Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2014

Luís Miguel Miranda da Silva

Sistema de autenticação usando Android e NFC

"Nem todos os que tentaram conseguiram,
mas todos os que conseguiram tentaram."

— Anónimo

Luís Miguel Miranda da Silva

Sistema de autenticação usando Android e NFC

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica de Doutor André Venturada Cruz Marnoto Zúquete, Professor Auxiliar do Departamento de Engenharia Eletrotécnica, Telecomunicações e Informática da Universidade de Aveiro e de Doutor Diogo Nuno Pereira Gomes, Professor Auxiliar Convidado do Departamento de Engenharia Eletrotécnica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Tomás Oliveira e Silva
Professor Associado da Universidade de Aveiro

vogais / examiners committee

Manuel Eduardo Correia
Professor Auxiliar da Faculdade de Ciências da Universidade do Porto

André Ventura da Cruz Marnoto Zúquete
Professor Auxiliar da Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

No plano de estudos, a dissertação é mais uma unidade curricular. Para mim é mais do que isso, é o culminar de um percurso académico de cinco anos de inúmeras experiências. Por isto, são muitos aqueles a quem teria de dirigir os meus agradecimentos, mas como o espaço é curto vou apenas referenciar aqueles que estiveram mais presentes nesta reta final.

Em primeiro lugar, quero agradecer aos meus ídolos, os meus pais. Ao meu pai, Luís, por desde cedo ser um exemplo de sucesso, força de vontade e profissionalismo. À minha mãe, Palmira, por sempre me ter apoiado incondicionalmente e ter acreditado sempre em mim, mesmo quando eu próprio não o fazia. Espero um dia ter a oportunidade de retribuir tudo o que fizeram por mim. À minha namorada, Catarina, por me ter aturado desde sempre, pela companhia, compreensão dos meus desabafos e pelo carinho. Obrigado aos três por não me terem deixado atirar a toalha ao chão quando essa me parecia a única solução.

Não menos importantes, obrigado ao meus orientadores. Agradeço ao Professor Zúquete a paciência e disponibilidade de me ter recebido, vezes sem conta, no seu gabinete, mesmo quando eu só precisava de desabafar. Tenho de agradecer também as vezes em que sentou ao meu lado para me ajudar, quando não tinha nenhuma obrigação de o fazer. Ao Professor Diogo agradeço a persistência de, em todos os momentos, querer levar este trabalho sempre mais além. Obrigado por, naquele dia de Dezembro, quase me ter obrigado a explorar o modo de emulação de cartões no Android que, sem dúvida, fez toda a diferença para o sucesso deste projeto. Aos dois, obrigado pelo voto de confiança, pela motivação, orientação e oportunidade de trabalhar convosco, foi um prazer.

Ao meu colega e amigo de laboratório, Miguel (Mrock), obrigado por tudo. A ajuda, a motivação, os almoços e cafés e as infindáveis brincadeiras são algo que guardarei comigo. Ao Telmo, eterno colega de trabalho e amigo que me acompanhou desde o início do percurso académico, obrigado por, em todos os momentos, me convenceres que "o que é preciso é calma, as coisas nunca estão tão más quanto parecem". Ao Tiago (Moltos) agradeço a sua capacidade de me abstrair das preocupações com o trabalho, afinal "temos mais que fazer".

A estes e a todos os outros que fizeram parte disto, obrigado. Isto também é vosso.

Resumo

A autenticação por RFID é um processo presente no quotidiano dos cidadãos desde há muitos anos, suportando lógicas de negócio tão variadas como transportes, fidelização, bilhética, entre outros. No entanto, apesar de massificado e aceite, este sistema tem dois problemas óbvio: o de impingir ao utilizador um cartão de proximidade por cada prestador de serviços onde este se queira autenticar e a facilidade de personificação em caso de roubo dos cartões.

Paralelamente à utilização do RFID para autenticação, surgiu a tecnologia NFC que permite comunicações de curto alcance. O primeiro conjunto de especificações para o NFC surgiu em 2006, pelo que, à data de escrita deste documento, já tem perto de uma década de existência. No entanto, só recentemente o NFC começou a ser uma alternativa válida aos cartões de proximidade RFID para processos de autenticação uma vez que a sua presença nos smartphones é cada vez maior.

Neste projeto foi estudada uma arquitetura composta por utilizadores, prestadores de serviços, um ponto de registo e certificação e múltiplos pontos de confiança. A autenticação dos utilizadores nos prestadores de serviços é feita com uma aplicação desenvolvida para equipamentos Android com interface NFC perante um pórtico da entidade prestadora de serviços. O registo de utilizadores é feito no ponto de registo e certificação e permite um único par de credenciais para todo o universo institucional. Os prestadores de serviços federam-se no ponto de registo e certificação, que funciona também como uma CA, e associam-se a um ponto de confiança. Os utilizadores podem, a qualquer momento, revogar cartões com um simples clique.

A implementação desta arquitetura permite não só reduzir custos relacionados com a emissão de cartões como também agilizar o processo de revogação dos cartões em caso de roubo. Para além disso, dispensa o transporte diário de todos os cartões utilizados em processos de autenticação.

Abstract

From many years, the authentication process by RFID is present in the daylife of citizens, supporting business logic as varied as transportation, loyalty, ticketing and others. However, although mass-produced and accepted, this systeam has two obvious problems: (i) people carry one proximity card per service and (ii) it's easy for an attacker to purloin someone's identity if his wallet was stolen.

Near Field Communication (NFC) is another technology appeared alongside RFID for authentication purposes, in market since 2006. With the increasing demand of Smartphone users in present era, NFC has become a valid alternative to the proximity cards for authentication processes.

In this project, it was studied an architecture composed by users, service providers, point of sign and accreditation and multiple trust points. To accomplish this purpose, an application providing users authentication in the service providers was developed running on an Android device enabled with NFC, developed for this purpose. The users registration was made at the point of sign and accreditation and allows them to use an unique pair of credentials for all the institucional universe. Service providers are federated at the point of sign and accreditation, which also works as a CA, and are associated to a trust point. This makes users free to revoke their cards with a simple click, at any moment.

The development process of the proposed architecture allows not only to reduce the production costs related with the proximity cards but also to streamlining the revoking process in case of stealing/theft. In addition, it waives the necessity of carrying proximity cards in the daily life.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ACK *Acknowledge*

AID *Applicadion IDentifier*

ANR *Application Not Responding*

APDU *Application Protocol Data Unit*

API *Application Programming Interface*

ART *Android RunTime*

CA *Certification Authority*

CLA *Class byte*

CSR *Certificate Signing Requests*

DAC *Discretionary Access Control*

DF *Directory File*

DH *Diffie-Hellman*

DSA *Digital Signature Algorithm*

EEPROM *Electrically-Erasable Programmable Read-Only Memory*

EE *Enterprise Edition*

EF *Elementary File*

HCE *Host Card Emulation*

HTTP *HyperText Transfer Protocol*

HTTPS *HyperText Transfer Protocol*

I/O *Input/Output*

I2C *Inter-Integrated Circuit*

IANA *Internet Assigned Numbers Authority*

IDE *Integrated Development Environment*

INS *Instruction byte*

ISO *International Organization for Standardization*

JAR Java ARchive

JDBC *Java DataBase Connectivity*

JSON *JavaScript Object Notation*

JVM *Java Virtual Machine*

MAC *Mandatory Access Control*

MF *Master File*

NDEF *NFC Data Exchange Format*

NFC *Near Field Communication*

OTA *Over The Air*

P1-P2 *Parameter bytes*

P2P *Peer-to-Peer*

PAP *Physical Access Point*

PDC *Proximity Coupling Device*

PICC *Proximity Integreted Circuit Card*

PIN *Personal Identification Number*

RBAC *Role-based Access Control*

REST *REpresentational State Transfer*

RFID *Radio-Frequency IDentification*

SD *Secure Digital*

SDK *Software Development Kit*

SE *Standard Edition*

SGBG Sistema de Gestão de Base de Dados

SHA *Secure Hash Algorithm*

SIM *Subscriber Identity Module*

SMS *Short Message Service*

SNEP *Simple NDEF Exchange Protocol*

SPI *Serial Peripheral Interface*

SQL *Structured Query Language*

SSL *Secure Sockets Layer*

SW1-SW2 *Status bytes*

TCP *Transmission Control Protocol*

TrEE *Trusted Execution Environment*

URL *Uniform Resource Locator*

UUID *Universally Unique IDentifier*

XML *eXtensible Markup Language*

Lista de Extratos

Capítulo 1

Introdução

1.1 Enquadramento

A tecnologia RFID [8] permite, de forma automática, fazer a identificação de dispositivos denominados por marcas, utilizando ondas rádio para o efeito. Cada marca tem armazenado em si um conjunto de dados em que, dependendo do tipo desta, alguns deles podem ser reescritos. De modo a proceder-se à identificação é necessário que a marca, no conjunto de dados que transporta, inclua um identificador único que permita que esta seja distinguida das demais. As marcas podem ou não possuir uma fonte de energia própria para efeitos de comunicação e computação local. Denominam-se por marcas passivas ou semi-passivas as que requerem um fornecimento de energia externa suplementar para poderem cumprir a sua função.

O processo de identificação requer um dispositivo capaz de ler os dados presentes nas marcas, normalmente designado por leitor. Este leitor tem a capacidade de gerar um campo eletromagnético que terá o objetivo de ser a fonte de alimentação das marcas passivas ou semi-passivas. Dependendo das ondas emitidas pelo leitor, a área abrangida pelo campo eletromagnético poderá variar bastante, desde poucos centímetros a dezenas de metros. Assim que uma marca penetra o campo eletromagnético anteriormente referido é detetada pelo leitor, permitindo que este último aceda aos seus dados e, consequentemente, possa com eles proceder ao processo de identificação.

Esta tecnologia, à data da escrita desta dissertação, já não é passível de ser considerada recente. Na verdade, a sua utilização verifica-se por todo o mundo devido à flexibilidade, simplicidade e rapidez com que é efetuado todo o processo de identificação.

A utilização de cartões de passe ou viagens nos transportes públicos é um caso prático da tecnologia RFID: o cartão contém uma marca com um identificador único que permitirá ao leitor validar a elegibilidade do seu possuidor ingressar na rede de transportes. Outro caso prático da utilização desta tecnologia é o controlo de *stocks* em armazéns [1]: cada produto possui uma marca com um identificador que permitirá ao leitor monitorizar, em tempo real, as quantidades disponíveis de cada produto.

Apesar da simplicidade inerente à tecnologia, esta contém algumas desvantagens. Voltando ao caso prático dos cartões que contêm marcas, é de notar que não existe nenhum mecanismo de segurança inerente ao processo de identificação do seu possuidor. Assim, é possível a qualquer pessoa que não o real dono do cartão fazer a sua personificação [9] após o roubo.

[1] `http://wtnnews.com/articles/8215/`

Mais, apesar de ser possível revogar um cartão em caso de roubo, este processo não é, de todo, nem simples nem ágil. No caso do roubo de uma carteira, composta por um conjunto de cartões, será necessário entrar em contacto com cada uma das entidades emissoras dos cartões furtados a fim de os invalidar. Por fim, existe ainda a desvantagem da necessidade de transportar um cartão por cada serviço onde nos queiramos identificar.

Paralelamente, a tecnologia RFID evoluiu e levou ao surgimento de um novo protocolo de comunicação a curta distância (4 cm, aproximadamente) [10] designado NFC (Near Field Communication).

Um caso prático da utilização de NFC em dispositivos móveis é o Google Wallet [11]: uma carteira eletrónica onde os possuidores de um smartphone Android com tecnologia NFC específica têm a possibilidade de efetuar pagamentos aproximando o seu dispositivo de um terminal de pagamento com interface NFC.

Apesar da grande adesão e aceitação dos smartphones Android por parte dos consumidores, apenas os dispositivos mais caros, tipicamente os topo de gama dos fabricantes destes equipamentos, implementam uma interface NFC. No entanto, à data da escrita desta dissertação, começa a verificar-se a inclusão deste tipo de tecnologia em *smartphones* economicamente mais acessíveis, pelo que dentro de algum tempo é natural que a tecnologia NFC seja uma realidade no dia-a-dia das pessoas [12]. Isto torna estes aparelhos, que se tornaram omnipresentes no quotidiano dos cidadãos, em plataformas passíveis de substituírem os vulgares cartões utilizados na identificação de pessoas.

1.2 Objetivos

De acordo com a Nielsen [2], uma empresa focada em estudos de mercado e com sede em Nova Iorque, os *smartphones* tiveram uma forte penetração junto da população americana, representando cerca de 61% dos clientes de serviços de comunicações móveis. Este tipo de produto afirmou-se, desde Março de 2012, como o tipo de dispositivo móvel dominante nos Estados Unidos da América. É um objeto que faz parte do quotidiano das pessoas. De notar ainda que, à data da escrita desta dissertação, os *smartphones* apresentam quer um poder computacional quer uma facilidade de integração com a camada de rede assinaláveis, proveniente dos recursos de hardware e software que disponibilizam e que lhes confere a capacidade de correr aplicações com um elevado grau de complexidade[3].

Esta dissertação tem por objetivo resolver os problemas anteriormente referidos relativamente aos cartões RFID, integrando os *smartphones* em parte da solução, quer pela sua larga aceitação e naturalidade de utilização junto da população, quer pelas suas características ao nível computacional. Outros fatores importantes a ter em conta na adoção de novas soluções são o custo que as mudanças e adaptações acarretam e a usabilidade delas. Por isso, tanto o baixo custo de produção como a facilidade de utilização quer por parte dos utentes quer das instituições foram pontos mantidos em mente durante o trabalho. Assim, estudou-se uma infraestrutura de autenticação onde os cartões foram virtualizados com recurso a *smartphones* equipados com NFC, o que elimina custos de produção e emissão de cartões físicos, e os pórticos implementados com recurso a um micro-controlador Arduino com um módulo de interface NFC e um Raspberry Pi. Mais ainda, para facilitar a gestão de cartões virtuais (geração, clonagem, revogação, etc.), foi considerada a federação de várias entidades que os fornecem

[2] `http://mashable.com/2013/06/06/smartphones-61-percent/`

[3] `http://www.techautos.com/2010/03/14/smartphone-processor-guide/`

sob um serviço de gestão de cartões virtuais. Este serviço não é impeditivo de que as entidades ofereçam, através dos seus próprios recursos, os seus serviços de gestão de cartões e utentes. O principal objetivo no desenvolvimento deste serviço prende-se com o facto de existirem instituições que não possuem portais próprios para o efeito e em que uma solução deste tipo prestada por terceiros faz sentido.

A infraestrutura considerada assenta na emissão de cartões virtuais aos utilizadores, por parte das instituições federadas, que podem ser descarregados para uma aplicação Android a correr num dispositivo móvel com interface NFC e, consequentemente, participarem num processo de autenticação que envolve a comunicação com um pórtico representativo de uma instituição. O NFC, pelo baixo alcance suportado [10], não acarreta consigo alguns problemas de segurança presentes no RFID [13], tais com a impossibilidade de detetar a presença de marcas sem que o possuidor tenha conhecimento ou a intercepção de comunicação por parte de um atacante. Mais, quando um equipamentos Android é colocado em modo de emulação de cartões, o identificador da marca é aleatoriamente gerado em cada utilização [14] o que invalida processos de rastreamento de utilizadores. Os cartões virtuais, assinados digitalmente por quem os emite, serão posteriormente utilizados para identificação e autenticação dos utilizadores perante as instituições. Pretende-se, deste modo, que a aplicação Android substitua, na totalidade, os vulgares cartões de identificação que transportamos, funcionado deste modo como uma carteira virtual. Mais, surge aqui a possibilidade de acrescentar uma camada importante de segurança que proteja os cartões, por exemplo, em caso de roubo do dispositivo. Para isso, o utilizador terá de associar um padrão de segurança à carteira virtual e terá de o introduzir para a desbloquear, podendo posteriormente utilizar os seus cartões num processo de autenticação. Será ainda possível tanto à instituição como ao utilizador revogar um ou mais cartões emitidos e/ou associados a carteiras virtuais de forma ágil através de aplicações desenvolvidas para efeito.

1.3 Contribuição

De forma a atingir os objetivos a que a dissertação se propõe, foi desenvolvida uma infraestrutura de autenticação flexível, tendo em conta as necessidades reais da sociedade e da indústria, assente sobre um sistema de informação cuidadosamente estruturado. Esta solução pretende resolver, na totalidade, problemas anteriormente identificados sem que dela resulte um prejuízo no quotidiano dos atuais utilizadores de cartões de identificação. Por outras palavras, pretende-se que a sua adoção seja o mais natural possível, aproveitando hábitos e costumes presentes na sociedade e na indústria.

De modo a que a solução não represente nenhum problema de segurança de dados, quer para instituições, quer para utilizadores e para assegurar a integridade das comunicações entre estes, foi introduzido na solução um elemento denominado de ponto de confiança onde são armazenados dados relativos a cartões de utilizadores. Assim, é disponibilizado um conjunto de serviços úteis de entre os quais a emissão e aprovação de cartões virtuais. Para além disto, foi ainda tido em conta um elemento denominado de ponto de registo e certificação que oferece serviços para registo de utilizadores e federação de instituições a um ponto de confiança. Isto possibilita que os clientes, com um único registo no ponto de registo e certificação, possam interagir com um número ilimitado de pontos de confiança e instituições registadas no serviço.

No âmbito desta dissertação, foram desenvolvidos vários protocolos, aplicações e bibliotecas que se complementam para fornecer a infraestrutura que visa a autenticação. Para os

utentes foram desenvolvidas (i) uma aplicação Java Swing para registo de utilizadores e para que estes pudessem efetuar a gestão das suas carteiras virtuais e (ii) uma aplicação Android que representa uma carteira virtual para onde é possível descarregar cartões e, utilizando a interface NFC do dispositivo, efetuar um processo de autenticação perante uma instituição. Para as instituições, foi desenvolvida (iii) uma biblioteca Java, disponibilizada sob a forma de um JAR, que lhes permite facilmente gerir cartões dos utilizadores associados. A opção pelo desenvolvimento de uma biblioteca, em detrimento de uma aplicação, passa por dar liberdade às instituições para produzirem, da forma que acharem mais adequada, uma solução que possa, caso necessário, ser integrada com os seus sistemas de informação atuais. Ainda assim, foi também desenvolvida (iv) uma aplicação na tecnologia Java Swing para permitir a gestão de cartões e utentes às instituições que pretendam uma solução de controlo de acessos completamente disponibilizada por terceiros e que servirá, no âmbito desta dissertação, como prova de conceito. No que toca ao processo de autenticação entre utilizador e instituição, foi estudado e definido um protocolo seguro e robusto, desenvolvido sobre o ISO 7816-4 [15] (camada aplicacional). Por fim, foi desenvolvido (v) um pórtico de suporte ao processo de autenticação, que comunica com os *smartphones* via NFC. Para tal foram utilizados três componentes: um micro-controlador Arduino Uno, um módulo (shield) NFC da SeedStudio e um Raspberry Pi. A escolha destes equipamentos deveu-se sobretudo ao baixo custo de aquisição. No entanto, a assegurada compatibilidade do micro-controlador com o módulo NFC e o facto do fabricante do último disponibilizar uma biblioteca de programação foram também fatores decisivos.

Tendo em conta um estudo em 2013, levado a cabo pelo Business Insider [4], a adoção de smartphones pelos utilizadores representa já cerca de 22% da população mundial. Outro fato interessante é o crescimento do mercado de *tablets*, que representavam, à data do estudo, cerca de 6% da população mundial. Já a Mashable [5], em Fevereiro de 2012, lançava uma notícia que indicava que no fim daquele ano deveriam existir mais *smartphones* do que pessoas, sendo que em 2016 o número de equipamentos deste tipo se deveria situar nos 10 biliões. Torna-se, por isto, natural concluir que existem pessoas com mais de um dispositivo móvel com um poder computacional elevado. Assim, na modelação da solução, foi tido em conta que poderiam existir utilizadores para quem fizesse sentido ter mais do que uma carteira virtual. Na solução onde são utilizados cartões físicos para autenticação, é comum que a cada utilizador seja atribuído um, e um só, cartão por instituição onde se pretenda autenticar. A clonagem de cartões pode, em algumas situações, representar um problema grave de segurança para a instituição, pois permitirá, por exemplo, usos abusivos (e.g. bilhética). No entanto, existem ambientes de utilização onde a clonagem de cartões não é um entrave (e.g. controlo de acesso a infraestruturas). Assim, a possibilidade de clonagem do mesmo cartão para várias carteiras virtuais de um utilizador fica ao critério da instituição no ato de credenciação. De notar que a possibilidade de clonagem traz uma vantagem clara para o utilizador em caso de roubo de um dispositivo, pois, caso possua mais do que uma carteira virtual, pode continuar a autenticar-se perante as instituições que permitem clonagem de cartões emitidos.

O universo institucional é muito vasto pelo que a lógica pós-autenticação vai, inevitavelmente, variar. Mesmo perante a mesma instituição, utilizadores diferentes podem ter permissões diferentes. É, então, necessário que cada cartão tenha a si associado um perfil de utilização consoante o utente a quem foi emitido. Assim, em cada cartão virtual está disponível uma zona de memória definida pela instituição no momento da emissão do cartão que pode servir,

[4] `http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10`
[5] `http://mashable.com/2012/02/14/more-smartphones-than-humans/`

por exemplo, para colocar uma fotografia do dono ou para armazenar um perfil de utilização. Esta área de memória, por questões de segurança, não é passível de ser reescrita. Ao contrário do que é comum na bilhética, em que as áreas de memória do cartão são reescritas com frequência, num ambiente de controlo de acessos as zonas de memória permanecem inalteradas, pelo que a memória dos cartões virtuais permanecerá constante após o processo de emissão.

Como referido no início desta dissertação, os típicos cartões físicos utilizados em processos de autenticação não garantem, em caso de roubo, nenhuma segurança ao lesado. Para resolver esta situação, o utilizador pode a qualquer altura revogar um clone, um conjunto de clones ou até todos os clones dos seus cartões fazendo recurso à aplicação desenvolvida em Java Swing. Isto invalida, no momento, todos os clones válidos selecionados presentes nas carteiras virtuais, evitando a possibilidade de personificação. Existe ainda uma camada de segurança adicional, que pretende proteger o utilizador entre o período em que o dispositivo é roubado e o momento em que os clones são revogados. Para tal, a aplicação permite ao utilizador definir um padrão gráfico de bloqueio, que tem sempre de ser introduzido para utilizar a aplicação. Deste modo, mesmo que os cartões não estejam revogados, o atacante não se poderá personificar com os clones presentes na carteira virtual, pois é-lhe colocada uma barreira de segurança. Para além do padrão de segurança, é necessário introduzir a palavra-passe na aplicação móvel em cada processo de clonagem de cartões, pelo que o atacante terá mais este obstáculo para contornar.

Por fim, não esquecendo as questões de privacidade, foi mantida uma base de dados local às entidades institucionais dos utentes e cartões virtuais emitidos. Esta base de dados é atualizada periodicamente juntamente do serviço central. Assim, quando um utente despoleta um processo de autenticação junto de um pórtico, os sistemas de informação da instituição não têm de verificar a validade do cartão junto do ponto de confiança tendo, para esse efeito, a base de dados local. Note-se que uma verificação da validade do cartão em tempo real, apesar de possível, acarreta consigo pontos a considerar como atrasos e disponibilidade da rede. Para além disso, e não menos importante, a verificação em tempo real permitiria ao ponto central criar um padrão de utilização dos cartões dos utilizadores, o que colocaria em causa a questão da privacidade, tema de discussão deste parágrafo.

Para melhor compreensão da arquitetura proposta, esta foi detalhada no capítulo Arquitetura e representada na figura 4.1.

1.4 Estrutura da dissertação

O presente documento, intitulado "Sistema de autenticação usando Android e NFC", aborda projetos existentes que recorrem ao RFID ou ao NFC para acelerar processos do quotidiano tais como pagamentos, autenticação, entre outros e propor uma solução de autenticação com equipamentos Android com interface NFC.

A estrutura do documento possui seis capítulos, para além do atual:

1. Estado da Arte;
2. Contexto;
3. Arquitetura;
4. Implementação;
5. Análise de segurança;

6. Casos de uso;

7. Considerações finais.

No capítulo 2 são abordadas algumas soluções existentes na área do RFID e NFC com vista a agilizar processos comuns do quotidiano. No capítulo 3 é feita uma análise aos recursos de *software* e *hardware* utilizados na implementação da solução proposta. De seguida, no capítulo 4, é proposta uma arquitetura e um protocolo de autenticação para atingir o objetivo proposto, cujas especificações e detalhes são abordados no capítulo sobsequente. Por se tratar de um projeto na área da autenticação, estão envolvidas questões na área da segurança e privacidade dos dados, pelo que foi feita uma análise de segurança no capítulo 6. Existe ainda uma análise a casos de uso passíveis de implementarem a solução desenvolvida no âmbito desta dissertação no capítulo 7. Por último, no capítulo 8 é feita uma análise geral do trabalho desenvolvido ao longo desta dissertação e daquilo que poderia ser feito no futuro.

Capítulo 2

Estado da Arte

O presente capítulo tem por objetivo dar a conhecer não só aquilo que já foi desenvolvido no âmbito da autenticação utilizando a tecnologia NFC mas também as aplicações desta tecnologia quando utilizada em conjunto com smartphones. Pretende-se aqui identificar quais os pontos fortes de cada solução e quais os pontos críticos que, de alguma forma, colocam em causa a fiabilidade do processo de autenticação. Deste modo será possível possível perceber quais os pontos passíveis de serem melhorados e, consequentemente, acrescentar qualidade à solução desenvolvida e agregar valor no âmbito desta dissertação.

2.1 Sistemas de suporte a cartões de fidelização

Esta secção pretende dar a conhecer soluções de carteiras virtuais já existentes, identificar possíveis problemas e, se possível, discutir soluções.

2.1.1 MEO CardMobili

Descrição

O MEO CardMobili disponibiliza uma carteira virtual para *smartphones*, sob a forma de uma aplicação, cujo objetivo é "substituir todos os cartões do utilizador" [16]. Existe um conjunto de cartões de identificação pessoal como o cartão do cidadão, carta de condução, etc e um outro conjunto de cartões associado a superfícies comerciais, também conhecidos como cartões de fidelização. No caso destes últimos, o utilizador é ainda notificado com promoções das superfícies comerciais relativas aos cartões que possui na carteira virtual.

Problemas

Todo o processo de associação do cartão à carteira virtual é efetuado pelo utilizador. De notar que não existe qualquer tipo de intervenção da entidade emissora do cartão no processo de associação deste à carteira virtual do utente. Assim, por não existir nenhum processo de validação dos cartões presentes na carteira, o utilizador nunca poderá fazer prova que é o real dono do cartão em questão.

Conclusões

O problema anteriormente mencionado seria facilmente resolvido com a participação da entidade emissora do cartão no processo de associação do cartão à carteira virtual, tendo a responsabilidade de validar ou não a ação do utente. No caso de a ação ser validada, a emissão de uma assinatura digital, à semelhança do que foi feito no âmbito desta dissertação, seria suficiente para a entidade emissora confiar num cartão que lhe fosse apresentado.

2.2 Sistemas de controlo de acessos utilizando NFC e/ou RFID

Esta secção pretende abordar alguns sistemas de controlo de acessos que utilizem o NFC como meio de comunicação.

2.2.1 SmartTokens: Delegable Access Control with NFC-enabled Smartphones

Descrição

A solução abordada em [1], à semelhança do acontece com esta dissertação, pretende oferecer uma solução de controlo de acessos baseada em *smartphones* capazes de realizar comunicação via NFC. Pretende oferecer, para além de um sistema de controlo de acessos genérico, uma solução que permita ao utilizador delegar parte dos seus direitos a outros utilizadores sem que para isso seja necessário a intervenção da entidade lhos atribuiu. Da contribuição dada pelos autores do artigo, destacam-se:

- **arquitetura de segurança multinível** - pretende proteger as credenciais dos utilizadores assim como os segredos inerentes ao processo de autenticação. Combina um ambiente de execução confiável (do inglês TrEE - *Trusted Execution Environment*) com um software de controlo de acessos ao TrEE.

- **sistema de delegação de direitos** - permite aos utilizadores delegarem parte dos seus direitos de acesso a terceiros sem necessidade de intervenção da entidade que os definiu.

- **isolamento da aplicação** - para além da segurança oferecia pela arquitetura mencionada acima, os autores pretenderam, concetualmente, vincular as operações NFC ao TrEE.

Objetivos e requisitos

Segundo o artigo, o objetivo de toda a solução passa por invalidar ataques que permitam a atacantes autenticarem-se num fornecedor de serviços por terceiros, ou seja, personificação. Segundo os mesmos, o desenho do protocolo deve ser capaz de invalidar posteriores ataques contra o processo de autenticação. Quanto à arquitetura de segurança, é referido que deve ter a seu cargo a responsabilidade de assegurar que o atacante não é capaz de aceder às credenciais do utilizador no dispositivo assim como não é capaz de as modificar.

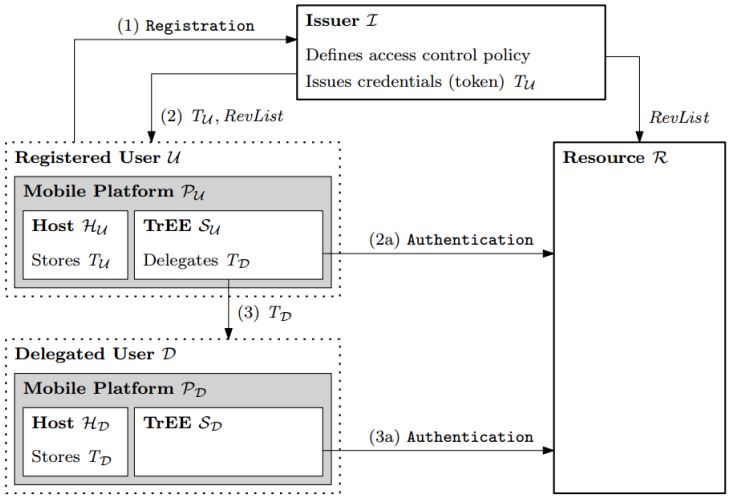

Figura 2.1: Arquitetura da solução SmartTokens, retirada do artigo [1]

Funcionamento

O funcionamento deste sistema pode ser observado na figura 2.1.

A entidade emissora (*Issuer*) é responsavel por definir políticas de controlo de acesso a recursos aos utilizadores registados. Essas políticas são armazenadas e é emitido um *token* ao utilizador registado relativo a elas. O utilizador registado, na posse do *token* de acesso, pode agora utilizá-lo para aceder a um recurso controlado por um política de acesso ou para delegar os seus direitos, ou parte deles, a outros utilizadores. Os utilizadores que possuam *tokens* delegados podem utilizá-los para efeitos de autenticação, mas não podem delegar os direitos que esse *token* lhe concede a outros utilizadores. Por fim, a solução providencia ainda um processo de revogação de *tokens* que pode ser desencadeado pela entidade emissora. Qualquer *token* pode ser invalidado mas note-se que a revogação de um deles que tenha sido atribuído a um utilizador registado invalidada também todos os *tokens* derivados deste através de processos de delegação de direitos de acesso.

2.2.2 Controlo de acessos usando Android e NFC

Descrição

A dissertação [12] abordada neste tópico foi desenvolvida no ano letivo 2012/2013 e tinha por objetivo "desenvolver um protótipo funcional de um sistema de controlo de acessos físicos" utilizando, na solução, um *smartphone* Android que permitisse tirar partido da tecnologia NFC "como meio de identificação/autenticação". Paralelamente, era pretendido "validar o NFC como alternativa tecnológica para aplicações de autenticação em sistemas de controlo de acessos físicos". A solução era destinada a um ambiente doméstico para abertura de portas recorrendo, para isso, a um sistema de fechaduras eletrónicas.

Por forma a atingir os objetivos, foi pensada uma arquitetura composta por três módulos distintos, que comunicam entre si no processo de autenticação: um dispositivo Android de suporte a uma aplicação para dispositivos com interface NFC, um PAP (*Physical Access Point*) e um módulo central. Apresenta-se de seguida as características de cada um dos módulos:

- **Módulo central** - oferece capacidades de gestão de utilizadores e pontos de acesso físicos. Para isso, todas as entidades que se pretendam autenticar num PAP têm de estar inevitavelmente registadas na base de dados local presente neste módulo. A cada entidade está ainda associado um PIN de PAP e um conjunto de permissões que lhe permitirá, segundo o autor, participar mais tarde no processo de autenticação. Este módulo encontra-se ainda dotado de um dispositivo de comunicação sem fios que deverá comunicar com recurso à tecnologia ZigBee de modo a disponibilizar a interoperabilidade com outros componentes envolvidos na implementação da solução.

- **Dispositivo móvel** - este módulo, desenvolvido sob para plataforma Android, disponibiliza, sob a forma de uma aplicação simplista, a possibilidade do utilizador introduzir um PIN que lhe permitirá desbloquear a fechadura da porta pela qual o PAP está responsável. Tem, para isso, depois de introduzido o PIN, de aproximar o *smartphone* da interface NFC disponibilizada pelo PAP. No caso de uma comunicação sobre NFC bem sucedida entre os dois elementos referidos, será enviada uma mensagem NDEF, utilizando o protocolo SNEP, que contém, num dos seus registos, o código com o qual o utilizador pretende abrir a porta.

- **PAP** - este módulo é responsável por orquestrar as interações entre os dois módulos referidos anteriormente. Para tal, conta com um microcontrolador capaz de operar duas interfaces de comunicação: uma NFC para comunicar com o dispositivo móvel e uma ZigBee para comunicar com o módulo central. O PAP conta ainda com uma EEPROM onde são armazenados PINs válidos e que permitirão, aos utilizadores, desbloquear a porta a cargo deste. Quando o PAP recebe uma mensagem, pela interface NFC, extrai-lhe o PIN e verifica a sua existência na EEPROM. No caso deste não existir, ou seja, não ser válido, o PAP emitirá quer uma sinalização sonora, quer uma sinalização visual que permitirão ao utilizador perceber o insucesso do processo. Caso contrário, serão emitidas sinalizações sonora e visual de forma a dar *feedback* ao utilizador do sucesso da operação, a fechadura da porta é desbloqueada e, por fim, será enviada uma mensagem ao módulo central, utilizando a interface ZigBee, que servirá para efeitos de histórico.

Problemas

O facto de apenas ser necessária a validação de um PIN para garantir o desbloqueio da fechadura coloca problemas ao nível da segurança do sistema implementado. O processo de autenticação contempla apenas a troca de uma mensagem entre o dispositivo móvel e o PAP, sendo que a mensagem tem origem no primeiro. Esta mensagem contém um partilhado entre o utilizador e o PAP, o PIN já anteriormente referido. Deste modo, qualquer atacante que intecepte a mensagem conseguirá, sem dificuldade, reproduzi-la num outro dispositivo e abrir uma porta controlada pelo PAP. De destacar ainda que, como não foi tirado proveito das capacidades computacionais do terminal Android para implementação, por exemplo, de um protocolo desafio resposta, não será sequer necessário a utilização deste tipo de dispositivos para realizar um ataque. Na verdade, o dispositivo Android é perfeitamente passível de ser substituído por um teclado numérico físico. Por fim, uma vez que o PIN é apenas composto por quatro dígitos, o que resulta num universo de dez mil possibilidades, posso ainda concluir que esta solução é também bastante vulnerável a ataques de força bruta uma vez que as mensagens NDEF, depois de estabelecida a ligação NFC entre o dispositivo e o PAP, podem

ser enviadas repetidamente num curto espaço de tempo que resultará, em tempo útil, num ataque bem sucedido.

Apesar de o autor referir a existência de quer de um nome de utilizador quer de um conjunto de permissões para cada utilizador, estes dados não são utilizados no processo de autenticação. Aquando da receção da mensagem, via NFC, por parte do PAP apenas é feita a verificação da existência da password na EEPROM.

Soluções

Uma vez que foi feito recurso a um dispositivo Android e já que estes, devido aos recursos de processamento, comunicação e armazenamento que disponibilizam, permitem correr aplicações móveis com elevado grau de complexidade deveria ter sido implementado um protocolo que se revelasse robusto no que toca à segurança. Para isso e uma vez que a tecnologia NFC suporta comunicação ponto-a-ponto bi-direcional o recurso a um protocolo desafio resposta seria uma possível solução pois permitiria introduzir alguma entropia nas mensagens trocadas evitando ataques de repetição. Outra solução para este problema seria o uso de PINs descartáveis. Neste caso, mesmo que o atacante interceptasse a mensagem de autenticação enviada do dispositivo móvel para o PAP, não lhe seria possível fazer um ataques de repetição já que a senha enviada perde a validade aquando da sua utilização.

2.2.3 Controlo de acessos usando uma marca RFID num telemóvel

A patente número US20060111053 A1[17], registada nos Estados Unidos da América, propõe uma solução baseada em controlo de acessos utilizando telemóveis, marcas RFID e uma operador de redes de comunicação móveis. A ideia principal é eliminar os cartões físicos e substituindo-os por uma marca RFID discreta presente no telemóvel. Segundo a patente em questão, os telemóveis são produtos utilizados como acessórios de moda, o que torna natural a sua integração no quotidiano das pessoas. O controlo de acessos é efetuado da seguinte forma: sempre que a marca é detetada por um leitor RFID, é enviada uma SMS pelo telemóvel onde ela está colocada para um conjunto de números pré-definidos pelo utilizador. A principal desvantagem desta solução é a dificuldade de integração com o hardware dos equipamentos já existente. Para além disso está em causa a privacidade do utilizador, uma vez que a identificação da marca que transporta consigo é efetuada de forma automática e sem os seus conhecimento e consentimento. Por fim, é necessária a cooperação da operadora de redes móveis para o envio das SMS, o que acarreta custos para o utilizador.

2.3 Sistemas de pagamento utilizando smartphones e NFC

À data de escrita desta dissertação existem várias soluções no mercado que permitem realizar pagamentos recorrendo a um smartphone com interface NFC. Serão abordadas as duas soluções com mais impato do mercado.

2.3.1 Microsoft Wallet

A Microsoft Wallet [18] pretende afirmar-se como uma alternativa válida aos tradicionais cartões de identificação, fidelização e cupões de desconto. Esta aplicação, introduzida na versão 8 do sistema operativo Windows Phone, permite:

- associar produtos da carteira virtual a aplicações para a realização de micro transações;
- gerir os métodos de pagamento nas lojas de aplicação e música disponibilizadas pelo sistema operativo;
- efetuar transações, usando a interface NFC dos dispositivos, em lojas que suportem o métodos de pagamento.

De modo a agilizar o processo de adoção da sua solução, a Microsoft disponibiliza uma API para que seja possível a terceiros integrarem as suas aplicações com esta carteira virtual, abrindo assim portas para a realização de transações de forma transparente. Para além disto, visando permitir a instalação da aplicação no máximo número de dispositivos possível, o elemento seguro escolhido, encarregue de efetuar as operações necessárias, localiza-se no cartão SIM do equipamento. Apesar dos esforços para facilitar a adoção da aplicação quer por utentes, quer por comerciantes, o número de utilizadores de equipamentos com o sistema operativo Windows Phone 8 é bastante reduzido quando comparado com os dois concorrentes diretos: Android e iOS. Segundo uma notícia avançada pela CNET [1], em 12 de Novembro de 2013, o sistema operativo Android representava mais de 80% do mercado de dispositivos móveis, ao passo que o Windows Phone estava abaixo dos 5%. Assim, apesar da solução ser interessante e bem pensada, a penetração e aceitação no mercado será difícil.

2.3.2 Google Wallet

A Google Wallet [18] é a solução da Google para responder à necessidade de um sistema de pagamento livre de cartões bancários baseado em dispositivos móveis com interface NFC. Foi apresentada em 2011 e, até à data de escrita desta dissertação, só se encontra disponível nos Estados Unidos.

Nesta solução o utilizador é responsável por definir que cartões bancários pretende associar à sua carteira virtual. Depois disso está elegível para realizar pagamentos, com recurso ao seu smartphone Android, em estabelecimentos comerciais que possuem um terminal disponibilizado pela Google.

Um dos principais problemas desta arquitetura é o facto de não bastar ter um dispositivo Android com interface NFC, é necessário que o aparelho disponha de um elemento seguro embebido no chip NFC. Consequentemente, o universo de dispositivos válidos para suportarem a aplicação é mais reduzido.

2.3.3 Outras soluções

De acordo com o autor da dissertação [19], existem outras soluções pagamento baseadas na arquitetura abordada nesta secção. Segundo o mesmo, os pagamentos dividem-se em 4 categorias:

- micro pagamentos: caracterizam pagamentos de, no máximo, 2 €.;
- macro pagamentos: caracterizam pagamentos superiores a 25 €.;
- pré pagamentos;

[1] `http://www.cnet.com/news/android-dominates-81-percent-of-world-smartphone-market/`

- pós pagamentos.

A solução desenvolvida na dissertação anteriormente mencionada é, também ela, uma alternativa às propostas das gigantes Google e Microsoft. De notar ainda que, à data de escrita desta dissertação, já é comum encontrar aplicações que promovem a fidelização de clientes a cadeias de lojas presentes nas grandes superfícies comerciais. Este tipo de aplicação promove o pagamento de produtos e serviços não só com dinheiro mas também com pontos e cupões.

2.4 Sistemas de bilhética utilizando smartphones e NFC

2.4.1 NFCTicketing

Esta é uma investigação [2] destinada ao mercado de bilhética, abordado nesta secção, e tem como principais objetivos reduzir a complexidade das soluções tradicionais e, em simultâneo, reduzir custos associados. Os elementos que compõem esta solução são:

- *smartphone* Nokia 6131 com interface NFC e elemento seguro;
- um *SmartPoster* contendo uma marca MIFARE;
- uma aplicação para o utente que lhe permite verificar adquirir bilhetes e verificar a sua validade;
- uma aplicação para o elemento seguro, onde são guardadas informações relativas ao bilhetes;
- um validador com um ecrã e interface NFC que permite aos utentes validar o seu bilhetes;
- um *smartphone* que permita verificar se a validação do bilhete foi efetuada.

A arquitetura da solução proposta está representada na imagem 2.2.

Ao utente é-lhe possível, recorrendo ao seu *smartphone*, ler as marcas presentes nos *SmartPosters* e adquirir bilhetes junto do servidor NFCTicketing. A comunicação entre o equipamento do utente o servidor de bilhetes é efetuada via OTA (*Over The Air*), pelo que o NFC é deixado de parte neste processo. A validação dos bilhetes adquiridos e feita, por NFC, numa comunicação entre o dispositivo do utente e o do validador. Por fim, de modo a ser possível à entidade emissora dos bilhetes verificar se o utente possui ou não um bilhete válido, existe uma aplicação instalada num *smartphone* que, através de uma comunicação NFC com o dispositivo do utente, o permite determinar.

Apesar de interessante, a solução está limitada ao mercado da bilhética, o que reduz de forma significativa os cenários onde pode ser aplicada. Outra questão passível de ser melhorada é a aquisição dos bilhetes que necessita da ação do utilizador junto de um *SmartPoster*.

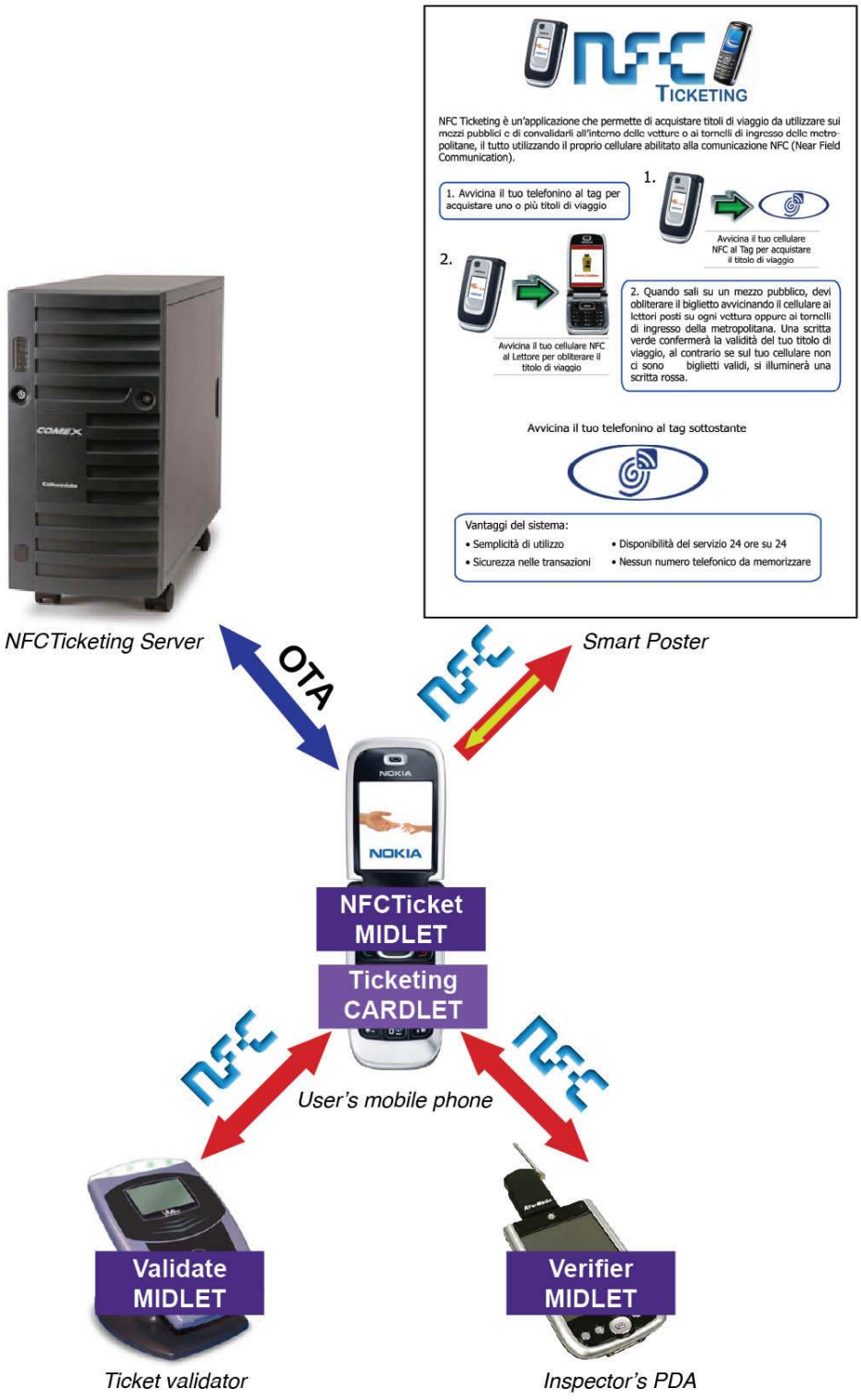

Figura 2.2: Representação da arquitetura da solução NFCTicketing, retirada do artigo [2]

Capítulo 3

Contexto

O presente capítulo pretende enumerar as tecnologias utilizadas no desenvolvimento da solução proposta nesta dissertação, onde é que elas foram utilizadas e, por fim, explicar o porquê da sua escolha em detrimento de outras.

O critério base de cada escolha foi, como indicado anteriormente nesta dissertação, o custo que ela acarretaria à solução final. O segundo critério utilizado foi a familiaridade com as linguagens de programação e plataformas, para que o desenvolvimento fosse agilizado.

3.1 NFC

3.1.1 Descrição

O NFC aparece como uma derivação da tecnologia de identificação por radiofrequências (RFID). Suporta comunicações, sem fios, a curtas distâncias num máximo de aproximadamente 4 cm [10]. O seu desenvolvimento partiu de uma cooperação entre Philips, Sony e Nokia que, no ano de 2004, fundaram o NFC Forum [1], grupo que desde então se "dedica a promover a segurança, usabilidade e popularidade" [20] desta tecnologia. O objetivo deste grupo é manter uma conjunto de normas standard, que deve ser respeitada por aqueles que implementam a tecnologia nos seus produtos e serviços, de modo a que a interoperabilidade entre dispositivos de fabricantes diferentes não seja comprometida.

O NFC pretende oferecer, acima de tudo, segurança na comunicação entre dois dispositivos eletrónicos. No entanto, pela sua sua simplicidade, a utilização desta tecnologia como meio de comunicação é também bastante intuitiva. Relativamente à segurança, devido ao alcance reduzido, torna-se muito difícil a um atacante intercetar comunicações. No que toca à usabilidade, esta fica assegura pela simplicidade inerente à tecnologia: os utilizadores de dispositivos NFC necessitam apenas de aproximar ou tocar os seus dispositivos para iniciar uma comunicação. Deste modo, a tecnologia suporta um conjunto de casos de uso elevado, pelo que os *smartphones* são um mercado apetecível para o desenvolvimento de aplicações, destacando-se exemplos como:

- descarregar conteúdos de um *SmartPoster*;
- trocar de cartões de visita entre *smartphones*;

[1] `http://nfc-forum.org/`

- efetuar pagamentos;
- acelerar emparelhamento Bluetooth;

Podem ainda destacar-se vantagens como:

- conhecimento, por parte do utilizador, que o dispositivo está a ser alvo de comunicação, pois é o primeiro que tem de proporcionar as condições para tal;
- compatibilidade com os cartões de identificação presentes no RFID;
- baixo consumo de energia;
- praticabilidade e rapidez que derivam inexistência da necessidade de autenticação entre os dispositivos envolvidos na comunicação, ao invés do que acontece, por exemplo, com a tecnologia Bluetooth que requer um processo em emparelhamento.

Existes dois modos de comunicação em que um equipamento NFC pode operar:

- passivo - o equipamento, sem fonte de energia própria, é ativo quando penetra um campo eletromagnético que lhe permite comunicar. O leitor NFC é, na maioria dos casos, o dispositivo ativo, responsável por gerar o campo eletromagnético;
- ativo - o equipamento, nesta situação, possui uma fonte de energia própria que lhe permite gerar um campo eletromagnético. Este campo é capaz de ativar uma marca a funcionar em modo passivo ou então permitir a comunicação com outra marca em modo ativo.

À data de escrita desta dissertação, existem três modos especificados em que um dispositivo NFC é capaz de operar:

- ponto a ponto (do inglês *Peer-to-Peer* (P2P)) - este modo permite ao dispositivo NFC comunicar com outro dispositivo através da troca de mensagens. Os dispositivos envolvidos operam, alternadamente, entre o modo ativo e passivo ou, alternativamente, ambos em modo ativo;
- emulação de cartão - o dispositivo funciona como um *smartcard* RFID de proximidade utilizando como standard o ISO 14443 [21]. Para o leitor de cartões é transparente se está a ler um cartão físico ou a comunicar com um equipamento NFC;
- leitura e escrita - o dispositivo NFC é capaz de se comportar como um típico leitor da tecnologia RFID.

Tendo em conta o objetivo de desenhar um protocolo de autenticação que permitisse a um utilizador provar a sua identidade perante uma instituição, ficam dois modos de operação especificados, dos referenciados anteriormente, disponíveis para o efeito: ponto a ponto e emulação de cartões. Estes dois modos são explicados com mais detalhes nas subsecções seguintes.

3.1.2 Comunicação ponto a ponto

A comunicação ponto a ponto permite que dois dispositivos NFC comuniquem entre si trocando mensagens. De modo a garantir a interoperabilidade de equipamentos de diferentes fabricantes neste modo de operação, o *NFC Forum* especificou um formato de mensagens a respeitar, o *NFC Data Exchange Format* (NDEF) e um protocolo para suportar a troca delas entre dispositivos, o *Simple NDEF Exchange Protocol* (SNEP).

SNEP

O SNEP é um protocolo síncrono do tipo pedido/resposta, onde existem duas entidade ativas: um cliente e um servidor, como pode ser observado na figura 3.1.

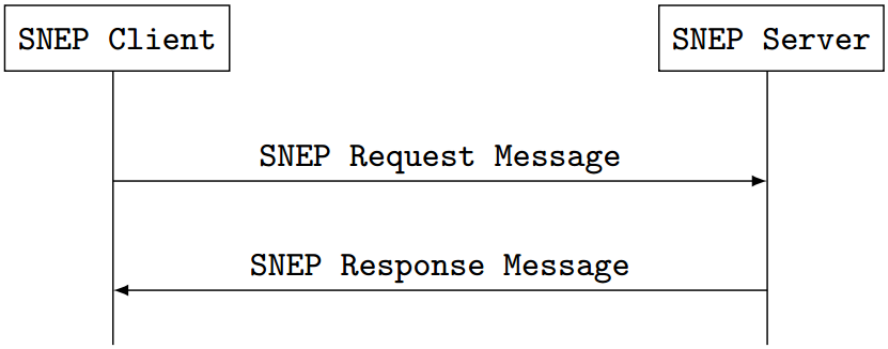

Figura 3.1: Modelo de comunicação do SNEP, retirado da especificação [3]

O cliente envia um pedido ao servidor, respeitando o protocolo, onde são especificados o método do pedido, o tamanho do campo de dados em octetos e o campo de dados. O dispositivo que intervém como servidor executa então a ação especificada no pedido usando os dados providenciados e, depois, responde com uma mensagem que contém a versão do protocolo, um código de estado que indica sucesso ou fracasso da chamada do método, o tamanho do campo de dados em octetos e o campo de dados em si. É então possível ter noção de dispositivos iniciador e alvo: o iniciador é aquele que envia a primeira mensagem.

Como referido anteriormente, o SNEP propõe-se a protocolar a troca de mensagens NDEF. Estas são transmitidas no campo de informação dos pedidos e respostas das mensagens SNEP. O tamanho máximo de uma mensagem NDEF a ser transmitida é $2^{32} - 1$ (o maior inteiro possível de codificar no campo do tamanho, presente no cabeçalho, de um pedido ou resposta de uma mensagem SNEP). A figura 3.2 ilustra um SNEP request de um PUT e a inclusão de uma mensagem NDEF no campo de informação. De notar ainda que se o tamanho de um pedido ou resposta SNEP exceder a capacidade total da unidade de dados inerente ao serviço de transporte do protocolo subjacente, a mensagem deverá ser transmitida em fragmentos.

NDEF

NDEF é o formato definido pelo NFC Forum para uma comunicação ponto a ponto entre dois dispositivos NFC. As mensagens NDEF não têm um tamanho definido consequência do seu campo de dados que é composto por um número de registos variável. Estes registos podem assumir diferentes tipos, e permitem encapsular conteúdos tais como:

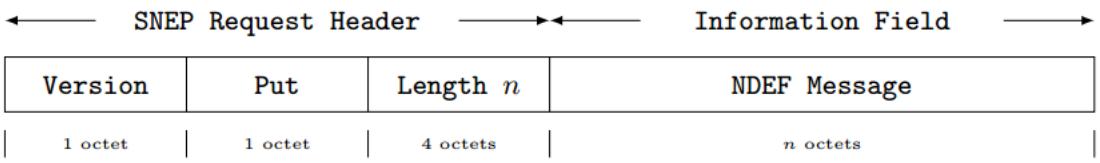

Figura 3.2: Ilustração de um pedido PUT que contém uma mensagem NDEF, retirado da especificação[3]

- *Uniform Resource Locator* (URL);
- imagens;
- texto plano;
- lógica aplicacional.

Cada registo da mensagem NDEF é composta por um cabeçalho e um campo de dados ("Carga" na figura 3.3). O cabeçalho é responsável por caracterizar o registo em questão pelos seguintes parâmetros:

- identificador - identifica o registo;
- tamanho - indica o tamanho do campo de dados ("Carga" na figura 3.3);
- tipo - indica o tipo de registo, o que permite ao recetor fazer a correta interpretação do campo de dados.

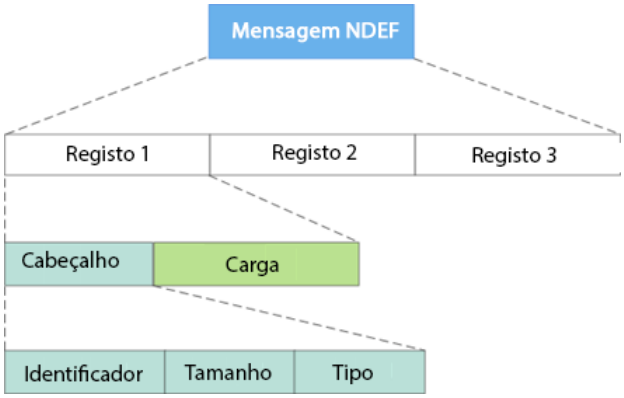

Figura 3.3: Estrutura de uma mensagem NDEF, figura retirada de [4]

3.1.3 Emulação de cartões

O modo de emulação de cartões [6] permite a um dispositivo NFC comportar-se como um cartão de proximidade, para identificação por RFID. Isto é especialmente vantajoso pois, de forma controlada, permite a uma pessoa substituir os vulgares cartões de identificação que transporta na carteira. Utilizando um protocolo standard como o ISO 14443 [21] é possível ao dispositivo NFC comunicar com um leitor de cartões da mesma forma que um cartão de proximidade. Este protocolo divide-se em quatro partes:

- ISO 14443-1 - características físicas;
- ISO 14443-2 - interface de sinal por radio frequência;
- ISO 14443-3 - inicialização e anti-colisão;
- ISO 14443-4 - protocolo de transmissão.

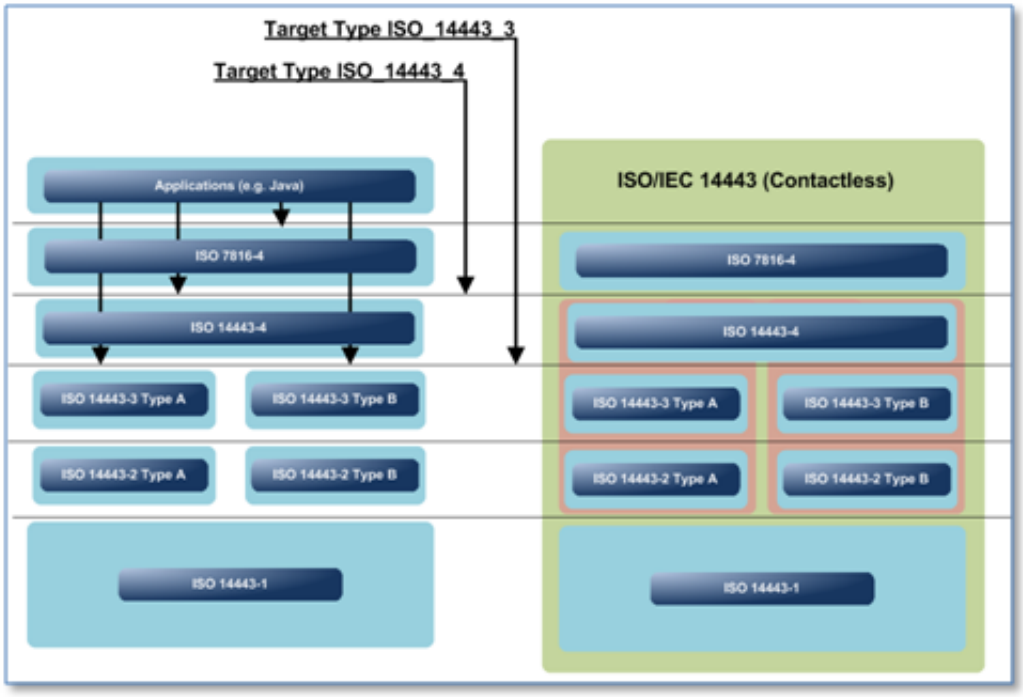

Figura 3.4: Pilha protocolar 14443, figura retirada de [5]

Existem dois tipos de cartões a operar sobre este protocolo: A e B. Estes tipos de cartões, apesar de usarem o mesmo protocolo de transmissão e a mesma frequência rádio para operação (13.56 MHz), usam métodos diferentes de modulação, codificação e procedimentos de inicialização. A diferença de modos de operação destes tipos de cartão sai fora do propósito desta dissertação, pelo que não será detalhada. Como o objetivo desta dissertação passa pelo desenho de um protocolo de autenticação, usando NFC como meio de comunicação, será feita uma análise ao protocolo de transmissão sobre o qual se trabalhou.

De forma a identificar os elementos envolvidos na comunicação NFC, o ISO 14443 usa os seguintes termos:

- *Proximity Coupling Device* (PCD) - para designar o leitor de cartões
- *Proximity Integreted Circuit Card* (PICC) - para designar o cartão de proximidade

ISO 14443-4

O ISO 14443-4 é a quarta parte da especificação do ISO 14443. Esta parte do protocolo descreve como é efetuada a comunicação entre um leitor e um cartão de proximidade. Neste

âmbito, foi especificada uma unidade de comunicação denominada de *Application Data Unit* (APDU) pelo standard ISO 7816-4 que será detalhado adiante. As comunicações efetuadas sobre este protocolo são do tipo *half duplex*, ou seja, cada um dos pontos envolvidos só pode estar a receber ou enviar dados em cada momento.

ISO 7816-4

O ISO 7816-4 define um conjunto de regras que permite a interoperabilidade e comunicação entre cartões e leitores:

- conteúdos, comandos e codificação das mensagens trocadas;
- a estrutura sobre a qual devem ser armazenados os dados;
- métodos para potenciar o acesso aos ficheiros e dados armazenados nos cartões;
- uma arquitetura de segurança para a troca de mensagens.

O formato dos APDUs está bem definido e divide-se em dois tipos:

- *Command* APDU - mensagem com origem no leitor;
- *Response* APDU - mensagem com origem no cartão, em resposta à primeira.

Segundo o standard ISO 7816-4, o *Command* APDU é sempre composto por um cabeçalho com os seguintes parâmetros e respetiva ordem:

Parâmetro	**Tamanho (bytes)**	**Descrição**
CLA	1	classe da instrução - especifica o tipo de comando
INS	1	código da instrução - especifica um comando pertencente à classe especificada
P1	1	parâmetro relativo à instrução especificada
P2	1	parâmetro relativo à instrução especificada
Lc	0, 1 ou 3	especifica a quantidade de bytes enviada campo de dados (Nc)
Dados	Nc	campo de dados
Le	0, 1, 2 ou 3	especifica a quantidade máxima de bytes do campo de dados esperada no *Response* APDU

Tabela 3.1: Parâmetros de um *Command* APDU

Da mesma forma que o anterior, o *Response* APDU também contém um conjunto de parâmetros que têm de ser respeitados para promover a interoperabilidade:

Parâmetro	**Tamanho (bytes)**	**Descrição**
Dados	Nr (pelo menos igual a Ne)	campo de dados
SW1	1	especifica o estado do processamento do *Command* APDU
SW2	1	especifica o estado do processamento do *Command* APDU

Tabela 3.2: Parâmetros de um *Response* APDU

O ISO 7816-4 comporta um vasto conjunto de classes (CLA) e códigos de instrução (INS) pelo que só serão abordados os utilizados no âmbito desta dissertação.

Um *smartcard* tem a si associado um sistema de ficheiros. Cada ficheiro é identificado por via de um nome ou número e pode ser de três tipos distintos:

- *Master File* (MF) - raíz do sistema de ficheiros cujo o identificador é 0x3F00;
- *Elementary File* (EF) - ficheiro comum. O seu tamanho é determinado aquando da sua criação;
- *Dedicated File* (DF) - semelhante a um diretório. Pode conter outros EFs e DFs.

De modo a ser possível navegar pelo sistema de ficheiros, o ISO 7816-4 disponibiliza um comando denomidado SELECT. Este comando pode selecionar quer um DF (que pode ser o MF), quer um EF. A seleção de um DF tem um EF implícito, no entando pode ser selecionado, enviando outro comando de SELECT, um outro EF ou DF após a seleção do primeiro. A seleção é efetuada através do seu identificador, do caminho desde o DF atual ou pelo nome do DF. Para tal é necessário construir o *Command* APDU da seguinte forma:

Parâmetro	Valor
CLA	0x00 (de acordo com o ISO 7616-4)
INS	0xA4 (SELECT)
P1	de acordo com o tipo de seleção
P2	de acordo com o tipo de seleção
Lc	tamanho do identificador, caminho ou nome inserir no campo de dados
Dados	identificador, caminho ou nome do ficheiro
Le	vazio ou a quantidade máxima de bytes esperados na resposta

Tabela 3.3: Construção de um *Command* APDU de SELECT

O *smartcard* deve, através de um *Response* APDU, indicar o estado da operação anterior. O APDU deve ser construído da seguinte forma:

Parâmetro	Valor
Dados	de acordo com o parâmetro P2 (pelo menos Le bytes)
SW1	estado da operação anterior (0x90 para sucesso)
SW2	estado da operação anterior (0x00 para sucesso)

Tabela 3.4: Construção de um *Response* APDU a um SELECT

O ISO 7816-4 contempla também instruções para leitura e escrita de dados. O comando *GET DATA* permite obter do *smartcard* um ou mais objetos de dentro de um contexto que pode ser um DF ou um ambiente específico da aplicação. Este comando é representado num *Command* APDU da seguinte forma:

Parâmetro	Valor
CLA	0x00 (de acordo com o ISO 7616-4)
INS	0xCA (GET DATA)
P1	de acordo com o APDU GET DATA (0x01 para codificação proprietária)
P2	de acordo com o APDU GET DATA (de 0x00 a 0xFF para codificação proprietária)
Lc	vazio, pois não existe campo de dados
Dados	vazio
Le	quantidade máxima de bytes esperados na resposta

Tabela 3.5: Construção de um *Command* APDU de GET DATA

Caso o comando anterior seja validado no *smarcard* e a sua execução seja efetuada sem erros a resposta deve conter um campo de dados seguido de dois bytes que indicam o estado da operação. Caso contrário, o campo de dados pode estar vazio e o APDU de resposta ser apenas constituído pelos bytes de estado. O comando em questão pode não ser executado com sucesso pelas seguintes razões:

- parte dos dados requisitados estão corrompidos;
- tamanho especificado em "Le" não é adequado aos dados solicitados (neste caso SW2 poderá indicar o tamanho correto);
- condições de segurança não se verificam;
- função especificada em P1 e P2 não suportada.

A resposta ao APDU com o comando GET DATA deve ser formatada da seguinte forma:

Parâmetro	Valor
Dados	vazio se ocorrer algum erro, totalidade dos dados requisitados ou parte deles caso estejam corrompidos
SW1	estado da operação anterior (0x90 para sucesso)
SW2	estado da operação anterior (0x00 para sucesso)

Tabela 3.6: Construção de um *Response* APDU a um GET DATA

Ao invés do efeito do comando referido anteriormente, podemos também armazenar um ou mais objetos dentro do contexto atual (DF selecionado ou um ambiente específico da aplicação). Para tal, é necessário usar o comando *PUT DATA* que, ao contrário do comando *GET DATA*, já não tem vazios os campos Le e Dados pois no primeiro é indicada a quantidade de informação, em bytes, presente no campo de dados e no segundo estão presentes os dados que se pretendem transferir do leitor para o *smartcard*.

Parâmetro	Valor
CLA	0x00 (de acordo com o ISO 7616-4)
INS	0xDA (PUT DATA)
P1	de acordo com o tipo de PUT DATA (0x01 para codificação proprietária)
P2	de acordo com o tipo de PUT DATA (de 0x00 a 0xFF para codificação proprietária)
Lc	tamanho, em bytes, do campo de dados
Dados	informação a transferir do cartão para o *smartcard*
Le	vazio

Tabela 3.7: Construção de um *Command* APDU de PUT DATA

Como é um comando de transferência de dados do leitor para o cartão só é esperado que a resposta deste último indique o sucesso ou o insucesso e consequente erro resultante da operação e por isso tenha um campo de dados vazio. Assim, o APDU de resposta deve obedecer à seguinte conceção:

Parâmetro	Valor
Dados	vazio
SW1	estado da operação anterior (0x90 para sucesso)
SW2	estado da operação anterior (0x00 para sucesso)

Tabela 3.8: Construção de um *Response* APDU a um PUT DATA

Com os três comandos abordados torna-se possível implementar um protocolo desafio-resposta que vise a autenticação. O leitor é capaz de selecionar um cartão através dos comandos de SELECT, enviar dados relativos ao desafio para através de comandos PUT DATA e, por fim, obter respostas ao desafio através de comandos GET DATA. Não menos importante, o leitor também se deve autenticar perante o cartão para evitar ataques, processo também passível de ser implementado com recurso aos comandos anteriormente mencionados.

3.2 Android

O Android é um sistema operativo de código aberto baseado em Linux. O alvo principal são os dispositivos móveis, no entanto, à data de escrita desta dissertação, é possível encontrá-lo instalado em outro tipo de dispositivos como computadores de bordo de automóveis e televisões.

O desenvolvimento de aplicações por terceiros é facilitado devido à disponibilização de um *Software Development Kit* (SDK) para a linguagem de programação Java. As aplicações são compiladas para Dalvik bytecode e executadas numa máquina virtual Dalvik. De notar que, apesar do SDK ser disponibilizado para a linguagem de programação *Java*, a máquina virtual *Dalvik* não é uma *Java Virtual Machine* (JVM) uma vez que o bytecode gerado para a primeira não é igual ao suportado pela segunda. Já depois do início dos trabalhos desta dissertação surgiu a versão 4.4 do Android, denominada KitKat, que trouxe consigo uma nova máquina virtual chamada ART (Android RunTime) e que se propõe a acelerar o carregamento e execução de aplicações. Os métodos disponibilizados pelo SDK em conjunto com a máquina virtual permitem uma elevada abstração ao programador do hardware do dispositivo e da

versão do sistema operativo. Algumas das vantagens do desenvolvimento para esta plataforma relevaram-se interessantes para o objetivo a que esta dissertação se propôs a atingir:

- abstração na manipulação de recursos de hardware como NFC e rede;
- armazenamento de dados, dos quais se destaca o suporte a bases de dados relacionais através do SGBD SQLite [2];
- rápido desenvolvimento de interfaces gráficas consequente do vasto leque de objetos gráficos oferecido;
- suporte a operações criptográficas pelo pacote javax.crypto [22] da linguagem de programação Java;
- suporte a operações concorrentes e e execução de serviços em segundo plano.

3.2.1 *Service*

Um serviço, do inglês *Service* [23], é um componente de uma aplicação Android cujo objetivo é a execução de operações de longa duração.

Este componente caracteriza-se por não ter nenhuma interface gráfica que permita a interação com o utilizado sendo então possível que uma aplicação tenha mais do que um serviço em execução. Este tem de ser lançado por um outro componente da aplicação e a sua principal vantagem é o facto de a sua execução não ser interrompida caso o utilizador troque de aplicação. De notar que, apesar da execução dos serviços ser efetuada em segundo plano, o processamento destes é efetuado na mesma *thread* do processo responsável pela aplicação pelo que operações computacionalmente intensivas, como a reprodução de música, ou bloqueantes, como comunicação com a rede, devem usar uma *thread* separada. Assim, a *thread* principal deve ficar dedicada às interações com o utilizador para promover a responsividade da aplicação e evitar erros como "A aplicação não está a responder" (ANR).

3.2.2 Integração com NFC

O suporte do sistema operativo à tecnologia NFC [24] surge na versão 2.3.3 (GingerBread) quando o SDK atinge a versão 9 com a adição do pacote android.nfc [25]. No entanto, nesta altura o suporte do SDK às operações com NFC era bastante limitado e o nível de abstração não era tão elevado quanto o desejado. Mais tarde, o lançamento da versão 4.0 do sistema operativo Android trouxe consigo a versão 14 do SDK que incluía melhorias notórias no que ao lidar com a tecnologia NFC diz respeito.

SNEP

Como referido anteriormente, o protocolo SNEP pretende normalizar o processo de comunicação ponto a ponto entre dois equipamentos com interface NFC. Esta comunicação é feita com recurso a mensagens NDEF, constituídas por um conjunto de registos.

O SDK do sistema operativo Android oferece suporte para este tipo de comunicação desde a API 9 [26], apesar de nas versões seguintes da API terem introduzidas um novo conjunto de funcionalidades importantes que ajudam, por exemplo, a construir e interpretar registos

[2] `http://www.sqlite.org/`

das mensagens NDEF. No entanto, apesar do amadurecimento da API, continua a não ser disponibilizado o acesso e manipulação de toda a pilha do protocolo SNEP. Mais, segundo a dissertação[19], uma das limitações da API do Android para o NFC é só "permitir o envio ou receção de uma mensagem NDEF por cada estabelecimento de um canal". O mesmo autor concluí então que para se trocarem n mensagens é necessário encostar e afastar o dispositivo n vezes. Outra limitação da plataforma neste protocolo é o fato de o utilizador ter de tocar no ecrã do equipamento para enviar uma mensagem NDEF para outro equipamento, retratado na figura 3.5.

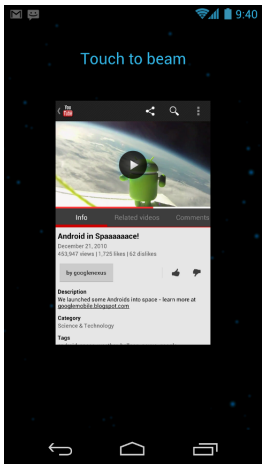

Figura 3.5: *Smartphone* Android a aguardar confirmação do utilizador para enviar uma mensagem NDEF

Host Card Emulation

O *Host Card Emulation* (HCE) [6] é uma funcionalidade introduzida na versão 4.4 do sistema operativo Android e, como nome indica, permite a emulação de cartões. Segundo a documentação do Android sobre este assunto, já existem bastantes dispositivos com NFC a suportar emulação de cartões através de um *chip* chamado elemento seguro. Este último está presente nos, por exemplo, cartões de memória *Secure Digital* (SD) ou nos cartões SIM da operadora. Neste modo, quando o dispositivo é colocado perante um leitor todos os APDUs trocados entre estes são enviados para o elemento seguro. Este processo está representado na figura 3.6.

Apesar de o elemento seguro oferecer um ambiente de execução para operações, não só nem todos os equipamentos Android possuem cartões SD por limitações de espaço, hardware ou decisão do utilizador como também existem dispositivos, como alguns *tablets*, que não possuem cartões SIM de operadora pelo que um equipamento com interface NFC pode ser privado de emular cartões. Com o propósito de resolver este problema, a versão 4.4 da plataforma, através da versão 19 da API, introduz um novo método de emulação de cartões sem recurso a elemento seguro. Ao contrário da solução que envolve o elemento seguro, nesta todos os APDUs são enviados para o processador do dispositivo, onde as aplicações estão a correr (ver figura 3.7). Assim, a aplicação fica totalmente encarregue de emular o cartão e consequentemente suportar as operações do elemento seguro, comunicando diretamente com o leitor NFC.

O HCE suporta a emulação de cartões de proximidade baseados no ISO 14443-4 e APDUs

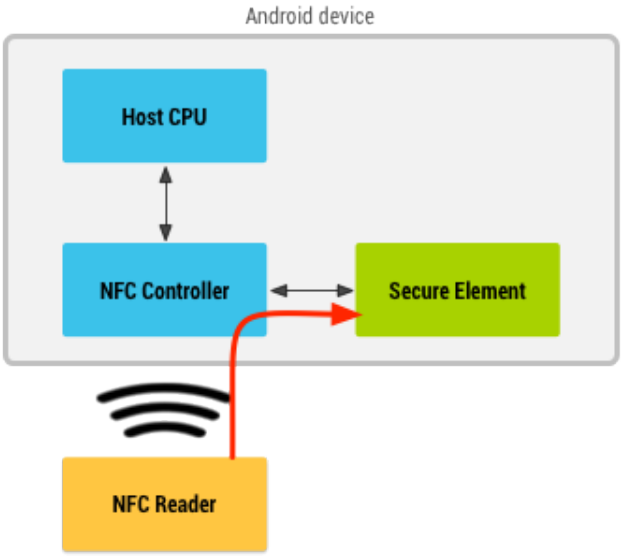

Figura 3.6: Arquitetura da emulação de cartões por NFC com recurso ao elemento seguro em Android, retirada de [6]

.

---

definidos no ISO 7816-4. A versão 19 da API contém uma classe nova que permite ao programador implementar um *Service* para efetuar HCE: HostApduService. Esta classe declara dois métodos abstratos que têm de ser implementados pelo programador. Estes métodos não são mais do que duas *callbacks* que permitem controlar as interações entre leitor e dispositivo:

- processCommandApdu - neste método é possível tratar o APDU recebido do leitor. Caso seja possível, no imeditado, enviar um APDU de resposta deve ser construído um vetor do tipo byte e usado como retorno da função. Caso contrário, o método deve retornar null.

- onDeactivated - neste método é possível saber quando a comunicação entre o leitor e a nossa aplicação foi interrompida. Existem duas causas possíveis:

  1. foi selecionada outra aplicação;
  2. o canal NFC foi perdido.

É possível que existam mais do que um serviço do tipo HostApduService no dispositivo. Deste modo, é necessário ao leitor efetuar uma seleção de qual o serviço/aplicação com o qual quer comunicar. Isto é feito com recurso à especificação de como selecionar aplicações, definida no ISO 7816-4, baseada no identificador da aplicação (AID). Um AID pode ter, no máximo, 16 bytes e deve passar por um processo de registo, definido no ISO 7816-5, de modo a evitar colisões com outras aplicações. O Android permite que a uma aplicação tenha um ou mais AIDs associados a uma aplicação. Uma desvantagem da solução disponibilizada pelo SDK é o facto dos AIDs terem de ser definidos num ficheiro XML e que, consequentemente, sempre que se queiram adicionar novos AIDs seja necessário alterar o ficheiro e compilar de novo a aplicação. Este processo poderia ser agilizado se a API proporciona-se métodos para gerir os AIDs associados à aplicação.

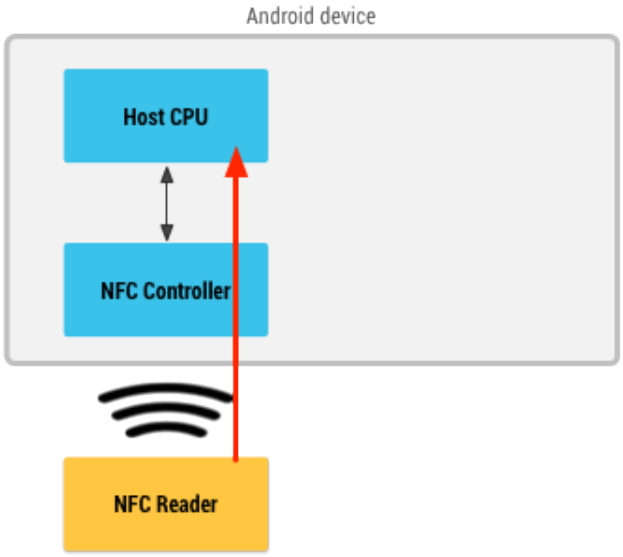

Figura 3.7: Arquitetura da emulação de cartões por NFC com HCE, retirada de [6]

.

3.3 Arduino

Arduino [3] é uma marca que representa um conjunto de microcontroladores concebidos para permitirem uma prototipagem rápida, flexível e com uma relação custo/benefício competitiva. O desenvolvimento destes equipamentos tem presente o conceito de hardware livre, do inglês *Open Source Hardware*, numa perspetiva semelhante ao software de código aberto.

Existem vários modelos de Arduino que diferem uns dos outros ao nível das características de harwdare: poder de processamento, quantidade de memória disponível, número de portos de entrada e saída (I/O), número de *timers*, etc. Um aspeto a ter em conta é a forma como os pinos de I/O são colocados: isto permite que que o equipamento seja facilmente ligado a módulos externos conhecidos como *shields*, facilmente encaixáveis sobre a forma de pilha. A comunicação com os *shields* pode ser efetuada de diversas formas recorrendo a protocolos de comunicação conhecidos, tais como I2C ou SPI. A utilização destes protocolos acarreta a vantagem de permitir a utilização de mais do que um *shield* no mesmo microcontrolador. O facto dos protocolos estarem padronizados, facilita o desenvolvimento de soluções ao programador. No entanto, o protocolo de comunicação costuma ser transparente para quem desenvolve uma vez que o fabricante do *shield* costuma disponibilizar uma biblioteca de interação para facilitar e agilizar o processo de prototipagem. Mais, o Arduino tem uma grande comunidade de utilizadores pelo que, caso o fabricante não disponibilize uma biblioteca junto com o módulo, a primeira costuma encarregar-se de o fazer. Acontece até, em alguns casos, existir mais do que uma biblioteca para interação com o mesmo periférico pelo que cabe ao programador decidir qual utilizar. O desenvolvimento pode ser efetuado num computador pessoal recorrendo a um ambiente de desenvolvimento integrado (Arduino IDE) multi plataforma. Os programas para o Arduino, independentemente do modelo, são escritos na linguagem C ou C++ e compilados através do AVR Libc, um *cross-compiler*.

[3] "http://arduino.cc/"

3.3.1 *Shield NFC*

À data do início do trabalho prático relativo a esta dissertação existiam dois *shields* NFC para Arduino que se destacavam no mercado:

- Adafruit PN532 NFC/RFID Controller Shield;
- SeedStudio NFC Shield.

Os protocolos suportados para comunicação entre o microcontrolador e o *shield* são I2C e SPI mas, em ambos os casos, o fabricante disponibiliza uma biblioteca de interação de modo a abstrair o programador. Ambas as bibliotecas oferecem um conjunto considerável de métodos que permitem tirar proveito das capacidades da tecnologia NFC. No entanto, no início do trabalho prático desta dissertação, em Outubro de 2013, a biblioteca da SeedStudio ainda se encontrava em desenvolvimento e apresentava alguns erros, ao contrário da primeira que, posto isto, se podia, à data, considerar mais madura. Mais ainda, como ambas as soluções de hardware utilizam o módulo NFC PN532, a biblioteca da SeedStudio é fortemente baseada na disponibilizada pelo fabricante concorrente.

Como referido anteriormente, o custo final da solução era um fator a ter em conta e, por isso, foi escolhido o equipamento da SeedStudio ao invés do oferecido pela Adafruit.

ISO 14443 e ISO 7816-4

De modo a atingir o objetivo desta dissertação, o Arduino em conjunto com o *shield* NFC pode funcionar como leitor de cartões de proximidade. A biblioteca disponibilizada pelo fabricante oferece três métodos para operar com marcas RFID segundo o protocolo 14443:

- inListPassiveTarget - coloca o módulo NFC a operar como leitor;
- readPassiveTargetID - lê o identificador da marca;
- inDataExchange - envia um APDU para a marca e bloqueia até que uma resposta seja recebida.

Relativamente aos APDUs especificados pelo ISO 7816-4, a biblioteca disponibilizada pela SeedStudio não oferece nenhum tipo de facilidades quer na construção, quer na análise. Assim, o programador terá de implementar toda a lógica inerente manipulação de APDUs ou utilizar uma biblioteca disponibilizada por terceiros.

3.4 Criptografia

A criptografia [27] dedica-se ao estudo de métodos através dos quais um conjunto de dados pode ser transformado, através de um processo de cifra, de forma a que o resultado se torne ininteligível a terceiros. Assim, os dados resultantes do processo de transformação podem ser trocados entre um conjunto limitado de entidades sem que isso represente um risco segurança caso sejam intercetados por terceiros, ou seja, alguém não esteja autorizado, pois dados cifrados são ininteligíveis para quem não sabe como os decifrar. Os algoritmos que suportam estas transformações, à data de escrita desta dissertação, estão bastante desenvolvidos.

Hoje em dia a segurança é fator a ter em conta no desenvolvimento de soluções de software e, por isso, o objetivo a que esta dissertação se propõe a atingir teve de ter este tópico em

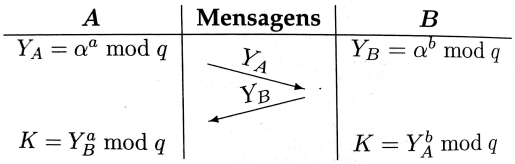

Figura 3.8: Algoritmo Diffie-Hellman, imagem retirada de [7]

conta. Num protocolo de autenticação urge a necessidade de assegurar a autenticidade quer das mensagens entre o prestador de serviços e o utilizador, quer dos dados que estas transportam. Já num sistema de informação é necessário garantir a privacidade dos dados trocados entre utilizador e servidor, evitando que, caso estes sejam intercetados por terceiros, sejam elegíveis. Em ambos os casos, a criptografia disponibiliza uma solução que permite assegurar a segurança dos processos.

3.4.1 Algoritmo Diffie-Hellman

O algoritmo Diffie-Hellman [28] é um método criptográfico para troca de chaves de sessão entre duas entidades apresentado em 1976. Este foi a primeira técnica assimétrica publicada [29] (ver 3.4.2).

Através de duas mensagens trocadas entre um par de entidades é possível que estas cheguem a um valor secreto comum, sem que este seja parte integrante de nenhuma das primeiras. Para isso, o processo recorre a conceitos da teoria dos números que por sua vez utiliza valores públicos globais e valores escolhidos por cada uma das entidades. Os primeiros são do conhecimento prévio dos interlocutores e os segundos são trocados nas mensagem previamente referenciadas. A dificuldade de cálculo de logaritmos modulares de números grandes é o suporte à segurança do algoritmo.

Assumindo que existem duas entidades A e B que pretendem chegar a um segredo comum para ser utilizado como chave de cifra e que ambas têm qu como um número primo grande e α como uma raíz primitiva mod qu, vem:

1. A e B escolhem dois valores secretos, tipicamente aleatórios

2. A calcula : $Y_A = \alpha^a mod\ q$

   B calcula : $Y_B = \alpha^b mod\ q$.

   Aos resultados destes cálculos (Y_A e Y_B) são chamados valores públicos de Diffie-Hellman.

3. A envia uma mensagem para B com o seu valor público.

   B envia uma mensagem para A com o seu valor público.

4. A calcula : $K = Y_B^a mod\ q$

   B calcula : $K = Y_A^b mod\ q$

   Os resultados destes cálculos (K) são iguais, pelo que ambas as entidades possuem um valor secreto comum passível de ser usado como chave de sessão.

3.4.2 Criptografia assimétrica

A criptografia assimétrica, também conhecida como criptografia de chave pública, vem resolver os problemas da partilha de chaves associados à criptografia assimétrica:

- distribuição de chaves - num grupo constituído por n entidades, é necessário partilhar n chaves públicas para estabelecer canais seguros entre quaisquer duas entidades;
- pré-acorde de chaves - num grupo constituído por n entidades é necessária uma distribuição de chaves sempre que é incluído um novo elemento, o que torna o processo pouco flexível.

O conceito de cifra assimétrica acenta em duas ideias:

1. par de chaves distintas, mas que se relaciona - uma pública para cifra e uma privada para o processo inverso;
2. o acesso à chave pública do par não fornece qualquer informação sobre a chave privada que o complementa.

Alguns algoritmos de cifra permitem também efetuar a cifra de modo inverso, isto é, usam a chave privada no processo de cifra e a chave pública no processo contrário. Este processo não tem o objetivo de esconder a informação envolvida na transformação, uma vez que esta é decifrável com a chave pública, mas sim de garantir a autoria.

Os pares de chaves assimétricas estão associados a uma entidade que pode ser uma pessoa, um serviço ou um servidor. Só a chave pública é que deve ser conhecida fora do domínio da entidade detentora do par, a chave privada só deve ser do conhecimento do seu possuidor e ser usada apenas por este.

Relativamente às cifras assimétricas, o tipo de cifra abordado nesta secção tem a vantagem de apenas necessitar de n chaves para criação de canais de comunicação seguros num grupo de n entidades. Por outro lado, o processo computacional relativos às transformações da informação são várias ordens de grandeza menos eficientes.

3.4.3 Assinaturas digitais

As cifras assimétricas foram um importante contributo no que toca à noção de assinatura digital de documentos. As assinaturas digitais, produzidas com base em algoritmos de criptografia assimétrica, permitem garantir três propriedades:

1. **integridade** - após a produção da assinatura, a mensagem não sofreu qualquer tipo de modificação;
2. **autenticidade** - a identidade do produtor da assinatura é passível de ser confirmada;
3. **não repúdio** - a identidade do produtor da assinatura é passível de ser comprovada.

As assinaturas devem poder ser validadas a nível global e os dados assinados digitalmente passíveis de serem associados a uma e uma só entidade. Para o efeito é utilizada criptografia assimétrica uma vez que um par de chaves está naturalmente associado a uma entidade.

A emissão de uma assinatura digital de um conjunto de dados é feita cifrando-os com a chave privada da entidade. Como este só pode ser decifrado com a chave pública da mesma entidade, fica garantido que esta é a autora caso o resultado da operação de decifra seja igual aos conjunto de dados original. Note-se que o objetivo da cifra não é tornar o conjunto de dados ininteligível, apenas garantir a origem dos mesmos.

As assinaturas digitais, para além de garantirem a origem dos dados, conseguem ainda garantir a integridade dos mesmos, ou seja, que estes se encontram da mesma forma que estavam antes de serem submetidos a um processo de assinatura. Outra vantagem inerente a este processo é o facto de dados diferentes terem assinaturas consequentemente diferentes, o que na prática significa que as assinaturas são inúteis se não foram acompanhadas pelos dados que se propõem assinar.

Caso a quantidade de dados a assinar seja elevada, o processo de assinatura é efetuado sobre o resultado de uma função de síntese aplicada sobre eles. O processo de cifra assimétrica representa um custo computacional elevado e por isso, usando funções de síntese, obtêm-se assinaturas mais pequenas, igualmente diferentes entre si e mais eficientes computacionalmente.

Segundo o autor de [7], os algoritmos de assinatura mais utilizados são o RSA e o DSA.

3.4.4 SSL

O protocolo *Secure Sockets Layer* (SSL)[30], inicialmente desenvolvido pela Netscape, foi pensado para assegurar segurança nas já implementadas comunicações HTTP. Segundo [7] a versão 3 deste protocolo é última desenvolvida.

O objetivo do SSL é assegurar a segurança na comunicação entre cliente e servidor sobre ligações de transportes, caso do protocolo TCP. Assim, o SSL decompõem-se em dois sub protocolos:

- SSL Handshake - responsável por criar e, posteriormente, manter de sessões seguras;
- SSL Transport - responsável por assegurar o transporte seguro dos dados a transmitir sobre um canal de transporte inseguro, usando, para isso, características resultantes do estabelecimento da sessão segura. Oferece ainda um mecanismo de compressão.

A negociação de sessões seguras pode ser efetuada com recurso a três formas distintas de autenticação:

- **nenhuma** - o canal seguro para comunicação entre cliente e servidor é criado com recurso a uma chave partilhada negociada pelo algoritmo de Diffie-Hellman. De notar que as mensagens que transportam os valores públicos não são autenticadas e, por isso, são passíveis de serem intercetadas;
- **autenticação do servidor** - como o nome indica, o servidor autentica-se perante o cliente. Para isso, utiliza um par de chaves assimétricas e um certificado X.509 da sua chave pública para se autenticar. A chave de sessão pode ser definida de duas formas:
  1. pelo cliente e enviada para o servidor de forma secreta, isto é, cifrada com a chave pública deste último;
  2. negociada através do algoritmo de Diffie-Hellman.

Serviço		Porto
Nome simbólico	Descrição	TCP
HTTPS	HTTP sobre SSL	443
SMTPS	SMTP sobre SSL	465
NNTPS	NNTP sobre SSL	563
LDAPS	LDAP sobre SSL	663
IMAPS	IMAP sobre SSL	993
POPS	POP sobre SSL	995

Tabela 3.9: Portos TCP atribuídos pela IANA a serviços quando são acesso lhes é efetuado via SSL, retirada de [7]

- **autenticação mútua** - ambos os interlocutores utilizam um par de chaves assimétricas e um certificado X.509 para se autenticarem. A chave de sessão é gerada sobre um canal seguro e o o seu processo de definição baseia-se numa das duas alternativas apresentadas no ponto anterior.

O SSL é maioritariamente utilizado por aplicações que implementam, de forma insegura, comunicações com protocolos aplicacionais sobre TCP. Consequência disso é que as aplicações têm de implementar a lógica inerente ao protocolo SSL, ou seja, têm de lidar com as negociações das sessões SSL e fazer uso delas para contornar os problemas de um canal de comunicação inseguro. Isto obriga a que sejam utilizados outros portos TCP para além dos que já eram utilizados nas comunicações por canais inseguros. A IANA, autoridade responsável pela atribuição de números para a internet, tem definidos alguns destes portos para implementação de comunicações seguras sobre alguns protocolos já existentes, como se pode constatar na tabela 3.9.

3.4.5 Certificação digital

O processo de emissão de certificados digitais para chaves públicas designa-se certificação digital. Entre outros elementos, todos os certificados possuem uma chave pública de uma dada entidade que se pretende certificar e uma assinatura digital gerada pela entidade que emite o certificado, tipicamente designada por Entidade Certificadora (CA, do inglês *CertificationAuthority*. Os certificados digitais têm uma validade limitada que pode ser manipulada de suas formas:

1. definindo a valididade nos parâmetros do certificados;
2. emitindo um certificado de revogação.

Um certificado de revogação tem o objetivo de garantir a invalidade de uma dada chave pública pertencente a uma entidade. Este tipo de certificados tem de ser emitido por uma CA, não necessariamente pela mesma que emitiu o certificado de chave pública.

Capítulo 4

Arquitetura

4.1 Descrição

A dissertação em causa tem o objetivo de libertar o utilizador dos cartões de proximidade físicos que o acompanham no seu quotidiano. De forma a atingir o objetivo foi desenvolvido um protocolo de autenticação para ser utilizado num contexto de cartões virtuais que envolve equipamentos Android e a tecnologia NFC.

Foi pensada uma arquitetura que suportasse os casos de uso associados aos cartões de proximidade existentes de modo a permitir autenticação de utilizadores perante entidades institucionais, tais como:

- marcação de ponto;
- bilhética;
- cupões de desconto;
- cartões de fidelização;

Neste sentido, foi necessário pensar aquilo que seria o cartão virtual na arquitetura proposta. Como referido anteriormente neste documento, começa a ser comum um utilizador ter vários dispositivos móveis Android e, por isso, foi também tida em conta a possibilidade de ter clones dos cartões. A cada dispositivo móvel está associada uma carteira virtual independente e, a cada uma destas, é possível adicionar um clone de um cartão, nunca o cartão em si. Os processos de emissão e clonagem de cartões trabalham com dados sensíveis, como veremos mais à frente, pelo requereram, inevitavelmente, um conjunto de medidas de segurança.

Outro aspeto importante no desenvolvimento desta solução foi o de disponibilizar ao utilizador os seus cartões numa só plataforma evitando a necessidade de uma aplicação proprietária por cada entidade institucional. Assim, foi integrado na solução um elemento onde os utilizadores e prestadores de serviços de autenticação efetuam o registo, designado por ponto de registo e certificação. De modo a auxiliar a comunicação entre utilizadores e entidades institucionais foi ainda introduzido um outro elemento na arquitetura, designado ponto de confiança, onde são armazenados dados relativos a cartões e processos de clonagem. Todas as instituições têm, obrigatoriamente, de se federar junto de um ponto de confiança. O facto de o ponto de confiança conter dados passíveis de serem considerados sensíveis, por serem

armazenados fora do contexto da instituição, levanta questões relativas à segurança e privacidade dos mesmos. No entanto, o serviço de ponto de confiança pode ser oferecido por qualquer entidade, pelo que se a instituição desejar pode ela própria implementar esse papel e federar-se nele, ficando assim a segurança e privacidade dos dados relativos aos seus clientes à sua responsabilidade. Para que a integração dos sistemas de informação institucionais com um ponto de confiança não seja um problema, foi especificada uma API REST que todos os pontos de confiança têm de implementar. Mais, as entidades que queiram oferecer os serviços do ponto de confiança têm de ser certificadas pelo ponto de registo e certificação, que lhes emite um certificado a ser usado na disponibilização da API REST sobre HTTPS.

Posto isto, estão identificadas quatro entidades a integrar a arquitetura:

1. ponto de registo e certificação;
2. pontos de confiança;
3. prestadores de serviços/instituições;
4. utilizadores.

A arquitetura implementada está representada na figura 4.1.

4.2 Ponto de registo e certificação

O ponto de registo e certificação é responsável por suportar o registo de utilizadores, proporcionando-lhes a possibilidade de utilizar um único conjunto de credênciais para todo o sistema de informação, independentemente do ponto de confiança onde as instituições emissoras dos seus cartões estejam federadas. Para além disto, é uma entidade certificadora que emite um certificado digital às entidades que atuam como ponto de confiança, facto que permite garantir a origem dos serviços disponibilizados e confidencialidade dos dados trocados com eles.

4.3 Ponto de confiança

O ponto de confiança é o módulo responsável por armazenar os dados relativos aos cartões dos utilizadores e aos prestadores de serviços nele federados. Através dele, pelo certificado que utiliza, os utilizadores confiam nas entidades institucionais e, consequentemente, nos cartões emitidos pelas mesmas. Disponibiliza um conjunto de serviços para as instituições através uma API REST especificada e comum a todos os pontos de confiança, cuja segurança é assegurada pelo protocolo HTTPS. Estes serviços estão também acessíveis, por parte das instituições, através de uma biblioteca Java cujo objetivo é facilitar o desenvolvimento e integração dos seus de sistemas de informação.

4.4 Instituições

As instituições são entidades interessadas em disponibilizar os seus serviços de autenticação através um pórtico capaz de interagir com equipamentos Android dotados de interface NFC, libertando o utilizador dos típicos cartões de proximidade e reduzindo custos associados à emissão e manutenção destes últimos.

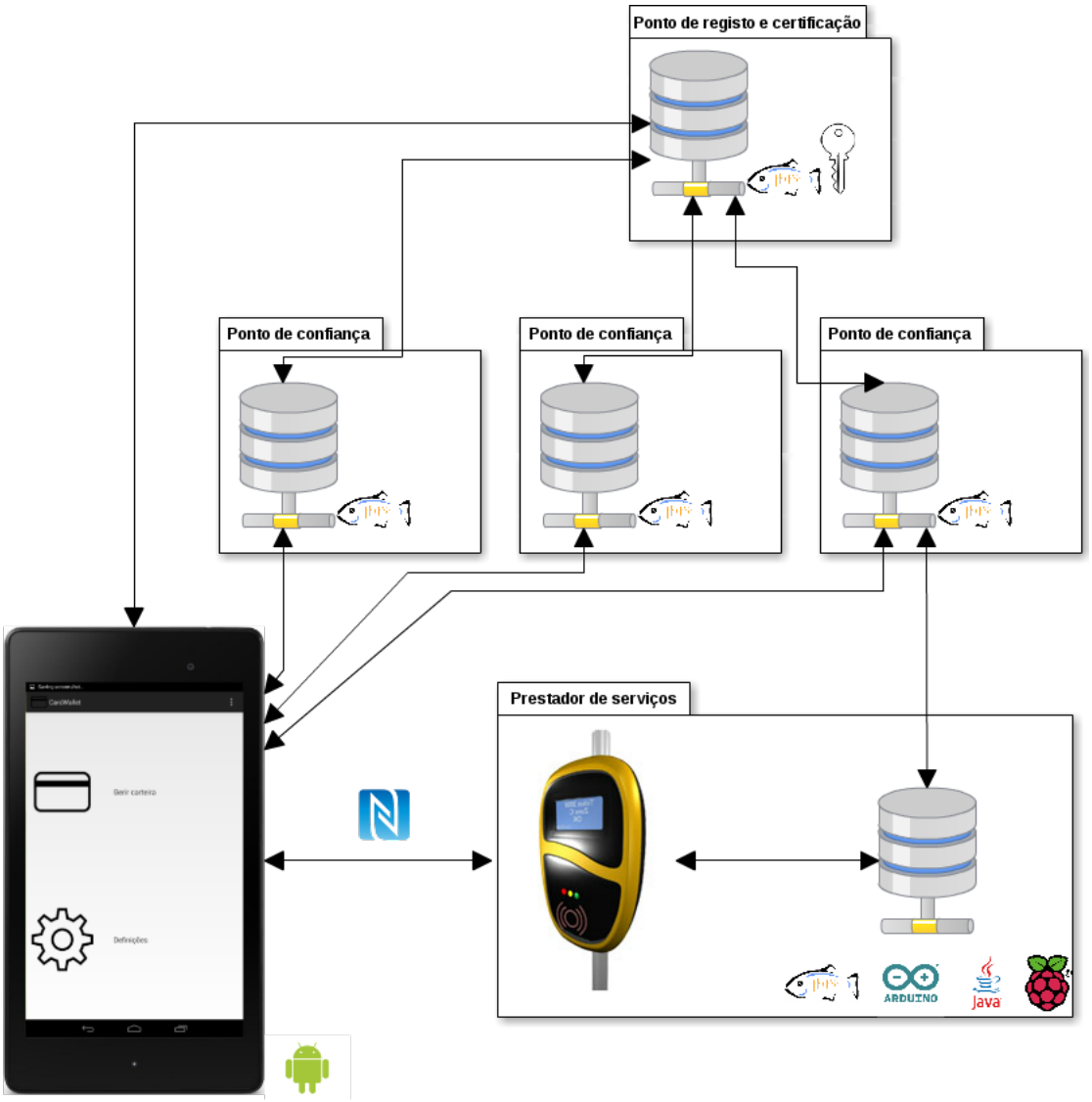

Figura 4.1: Esboço da arquitetura proposta

4.5 Utilizadores

Os utilizadores possuem uma carteira virtual por dispositivo, onde podem guardar os cartões emitidos pelas instituições para posteriormente utilizarem em processos de autenticação perante estas últimas.

4.6 Requisitos

Assim, foi levantada seguinte lista de requisitos:

- registo de pontos de confiança e emissão de certificado;
- registo de utilizadores no ponto de registo e certificação;

- registo de instituções no ponto de registo e certificação e consequente federação num ponto de confiança;
- emissão de cartões;
- clonagem de cartões;
- aprovação e assinatura digital de clones;
- geração de pares de chaves assimétricas nas instituições para assinatura de clones;
- biblioteca de programação para interação entre o sistema de informação da instituição e ponto de confiança;
- comunicação segura entre entidades;
- pórtico de autenticação;
- aplicação móvel para gestão de carteira e autenticação via NFC;
- aplicações para gestão instituições, utilizadores e cartões;
- protocolo de autenticação.

Os detalhes de implementação dos requisitos são abordados no capítulo de implementação 5.

4.7 Cartão

O cartão é o elemento que se pretende virtualizar no âmbito desta dissertação. Assim, foi necessário identificar os elementos comuns aos diferentes tipos de cartão de proximidade mais utilizados no mercado e garantir que também estariam presentem nos cartões virtuais. Daqui resultaram dois elementos relevantes ao funcionamento da lógica de negócio das instituições:

1. identificador único, ao qual chamaremos pseudónimo;
2. zona de memória.

4.7.1 Pseudónimo

Tipicamente o identificador único é uma propriedade que distingue um dado clone dos demais no contexto da instituição e tem o intuito de associar o primeiro a uma entidade como um utilizador.

A aplicação desenvolvida para a plataforma Android, quando se encontra no modo *Host Card Emulation*, gera um identificador aleatório (*NFC Identifier* [31]) para cada estabelecimento de canal de comunicação NFC. Esta característica, por um lado, revela-se uma vantagem pois impede o rastreamento de equipamentos e utilizadores, por outro, implica que a identificação fique a cargo da camada aplicacional. Assim, como na solução proposta nunca é o cartão mas sempre os seus clones que estão associados a carteiras virtuais, o pseudónimo revela-se um propriedade importante pelas razões referidas no parágrado anterior, sendo único, pelo menos, no domínio institucional.

4.7.2 Zona de memória

Segundo o artigo [32], as marcas RFID têm a capacidade de armazenar quantidades de dados na ordem das dezenas de kilobytes. A utilidade destes dados varia conforme a lógica de negócio da entidade institucional em causa, mas normalmente guardam um perfil de utilizador. De notar que algumas emissoras de cartões não fazem uso da zona de memória, limitando-se a associar o identificador a um perfil de utilizador armazenado nos seus sistemas de informação. Deste modo, garantem que a possível adulteração do perfil por parte de um atacante não coloca em causa a sua lógica de negócio.

Como os equipamentos móveis Android atuais disponibilizam quantidades memória na ordem dos Gigabytes, cada cartão poderia ter a si associada uma zona de memória muito maior que as comuns marcas RFID. No entanto, a zona de memória, para além de ter de ser replicada em cada clone do cartão, situação que requer recursos de rede, poderá também ter de ser transmitida durante o processo de autenticação via NFC. Ora, segundo a documentação do Android sobre *Host Card Emulation*, os utilizadores só devem ter a necessidade de manter o equipamento junto ao leitor NFC por um período curto de tempo, sendo que a quantidade de dados recomendada, trocada entre as partes, é 1 Kilobyte. Ainda segundo a mesma, a troca de 1 Kilobyte de dados é um processo que necessita de cerca de 300 ms para ser concluído. Posto isto e tendo em conta que o protocolo de autenticação também necessita de trocar uma quantidade considerável de dados, concluiu-se que a zona de memória do cartão virtual não deveria representar mais de um décimo de 1 Kilobyte para garantir que o processo de autenticação se concluía em tempo útil. A quantidade escolhida para armazenar o perfil do utilizador foi então 128 bytes.

4.8 Clone

Um conceito de clonagem de cartões está normalmente associado a um processo malicioso. No entanto, dependendo da entidade perante a qual o utilizador se está a autenticar, a clonagem de cartões, isto é, a presença do mesmo cartão em mais do que um dispositivo, pode não representar um problema. Assim, no processo de federação de uma instituição, é definido se os cartões emitidos ao utilizador são ou não passíveis de serem clonados. Este parâmetro pode ser alterado pela instituição a qualquer momento se esta assim o entender.

O conceito de clone, no âmbito desta dissertação, é apenas o de replicar um cartão e o objetivo da sua implementação é permitir a revogação seletiva, ao contrário do que aconteceria na inexistência do primeiro, pois isto implicaria a utilização de um só cartão em todas as carteiras virtuais do utilizador onde uma revogação significava a impossibilidade de autenticação perante a instituição emissora. Assim, o cartão, como referido anteriormente, nunca é associado a uma carteira, só os clones dele. Uma instituição que não permita a clonagem dos seus cartões terá, no máximo, um clone válido por cartão emitido e, consequentemente, um utilizador ao clonar um cartão desta instituição revoga automaticamente o clone válido anterior. Por consequência, a revogação do clone implica a impossibilidade de autenticação perante a instituição emissora.

Para além da zona de memória que armazena o perfil do utilizador, igual em todos os clones de um dado cartão, estes têm ainda os seguintes dados:

- pseudónimo;
- valor público de Diffie-Hellman;

- assinatura digital;
- validade.

4.8.1 Pseudónimo

O pseudónimo, como referido anteriormente, tem o propósito de identificar o clone no contexto institucional. Este identificador é gerado no ponto de confiança quando o utilizador desencadeia um processo de clonagem de um dado cartão.

4.8.2 Valor público de Diffie-Hellman

O valor público de Diffie-Hellman, como descrito na secção seguinte, faz parte protocolo de autenticação e tem como objetivo acrescentar segurança ao último uma vez que é parte integrante do desafio. Este valor público, em conjunto com um valor público presente num dado pórtico, permite, com recurso ao algoritmo de Diffie-Hellman, chegar a um segredo comum.

4.8.3 Assinatura digital

A assinatura digital é emitida no momento em que a entidade institucional valida o clone e tem por objetivo assegurar a integridade e autenticidade do valor público de Diffie-Hellman, perfil e pseudónimo de um clone. A obtenção da assinatura digital para estes dados contempla as seguintes etapas:

1. o perfil de utilizador é concatenado com o valor público de Diffie-Hellman do clone e com o pseudónimo;
2. o resultado da etapa 1 é submetido à função de síntese SHA-1;
3. o resultado da etapa anterior é cifrado com a chave privada da entidade institucional através do algoritmo RSA.

4.9 Protocolo de autenticação

O protocolo de autenticação desenvolvido no âmbito deste projeto é do tipo desafio-resposta e faz uso de dois elementos de forma a realizar a prova de autenticidade:

1. algo que se possui;
2. algo que se sabe.

O primeiro elemento de prova é a posse do clone do cartão emitido pela instituição, presente na carteira virtual associada à aplicação Android, cujo pseudónimo é único no contexto da instituição perante quem o utilizador se pretende autenticar. O segundo elemento de prova é a resposta a um desafio colocado pelo pórtico e que resulta de um conjunto de operações criptográficas, abordadas no capítulo seguite, e que envolvem vários dados do cartão.

O protocolo, representado da figura 4.2, assenta na troca de três mensagens para a realização do processo de autenticação:

1. desafio;
2. resposta;
3. resultado do processo de autenticação.

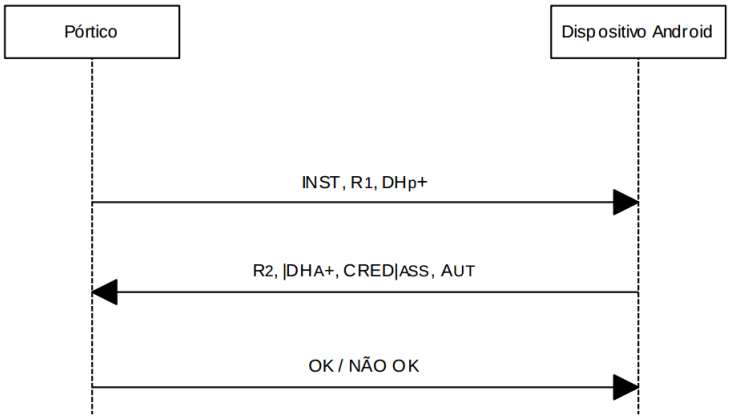

Figura 4.2: Protocolo de autenticação

No etapa 1, o pórtico é responsável por iniciar a comunicação e enviar ao equipamento Android a identificação da instituição que representa, um número aleatório (R1) e o seu valor público de Diffie-Hellman, sendo que os dois últimos fazem parte do desafio e, por isso, são utilizados na aplicação para o cálculo da resposta.

Na segunda etapa do protocolo, o equipamento Android tem selecionar o cartão associado à instituição e aplicar o algoritmo de Diffie-Hellman aos valores públicos do pórtico e do cartão. De seguida, a aplicação gera um número aleatório (R2) e aplica a função de síntese SHA-1 à concatenação do resultado do algoritmo de Diffie-Hellman com R1 e R2 cujo resultado designaremos como segredo. Por fim, envia ao pórtico os seguintes dados:

- segredo;
- R2;
- pseudónimo;
- valor público de Diffie-Hellman;
- perfil.

Os três últimos dados são relativos ao clone selecionado para autenticação perante a instituição que o pórtico representa, sendo que os últimos dois estão assinados digitalmente pela entidade emissora.

Por fim, o pórtico determina o resultado do processo de autenticação. Para isso, começa por recorrer à chave pública da instituição que valida a assinatura digital por forma a garantir a autenticidade e integridade do valor público de Diffie-Hellman, do perfil e pseudónimo do clone recebidos. Caso o pórtico determine a autenticidade e integridade dos dados, efetua o mesmo conjunto de operações da aplicação Android. Se do processamento delas resultar o

segredo recebido, o pórtico verifica junto do sistema de informação da instituição se o clone foi revogado. Caso o clone não tenha nenhuma nota de revogação a si associado então o processo de autenticação é validado e comunicado ao equipamento Android.

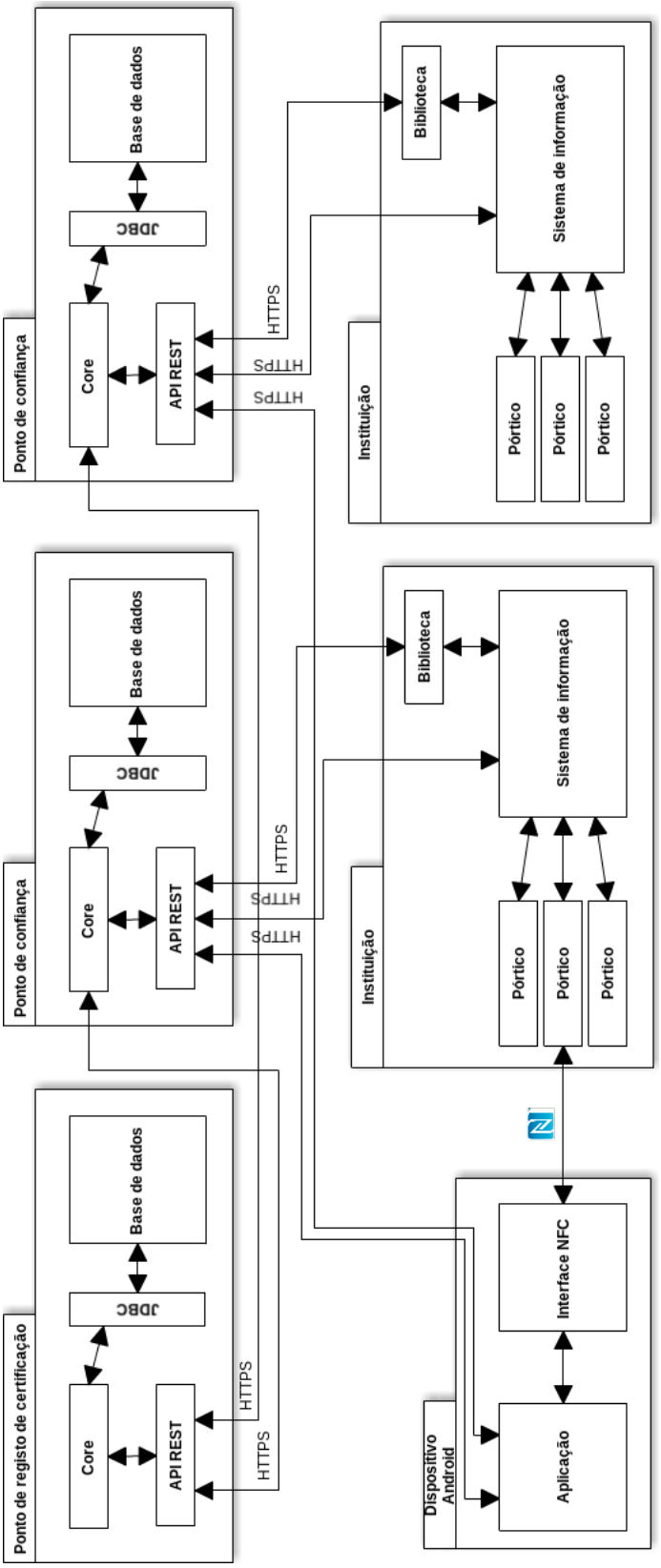

Figura 4.3: Arquitetura a implementar

Capítulo 5

Implementação

O presente capítulo pretende elucidar o leitor, com detalhe, para a solução desenvolvida no âmbito desta dissertação. Assim, serão aqui discutidos todos os elementos integrantes da arquitetura proposta (ver figura 4.3), o seu funcionamento e a sua integração.

5.1 Ponto de registo e certificação

O módulo abordado nesta secção tem o objetivo de suportar o registo de utilizadores, instituições que pretendam oferecer serviços de autenticação e entidades que pretendam oferecer serviços de ponto de confiança. Para que seja possível foi utilizado o seguinte software:

- sistema se gestão de base de dados (SGBD) - armazena informações relativas a utilizadores e prestadores de serviços;
- servidor aplicacional - suporta a lógica da aplicação e a comunicação com utilizadores e pontos de confiança;
- XCA [1] - programa para criar e gerir certificados SSL.

Para a interação com este módulo foi desenvolvida ainda uma aplicação de administração em Java Swing para facilitar o registo e instituições e pontos de confiança.

5.1.1 Certificados SSL

Os certificados SSL emitidos aos pontos de confiança integram uma cadeia de certificados cuja autoridade de certificação (do inglês *Certification Authotity (CA)*) corresponde ao ponto de registo e certificação.

Criação da CA

Com o auxílio do software XCA, começou por ser gerarada uma chave privada para o ponto de registo e certificação. Esta chave é do tipo RSA e, por se tratar de uma chave relativa a uma autoridade certificadora, foi gerada com um tamanho de 4096 bits. De seguida, foi criada a CA gerando um certificado auto-assinado que faz par com a chave privada referida

[1] `http://xca.sourceforge.net/`

anteriormente. Neste ponto, está implementada uma CA pelo que daqui em diante é possível gerar pedidos de assinatura de certificados (do inglês *Certificate Signing Requests (CSR)*) e assiná-los com a primeira.

Emissão de certificados

Sempre que surge um ponto de confiança, este necessita de um certificado assinado pela CA, a ser utilizado para providenciar SSL na sua API REST. Para obter este certificado foi gerada uma chave privada do tipo RSA e de tamanho 2048 bits e, de seguida, criado um CSR. Este CSR foi importado para o software XCA e, com o auxílio do último, assinado pela CA criada, dando assim origem a um novo certificado a ser utilizado pelo novo ponto de confiança. Por fim, foram exportados os certificados da CA e do novo ponto de confiança, prontos a serem utilizados no servidor do último.

5.1.2 Registo de utilizadores

O registo de utilizadores é um serviço oferecido pelo módulo abordado nesta secção. O registo destas entidades neste módulo permite que estas, com um único par de credenciais, acedam aos serviços de todos os pontos de confiança. Caso o registo de utilizadores fosse efetuado ao nível do ponto de confiança, seria necessário ao utilizador o registo em cada um deles, algo que não é exequível tendo em conta a possibilidade que existe de o número de pontos de confiança escalar. Além disso, o utilizador teria, na aplicação móvel, de efetuar o *login* em cada um dos pontos de confiança onde estivessem federadas instituições do seu interesse, algo que não favorece, de todo, a experiência de utilização. O processo de registo de utilizadores é feito numa aplicação Java desenvolvida para o efeito e também para suportar processos de revogação.

5.1.3 Registo de um ponto de confiança

O registo de pontos de confiança neste módulo envolve a emissão de um certificado assinado pela CA, como referido anteriormente. Este registo permite não só saber em que ponto de confiança está federada cada instituição como também alimentar, de forma dinâmica, a aplicação Android da lista de pontos de confiança com os quais tem de comunicar.

Para o processo de registo foi desenvolvida uma aplicação de administração em Java Swing que comunica com o ponto de registo e certificação através de uma API REST por ele disponibilizado. Para o processo de registo são necessários o nome do ponto de confiança o endereço na internet onde este disponibiliza a API REST especificada, como se pode observar na figura 5.1.

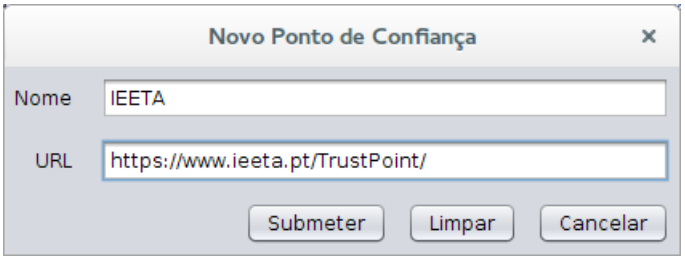

Figura 5.1: Janela de registo de um novo ponto de confiança, retirada da aplicação de administração desenvolvida

5.1.4 Registo de instituições e federação

O registo de instituições é feito no ponto de registo e certificação por forma a aproveitar o serviço de registo de utilizadores oferecido pelo último. Para além disso, a instituição fica, no momento do registo, federada num ponto de confiança pelo que, mais tarde, será possível determinar em que pontos de confiança operam as insituições. O processo de *login* de uma instituição, por exemplo, é efetuado ao nível do ponto de registo e certificação que, caso receba um conjunto de credenciais válidas, inclui na resposta um *token* de sessão e o endereço na internet onde o ponto de confiança onde está federada disponibiliza a API REST.

Este pode então federar uma instituição, introduzindo os dados relativos a esta num formulário5.2 que serão depois submetidos, através de um serviço REST disponibilizado pelo servidor aplicacional, à base de dados do ponto de confiança. Os dados necessários à federação de uma instituição são:

- nome;
- endereço de correio eletrónico;
- nome de utilizador (*username*);
- palavra passe (*password*);
- permissão de clonagem dos cartões emitidos;

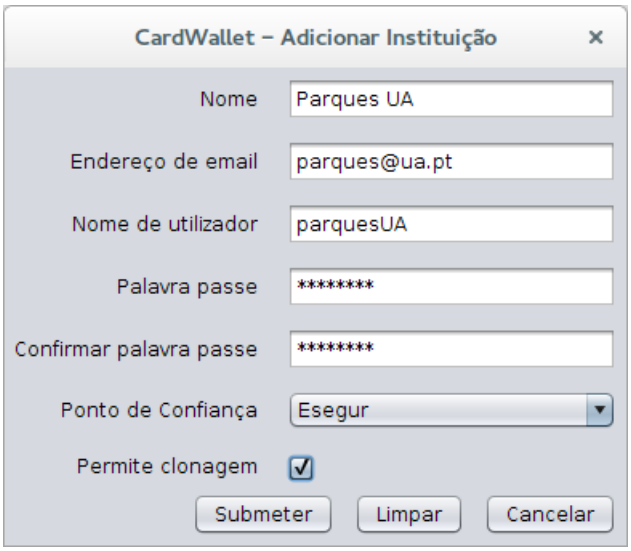

Figura 5.2: Janela de registo de uma nova instituição, retirada da aplicação de administração desenvolvida

Caso o prestador de serviços opte por não permitir a clonagem dos cartões emitidos, o utilizador só poderá ter um e um só clone válido de um dado cartão no seu conjunto de carteiras. Neste caso, se o utilizador optar por clonar um cartão e já exista um clone válido noutra carteira desse mesmo cartão, este último será automaticamente revogado.

5.2 Ponto de confiança

O serviço de ponto de confiança pode ser oferecido por qualquer entidade desde que respeite a API REST especificada. Assim, os recursos de *software* utilizados para implementar a lógica deste módulo ficam a cargo da entidade que oferece o serviço. Como para efeitos de demonstração da solução era necessário implementar um ponto de confiança, foi utilizado o seguinte *software*:

- sistema de gestão de base de dados (SGBD) - armazena informações relativas cartões emitidos e clones derivados;
- servidor aplicacional - suporta a lógica do ponto de confiança e a comunicação com utilizadores e instituições.

5.2.1 Base de dados

Como SGBD foi escolhido o sistema MySQL e utilizado um modelo de dados relacional. A opção pelo MySQL deve-se, em primeiro lugar, à utilização da linguagem SQL para interação com o SGBD e, em segundo, à forte penetração e aceitação pelo mercado o que, consequentemente, gerou uma comunidade e suporte relevantes. O modelo relacional, que tem por princípio o armazenamento dos dados sob a forma de tabelas mostrou-se adequado ao propósito.

A base de dados criada com o objetivo de armazenar os dados necessários à implementação do ponto de confiança e representada na figura 5.3 é composta pelas seguintes tabelas:

- Clone - armazena as clonagens de cartões efetuadas pelos utilizadores;
- Carteira - armazena as carteiras dos utilizadores;
- Cartao - armazena os cartões emitidos pelas instituições;
- Organizacao - armazena as instituições federadas no ponto de confiança;
- Utilizador - armazena os utilizadores com cartões naqueles pontos de confiança.

O acesso, inserção e edição dos dados presentes nas tabelas é efetuado com recurso a *stored procedures*, de forma a expor só aquilo que estritamente necessário. No auxílio às *stored procedures* estão também presentes algumas funções.

5.2.2 Servidor aplicacional

A plataforma Java EE oferece um conjunto de bibliotecas adicionais relativamente à edição SE do Java para facilitar o desenvolvimento de *software* para a web, como por exemplo os *web services* oferecidos pela API REST. O *software* desenvolvido para esta plataforma é executado num servidor aplicacional.

Pelas razões mencionadas no parágrafo anterior e pela familiariedade com a linguagem Java, foi esta a plataforma escolhida para a implementação da lógica de operação de um ponto de confiança.

Como após o processo de compilação, a plataforma exporta os serviços desenvolvidos sob a forma de um ficheiro de extensão *.war*, que terá de ser carregado para um servidor aplicacional

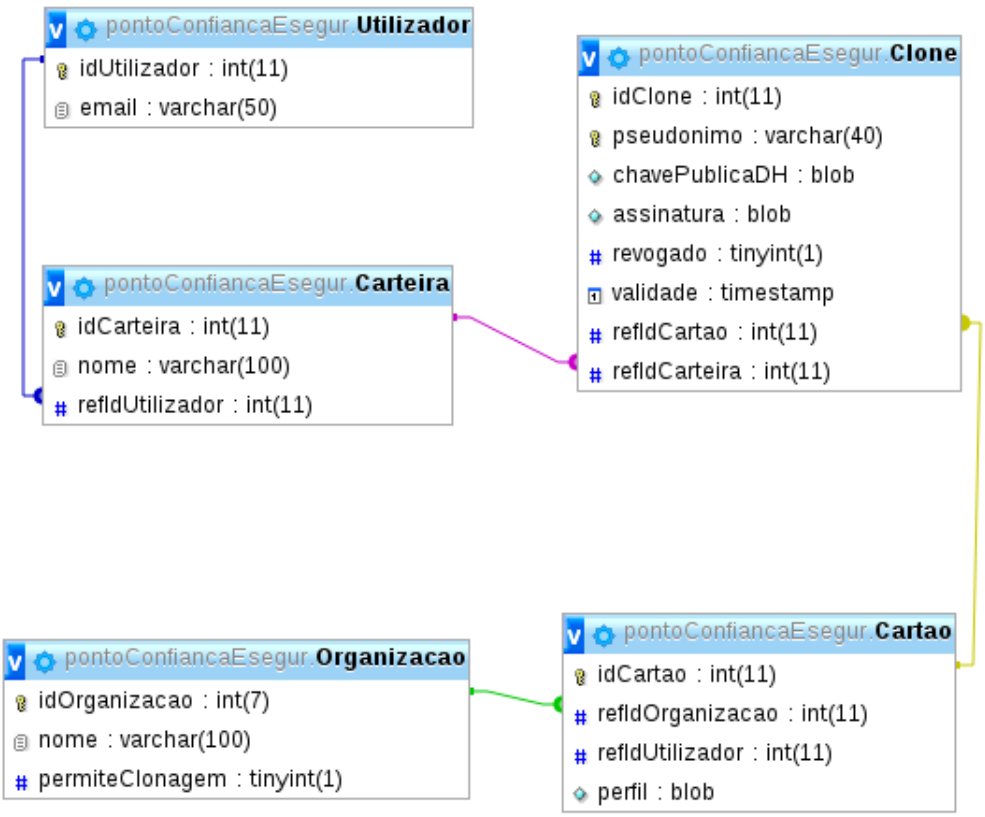

Figura 5.3: Tabelas utilizadas no base de dados do ponto de confiança

que os suporte, foi necessário escolher um. À data de escrita desta dissertação, existiam três servidores aplicacionais para *Java EE* que se destacavam:

- Tomcat;
- JBoss;
- Glassfish.

O Glassfish é uma solução interessante pois possibilita integração com o ambiente de desenvolvimento (IDE) Netbeans o que facilita o *debug* dos serviços. Para além disso, é compatível com uma vasta quantidade de recursos oferecida pela plataforma *Java EE* e tem uma edição de comunidade gratuita e de código aberto. Pelas razões mencionadas, foi a alternativa escolhida.

5.2.3 Serviços REST

Os serviços para interação com o ponto de confiança foram desenvolvidos na plataforma *Java Entreprise Edition (Java EE)* e disponibilizados sob a forma de uma API REST. A escolha por REST deve-se ao facto do seu funcionamento acentar sobre o protocolo HTTP o que facilita a integração com outros sistemas de informação pois este protocolo não bloqueado

por *firewalls* de instituições e a sua utilização é independente da linguagem programação escolhida. Os dados trocados entre as entidades e o servidor aplicacional estão no formato JSON o que, mais uma vez, favorece a interoperabilidade de sistemas de informação pois a sua utilização é independente da linguagem de programação.

Para o desenvolvimento dos serviços REST foi utilizada a API *Java Servlet* pois permite facilmente explorar os métodos do protocolo HTTP através da classe HttpServlet [33]. A classe a implementar, responsável por gerir os pedidos HTTP, tem de ser extendida à classe HttpServlet de modo a implementar os métodos por esta última declarados:

- doGet - método a implementar se a classe suportar pedidos HTTP GET;
- doPost - método a implementar se a classe suportar pedidos HTTP POST;
- doPut - método a implementar se a classe suportar pedidos HTTP PUT;
- doDelete - método a implementar se a classe suportar pedidos HTTP DELETE;

No desenvolvimento da solução proposta foram utilizados os métodos HTTP GET e HTTP POST, encapsulados pelos métodos doGet e doPost respetivamente. O método HTTP GET foi utilizado para disponibilizar sobretudo serviços de consulta como por exemplo lista de utilizadores, lista de cartões, entre outros. Por outro lado, o método HTTP POST foi utilizado para disponibilizar serviços que submetem dados ao servidor, tais como *login*, registo de entidades, emissão de cartões, entre outros. O protocolo HTTP[34] especifica que o método POST tem um campo *body* onde se podem colocar dados. Este campo foi utilizado para transportar os dados, em JSON, das aplicações cliente para o servidor aplicacional.

O recurso à linguagem de programação Java permite facilmente obter parâmetros enviados aquando da chamada do serviço, realizar operações sobre a base de dados MySQL e com base nos resultados destas gerar conteúdo dinâmico, em JSON, para a resposta.

A integração da plataforma Java EE com o MySQL é bastante simples, requerendo apenas a utilização do driver MySQL Connector/J que disponibiliza uma API para manipulação da base de dados. Para a manipulação do conteúdo JSON foi utilizada a biblioteca GSON [35] pois para além de permitir a manipulação dos objetos JSON de forma procedimental, permite também a conversão direta de objetos Java em objetos JSON e vice-versa.

À excepção do serviço de registo de novos utilizadores e do próprio *login*, todos os outros requerem autenticação. Para isso, a chamada de um destes últimos serviços tem sempre se incluir um *token* de sessão, nos parâmetros do URL associado ao serviço, gerado pelo serviço de *login* caso as credênciais sejam válidas. Quando o servidor aplicacional recebe um pedido HTTP verifica sempre, antes de qualquer outra operação, a existência e consequente validade do *token* de sessão. Para isso, como a responsabilidade de emissão de *tokens* está a cargo do ponto de registo e certificação, o ponto de confiança usa um *web service REST* disponibilizado no anterior para valiar o *token* e assim determinar se o chamador do serviço tem permissões para efetuar a operação. Estes *tokens* de sessão são emitidos pelo serviço de *login*, disponibilizado no ponto de registo e certificação, com o auxílio da classe UIID[36] da linguagem Java, que representa um identificador único universal imutável constituído por 128 bits, e armazenados na base de dados com uma validade de sete dias desde o momento da chamada do serviço. Por razões de segurança os *tokens* de sessão podem ser revogados.

5.2.4 Segurança das comunicações

Para garantir a segurança das comunicações, todos os serviços são disponibilizados sobre o protocolo HTTPS que, ao protocolo HTTP, acrescenta uma camada se segurança através de SSL. O servidor aplicacional Glassfish já vem configurado com dois certificados, na sua *keystore*, que são usados pelo protocolo SSL o que elimina a necessidade, ao administrador, de configuração do servidor aplicacional para o uso de SSL. Por outro lado, isto significa também que qualquer pessoa tem acesso a eles e aos pares de chaves pública e privada a eles associados. Para contornar este problema, foram eliminados os certificados existentes, por omissão, no Glassfish e carregados novos. Os certificados foram gerados e carregados para a *keystore* através da ferramenta de gestão de chaves e certificados keytool e utilizam o algoritmo RSA e chaves de 2048 bits. Para além disso foi carregado o certificado emitido pelo ponto de confiança e certificação, o qual foi utilizado para implementar SSL nos serviços web. Por último, para os serviços serem apenas disponibilizados sobre HTTPS é necessário adicionar uma *securityconstraint* no ficheiro web.xml relativo ao projeto onde os serviços foram desenvolvidos:

```
<security-constraint>
    <display-name>ConstraintHTTPS</display-name>
    <web-resource-collection>
        <web-resource-name>CentralServices</web-resource-name>
        <description/>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <description/>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Extrato 5.1: Configuração dos serviços REST para serem disponibilizados apenas sobre HTTPS

5.2.5 Biblioteca de interação

Como referido anteriormente, a API REST e o formato de dados JSON têm o propósito da compatibilidade entre diferentes tecnologias e plataformas. No entanto, para acelerar o processo de integração, foi desenvolvida uma biblioteca *Java* que abstrai o programador da camada de rede e do formato dos dados.

Esta biblioteca pretende acelerar a integração dos sistemas de informação de uma instituição com a solução proposta. Os objetos e métodos presentes não incorporam a totalidade dos serviços REST disponibilizados pelo ponto de confiança, apenas aqueles que se revelam necessários aos prestadores de serviços de autenticação. Esta biblioteca foi utilizada no desenvolvimento de um protótipo de sistema de informação e pórtico de uma entidade institucional.

5.3 Aplicação Android

A arquitetura da aplicação desenvolvida para a plataforma *Android* está representada na figura 5.4 e é composta pelo seguinte conjunto de módulos:

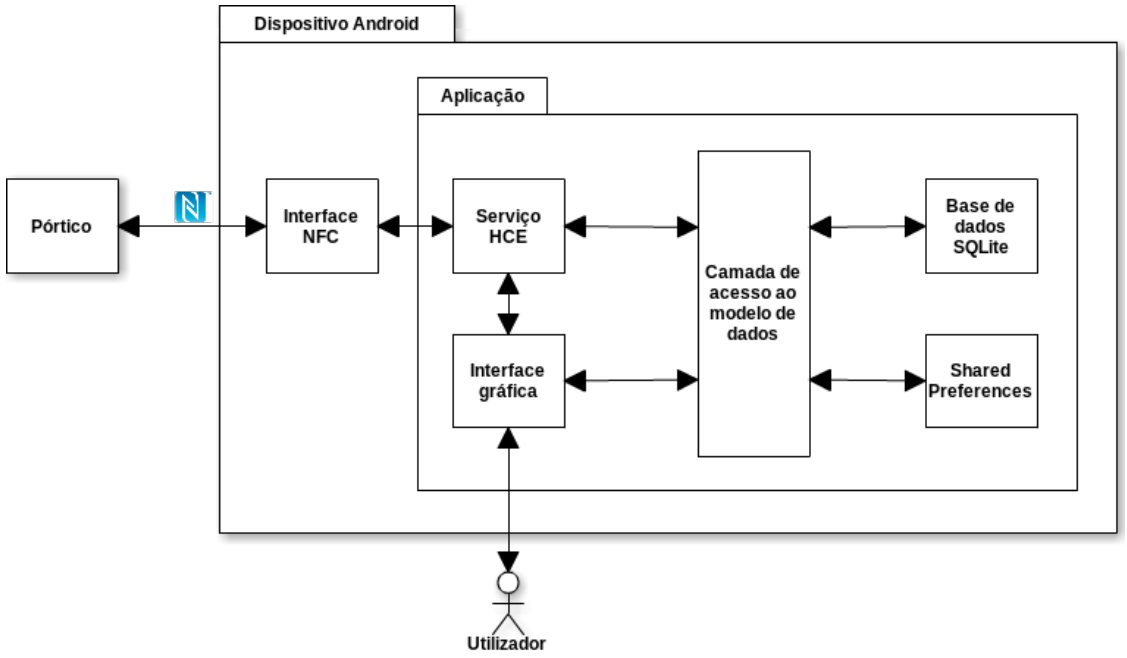

Figura 5.4: Arquitetura da aplicação Android desenvolvida

- *Service* para *Host Card Emulation*;
- aplicação gráfica;
- camada de acesso e manipulação aos dados;
- base de dados SQLite;
- *SharedPreferences*.

5.3.1 *Host Card Emulation Service*

O *Service* é responsável por lidar com as ações relativas ao NFC e o utilizador não tem qualquer tipo de interação com ele. A classe que implementa este *service* tem de ser extendida à classe HostApduService que disponibiliza duas *callbacks* que têm de ser implementados pela primeira: processCommandApdu e onDeactivated. O dispositivo funciona como se de um cartão de proximidade se tratasse e este serviço é selecionado, através de um SELECT APDU, como se de uma aplicação se tratasse. A aplicação tem a si associada um AID que tem de ser utilizado para a referenciar no SELECT APDU. Quando o sistema operativo recebe um APDU de seleção com o AID associado à aplicação desenvolvida encaminha-o para o *service* implementado, despoletando a *callback* processCommandApdu que o processa. Todos os APDUs seguintes, à excepção de um APDU de seleção com AID de uma outra aplicação, são processados na mesma *callback*. Assim, surgiu a necessidade de os diferenciar tendo para isso sido desenvolvido um conjunto de objetos que representam os tipos de APDUs utilizados, caraterizados na tabela 5.1.

Objeto	Descrição
APDUClass	enumaredo que identifica o tipo de APDU
APDU	classe que representa um APDU genérico
CommandAPDU	classe que representa um *Command APDU*
ResponseAPDU	classe que representa um *Response APDU*

Tabela 5.1: Objetos desenvolvidos para manipular APDUs 7816-4

A implementação destas classe tem presente um mecanismo de herança: as classes CommandAPDU e ResponseAPDU extendem à classe APDU. Os APDUs são recebidos no dispositivo são sempre do tipo *Command APDU* e chegam à *callback* na forma de vetor de bytes. Por isto, o construtor da classe aceita um vetor de bytes. Para a identificação da instrução do APDU foi desenvolvido o método getApduClass que devolve um enumerado APDUClass. Os APDUs enviados como resposta são sempre do tipo *Response APDU* e, por isto, são construidos com o suporte da classe ResponseAPDU. Os APDUs de resposta têm sempre dois bytes de estado e podem ter, ou não, ou um campo de dados, o que implicou que a classe ResponseAPDU disponibilizasse dois construtores com os seguintes parâmetros:

1. vetor de bytes do campo de dados e bytes de estado SW1 e SW2;
2. bytes de estado SW1 e SW2.

O processamento do APDU é caraterizado na figura 5.5.

Depois de identificada a instrução do APDU é verificado se esta é do tipo SELECT. Em caso afirmativo é construído um APDU de resposta através do segundo construtor da classe ResponseAPDU, representado na tabela seguinte:

APDU		
Campo de dados	**SW1**	**SW2**
(vazio)	0x90	0x00

Tabela 5.2: Response APDU ao APDU de SELECT

Caso a instrução do APDU não seja de seleção é então verificado se se trata de uma instrução do tipo PUT DATA. Em caso afirmativo, são verificados os parâmetros do cabeçalho do APDU: P1 e P2. Segundo o ISO 7816-4, P1 e P2 podem assumir dados aplicacionais, encapsulando operações proprietárias, desde que utilizados na gama de valores entre 0x0100 e 0x01FF e, por isso, foram naturalmente escolhidos para indicar quais os dados a armazenar no dispositivo. A tabela seguinte demonstra o significado atribuído aos parâmetros P1 e P2 nos APDUs PUT DATA:

P1	P2	Dados a armazenar no dispositivo
0x01	0x00	valor público de Diffie-Hellman e número aleatório do pórtico e *hash* identificadora da instituição
0x01	0x01	resultado do processo de autenticação

Tabela 5.3: Valores de P1 e P2 proprietários utilizados nos APDUs GET DATA

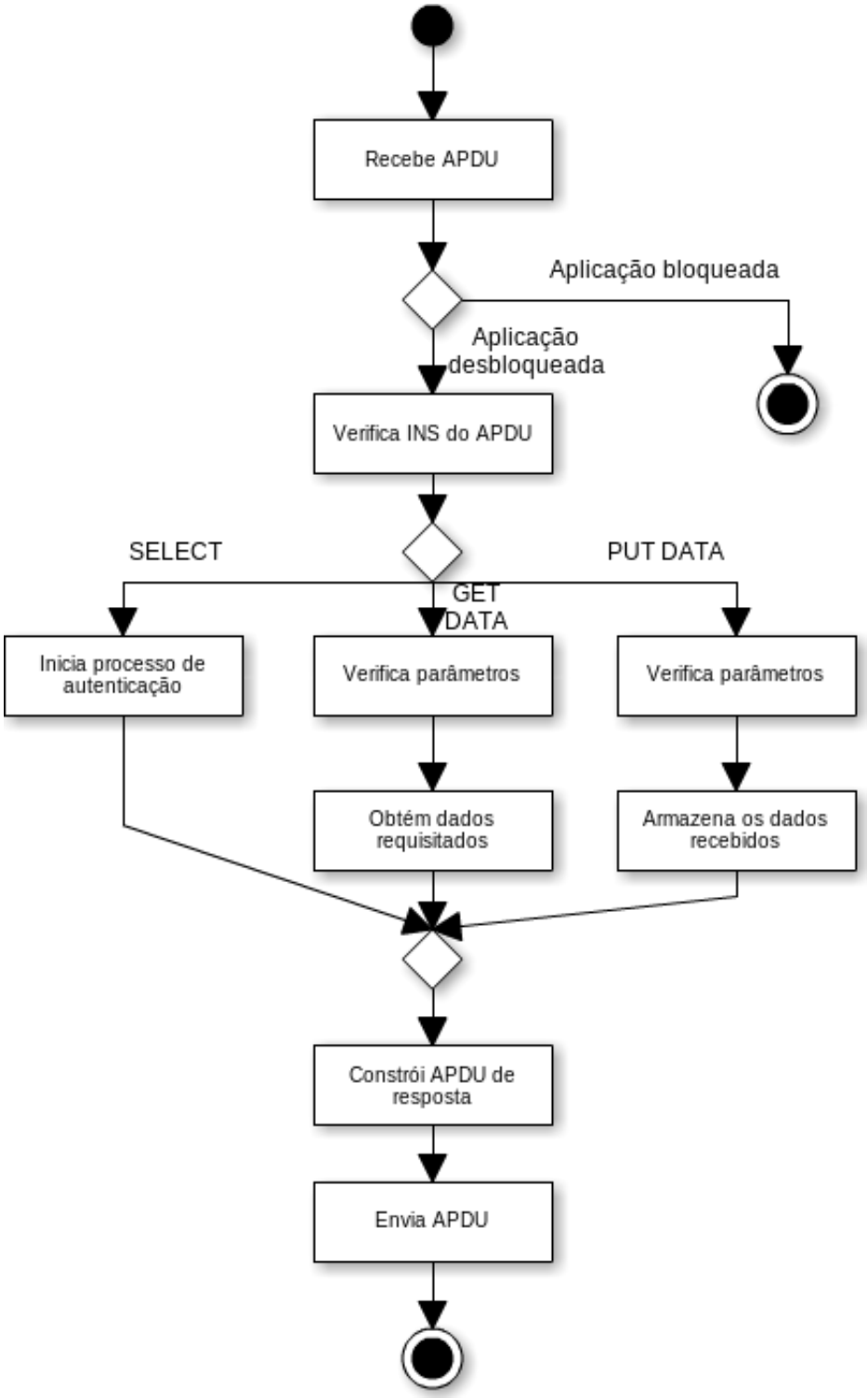

Figura 5.5: Etapas do processamento de um APDU pelo HCE *Service*

O APDU de resposta é construido através do segundo construtor da classe ResponseAPDU, uma vez que não tem campo de dados, e está APDU representado na tabela 5.4.

APDU		
Campo de dados	**SW1**	**SW2**
(vazio)	0x90	0x00

Tabela 5.4: APDU de resposta a um APDU PUT DATA

Por fim, caso a instrução do *Command APDU* recebido não seja nem de SELECT nem de PUT DATA, é verificado se a esta é do tipo GET DATA. Caso seja, à semelhança do processo efetuado no caso de APDU PUT DATA, também aqui são verificados os parâmetros P1 e P2 do cabeçalho para identificar que dados a armazenar no dispositivo. A tabela 5.5 representa o que significam os valores atribuídos a P1 e P2 num GET DATA APDU.

P1	**P2**	**Dados a recolher do dispositivo**
0x01	0x00	perfil do clone
0x01	0x01	valor público de Diffie-Hellman do clone
0x01	0x02	assinatura do clone
0x01	0x03	cálculo de um segredo com base no algoritmo de Diffie-Hellman
0x01	0x04	pseudónimo do clone

Tabela 5.5: Valores de P1 e P2 proprietários utilizados nos APDUs GET DATA

O APDU de resposta é construido através do primeiro construtor da classe ResponseAPDU e APDU está representado na tabela 5.6.

APDU		
Campo de dados	**SW1**	**SW2**
dados requisitados, de acordo com a tabela 5.5	0x90	0x00

Tabela 5.6: APDU de resposta a um APDU GET DATA

O ISO 7816-4 define valores para SW1 e SW2 cado a instrução presente no *Command* APDU recebido não seja suportada ou seja inválida. Assim, caso o APDU recebido não contenha nenhuma das instruções referidas anteriormente, é enviado o *Response APDU* representado na tabela 5.7.

Caso o *Command* APDU contenha uma instrução válida, das referidas anteriormente, mas os parâmetros não estajam de acordo com o esperado são coficados SW1 e SW2 de acordo com o valor definido no ISO 7816-4 para o efeito. O *Response APDU*, para este caso, está representado na tabela 5.8.

Por fim, quando é recebido o APDU PUT DATA que indica o sucesso do processo de autenticação é lançada um notificação no sistema operativo que dá *feedback* ao utilizador do sucesso ou insucesso da operação. Para além da notificação visual, é também reproduzido um som com o objetivo de fornecer *feedback* auditivo. Caso o utilizador clique na notificação visual a aplicação é aberta.

APDU		
Campo de dados	**SW1**	**SW2**
(vazio)	0x6D	0x00

Tabela 5.7: APDU de resposta a um APDU GET DATA

APDU		
Campo de dados	**SW1**	**SW2**
(vazio)	0x6B	0x00

Tabela 5.8: APDU de resposta a um APDU GET DATA

5.3.2 Padrão de segurança

Foi feito uso de um padrão de segurança para proteção da carteira virtual contra usos indevidos. Este padrão é definido sobre uma matriz de pontos de dimensão nove por nove onde o utilizador tem de ligar quatro ou mais pontos adjacentes.

A aplicação desenvolvida é caraterizada por implementar uma máquina com dois estados: bloqueada e desbloqueada. Por omissão, bloqueia ao fim de dois minutos sem utilização. Por utilização entenda-se interação com os elementos gráficos por parte do utilizador ou aproximação do dispositivo a um leitor de cartões de proximidade que, através de um SELECT APDU, selecione a aplicação desenvolvida. O tempo de bloqueio pode ser redefinido pelo utilizador se assim o desejar.

5.3.3 Interface gráfica

A interface gráfica pretende ser a ponte que faz a ligação entre o utilizador e a camada de dados. No desenvolvimento para a plataforma *Android*, a classe Activity é a responsável por criar e lidar com uma janela constituída por elementos gráficos com os quais o utilizador interage. Assim, as classes lidam com janelas de ambientes gráficos têm de ser estendidas à classe Activity através do processo de herança. A interface gráfica é desenvolvida num ficheiro XML onde os elementos gráficos são declarados. A classe que lida com uma janela de interação com o utilizador, tem de definir qual a interface a apresentar através do método setContentView. Os eventos associados às ações do utilizador nos elementos gráficos são tratados em *callbacks* específicas para o efeito, declaradas na Activity que lida com a janela gráfica em execução.

A aplicação desenvolvida no âmbito desta dissertação tem o seguinte conjunto de janelas:

1. *login*;
2. definição de padrão de segurança;
3. desbloqueio da aplicação através do padrão de segurança;
4. menu da aplicação;
5. gestão de carteira;
6. opções da aplicação.

Login

A janela de *login* permite ao utilizador introduzir as suas credenciais e autenticar-se perante o ponto de registo e certificação utilizando um serviço de disponibilizado pela API REST. Caso estas estejam corretas o ponto de registo e certiicação devolve um *token* de sessão que será utilizado nas chamadas subsequentes de serviços dos pontos de confiança.

Definição do padrão de segurança

Caso o utilizador esteja a fazer o *login* pela primeira vez, a aplicação vai mostrar uma janela para a definição de um padrão de segurança. Depois de introduzido, o utilizador terá de o confirmar introduzindo-o de novo.

Esta funcionalidade foi implementada com recurso à biblioteca android-lockpattern [37] disponiblizada, à data de escrita desta dissertação, na plataforma *Google Code*. A biblioteca em questão utiliza o código fonte do padrão de bloqueio do sistema operativo Android. Este código foi ligeiramente modificado, pelos contribuidores da biblioteca, para facilitar a sua utilização em aplicações desenvolvidas por terceiros, sendo assim a biblioteca disponibilizada sobre a forma de um *library project*.

Para criar um novo padrão a aplicação desenvolvida tem de criar uma Intent que requisita uma Activity, externa à aplicação, onde o utilizador poderá introduzir o padrão de segurança. Este pedido tem a si associado um identificador que serve, mais tarde, para identificar o resultado.

```
protected static final int REQUEST_PATTERN = 0;

...

Intent reqPinPattern = new Intent(LockPatternActivity.ACTION_CREATE_PATTERN,
        null,
        this,
        LockPatternActivity.class);

startActivityForResult(reqPinPattern, REQUEST_PATTERN);
```

Extrato 5.2: Instanciação da Intent que irá lançar uma Activity para introdução de padrão

Depois da etapa anterior, o resultado do processo de definição do padrão é obtido na forma de um vetor do tipo char. Para isso, a Activity que lança a Intent tem de implementar uma *callback*, onActivityResult, onde o resultado será obtido.

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
       Intent data) {

    switch (requestCode) {

      case REQUEST_PATTERN:

        if (resultCode == RESULT_OK) {

          char[] pattern = data.getCharArrayExtra(
                          LockPatternActivity.EXTRA_PATTERN);

          appToken.setPinPattern(new String(pattern));
        }
        break;
      ...
    }
}
```

Extrato 5.3: Callback onde é tratado o padrão introduzido pelo utilizador

Desbloqueio da aplicação

Ao fim de dois minutos sem utilização a aplicação bloqueia e é necessário introduzir o padrão de segurança para a desbloquear. O bloqueio é válido quer para interações com os elementos gráficos, quer para comunicações pela interface NFC. Caso o utilizador aproxime o dispositivo de um leitor de cartões de proximidade, este selecione a aplicação desenvolvida e esta última esteja bloqueada, o segundo responde com APDUs vazios e lança uma notificação visual e sonora para dar conhecimento ao utilizador de que é necessário proceder ao desbloqueio. Se o utilizador clicar na notificação lançada é aberta a aplicação desenvolvida.

Quando a aplicação é aberta e se encontra no estado bloqueado, à semelhança da situação de definição do padrão, é lançada uma Intent que requisita uma Activity externa onde o utilizador pode desbloquear a primeira. Também aqui, o padrão introduzido pelo utilizador é recebido na Activity que lança a Intent na *callback* onActivityResult.

Menu da aplicação

Esta é a janela raiz da aplicação pois é aquela mostrada ao utilizador quando ele a abre a segunda. Quando a aplicação é aberta, se não existir ou for inválido o *token de sessão* esta Activity dá lugar à de *login*. Caso contrário, esta Activity verifica se a aplicação está desbloqueada e, se não estiver, é requisitado ao utilizador que introduza o padrão de segurança.

Quanto à interface desenvolvida, optou-se por algo minimalista: um botão que dá acesso à gestão da carteira virtual e outro que permite editar as definições da aplicação.

Gestão de carteira

Quando o utilizador usa aplicação pela primeira vez, após um processo de *login* efetuado com sucesso, esta apresenta-lhe uma janela em forma de *popup* com uma sugestão para o nome da carteira virtual. O nome sugerido é gerado a partir da concatenação de um nome aleatório

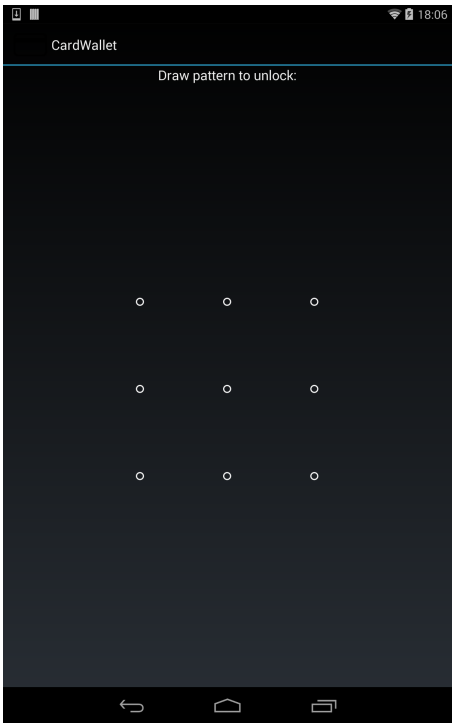

Figura 5.6: *Activity* onde o utilizador tem de introduzir o padrão de segurança, retirada da aplicação Android desenvolvida

---

de cinco caracteres, do caráter "@" e do modelo do equipamento Android. Os primeiro cinco carateres são gerados com recurso à classe UUID do Java e o modelo do dispositivo é obtido da constante MODEL da classe Build. Se o utilizador preferir pode alterar o nome sugerido e defini-lo como entender.

```
public Wallet() {

  this.name = new String(UUID.randomUUID().toString()).substring(0, 5) + "@" + Build.MODEL;
}

public Wallet(String name) {

  this.name = name;
}
```

Extrato 5.4: Construtores da classe Wallet

O nome escolhido para a carteira virtual é submetido ao ponto de registo e certificação, através de um método disponibilizado pela sua API REST, para a criação da carteira. Caso o utilizador já possua uma carteira com o nome submetido ao serviço será informado de tal e terá de escolher um outro nome.

No âmbito da aplicação foi ainda desenvolvida uma Activity que permite ao utilizador gerir a sua carteira. A janela contém duas listas dispostas verticalmente:

1. cartões emitidos ao utilizador;

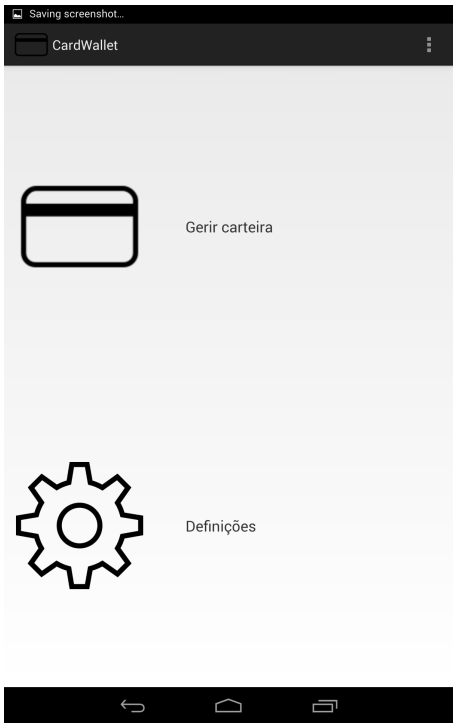

Figura 5.7: *Activity* principal, retirada da aplicação Android desenvolvida

2. cartões clonados para a carteira.

A primeira lista contém, em cada elemento, o nome da instituição emissora do cartão, a informação se o cartão é passível de ser clonado e, caso não seja, se já está associado a uma outra carteira e um botão que permite clonar o cartão. Para o seu preenchimento, a aplicação começa por obter uma lista de todos os pontos de confiança, com os respetivos endereços *web* da API REST, registados junto do ponto de registo e certificação. De seguida, usa um serviço de cada um dos pontos de confiança para os interrogar se têm armazenado algum cartão emitido aquele utilizador.

A segunda lista contém os clones presentes na carteira virtual do dispositivo. Cada clone é representado pelo nome da instituição emissora e estado do processo de clonagem.

O processo de clonagem começa por pedir a palavra-passe ao utilizador, por razões de segurança, pois um atacante, com acesso ao dispositivo, poderia encontrar a aplicação desbloqueada e assim clonar, para aquela carteira virtual, todos os cartões emitidos ao utilizador. Assim, a palavra-passe é requisitada ao utilizador em cada processo de clonagem de um cartão e enviada, juntamente com o nome de utilizador, ao ponto de confiança para validação que por sua vez terá de confirmar as credenciais junto do ponto de registo e certificação. Caso a palavra-passe esteja correta, o dispositivo gera um par de chaves de Diffie-Hellman para o clone e submete, de seguida, o pedido de aprovação da clonagem ao ponto de confiança. Este, ao receber o pedido, gera um pseudónimo único capaz de identificar o clone no contexto da instituição.

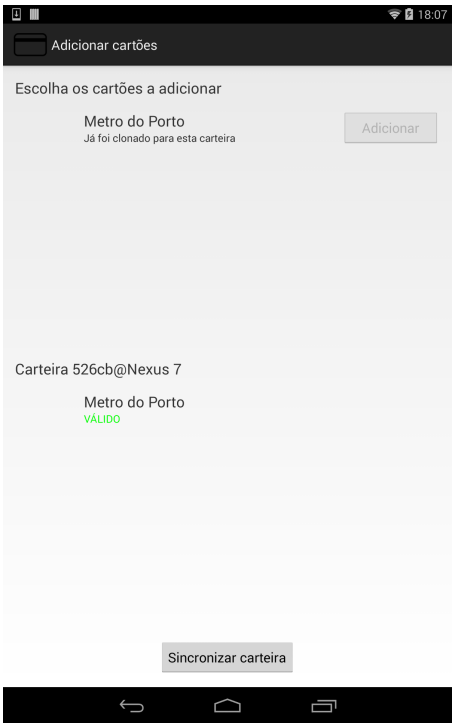

Figura 5.8: *Activity* de gestão de carteira, retirada da aplicação Android desenvolvida

---

Opções da aplicação

A janela de opções pretende que o utilizador possa adaptar a aplicação às suas necessidades. Assim, se este o desejar, a *Activity* desenvolvida no âmbito das opções da aplicação permite redefinir os seguintes dois parâmetros:

1. padrão de segurança;
2. tempo de desbloqueio da aplicação.

Pelos motivos se segurança referidos anteriormente, a alteração se qualquer um destes parâmetros requer a prévia inserção da palavra-passe por parte do utilizador.

5.3.4 Armazenamento

Base de dados *SQLite*

A aplicação desenvolvida tem a necessidade de armazenar dados no dispositivo relativos aos processos de clonagem. Para o efeito, foi criada uma tabela no SGBD *SQLite*, incluído na plataforma *Android*. Segundo a documentação do *Android* [38], qualquer base de dados criada pode ser acedida a partir de qualquer classe da aplicação, mas não fora da aplicação, facto que se revela importante para a segurança dos dados armazenado. Assim, as informações armazenadas, relativas aos clones, são:

- pseudónimo;

- nome da instituição;
- chave pública de Diffie-Hellman;
- chave privada de Diffie-Hellman;
- perfil;
- assinatura digital;
- validade;
- estado.

O SDK do *Android* disponibiliza a classe SQLiteOpenHelper [39] que permite gerir quer a criação, quer a versão da base de dados. Esta gestão é efetuada através da classe DBHelper, criada para o efeito, que está extendida à classe SQLiteOpenHelper e que implementa dois métodos da última:

1. onCreate - é executado o SQL de criação da tabela onde os dados relativos aos clones de cartões estão armazenados;
2. onUpgrade - a tabela dos clones é removida e, de seguida, é chamado o método onCreate.

O acesso aos dados armazenados na base de dados é encapsulado na classe ClonesDataSource, implementada para o efeito. Esta classe disponibiliza um conjunto de métodos públicos que auxiliam a inserção, atualização e consultas de dados:

```
public class ClonesDataSource {

  ...

  public synchronized void insertClone(Clone clone) {...}

  public synchronized boolean exists(String clonePseudonym) {...}

  public synchronized Clone getClone(String pseudonymToSearch) {...}

  public synchronized Clone getCard(byte[] orgHash) {...}

  public synchronized void update(Clone clone) {...}

  public synchronized void getAllClones(List<Clone> myClones) {...}

  public synchronized long numberOfClones() {...}

  public synchronized void deleteAllClones() {...}
}
```

Extrato 5.5: Métodos desenvolvidos na classe ClonesDataSource

SharedPreferences

Para além dos dados relativos aos clones, armazenados na base de dados referida anteriormente, é também necessário guardar o *token* de sessão e o estado da aplicação. Para este efeito não se justifica o armazenamento numa base de dados pelo que foi utilizado a *framework* SharedPreferences do SDK do Android. Segundo a documentação [38], a classe SharedPreferences permite armazenar tipos de dados primitivos em pares chave-valor. Por omissão, os dados armazenados são privados, isto é, só a aplicação que os armazenou é que, mais tarde, os pode consultar, modificar e apagar. A classe SharedPreferences armazena os dados num ficheiro cujo acesso, por omissão, só é permitido pela aplicação a quem este diz respeito, garantindo assim que outras aplicações não têm acesso aos dados.

A referência para a classe SharedPreferences é obtida através do métodos getSharedPreferences da classe Context e requer dois arumentos:

1. nome do ficheiro;
2. modo de operação.

O nome do ficheiro utilizado é da escolha do programador e é necessário para acessos e edições subsequentes e foi definido no ficheiro strings.xml uma vez que é constante. O modo de operação utilizado foi o MODE_PRIVATE pois, segundo a documentação, desta forma "o ficheiro criado só pode ser acedido pela aplicação chamadora".

```
private AppToken(Context context) {

  /* expireDate é a data/hora atual */
  expireDate = new GregorianCalendar();
  /* referência para as SharedPreferences */
  sharedPrefs = context.getSharedPreferences(
          context.getString(
            R.string.shared_prefs_name),
          Context.MODE_PRIVATE);
}
```

Extrato 5.6: Obtenção da referência para o objeto SharedPreferences

5.4 Prestador de serviços

O prestador de serviços, entidade federada num ponto de confiança, oferece um serviço de autenticação, utilizando um equipamento Android com NFC, aos seus colaboradores. No âmbito deste serviço, foi desenvolvido um protocolo de autenticação, abordado mais à frente, que tem de ser respeitado de forma a não comprometer o correto funcionamento do mesmo. Deste modo, o prestador de serviços é livre de utilizar os recursos *hardware* e *software* que entender para lá do protocolo de autenticação. No âmbito desta dissertação, foi desenhada e implementada uma arquitetura para a entidade prestadora de serviços composta pelos seguintes módulos:

- pórtico;
- sistema de informação;
- ponto de confiança.

5.4.1 Pórtico

O pórtico é o módulo da arquitetura responsável por comunicar com os dispositivos Android dos utilizadores suportando o processo de autenticação destes perante a instituição. O pórtico, sozinho, pode ser capaz de processar todas as etapas do protocolo de autenticação ou, alternativamente, ser integrado com o sistema de informação da instituição e/ou com o ponto de confiança de forma a ser auxiliado. No protótipo desenvolvido, o pórtico é capaz de funcionar nas duas situações anteriormente referidas.

Na prototipagem deste módulo foram utilizador os seguintes elementos:

- Arduino com *shield* NFC;
- leds vermelho e verde para *feedback*;
- Raspberry Pi.

O Arduino tem a função de suportar a comunicação com o dispositivo móvel sobre NFC e o Raspberry Pi de realizar operações computacionalmente exigentes. A necessidade de integração dos dois elementos referidos surgiu porque o Arduino não tem poder computacional suficiente para realizar as operações criptográficas, envolvidas no protocolo de autenticação, em tempo útil, tendo estas sido delegadas para o Raspberry Pi. O funcionamento destes dois está detalhado nas secções seguintes.

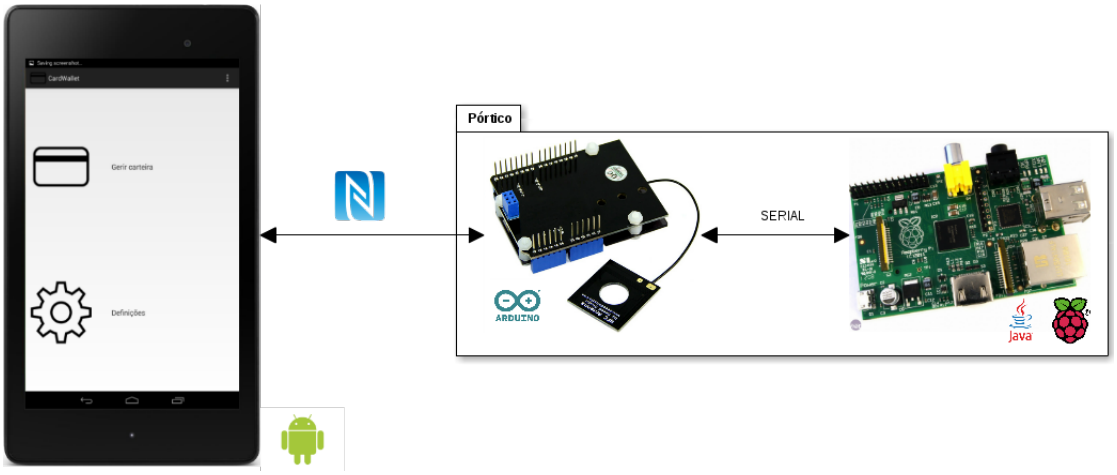

Figura 5.9: Arquitetura do pórtico

Arduino e *shield* NFC

O Arduino começa por configurar a comunicação com *shield* NFC que é feita com recurso ao protocolo SPI. O porto 10 foi o utilizado como pino de ativação do periférico. Durante a configuração, o microcontrolador tenta detetar o periférico e, se tal não for possível, repete o processo do início. Caso o *shield* seja detetado e configurado corretamente, o microcontrolador configura os portos onde estão ligados os leds no modo saída e deixa-os desligados. De modo a identificar a instituição perante os dispositivos, é requisitado ao Raspberry Pi uma *hash* do nome da instituição detentora de pórtico. De sequida, o Arduino pede o valor público de

Diffe-Hellman do pórtico, gerado na aplicação que corre no Raspberry Pi. Fica assim concluído o processo de configuração e obtenção de dados inicial, podendo agora o elemento suportar processos de autenticação.

O Arduino bloqueia então à espera de um cartão de proximidade que, neste caso, será um dispositivo Android com interface NFC capaz de emular o anterior. Assim que um dispositivo for detetado o microcontrolador cria um novo processo de autenticação no sistema computacional que o auxilia. Envia, de seguida, um SELECT APDU ao dispositivo com o AID definido na aplicação Android desenvolvida. O APDU recebido como resposta é validado e, se não tiver o formato esperado, o Arduino vai interromper o processo de autenticação e aguardar por uma nova comunicação com um dispositivo NFC. Por outro lado, caso o APDU se resposta seja corretamente validado, é enviado ao dispositivo o valor público de Diffie-Hellman do pórtico, um número aleatório e a *hash* do nome da instituição com recurso da PUT DATA APDUs. Se o dispositivo não responder indicando sucesso, o processo de autenticação é interrompido e reiniciado. Caso contrário, o microcontrolador requisita ao dispositivo, através de pedidos encapsulados em GET DATA APDUs, os seguintes dados do clone:

- perfil;
- valor público de Diffie-Hellman;
- assinatura digital;
- pseudónimo;
- segredo.

Caso algum APDU de resposta a estes pedidos mencione algum problema no processo, este é reiniciado. Por outro lado, caso as respostas aos pedidos se demonstrem válidas, os dados são enviados para a aplicação a correr no Raspberry Pi.

Por fim, o microcontrolador dá ordem à aplicação a correr no Raspberry Pi para validar o processo de autenticação e esta, com os dados recebidos, efetua as operações criptográficas necessárias. Com base no resultado, é enviado um PUT DATA APDU ao dispositivo móvel a a informar o resultado do processo de autenticação enquanto a ligação NFC estiver ativa.

Quando a ligação NFC se perde, o microcontrolador aguarda novamente que um dispositivo se aproxime e o todo o processo de autenticação é reiniciado.

Raspberry Pi

O Raspberry Pi é um sistema computacional de baixo custo capaz de correr um sistema operacional Linux e uma máquina virtual Java. Como tal, foi desenvolvida uma aplicação nesta linguagem cuja função é auxiliar o Arduino no processo de autenticação, ficando as operações criptográficas e comunicações de rede a cargo da primeira.

A aplicação desenvolvida começa por gerar um par de valores de Diffie-Hellman para serem utilizados no âmbito do protocolo de autenticação. De seguida, o programa carrega a chave pública da instituição que será utilizada para verificar a validade das assinaturas digitais recebidas dos clones. Por fim, o programa inicia a comunicação por série com Arduino recorrendo para isso à biblioteca RXTX [40] que facilita o desenvolvimento de aplicações, na linguagem Java, que utilizem a comunicação serial. Este último passo faz com que o programa a correr no microcontrolador reinicie, seja qual for o ponto em que se encontre. Daqui em

diante a aplicação aguarda que o microcontrolador comunique com ela tendo sido para isso implementada um *callback* que é despoletada quando são recebidos dados do microcontrolador. Os detalhes da comunicação com o microcontrolador estão detalhados na secção Integração.

Integração

Como referido anteriormente, a integração do microcontrolador com a aplicação Java a correr no Raspberry Pi é feita através de comunicação serial. No Arduino a comunicação serial é feita com o suporte da classe Serial [41]. O modelo de Arduino utilizado para implementação foi o Uno que permite comunicação serial através dos pinos 0 (RX) e 1 (TX) ou então através da porta USB. A porta USB foi a utilizada neste projeto pela facilidade de ligar ao Raspberry Pi. O *buffer* de comunicação está deste microcontrolador tem 64 bytes, dos quais 63 são utilizáveis. Posto isto, para quantidades de dados superiores a 63 bytes foi necessário fazer a divisão e, consequentemente, efetuar várias transmissões.

Os dados trocados entre o microcontrolador são:

- valor público de Diffie-Hellman do pórtico;
- valor público de Diffie-Hellman do clone;
- pseudónimo do clone;
- perfil do clone;
- segredo.

A comunicação entre os dois elementos é síncrona, partindo sempre o pedido do microcontrolador e a resposta da aplicação a correr no Raspberry Pi. Para isso, foi definido um conjunto de comandos que permitem ao Raspberry Pi identificar a operação a realizar ou o conjunto de dados a receber:

```
// comandos possveis ARDUINO -> PC
#define UNKNOW_ERROR 0x00
#define GET_DH_PUB_KEY_1 0x01
#define GET_DH_PUB_KEY_2 0x02
#define GET_DH_PUB_KEY_3 0x03
#define GET_RANDOM_VALUE 0x04
#define GET_ORG_HASH 0x05
#define NEW_CARD 0x06
#define SET_CARD_PROFILE 0x07
#define SET_CARD_DH_PUB_VALUE 0x08
#define SET_CARD_SIGNATURE 0x09
#define SET_SECRET 0x0A
#define GET_CARD_VALIDATION 0x0B
#define SET_PSEUDONYM 0x0C

// comandos possveis PC -> ARDUINO
#define ACK 0x01
#define NACK 0x00
```

Extrato 5.7: Lista de comandos e respetivos códigos usados na comunicação entre Arduino e Raspberry Pi

O microntrolador, sempre que pretende interagir com a aplicação Java, começa por enviar um byte com o comando da operação e aguarda a receção da confirmação também na forma de um byte. Por fim, a aplicação Java implementa uma máquina de estados que se altera consoante o comando recebido.

Protocolo de autenticação

O protocolo de autenticação proposto na arquitetura efetua uma troca de três mensagens entre o dispositivo Android e pórtico. No entanto, o pórtico é composto por dois elementos já abordados anteriormente e alguns dos dados recebidos no microcontrolador pela interface NFC têm de ser passados para a aplicação a correr no Raspberry Pi. Por isto, surgiu a necessidade de decompor o protocolo num conjunto de mensagens. Mais, depois de vários testes com o microcontrolador chegou-se à conclusão que este não suportava o envio de APDUs em toda a gama de tamanhos suportada no ISO 7816-4. Consequentemente, os dados que na proposta de protocolo eram enviados numa só mensagem tiveram de ser enviados de forma particionada em vários APDUs. A decomposição do protocolo de autenticação em mensagens entre dispositivo Android e microcontrolador e também entre este último e Raspberry Pi está representada na figura 5.10.

Sempre que a interface NFC estabele um canal de comunicação com um dispositivo Android o microcontrolador envia o comando NEW_CARD à aplicação Java para que esta limpe possíveis dados de um processo de autenticação anterior. De seguida o microcontrolador envia um SELECT APDU ao equipamento Android e caso receba um *Response* APDU indicando o sucesso da operação pede um número aleatório à aplicação que corre no Raspberry Pi através do comando GET_RANDOM_VALUE. Recebido o número aleatório (R1), este é encapsulado num APDU PUT DATA juntamente com uma síntese do nome da instituição e com o valor público de Diffie-Hellman do pórtico. O microcontrolador pede então, por ordem, os seguintes dados do clone selecionado, cada um encapsulado num APDU do tipo GET DATA:

1. perfil;
2. valor público de Diffie-Hellman;
3. assinatura digital;
4. pseudónimo;
5. número aleatório (R2) e segredo.

Todos estes dados recebidos são enviados por serial, juntamente com o comando respetivo, à aplicação presente no Raspberry Pi para validação do processo de autenticação sendo que os que têm mais de 63 bytes são divididos em *chunks* deste tamnho ou menor. Por último, o microcontrolador envia o comando GET_CARD_VALIDATION para que a aplicação Java execute as operações criptográficas necessárias à obtenção do resultado do processo de autenticação. Conforme o resultado recebido pelo microcontrolador da operação anterior, este vai acender o led verde ou vermelho de forma a dar *feedback* visual ao utilizador e enviar o primeiro ao equipamento Android na forma de um APDU do tipo PUT DATA.

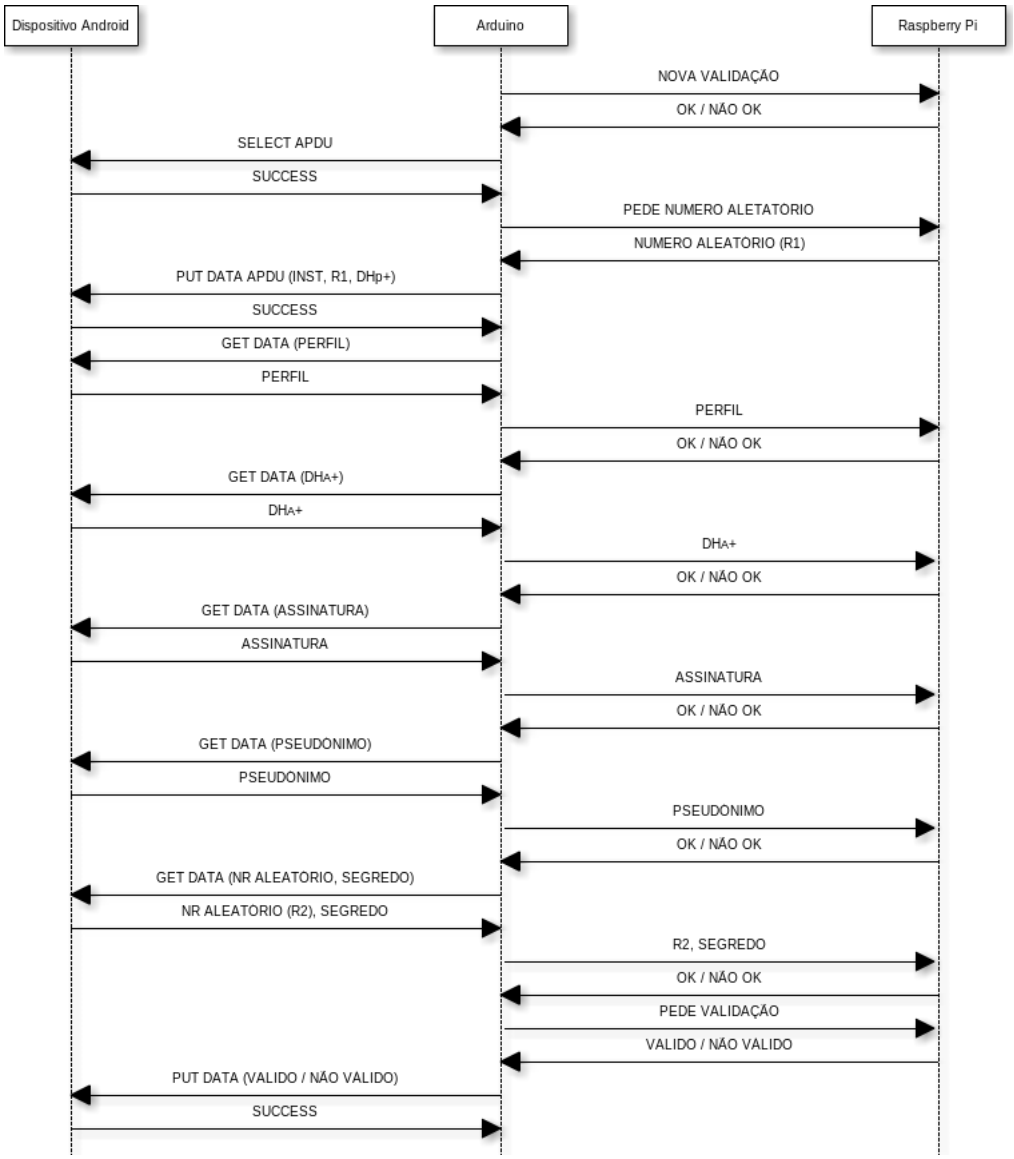

Figura 5.10: Divisão do protocolo proposto em mensagens

5.4.2 Sistema de informação

No âmbito desta dissertação foi desenvolvida uma aplicação para o prestador de serviços para ser utilizada como protótipo. Esta aplicação utiliza a biblioteca Java que permite uma abstração à API REST dos pontos de registo e certificação e também de confiança e permite as seguintes operações:

- *login*;
- emissão de cartões;
- consulta de pedidos pendentes;
- consulta de detalhes de cartões;
- aprovação de clones;
- revogação de clones.

Emissão de cartões

A aplicação desenvolvida permite emitir cartões através do preenchimento de um formulário onde tem de ser selecionado o utilizador alvo e definido o pefil do cartão que, para efeitos de prototipagem, foi preenchido aleatoriamente. Estes dados são posteriormente submetidos ao ponto de confiança, com o auxílio da biblioteca Java desenvolvida, onde o prestador de serviços está federado. Daqui em diante o utilizador a quem foi emitido o cartão conseguirá visualizá-lo na aplicação móvel desenvolvida e poderá cloná-lo para qualquer uma das suas carteiras virtuais.

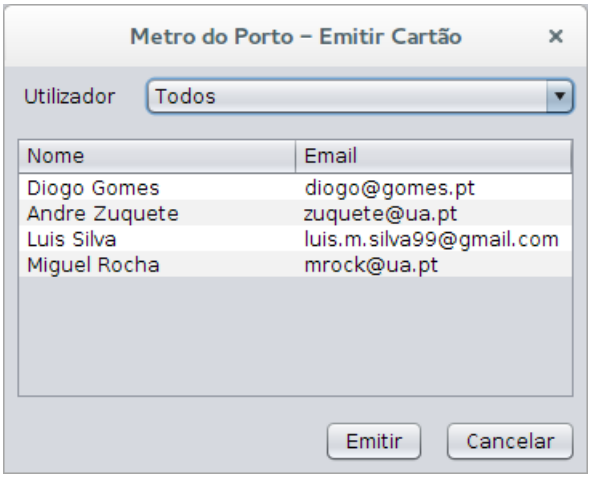

Figura 5.11: Janela de emissão de cartões, retirada da aplicação desenvolvida

Consulta de pedidos pendentes e aprovação de clones

Assim que um utilizador efetua a clonagem de um cartão o primeiro fica a aguardar a aprovação do prestador de serviços. Assim, a aplicação permite consultar os pedidos de

aprovação de clonagens de cartões dos utilizadores. Se o prestador de serviços optar por aprovar o clone é necessário definir a validade do mesmo para que a aplicação, de seguida, emita uma assinatura digital tanto do valor público de Diffie-Hellman como do perfil do segundo e submeta os dados resultantes desta operação ao ponto de confiança. Esta assinatura permite, mais tarde, comprovar a autenticidade e integridade dos dados no processo de autenticação.

Figura 5.12: Janela principal da aplicação, retirada da aplicação desenvolvida

Consulta de detalhes de cartões e revogação de clones

Para todos os cartões emitidos é possível consultar o seu detentor, a lista de clones de cada um e informações associadas a estes últimos:

- pseudónimo;
- carteira para a qual foi clonado;
- validade;
- estado.

Uma das vantagens da solução implementada nesta dissertação é o ágil processo de revogação de clones relativamente ao sistemas comuns de identificação por cartões de proximidade. Assim, na mesma janela o prestador de serviços pode revogar rapidamente um clone em específico ou até todos. A revogação de um clone é comunicada ao prestador de serviços e, mesmo que a carteira virtual que contém um clone revogado não seja atualizada após este processo, o processo de autenticação falha aquando da verificação do estado do primeiro por parte do pórtico.

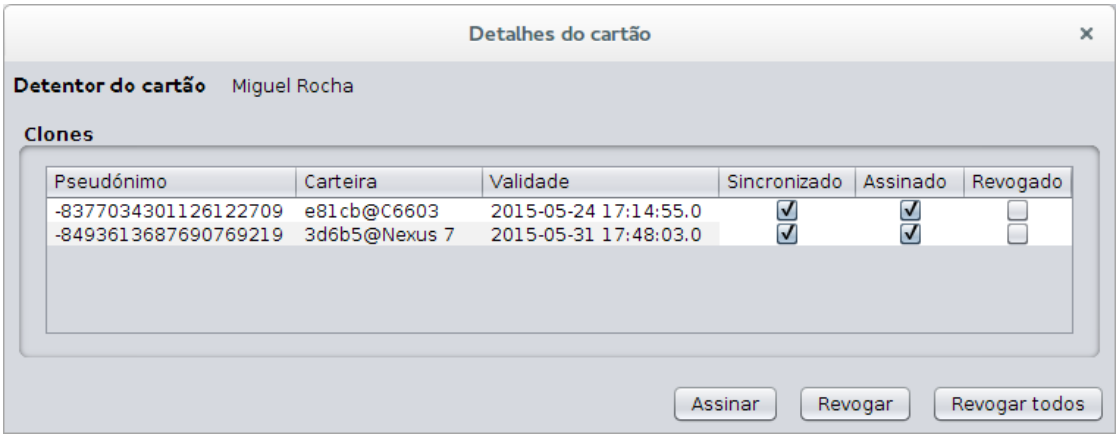

Figura 5.13: Janela que mostra os detalhes de um cartão emitido, retirada da aplicação desenvolvida

Capítulo 6

Análise de segurança

O presente capítulo pretende fazer uma análise de segurança ao protocolo de autenticação tentando abordar possíveis vulnerabilidades e soluções para as mesmas.

6.1 Modelo Dolev-Yao no protocolo de autenticação

O modelo Dolev-Yao [42] pressupõe uma troca de mensagens entre máquinas onde o atacante é quem transporta as primeiras, isto é, tem a capacidade de as intercetar e analisar, sendo as operações criptográficas inerentes às mensagens o seu único obstáculo.

Na implementação do protocolo de autenticação foi necessário dividir o protocolo original num conjunto de mensagens de tamanho mais pequeno, passíveis de serem transmitidas pelo canal de comunicação NFC. Admitindo que um atacante consegue capturar e armazenar um conjunto vasto de mensagens trocadas em processos de autenticação válidos, caso este consiga identificar um APDU PUT DATA vindo do pórtico, que transporta um identificador do prestador de serviços, um número aleatório e um valor público de Diffie-Hellman do pórtico, poderá autenticar-se com sucesso se reproduzir as mensagens de resposta seguintes. Para tal, é necessário também que o clone que originou as mensagens num processo de autenticação anterior não tenha sido revogado entretanto.

Admitindo, mais um vez, que o atacante tem na sua posse um conjunto vasto de mensagens, este pode forjar o identificador da instituição presente no APDU PUT DATA enviado pelo pórtico ao equipamento Android. Caso o equipamento Android não tenha nenhum clone de cartão capaz de responder aquele identificador, o APDU de resposta indica insucesso. O contrário também é válido, caso o identificador seja alterado e o equipamento responda com sucesso, significa que tem um clone capaz de autenticar o possuidor do equipamento naquele pórtico. Deste modo, o atacante consegue saber se aquela carteira virtual tem ou não clones capazes de integrar um processo de autenticação perante um dado prestador de serviços.

Por fim, o atacante pode ainda tentar forjar os dados relativos ao clone e que são enviados do equipamento Android para o pórtico. No entanto, este processo não lhe dá nenhuma vantagem pois a assinatura digital do clone serve precisamente para garantir a origem e integridade dos dados com origem no *smartphone*.

6.2 Ataque de negação de serviço ao pórtico

Um possível ataque é a negação de serviço do pórtico, o que coloca em causa o normal funcionamento do sistema impedindo processos de autenticação dos utilizadores perante os prestadores de serviços.

No caso do pórtico implementado, assim que é criado um canal de comunicação por NFC, é o microcontrolador a enviar o primeiro APDU, colocando-se de seguida à espera de um APDU de resposta. Este APDU, proveniente de um dispositivo externo, é validado à chegada no seu conteúdo e tamanho. Caso alguma destas caraterísticas não cumpra os requisitos esperados o microcontrolador interrompe o canal de comunicação e ativa o led vermelho, de forma intermitente, durante cerca de três segundos e, consequentemente, o processo de autenticação é interrompido. No fim deste tempo o pórtico coloca-se novamente à escuta de dispositivos NFC, tentando criar um canal de comunicação com um equipamento que surja no seu raio de ação para proporcionar autenticação. A validação dos APDUs aplica-se a todas as mensagens externas recebidas e o procedimento seguido pelo microcontrolador, caso as suas caraterísticas não sejam as esperadas, é sempre o referido anteriormente.

Outra possível situação anormal é o canal de comunicação ser interrompido por iniciativa do atacante durante a receção ou envio de um APDU. A biblioteca utilizada para manipulação do módulo NFC prevê esta situação, retornando um código de erro que tem de ser tratado pelo programador. Neste caso, a abordagem foi a mesma seguida aquando da receção de APDU indevidamente formatados: o led vermelho acende de modo intermitente, durante três segundos, e após esse período o microcontrolador fica novamente na expetativa de estabelecer um novo canal de comunicação por NFC com um dispositivo que penetre o seu raio de ação de forma a desencadear um processo de autenticação.

6.3 Ataque de negação de serviço a um ponto de confiança

O ponto de confiança é um módulo fulcral ao bom funcionamento do serviço uma vez que armazena um conjunto de dados relativos aos cartões virtuais emitdos, resptivos processos de clonagem e estado dos clones. Para além disto, o ponto de confiança é um elemento de suporte a comunicações entre utilizadores e prestadores de serviços, pelo que, admitindo um ataque de negação de serviço bem sucedido, o seu normal funcionamento colocaria em causa as seguintes operações:

- federação de prestadores de serviços;
- emissão, clonagem e revogação de cartões;
- verificação do estado dos clones.

Todos os serviços disponibilizados pela API REST do ponto de confiança requerem um *token* de sessão válido para a execução das operações. Assim, como a validade do *token* é verificada antes de qualquer outra operação, junto do ponto de registo e certificação, não é utilizada a totalidade dos recursos que o serviço chamado necessita, uma vez que a sua execução é interrompida quando o *token* não é válido, evitando alguma sobrecarga em consultas à base de dados e manipulação de dados em JSON.

Apesar de uma possível falha do ponto de confiança, o processo de autenticação dos utilizadores junto dos emissores dos cartões através de NFC não fica comprometido pois todas

operações necessárias ao seu funcionamento são efetuadas localmente, no contexto do prestador de serviços. No entanto, como a revogação não está disponível no ponto de confiança, esta tem de ser efetuada diretamente junto dos prestadores de serviços de modo a não permitir que o atacante beneficie da situação.

6.4 Ataque de homem no meio ao ponto de confiança

Como referido no anteriormente, o servidor aplicacional do ponto de confiança dispõe de um certificado assinado pela CA, representada nesta arquitetura pelo ponto de registo e certificação, que lhe permite disponibilizar toda a API REST através do protocolo HTTPS com o objetivo de oferecer segurança às comunicações.

Segundo o artigo [43], o protocolo HTTPS, por si só, não garante nenhum tipo de segurança e, nas circunstâncias que iremos abordar, é vulnerável a ataques de homem no meio. Refere ainda que este protocolo é usado pelas aplicações web para "garatir a privacidade e segurança" de dados sensíveis e os utilizadores tendem a confiar nos websites assim que o navegador lhes mostra o aloquete alusivo a uma ligação segura. O problema apontado pelo artigo é a utilização excessiva de certificados autoassinados, nos quais o utilizador tende a confiar, mesmo após o alerta de segurança típico do navegador e que abrem uma vulnerabilidade no protocolo. Quando a aplicação cliente carrega conteúdo através de HTTPS de um servidor tem, em primeiro lugar, de receber o certificado de chave pública. Se um atacante intercetar o envio desse certificado, pode substituí-lo por um outro certificado auto-assinado gerado por si, no qual a aplicação cliente confiará. Assim, a aplicação cliente, ao invés de efetuar uma comunicação segura com o servidor, está, na realidade, a comunicar com o atacante. Este último é capaz de intercetar os dados trocados, de os decifrar com a chave privada correspondente ao certificado de chave pública por si gerado, de os manipular e, por fim, de os cifrar com a chave pública do servidor antes de os submeter a este último. No entanto, na solução implementada, o certificado raíz da CA é distribuído juntamente com as aplicações desenvolvidas, pelo que não será possível a um atacante forjá-lo e realizar um ataque de homem no meio. Note-se que ao confiar no certificado raíz da CA, as aplicações confiam também em qualquer certificado presente numa cadeia do qual o primeiro é raíz.

6.5 Ataque ao dispositivo Android

O dispositivo Android, como referido anteriormente, tem a si associado uma carteira virtual e armazena todos os dados dos clones a ela associados. Tendo em conta a sensibilidade dos dados, é um ponto de interesse para um atacante pois caso consiga o acesso aos dados poderá facilmente personificar o real dono dos cartões.

Existem dois tipos de atacante ao dispositivo Android que interessa realçar: o seu possuir e os demais.

6.5.1 Ataque por parte do dono do dispositivo

Supondo que o dono do dispositivo consegue acesso a todos os dados relativos à aplicação, isto é, ao *token* de sessão, ao padrão de segurança e a todos as informações relativas à carteira virtual, não conseguirá nada mais do que autenticar-se a si próprio, não colocando em causa o normal funcionamento do sistema implementado.

Caso o atacante consiga reproduzir, na íntegra, os dados num outro dispositivo toda a carteira virtual será clonada, isto é, os dados de cada clone são copiados para outro dispositivo que permitirá também processos de autenticação. Este processo não representa prejuízo para os prestadores de serviços que já permitiam a clonagem de forma protocolar. No entanto, isto poderá representar um problema para as instituições que originalmente não permitem a clonagem dos seus cartões. O processo de clonagem implementado prevê um pseudónimo, um par de valores Diffie-Hellman, uma validade e assinaturas digitais diferentes para clones do mesmo cartão, algo que não acontece neste tipo de ataque. Consequentemente, o processo de revogação, que serve para revogar um dado clone sem revogar os demais, invalidaria também os clones obtidos através do ataque aqui abordado, por cópia direta.

6.5.2 Ataque por parte de terceiros

O ataque mais simples que alguém mal intencionado poderá fazer, na posse de um equipamento roubado, é o de tentar desbloquear a aplicação através de tentativas, esperando que alguma corresponda ao padrão de segurança definido pelo proprietário do equipamento. Caso o *token* de sessão tenha expirado, o atacante terá também de introduzir o par de credenciais válidas para conseguir interagir, de alguma forma, com a aplicação.

À semelhança do que foi descrito anteriormente, admitindo que um atacante na posse de um dispositivo roubado consegue reproduzir os dados num outro equipamento, este pode beneficiar de processos de autenticação bem sucedidos com a diferença de que, neste caso, está também em causa uma personificação.

Em situações de roubo é espectável que o lesado tire partido do solução implementada para revogar, o mais rapidamente possível, os clones associados às carteiras virtuais afetadas, invalidando assim qualquer tipo de ataque por parte de terceiros. O processo de autenticação falhará assim que o pórtico verifique o estado relativo ao clone com aquele pseudónimo, que neste caso está revogado.

Por último, existe ainda a situação em que o dispositivo Android pode ter um *software* malicioso a correr, sem consentimento e conhecimento do seu dono, em que este reporte os dados relativos à aplicação para um contexto externo ao do *smartphone*. Neste caso poderia ocorrer o mesmo processo de replicação dos dados da aplicação para um outro dispositivo, que acarreta as consequências já descritas anteriormente com a desvantagem de o dono das credenciais não tomar a decisão de revogar os clones por desconhecimento de causa.

Capítulo 7

Casos de uso

Este capítulo aborda situações do quotidiano onde a solução proposta é passível de ser implementada.

7.1 Bilhética

Segundo G. Lopes [44], o universo da bilhética não contempla "um padrão de desenvolvimento de aplicações" para tratar este assunto. Segundo o mesmo, existem um conjunto de soluções generalistas, principalmente baseadas em *hardware* e com escasso desenvolvimento de *software* de suporte. Este facto deve-se sobretudo à dificuldade que existe em identificar os parâmetros importantes da área que levam à falta de normas e padrões. O autor, no universo do sistema de transportes, identifica ainda um conjunto variado de cenários:

- tipologia de funcionamento: por zona, destino, entre outras;
- tipo de transporte: autocarro, comboio, expresso;
- tipo de cobrança: por viagem, por tempo, por zona.

A solução implementada no âmbito da dissertação é flexível a todos estes contexto, pois a zona de memória disponibilizada pode ser escrita com aquilo que o prestador de serviços entender. Para além disso, o sistema de informação da instituição também pode armazenar dados relativos ao cartão, proporcionando ainda mais liberdade na implementação de soluções.

7.1.1 Espetáculos

À semelhança dos bilhetes de transportes, também os bilhetes de espetáculos têm um conjunto de parâmetros variável. No caso de um bilhete que só dê acesso a um espaço uma única vez, assim que o pórtico validar e permitir o acesso deve revogar imediatamente todos os clones do cartão que representa o bilhete. Por outro lado, caso seja suposto o bilhete dar acesso a um local em mais do que uma ocasião, os clones não podem ser revogados e lógica de negócio tem de ser mantida do lado dos sistemas de informação do prestados de serviços. Por exemplo, num bilhete que permita um número de acessos limitado a um espaço, o número restante de acessos tem de ser mantido no sistema de informação e assim que esgotar o cartão e todos os clones devem ser automaticamente revogados.

7.1.2 Transportes

A situação dos transportes é em tudo semelhante à da bilhética de espetáculos. Na solução desenvolvida, a zona de memória não é passível de ser alterada após a emissão do cartão, pelo que casos de emissão de bilhetes onde é necessário um contexto (tempo restante, viagens por utilizar, validade do passe mensal) requerem uma cooperação do sistema de informação do prestador de serviços. Esta cooperação é também importante para invalidar casos de fraude, garantindo que hipotéticos ataques à zona de memória e respetiva assinatura digital não representam o prejuízo do prestador de serviços.

7.2 Controlo de acessos

O controlo de acessos pretende ditar que utilizadores têm permissões para aceder a um dado objeto. Existem vários modelos de controlo de acessos, nos quais a solução desenvolvida pode ser utilizada:

- *Discretionary Access Control* (DAC);
- *Mandatory Access Control* (MAC);
- *Role-based Access Control* (RBAC).

7.2.1 *Discretionary Access Control* (DAC)

No modelo DAC as permissões de acesso aos objetos são tipicamente definidas numa matriz bidimensional. Numa das suas dimensões estão representados os primeiros e na outra dimensão estão representados os utilizadores, criandoa assim uma *Access Control List* (ACL).

Na solução implementada os objetos a que um utilizador tem acesso podem ser armazenados, por exemplo, na zona de memória do cartão. Como os dados aí armazenados estão digitalmente assinados, o pórtico pode confiar nos dados que recebe, caso a assinatura seja válida, e verificar se o possuidor do equipamento Android tem ou não acesso ao objeto. Neste caso é desnecessário o suporte do sistema de informação do prestador de serviços, podendo todo o processo de autenticação ser realizado ao nível do pórtico.

7.2.2 *Mandatory Access Control* (MAC)

Neste modelo todos os objetos encontram-se qualificados com base na sua importância, tais como secreto, muito secreto, entre outras. A qualificação do objeto é frequentemente denominada de *security label*. Os utilizadores têm também atribuído um nível de acesso que lhes permite aceder a todos os objetos qualificados com níveis iguais ou inferiores.

Na solução implementada o perfil do cartão pode conter o nível de acesso do utilizador. Este atributo é imutável e a sua integridade é assegurada pela assinatura digital emitida pelo prestador de serviços. Caso o pórtico tenha noção do nível de segurança do objeto que protege, a validação do acesso é passível de ser efetuada ao nível do primeiro, sendo desnecessária a comunicação com o sistema de informação da entidade institucional que representa.

7.2.3 *Role-based Access Control* (RBAC)

No modelo RBAC um utilizador fica associado a um grupo ou papel e a validação do acesso acente num destes últimos. Cada objeto tem um conjunto de permissões que determina que grupos ou papéis lhe podem aceder, nunca especificando utilizadores.

Na solução desenvolvida o perfil associado ao cartão pode armazenar o conjunto de grupos e papés aos quais o utilizador pertence. Como referido anteriormente, o perfil está assinado digitalmente pela entidade emissora do cartão pelo que é possível garantir ao pórtico, através de operações criptográficas, que o primeiro não foi forjado. Deste modo, será possível autenticar o utilizador localmente e validar ou negar o acesso deste ao objeto.

Capítulo 8

Considerações finais

Neste capítulo é feita uma análise geral ao trabalho desenvolvido ao longo da dissertação e aquilo que poderá ser feito como trabalho futuro.

8.1 Conclusões

Idealmente, o sistema de autenticação com Android via NFC deveria de ser capaz de substituir qualquer cartão de proximidade, sem que para isso fosse necessário o desenvolvimento de um novo protocolo de autenticação, um pórtico ou qualquer outro *software* de suporte. No entanto, devido heterogeneidade de soluções de autenticação com cartões de proximidade, pórticos e protocolos (muitos deles proprietários) não foi possível reutilizar tecnologia e substituir apenas os cartões pelos equipamentos Android. Assim, foi feita uma lista de requisitos do projeto, pensado aquilo que poderia ser o cartão virtual e, com base nisto, desenvolvido não só um protocolo de autenticação para equipamentos Android com NFC mas também um conjunto de *software* de suporte ao funcionamento da solução.

A pesquisa ao nível da integração da tecnologia NFC com o sistema operativo Android começou a ser feita em meados de Outubro de 2013 e, na altura, quer o sistema operativo, quer o SDK da plataforma ainda não suportavam emulação de cartões. Por isso, nessa altura, as experiências efetuadas com o equipamento Android e o *shield* NFC do Arduino recairam no protocolo SNEP que não se revelou o ideal para a suportar um protocolo de autenticação, devido às limitações impostas pelo SDK do Android que não expõe toda a pilha protocolar. Mais tarde, a versão 4.4 do Android trouxe o modo de emulação de cartões e a possibilidade de utilizar o ISO 7816-4 para suportar o protocolo de autenticação. Caso isto não acontecesse a tarefa de desenvolver o protocolo de autenticação sobre a tecnologia NFC teria sido mais exigente, requerendo provavelmente a utilização do NDK [45] do Android para um trabalho a um nível mais baixo no que à tecnologia de comunicação diz respeito.

Assim que foi conseguida uma forma de comunicação por NFC robusta entre o equipamento Android e o microcontrolador, objetivo inicial, começou a ser pensada a lógica de negócio que poderia ser suportada pela solução a desenvolver. Assim, chegou-se a uma lista de requisitos do sistema que permitiu pensar no conceito daquilo que seria o cartão virtual. Por fim, foi desenhado o protocolo de autenticação para autenticar pessoas com cartões virtuais, especificadas as APIs REST necessárias e as bases de dados. Posto isto, iniciou-se a fase de implementação do projeto que, inevitavelmente, contou com alguns obstáculos que foram ultrapassados com a orientação e motivação prestada pelos Professores André Zúquete e Diogo

Gomes. Aos dois, uma vez mais, o meu obrigado.

8.2 Satisfação face aos objetivos iniciais

A dissertação em causa proponha-se a desenvolver um sistema de autenticação utilizando equipamentos Android com tecnologia NFC, retirando aos cidadãos a necessidade de utilização de cartões de proximidade. Para além disto, proponha-se a agilizar o processo de revogação, bastante moroso caso o número de cartões a invalidar seja elevado. À data de escrita deste documento, estes objetivos tinham sido alcançados com o desenvolvimento das aplicações mencionadas em capítulos anteriores. Penso ainda que posso afirmar que se foi para além do objetivo inicial, tendo sido desenvolvida uma arquitetura capaz de suportar toda uma lógica de negócio que não estava prevista inicialmente.

8.3 Trabalhos futuros

Apesar do foco central da dissertação ser um sistema de autenticação com equipamentos Android e tecnologia NFC, foram desenvolvidas outras questões para serem utilizadas como suporte ao primeiro. Assim, durante o período de desenvolvimento, foram surgindo novas ideias e também a necessidade de adaptar algum do trabalho que já tinha sido feito. Como o tempo é limitado, não foi possível realizar todas as ideias que surgiram, no entanto ficam algumas delas:

1. implementação da aplicação móvel noutros sistemas operativos (iOS, Windows Phone);
2. implementação das ferramentas desenvolvidas em Java Swing em ambiente *web*, eliminando a complexidade de distribuição das aplicações;
3. implementação do pórtico com recurso a um Raspberry Pi em conjunto com um *shield* NFC compatível com o mesmo, eliminando o Arduino, o que reduz custos e tempo gasto com a comunicação série;
4. melhoramentos na interface gráfica da aplicação Android, visando uma experiência de utilização mais rica e intuitiva;
5. exploração de outros mecanismos de autenticação na aplicação Android, como por exemplo reconhecimento facial;
6. redundância ao nível do ponto de registo e certificação e pontos de confiança para evitar possíveis quebras de serviço;
7. introdução de novas instruções no protocolo de comunicação desenvolvido, para permitir a manipulação de dados de cartões e clones via NFC;
8. utilização de certificados no processo de autenticação, emitidos pelos prestadores de serviços;
9. introdução de OAuth [46] na arquitetura;
10. disponibilização de bibliotecas de programação para abstração da API REST em outras linguagens;

11. automatização do processo de aprovação de clones.

Bibliografia

[1] Alexandra Dmitrienko, Ahmad-Reza Sadeghi, Sandeep Tamrakar, and Christian Wachsmann. Smarttokens: Delegable access control with nfc-enabled smartphones. In Stefan Katzenbeisser, Edgar Weippl, L.Jean Camp, Melanie Volkamer, Mike Reiter, and Xinwen Zhang, editors, *Trust and Trustworthy Computing*, volume 7344 of *Lecture Notes in Computer Science*, pages 219–238. Springer Berlin Heidelberg, 2012.

[2] S.L. Ghiron, S. Sposato, C.M. Medaglia, and A. Moroni. Nfc ticketing: A prototype and usability test of an nfc-based virtual ticketing application. In *Near Field Communication, 2009. NFC '09. First International Workshop on*, pages 45–50, Feb 2009.

[3] NFC Forum. *Simple NDEF Exchange Protocol Technical Specification*, 08 2011. Rev. 1.0.

[4] hamishwillee (nome de utilizador). Entendendo as mensagens NFC Data Exchange Format (NDEF) - Nokia Developer Wiki. "`http://developer.nokia.com/community/wiki/Entendendo_as_mensagens_NFC_Data_Exchange_Format_(NDEF)`", 2013. [Online; acedido 22-Outubro-2013].

[5] mwoolley (nome de utilizador). NFC - Virtual Target Emulation - BlackBerry Support Community Foruns. "`http://supportforums.blackberry.com/t5/Java-Development/NFC-Virtual-Target-Emulation/ta-p/1509687`", 2012. [Online; acedido 26-Outubro-2013].

[6] Google. Host-based Card Emulation | Android Developers. "`https://developer.android.com/guide/topics/connectivity/nfc/hce.html`", 2013. [Online; acedido 21-Dezembro-2013].

[7] André Zúquete. *Segurança em Redes Informáticas*. FCA - Editora de Informática, Abril 2013.

[8] Manish Bhuptani and Shahram Moradpour. *RFID Field Guide: Deploying Radio Frequency Identification Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[9] António Almeida, Ricardo André Fernandes Costa, Luís Lima, and Paulo Novais. Non-obstructive authentication in AAL environments. *IOS Press*, 2011.

[10] Google. Near field communication. "`http://developer.android.com/guide/topics/connectivity/nfc/index.html`", 2013. [Online; acedido 11-Outubro-2013].

[11] Google. A smart, virtual wallet for in-store and online shopping - google wallet. "`http://www.google.com/wallet/`", 2013. [Online; acedido 19-Outubro-2013].

[12] César Filipe Duarte Cardoso. Controlo de acessos. Master's thesis, Universidade de Aveiro, 2013.

[13] T. Dimitriou. A Lightweight RFID Protocol to protect against Traceability and Cloning attacks. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 59–66, Sept 2005.

[14] NFC guy (nome de utilizador). Retrieve NFC Id in Android P2P. "http://stackoverflow.com/questions/11308636/retrieve-nfc-id-in-android-p2p". [Online; acedido 8-Dezembro-2013].

[15] Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange, 2013.

[16] MEO CardMobili. "https://www.cardmobili.com", 2014. "[Online; acedido 14-Maio-2014]".

[17] B.C. Wu, P.F. Liang, M.H. Hsu, and W.C. Ho. RFID system using mobile phone as device of access control and security response, May 25 2006. US Patent App. 11/041,946.

[18] Microsoft. Wallet for Windows Phone 8. "http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj207032(v=vs.105).aspx". [Online; acedido 5-Junho-2014].

[19] Fernando Manuel Pires Mateus das Graças. Pagamentos em dispositivos Android com NFC. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2012.

[20] History of Near Field Communication. "http://www.nearfieldcommunication.org/history-nfc.html". [Online; acedido 14-Maio-2014].

[21] Identification cards – Contactless integrated circuit(s) cards – Proximity cards, 2000.

[22] Oracle. javax.crypto (Java Platform SE 7). "http://docs.oracle.com/javase/7/docs/api/javax/crypto/package-summary.html". [Online; acedido 23-Fevereiro-2014].

[23] Google. Services | Android Developers. "http://developer.android.com/guide/components/services.html", 2013. [Online; acedido 14-Novembro-2013].

[24] Google. NFC Basics | Android Developers. "http://developer.android.com/guide/topics/connectivity/nfc/nfc.html", 2013. [Online; acedido 29-Setembro-2013].

[25] Google. android.nfc | Android Developers. "http://developer.android.com/reference/android/nfc/package-summary.html", 2013. [Online; acedido 28-Setembro-2013].

[26] Google. Ndef | Android Developers. "http://developer.android.com/reference/android/nfc/tech/Ndef.html", 2013. [Online; acedido 26-Setembro-2013].

[27] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[28] M.E. Hellman, B.W. Diffie, and R.C. Merkle. Cryptographic apparatus and method, April 29 1980. US Patent 4,200,770.

[29] E. Rescorla. Diffie-Hellman Key Agreement Method. RFC 2631 (Proposed Standard), June 1999.

[30] A. Freier, P. Karlton, and P. Kocher. The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (Historic), August 2011.

[31] ECMA. *Near Field Communication Interface and Protocol (NFCIP-1)*, 2nd edition, December 2004.

[32] Keunwoo Rhee, Jin Kwak, Seungjoo Kim, and Dongho Won. Challenge-response based rfid authentication protocol for distributed database environment. In Dieter Hutter and Markus Ullmann, editors, *Security in Pervasive Computing*, volume 3450 of *Lecture Notes in Computer Science*, pages 70–84. Springer Berlin Heidelberg, 2005.

[33] Apache Software Foundation. HttpServlet. "`https://tomcat.apache.org/tomcat-5.5-doc/servletapi/`". [Online; acedido 28-Maio-2014].

[34] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785, 6266, 6585.

[35] GSON. "`https://code.google.com/p/google-gson/`". [Online; acedido 28-Maio-2014].

[36] Oracle. UUID. "`http://docs.oracle.com/javase/7/docs/api/java/util/UUID.html`". [Online; acedido 28-Maio-2014].

[37] android-lockpattern. "`https://code.google.com/p/android-lockpattern/`". [Online; acedido 10-Março-2014].

[38] Google. Storage Options | Android Developers. "`http://developer.android.com/guide/topics/data/data-storage.html`", 2013. [Online; acedido 4-Abril-2014].

[39] Google. SQLiteOpenHelper | Android Developers. "`http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html`", 2013. [Online; acedido 4-Abril-2014].

[40] rxtx - a java cross platform wrapper library for the serial port. "`https://github.com/rxtx/rxtx`". [Online; acedido 10-Março-2014].

[41] Arduino. Arduino - serial. "`http://arduino.cc/en/reference/serial`", 2014. [Online; acedido 2-Abril-2014].

[42] D. Dolev and Andrew C. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, Mar 1983.

[43] F. Callegati, W. Cerroni, and M. Ramilli. Man-in-the-Middle Attack to the HTTPS Protocol. *Security Privacy, IEEE*, 7(1):78–81, Jan 2009.

[44] Guilherme Cardoso Lopes. Sistema de bilhética e apoio á exploração. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2008.

[45] Google. Android NDK. "`https://developer.android.com/tools/sdk/ndk/index.html`". [Online; acedido 4-Abril-2014].

[46] Barry Leiba. OAuth Web Authorization Protocol. *IEEE Internet Computing*, 16(1):74–77, 2012.