisitos funcionais

Considerando o objectivo de implementar uma aplicação móvel que permita o envio de mensagens MDDE, o sistema deve ser capaz de realizar as funcionalidades que se seguem, agrupadas pelo componente correspondente. São também apresentados certos requisitos opcionais para esses componentes, que serão à partida considerados como funcionalidades a implementar, para efeitos de especificação da solução a desenvolver.

Núcleo MDDE

O núcleo MDDE deve ser capaz de:

- utilizar mensagens de correio electrónico interceptadas pelo serviço de reencaminhamento;
- enviar mensagens através do serviço MDDE;
- enviar mensagens sem o serviço MDDE (ou seja, encaminhar a mensagem original para os seus destinatários);
- permitir a configuração dos servidores de envio (tanto do servidor *proxy* SMTP MDDE como do servidor de correio electrónico do utilizador).

Opcionalmente, este componente poderia também dispor da funcionalidade de composição de mensagens através do próprio *plugin* móvel e iniciar a composição de mensagens a partir de aplicações com conteúdo multimédia, ao invés de utilizar mensagens compostas por clientes de *e-mail* do dispositivo. Como solução complementar, este componente poderia ainda permitir a configuração de modos de envio automáticos (enviar sempre com MDDE ou nunca enviar com este serviço) e dispor de mecanismos de validação das configurações

especificadas, de modo a permitir a análise e correcção de problemas durante o envio de mensagens. Além destas características opcionais, poderia também ser facultada a opção de actualizar a aplicação através deste componente, com o objectivo de corrigir eventuais erros de implementação ou disponibilizar funcionalidades adicionais.

Serviço de reencaminhamento

O serviço de reencaminhamento a desenvolver deve ser capaz de:

- Interceptar mensagens de correio electrónico compostas noutras aplicações (e destinadas ao serviço em questão);
- Reencaminhar as mensagens interceptadas para o Núcleo MDDE.

Além destas funcionalidades, o serviço de reencaminhamento pode também permitir a automatização do seu modo de funcionamento, de forma a iniciar automaticamente o serviço de intercepção de mensagens assim que o dispositivo móvel é ligado.

Smart Card

O componente para o *Smart Card* deve ser capaz de:

- realizar assinaturas digitais sobre conteúdo arbitrário transmitido pelo Núcleo MDDE (mais especificamente para assinar os pedidos de envio) através de comandos APDUs;
- permitir definir a chave privada utilizada para assinatura dos dados.

5.2.3 Requisitos operacionais

De seguida são descritos os requisitos operacionais para uma aplicação móvel de envio de mensagens MDDE, de forma a que as funcionalidades referidas sejam correctamente desempenhadas.

Núcleo MDDE

Para as operações definidas para este componente, estão estabelecidos os seguintes requisitos operacionais:

- o envio de mensagens MDDE deve ser totalmente independente do conteúdo do *e-mail* original — isto é, se tem ou não anexos, se utiliza uma codificação de texto diferente, etc;

- o utilizador deve ser informado do estado final da mensagem que pretende enviar (i.e. se foi ou não efectivamente enviada para os destinatários).

Em relação à implementação dos requisitos funcionais opcionais definidos anteriormente para este componente (em 5.2.2), estão também especificados os seguintes requisitos operacionais:

- para a composição de mensagens a partir da aplicação móvel, deve ser possível anexar ficheiros e definir vários endereços de destino (qualquer que seja o modo de envio — normal, CC ou BCC);
- caso ocorra um erro durante o envio de uma mensagem e estejam implementados mecanismos para a validação das definições, deve ser indicada a causa do erro — se está relacionado com a conectividade à Internet, se com/quais configurações de envio ou se o problema teve origem nas credenciais utilizadas;
- se for implementada a opção de actualização do sistema, esta deve ser possível de realizar através de uma loja de aplicações (e.g. Android Market) ou através de um servidor dedicado.

Serviço de reencaminhamento

Os requisitos operacionais para este componente, são:

- o serviço deve rejeitar mensagens de correio electrónico que não contenham o endereço de remetente esperado ou que não contenham qualquer endereço de destinatário;
- o serviço deve ser capaz de interceptar qualquer *e-mail* válido (segundo a condição anterior) que seja destinado para o mesmo, independente do conteúdo ou formato da mensagem em específico;
- deve ser apresentada uma notificação visual do estado do serviço quando este se encontra em execução;
- o serviço deve ser capaz de gerir e distinguir várias mensagens interceptadas em simultâneo, quer por um só cliente de *e-mail* quer por clientes de correio electrónico diferentes.

Smart Card

Para o módulo de assinatura, estão definidos os seguintes requisitos operacionais:

- deve ser possível a actualização da chave privada no *Smart Card*, posteriormente à sua inicialização;
- o componente deve ser capaz de gerir dados que excedam o comprimento máximo permitido para transporte de APDUs quando necessário (e.g. dados para a definição da chave privada);
- quando uma operação não poder ser executada ou quando ocorrerem erros durante a mesma, deve ser retornada uma resposta APDU com valores para os bytes de estado (nomeadamente "`SW1`" e "`SW2`") que detalhem o sucedido, de forma a ser possível detectar a origem do problema.

5.2.4 Requisitos de segurança

Sendo a Marca Do Dia Electrónica um serviço que pode ser usado para a transmissão de informação sensível — inclusive com validade jurídica —, a protecção das credenciais utilizadas (para a assinatura dos pedidos) é um factor essencial, uma vez que na eventualidade destas serem comprometidas e o serviço utilizado por terceiros, o utilizador pode ser responsabilizado por mensagens enviadas com a sua autoria. Assim sendo, torna-se necessário estabelecer os requisitos de segurança para a solução a implementar, de forma a evitar situações destas e apresentar características de segurança idênticas às fornecidas pelo *plugin Desktop*. Seguem-se as propriedades de segurança requeridas para cada um dos componentes da solução.

Núcleo MDDE

Uma vez que o sistema operativo designado para a solução prevista é o OS Android, é necessário ter em consideração as permissões de execução requeridas para o funcionamento do *plugin*. Desta forma e dados os objectivos indicados, para este componente serão necessárias as seguintes permissões (presentes na biblioteca "`android.permission`" da API do Android):

- acesso à Internet (permissão "`INTERNET`");
- informações sobre o estado da ligação (permissão "`ACCESS_NETWORK_STATE`").

Para além destas permissões, o componente deve ainda armazenar em memória interna (numa área de acesso restrito) as configurações de funcionamento — i.e. servidores de envio,

conta de *e-mail*, etc — de forma a ficarem inacessíveis a qualquer outra aplicação e a evitar alterações acidentais ou não autorizadas.

Serviço de reencaminhamento

Relativamente ao serviço de reencaminhamento, este deve obedecer aos seguintes requisitos de segurança:

- prevenir contra tentativas de envio de mensagens, por aplicações de terceiros, que não sejam explicitamente autorizadas pelo utilizador;
- assegurar que a mensagem interceptada é transmitida exclusivamente para o componente do Núcleo MDDE (e.g. através de *Intents* explícitos).

Para suportar a funcionalidade opcional de arranque automático deste componente, a aplicação a implementar deve ainda conter a permissão adicional "`RECEIVE_BOOT_COMPLETED`", que é utilizada para tomar conhecimento de quando o OS do dispositivo móvel é iniciado.

Smart Card

Dada a natureza sensível dos dados que o componente do *Smart Card* irá armazenar, este deve ter acesso condicionado às suas funcionalidades, nomeadamente através de:

- um código PIN ("*Personal Identification Number*") para realizar a assinatura digital de dados, com 3 tentativas de inserção e comprimento mínimo de 4 dígitos;
- um código PUK ("*Personal Unblocking Code*") para desbloquear um código PIN bloqueado, com 10 tentativas de inserção e comprimento mínimo de 8 dígitos;
- um código SLK ("*Secret Loader Key*") para inicializar ou redefinir a chave privada no cartão, com 5 tentativas de inserção e comprimento mínimo de 8 dígitos.

Além da existência destas credenciais de acesso, estão definidos os seguintes requisitos de segurança para este componente:

- caso o código PIN seja errado 3 vezes consecutivas e o código PUK também exceda o limite definido, o componente deve bloquear permanentemente;

- caso o código SLK seja bloqueado, o componente continua a poder ser utilizado mas deixa de ser possível a actualização da chave privada;
- tanto a chave privada guardada como os códigos de acesso (PIN, PUK e SLK), nunca são transmitidos para o exterior do cartão, sendo todas as operações correspondentes que necessitem das mesmas — assinatura, verificações e redefinições de credenciais — realizadas no próprio *chip* do cartão (de forma a que a informação não possa ser consultada ou alterada, quer via software que por hardware, como referido no capítulo 4);
- controlo de atomicidade nas operações do componente que estejam relacionadas com a inicialização ou actualização de credenciais.

5.3 Especificação do sistema

Através das funcionalidades e os requisitos referidos na secção anterior, foi deduzido o funcionamento geral da solução a implementar, estando representado na figura 5.2 o fluxo de dados e actividades a desempenhar durante o envio de uma mensagem através do *plugin* móvel MDDE. Com esta organização em mente, para a especificação do sistema, foi definido que a nível estrutural esta solução seria composta por duas aplicações distintas (descritas de seguida): uma aplicação Android para o *smartphone*, que engloba o Serviço de reencaminhamento e o Núcleo MDDE apresentados anteriormente, e uma *applet* Java Card, que corresponde ao componente de *Smart Card*.

5.3.1 Aplicação no *Smartphone*

A aplicação que se pretende implementar no *smartphone* representa a componente da solução que corresponde ao papel desempenhado pelo *plugin Desktop* MDDE. Esta é portanto a parte visível ao utilizador, responsável pela maior parte do funcionamento do sistema de envio no ambiente móvel, desde a intercepção de mensagens de correio electrónico até à produção e envio de pedidos para o servidor *proxy* MDDE.

Para a intercepção de mensagens através do *plugin* móvel, função da responsabilidade do Serviço de reencaminhamento, a aplicação deve recorrer a uma solução idêntica à utilizada pelo *plugin* existente — a execução de um servidor SMTP local, com a função de interceptar

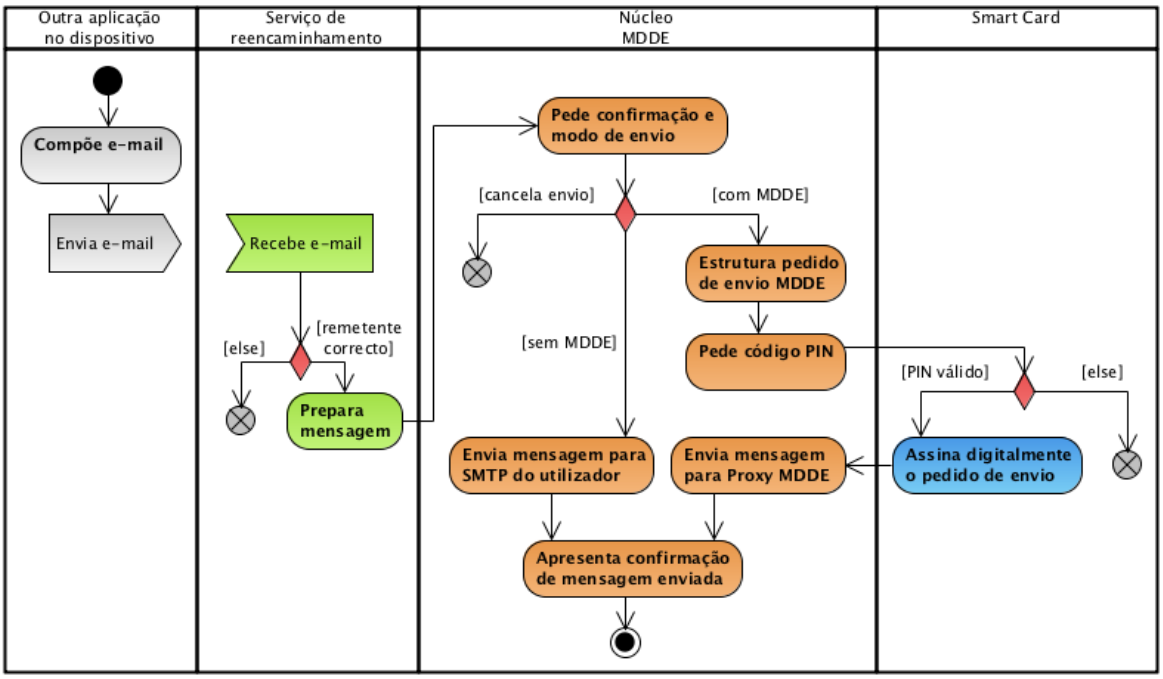

Figura 5.2: Funcionamento geral do *plugin* móvel MDDE

mensagens enviadas por clientes de *e-mail* devidamente configurados. Como se pretende desenvolver a aplicação para o OS Android, isto pode ser concretizado através de um *Service*, executado em segundo plano, que inicia e controla uma *thread* com o servidor SMTP local (que pode ser implementado com o uso de *sockets* em Java). Assim que o servidor local recebe uma mensagem, o *Service* deve reencaminhá-la para o Núcleo MDDE, que por sua vez deve iniciar uma *Activity* para pedir ao utilizador a confirmação da intenção de envio, de forma a evitar tentativas de envio maliciosas.

De modo a permitir a composição (opcional) de mensagens através do próprio *plugin*, deve ser então disponibilizada uma *Activity* com interface gráfica para a composição da mensagem, onde podem ser definidos os destinatários, assunto, corpo da mensagem e anexos, de forma semelhante a um cliente de *e-mail* (para envio apenas). Relativamente à composição de mensagens a partir de outras aplicações, para ser possível iniciar a composição com uma imagem em anexo seleccionada directamente a partir da galeria, por exemplo, isto pode ser conseguido através do uso de *Intent Filters* que capturem *Intents* de envio de conteúdo MIME, tal como exemplificado na figura 5.3.

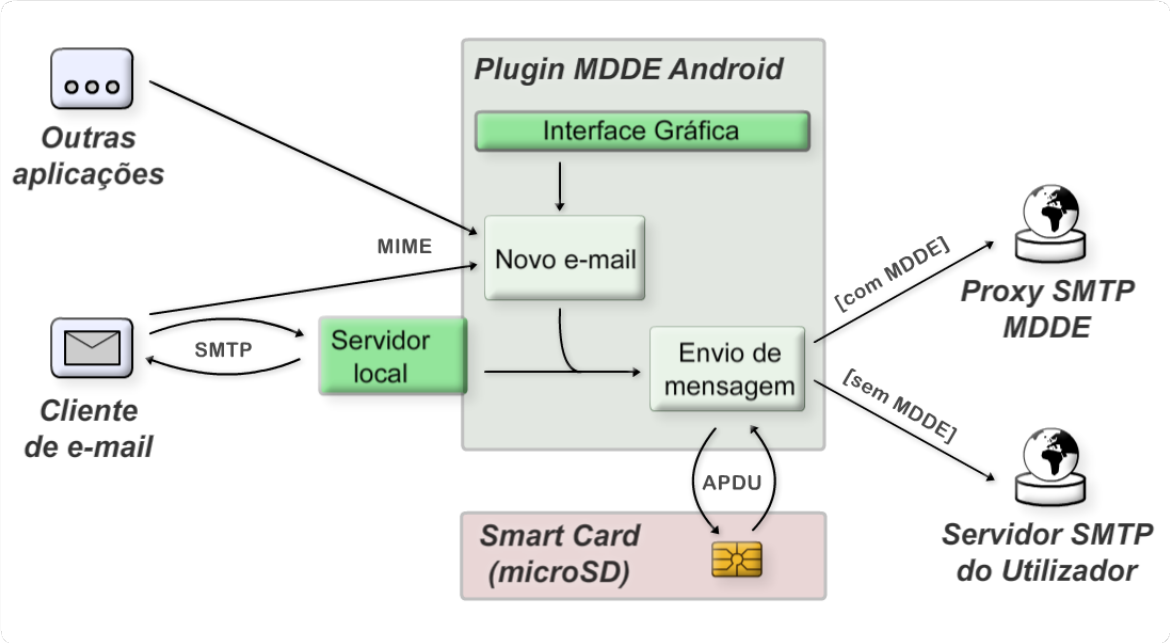

Figura 5.3: Interacção com o *plugin* MDDE para Android

Durante a assinatura dos pedidos de envio das mensagens, a comunicação entre a aplicação no *smartphone* e a *applet* no cartão é feita por comandos APDU. Como se pretende obter uma solução o mais genérica possível, para esta comunicação entre os dois dispositivos deve ser desenvolvido um módulo no *plugin* que permita obter a abstracção do protocolo utilizado (e, simultaneamente, a abstracção de qual o cartão em específico a que está conectado), pela separação lógica entre a camada de negócio e o protocolo de comunicação.

Uma vez feita a assinatura do pedido de envio, que requer a autenticação através do código PIN (inserido no *plugin* móvel mas verificado no cartão), a mensagem composta é enviada com ou sem MDDE, conforme a opção apresentada ao utilizador (através de GUI), ou especificada nas configurações — conforme a opção para o modo predefinido de envio seja de utilizar sempre o serviço MDDE ou nunca enviar por MDDE.

Para a implementação dos mecanismos de validação das configurações definidas, deve então ser realizada uma ligação de teste aos servidores SMTP (i.e. ao *Proxy* MDDE e ao servidor de *e-mail* do utilizador) e simulado o envio de uma mensagem — que é cancelado de seguida —, de modo a verificar a disponibilidade dos servidores e confirmar que estes tratam-se, de facto, de servidores SMTP. A validação das configurações deve também apresentar a

opção de testar as credenciais de assinatura, sendo para isto requirida a inserção do código PIN. Para testar as credenciais, o procedimento a seguir será a realização de uma assinatura digital sobre dados arbitrários e a posterior verificação face à chave pública presente no certificado do utilizador, bem como a validação da cadeia de certificados (desde o certificado do utilizador até ao certificado raíz, potencialmente da CA correspondente), caso esta exista.

5.3.2 *Applet* de assinatura para o Java Card

A *applet* a desenvolver é necessária para realizar a assinatura dos pedidos MDDE (gerados pelo *plugin* móvel), pelo que actua como um módulo de segurança amovível, requerido para utilização do sistema. Em caso de roubo ou extravio do cartão, o utilizador fica impossibilitado de enviar mensagens com MDDE, uma vez que a chave privada utilizada na assinatura está fisicamente armazenada no *chip* do *Smart Card*, pelo que nesta situação o utilizador terá de pedir que sejam revogadas as suas credenciais e geradas umas credenciais novas. Para prevenir a utilização indevida por terceiros, o uso da *applet* para a assinatura digital requer que seja verificado o código PIN, antes de qualquer operação que utilize as credenciais armazenadas.

Com isto em consideração e dados os requisitos apresentados, estão especificados na figura 5.4 o ciclo de vida da *applet* a desenvolver e os possíveis estados em que esta se encontra, bem como as principais funcionalidades que a *applet* deve ser capaz de desempenhar (e respectivas restrições de uso).

Instalação O estado de instalação corresponde à fase de instanciação e registo da *applet* no cartão (com a plataforma Java Card). Neste processo, são inicializados os códigos de acesso da *applet* através dos parâmetros de instalação, nomeadamente o código PIN, PUK e SLK. É também alocada a memória necessária para que possam ser realizadas as operações de assinatura e actualização da chave privada, uma vez que estas duas operações implicam a transmissão e armazenamento temporário de dados de dimensão considerável, que podem implicar a divisão dos comandos APDU em vários sub-comandos (tal como descrito em 4.3.2).

Estado seleccionável Corresponde ao estado em que a *applet* já foi instalada no cartão e pode ser seleccionada para utilização. Como representado na figura 5.4, a *applet* transita ou

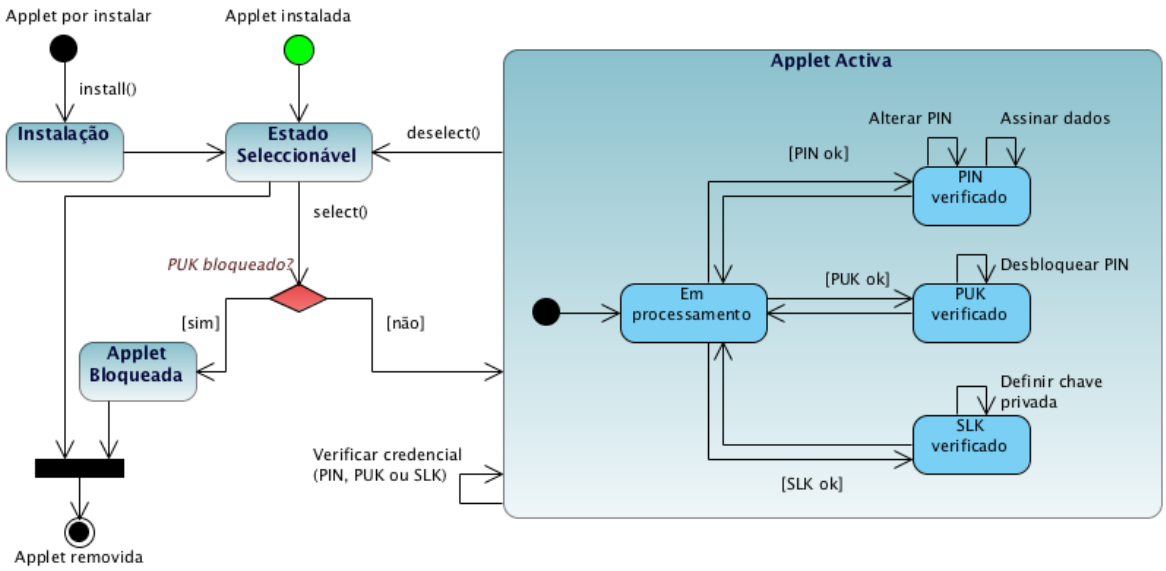

Figura 5.4: Modo de funcionamento da *applet* de assinatura

para o estado "bloqueada" ou para "activa", sendo a condição que determina esta transição o facto de o código PUK estar ou não bloqueado — caso o código PIN tenha excedido o limite máximo de tentativas consecutivas de verificação e o mesmo tenha acontecido com o código PUK, a aplicação não pode ser seleccionada para o estado "activa", sendo bloqueada definitivamente.

Applet bloqueada A partir deste estado, a *applet* deixa de poder realizar qualquer operação, como modo de prevenção contra utilizações não autorizadas (por tentativas de inserção de códigos PIN e PUK errados). Desta forma, o único modo de o *Smart Card* poder voltar a ser utilizado é pela reinstalação da *applet* (o que pode também implicar a emissão de novas credenciais de utilizador, caso não exista outra cópia da chave privada).

Applet activa Representa o estado em que a *applet* é efectivamente seleccionada. Em qualquer sub-estado desta fase activa da *applet*, é possível verificar os códigos PIN, PUK e SLK, que restringem o acesso às restantes operações da aplicação. Uma vez verificada uma credencial, a *applet* transita do sub-estado "em processamento" para o sub-estado correspondente à credencial autenticada[1]: com o PIN verificado podem ser assinados dados e alterado o

[1]Nota: é possível que a *applet* transite para um conjunto de sub-estados de autenticação, em que pode realizar as operações correspondentes às credenciais verificadas — a título de exemplo, se o PIN e o SLK estiverem validados, pode ser alterado o PIN, assinar dados e/ou definir a chave privada.

PIN; com o PUK validado é possível desbloquear o PIN (apenas se este tiver sido bloqueado, em primeiro lugar); com o SLK verificado é possível definir ou actualizar a chave privada.

5.4 Análise de segurança

Comparativamente à solução existente para computadores pessoais, de um ponto de vista global pode-se constatar que o racional adoptado para o desenho do *plugin* móvel é exactamente o mesmo que o apresentado pela versão para Desktop, e uma vez que não há qualquer alteração em relação aos restantes componentes do serviço — i.e. servidores *Proxy*, servidores MDDE e servidor de data e hora — torna-se apenas necessário fazer uma análise de segurança face às diferenças que resultam da mudança de plataforma e de ambiente de execução do *plugin*.

Assim sendo, como a solução proposta é destinada à utilização do *plugin* num ambiente móvel, é evidente a necessidade de considerar novos riscos que surgem das diferenças referidas, ou que não são tão significativos como quando considerada a utilização do *plugin* em computadores pessoais. Com isto em consideração, os principais recursos identificados que podem representar alvos de interesse para ataques informáticos, são (por ordem decrescente de importância):

1. Chave privada, utilizada para a assinatura dos pedidos;
2. Credenciais de envio de *e-mail* sem MDDE;
3. Códigos de acesso da *applet* (PIN, PUK e SLK);
4. Servidor local SMTP;
5. Configurações da aplicação;
6. Certificado digital do utilizador.

Considerados os recursos acima referidos, são de seguida analisadas as ameaças e possíveis ataques de maior impacto, que possam comprometer a segurança da aplicação especificada, bem como indicadas as contra-medidas existentes na solução definida (por ordem sequencial dos recursos identificados).

Exposição da chave privada Esta ameaça apresenta um grau de severidade elevado, uma vez que caso um atacante consiga obter a chave privada utilizada para a assinatura digital dos pedidos MDDE, é possível enviar mensagens MDDE em nome do utilizador, pois não existe qualquer outra restrição ou meio de verificar a autenticidade dos pedidos gerados. Na eventualidade de tal acontecer, será necessário a revogação (por parte da CA correspondente) das credenciais do utilizador. Caso o utilizador também utilize as mesmas credenciais para outros serviços/funcionalidades, como por exemplo cifrar *e-mails*, esta ameaça é de ainda maior gravidade pois também estes são comprometidos.
Contra-medidas: Armazenamento da chave privada na *applet*, pelo uso de objectos específicos para o efeito (disponibilizados pela API do Java Card) e sem métodos definidos para a extrair do *Smart Card*.

Eliminação ou substituição da chave privada Um ataque deste género visa impedir o correcto funcionamento do *plugin*, uma vez que sem a chave privada definida, ou com uma chave privada errada, o utilizador fica incapaz de enviar mensagens pelo serviço MDDE. No entanto, caso exista uma cópia de segurança da chave privada, é possível reestabelecer a mesma no cartão, pelo uso do código SLK.
Contra-medidas: Restrição de modificações da chave privada, sendo requerido o código SLK para eliminar ou alterar o conteúdo da chave privada. Tal como acontece com os restantes códigos de acesso, para o armazenamento e verificação do SLK existem objectos específicos (mais concretamente, o `OwnerPIN`), que devem ser utilizados para evitar implementações faltosas do controlo de acesso.

Exposição das credenciais utilizadas para envio sem MDDE No caso de ser desenvolvida uma solução com a opção de envio através do *plugin* móvel, é necessário que o mesmo tenha acesso às credenciais usadas para o envio de mensagens de correio electrónico sem MDDE. Se um atacante obtiver acesso às mesmas, compromete o *e-mail* do utilizador e pode enviar mensagens com a sua identidade (apesar de não afectar directamente o serviço MDDE).
Contra-medidas: Armazenamento das credenciais de envio no espaço de memória interna reservado para a aplicação, pelo sistema operativo. Com esta medida, e dadas as propriedades da *framework* Android, previne-se a leitura por aplicações maliciosas no *smartphone*

(à excepção de dispositivos que disponham de acesso *root*, visto que com essa permissão a medida de segurança referida pode ser contornada).

Exposição dos códigos de acesso Se um atacante tomar conhecimento dos códigos PIN/PUK e tiver o cartão em sua posse, pode utilizá-lo para enviar mensagens com MDDE. Isto pode acontecer caso um utilizador não seja cuidadoso em manter os códigos em segredo, ou estes sejam obtidos através de técnicas como "*eavesdropping*" ou engenharia social, e o *Smart Card* seja furtado. Em relação ao código SLK, este apenas deve ser mantido na posse da autoridade responsável pela definição/actualização das credenciais, pelo que é improvável a exposição deste código de acesso. De qualquer modo, os riscos que podem advir se tal situação ocorrer são os identificados no parágrafo correspondente à eliminação ou substituição da chave privada.

Contra-medidas: O campo de texto utilizado para a introdução do PIN e do PUK deve ser específico para o efeito (i.e. campo de palavra-passe, com os caracteres ocultos ou representados por asteriscos), e o SLK não deve ser do conhecimento do utilizador.

Cálculo dos códigos de acesso Uma vez que para a solução estão previstos códigos de acesso numéricos, existe a possibilidade de um atacante realizar tentativas de autenticação por força-bruta (i.e. calcular todas as combinações possíveis com n dígitos e testar os códigos gerados). Além desta ameaça que resulta do cálculo de combinações, a existência de relações ou semelhanças entre os códigos pode facilitar o cálculo dos mesmos, ou seja, se existir alguma ligação entre os códigos e por isso for possível descobrir um código através do conhecimento de outro.

Contra-medidas: Os comprimentos mínimos dos códigos de acesso são relativamente elevados, considerados os limites máximos de tentativas consecutivas de inserção erradas por cada código. Os códigos são também definidos de forma independente, pelo que não deve ser possível obter qualquer associação lógica entre os mesmos. É no entanto importante compreender que, caso um código PUK seja comprometido, o código PIN pode ser facilmente modificado (pelo bloqueio e consecutiva alteração do mesmo), e desta forma utilizada a *applet* de assinatura.

Negação de serviço (DOS) do servidor local SMTP Sendo executado no *smartphone* e acessível por qualquer aplicação, o servidor local SMTP pode ser um alvo susceptível a ataques de negação de serviço, em que um atacante impede o serviço de funcionar correctamente com o objectivo de não permitir que o utilizador use o serviço para o seu propósito efectivo de interceptar as mensagens de *e-mail*.
Contra-medidas: Não está prevista qualquer contra-medida para combater esta ameaça em específico, uma vez que para a plataforma considerada e com a solução concebida é praticamente impossível distinguir a origem dos pedidos locais (ou seja, se estes provêm de uma aplicação autorizada ou não), sendo esta uma vulnerabilidade conhecida.

Personificação do servidor local SMTP Quando combinado com a ameaça definida acima, se for executado os ataque por DOS de tal intensidade que force a terminação do processo do servidor local, um atacante pode conceber um servidor SMTP idêntico — executado na mesma porta do servidor legítimo e que apresente a mesma GUI de confirmação de envio ao utilizador — de forma a interceptar as mensagens a enviar e/ou a obter o código PIN da *applet*. Também é possível que uma aplicação maliciosa explore esta vulnerabilidade caso tenha a permissão de terminar os processos em execução em plano de fundo (permissão "`KILL_BACKGROUND_PROCESSES`").
Contra-medidas: Em relação à primeira ameaça, não existe uma solução evidente para o problema, apesar de que o utilizador será informado (por um "*popup*" apresentado pelo sistema operativo) do término do servidor local legítimo, podendo por isso o ataque ser de certa forma diagnosticado. Para a segunda ameaça, a única alternativa viável é a execução do serviço em "*foreground*", ou seja, executar o servidor com um nível de prioridade superior às aplicações executadas em plano de fundo. Desta forma, a permissão garantida não seria suficiente para forçar a saída de um processo activo, pelo que os únicos mecanismos de terminar a execução do servidor seriam ou através do painel de controlo das aplicações (i.e. manualmente, pelo utilizador) ou através de uma aplicação com acesso *root*.

Intermediário no envio de mensagens Caso um atacante consiga modificar as configurações da aplicação, é possível que este altere propositadamente os servidores de envio para negar a prestação do serviço ou para interceptar o conteúdo das mensagens enviadas. Este ataque pode ainda ser utilizado em conjunto com a personificação do servidor local, caso o

atacante consiga alterar a configuração relativa à porta do servidor (e sendo também concebido um servidor malicioso, idêntico ao original).
CONTRA-MEDIDAS: De modo a mitigar esta situação, todas as configurações da aplicação são armazenadas no espaço de memória dedicado exclusivamente à aplicação, tal como referido na ameaça que diz respeito à exposição das credenciais utilizadas para envio sem MDDE. Assim, partindo do pressuposto que o dispositivo onde o *plugin* é executado não dispõe de acesso *root*, não existe a possibilidade de um atacante ler ou modificar esta informação.

Divulgação dos dados pessoais do utilizador Uma vez que o certificado pessoal do utilizador o identifica pelos dados pessoais caracterizados no mesmo — tais como o nome e a morada —, uma aplicação mal intencionada pode usar tal informação para associar a identificação do utilizador ao contacto telefónico do dispositivo. Quando conjugado com a capacidade de monitorização da posição de um *smartphone* em tempo real, esta ameaça representa um risco considerável para a privacidade do utilizador.
CONTRA-MEDIDAS: A solução para esta questão passa por utilizar o mesmo suporte de dados previsto para as configurações, ou seja, a área de memória reservada para a aplicação. Também como nas ameaças relativas às configurações da aplicação, a capacidade de explorar esta vulnerabilidade surge, por isso, apenas em dispositivos com acesso *root*.

CAPÍTULO 6

Implementação e Resultados

Uma vez feita a abordagem ao problema de envio de mensagens com MDDE a partir de dispositivos móveis, com a especificação de um *plugin* para *smartphones* com o OS móvel Android, é apresentado neste capítulo o resultado de uma implementação da solução proposta. Assim sendo, são descritas as características dos dispositivos utilizados para teste da solução e quais as ferramentas utilizadas para o desenvolvimento da mesma. É também apresentada a estrutura das aplicações desenvolvidas, nomeadamente do *plugin* para Android e da *applet* para a plataforma Java Card. Por fim, o capítulo é concluído com uma avaliação da implementação desenvolvida bem como de outros resultados complementares.

6.1 Decisões de implementação

Como referido no capítulo 1, o ambiente de desenvolvimento utilizado — i.e. o OS Android — foi estipulado em conjunto com a MULTICERT, assim como os dispositivos de teste disponibilizados. Deste modo, são descritas de seguida as propriedades do *Smartphone* e do *Smart Card* utilizados (para testes da aplicação móvel e da *applet* de assinatura, respectivamente) e referidas as ferramentas utilizadas para o desenvolvimento das aplicações.

6.1.1 Dispositivos utilizados

Para o propósito de testes de implementação do *plugin* MDDE móvel, foram utilizados dois dispositivos: um *Smartphone* Android e um *Smart Card* no formato *microSD*.

Características do *Smartphone*

O dispositivo móvel utilizado durante o processo de desenvolvimento e de teste da aplicação foi o *Smartphone Gigabyte Boston* (mais especificamente, o modelo *GSmart G1305*), representado na figura 6.1. Relativamente às principais propriedades do dispositivo, o *Smartphone* contém de um processador `ARM 11` a 600 MHz, 256 MB de memória RAM, 512 MB de memória interna, entrada para cartões *microSD* e um ecrã *multitouch* de 3,2 polegadas (com uma resolução efectiva de `320x480` pixeis). Já quanto à conectividade, dispõe de uma antena Wi-Fi `802.11 b/g`, Bluetooth 2.0 e ligação 3G por HSDPA (*High-Speed Downlink Packet Access*).

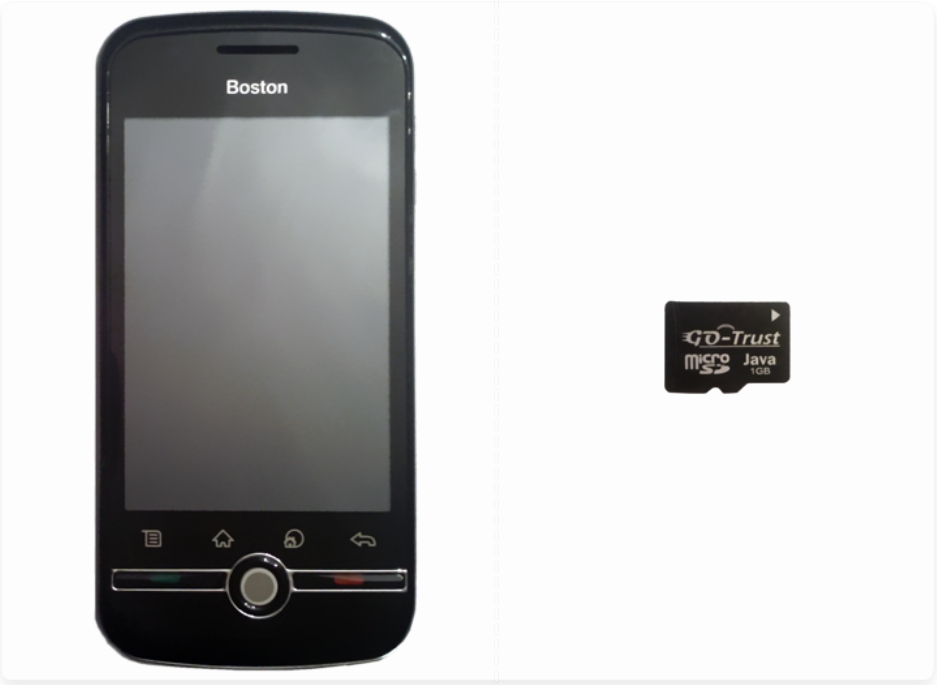

Figura 6.1: Dispositivos de teste utilizados:
Smartphone Gigabyte Boston (à esquerda) e *Smart Card Go-Trust microSD* (à direita)

A versão de Android presente neste dispositivo é a 2.2 (de nome "*Froyo*"), que tal como referido no capítulo 3 corresponde à versão do OS mais utilizada actualmente, com cerca de metade da quota de utilização. Uma vez que o sistema Android é retrocompatível e dada a informação sobre as distribuições actuais do OS (representada na figura 3.2), prevê-se que, com uma implementação direccionada para a versão 2.2 do sistema operativo, a solução poderá ser executada em aproximadamente 85% dos dispositivos com Android.

Características do *Smart Card*

Para o armazenamento e utilização da *applet* de assinatura, foi utilizado um *Smart Card microSD* da fabricante *Go-Trust*, como demonstrado na figura 6.1. Este cartão contém a versão 2.2.1 do Java Card, com a certificação de CC[1] EAL4+ para o *chip* do *Smart Card*, e como principais características físicas apresenta 64 kB de memória EEPROM, 1 kB de memória RAM e a inclusão de um co-processador criptográfico.

Em relação aos algoritmos criptográficos disponibilizados pelo cartão (e relevantes para a solução considerada), o cartão permite armazenar e utilizar, para assinatura digital, chaves *Rivest-Shamir-Adleman* (RSA) até 2048 bits — podendo as chaves privadas estar representadas na forma habitual (i.e. módulo + expoente privado), ou representadas pelos factores necessários para a computação optimizada de assinaturas RSA através do Teorema Chinês dos Restos (CRT) — e ainda chaves para criptografia de curvas elípticas até 192 bits. Quanto às plataformas de execução, este *Smart Card* pode ser utilizado em grande parte dos sistemas operativos móveis e de computadores pessoais, estando disponível para os seguintes OS: Android, BlackBerry, Linux, Symbian, Windows e Windows Mobile.

6.1.2 Ferramentas de desenvolvimento

Tendo em conta que a solução proposta consiste em duas partes, uma para o OS móvel Android e outra para a plataforma Java Card, são também necessárias diferentes ferramentas de desenvolvimento. Assim sendo, para desenvolver a aplicação do *Smartphone* foram utilizadas as ferramentas referidas em 3.2.2 em conjunto com o Eclipse, dada a integração do SDK do Android com este Ambiente Integrado de Desenvolvimento (IDE). O ambiente utilizado para este efeito foi o sistema operativo Mac OSX 10.6, com a versão "Helios" (1.3.2) do Eclipse e a versão 9.0.0 do SDK do Android.

Já para a *applet* de assinatura, o SDK fornecido pela Go-Trust para a instalação e gestão de *applets* no cartão apenas era suportado pelo OS Windows, apesar de os restantes processos

[1] CC — *Common Criteria*, padrão internacional de certificação de sistemas de computação (http://www.commoncriteriaportal.org/)

relativos ao desenvolvimento da aplicação (i.e. compilação e testes) poderem ser executados de forma independente e noutra plataforma. Por este motivo, foi utilizado o sistema operativo Windows XP para a instalação da *applet* no *Smart Card* e testes da mesma no simulador do JCWDE, sendo que a programação/compilação da *applet* foi realizada no mesmo ambiente de desenvolvimento da aplicação Android (i.e. o IDE Eclipse no Mac OSX).

6.2 Estrutura da aplicação Android

A aplicação para Android representa a parte da solução que disponibiliza a interface de utilização e configuração do *plugin*, sendo também responsável por todas as funcionalidades disponibilizadas pela solução, à excepção da operação correspondente à assinatura digital dos pedidos MDDE. Neste sub-capítulo, é descrita a estrutura da aplicação móvel desenvolvida e apresentada a interface gráfica resultante.

6.2.1 Funcionalidades implementadas

O *plugin* móvel MDDE para Android foi desenvolvido com atenção a todos os requisitos funcionais estipulados, à excepção da opção de actualizar a aplicação, pelo que estão presentes na implementação as seguintes funcionalidades:

- envio de mensagens com e sem MDDE;
- possibilidade de composição de mensagens na aplicação, iniciadas interna ou externamente (i.e. a partir de aplicações com conteúdo multimédia);
- servidor SMTP local para a intercepção de mensagens de clientes de *e-mail* externos;
- configuração dos servidores e modos de funcionamento da aplicação;
- automatização dos modos de envio e do arranque do servidor local;
- mecanismos de validação das definições da aplicação.

Para o envio de mensagens, como previsto, a decisão de implementação tomada foi disponibilizar o envio tanto a partir da própria aplicação como através de outros clientes de correio electrónico instalados no dispositivo. Desta forma, permite-se que sejam criadas mensagens na própria aplicação tanto para envio com como sem MDDE, e de igual modo à funcionalidade presente no *plugin Desktop* já existente, também existe a possibilidade de usar um qualquer

cliente de *e-mail* configurável, para que o processo de adaptação à utilização no dispositivo móvel seja minimamente intrusiva. Foi também desenvolvida uma funcionalidade de captura de *Intents* com conteúdo multimédia, de modo a permitir que uma aplicação externa seja capaz de iniciar a composição de uma mensagem no *plugin* móvel com o conteúdo pretendido em anexo.
É descrito, de seguida, o modo como foram abordadas as questões de envio de mensagens para a solução implementada.

Servidor SMTP local

Como previsto e especificado em 5.3.1, foi desenvolvido um servidor SMTP local, responsável por interceptar mensagens de correio electrónico a serem enviadas com o sistema MDDE, de forma análoga à desempenhada pelo *plugin Desktop*. Para que um *e-mail* seja enviado por um qualquer cliente no dispositivo móvel, apenas é necessário alterar as configurações de envio deste mesmo cliente de forma a que o servidor SMTP de envio seja o 127.0.0.1 (i.e. "*localhost*") e a porta do servidor configurada de acordo com a porta onde este estiver a correr (porta 2525, por predefinição). Este servidor SMTP local, por defeito, é executado no arranque do OS, sendo que pode ser parado/recomeçado a qualquer altura de forma a libertar recursos do sistema ou conforme este seja ou não necessário.

Quando é recebido um *e-mail* no servidor SMTP local, este é processado de forma a ser gerado um pedido MDDE, após ser confirmada (por GUI iniciada pelo servidor local) a intenção de enviar a mensagem correspondente via MDDE. Uma vez que a autenticação de um pedido MDDE perante o servidor *proxy* MDDE é baseada em factores inerentes à própria mensagem e relativa estrutura (nomeadamente: cabeçalhos da mensagem original, estrutura do pedido e respectiva assinatura digital), durante o tratamento de mensagens interceptadas pelo servidor SMTP local apenas é analisado o endereço do remetente, que deverá corresponder ao endereço definido nas configurações de envio do *plugin*. Por este motivo, qualquer tentativa de autenticação de um cliente de correio electrónico perante o servidor SMTP local é ignorada, sendo que para o caso de o utilizador pretender enviar o respectivo *e-mail* sem recorrer ao serviço MDDE, são usadas as configurações SMTP para envio sem MDDE definidas no *plugin*.

Envio através da aplicação

Os dispositivos móveis, apesar de cada vez mais se aproximarem das capacidades de processamento de computadores pessoais, continuam a ser dispositivos que necessitam de uma boa gestão de recursos para providenciarem um equilíbrio aceitável entre desempenho e autonomia. Por este motivo, a funcionalidade de enviar mensagens com MDDE através de um servidor SMTP local pode ser uma solução um pouco pesada para um dispositivo móvel, pelo que o ideal é também disponibilizar uma alternativa para o envio de mensagens com MDDE, através da composição de mensagens na aplicação.

Para o efeito de envio através da aplicação, é disponibilizada uma GUI com funcionalidades idênticas às de composição num qualquer cliente de *e-mail*, onde podem ser definidos os campos de Destinatário, CC, BCC, o Assunto, o Corpo da mensagem e respectivos Anexos. De modo a integrar da melhor forma o funcionamento da aplicação com o OS, estão definidos filtros de *Intents*, que são activados quando uma outra aplicação no dispositivo móvel pretende enviar qualquer tipo de conteúdo multimédia (e também a partir de URLs "`mailto`"), sendo então apresentada a opção de compor uma nova mensagem no *plugin* com esses mesmos dados — como exemplificado na imagem do centro na figura 6.8. Com isto em conta, pode-se afirmar que, na prática, é disponibilizada na aplicação uma extensão como cliente de correio electrónico, destituído de capacidades de recepção e leitura de *e-mails*.

Validação de configurações

Tal como referido no início deste sub-capítulo, foi implementada a funcionalidade de validação das configurações da aplicação. Esta funcionalidade realiza um diagnóstico ao *plugin*, através de diferentes testes que permitem identificar configurações conflituosas e problemas de conectividade, e descobrir assim a causa de problemas que possam surgir durante a utilização da aplicação. Assim sendo, o utilizador pode realizar as seguintes operações (individualmente ou em simultâneo):

- **Testar ligação** — Verifica se o dispositivo está conectado à Internet (independentemente de qual a ligação utilizada para o efeito);
- **Testar servidores SMTP** — Tenta efectuar uma ligação aos servidores configurados na aplicação (i.e. o *Proxy* MDDE e o servidor SMTP para envio sem MDDE) e caso a ligação seja bem sucedida, interpreta a resposta obtida de forma a verificar se estes são

efectivamente servidores SMTP;

- **Testar credenciais de assinatura** — Utiliza o *Smart Card* para realizar uma assinatura digital sobre dados gerados aleatoriamente e valida a mesma através do certificado do utilizador, sendo por isso requiridas as credenciais da *applet* (ou seja, o PIN).

No caso de não existir conectividade à Internet, é apresentada uma mensagem a indicar o mesmo ao utilizador, tanto para o primeiro teste como para o segundo (uma vez que se não houver uma ligação activa não é possível testar os servidores SMTP). Para o segundo teste, se algum dos (ou até ambos) servidores falharem, é descrito esse mesmo problema e indicado que o servidores não puderam ser alcançados ou que não são servidores SMTP válidos. Relativamente às falhas que podem ocorrer durante o teste das credenciais de assinatura, é possível que o cartão não esteja presente, as credenciais estejam erradas, a assinatura não possa ser realizada (por não estar definida uma chave privada na *applet*, por exemplo) ou que falhe a validação da assinatura realizada (por não existir um certificado digital ou por a chave pública deste não corresponder à chave privada usada para assinatura). Em qualquer um dos casos, e tal como nos restantes testes da aplicação, é indicada a causa do problema ocorrido.

6.2.2 Componentes da aplicação

Dado que a portabilidade da aplicação era um dos objectivos apresentados para a solução a desenvolver, a implementação foi realizada com distinção entre as camadas de interface e de negócio, e com a organização modular dos componentes de cada uma das camadas. Desta forma, existe uma separação evidente entre os componentes que estão dependentes do OS móvel em específico e os componentes mais genéricos que podem ser transportados para outra plataforma (sem alterações ou apenas com algumas modificações), como pode ser constatado na figura 6.2. A implementação dos componentes responsáveis pela lógica aplicacional — que correspondem aos componentes mais genéricos — foi realizada com atenção para a abstracção de dependências (pelo uso de interfaces Java), pelo que os componentes desenvolvidos podem ainda ser facilmente substituídos por outras implementações sem a necessidade de alterar o código fonte das classes que os utilizam, dada a modularidade do *plugin* apresentado.

Os principais módulos presentes na aplicação móvel (conforme representados na figura 6.2) são descritos de seguida.

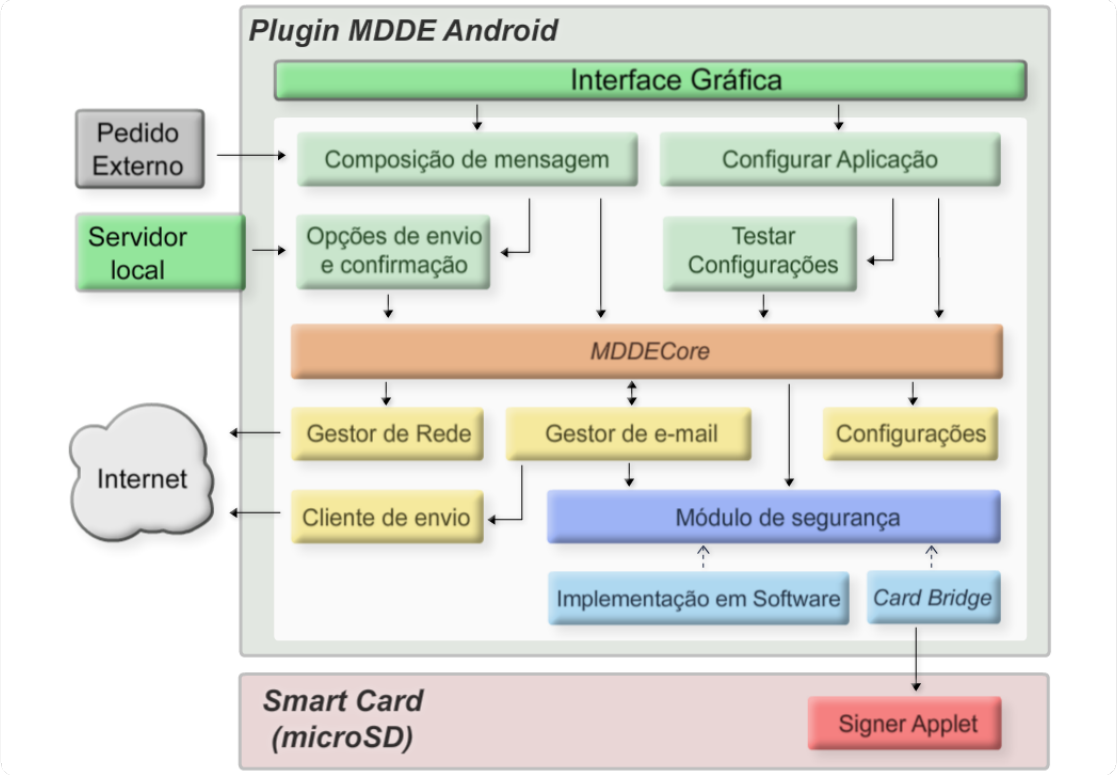

Figura 6.2: Componentes do *plugin* móvel

Composição de mensagem Componente responsável pela interface de composição de mensagens a enviar através do *plugin*, não intervindo, por isso, durante o processo de envio através do servidor SMTP local. Pode ser iniciado tanto internamente, como a partir de um pedido externo.

Opções de envio e confirmação Participa no processo de envio de mensagens, interceptadas pelo servidor SMTP local ou compostas na própria aplicação. A função deste módulo é apresentar uma janela de confirmação de envio da mensagem, sendo apresentada a opção de envio com MDDE ou enviar apenas como correio electrónico normal. Para o caso de mensagens compostas no *plugin*, são consideradas as definições de automatização de envio (ou seja, é possível que a janela não seja apresentada, se estiver definido um modo de envio automático para a aplicação).

Configurar aplicação Interface que permite definir as configurações da aplicação, bem como controlar o funcionamento do servidor SMTP local.

Testar configurações Componente de validação das configurações definidas. Permite testar a conectividade à Internet, a disponibilidade dos servidores de envio configurados e a integridade das credenciais de assinatura utilizadas.

Servidor local Executado e gerido através de um serviço Android, corresponde ao servidor SMTP responsável por captar as mensagens compostas em clientes de *e-mail* do dispositivo que estejam configurados para utilizar o *plugin* MDDE. Nas configurações da aplicação é possível definir o arranque automático do servidor local, para que o mesmo esteja disponível para utilização assim que o dispositivo é ligado, sendo também possível parar/reiniciar o servidor (de modo a evitar o consumo adicional de recursos do *smartphone*).

MDDECore O *MDDECore* representa uma "ponte" de abstracção entre a camada da interface com o utilizador, representada pelos componentes a verde acima do mesmo (na figura 6.2), e a camada de negócio, composta pelos restantes componentes do *plugin* móvel (por baixo do *MDDECore*). Por este motivo, os módulos da interface apenas contêm a referência para uma instância deste componente, sendo assim todas as funcionalidades dos restantes módulos executadas através de métodos disponibilizados pelo *MDDECore*.

Gestor de Rede Componente lógico utilizado para os testes de conectividade, responsável por verificar se existe ligação à Internet, se um endereço *web* está acessível e se um endereço indicado corresponde a um servidor SMTP.

Gestor de *e-mail* Responsável por criar mensagens no formato de *e-mail* apropriado — S/MIME ou apenas MIME, conforme o envio seja com ou sem MDDE, respectivamente — e compor os pedidos MDDE, se necessário. Uma vez feito isto, o gestor de *e-mail* reencaminha a mensagem a enviar para o cliente de envio.

Cliente de envio Recebe as mensagens compostas pelo gestor de *e-mail* e, após preparar a sessão SMTP para o envio das mesmas, remete-as para o servidor *Proxy* MDDE ou para o servidor SMTP do utilizador, conforme a modalidade pretendida para o envio das mensagens.

Configurações Componente com a função de gerir as configurações do *plugin* móvel. Disponibiliza métodos de consulta e modificação das definições armazenadas, para que outros componentes possam aceder às configurações (indirectamente, através do módulo *MDDECore*).

Módulo de segurança Participa na composição dos pedidos MDDE e na execução do teste correspondente à validação das credenciais, sendo a principal função a de assinar dados com as credenciais de assinatura do utilizador. A solução desenvolvida possui duas implementações, conforme visível na figura: uma implementação em *software* (que é descrita em 6.3.2) e uma "ponte" para a *applet* de assinatura (presente no *Smart Card*) que, para as operações a realizar, gera APDUs e transmite comandos/respostas entre o dispositivo móvel e o cartão *microSD*.

6.2.3 Interface gráfica

O desenvolvimento da interface gráfica do *plugin* móvel foi em grande parte facilitado pela sólida integração do SDK Android com o IDE Eclipse, que disponibiliza um construtor de GUI, para a edição dos ficheiros XML responsáveis pela disposição e conteúdo das *Views*. O uso de ficheiros XML para a estruturação dos textos utilizados na interface gráfica, foi também auxiliado e encorajado por esta plataforma de desenvolvimento, de modo a facilitar a posterior adaptação/localização para outras línguas (se pretendido).

Assim sendo, foi construída uma solução que contém uma interface "escalável" e compatível com todos os dispositivos Android de versão superior ou igual à 2.2, incluindo tanto *smartphones* como *tablets* com este Sistema Operativo. Os principais componentes da interface gráfica resultante são apresentados de seguida, conforme representados nas figuras 6.3 a 6.8[2].

Ecrã principal (figura 6.3) — Nesta figura estão visíveis os pontos de entrada para o *plugin* MDDE, nomeadamente a notificação do servidor SMTP local em execução (que apenas está presente quando o servidor se encontra activo) e os ícones da aplicação — o ícone principal e o atalho para o ecrã de composição de nova mensagem. Na imagem da direita, encontra-se o menu principal do *plugin* móvel, que disponibiliza as funcionalidades de compor mensagens, configurar as definições de envio, consultar outras opções da aplicação e obter informação sobre o *plugin*.

[2]Nota: As capturas de ecrã representam o estado final da aplicação desenvolvida, tendo sido obtidas através do dispositivo móvel de teste.

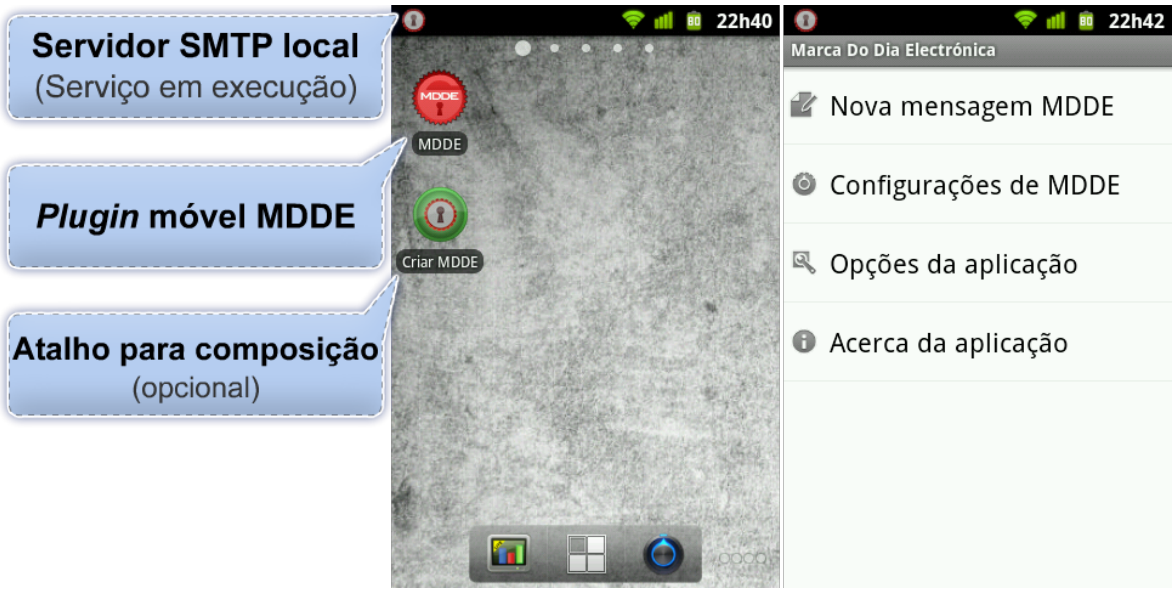

Figura 6.3: Interface gráfica do *plugin* — Ecrã principal

Figura 6.4: Interface gráfica do *plugin* — Composição de mensagens

A composição de mensagens (figura 6.4) — O ecrã de composição pode ser iniciado a partir do menu principal, do atalho existente, ou de um pedido (*Intent*) externo. Além dos campos visíveis na imagem da esquerda, é ainda possível adicionar/remover endereços CC e BCC no menu da *View*. Os botões na barra inferior permitem enviar a mensagem, adicionar anexos, ir para as configurações de envio e ainda obter informações sobre a mensagem a

enviar, respectivamente. Na imagem central, está representada a janela de confirmação e de opção de envio — não apresentada quando o modo de funcionamento está definido para uma opção com envio automático que não requeira confirmação de envio. Por fim, na imagem da direita, encontra-se a janela para a introdução das credenciais de assinatura, que são necessárias para o envio de mensagens com MDDE.

Figura 6.5: Interface gráfica do *plugin* — Validação de definições

Testar as configurações (figura 6.5) — O menu para a verificação das configurações da aplicação contém as opções representadas na imagem da esquerda, sendo que quando um ou mais testes são realizados, é apresentada uma indicação visual do estado dos mesmos (a verde para testes bem sucedidos e a vermelho para testes falhados). Em caso de erro, é ainda descrita a origem do problema, como representado na imagem da direita.

Configurações do *plugin* (figura 6.6) — Nesta figura estão representados os ecrãs de configurações do *plugin*, correspondentes às opções "Configurações MDDE" (imagem da esquerda e central) e "Opções da aplicação" (imagem da direita) do menu principal. No primeiro menu, estão listadas as configurações necessárias para o envio de mensagens pelo *plugin*, enquanto que o segundo menu corresponde a outras opções complementares e ao controlo do servidor SMTP local[3].

[3] Apesar de estar presente nas opções da aplicação, a funcionalidade de actualizar a aplicação não se encontra implementada.

Figura 6.6: Interface gráfica do *plugin* — Configurações da aplicação

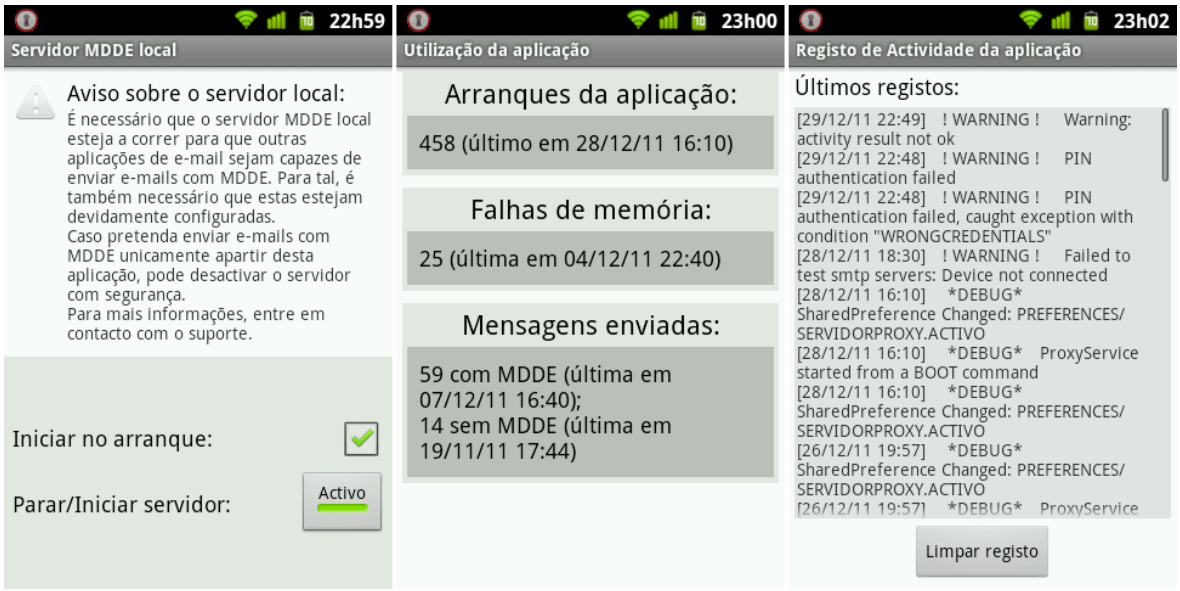

Figura 6.7: Interface gráfica do *plugin* — Servidor, estatísticas e registos

Outras opções (figura 6.7) — As opções aqui demonstradas podem ser acedidas através do menu "Opções da aplicação", e permitem controlar o funcionamento do servidor (imagem da esquerda), consultar certas estatísticas sobre a utilização do *plugin* (imagem do centro) e ler as últimas 100 entradas do registo de actividade (imagem da direita). O ecrã de configuração do servidor local permite iniciar ou parar o servidor, bem como definir o modo de

arranque automático para que este seja iniciado assim que o dispositivo é ligado. No menu de estatísticas, é possível obter a informação sobre o número de arranques da aplicação e a data do último arranque, sobre o número de vezes que a aplicação foi terminada por falta de recursos do dispositivo e sobre a quantidade de mensagens enviadas pelo *plugin*. Já o registo da actividade (ou "*log*" da aplicação) disponibiliza informação útil para depuração, que pode ser utilizada para manutenção da aplicação e eventuais correcções que esta necessite. Este "*log*" é configurável por tipo de registo — i.e. nenhum, escrita para ficheiro de texto, *logcat* do Android, ou ambos – e por tipo de conteúdo a registar (erros, avisos, etc), e não guarda informação sobre o assunto ou conteúdo das mensagens enviadas (apenas regista o pedido MDDE composto.

Figura 6.8: Interface gráfica do *plugin* — Pedidos externos

Pedidos externos (figura 6.8) — Durante o envio de uma mensagem composta num cliente de *e-mail* externo para o servidor local, é apresentada a janela de confirmação de envio representada na imagem da esquerda. Na imagem do centro é visível a integração da solução com o Sistema Operativo, sendo mostrada a opção de partilha por MDDE, que inicia a composição de uma nova mensagem no *plugin* com os ficheiros seleccionados em anexo (imagem da direita).

6.3 O módulo de segurança

O módulo de segurança desenvolvido é o componente da solução responsável pela gestão das credenciais de assinatura e pela realização de assinaturas digitais sobre os pedidos MDDE a enviar. Para tais funcionalidades, foi implementada a solução prevista pelo desenvolvimento de uma *applet* de assinatura para o cartão *microSD*, que é acedida e utilizada através do componente de abstracção do módulo de segurança, conforme descrito em 6.2.2. Além desta solução, foi ainda desenvolvida uma implementação em *software*, com o objectivo de testar o funcionamento e o desempenho da aplicação Android com duas soluções alternativas (i.e. com e sem o módulo de segurança físico).

6.3.1 *Applet* de assinatura

Para fornecer as funcionalidades previstas, foram implementados os métodos listados na tabela 6.1. A autenticação de credenciais é assim feita através de 3 operações específicas para o efeito, nomeadamente de verificar o PIN, o PUK e o SLK (como especificado nos requisitos de segurança da solução, no sub-capítulo 5.2.4). Estas credenciais são necessárias para as operações que resultam em alterações do estado interno da *applet* e para a operação de assinatura. Além das funcionalidades requiridas, foram também desenvolvidos métodos que permitem obter um código de versão da *applet* (particularmente útil para propósitos de depuração e controlo de compatibilidade) e obter o número de tentativas de autenticação restantes, para cada uma das credenciais utilizadas.

Relativamente ao processo de personalização da *applet*, as três credenciais de acesso são definidas através dos parâmetros de instalação da mesma. Para tal, foi definida uma subestrutura específica, composta por: comprimento do PIN; valor do PIN; comprimento do PUK; valor do PUK; comprimento do SLK; valor do SLK. Para o comprimento de cada credencial, é utilizado um *byte*, sendo o comprimento efectivo dessa mesma credencial limitado pelas dimensões mínimas e máximas impostas pela *applet* (2 a 4 *bytes* para o PIN, 4 a 8 *bytes* para o PUK e 4 a 12 *bytes* para o SLK). Caso o comprimento de uma credencial não obedeça a estes parâmetros, a instalação da *applet* é cancelada.

Durante o método de instalação, é também inicializada a chave privada e a instância da cifra utilizada para a assinatura de dados. Como a implementação apresentada apenas se

Operação	Credencial requirida	Dados de entrada	Resultado da operação
Verificar PIN	-	PIN	-
Verificar PUK	-	PUK	-
Verificar SLK	-	SLK	-
Alterar PIN	PIN	Novo PIN	-
Desbloquear PIN	PUK	Novo PIN	-
Obter contador PIN	-	-	Tentativas restantes (PIN)
Obter contador PUK	-	-	Tentativas restantes (PUK)
Obter contador SLK	-	-	Tentativas restantes (SLK)
Obter versão	-	-	Versão da *applet*
Limpar Chave Privada	SLK	-	-
Definir Chave Privada	SLK	Chave Privada	-
Assinar Dados	PIN	*Hash* de dados	Assinatura digital

Tabela 6.1: Operações da *applet* de assinatura

trata de uma solução conceptual, a *applet* desenvolvida só está preparada para a utilização de chaves privadas RSA de 1024 *bits* em assinaturas digitais com o protocolo `PKCS#1`. De modo a optimizar o processo de assinatura, a chave privada no cartão é definida de acordo com os valores utilizados pelo algoritmo CRT. Para especificar qual o valor que pretende definir durante a inicialização da chave privada, é utilizado o parâmetro "`P2`" do comando APDU, uma vez que tal inicialização não é possível durante a fase de instalação da *applet* (dado o limite do número de *bytes* transmitidos por um APDU).

6.3.2 Uma alternativa em *software*

Um dos objectivos da solução proposta é a portabilidade para outras plataformas. Com isto em conta, o sistema foi especificado e implementado com particular atenção para a modularidade dos componentes desenvolvidos, o que inclui o próprio módulo de segurança da aplicação Android. Como prova de conceito, além da solução que utiliza o *Smart Card* para a realização das assinaturas digitais e suporte da chave privada, foi também desenvolvida uma alternativa em *software*, que se pode demonstrar útil caso seja pretendido, posteriormente,

transportar a solução para outras plataformas não compatíveis com o cartão utilizado, ou ainda utilizar a mesma em dispositivos que não disponham de entradas *microSD*.

Esta alternativa foi desenvolvida por um grupo de alunos — Arley Pinto e Miguel Lopez — do primeiro ano de mestrado em Engenharia Informática da Universidade do Minho, tendo sido conseguida através da implementação dos métodos presentes na interface do módulo de segurança do *plugin* para Android, igualmente utilizada para a integração da solução com o *Smart Card* através do "*Card Bridge*", como representado na figura 6.2. Deste modo, tal como acontece com os restantes componentes da aplicação para Android, não existe a necessidade de reformular o modo de operação dos componentes que utilizam este módulo, sendo apenas necessário escolher previamente qual a instância a utilizar — se o módulo de segurança em *software* ou se a versão com o *Smart Card*.

Para compilar a aplicação Android para uma ou outra solução, com a preservação da abstracção de dependências entre os componentes, o mecanismo utilizado foi o recurso a uma "*flag*" de compilação no módulo *MDDECore*, ou seja, um valor booleano que determina qual a instância a utilizar. Este parâmetro é ainda utilizado para determinar certos aspectos dos componentes da interface gráfica da aplicação, uma vez que o modo de autenticação altera entre as duas soluções (apesar de se manterem intactos os componentes correspondentes à lógica do negócio). Assim, as principais diferenças de operação entre as soluções são as seguintes:

- ***KeyStore*** — Passa a ser utilizado um *KeyStore* Java para o armazenamento das credenciais, tanto para a chave privada como para o certificado do utilizador. Este *KeyStore* é de acesso restrito, através de uma palavra-passe, necessária para a leitura da chave privada;

- **Configurações** — Dado o comprimento arbitrário da palavra-passe da *KeyStore*, que habitualmente é bastante extenso como forma de combater ataques por força-bruta, foi tomada a decisão de permitir definir o identificador da chave privada ("*alias*") e a palavra-passe nas configurações da aplicação;

- **Requisição de credenciais** — Visto que as credenciais são armazenadas através das preferências da aplicação, deixam de ser necessários os pedidos de autenticação para o envio de mensagens com MDDE e para a validação das configurações.

Apesar de a alternativa por *software* apresentar a vantagem de abranger uma maior gama de dispositivos-alvo e de ser, do ponto de vista funcional, equivalente à solução que utiliza o *Smart Card*, surgem algumas diferenças quando avaliada a nível de segurança. A título de exemplo, existe a possibilidade de um atacante copiar a *KeyStore* utilizada para tentar obter a chave privada posteriormente, enquanto que, no caso da solução que utiliza um *Smart Card*, a chave privada não sai do cartão (e existe protecção física contra tentativas de cópia dos dados). Outro factor a considerar é o armazenamento da palavra-passe da *KeyStore*, uma vez que esta pode ser obtida no caso de ser utilizado um dispositivo com acesso *root*, e obtida desta forma a chave privada, enquanto que o conhecimento do PIN (no caso da solução por *hardware*) não permite obter a chave privada. Por estes motivos, uma solução para o envio de MDDE em dispositivos móveis que recorra unicamente a *software* para a gestão de credenciais e assinatura dos pedidos teria que ser melhor estudada, pelo que esta alternativa apresentada deve ser entendida apenas como objecto de estudo para demonstrar a modularidade da solução e comprovar que seria possível uma solução "híbrida", de fácil manutenção e alteração entre as duas modalidades.

6.4 Resultados obtidos

Após o desenvolvimento da solução para envio de mensagens MDDE em dispositivos móveis, e de modo a avaliar as aplicações apresentadas, torna-se necessário realizar a validação da implementação face aos requisitos definidos. Neste sub-capítulo, são também referidas as principais dificuldades que ocorreram durante o processo de implementação e que implicaram o estudo e desenvolvimento de soluções complementares para o problema apresentado.

6.4.1 Validação dos requisitos definidos

Todos os objectivos e requisitos funcionais especificados no capítulo 5 para a solução a desenvolver foram cumpridos, à excepção da funcionalidade de actualização da aplicação para Android, como referido em 6.2.1. A implementação apresentada obedece também aos requisitos de segurança indicados, nomeadamente:

- Restrição de acesso às funcionalidades da *applet*, pelos códigos PIN, PUK e SLK;
- Comprimentos mínimos e tentativas máximas de autenticação para os códigos de acesso;
- Bloqueio permanente da *applet* caso o PUK seja bloqueado;

- Não é disponibilizado qualquer meio de transmissão dos códigos de acesso nem da chave privada para o exterior;
- As tentativas de envio sem autorização do utilizador são identificadas, sendo sempre apresentada a janela de confirmação de envio para mensagens interceptadas pelo servidor SMTP local;
- Utilização de classes específicas e métodos disponibilizados pela API do Java Card para controlo de atomicidade sobre os objectos que armazenam as credenciais da *applet*;
- Armazenamento em memória interna das configurações da aplicação móvel.

Em relação à opção de actualizar o *plugin* Android, apesar de a funcionalidade não ter sido desenvolvida, foram implementados alguns métodos que permitem testar a conectividade a um (eventual) servidor de actualização, e a aplicação está também preparada para actualizações através do *Android Market* — não sendo necessária qualquer alteração a nível de código fonte do *plugin*, neste caso.

6.4.2 Avaliação geral da solução

De um modo geral, o *plugin* MDDE encontra-se totalmente funcional para os objectivos propostos, não tendo surgido qualquer problema de utilização durante os testes realizados com a versão final da solução desenvolvida (quer com o módulo de segurança em *software* quer com o uso do *Smart Card*). De modo a fornecer uma avaliação mais precisa da implementação apresentada, foram realizados testes de desempenho da assinatura digital de pedidos MDDE e foi testada a estabilidade do *plugin* em condições de fraca memória (i.e. quando existe pouca memória RAM disponível).

Para os testes de desempenho, foram efectuadas 5 medições (com cada implementação do módulo de segurança) do tempo necessário para a assinatura digital de pedidos MDDE[4]. O processo de assinatura inclui, em ambos os casos, o cálculo prévio do resumo digital dos dados, tendo sido usada a mesma implementação em *software* para as duas alternativas, de modo a mitigar possíveis diferenças influenciadas por este factor. A média (aproximada às unidades) resultante das medições foi de 1900 milissegundos para a solução com o *Smart Card*

[4]O tempo de execução foi obtido pela comparação de duas instâncias da classe "`Date`" no código Java da solução, criadas imediatamente antes e depois da assinatura digital.

e de 2254 milissegundos para o módulo de segurança em *software*, o que permite concluir que a assinatura através da *applet* é mais eficaz, mesmo que por uma pequena margem. Isto pode ser influenciado por vários factores, como por exemplo o facto de ser usada a optimização com o CRT na versão de assinatura pela *applet* e a presença de um co-processador criptográfico no cartão *microSD*, que auxilia a computação da assinatura. De qualquer forma, é interessante verificar que o recurso ao *Smart Card* para assinar os pedidos MDDE se demonstra ser uma solução bastante eficiente para a resolução do problema.

Relativamente aos testes de execução com recursos limitados, mais especificamente limitações de memória RAM, foi utilizado o ADB como ferramenta de depuração em tempo real, para reconhecimento de tais eventos. Estas ocorrências foram também registadas na aplicação móvel através da implementação do método "`onLowMemory()`", que fecha as actividades da aplicação em segundo plano e regista a falha de memória ocorrida, podendo este evento ser observado no menu de estatísticas, como exemplificado na captura de ecrã presente na figura 6.7 (imagem central). Além da medida de fechar as actividades existentes quando o OS necessita de libertar recursos, foi também especificado que caso o servidor local SMTP seja forçosamente encerrado por este mesmo motivo, o sistema Android deve reiniciar o servidor de imediato, assim que possível. Os testes realizados consistiram em abrir a aplicação, navegar por alguns menus (para colocar uma "*stack*" de actividades em memória) e de seguida executar várias aplicações pesadas, de modo a forçar a falta de memória disponível no dispositivo. Os resultados foram os esperados, com as actividades do *plugin* a serem terminadas com sucesso de modo a assegurar (quando possível) a permanência do servidor local em execução. Em situações mais extremas, o servidor local era também terminado por falta de recursos disponíveis, sendo iniciado automaticamente assim que as condições retomavam a normalidade.

Por realizar, poderiam ter sido efectuados testes de *stress*, com um número elevado de pedidos de envio em simultâneo para o servidor local SMTP. No entanto prevê-se que o comportamento resultante seja a instabilidade do OS e a perda de parte das mensagens capturadas, uma vez que o sistema Android trata de terminar actividades em segundo plano quando são necessários mais recursos para a actividade em plano principal (e, desta forma, termina as janelas de confirmação de parte das mensagens a enviar).

6.4.3 Principais dificuldades e resultados complementares

Durante o desenvolvimento do *plugin* para Android, surgiram vários problemas inesperados que necessitaram de soluções específicas para que a aplicação desempenhasse as funcionalidades correspondentes. As principais dificuldades associadas à aplicação móvel foram relacionadas com o envio e intercepção de mensagens de correio electrónico, como explicado de seguida (e descritas as respectivas soluções adoptadas).

Envio de *e-mails* A *framework* do sistema Android simplifica acções simples como a de enviar mensagens de correio electrónico, através da integração com o OS pelo uso de *Intents* e de filtros para os mesmos. No entanto, uma aplicação que pretenda utilizar tal funcionalidade, apenas inicia essa operação — i.e. são enviados os dados para uma aplicação de envio, mas a aplicação que inicia a actividade não controla a sua concretização. O que acontece, na prática, é que apenas são transmitidos à aplicação de destino certos elementos soltos da mensagem a enviar, como os destinatários, assunto e texto da mensagem, não sendo possível fornecer uma mensagem MIME completa nem garantir a integridade da mensagem a enviar. No caso do *plugin* MDDE, existia a necessidade de o próprio *plugin* definir a estrutura final da mensagem de *e-mail* (para a produção de pedidos MDDE) e ser capaz de enviar a mensagem sem dependência de aplicações de terceiros.
Solução adoptada: Para a linguagem Java, uma das *frameworks* mais conhecidas e utilizadas para composição e envio de mensagens de *e-mail* é o JavaMail. No entanto, esta *framework* contém dependências de outras bibliotecas como o AWT ("*Abstract Window Toolkit*", ferramenta para GUI Java), não disponíveis para o OS Android. Por este motivo, a solução passou por utilizar uma versão adaptada do JavaMail especificamente para Android, de nome "javamail-android"[5].

Mensagens S/MIME Um dos problemas do OS Android é a falta de correcção relativamente a implementações dos *Security Providers*, como descrito em 3.2.5. Também o desenvolvimento do *plugin* MDDE foi afectado por esta questão, uma vez que o *Security Provider* utilizado para estruturar as assinaturas dos pedidos (durante a composição do S/MIME) — o "Bouncy Castle" — não estava a realizar assinaturas digitais válidas. Após a comparação de várias mensagens S/MIME, assinadas quer com recurso ao *Security Provider* faltoso quer através de clientes de *e-mail* em computadores pessoais, e através da análise da estrutura

[5] http://code.google.com/p/javamail-android/

`ASN.1`[6] das assinaturas, foi descoberta a origem do problema: além de estar a ser utilizada uma versão desactualizada da biblioteca (versão 1.38, quando na altura já estava disponível a versão 1.46), existiam alterações no código fonte da biblioteca, que resultavam na omissão de um parâmetro requerido na estrutura. Mais especificamente, o campo em questão diz respeito aos parâmetros do algoritmo de *hash* da assinatura, em que para o esquema de assinatura utilizado — `RSASSA-PKCS1-v1_5` — deve estar explicitamente definido o valor "`NULL`"[7].
SOLUÇÃO ADOPTADA: Para contornar este problema, foi incluída (no projecto do *plugin*) a versão 1.46 da biblioteca. Para isto, foi necessário recompilar o Bouncy Castle com um nome de *package* diferente, para que não fosse utilizada a implementação do sistema.

Apesar de não serem motivadas por problemas de implementação, também no desenvolvimento da *applet* para Java Card houve a necessidade de recorrer a soluções que não estavam inicialmente previstas.

Compilação da *applet* Com o propósito de auxiliar e acelerar o desenvolvimento de aplicações para a plataforma Java Card, a solução ideal é utilizar um IDE preparado para esse mesmo efeito. No entanto, não foram encontradas extensões compatíveis com a versão necessária do Java Card (2.2.1) que permitissem compilar *applets* no mesmo OS utilizado para o desenvolvimento da aplicação para Android (i.e. Mac OSX).
SOLUÇÃO ADOPTADA: Uma vez que o Java Card SDK disponibiliza as ferramentas de desenvolvimento para sistemas UNIX, foi escrito um *Ant Script* para facilitar a compilação da *applet* através do IDE Eclipse. Desta forma, directamente do IDE é possível configurar parâmetros como o AID, compilar a *applet* e gerar os APDUs de instalação da mesma.

APDUs de inicialização Os comandos e respostas APDU são compostos unicamente por sequências de *bytes*, sem qualquer encapsulamento por objectos, de modo a acelerar tanto a transmissão como a leitura e escrita dos mesmos. Quando se considera a inicialização de objectos que requerem grandes quantidades de dados — como a chave privada no cartão, definida por 5 parâmetros de 128 *bytes* (chave RSA 1024 *bits* com CRT)—, torna-se impossível a transcrição manual destes valores, especialmente por ser necessário executar este procedimento a cada nova versão (ou instalação) da *applet*, durante a fase de testes.

[6] ASN.1 ("*Abstract Syntax Notation One*") — notação utilizada para descrever regras e estruturas de dados, definida nos padrões `ISO/IEC 8824` e `8825`.

[7] Conforme definido no apêndice *A*.2.4 da especificação do padrão PKCS#1: http://tools.ietf.org/html/rfc3447#appendix-A.2.4.

SOLUÇÃO ADOPTADA: Foi desenvolvida uma aplicação para *Desktop*, em Java, que permite seleccionar um repositório *Public-Key Cryptography Standards* (PKCS)#12 com a chave privada do utilizador e gerar os APDUs de inicialização da chave privada, pela descomposição da chave nos factores necessários para o CRT e conversão destes dados em APDU válidos para a *applet*.

CAPÍTULO 7

Conclusões e Trabalho Futuro

Neste capítulo, são apresentadas as conclusões sobre o trabalho desta dissertação, bem como apresentadas algumas sugestões para trabalho futuro.

7.1 Conclusões

O MDDE é um serviço de troca de mensagens que utiliza o *e-mail* como base e visa assegurar a autenticidade do conteúdo das mensagens e fornecer a prova de envio das mesmas, disponível para utilização apenas em computadores pessoais com o sistema operativo Windows. Esta limitação serviu de motivação ao estudo de uma possível alternativa em ambientes móveis, dadas as vantagens apresentadas pelos mesmos e o potencial que um serviço como o MDDE poderia obter com tal solução.

O objectivo desta dissertação foi investigar se as características actuais dos dispositivos móveis mais utilizados de momento — os *smartphones* — e dos *Smart Cards* existentes compatíveis com tais dispositivos (mais especificamente, *Smart Cards microSD*) poderiam ser conjugados numa solução de envio de mensagens MDDE que fornecesse características funcionais e de segurança idênticas às apresentadas pelo *plugin* existente para Windows. Para o sistema pretendido, foram utilizados como alvos principais de estudo os *smartphones* com o OS Android e *Smart Cards* com a plataforma *Java Card*, ambos passíveis de personalização através de aplicações específicas, dado que também era pretendida uma solução o mais genérica possível.

A abordagem tomada para o problema apresentado foi a especificação de uma solução com

funcionamento análogo ao *plugin* MDDE existente: para interceptar as mensagens a enviar pelo serviço, seria utilizado um servidor SMTP para o *plugin* móvel, executado localmente, com a função de redireccionar as mensagens recebidas para outro componente da aplicação, que por sua vez seria responsável pela composição e envio dos pedidos MDDE. No entanto, dado que estes dispositivos apresentam uma capacidade computacional inferior à dos computadores pessoais, foi também considerada a hipótese de permitir a composição de mensagens na própria aplicação, como forma de colmatar a limitação de recursos. Já para a assinatura digital dos pedidos de envio MDDE, foi especificada uma *applet* para o *Smart Card*, capaz de armazenar as credenciais necessárias para o mesmo e desempenhar a respectiva função.

Um dos principais desafios do trabalho realizado durante a especificação e implementação das aplicações, prendia-se com o facto de não ter existido qualquer experiência prévia de desenvolvimento para as duas plataformas utilizadas — Android e Java Card. Dada a diferença, tanto estrutural como operacional, entre as aplicações para *Desktop* e as aplicações móveis para Android e *applets* para Java Card, foi necessário estudar aprofundadamente ambas *frameworks* e os ciclos de vida das aplicações para estas plataformas, detalhadas nos capítulos 3 e 4. No entanto esta dificuldade foi ultrapassada, tendo sido realizada a implementação do *plugin* móvel conforme previsto, com uma solução para envio de mensagens com MDDE a partir de dispositivos Android, que utiliza *Smart Cards* no formato *microSD* para a assinatura digital dos pedidos de envio. Para tal, foi também implementada a funcionalidade de composição de mensagens na própria aplicação, sendo por isso fornecidas duas opções para a composição: realizada pela interface do próprio *plugin* ou então utilizar mensagens interceptadas pelo servidor SMTP local.

Além da solução que utiliza o módulo de segurança físico (i.e. o *Smart Card*), foi também desenvolvida uma alternativa em *software* para a assinatura digital dos pedidos MDDE e gestão das credenciais de assinatura correspondentes. Esta componente tirou partido da estrutura modular do *plugin* móvel implementado, estrutura esta que foi utilizada com o propósito de alcançar o objectivo de concretizar uma solução o mais genérica possível: uma vez organizada por componentes distintos e independentes, também a implementação lógica da aplicação é independente da interface gráfica da aplicação, o que significa que para transportar a solução desenvolvida para outra plataforma móvel que utilize a mesma linguagem de programação (Java), apenas é necessário desenvolver a GUI para a nova aplicação, podendo

ser reaproveitados os restantes componentes (sem alterações ou apenas com algumas modificações).

Com a implementação realizada, é possível concluir que o *plugin* móvel pode ser executado em qualquer dispositivo Android que apresente no mínimo a versão 2.2 do OS e características semelhantes (ou superiores) às utilizadas pelo dispositivo de teste. O mesmo pode também ser afirmado para o *Smart Card*, sendo apenas necessário que o cartão *microSD* utilizado contenha a plataforma Java Card versão 2.2.1 (ou mais recente) e disponibilize uma *framework* para acesso através de dispositivos Android.

Comparativamente ao *plugin* existente para computadores pessoais, foi possível demonstrar que a solução proposta é uma solução viável, eficiente e completa para o envio de mensagens através do serviço MDDE, com as vantagens acrescidas da mobilidade e ubiquidade característica destes dispositivos.

7.2 Trabalho Futuro

Apesar de o processo de envio corresponder à componente do sistema MDDE que mais tira proveito das características destes dispositivos móveis, seria interessante estudar e implementar a sua contraparte, ou seja, uma solução móvel para a validação de mensagens MDDE. Este seria, portanto, o próximo passo a tomar relativamente a trabalho futuro no tópico, sendo que para tal poderia ser aproveitada a modularidade da solução aqui apresentada (e possivelmente integrar as duas soluções numa só). Também como alvo de trabalho futuro, poderia ser considerada a especificação e implementação do *plugin* de envio noutras plataformas móveis, bem como estudar a a possibilidade de adaptar esta solução para o ambiente *Desktop*, como forma de disponibilizar a utilização do serviço noutros sistemas operativos que não o Windows.

Em relação a outros alvos de estudo para solução desenvolvida, seria interessante permitir gerar as credenciais do utilizador — chave pública e privada — no próprio *Smart Card*, de modo a que a chave privada nunca fosse conhecida no exterior do cartão (ao contrário do que acontece actualmente, uma vez que esta tem de ser conhecida pelo menos para a inicialização da *applet* no cartão). Adicionalmente, poderia também ser estudada a possibilidade de

utilizar o *Smart Card* para armazenar e/ou cifrar certos dados de configuração da aplicação móvel, tais como as credenciais SMTP e as configurações dos servidores de envio. Seria ainda interessante estudar a hipótese de requerer um factor de autenticação adicional durante o envio de *e-mails* através do servidor local, recorrendo à validação de credenciais SMTP que seriam definidas no *plugin* e posteriormente especificadas nos clientes de *e-mail*, de modo a evitar tentativas de envio não autorizadas por outras aplicações no dispositivo.

Quanto a possíveis melhorias, poderiam ser efectuadas algumas modificações ao servidor SMTP local de modo a que este alterasse para um estado de "pausa" cada vez que o dispositivo fosse bloqueado (e o ecrã desligado), e resumida a execução assim que o utilizador retomasse a presença. Desta forma, seriam poupados recursos computacionais dos dispositivos móveis sem afectar a disponibilidade do serviço, uma vez que apenas é necessário que o servidor intercepte mensagens assim que o utilizador termine a composição das mesmas, e para tal o dispositivo tem de estar activo. Esta alteração poderia ter um impacto significativo na autonomia dos dispositivos, visto vez que de acordo com [42], estes aparelhos passam cerca de 89% do tempo no estado inactivo.

Bibliografia

[1] C. Adams, P. Cain, D. Pinkas, e R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), 2001.

[2] Giuseppe Ateniese, Breno de Medeiros, e Michael T. Goodrich. Tricert: A distributed certified e-mail scheme, 2001.

[3] Zhiqun Chen. *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, Junho 2000.

[4] Roberto Civita. *Enciclopédia Conhecer*, volume 8, páginas 1121–1123,1259. Editora Abril Cultural, 1988.

[5] Roberta Cozza, Annette Zimmermann, Carolina Milanesi, Hugues J. De La Vergne, Atsuro Sato, CK Lu, David Glenn, Tuong Huy Nguyen, Sandy Shen, e Anshul Gupta. Market Share: Mobile Communication Devices by Region and Country, 2Q11. Relatório Técnico G00219350, Gartner, Agosto 2011.

[6] CTT e MULTICERT. *Manual de validação de mensagens de correio electrónico com MARCA DO DIA ELECTRÓNICA (MDDE)*, 1^a edição, Novembro 2003.

[7] Decreto-Lei n.º 290-D/99 de 2 de Agosto. Aprova o regime jurídico dos documentos electrónicos e da assinatura digital. Diário da República, Agosto 2009. S.1-A n.º 178, 1.º suplemento, p. 4990-(2) a 4990-(11).

[8] Deloitte's Research. Killer apps – the promises and pitfalls of a smarter world. Relatório técnico, Deloitte LLP, Julho 2011.

[9] Deloitte's Technology, Media and Telecommunications. Deloitte analysis of top technology trends for 2011. Relatório técnico, Deloitte LLP, Janeiro 2011.

[10] Whitfield Diffie, Paul C. Oorschot, e Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992. 10.1007/BF00124891.

[11] David C. Douglas, Yassir K. Elley, Radia J. Perlman, Sean J. Mullan, e Anne H. Anderson. Replacing an email attachment with an address specifying where the attachment is stored, Maio 2006.

[12] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, e Deborah Estrin. Diversity in smartphone usage. Em *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, páginas 179–194, New York, NY, USA, 2010. ACM.

[13] Tom Farley. Mobile telephone history. *Telektronikk*, página 26, Abril 2005.

[14] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, e David Wagner. A survey of mobile malware in the wild. Em *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, SPSM '11, páginas 3–14, New York, NY, USA, Outubro 2011. ACM.

[15] Uwe Hansmann, Martin S. Nicklous, Thomas Schäck, e Frank Seliger. *Smart Card Application Development Using Java*. 1 edição, Outubro 1999.

[16] R. Housley. Cryptographic Message Syntax (CMS). RFC 5652, Internet Engineering Task Force, Setembro 2009. http://tools.ietf.org/html/rfc5652; acedido em 30 de Setembro de 2011.

[17] IDC Worldwide Quarterly Mobile Phone Tracker. Apple Rises to the Top as Worldwide Smartphone Market Grows 65.4the Second Quarter of 2011. Relatório Técnico US22974611, IDC, Agosto 2011.

[18] IDC Worldwide Quarterly Mobile Phone Tracker. Smartphones Outstrip Feature Phones for First Time in Western Europe as Android Sees Strong Growth in 2Q11. Relatório Técnico US22871611, IDC, Setembro 2011.

[19] IDC Worldwide Quarterly Mobile Phone Tracker. Worldwide Smartphone Market Expected to Grow 55% in 2011 and Approach Shipments of One Billion in 2015. Relatório Técnico US22871611, IDC, Junho 2011.

[20] ITU. Measuring the Information Society. Relatório Técnico 36736, International Telecommunication Union, Setembro 2011.

[21] Timo Kopomaa. *The city in your pocket: Birth of the Mobile Information Society*. Gaudeamus, 1^a edição, 2000.

[22] Kwok-Yan Lam e Thomas Beth. Timely Authentication in Distributed Systems. Em *Proceedings of the Second European Symposium on Research in Computer Security*, páginas 293–303, London, UK, 1992. Springer-Verlag.

[23] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, e Stephen Wolff. A brief history of the Internet. *SIGCOMM Comput. Commun. Rev.*, 39:22–31, Outubro 2009.

[24] Rich Ling. *The Mobile Telephone and Teens*, páginas 83–121. Morgan Kaufmann, San Francisco, CA, 2004.

[25] Steve Litchfield. The History of Psion. http://stevelitchfield.com/historyofpsion.htm; acedido em 2 de Outubro de 2011, 2005.

[26] Vašek Lorenc, Tobiáš Smolka, e Petr Švenda. Automatic source code transformations for strengthening practical security of smart card applications. Em *Europen 2010*.

[27] Keith Mayes e Konstantinos Markantonakis. *Smart Cards, Tokens, Security and Applications*. Janeiro 2008.

[28] Mary Meeker, Scott Devitt, e Liang Wu. Internet Trends. Relatório técnico, Morgan Stanley Research, Abril 2010.

[29] Alfred J. Menezes, Scott A. Vanstone, e Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1^a edição, 1996.

[30] Collin Mulliner. NFC and NDEF Hacking. Em *NinjaCon*, Vienna, Austria, Junho 2011.

[31] Christy Pettey e Laurence Goasduff. Worldwide Mobile Application Store Revenue Forecast to Surpass $15 Billion in 2011. Relatório técnico, Gartner, Janeiro 2011.

[32] D. Pinkas. Time Stamping profile. Relatório Técnico 1.3.1, European Telecommunications Standards Institute (ETSI), Janeiro 2006.

[33] D. Pinkas, J. Ross, e N. Pope. Electronic Signature Formats for long term electronic signatures, 2001.

[34] Hester Plumridge e Martin Peers. Nokia Calls Microsoft for Help. *The Wall Street Journal*, Fevereiro 2011.

[35] B. Ramsdell e S. Turner. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 7571, Internet Engineering Task Force, Janeiro 2010. http://tools.ietf.org/html/rfc5751; acedido em 30 de Setembro de 2011.

[36] Wolfgang Rankl e Wolfgang Effing. *Smart Card Handbook*. John Wiley & Sons, Inc., New York, NY, USA, 3 edição, Dezembro 2003.

[37] Suprateek Sarker e John D. Wells. Understanding mobile handheld device use and adoption. *Commun. ACM*, 46:35–40, Dezembro 2003.

[38] Bruce Schneier e James Riordan. A certified e-mail protocol. Em *14th Annual Computer Security Applications Conference. ACM*, páginas 347–352, 1998.

[39] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, e Shlomi Dolev. Google Android: A State-of-the-Art Review of Security Mechanisms. *CoRR*, abs/0912.5101, Dezembro 2009.

[40] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, e Shlomi Dolev. Google Android: A State-of-the-Art Review of Security Mechanisms. *CoRR*, abs/0912.5101, 2009.

[41] John H. Shannon e David A. Rosenthal. Electronic mail and privacy: Can the conflicts be resolved? *Business Forum*, 18(1/2):31, 1993.

[42] Alex Shye, Benjamin Scholbrock, Gokhan Memik, e Peter A. Dinda. Characterizing and modeling user activity on smartphones: summary. Em *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '10, páginas 375–376, New York, NY, USA, 2010. ACM.

[43] UPU. *Electronic PostMark (EPM) Interface Specification (S43-3)*. UNIVERSAL POSTAL UNION, Março 2006.

[44] UPU. The Digital Postmark: Security for Cyberspace Mail, 2008.

[45] José M. Valença. Segurança Informática, Maio 2005.

[46] Timothy Vidas, Daniel Votipka, e Nicolas Christin. All Your Droid Are Belong To Us: A Survey of Current Android Attacks. Em *Proceedings of the 5th USENIX Workshop on Offensive Technologies*, Agosto 2011.

[47] Jan Vossaert, Jorn Lapon, e Vincent Naessens. Developing secure Java Card applications. MSec Research Group, Junho 2010.

[48] Dan Wallach. Smartphone Security: Trends and Predictions. Rice University, Fevereiro 2011.

[49] Ran Wei. Motivations for using the mobile phone for mass communications and entertainment. *Telemat. Inf.*, 25:36–46, Fevereiro 2008.

[50] Annette Zimmermann, Carolina Milanesi, e George Shiffler. Forecast Analysis: Mobile Devices, Worldwide, 2008-2015, Update 3Q11. Relatório Técnico G00219350, Gartner, Setembro 2011.