UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Implementing a Loyalty Card for Smartphones using a Bitcoin Like Approach

Sérgio Manuel Rodrigues da Costa

Dissertação para obtenção do grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Pedro R. M. Inácio

Covilhã, Outubro de 2014

Dedication

"... to my beloved parents, for the love, strength and patience that they gave me during the development of this work."

Acknowledgments

During the year in which I have been involved in the development of this masters programme, I had the fortunate oportunity to rely on the help and support of many people, who directly or indirectly contributed to its successful conclusion.

First of all, I would like to express my gratitude to my supervisor, Professor Pedro Ricardo Morais Inácio, first for accepting to be my advisor, for the frankness with which he led this unfolding of hard work, for his expertise, guidance, support and tireless encouragement in every step of this journey.

I am most grateful to the University of Beira Interior colleagues and Multimedia Signal Processing - Covilhã (MSP-Cv) group members for all their support. MSP-Cv is a group of the Instituto de Telecomunicações with a laboratory in University of Beira Interior (UBI), which provided me with a comfortable environment to work in.

I owe a particular acknowledgement to my friend Orlando Pereira, for all his support and patience.

I am very grateful to my fellow, Fábio Rodrigues, for his help and revision of this document.

To all my friends, especially to my girlfriend Dóris, for encouraging me in the most complicated moments and inspiring me to not quit.

Last, but not the least, I would like to thank my father Valdemar and mother Madalena, specially for their love and strength throughout these months of dedication.

Thank you all.

Resumo

As moedas criptográficas têm vindo a prosperar nos últimos cinco anos, especialmente desde o aparecimento da Bitcoin em 2009. Fatores como as vantagens específicas deste tipo de moedas, a atual conjetura económica e a evolução tecnológica, estão a estimular a sua popularidade. Em alguns países, os sistemas monetários criptográficos são considerados pelo governo, alternativas viáveis ao dinheiro real e a Bitcoin está efetivamente a ser utilizada em transações por todo o mundo. O sucesso da Bitcoin é essencialmente baseado na sua elegante descrição matemática, segurança comprovada pelos seus princípios, pelo seu carácter descentralizado e pela garantia de anonimato. Para além do esforço inicial para implementar corretamente e de forma segura o sistema, e da manutenção das aplicações, a Bitcoin funciona automaticamente com a contribuição dos nós de uma infraestrutura descentralizada. A especificação completa do protocolo está facilmente disponível, por exemplo na Internet, e pode ser utilizada por qualquer pessoa. Este programa de mestrado explorou a possibilidade de usá-lo, com modificações, como uma forma de implementar um sistema de cartões de fidelização eletrónicos. De forma a fazer isso, a moeda criptográfica acima mencionada, foi estudada em detalhe, foram especificados um conjunto de requisitos para o novo sistema e modificações ao protocolo original, e um sistema em *software* foi projetado e implementado na linguagem de programação Java. A especificação das modificações foi realizada, tendo em conta o cenário desta aplicação em particular. As restrições resultantes do cenário da aplicação foram maioritariamente dominadas pelo fato de que a infraestrutura *Peer-to-Peer* (P2P) subjacente era constituída apenas por *smartphones*.

O resultado mais visível deste programa de mestrado é o protótipo completamente funcional de um sistema de cartões de fidelização, composto por uma aplicação para dispositivos móveis, e outra para ser executada do lado do servidor. Este protótipo implementa de raiz, a parte do Bitcoin, a partir do trabalho seminal que o define, juntamente com as alterações que introduzem um agente central para controlar melhor a quantidade de moedas por cliente, e auxiliar no estabelecimento de ligações P2P entre duas aplicações móveis. A versão modificada do sistema é aqui chamada de Bitpoints e a moeda é em vez disso, constituída por pontos. Os benefícios da implementação deste sistema de cartões de fidelização, e de algumas das vantagens da popular moeda criptográfica são, nomeadamente o acesso público à cadeia de blocos para verificação isolada de todas as transações. O cartão de fidelização permite a troca de pontos entre utilizadores e novos pontos de mineração, que é fundamentalmente diferente dos cartões de fidelização atualmente disponíveis.

Dentro do contexto deste programa de mestrado, foi distribuído um inquérito por uma população de 34 indivíduos, que responderam a um conjunto de questões relativas ao manuseamento do protótipo referido anteriormente. A análise dos resultados obtidos permitiu induzir que as pessoas se sentem confortáveis com a aplicação, e que aceitaram o conceito na qual esta é

baseada, preferindo provavelmente um sistema semelhante ao proposto.

Palavras-chave

Android, Bitcoin, Bitpoints, Moeda Eletrónica, Comércio Eletrónico, Cartão de Fidelização, Mineração, Aplicação Móvel.

Resumo alargado

Introdução

Este capítulo sintetiza, mas de forma alargada e em língua portuguesa, o trabalho descrito nesta dissertação. Este capítulo começa pela descrição do seu enquadramento, do problema abordado e os objetivos a alcançar. Posteriormente, são enumeradas e brevemente descritas, as principais contribuições do trabalho apresentado. De seguida, é feita uma revisão sobre o trabalho relacionado na área, bem como a descrição de alguns conceitos fundamentais à discussão aqui contida. A secção seguinte descreve as tecnologias utilizadas no contexto deste programa de mestrado, assim como a engenharia de *software* do sistemas para o qual foi desenvolvido um protótipo. A implementação do protótipo referido é descrita e discutida posteriormente. A penúltima secção apresenta o questionário realizado no âmbito desta dissertação e uma análise resumida dos resultados obtidos. O capítulo termina com uma breve discussão das principais conclusões obtidas e com a apresentação de algumas direções para trabalho futuro.

Neste resumo alargado, os acrónimos estão definidos em Português nas suas formas longas. Contudo a forma curta dos acrónimos utilizados está de acordo com o respetivo acrónimo na língua Inglesa, de maneira a manter a consistência. Adicionalmente, alguns acrónimos poderão não ter tradução direta (por exemplo, referências a serviços ou aplicações), pelo que é mantida a sua designação inglesa e tratados como estrangeirismos.

Enquadramento, Descrição do Problema e Objetivos

Atualmente, temos assistido a um crescimento acentuado do comércio eletrónico. Este tipo de comércio depende significativamente de instituições financeiras que lidam com dinheiro na vida real. Esta dependência encontra-se em mudança, com o aparecimento das moedas criptográficas, motivada pelo facto do sistema clássico para lidar com transações apresentar riscos e desvantagens. Por exemplo, o sistema clássico não garante que uma transação seja irreversível ou que as entidades envolvidas permaneçam anónimas. Aliado ao problema previamente referido, atualmente existem ainda muitos países, em cujas corrupção e inexistência de infraestruturas de segurança financeira, impossibilitam o acesso a instituições financeiras, depósitos bancários seguros ou até mesmo comércio internacional.

Com estes problemas em mente, inspirado pela crise económica que se tem sentido por todo o mundo, e com o enfoque no comércio eletrónico, foi desenvolvido um sistema auto-regulado para criação e suporte de moedas virtuais. O protocolo Bitcoin, criado por *Satoshi Nakamoto*, consiste numa moeda digital completamente descentralizada, suportada por uma rede *Peer-to-Peer* (P2P). O atual e elevado interesse neste protocolo deve-se ao fato de, ao contrário dos bancos reais, ser totalmente anónimo e não existir uma entidade central a regular a gestão de dinheiro. Isto acontece com o auxílio de um histórico de registos público, denominado também

por *blockchain*, partilhado por todos os clientes do sistema e onde é possível explorar todas as transações. A *blockchain* contém, entre outros dados, o endereço de origem, o endereço de destino e o valor da transferência para todas as transações realizadas desde a origem da moeda até ao momento. Este protocolo é baseado numa prova-de-trabalho, realizada pelos clientes ligados, que permite a troca segura de moedas.

Atualmente existe também um grande número de empresas que usam cartões de fidelização de maneira a aumentar a sua base de clientes, seguir os seus hábitos e oferecer descontos em produtos, ofertas, e ainda acumulação de dinheiro que permite a interação com outras companhias ou serviços. Este tipo de fidelização acaba por não ser prática, no sentido em que existe uma enorme quantidade de cartões diferentes, e ainda existem outras maneiras de promover uma marca ou empresa. Um exemplo real desta inconveniência é que muitos destes cartões acabam por ser pouco ou nunca utilizados. Assim, torna-se interessante centralizar todos estes cartões, ou alguns deles, numa simples aplicação, simplificando o seu uso. Os sistemas de pontos na maior parte dos sistemas de fidelização atuais são controlados centralmente e permitem o acesso a um número reduzido de funcionalidades. Por exemplo, não permitem a troca de pontos entre clientes, nem que diferentes companhias aceitem pontos como pagamentos.

Atualmente, a utilização de dispositivos móveis com ligação à Internet é bastante comum nos países desenvolvidos. Existe um grande aumento do número de pessoas que possuem *smartphones* com ligação à Internet. Este programa de mestrado é focado no desenvolvimento de um sistema, composto por uma aplicação para dispositivos móveis (também denominada por Bitpoints), e uma aplicação servidor (destinada ao controlo e configuração do sistema), cujo objetivo principal consiste na substituição dos cartões de fidelização tradicionais oferecidos por marcas e empresas. Este sistema de pontos virtual é baseado no protocolo Bitcoin, alterado de modo a preencher alguns dos requisitos específicos deste cenário.

Esta dissertação apresenta um trabalho que se enquadra na interceção das áreas de *computação móvel* e *desenvolvimento de software*, *comércio eletrónico*, *redes* e *criptografia*. O problema abordado nesta dissertação recai sobre a falta de funcionalidades de alguns cartões de fidelização existentes atualmente, nomeadamente a possibilidade de transferir crédito de um cartão para outro. A limitação da capacidade de processamento e armazenamento nos dispositivos móveis também constitui um problema. A ideia essencial deste programa consiste em implementar um sistema robusto, seguro e mais rico em funcionalidades, completamente suportado por software em que nenhum cartão físico será necessário, munido de um sistema de gestão centralizado.

De forma a atingir os objetivos propostos, o trabalho de mestrado foi estruturado de acordo com as seguintes tarefas:

1. Contextualização com os objetivos propostos e familiarização com as tecnologias envolvidas, nomeadamente com redes P2P totalmente descentralizadas e com o protocolo Bitcoin. Outros sistemas de cartões de fidelização devem também ser estudados durante esta tarefa inicial;

2. A segunda tarefa compreende um conjunto preliminar de experiências relativas à implementação e teste da rede P2P com dispositivos móveis;

3. A terceira tarefa consiste no desenho do novo sistema, a partir da engenharia de *software* e identificação de requisitos;

4. A quarta tarefa foca-se na implementação, teste e melhoramento do sistema concebido. Nesta tarefa, foi feito um esforço para abordar e discutir aspetos de segurança;

5. A quinta tarefa consiste na obtenção da opinião de utilizadores em relação à aplicação desenvolvida e ao sistema de cartões de fidelização a partir de um questionário.

Principais Contribuições

Esta seção apresenta sinteticamente as principais contribuições resultantes do trabalho apresentado nesta dissertação:

1. A principal contribuição consiste na implementação de um protótipo funcional de um cartão de fidelização eletrónico, baseado inteiramente na descrição e especificações do protocolo Bitcoin. O sistema é composto por uma aplicação Android para *smartphones* e outra para o servidor, ambas desenvolvidas na linguagem de programação Java. Uma vez que este sistema não é tão crítico quanto o sistema de moeda eletrónica, este deverá refletir essa diferença. Uma das maiores diferenças é que o sistema não necessita de ser completamente descentralizado e uma entidade central pode ser tolerada. Esta entidade central é implementada como um *rendezvous host*, também chamado Bootstrap *server*. Esta dissertação discute as responsabilidades e contra-partidas associadas a este servidor, apresentando também uma justificação pela qual foi utilizado.

2. A segunda contribuição consiste no facto do sistema ter sido desenhado especificamente, desde o início, para *smartphones*, tendo em conta questões relacionadas com o desenvolvimento de aplicações móveis com cálculos potencialmente pesados. A dissertação também contém a engenharia de software detalhada das aplicações desenvolvidas.

O protótipo desenvolvido para um cartão de fidelização eletrónico avança o estado da arte nesta área específica. Por exemplo, adiciona funcionalidades que os cartões físicos não possuem, nomeadamente a possibilidade de trocar pontos com outros clientes. A segurança do protocolo é assegurada pelo conceito e mecanismos por trás do Bitcoin.

Trabalho Relacionado

Uma das partes mais importantes deste mestrado, refletida no Capítulo 2, diz respeito ao estudo prévio de sistemas de fidelização e moedas criptográficas existentes no mercado atualmente, nomeadamente o protocolo Bitcoin. O capítulo começa por introduzir os cartões de pontos físicos, também conhecidos como cartões de clientes e a sua importância na cativação de clientes [Sev03]. São também apresentadas algumas limitações, como por exemplo a consulta de saldo em qualquer situação, ou a troca de pontos entre clientes.

O capítulo evolui para o estudo de conceitos e mecanismos associados ao protocolo Bitcoin [Nak09]. Antes de iniciar a explicação do protocolo, existem alguns conceitos importantes que devem ser esclarecidos, de maneira a auxiliar a sua compreensão, como por exemplo valores e funções de *hash*, assinaturas digitais, primitivas da criptografia sobre curvas elípticas (da designação inglesa *Elliptic Curve Cryptography* (ECC)), árvores de *Merkle* e endereços públicos [Vau10, PP10, Mer80, Dri13]. São também definidos mais pormenorizadamente conceitos como *blockchain*, *mining*, prova-de-trabalho e validação de transações. Posteriormente, é feita uma explicação mais detalhada do conceito de blocos e transações, sendo apresentados esquemas e exemplos para auxiliar a sua compreensão. Seguidamente, este capítulo possui uma secção dedicada à privacidade e segurança assegurada pelo protocolo, onde é apresentada uma possibilidade de ataque ao protocolo, e como esta é contornada. Para terminar este capítulo, é detalhada, na Tabela 2.1, uma apresentação e comparação das diferentes implementações do protocolo Bitcoin, também chamadas de *wallets* [But12]. É também feita a distinção entre as aplicações de Bitcoin para dispositivos móveis e para computador, descrevendo as respetivas funcionalidades e diferenças. A Tabela 2.2 apresenta também uma análise comparativa das diferentes moedas criptográficas existentes no mercado, nomeadamente *Bitcoin*, *Litecoin*, *Peercoin* e *Cryptoescudo* [Eis, Cry14]. Implementações de cartões de fidelização eletrónicos, que foram alvo de investigação neste trabalho são também referidas, nomeadamente a aplicação móvel do supermercado *Continente* e da cadeia de lojas *Currito* [Nex14, Cur13]. Uma breve discussão sobre as funcionalidades destas aplicações e respetivas limitações é incluída no capítulo.

Tecnologias Usadas e Engenharia de Software

Uma vez concluída a revisão das tecnologias existentes no mercado, foi possível identificar quais as ferramentas e tecnologias necessárias ao desenvolvimento do sistema Bitpoints, desenvolvido durante este mestrado. O Capítulo 3 começa por apresentar as ferramentas e tecnologias utilizadas, nomeadamente o pacote *Android Development Tools* (ADT) que se encontra associado ao *Integrated Development Environment* (IDE) Eclipse. O sistema foi desenvolvido recorrendo à linguagem de programação Java no sistema operativo Ubuntu 13.10. Uma vez que o emulador incluído no ADT é computacionalmente pesado, optou-se por testar o sistema num dispositivo real. Neste ponto o capítulo descreve os passos necessários à configuração do sistema de maneira a permitir o teste e depuração de aplicações em dispositivos físicos [Tho13, Dev]. Adi-

cionalmente, foi identificada a tecnologia *JavaScript Object Notation* (JSON) para armazenar e interagir com os dados do sistema. De maneira a auxiliar o manuseamento e depuração das estruturas de dados JSON foram utilizadas também as plataformas *JSON Lint Validator* e *JSON Editor Online* [CC, dJ]. Posteriormente, representado na Figura 3.1, encontra-se o diagrama onde é ilustrada a arquitetura híbrida (P2P e cliente-servidor) do sistema, suportada pelo protocolo de comunicação *Transmission Control Protocol* (TCP)/*Internet Protocol* (IP). O sistema é composto por uma aplicação do lado do cliente destinada aos utilizadores finais (aplicação móvel Bitpoints), e por uma aplicação do lado do servidor destinada ao controlo e configuração do sistema (servidor Bootstrap).

Este capítulo evolui para a descrição da engenharia de *software*, começando pela análise de requisitos, onde são discutidos os requisitos funcionais e não-funcionais do sistema. Seguidamente, é realizada a apresentação dos diagramas dos casos de uso das aplicações desenvolvidas onde foram identificados os dois atores do sistema (utilizador comum e administrador).

Posteriormente, o capítulo inclui a apresentação e uma breve descrição dos diagramas de atividades do sistema.

Este capítulo termina com a apresentação dos diagramas de classes que representam a estrutura e relação entre as classes implementadas no sistema. Os diagramas de classes para a aplicação do cliente e do servidor podem ser consultados nas Figuras 3.8 e 3.9.

Implementação

Uma das fases mais importantes deste mestrado diz respeito à implementação de um sistema de cartões de fidelização baseado na moeda criptográfica Bitcoin. O Capítulo 4 começa por descrever o funcionamento do protocolo Bitpoints e algumas decisões relevantes tomadas durante a sua implementação.

Durante a fase inicial da implementação do protocolo foi realizado um estudo às mais conhecidas *Application Programming Interfaces* (APIs) P2P existentes no mercado: *JuXTApose* (JXTA), *FreePastry* e *TomP2P* [Ora13, Hoy09, Boc13]. Seguidamente, é feita um discussão sobre as razões que levaram a ignorar estas APIs e implementar uma rede P2P de raíz.

Posteriormente, as Figuras 4.2 e 4.3 apresentam as mensagens entre as entidades da rede durante uma transação e a respetiva confirmação. Estas mensagens são seguidas das respetivas descrições.

Neste ponto é realizada a apresentação da implementação e do funcionamento do servidor Bootstrap. As mensagens referidas anteriormente foram definidas de maneira a auxiliar a explicação da implementação do servidor. O capítulo evolui para a apresentação da aplicação móvel Bit-

points e o seu funcionamento.Estas secções fazem referência às classes e métodos utilizados, descrevendo alguns detalhes de como foram implementadas as funcionalidades de ambas as aplicações.

Para terminar, foi incluída uma secção com o objetivo de discutir questões relacionadas com a privacidade, segurança e eficiência do novo sistema, e qual o impacto causado pela inclusão de uma entidade central.

Teste e Análise do Protótipo

O Capítulo 5 descreve o planeamento e a distribuição de um questionário.Este questionário foi elaborado com o propósito de testar a usabilidade (anonimamente) do protótipo implementado e pode ser consultado no Anexo A.

O questionário compreende quatro tipos diferentes de questões: (i) questões para obter informação sobre o inquirido e o seu conhecimento sobre o tema; (ii) questões para avaliar o dispositivo usado no teste; (iii) questões para avaliar o funcionamento do protótipo; e (iv) questões de resposta pessoal e opcional. Estas últimas possibilitaram ao utilizador sugerir alterações ou criticar a aplicação.

Seguidamente, o capítulo apresenta os resultados obtidos, a sua análise e discussão. Os resultados obtidos neste inquérito são bastante satisfatórios uma vez que, maior parte da população avaliou a aplicação como sendo fácil de utilizar e compreender, e mostrou-se confortável com a experiência.

Para terminar, as sugestões e críticas dos inquiridos são tomadas em consideração, sendo apresentadas e discutidas nas últimas secções deste capítulo.

Conclusões e Trabalho Futuro

As conclusões e trabalho futuro são enumeradas no Capítulo 6.

Uma secção inicial apresenta algumas reflexões acerca do problema apresentado, no que diz respeito ao sistema tradicional de cartões de fidelização. Foi também apresentada a importância da moeda criptográfica neste âmbito e como esta colmatou as limitações dos sistemas existentes.

O capítulo evolui para a discussão das questões de segurança em relação aos sistemas já existentes, com a introdução de uma entidade central.

Posteriormente, foram extraídas e apresentadas as principais conclusões, que dizem respeito à análise dos resultados obtidos do inquérito referido no Capítulo 5.

De maneira a finalizar as principais conclusões desta dissertação, foi realizado um estudo comparativo entre o protocolo Bitcoin, os sistemas de pontos tradicionais e o sistema resultante deste mestrado. A Tabela 6.1 apresenta a relação entre funcionalidades, vantagens e desvantagens entre os três sistemas. Assuntos como segurança, privacidade e eficiência foram também alvos de análise.

Olhando para as linhas de trabalho futuro, propõe-se a implementação de mecanismos que melhorem a segurança do sistema, quer localmente, quer nas comunicações de rede. Mecanismos de *backup* de carteira também seriam aspetos interessantes a considerar. Considerando a capacidade de memória dos dispositivos móveis, acaba por ser importante ter em consideração que esta é limitada e que a *blockchain* cresce com o decorrer do tempo. É importante considerar a implementação de mecanismos que guardem apenas referências para as transações (ao invés de guardar a *blockchain*) e que utilizem as árvores de *Merkle* para fazer as respetivas verificações.

O impacto de possuir uma entidade central deverá ser ivestigado mais aprofundadamente, uma vez que parte do sucesso do Bitcoin recai no facto de ser um sistema descentralizado, e a inclusão de uma entidade central pode levantar novas questões.

Abstract

Cryptographic currencies have been thriving in the last 5 years, specially since the appearance of Bitcoin in 2009. Factors, as the particular advantages of this type of currency, the current economy conjecture and the evolution of technology are fuelling their popularity. In some countries, cryptographic currency systems are considered to be feasible alternatives to real money by the government and Bitcoin is actually being used in transactions worldwide. The success of Bitcoin is mostly due to its elegant mathematical description, proven security under its assumptions, its decentralized character and anonymity assurance. Apart from the initial effort to securely and correctly implement the system and of the maintenance of the applications, Bitcoin works automatically with the contribution of the nodes of a fully decentralized infrastructure. The full specification of the protocol is readily available, e.g., in the Internet, and it can be used by anyone. This masters programme explored the possibility to use it, with modifications, as a means to implement a system for electronic loyalty cards. In order to do so, the aforementioned cryptographic currency was studied in detail, a set of requirements for the new system and modifications to the original protocol were specified, and a software system was engineered and implemented in the Java programming language. The specification of the modifications was performed while taking the particular application scenario into account. The restrictions deriving from the application scenario were mostly dominated by the fact that the underlying Peer-to-Peer (P2P) infrastructure was to be constituted by smartphones only.

The most visible outcome of this masters programme is the fully working prototype of the loyalty card system, comprised by an application for mobile devices and by a server side application. This prototype implements part of the Bitcoin from scratch, starting from the seminal work that defines it, along with the modifications that introduce a central agent for better controlling the quantity of currency per client and aid in the establishment of the P2P connections between two mobile applications. The modified version of the system is herein called Bitpoints, and the currency is instead constituted by points. The implemented loyalty card system benefits of some of the advantages of the popular cryptographic currency, namely the public access to the ledger for isolated verification of all transactions. The loyalty card permits the exchange of points between users and mining new points, which is fundamentally different than currently available loyalty cards.

Within the context of this masters programme, a survey was delivery to a population constituted of 34 individuals, who answered a set of questions concerning the handling of the aforementioned prototype. The analysis of the obtained results allows to induce that people would feel comfortable with this application and accept the concept on which is based on, probably preferring a system similar to the proposed one.

Keywords

Android, Bitcoin, Bitpoints, Electronic Commerce, Electronic Currency, Loyalty Card, Mining, Mobile Application.

Contents

List of Figures

List of Tables

List of Acronyms

ACM Association for Computing Machinery

ADB Android Debug Bridge

ADT Android Development Tools

API Application Programming Interface

ATM Automated Teller Machine

AVD Android Virtual Device

AVDM Android Virtual Device Manager

BTC Bitcoin unit

BTP Bitpoints unit

DHT Distributed Hash Table

ECC Elliptic Curve Cryptography

GUI Graphical User Interface

ID IDentification

IDE Integrated Development Environment

IMEI International Mobile Equipment Identity

IMSI International Mobile Subscriber Identity

IP Internet Protocol

ISO International Organization for Standardization

J2ME Java 2 Micro Edition

JCE Java Cryptography Extension

JDK Java Development Kit

JSON JavaScript Object Notation

JXTA JuXTApose

MAC Media Access Control

MD5 Message Digest algorithm 5

MIPS Million Instructions Per Second

MSP-Cv Multimedia Signal Processing - Covilhã

NIST National Institute of Standards and Technology

OS Operating System

P2P Peer-to-Peer

PC Personal Computer

PHP Hypertext PreProcessor

PIN Personal Identification Number

QR Quick Response

RAM Random Access Memory

RSA Rivest, Shamir and Adleman

SDK Software Development Kit

SHA Secure Hash Algorithm

SHA-256 Secure Hash Algorithm 256 bits

SIM Subscriber Identity Module

TCP Transmission Control Protocol

UBI University of Beira Interior

UI User Interface

USB Universal Serial Bus

UML Unified Modeling Language

URL Uniform Resource Locator

XML eXtensible Markup Language

Abbreviations

Please consider the following abbreviations with the respective meaning when later invoked in the text:

e.g. originated from the Latin expression *exempli gratia* which means ”for example”.

i.e. originated from the Latin expression *id est* which means ”that is” or ”in other words”.

xxx

Chapter 1

Introduction

This dissertation, and the project it refers to, were developed in the context of the 2nd cycle of studies of Computer Science and Engineering of the University of Beira Interior, Covilhã, Portugal. It consists on the implementation of a loyalty card for smartphones by using the techniques and the protocol of the most popular cryptographic coin: Bitcoin. The following section will be focused on the motivation and the scope behind the development of this work. Then, the discussion elaborate on the problem statement and on the proposed objectives. The next-to-last section presents the main contribution of this masters programme and the last section will describe and summarize the structure of this document.

1.1 Motivation and Scope

In the last decades, we have witnessed an increase of the number of users who buy products or services via Internet, fostering a significant growth of electronic commerce (e-commerce). This commerce relies typically on financial institutions, the classical cornerstone for dealing with money in real life, which serve as third parties in the processing of the payment transactions, though that dependence is changing with the emergence of relatively new cryptographic coins. The classical way of dealing with transactions, though it has matured for years, is not perfect and presents some risks or disadvantages. It can be said that it works for most transactions. Nonetheless, this system cannot assure that a transaction is non-reversible, for example, and does not typically support the possibility to perfectly preserve the anonymity of the entities involved in the transactions [Jin13]. Furthermore, according to the British government, more than 2.5 billion people have no access to financial institutions or insurance providers and many countries still lack access to secure banking deposits or even international trade. This occurs in developing countries where there is a lot of corruption and a secure financial infrastructure is non-existent. In these cases, mobile phone services represent a means to access mobile banking for many people [Dar14].

With the previous problems in mind and with the focus on e-commerce, a protocol for creating and supporting virtual coins was proposed in 2008 by an unknown party under the alias *Nakamoto Satoshi*. The Bitcoin protocol [Nak09] is a completely decentralized digital currency, supported by a Peer-to-Peer (P2P) network, that has captivated the interest of users in general, mostly thanks to its interesting features like self-regulation, the possibility for full anonymity, peer verification of all transactions ever made and transactions irreversibility. A lot of brands and

companies are already accepting Bitcoin units (BTCs) as payment for their products or services. Bitcoin is based on a concept known as *proof-of-work*, which consists in proving that a given difficult mathematical problem has been solved, and on the assumption that more than half of the P2P network is dominated by legitimate users (contrasting with malicious ones). Bitcoin makes all information regarding transactions public, but supports full anonymity of the involved entities.

Nowadays, the popularity [Dar14] of Bitcoin has been partially fuelled by the economy crisis affecting many countries of the world. As people began to lose faith in the local economy, some of them decided to bet on a currency that is not governed by any central bank or entity [Jin13]. Nowadays, BTC is a currency aiming at achieving the Euro or Dollar statute, but on the digital domain. It is actually possible to convert this digital currency in Automated Teller Machines (ATMs) [Wag13].

The first implementation of a Bitcoin client was in 2009, in the form of an open-source P2P application. Also called as a *virtual wallet*, this application allowed mining new coins and perform transactions and payments between users. Both concepts (of mining and transactions) will be described with more detail in the next chapter.

Two of the most interesting features of Bitcoin is that there is no central entity performing the management of money, contrarily to the case of the real banks and that it is a fully anonymous system. As such, it is not possible to link users to transfers or tracking them. Because of that, Bitcoin is attractive for, e.g., drug dealers or other illegal businesses. However, to accomplish this without a trusted party, transactions must be publicly announced [Dai98]. There is an history (blockchain or ledger), shared by all nodes of the system, that allows exploring all transactions. These transactions include, amongst additional data, the destination address, the sender address and the value of the transfer. This ledger lists all the transactions ever made since the creation of the currency until the current moment, which makes this system so fascinating, because one knows which addresses are making transactions, but he or she will never know who are the owners of those addresses.

BTC is quite a volatile currency, with its starting price defined by the *BitcoinTalk* users. The first transaction was to trade a pizza by 10000 BTCs [Hos]. At the time of the writing of this dissertation, it has an incredible value (about 800 €), which is constantly varying over time.

This coin was designed with the objective of solving several problems of the real currencies, such as the transaction irreversibility and transactions verification. Based on a proof-of-work, this protocol provides the means to assure the secure exchange of the coins. The digital coin has no physical representation or value if people do not decide to give it one. This is an advantage because it can be anything and it may be referred to as points of a loyalty card, which is actually

the designation of this research work.

Nowadays, a large number of companies use loyalty cards to increase their customers base, track their habits and offer discounts on products, giveaways or accumulation of money that even allows interaction with other companies or services. However, most physical loyalty cards end up representing systems that are not that practical, because there are a lot of different cards and a lot of different ways to promote the brand or the company, and most of these cards are never used or used just a few times. It is interesting to centralize all the physical cards or some of them in only one application, that offers their simplified usage.

The systems of points utilized in most currently available loyalty systems are centrally managed and allow only a very limited set of functionalities. For example, they do not allow customers to exchange points between them. They do not allow for different companies to accept points for a given loyalty card as payment in an ad-hoc mode.

Nowadays, the usage of mobile devices is a common place in developed countries. There is an increasing number of persons owning smartphones with Internet connectivity. This masters programme was focused on developing a mobile application, whose main objective is to replace traditional loyalty cards, already offered by brands or companies, using a virtual points system. The virtual points system is based on the Bitcoin protocol, altered so as to accommodate some specific requirements of the application scenario. As such, the scope of this work falls within the intersection between the areas of mobile computing and software development, electronic commerce, networking and cryptography. Under the 2012 version of the Association for Computing Machinery (ACM) Computing Classification System, a *de facto* standard for computer science, the scope of the masters programme, reflected in this dissertation, is defined by the categories named:

- **Human-centered computing~Ubiquitous and mobile devices**;
- **Computer systems organization~Peer-to-peer architectures**;
- Security and privacy~Cryptography.

1.2 Problem Statement and Objectives

The main problem addressed in this masters programme is, fundamentally, the lack of features of some of the currently available loyalty card systems. The fact that smartphones and mobile devices, with their increasing processing and storage capabilities, are not used as leverage to improve those systems constitutes a problem also. It is our belief that using open-source and readily available technologies, it is possible to build a system that is much richer, in terms of functionalities, than most of the ones utilized nowadays. For example, loyalty systems do not

typically enable a user to send credit to other user, nor vendors without an explicit agreement with the owner of the system to accept payments using their cards. These systems, also do not allow mining rewards by confirming transactions.

The main objective of this project is to implement a robust and secure electronic loyalty card system, specially designed for mobile devices. This system should be fully supported by software, i.e., no physical card should be needed, and it should at least provide the functionalities mentioned above. Even thought the system should enable a centralized management, its operation should be clear to any of its intervenients, as it happens for many virtual coins. Some of the secondary objectives are thus to: (i) study mechanisms and concepts supporting cryptographic coins; (ii) implement a prototype of a system; and (iii), describe the adjustments that need to be done for such a system to work properly with a network of a smartphones only, assess if a central entity is required in this case and if there are guarantees that the system continues to be secure in such environment.

1.3 Adopted Approach for Solving the Problem

The main concept behind the adopted approach was to use Bitcoin, or at least some of its mechanisms and protocol, to implement the loyalty card system. From a general perspective, the adopted approach for achieving the objectives of this programme can be described as follows:

1. The first phase consisted in the contextualization with the objectives of this programme and familiarization with the technologies involved, namely with fully decentralized P2P networks and with Bitcoin. Other loyalty card systems were also studied during this initial phase;

2. The second phase comprised a preliminary set of experiments concerning the implementation and testing of a P2P network with mobile devices;

3. The third phase consisted on designing the new system, via the software engineering and requirements identification;

4. The forth phase was then to implement, test and improve the engineered system. In this phase, an effort was made to address and discuss the security aspects;

5. The fifth phase consisted on obtaining the opinion of users regarding the developed application and loyalty card system via a survey.

The overall organization of the dissertation (refer to section 1.5) reflects the way this research work was conducted.

1.4 Main Contributions

The main contribution of this masters programme is the delivery of a functional prototype of an electronic loyalty card based entirely on the Bitcoin protocol. The system is composed by an application for Android smartphones and another one for a server, both built in Java programming language. The implementation is based entirely on the Bitcoin description and specifications, notably from [Nak09], and not from other implementations available in the Internet. The particular application scenario for the developed system is not as critical nor embodies the same requirements as the analogous electronic currency (also known as e-currency) scenario and, as such, the implemented system also needed to reflect that difference. One of the major differences is that the system does not need to be completely decentralized and that a central entity may be tolerated. This central entity was implemented as a rendezvous host, herein also called Bootstrap server. This dissertation discusses the responsibilities and trade-offs associated with this bootstrap server, presenting also a justification for why it was utilized.

As a secondary contribution, it is possible to mention that the system was designed specifically for smartphones from the start, taking some of the issues related with building mobile applications with potentially heavy computations into account. The dissertation also contains a detailed software engineering of the smartphone and server applications.

The prototype developed for an electronic loyalty card advances the state of the art in this specific area. For example, it adds features that implementations based on physical cards do not have, namely the possibility to exchange points with other peers. The security is assured by the concept and mechanisms behind Bitcoin.

1.5 Dissertation Organization

This dissertation is organized in six chapters. Its main body is comprised of four chapters, preceded and succeeded by the *Introduction* and *Conclusions and Future Work* chapters, respectively. The contents of each one of the chapters can be summarized as follows:

- Chapter 1 - **Introduction** - presents the context of the dissertation and the motivation behind the described work. It also includes the objectives, adopted approach, main contributions for the advance of knowledge and the dissertation overview.

- Chapter 2 - **Related Work** - starts with a brief introduction to the main concepts, mechanisms and technologies related with this project, and ends up with the presentation of some relevant works concerning virtual loyalty cards and Bitcoin clients.

- Chapter 3 - **Technology and Software Engineering** - elaborates on the software engineering of the mobile and server side application, via the identification of use cases, discussion

of activity and classes diagrams, and by defining the architecture of the system. It also describes the technologies and tools that were used in the respective development.

- Chapter 4 - **Implementation** - describes the implementation of the engineered system, as well as some of the most important choices that define it. The chapter evolves to the presentation of the developed prototype, and it ends up with a small discussion regarding privacy, security and performance.

- Chapter 5 - **Testing and Analysing the Prototype** - is focused on the planning and delivering of an usability survey. An analysis of the results of that survey is also performed towards the end of that chapter.

- Chapter 6 - **Conclusions and Future Work** - wraps up the most important conclusions of this masters programme and discusses potential future lines of work.

The **Survey of Usability Test** (Appendix A) was also included as annexe, for the sake of consistency. The annexe presents the survey delivered to the respondents along this masters programme, to assess the usability of the developed prototype.

Chapter 2

Related Work

2.1 Introduction

At the time of writing this dissertation, crypto-currencies were still gaining momentum in terms of popularity, being used in countless situations and by a large number of users. This chapter addresses the actual state of crypto-currencies, namely the Bitcoin protocol, as well as its functioning under the hood. The related work on virtual loyalty cards and Bitcoin clients are included in here as well.

The next section (Section 2.2) describes the related work on loyalty cards, focused mainly on the ones supported by physical cards. The third section (Section 2.3) will be devoted to the description of the functioning and specifications of the Bitcoin protocol and finally, Section 2.4, is where some existing Bitcoin and Electronic loyalty card implementations will be mentioned and explained.

2.2 Loyalty Cards

A loyalty card, of which some examples are represented in Figure 2.1, also known as a customer card, is a plastic or paper card that serves as a physical support for companies or brands to track and captivate their customers with bonuses like discounts, prizes, special prices or points that can be exchanged for money, services or other goods.

Figure 2.1: Examples of loyalty cards.

On the current context, loyalty is the act of making clients faithful to a brand, product or service. When shopping every day at the same store, means that the customer is loyal to the products of a company, and that he or she is satisfied with the quality and price of the services. The advantages of loyalty cards are that the client can use them to collect points or money that allows him or her to have special benefits, like the previously referred ones. These benefits may take several forms, namely products and services from the company issuing the loyalty card, or

from other companies in which the later may have an agreement. For example a supermarket chain may allow for the points of its loyalty cards to be exchange by gas from a specific loyal company.

Even thought there is an excess of existing loyalty cards on the market these days, according to a study conducted jointly by Princeton, Total Research Corp. and Carlson Marketing Group, it was noted that the spending of the customers increases by 46% when they perform their shopping at companies that have loyalty based rewarding systems [Sev03]. According to the same survey, it is more expensive to acquire a new customer than to keep an existing one. Approximately 30% of the company customers are actually not interested in these type of systems, the others (aproximately 70%) do not mind having a wallet full of cards, if it implies a lower investment when purchasing products or services. Nonetheless, the aforementioned reasons are not the only ones justifying the importance of loyalty cards to companies. It is known that they are used to collect customer information and to, therefore, support marketing and decision taking processes. Such information may be used by the company to generate vouchers to offer to the customer, as a way to possibly encourage him to purchase even more products. A series of strategies can then be used to further captivate the customer, namely via lottery type games or directed promotions.

Physical cards are practical only in the sense that the customer does not have to worry to much about its management. In order to use it, customer only has to typically show it in specific moments, namely when paying. They do not typically offer any kind of additional interaction, as for example, consulting the balance any time or exchanging credit with other customers.

2.3 Bitcoin and Crypto-currencies

A common user of Bitcoin, a non official International Organization for Standardization (ISO) crypto-currency [RB14], does not need to have full knowledge of the operation of the protocol. Nonetheless and since it useful for the explanation contained in this dissertation, this section will cover the essential topics of the system. Some concepts and mechanisms concerning the operation of the protocol are going to be discussed first (subsection 2.3.1) to then evolve to the description of the blockchain (subsection 2.3.2). The mechanism supporting transactions (subsection 2.3.3), and mining (subsection 2.3.5) are the subject of subsequent sections, along with the definition of blocks (subsection 2.3.4) and proof-of-work (subsection 2.3.6). A brief description on the security and privacy of Bitcoin is included in subsection 2.3.7.

2.3.1 Concepts Related to Bitcoin

Before proceeding with the explanation of the protocol, there are some concepts that need to be defined. Bitcoin makes extensive use of *hash* values and functions. In order to assure the integrity, authenticity and non repudiation of *transactions*, it also uses *digital signatures*.

Currently available implementations of the protocol use Elliptic Curve Cryptography (ECC) [PP10] as basis for the digital signature scheme. On the other hand, *merkle trees* are used as means to compress block information and enable easier revalidation of transactions. Bitcoin users are typically identified by the so called *addresses*. The concepts emphasized with italic typeface can be explained as follows:

Hashes - Cryptographic hash values, like the ones used in Bitcoin, also known as checksums or hash codes, are values obtained via the application of a cryptographic hash function like Message Digest algorithm 5 (MD5) or Secure Hash Algorithm (SHA). Cryptographic hash functions are efficient one-way functions capable of mapping an arbitrary long stream of bits into a fixed length code [Vau10]. Bitcoin uses the Secure Hash Algorithm 256 bits (SHA-256), which outputs codes with 256 bits. These hash functions are used in the digital signature scheme, for constructing the Merkle trees and for the proof-of-work. Since it is still considered to be a safe function nowadays [oST14], SHA-256 was also used in the implementation of the loyalty card system developed along this work.

Merkle trees - A Merkle or hash tree [Mer80] is a tree in which every node is labelled with the hash of the labels of its children nodes. Merkle trees are commonly used to make verification of the integrity of large data structures more efficient, while preserving security. These trees are used essentially on the so-called *light version* of the Bitcoin protocol, to save disk space, which in mobile devices constitutes a constraint. Figure 2.2 depicts a Merkle tree.

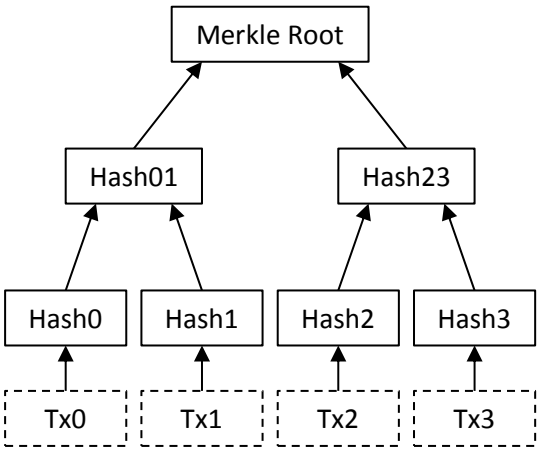


Figure 2.2: An example of how Merkle trees are used to compress information regarding transactions.

The leaf nodes of the depicted tree represent transactions (or the data conveying that information). The tree is constructed by initially calculating the hash values of those transactions separately, and then successively compute apply the hash function to the combination of each two hash values in order. For example, from `Tx1` and `Tx2`, one obtains `Hash0` and `Hash1`, respectively. `Hash01` is the hash value of the concatenation of the

latter two. The root is a value that ultimately summarizes the entire dataset comprising the leafs. The advantage of using a Merkle tree to summarize data is that it enable the efficient verification of a given subset of that data, namely by following only the branches associated with it. E.g., verifying the integrity of `Tx1` would require only the access to the root, `Hash01`, `Hash23`, `Hash0` and `Tx1`.

Address - Within the crypto-currencies context, the address of a wallet is the public key of a pair of asymmetric keys of a public key crypto-system. In many crypto-currencies, it is a 256 bit long value written in hexadecimal and representing the public key of an ECC key pair. It can be freely distributed and it is necessary to perform transactions. Thought their meaning, in terms of usefulness is similar to the one of email addresses, they cannot typically be validated because they are entirely random and, as such, have no structure. It is estimated that planet Earth contains about 7.5×10^{18} grains of sand and supposing that each grain was another planet Earth, in the end, the total number of sand grains would still be less than the total number of possible addresses [Dri13].

Block and Transaction - In a nutshell and within this context, a block is a data structure comprising transactions, and a transaction is data structure containing information representing the movement of a number of virtual coins from a given entity to another. The entities are typically represented by their public addresses in this data structure, which also normally contains a digital signature of the sending entity. More on this subject below.

Signatures and transactions verification - A digital signature is a fixed size value (e.g., 256 bit long), calculated over the data it applies to using a private key, that can be used several properties regarding that data and its originating party, namely the data was not tampered with and that it comes from the owner of the respective public key. It also comprises a means to assure that the signing party cannot later argue that it did not sign or processed that data. In many crypto-currencies, the verification of a transaction occurs at the recipient side to check if a transaction is valid or not.

2.3.2 The Blockchain

The blockchain, ilustrated in Figure 2.3, consists basically of a public file (also known as *ledger*), where all transactions since the beginning of the crypto-currency are stored. This file is sequentially structured, and blocks are added as transactions are made. Pending transactions are included in a new block, which will have a reference to the previous block and which, in turn, after confirmed, will be included in this ledger. Once all transactions are registered in this ledger, a wallet client, or any other application devised for that specific purpose, can know the exact amount of coins of a given address, thus allowing its owner to perform transactions with those coins. All transactions all thus completely public, though that is no way to associate a public address with a real world identity. Each node connected to the network has a copy of

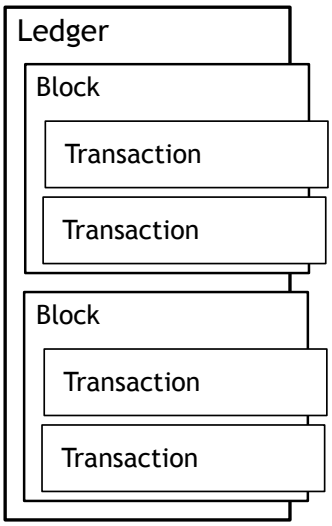


Figure 2.3: A scheme of the blockchain.

this file which, due to the functioning of the protocol, grows over time. The longest valid chain is always the one considered by the clients.

2.3.3 Transactions

A transaction is a structure with several fields representing the exchange of e-currency units between public addresses, digitally signed with the private key of the source. Transactions are broadcasted to every node connected to the network and inserted into blocks. Once validated, these blocks are then inserted into the ledger.

According to Satoshi [Nak09], an e-currency is constituted by a chain of digital signatures. As seen in Figure 2.4, a given client transfers coins to another one by signing the hash of the previous transactions and the public key of the next owner, along with a number representing the transferred amount. The receiver can immediately verify if the digital signature is valid or not, though he or she cannot be certain that the money was transferred. That certainty is only obtained after the network validated the block in which the transaction is included.

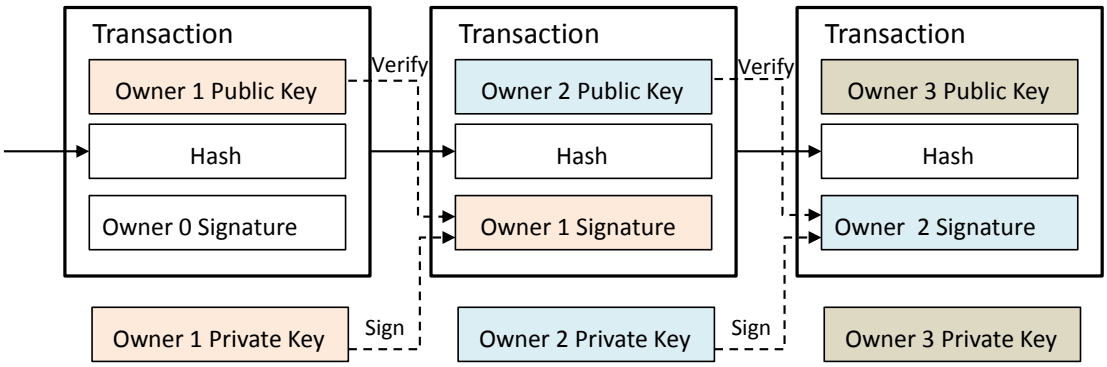


Figure 2.4: Representation of the data structure of transactions and of the chain of digital signatures.

In the Bitcoin protocol, a transaction is used to send BTCs to another entity and includes, along with other parameters, two fields:

Inputs - An input is a reference to the previous transactions that gave money to the user, allowing him or her to make other transactions;

Outputs - An output is a reference to a specific destination address and its specific value.

It is common to represent transactions of crypto-currencies using JSON. To provide its explanation with a concrete example, the next figure contains the JSON representation of a real transaction of the Bitpoints protocol, used by the prototype developed along this masters programme (some of the values were abbreviated for the sake of clarity).


```
{
"hash":"dcc43829dfda54a7d916c38d0f705b57d4e42c140cf0dc9df9fdf198b9d3658f…",
"pub_key":"3059301306072a8648ce3d020106082a8648ce3d03010703420004955b980…",
"signature":"3045022100c9e53c36873de91dbeee13a649dc02b7633234c07f7e36c18…",
"inCounter":3,
"outCounter":2,
"outputsList":[
        {"address":"3059301306072a8648ce3d020106082a8648ce3d0301070342000…",
         "value":500},
        {"address":"3059301306072a8648ce3d020106082a8648ce3d030107034200…",
         "value":3}
],
"inputsList":[
        {"prevTx":"f8acde092b13e790313feea347b45e7f14109ee3684324bc99f482…",
         "index":0},
        {"prevTx":"1d712ae463a54ac30ee3a5704fac836fad36eaf963bed172a596c1…",
         "index":0},
        {"prevTx":"69181e29078890c4fdbab23d26552503f07b7984bf3a479ba02dba…",
         "index":0}],
}
```


Figure 2.5: A transaction JSON example.

The several fields shown in the figure can be summarized as follows:

hash - The hash value of the current transaction;

pub_key - The public key of the sender of the transaction;

signature - The digital signature computed with the sender private key;

inCounter - The number of inputs included in the transaction;

outCounter - The number of outputs included in the transaction;

outputsList - List of the outputs included in the transaction;

address - The address of the recipient of the transaction;

value - The amount of Bitpoints units (BTPs) to send to the recipient;

inputsList - List of the inputs included in the transaction;

prevTx - Reference to all previous transactions, which allow the transfer of the amount of BTPs specified on this transaction;

index - The position where the previous transaction output is (a transaction can have multiple outputs depending on how much recipients it has).

As can be seen, the structure utilized to represent BTPs transactions is very similar to the one of the original Bitcoin protocol.

Transactions over the Bitcoin protocol follow a fairly simple operation mode: the value to send is first combined with previous transactions (called inputs) to make up the total amount of BTCs, and then the total amount is split by all receivers (called outputs). If the combined value is greater than the value that needs to be send, the excess is send back to its owner (automatically) in one of the outputs of the transaction. Figure 2.6 illustrates two possible scenarios for transactions. The situation of a single output is represented on the left side, while multiple outputs are represented on the right side. In both cases, part of the amount is being returned to the sender.

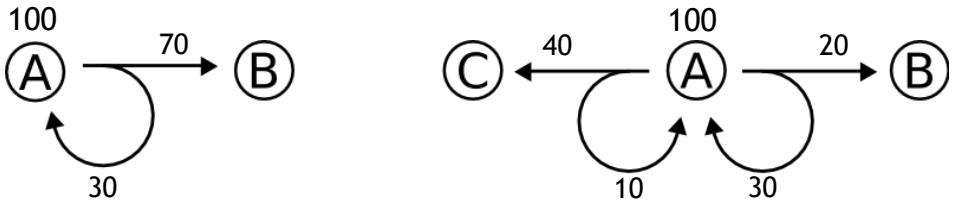


Figure 2.6: Possible scenarios for a transaction.

Figure 2.7 shows a chain of transactions with values for exemplification purposes. Assuming that the owner of address A wants to make a transaction with 100 BTCs (Tx 2) to address B and he has 100 BTCs coming from another previous transaction (say Tx 1), then the scenario is the simplest one, since the values of the input and output are equal.
Now assume that the owner of address B wants to make a transaction of 175 BTCs to address C (Tx 5). In this case, address B has to combine more than one incoming transactions (one from

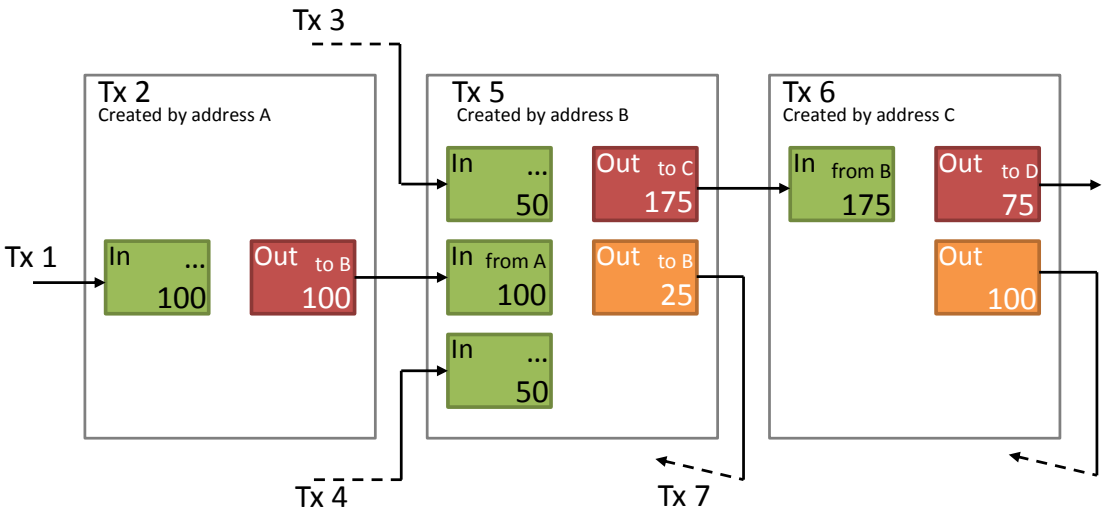


Figure 2.7: An example of a chain of transactions.

address A and two others), because the value of a single input is not sufficient to sum up to the required amount. Therefore, the total value that needs to be send will result from the combination of all previous transactions, until the sum is achieved. Nonetheless, as depicted in the figure, the combination of all transactions gives a total of 200 BTCs and, as address B just wants to send 175 BTCs, the remaining 25 BTCs are sent back to address B.

2.3.4 Blocks

A block is a structure constructed by the nodes connected to the Bitcoin network. All connected nodes are listening and waiting for new incoming transactions, because processing them gives is rewarding (rewarding process will be described afterwards). Each node collects the incoming transactions and builds a block with them. An approximation of the resulting structure is schematized in Figure 2.8. As depicted in the figure, the block will contain, along with other

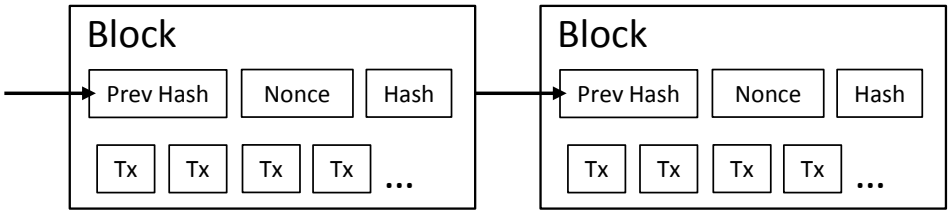


Figure 2.8: Diagram representing a chain of blocks.

fields, a reference to the previous block, the list of collected transactions, the hash of the block and a nonce, that will be used on the mining process (explained in the next section).

Figure 2.9 provides a real example of a block in JSON, with summarized fields and suppressed transactions.

```
{
"version":1,
"prev_block":"0057b4635a9eb412fa333f47817b89acf2cceb0f009af63974b3d3de4b42347d"
"hash":"003632e86e799c0819495d2f7c0da48ffbfe2c56fab9f10bf2694502525f38ff",
"nonce":624497333,
"timestamp":"1400169065",
"tx_num":2,
"tx":[suppressed txs],
"merkle_root":"200f646e3b3cb61f567256887ea844ee44253950562f2a9899c69e36fb547a61",
"merkle_tree":"dcc43829dfda54a7d916c38d0f705b57d4e42c140cf0dc9df9fdf198b9d3658f
              ,2752aba1f7a1733c33a3cfa6f784f188f278ff1971fe21acc03dcbc3bbc6ff4c
              ,200f646e3b3cb61f567256887ea844ee44253950562f2a9899c69e36fb547a61",
}
```


Figure 2.9: Example of a block written in JSON.

The several fields included in the representation can be briefly described as follows:

version - Version of the protocol used to calculate this block;

prev_block - Hash value of the previous block (it is a reference to the previous block, effectively creating the chain);

hash - Hash value of the current block;

nonce - Random value to compute the proof-of-work;

timestamp - Date of creation of the block;

tx_num - Number of transactions included in this block;

tx - List of all transactions included in this block;

merkle_root - Root hash of the Merkle tree;

merkle_tree - List containing all Merkle tree hashes.

2.3.5 Mining

Mining is the process of adding blocks to the public blockchain and receive a reward for contributing for that process. A block can only be added when all the transactions are confirmed using a concept called proof-of-work, which is a process that consumes time and resources. A proof-of-work effectively shows that a specific node spent time processing that block.

2.3.6 Proof-of-work and Validating Transactions

As mentioned before, the proof-of-work is a mathematical puzzle that takes time and computer processing resources to solve. In the case of Bitcoin, this puzzle consists in finding a number, called nonce which, after combined with a block should result in a hash value with specific properties. The proof-of-work starts with a specific node collecting some transactions that are disseminated in the network, and placing them inside a block created by itself. This block will have a reference for the last block of the chain (`prev_block`) and a few other fields. After that, a nonce is generated and the node tries to calculate the hash value of the block with the generated nonce. The calculated hash must have a certain number of bits equal to zero on the beginning, as can be seen on the following example:

`"hash":"00000000d1145790a8694403d4063f323d499e655c83426834d4ce2f8dd4abd32".`

If the hash does not fulfil the aforementioned condition, the nonce is incremented. In order to clarify this concept, assume that the proof-of-work will be performed on the string `Hello, world!` and the difficulty is set to find 12 bits equal to zero in the beginning of the hash. The client generates a nonce, that in this case will start in 0 to simplify the explanation, and concatenates that value to the string as follows: `Hello, world!0`. The next step is to calculate the hash value and verify if it starts with 12 bits equals to zero:

`Hash(Hello, world!0)=`
`1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64.`

Since the output does not have the required number of bits equal to zero, it will generate another nonce and repeat the process. It will do this until it finds the correct hash value, i.e.:

`Hash(Hello, world!1)=`
`e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8,`
`Hash(Hello, world!2)=`
`ae37343a357a8297591625e7134cbea22f5928be8ca2a32aa475cf05fd4266b7,`
`...`
`Hash(Hello, world!4248)=`
`6e110d98b388e77e9c6f042ac6b497cec46660deef75a55ebc7cfdf65cc0b965,`
`Hash(Hello, world!4249)=`
`c004190b822f1669cac8dc37e761cb73652e7832fb814565702245cf26ebb9e6,`
and
`Hash(Hello, world!4250)=`
`0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9.`

This proof-of work was computed 4251 times prior to the discovery of the correct nonce value (4250). The hash function produces an unpredictable output, and it is not possible to know beforehand the value of the nonce that will fulfil the condition. This mathematical property assures that some resources are spent in solving this challenge. The value of BTC derives also from this resources consumption. After the nonce is found, the block is ready to be inserted into

the blockchain, and it is broadcasted to the network. If the new block is received by someone that is still working on some of the already confirmed transactions, these are discarded from the current proof-of-work and this node keeps working on the non confirmed transactions.

2.3.7 Privacy and Security Provided by the Protocol

The concept of privacy is limited in the traditional banking model, because there is a trusted third party involved in the transactions process which may have access to all information of a given user by request. On the other hand, the Bitcoin protocol keeps all the information of the transactions public. Nonetheless, it does not provide the association between public keys and their respective owners. Everyone can see that some address is sending or receiving a certain amount of BTCs, but no one can link these addresses to a real life entity. To make it even more difficult to link the addresses to their owners, the protocol recommends that a client must generate a new address for each transaction.

Notice that is possible that two blocks are verified at the same time or just a few seconds apart by different nodes in the network. When that happens, the nodes who receive the two blocks will keep both in memory and they will work on the block which was first received by default. This may generate two branches of the blockchain, as shown in Figure 2.10. Eventually (meaning with an increasingly high probability), one branch will get longer than the other one and the shorter will be later on discarded (gray branch in figure). The protocol is designed to avoid one

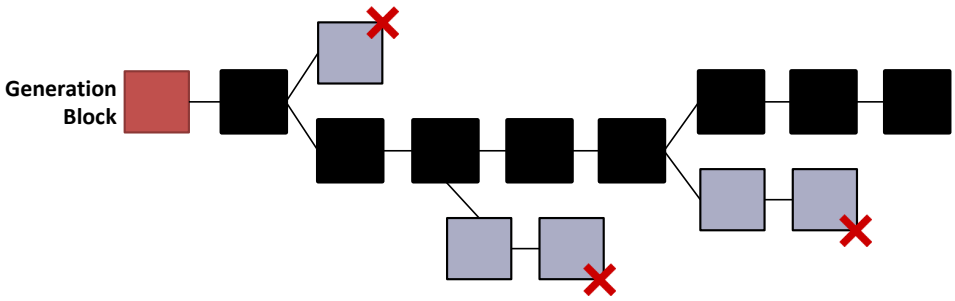


Figure 2.10: Ilustration of the branching artifact of the blockchain.

of the major threats to e-currency known as double-spending. Double-spending occurs when a user buys some service or product and, within a short period of time (for example in the first subsequent minutes), he or she tries to make another transaction with the same virtual money. In this situation, and depending on multiple factors (for example communication speed between different peers), it may happen that the second transaction arrives first to the network, and it is confirmed first. If this situation happens, the first transaction is considered invalid in the Bitcoin. Nonetheless, if the first seller already gave the product or service to the user, double-spending was successful. Bitcoin purposes the seller to wait for the confirmation of the transactions, before handing the product out. Actually, depending on the value of the transaction, it is suggested that the seller should wait for the block containing the transaction

to be buried deep in the blockchain (e.g., after six blocks are confirmed). In the real world, it is recommended to wait approximately 60 minutes before considering a given transaction to be valid (each Bitcoin block takes approximately 10 minutes to be verified). This waiting period will guarantee that even the double branches, created by malicious users, are discarded. The deeper the deeper the block where the transaction is, the less likely it is to be discarded.

2.4 Implementations of Bitcoin and Electronic Loyalty Cards

Nowadays, traditional loyalty card systems begin to lose ground for virtual loyalty card systems. With the global adoption and proliferation of mobile devices, and the ease of access to computer networks, namely the Internet, it became obvious that it would be easier to tackle customer loyalty through these devices. These technological advances enable changing from plastic cards to digital media, cutting expenses from the companies side. It is also simpler to maintain the system, make promotions, collect user information and direct advertisements. For the user that is also simpler to interact with the system. It is possible to find applications designed to be loyalty cards and some others that aim at digitalizing and managing physical loyalty cards. In Portugal, for example the supermarket chain *Continente* offers a free application for the i Operating System (OS) and Android platforms. Figure 2.11 contains some screenshots of the *Continente* virtual loyalty card application. The depicted application consists of a small virtual

(a) Continente virtual card. (b) Continente coupons. (c) Continente coupons.

Figure 2.11: Screenshots of the virtual loyalty card application available for the Continente supermarket chain.

loyalty card that relies on an Internet connection to provide functionalities to the user. Amongst other options, this application allows the user, to check his balance and the amount credited due to purchases. It also enables the user to utilize the virtual card in the store or consult or

edit the shopping list. The users of this mobile application also have access to a list of coupons with possible discounts or promotions. One of the most important features is handed out by the leaflets tab, which shows relevant information for products with discount. The application does not enable a user to, for example, transfer money to other user.

Currently, there are a few companies dedicated to the implementation of virtual loyalty cards, as is the case of VirtualNext [Nex14]. VirtualNext has a unique approach to mobile loyalty and payment systems, without the hassle of complicated applications. This system offers a so called place holder of loyalty, electronic payment, gift cards and coupons to stores and companies, with the advantage that VirtualNext uses the same (simple) template for all of its clients. It is an easy to use system to buy and sell products or services. An interested client orders the application, providing the required information to VirtualNext, like the company name, logo and rewards for consuming, and the application is created almost automatically. An example of a store that uses the aforementioned system is Currito [Cur13], depicted in Figure 2.12. The

Figure 2.12: Screenshots of the VirtualNext virtual loyalty card application generated to the Currito store.

Currito is a store that sells *burritos*, uses this system, and due to its simplicity, it is a successful virtual loyalty card. This application just rewards the user for eating and also allows him to buy *burritos* with his phone.

Shifting the focus to the Bitcoin protocol, it should also be mentioned that there are a lot of different implementations of the so called wallets [But12]. These applications are simultaneously clients and servers of the supporting P2P network. Some of them are only clients and do not perform mining. At the time of writing of this dissertation, there were more than 10 different Bitcoin wallets on the market, each one with own specificities, allowing the user to simply choose the one that best fits his or hers profile. Table 2.1 compares some of those wal-

lets. Originally, when the Bitcoin client appeared, it was exclusively designed to run on desktop

Table 2.1: Differences between several Bitcoin wallets.

Client	Audience	Wallet Security	Network Security	Backups	Setup Time	Disk Space	Multi-user
Armory	Power users	Encrypted, on-device	Addon	One-time	Varies	6+ GB	Multi-wallet
Bitcoin Wallet	End-users	Isolated, on-device	Partial	Manual	Instant	15 MB	on JB tablets
Bitcoin-Qt	End-users	Encrypted, on-device	Full	Manual	Hours	6+ GB	No
bitcoind	Programmers	Encrypted, on-device	Full	Manual	Hours	6+ GB	Virtual accounts
Electrum	Power users	Encrypted, on-device	Minimal	Memorized	Minutes	5 MB	No

computers (Bitcoin-Qt). Such implementation had some limitations, particularly in terms of mobility and convenience. However, at the current time, this is considered the most complete and secure Bitcoin client implemented to this date [Lan13]. The security of the mobile applications is questionable and Bitcoin wallets are targets for attacks. As such, it is a risk to run the mobile clients (Bitcoin Wallet), since mobile phones or tablets contain a lot of personal data and vulnerabilities.

The implementation of the Bitcoin client for PC is presented below, resorting to some screenshots. The overview of the wallet is depicted in Figure 2.13. From this screen, the user can access the wallet general information like the balance, unconfirmed balance and recent transactions performed.

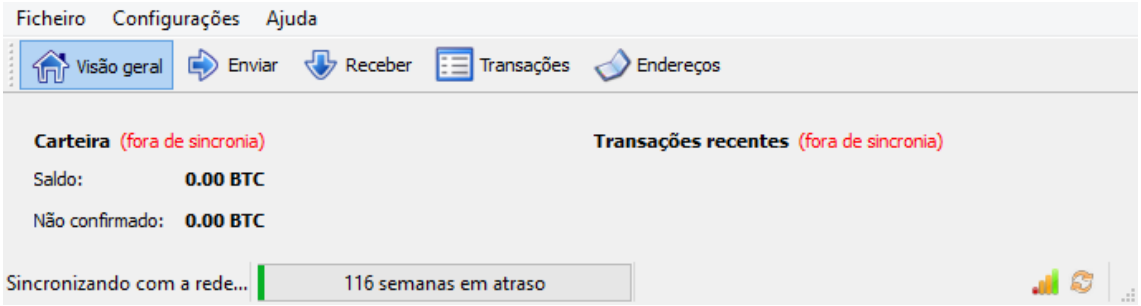


Figure 2.13: Screenshot of the home screen of the Bitcoin-Qt PC wallet.

Figure 2.14 shows the tab for performing a transaction in the Bitcoin-Qt wallet. This tab allows the user to send, to a specific address, a certain amount of coins. The destination address can be introduced directly or by selecting it in the addresses book, which was previously populated and stored.

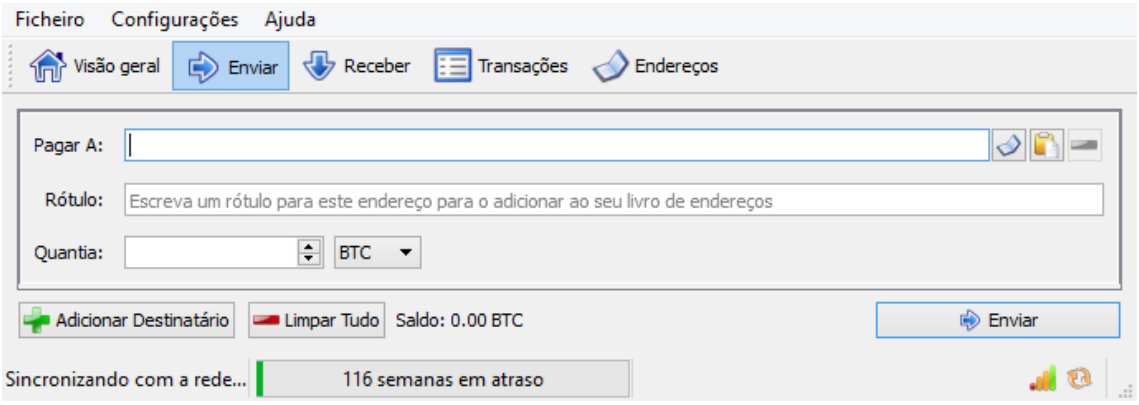


Figure 2.14: Screenshot of the tab for performing transactions in the Bitcoin-Qt PC wallet.

Finally, Figure 2.15 shows the tab where the user has the possibility to generate new personal addresses (i.e., a new key pair). As previously mentioned, it is recommended that the user generates a new address per transaction. The client has some additional features, as the case of digitally signing messages, export addresses or even generate Quick Response (QR) Codes for specific personal addresses. A few other options, like wallet configurations, are included as well.

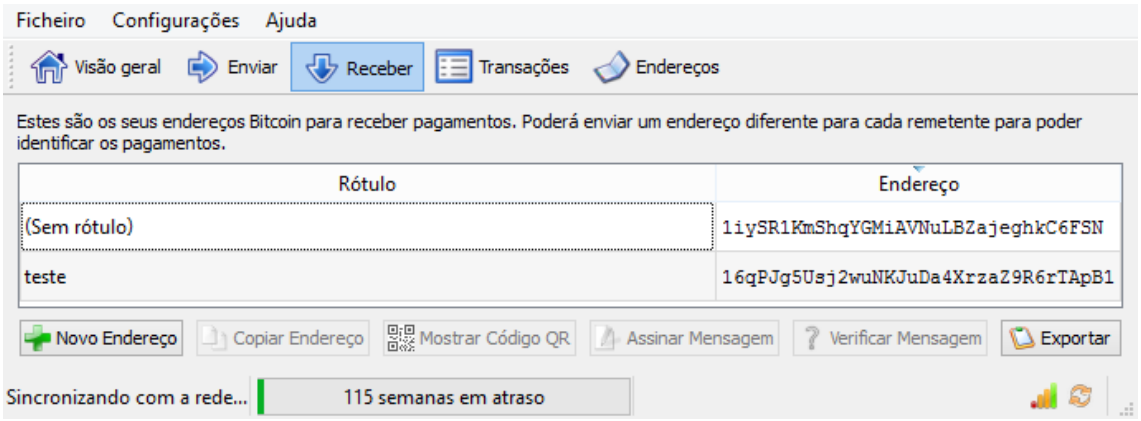
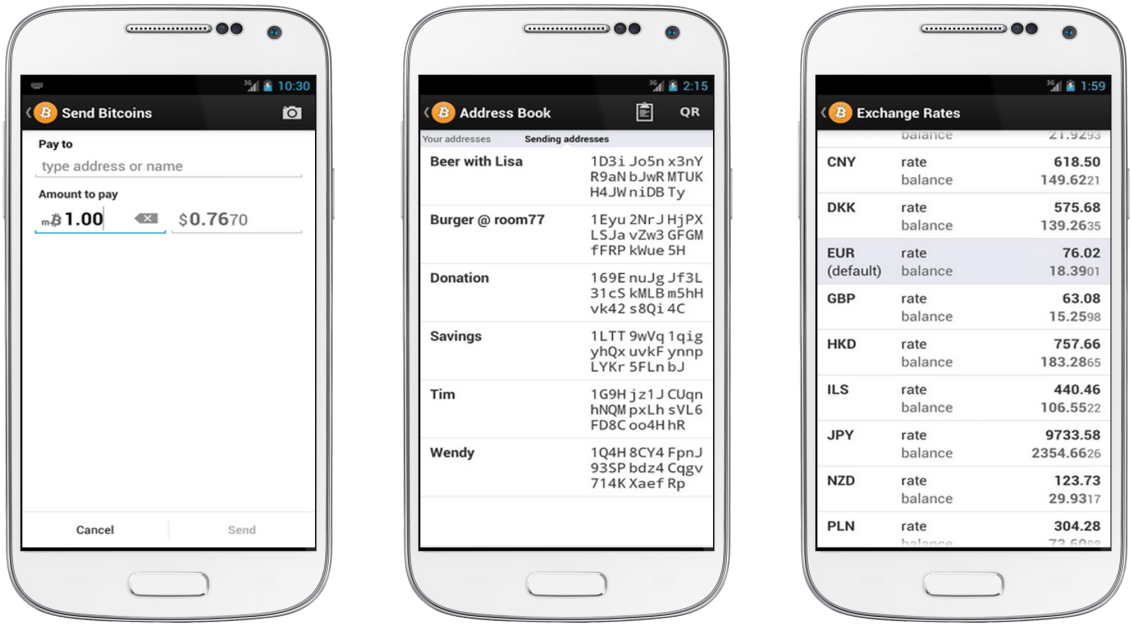


Figure 2.15: Screenshot of the tab concerning personal addresses of the Bitcoin-Qt PC wallet.

For providing this explanation with an example of a client for mobile devices, the wallet for the Android platform is presented in Figure 2.16 as well. Implementations for smartphones have emerged more recently, with the objective of leveraging mobility to Bitcoin users. The try to provide the same essential features of the original client.

In Figure 2.16, the application provides, among other features, the possibility of sending BTCs, manage personal and public addresses and a feature not included in the original client, a list with the exchange rate.

Table 2.2 mentions four different and currently available crypto-currencies, all of them based on the Bitcoin protocol. At the time of writing of this dissertation, there were approximately



(a) Bitcoin transaction tab. (b) Bitcoin address book tab. (c) Bitcoin exchange rate tab.

Figure 2.16: Screenshots of one of the Bitcoin wallets available for Android.

50 different crypto-currencies in the world. The number of alternative cypto-currencies keeps raising fast and their purposes are varied [Eis]. Some of them were implemented by governments to fuel the economy of the respective country, while others are just fictitious coins. Due to the large number of available crypto-currencies, most of the existing coins are not shown in the table, which draws a comparison between the most popular coins and the CryptoEscudo [Cry14]. CryptoEscudo is a Portuguese crypto-currency, similar to Bitcoin, that, according to its creator, was unofficially implemented to help Portugal overcome its debt. It comes with a set of pre-mined coins.

Table 2.2: Several existing crypto-currencies.

Crypto-currency	Code	Year	Website	Coins Already Released	Hashing Algorithm
Bitcoin	BTC	2009	bitcoin.org	60.14%	SHA-256
Litecoin	LTC	2011	litecoin.org	32.58%	scrypt
Peercoin	PPC	2012	peercoin.net	N/A	SHA-256
Cryptoescudo	CESC	2014	cryptoescudo.org	N/A	scrypt

2.5 Conclusions

The current chapter discussed some technologies related with the subject of this masters, namely crypto-currencies, the Bitcoin protocol and existing virtual and physical loyalty card systems. Some Bitcoin clients were also subject of discussion, mostly to emphasize some of their best features or problems. The Bitcoin protocol functioning and specifications in which the Bitpoints protocol will be based and engineered, are included here as well. At the time of writing of this dissertation, crypto-currencies were gaining popularity, particularly within the

mobile devices ecosystem, where convenience and the latest technology are combined.

It was noticed that there are a lot of different Bitcoin wallet implementations and of different crypto-currencies available nowadays. A user may choose from more than 10 mobile or desktop applications for a single crypto-currency. Different crypto-currencies were designed and deployed for different purposes around the globe, namely to motivate the economy growth of some countries. The system proposed in the scope of this dissertation constitutes another approach to this subject to the specific environment associated with the fidelization of users.

The existing virtual loyalty cards are, typically, digitalized representations of physical cards based systems, to which some features and services are added. This work will elaborate on some of the functionalities that these cards present, but with the purpose of adding non existing functionalities, namely the support for users to exchange coins. The following chapter will describe the planning phase and software engineering of the mobile and server side applications implementing the proposed electronic virtual loyalty card system.

Chapter 3

Technology and Software Engineering

3.1 Introduction

This chapter is focused on the software engineering of the Bitpoints system, developed along this masters programme, and it addresses the requirements analysis undertaken before the development phase. Prior to that, section 3.2 describes the tools and technologies used in the development of the project, along with a brief explanation. Section 3.3 is focused in the requirement analysis, where functional and non-functional requirements are described. Section 3.4 discusses the identified use cases. Sections 3.5 and 3.6 present the activity and class diagrams elaborated for the software system and, finally, section 3.7 presents a few conclusions on the topic of this chapter.

3.2 Tools and Technologies Used

Android is an OS for modern mobile devices, developed by Google and the Open Handset Alliance [And]. It is built upon a modified Linux kernel and it is available under an open source license. Android applications are mostly written in Java, however, unlike the Java 2 Micro Edition (J2ME), where Java applications are less privileged than native applications, Android Java applications comprise the entire application layer landscape.

The Android Development Tools (ADT) bundle, a plugin for the Eclipse Integrated Development Environment (IDE) to implement and build Android applications, was used for the development of the Bitpoints system. Bitpoints has been implemented in Java programming language in the Ubuntu 13.10. Eclipse is a free and open source software, mostly implemented in Java, that can be used to develop general purpose applications. With a few plugins, Eclipse may be used to develop applications in programming languages like C, C++, JavaScript, Hypertext PreProcessor (PHP), Python, etc. The Eclipse Software Development Kit (SDK), which includes the Java development tools like Java Development Kit (JDK), is mostly meant for Java developers. Furthermore, users can write their own plugins. The ADT bundle includes an Android Virtual Device Manager (AVDM), that provides a Graphical User Interface (GUI) in which a user can create and manage Android Virtual Devices (AVDs). An AVD is an Android emulator configuration that allows the user to define software and hardware options to be emulated in a virtual environment. However, the Android emulator is typically slow, which hinders the debugging of an application. As such it was decided to test the system on a real mobile device, since early stages. To make

this possible, the real device was configured in the debugging mode and some rules were added to the system, mostly because it was not readily appearing in the device list provided by Android Debug Bridge (ADB).

After installing the SDK, it was necessary to set up the system to detect the physical device. The first step of this configuration was to install an Application Programming Interface (API) specific to the real device. This was achieved by launching the Android SDK Manager on the Eclipse GUI or running the following command in the terminal: `./android` [Tho13]. In this case (Linux environment), it was required to create a `udev rules` file containing a Universal Serial Bus (USB) configuration for each type of device that was used for development. It was necessary to log in as `root` to create the file `/etc/udev/rules.d/51-android.rules`, followed by the execution of the command `chmod a+r /etc/udev/rules.d/51-android.rules`. This file contains an unique vendor IDentification (ID) that identifies the device manufacturer. For each device, a line needs to be added to the file, with a rule similar to:
`SUBSYSTEM=="usb", ATTRidVendor=="04e8", MODE="0666", GROUP="plugdev".`
It can be noticed that `"04e8"` represents the ID of the vendor, which was *Samsung* in this case [Dev]. The `MODE` attribute specifies read and write permissions, and `GROUP` defines which group owns the device.

In order to make the device recognizable by the system, it was also required to enable on-device developer options via Settings. Android devices have several developer options that enables the user to deploy the created application into a real device or do any type of debugging over USB.

Additionally, another technology was used to interact with the system. JSON technology was used to store data like, for example, the blockchain and users addresses. JSON, similarly to eXtensible Markup Language (XML), is a lightweight and a standard format to data-interchange, that makes the reading and writing of data easier to developers, being easy for machines to parse and generate [Len13]. Due to its simplicity and to the ease of writing a JSON parser, this data format, is now standard for many platforms. In order to ease JSON data manipulation, and its debugging, two platforms were used. *JSON Lint Validator* was used to construct the most complex JSON chains and verify their consistence (i.e., if the JSON structure was well constructed or not) [CC]. *JSON Editor Online* was used in order to have a more visual and formatted representation of the chain, to ease its handling and blocks debugging [dJ].

3.2.1 System Arquitecture

The diagram in Figure 3.1 provides a description of hardware and software components, as well as their interaction with other needed components for the Bitpoints prototype. As can be seen in the diagram, the architecture of the system is based on the P2P paradigm which, by its turn, is supported by the Transmission Control Protocol (TCP)/Internet Protocol (IP) communication

protocol suite.

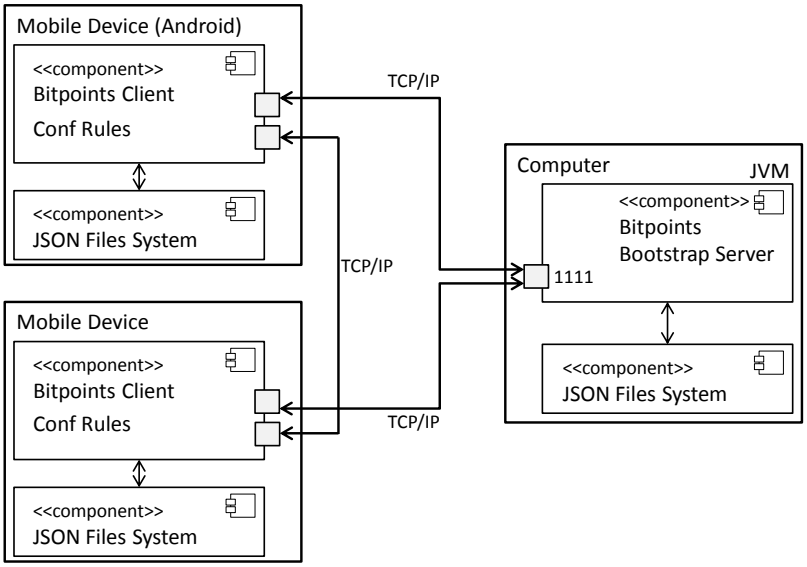


Figure 3.1: Components diagram for the Bitpoints system.

This system was developed in Java programming language, mainly focused on Android OS devices, and resorts to one or more files, where the information is stored. The system consists of two distinct applications: an application for the clients, willing to use the system (Bitpoints Client), and a server side application (Bootstrap application), to be used by the controlling entity of the system. Although all objectives of a normal crypto-currency could be achieved only with a P2P application, as mentioned earlier in this document, this Bootstrap application was added to the system in order to implement some client-server services. This was designed this way in order to reduce the computational load on mobile devices, facilitate the protocol implementation, and additionally assure some control over the generation of points.

3.3 Requirement Analysis

The requirements specification is one of the most important tasks in system analysis. This section identifies and briefly describes the functional and non-functional requirements identified for the system under analysis and, as such, it is divided into two main subsections.

3.3.1 Functional Requirements

The functional requirements refer to the features that the system should provide. For Bitpoints, the system should allow:

- the user to check his balance and the number of transactions performed;
- a transaction of a given amount for a specific address;

- the user to generate new personal addresses;
- the user to manage the personal addresses;
- the user to consult all transactions performed;
- adding a new public address to the contacts list;
- the user to manage the contacts list;
- the administrator to configure the system.

3.3.2 Non-functional Requirements

Non-functional requirements are those which are related to the environment where the system is inserted. In this subsection the different types of non-functional requirements are described.

The ease of use of the system is particularly important for mobile applications and for this specific system under analysis. The ease of use non-functional requirements are as follows:

- The system should provide features that are intuitive and easy to perform;
- The menus and the underlying functionalities should consist of a set with a reduced number of options in order to make the system less confusing.

The system performance is also important, since lengthy response times can demotivate the less patient users. The performance and scalability non-functional requirements are as follows:

- The response time of the initial synchronization, made when connecting the system, may differ according to the size of the chain to transfer;
- Apart from the initial synchronization, the system should provide the necessary information to the user in less than 3 seconds;
- The response time of the listing of all transactions will vary according to the number of transactions that were previously performed;
- The system must be scalable and should not have characteristics of any kind that prevent or restrict their full usage, in small or large institutions;
- It should guarantee its operation seamlessly, regardless of the number of users.

Failures or maintenance which prevent users of fully enjoying the system or adversely affect user experience, may lead the implementation to failure. The system availability non-functional

requirements are as follows:

- The system should be always available to users on every working days of the year;
- Planned or unplanned downtime for system maintenance or bug correction should not exceed 8 hours.

The technical non-functional requirements are also important, since they are necessary to establish rules and ensure the proper functioning of the system. The presented system is based on two distinct architectural models: the client-server and the P2P models, and it is important that the system satisfies, whenever possible and for a better use experience, the following technical specifications:

- The server must have a hardware environment with at least 4 GB RAM and one hard drive with at least 500 GB. The server OS environment may be one of the Microsoft Windows family, a Mac OS, or an Unix or Linux OS.
- The client must have an hardware environment (minimum recommended) with 512 MB of internal memory, 300 MB Random Access Memory (RAM), at least a 800 MHz processor and a 2 MP camera (if QR codes are to be used). The client OS environment (minimum recommended) must be Android version 2.X or higher.

3.4 Identification of Use Cases

This section presents the diagrams for the use cases that were identified during the preliminary analysis of the system. This type of diagrams represent a graphical overview of the features provided by the system to actors. Two actors were identified for the Bitpoints system: the common user and the administrator. The user interacts with the system at a higher level. The administrator is responsible for controlling and configuring the system over the aforementioned server application.

3.4.1 General Use Case

The most general use case diagram is included in Figure 3.2. It shows the most important functionalities provided by the Bitpoints system for both client-side and server-side applications.

Table 3.1 describes the actors included in the previously mentioned diagram with more detail. It starts from the description of a normal user, which is considered the main actor of the system, or at least it is the one for which functionalities are available.

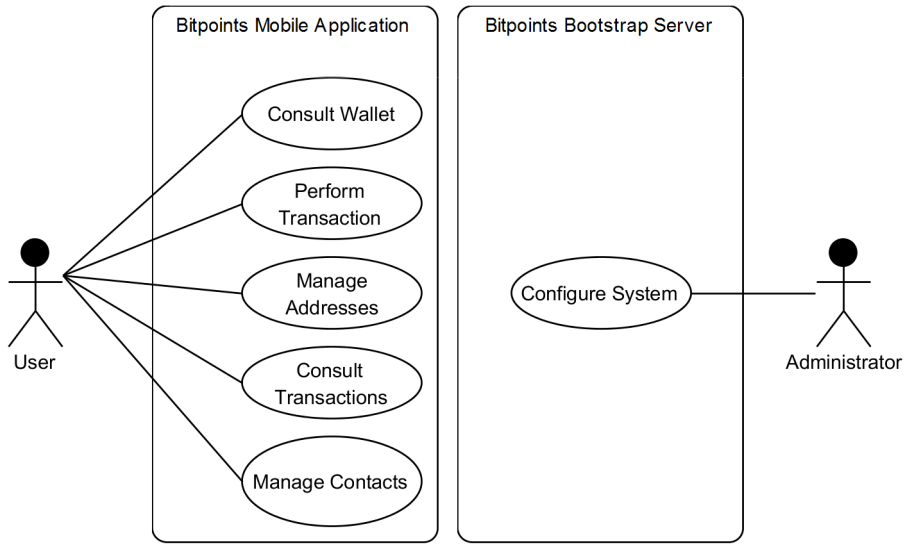


Figure 3.2: General use case diagram.

Table 3.1: Identification and description of actors.

Actors	Description
User	The main actor of this system is the mobile application user. This user is the only one who interacts with the application and has access to most of the existing features. When interacting with the system, the user can use the following features: *consult wallet, perform a transaction, manage private addresses, consult performed transactions* and *manage contacts*. All features have been thought and designed for the best user experience.
Administrator	This user is the entity capable of configuring and controlling the functioning of the system. When interacting with the system, the administrator can edit the configuration file where the system fetches the information used during execution. The configuration file enables changing the rewards, in terms of BTPs given to first time users and the value for mining blocks.

Below some specific use cases are further detailed using diagrams. Nonetheless, some of them, and because they are simple, are briefly described in this section:

Consult wallet - This use case starts when the user connects to the network supporting the system. During the first communication with the server, the mobile application should synchronize all necessary information, obtaining the history of all transactions until that moment. After the first synchronization, the user can access the wallet by clicking on the option Consult Wallet. This view should be updated each time a new deal is performed;

Perform transaction - In this use case, the user performs a transaction to another user, pays for a service or product. To this use case to be successfully achieved, the initial synchronization stage is mandatory;

Consult transactions - This use case concerns the scenario in which the actor tries to obtain

information about transactions. In order to be successful, the mobile application must be synchronized with the server. Therefore, the user can see all his or her transactions, delivered in a uniform user interface.

3.4.2 Manage Addresses Use Case

This section describes the functionalities related with the management of the user personal addresses. The use case diagram in Figure 3.3 adds detail to the third use case depicted in Figure 3.2.

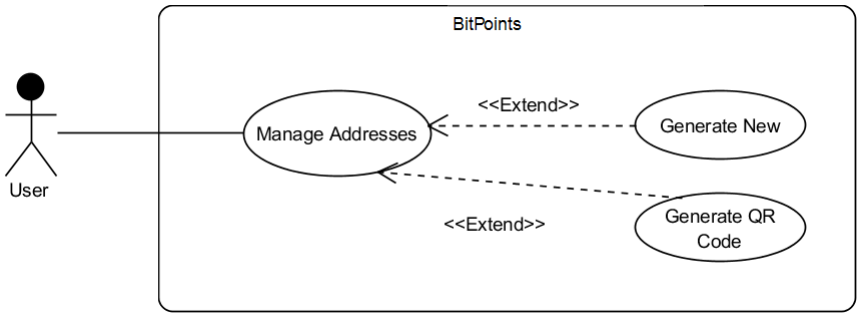


Figure 3.3: Use case diagram for the *manage addresses* functionality.

This use case is interesting because the functionality it refers to does not require initial synchronization with the network. This enables the user to consult locally all the unique key pairs (private and public keys), that belong to him or her.

As depicted in the diagram, the user is able to perform two distinct operations: Generate New and Generate QR Code. Through the first option, the user can generate a new personal address (a key pair) and is able to generate as many addresses as needed. Notice that the option to delete personal addresses directly is not modelled in the diagram. This subject will be discussed in the next chapter. The second option provide the means to generate a QR code from a specific public key, in order to ease the exchange of contacts with other users.

3.4.3 Manage Contacts Use Case

The *manage contacts* use case is described in this section, as well as its more detailed functionalities. This use case is very similar to the one described in the previous section. The respective diagram is included in Figure 3.4. It also contains two options to add contacts to the application, and one to remove contacts (public keys) from other users. Adding new contacts can either be done by manually inserting public keys or by reading QR codes with encoded public keys.

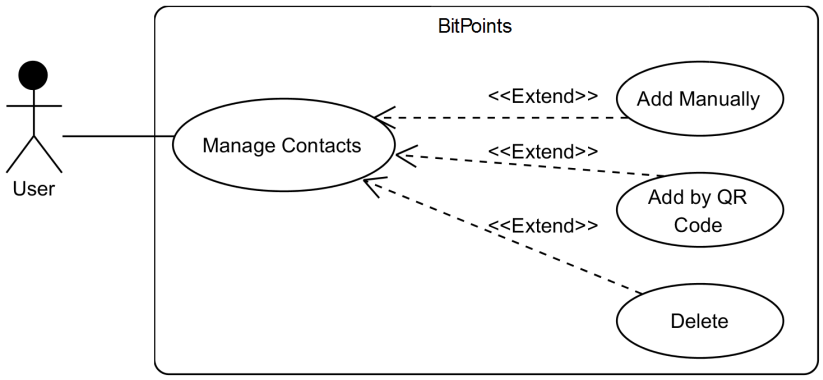


Figure 3.4: *Manage contacts* use case diagram.

3.5 Activity Diagrams

Activity diagrams are the part of software engineering that graphically represents the workflows of activities and actions in a stepwise manner. In this section some activity diagrams are going to be presented and briefly discussed. However, not all activities of the system will be outlined in this document. Only the ones that are considered to be more representative or complex will be subject to discussion.

3.5.1 Perform Transaction Activity

In this subsection, the *performing transaction* use case will be described resorting to the activity diagram represented in Figure 3.5. Notice that the client must be synchronized with the network to perform this operation.

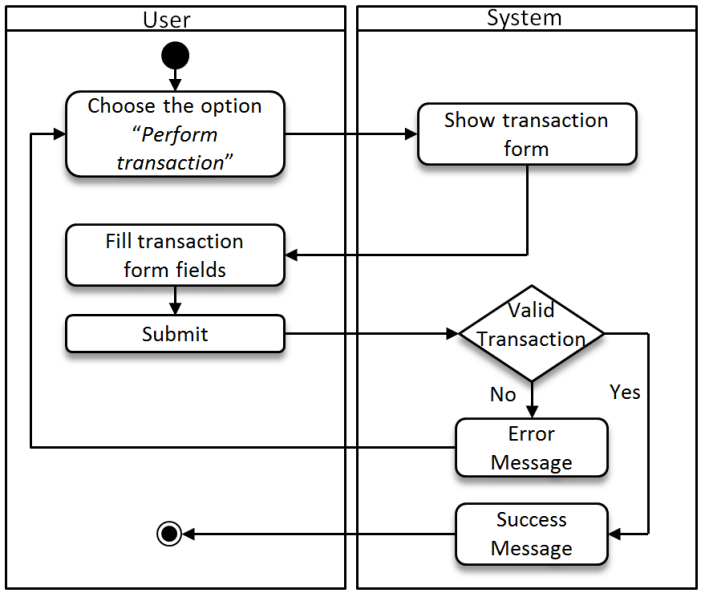


Figure 3.5: Activity diagram for the *perform transaction* use case.

Table 3.2 describes the steps that the user must follow to perform a transaction.

Table 3.2: Description of the steps to perform a transaction.

Step	Description
1	The user selects the perform transaction option.
2	The system displays the form with the details of the transaction (destination and amount).
3	The user enters the public key of the destination and the amount to transfer, and submits the transaction.
4	The system checks if the transaction is valid or not.
5	If the inserted data is valid and the transaction can be performed, the transaction is sent and the activity ends. The user can then choose any another options.
6	If the transaction is invalid, an error is displayed and the flow returns to the initial state.

3.5.2 Generate Address Activity

This subsection is focused on the *generate address* use case and the respective diagram is illustrated in Figure 3.6.

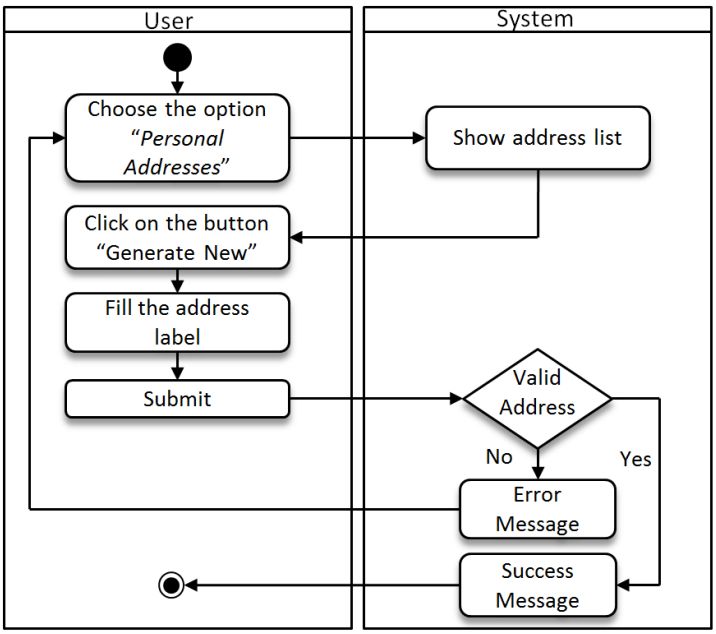


Figure 3.6: Activity diagram for the *generate address* use case.

Table 3.3 describes the steps defining the aforementioned activity.

Table 3.3: Steps of the activity to generate a new address.

Step	Description
1	The user selects the address book option.
2	The system displays the list of private addresses.
3	The user clicks on generate new address.
4	The user inserts a label for the generated address and submits it.

3.5.3 Insert Contact Activity

The *insert contact* specific use case will be described in this subsection using the activity diagram depicted in Figure 3.7. Notice that this is a functionality that does not need synchronization with the network.

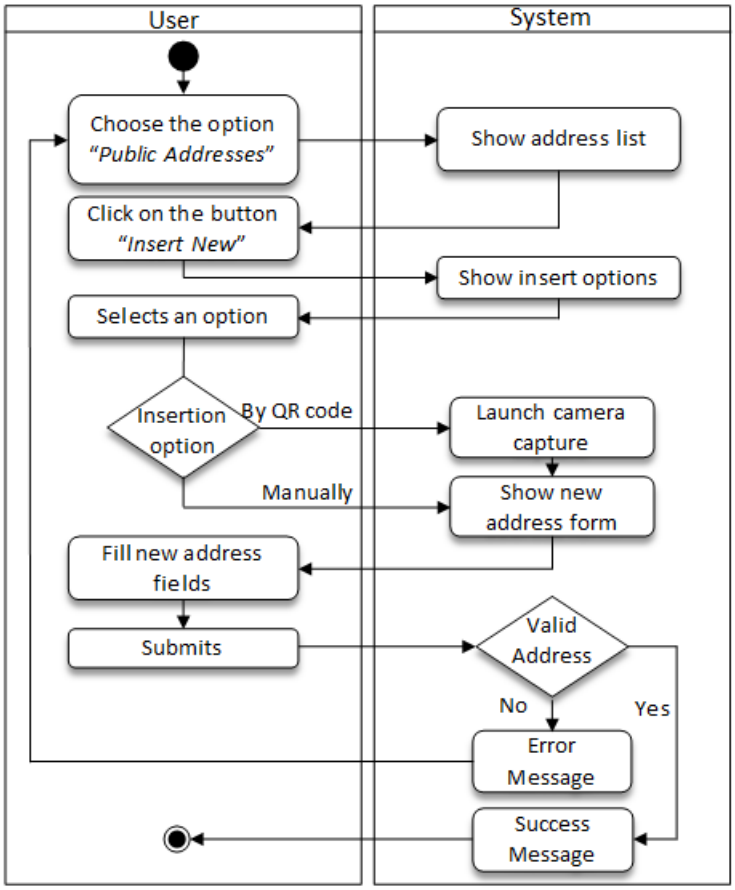


Figure 3.7: Activity diagram for the *insert contact* use case.

Table 3.4 describes the steps defining the aforementioned activity.

Table 3.4: Steps of the activity to add a new contact.

Step	Description
1	The user selects the Public Addresses option.
2	The system displays the list of public addresses.
3	The user clicks on Add New Address.
4	The system shows the available input options (manually or by QR code).
5	The user selects one of the options.
6	If the QR code option was selected, the system will launch the camera to scan the QR code, returning to the form at the end.Otherwise, if the selected option was manually the system goes directly to the form.
7	The user inserts the data of the new address and submits it.
8	The system checks if the address is valid or not.
9	If the inserted data is valid (the size of the key is 256 bits and characters are hexadecimal), the address is saved, a success message is displayed and the flow ends. The user can now choose another option on the system.
10	If the address is invalid, an error is displayed and the flow returns to the initial state.

3.6 Class Diagrams

Class diagrams represent the structure and relationships between the classes used in the system implementation. This section presents and describes de class diagrams of the system, but the attributes were taken out for the sake of readability. Only the more important and complex classes are explained below.

The essential classes concerning the Bootstrap server, represented in Figure 3.8, can be briefly described as follows:

`Bootstrap` - This class is responsible for starting the server, prepare it by obtaining the necessary data from the configuration file, reading the blockchain to memory, and launching the services that will be waiting for incoming connections. Herein, the Bootstrap will store the IP addresses of the connected devices to be used later;

`BootstrapSync` - This class will launch a thread that is responsible for synchronizing the Bootstrap blockchain with the connected peer, by sending it the blockchain file. The referred thread is thrown only the first time that a peer connects the system;

`BootstrapGetTime` - This class allows the peer, to obtain the system timestamp when needed;

`BootstrapFindPeer` - When a peer tries to reach another one in order to send a transaction, the thread encapsulated by this class is thrown. The software included in this class is responsible for handling the message and broadcasting it to all connected nodes;

`BootstrapReceiveProof` - When clients compute the proof-of-work they need to send it to Bootstrap server for validation. This is a critical class, and it is responsible for launching the thread that receives the computed block and validates it. With the help of the `BootstrapUtils` class, the software included is able to broadcast all transactions inside invalid blocks or broadcast the valid ones;

`BootstrapBroadcast` - The software in this class is responsible for sending the received transactions to all connected nodes;

`BootstrapVariables` - This class contains some global variables that are needed for the correct functioning of the server;

`_Input` - This class was introduced with the objective of handling the transactions input values more easely;

`BootstrapUtils` - As can be seen in Figure 3.8, this is the most important class of the system, and all auxiliary methods are included herein. The `prepareBootsrap()` method is responsible for reading and converting the blockchain to JSON data, and prepare the server to receive client communications based on the configuration file. Methods to validate blocks, transactions, and their respective hashes and digital signatures (*validateBlock()* and `verifyConfirmedTransaction()`) are included as well. To compute hash values (`computingHash()`), and broadcast confirmed blocks or unconfirmed transactions (`broadcastLastBlock()` and `broadcastUnconfirmedTxs()`), some methods were also implemented herein.

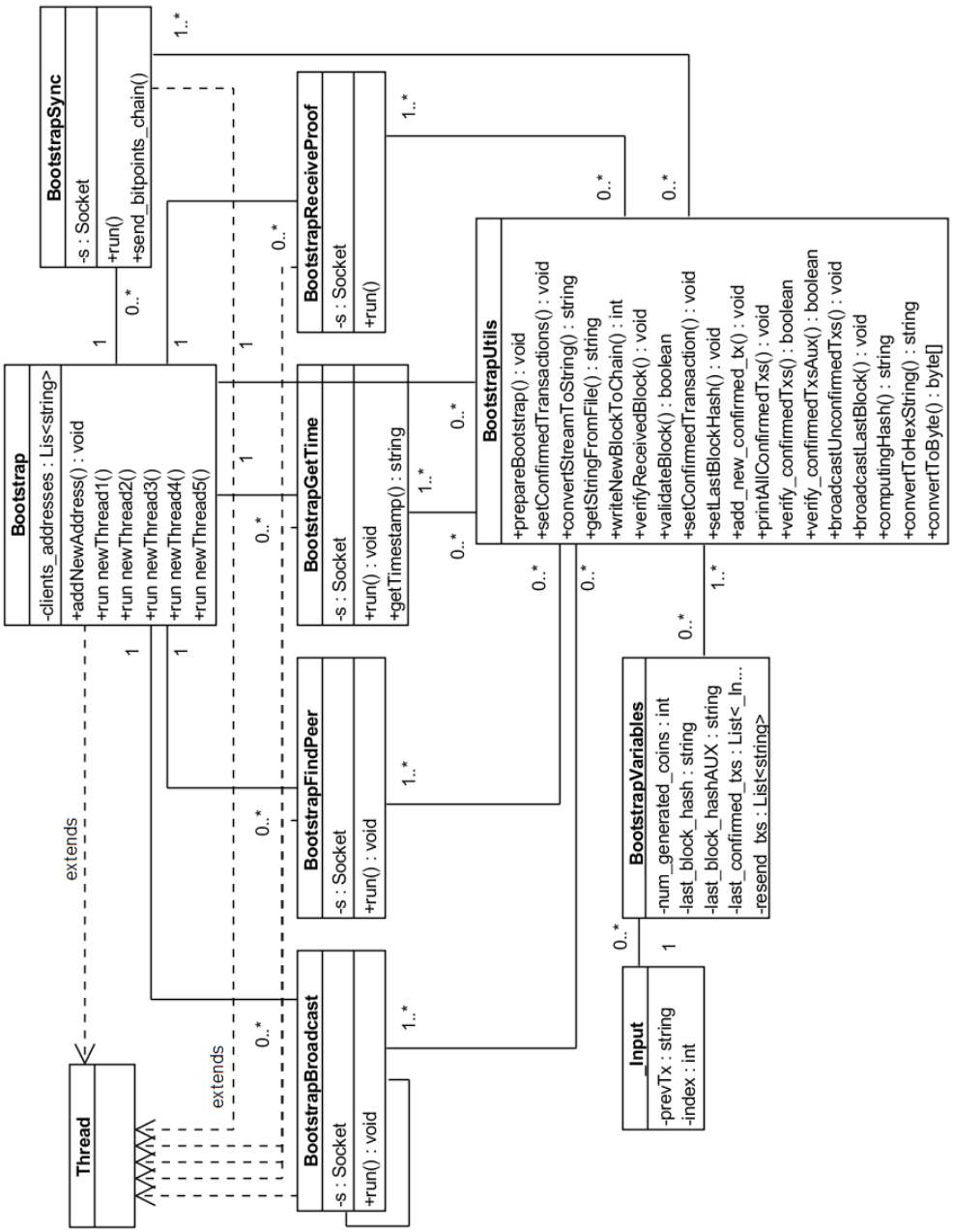


Figure 3.8: Class diagram for the Bootstrap application.

The essential classes comprising the model for the Bitpoints mobile application, represented in Figure 3.9, can be briefly described as follows:

`MainActivity` - This class is responsible for creating the main application structure, namely the application User Interface (UI) (tabs). The methods of this class create the tabs and functionalities. Initially, the application verifies if it is the first time it is being executed. In that case, one tutorial (implemented in the `Showcase` class) is executed. It will help and assist the user through its first interaction. Next, the application verifies if there is a QR code reader already installed on the device, otherwise a dialogue is shown allowing the installation of a default QR code reader. The suggestion to install one of the most popular scanners (the *Barcode Scanner* by *ZXing Team* [Pla14]) is harcoded in this class. The default (first) private address is also generated by one of the methods, followed by the synchronization of the system. Herein, services to for listening communications are also included;

`Showcase` - When the application is installed and ran for the first time, this class launches a small tutorial to help the user understand how to use the application;

`HomeActivity` - This activity is responsible for showing all wallet information to the user;

`SendActivity` - This activity contains the code to control the UI used to preform transactions;

`MyAddressesActivity` - This activity contains the code to control the UI for generating private addresses (private and public keys) and listing the existing ones;

`TransactionsActivity` - This activity shows the list of all transactions performed until moment;

`AddressesActivity` - This activity lists all user contacts (public keys) of the wallet and allows to add new ones;

`HandleConnections` - This class contains all necessary methods to handle communications over the network (i.e., with the Bootstrap and with other peers). There are methods to create sockets for listening (`listenPeerAndSend()` and `listenBootstrap()`), or even to establish communications with the Bootstrap server or peers (`sendTransaction()` and `sendProofOf-Work()`);

`ConnectBootStrap` - This class contains the methods used for synchronizing the mobile applications with the system. This class receives and handles the updated blockchain;

`BitpointsUtils` - As can be seen in Figure 3.9, this is one of the most important classes of the mobile application, and it includes useful methods needed by other classes. Some ex-

amples of those methods are the `createAppFolder()`, where the application is prepared and configured. The `genAndSaveAddress()`, `LoadPrivAddresses()`, `SavePubAddress()`, `LoadPubAddresses()` and `RemovePubAddress()` methods are responsible for handling the management of wallet addresses (generate addresses, load them for memory, add, save and remove). Methods responsible for computing hashes, digital signatures, proofs-of-work and the respective validation (`signTransaction()`, `ComputeHash()`, `ProofOfWork()` and `verifySignature()`), are also included herein; Additionally, some conversion methods, as the case of `convertTransactionToJSON()`, `convertBlockToJSON()`, `convertJSONToTransaction()` and `convertJSONToBlock()`, are also implemented in this class;

`BitpointsVariables` - This class contains some global variables needed for the mobile application correct functioning;

`BitpointsUtilsBlockChainOP` - This class refers to methods that were implemented exclusively to handle the blockchain. These methods are mainly intended to blockchain read or write the blockchain (`readChainToJSONArray()`, `updateChain()` and `writeNewBlockToChain()`), and to obtain transactions of a given wallet (`getMyInTransactions()` and `getNotSpentTransactions()`);

`_Block`, `_Transaction`, `_Input` and `_Output` - These classes implement software used easily handle system information, namely blocks, transactions, inputs and outputs, respectively;

`_InTransaction` and `_OutTransaction` - The purpose of these classes is similar to the one of the previous classes, but applied to the operations regarding the Bitpoints protocol. They enable to more easily organize the incoming and outcoming transactions;

`_KeyPair` and `_Address` - These classes are also used to assist the implementation of the protocol, and they are used to store wallet addresses and contacts, respectively.

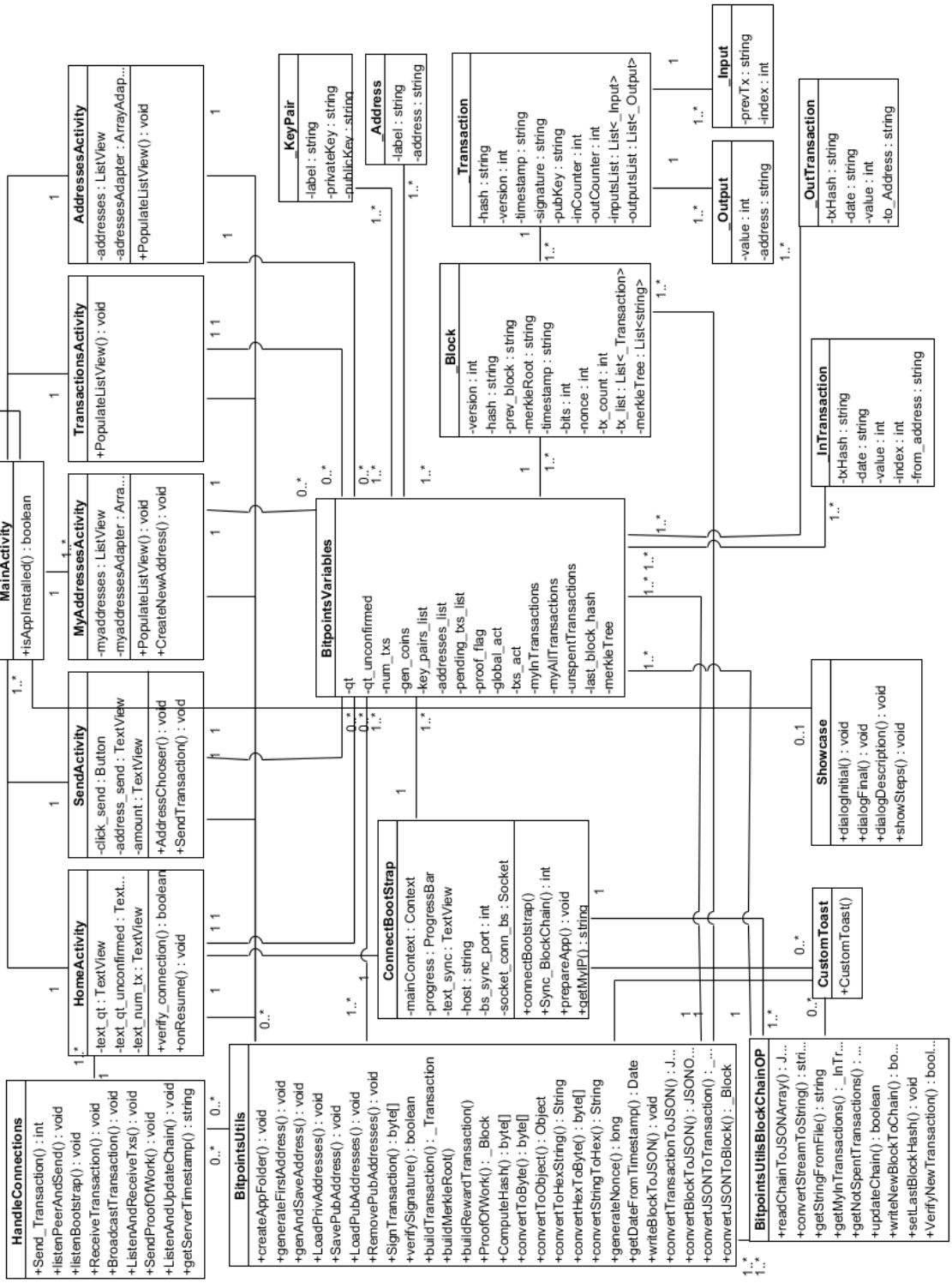


Figure 3.9: Class diagram for the mobile application.

3.7 Conclusions

This chapter on software engineering focuses on some of the details that were taken into consideration during the initial phase of this masters programme, namely during the design of the system that was later implemented. It shows that some attention was paid to the fact that the main interface to the system was an application for mobile devices, which was to be used by,

potentially, less technically skilled users.

The engineered system uses an hybrid architecture, combining the client-server and the P2P architectures. This design gives rise to a semi-decentralized system, in which a bootstrap server intermediates or helps initializing some of the communications, and will ease the implementation of additional features. The P2P architecture is required by the Bitcoin protocol, with which this system is mostly in agreement. The models presented in the class diagrams, namely in Figure 3.8, are also indicative of the similarity to Bitcoin, since most of the classes/objects for the server side correspond to structures defined by the Bitcoin system. The classes that are not directly derived from the Bitcoin specification are related with specific added features.

The bootstrap server is mostly needed due to the potentially resource constrained environment in which the system should operate in and to fulfil specific requirements. For example, fully decentralized operation would not enable the system to control the number of points generated and the usage of powerful computers to unbalance the virtual currency.

Chapter 4

Implementation

4.1 Introduction

One of the most important phases of this masters programme concerns the implementation of a loyalty card for smartphones using a Bitcoin like approach. This chapter describes this phase and discusses some decisions taken during the development of the proposed system and the security concerns. It also shows, via screenshots, the resulting mobile application. Section 4.2) outlines the functioning of the implemented protocol and mentions also the difficulties faced during the implementation of the Android P2P network. Section 4.3 is focused on the Bootstrap server implementation. The proposed mobile application is then described in section 4.4 and finally, security and performance issues are the subject of section 4.5.

4.2 Bitpoints Protocol Communication

The initial phase of the development was mostly dominated by small investigation tasks for solving basilar problems. One of those problems was to find an existing API and classes to build an Android P2P network, without the need of building it from scratch. After some research, it was noticed that, even though there were some implementations available, they were not up to date, and the existing documentation was also scarce. The most popular P2P APIs available on the market at the time of writing this dissertation are: (i) the JuXTApose (JXTA) API; (ii) the FreePastry API; and (iii) the TomP2P API.

JXTA API is an open source P2P protocol, based upon a set of open XML protocols, begun by Sun Microsystems in 2001 [Ora13]. During the initial phase of this masters, the mentioned API was abandoned by the developer, and as such was not used for the purpose at hands. The base code was old, and contributing to the library would require cleaning it first. The FreePastry API [Hoy09] is an open-source overlay and routing network for the implementation of a Distributed Hash Table (DHT), and the most recent FreePastry API was written in Java 5 programming language. The source code was also old and because of that was not used either. Finally, the TomP2P API is an advanced implementation of a DHT which provides a decentralized key-value infrastructure for distributed systems [Boc13]. It should be mentioned, however, that all of these APIs are written for the Java native implementation, which is not exactly the same used in the Android applications. Since these APIs were not specifically designed for Android, their usage raises many compatibility issues that are hard to solve, because the libraries available on

the Internet are poorly documented. The API that could perhaps be used was TomP2P because an Android implementation exists (version 4.0). The biggest disadvantage of this API is that its version is old, and since the latest release at the time of writing of this dissertation is 4.4 (for native Java), the Android implementation only works in the older OS versions.

Notwithstanding the problems to build a system compatible with different Android versions, it is important to highlight that an effort to target both older and newer versions is necessary, since there are many Android versions on the market and customers should not be prejudiced because having a specific OS version. Figure 4.1 depicts the market share, in terms of Android versions, in April, 2013 [Mic13]. Although version 4.x has been gaining ground in the market of mobile devices, approximately 38% of these devices still work under version 2.3, clearly showing that versioning and compatibility are important aspects to consider.

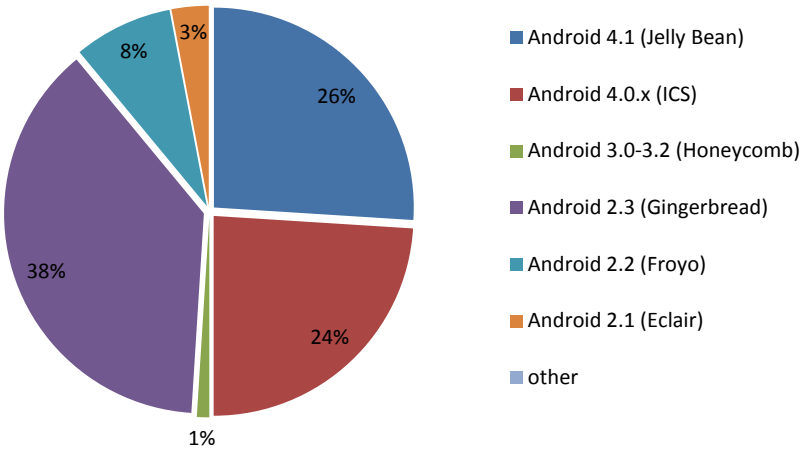


Figure 4.1: Market share in terms of the versions of the Android OS, as of April 10, 2013 [Mic13].

To overcome the aforementioned problems, it was decided to implement a completely functional P2P library from scratch for this project. This choice impacted the time schedule for the development phase, taking longer then expected. As previously mentioned, the system engineered in the scope of this masters includes a rendezvous host, also called the Bootstrap. Figure 4.2 schematizes the role of the intervenients in the P2P network and the interactions between the network entities. Even though this figure can not be considered a sequence diagram, a similar Unified Modeling Language (UML) notation was used, for the sake of consistence.
At the very beginning of this masters programme, the inclusion of the Bootstrap server was not the preferred choice, mostly due to the fact that Bitcoin does not use it either. Nonetheless, given the specific application scenario and purposes of the system, it was opted otherwise. Actually, the existence of a Bootstrap server enables relieving some computational load from the mobile devices, and to add additional controlling functionalities. As such, an application that acts constantly as a server was also implemented. Some functionalities will be described later on.

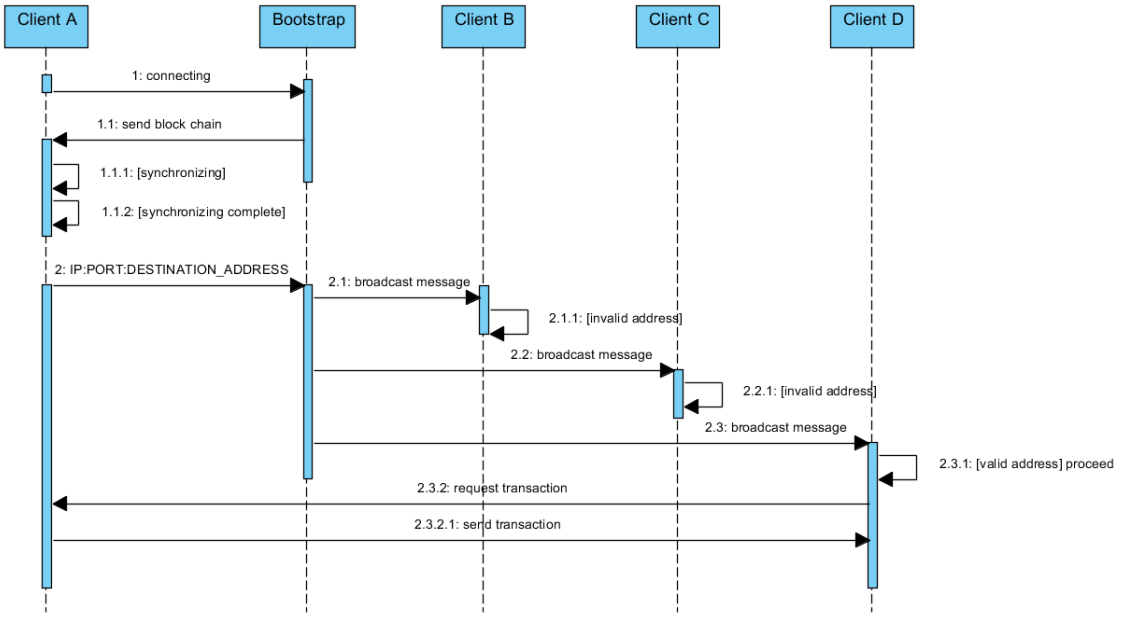


Figure 4.2: Representation of the interactions between the several network entities and the Bootstrap during a transaction.

The Bootstrap server holds key information regarding the functioning of the network, including the IP addresses of the connected nodes and the blockchain. When the server application starts, it keeps listening for incoming messages on port 1111. According to the diagram in Figure 4.2, when a node (Client A) tries to establish a connection for the first time (message 1 of the diagram), the Bootstrap server stores the information of the respective node in order to contact it later. Immediately after, the Bootstrap starts a synchronization process, by sending the blockchain to Client A. Upon the conclusion of the synchronization, the client is ready to use the network, perform the mining process (the node will be waiting for transactions to be verified) and establish communications with other connected peers.

When a client (Client A in this particular scenario) wishes to perform a transaction, a communication with the Bootstrap server is initiated. The server and the connected nodes will then perform the following small protocol, comprised by the next steps:

- Client A sends a message to the Bootstrap server containing its IP address, a randomly generated port number and a destination address (public key of the recipient). In that particular situation, the message 2 would be, e.g., `192.168.1.14:1050:30816...d8124` (notice that the values are separated by the colon punctuation mark). After that, Client A starts listening on port 1050 for the destination peer communication;

- The Bootstrap server receives the message, and since it does not know for who the message is, it broadcasts it (message 2.1) to all connected nodes, except to the sender;

- After the previous step, all the connected peers will receive the message spread by the

Bootstrap. The nodes will validate the message and only the owner (client D, in this case) will accept it. The remaining nodes will discard it and keep listening for other incoming messages. It can be proven that is no gain, other then delaying the transaction for a node to answer back when a message is not intended to it;

- In the next to last step of the diagram, Client D will try to reach Client A directly, by sending a request message to the IP address and port included in the message;

- Client A receives the request message and knows that the destination peer is waiting to receive the transaction. Finally, the message, which includes the transaction, is sent to the recipient node (Client D) and subsequently validated (further discussed below).

The aforementioned procedure was devised to enable P2P between two different nodes of the system (Client A and Client D, respectively). The destination node (Client D) is the interested party in confirming the transaction, because it is the recipient of the BTPs. It must assure that the transaction is inserted into a block after being validated, and buried into the blockchain. As such Client D must send the transaction to the Bootstrap server, so it can be broadcasted over the network. The procedure referred in last is depicted in Figure 4.3.

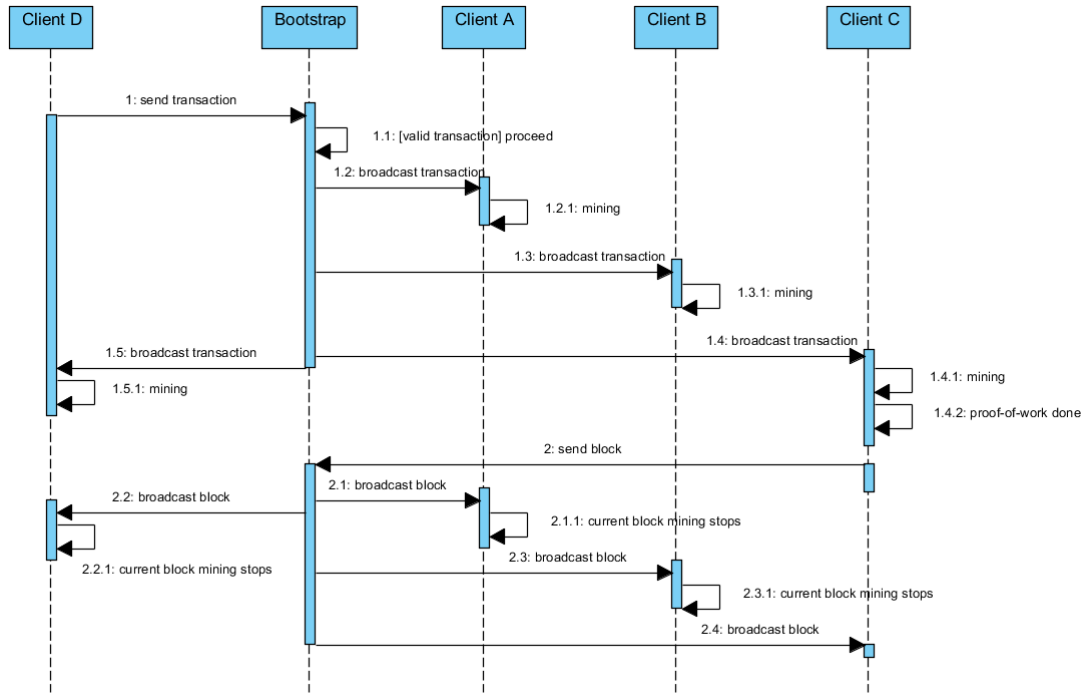


Figure 4.3: Representation of the interactions between the several network entities and the Bootstrap server during a transaction confirmation.

The steps performed by the Bootstrap server and the connected nodes can be discriminated as follows:

- The last step of the diagram depicted in Figure 4.2 consists of receiving and verifying

the transaction message for Client D. If the transaction is valid it sends it to the Bootstrap server (message 1 of Figure 4.3) to be broadcasted, otherwise the transaction is discarded;

- Once the Bootstrap receives the message, it also validates its digital signature. If the digital signature is valid, the message is broadcasted to the network to be confirmed, otherwise the confirmation process ends here;

- Once the message is broadcasted to the network, the mining nodes will receive, validate the digital signature and insert it into a new block. This block may contain other unconfirmed transactions and a transaction to itself with the reward. The mining nodes will then compute the proof-of-work over that block;

- The first node to find the correct nonce, i.e., the nonce that solves the proof-of-work, it will send the block and the nonce to the Bootstrap server. The node will wait for the block to be included in the blockchain to know if the reward was credited to it;

- Finally, the Bootstrap receives the block and validates it. If the block is valid (i.e., it comes with a correct nonce for the required difficulty), the message is inserted into the Bootstrap blockchain and spread to the network. At this point, the transactions included in this block are confirmed and the mining nodes who are still mining some of these confirmed transactions, will discard them and continue with the unconfirmed ones.

4.3 Bootstrap Implementation

As previously mentioned, a Bootstrap server was implemented in order to manage this semi-decentralized system. Several services were integrated into this server, most of them mentioned in Chapter 3. These services allow the synchronization of the nodes, as well as the interaction between them. The server is composed by several threads and server sockets listening in different ports, for the sake of the performance of the system. These services are responsible for synchronizing the connected clients, return the timestamp used in the transactions, find peers and send them transactions, broadcasting transactions and proofs-of-work.

In order to explain the Bootstrap implementation, the discussion included below follows the sequence of steps depicted in Figures 4.2 and 4.3 (to perform a transaction).

Before running the server, it is necessary to consider a configuration file included with the source code, where some rules can be established by the controlling entity. These rules shall include, for example, the number of points that can be offered to the first customers and which ones will receive them. The configuration file can only be edited before the Bootstrap starts and it is where it will obtain the essential information to the correct functioning of the network. When the Bootstrap starts, the `prepareBootstrap` class is called and the blockchain is read to

memory and prepared, as well as the configuration parameters.

When the server starts, a few server sockets are created and set to the listening state, awaiting for some peer to try to establish connection to the port 1111. When a peer is attempting to communicate with the server in this socket, it will trigger the `BootstrapSync` thread, which stores the peer IP address in a list to be used later. This list is updated from time to time by sending messages to all listed addresses. The IP of the peers that, for some reason, do not respond or for which a time-out exception is raised, will be discarded from the list and a new connection will be required for that peer. This thread is responsible for sending missing portions of the blockchain to the respective node also.

Meanwhile, another server socket is listening in port 1112, awaiting for a peer to try to reach another peer when performing a transaction. When the server receives the connection message (IP Address, port number and the destination public address), it will broadcast the message to connected nodes. Another communication with the server is established by the receiving party with the objective of having the transaction verified. This communication involves the receiving party sending the transaction to the Bootstrap. When the server receives the message, in port 1113, a new thread is thrown. This thread contains the procedures for validating and broadcasting the transaction to all peers.

Later on, once a mining node has received and successfully confirmed the transaction (by performing the proof-of-work), it will send the respective block to the Bootstrap, using the TCP port 1114. Upon the reception of this message, the most important thread will be thrown. The `BootstrapReceiveProof` thread performs important procedures related with privacy and security of the protocol. This step is very important because the server can not take the chance of updating the chain with an invalid block. This block must be validated before the blockchain is updated and broadcasted.

This validating thread starts by reading the block message from the socket. It will then go through the following steps:

- Verify if the block was, intentionally or not, modified during transmission by verifying its hash value;
- When the server is listening for new incoming blocks, it may happen that two blocks arrives at the same time. The server only considers the block which came first, based on the timestamp. The remaining blocks are discarded;
- Verify if the last block hash in the server (the correct one) matches with the received previous block field (*prev_block* hash);

- Verify whether any of the transactions has been previously confirmed (already spent);
- Verify whether any of the transactions was, intentionally or not, modified during transmission by verifying their digital signatures and their hashes.

If if one of the transactions is considered invalid during the verification process, the validation process ends and the block is considered invalid. All transaction fields and the block itself must be valid in order to be inserted into the blockchain. If the block is invalid, the embedded transactions which have not been confirmed yet, are disseminated back to the network. Finally, if the block passes the validation process, the blockchain is updated. The new transactions are stored as confirmed transactions, and the new block is broadcasted to the network.

When a peer is mining a block, it needs a valid timestamp. To cope with this requirement and also address problems related with forgery and different time zones, a service to return an universal timestamp was implemented. When the peer needs a timestamp, it requests it to the Bootstrap server.

4.4 Mobile Application Implementation

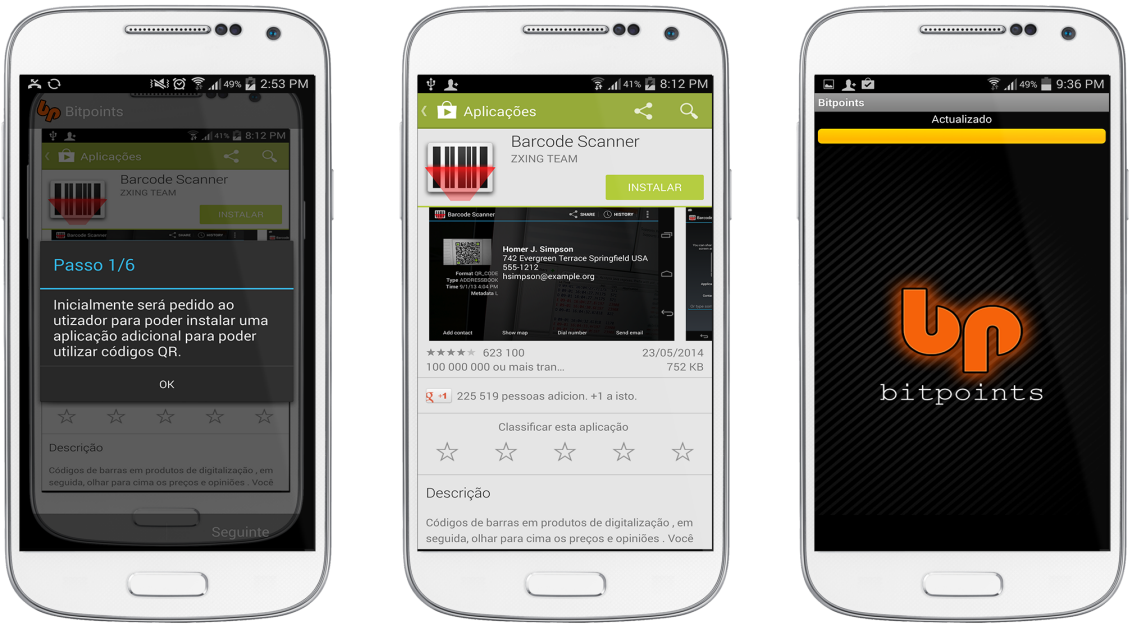
This section will be focused on the implementation of the mobile application developed in this masters programme. The explanation is divided into six main parts. The discussion will converge to the explanation of the application from the point-of-view of the users. This will include the functioning of the interface, its implementation and the flow of its essential steps.

4.4.1 Main Activity

The main activity, depicted in Figure 4.4, is one of the most important activities in the application. This activity is in charge of starting the application, preparing the application folder, presenting the application tutorial to the user and synchronize the client with the Bootstrap server, by communicating with him. It is also responsible for launching the remaining activities, namely (i) home; (ii) sending; (iii) private addresses; (iv) list of transactions; and (v) public addresses.

As can be seen in the left side of Figure 4.4, when the application starts for the first time, a small tutorial will be launched. This six-step tutorial was designed to guide the user, giving him a very detailed explanation of the application functioning and making it more suitable for him. To allow the system to be practical, and because it is difficult to handle addresses manually, a service for automatically generating and reading QR codes was also added. After completing the tutorial, a verification for QR code readers will be made and, if there is some QR code reader API installed on the device, the application flow proceeds to the next step, otherwise, an intent is issued. This intent leads the user to Google play and asks for the installation of

a QR code reader API. The application workflow may proceed whether or not the user installs the QR code reader. Subsequently, and only on the first run of the application, the application folder is prepared, meaning that a method for generating the first private address for the user is executed, and that the files to store the wallet data, like private and public addresses, are created.



(a) Application showcase. (b) QR code reader installation. (c) Application synchronization.

Figure 4.4: Screenshots of the first execution of the Bitcoin mobile client.

The mobile application developed in this masters programme makes use of ECC primitives, namely for the generation of the private and public addresses (the security provider for the Java Cryptography Extension (JCE) used in the scope of this work was `org.spongycastle.jcajce.provider.asymmetric.EC`) [Doc13]. ECC is emerging as an attractive public-key crypto-system mostly for mobile environments because it offers, compared to traditional crypto-systems like Rivest, Shamir and Adleman (RSA), equivalent security with smaller key sizes [Ora14]. Smaller sizes results in faster computations, lower computer processing resources, memory and bandwidth savings. The default key size of 256 bits was used in this implementation. Table 4.1 compares equivalent key sizes, in terms of security, for the RSA and ECC crypto-systems.

Table 4.1: NIST recommended asymmetric key sizes [Bom13].

Elliptic Curve Modulus	RSA Modulus	Expected Lifetime
111 Bits	512 Bits	Expired
160 Bits	1024 Bits	2010
224 Bits	2048 Bits	2030
256 Bits	3072 Bits	2031+
384 Bits	7680 Bits	2031+
512 Bits	15360 Bits	2031+

Finally, after the application folder is prepared, and if the device has network connection, the client tries to reach the Bootstrap synchronization service in port 1111. Once the synchronization is initialized, the application reads the blockchain and stores the transactions of that client, in global variables. The incoming transactions (i.e., not spent yet) are separated from the transactions that were already spent. After everything is synchronized and prepared, the application flow proceeds to the next activity: the `Home activity`.

The aforementioned synchronization and preparation procedure makes use of one of the most important classes in the system: the `BitpointsUtilsBlockChainOP`. As previously referred, this class is responsible for handling all the information regarding the blockchain of the system and it comprises some important methods, namely, `readChainToJSONArray()` and `getMyInTransactions()`, where the blockchain is loaded, and all transactions concerning a particular wallet are separated and stored in memory.

4.4.2 Home Activity

The `Home activity` is the place, in the interface, where the user can access an overview of its wallet. This activity provides information like the wallet balance, number of unconfirmed points and number of performed transactions.

When the `Home activity` starts, an analysis of the stored transactions is made and the interface widgets are filled with the respective information. Figure 4.5 contains a screenshot of this activity. At this point of the work flow, the application triggers a few important threads that will update the interface, whenever it receives new data, and that will handle the client communications, resorting to the `HandleConnections` class.

Figure 4.5: Screenshot of the Bitpoints home activity.

The threads launched in this activity and the corresponding description are as follows:

`listenBootstrapConnections` - lets the application waiting for some Bootstrap messages coming from another client. A server socket is created and when it receives a Bootstrap message (IP Address, port number and the destination public address), it verifies if the message is intended for it. If the message is not for this client, it will be ignored. Otherwise, the client will be able to reach the sender address requesting the transaction;

`listenAndReceiveTransactions` - creates a server socket in order to wait for incoming transactions to be confirmed. These transactions are broadcasted by the Bootstrap server and they are ready to be inserted into a block (proof-of-work computing). It is possible that the client receives several transactions at the same time and the received transactions are stored in a list (`pendingTransactionsList`). This list will be waiting that the proof-of-work thread inserts them into a block, and while the proof-of-work is paused (proof-of-work global flag equals to zero) more transactions will be added to that list. Whenever the client receives a new transaction, the digital signature will be verified using the public key of the sender and invalid transactions will always be ignored. The digital signature mechanism used in this implementation is `SHA256withECDSA`;

`computeProofOfWork` - a thread that is paused from time to time due to issues related to the device performance and that will be discussed in section 4.5. When this thread starts, it grabs the existing transactions from the previously mentioned list, and starts computing the proof-of-work. The proof-of-work implementation will be clarified below. When the proof-of-work is finished, the client sends it to the Bootstrap to be broadcasted into the network;

`listenAndUpdateChain` - this thread lets the application waiting for new blocks sent from the server. When a new block reaches the client, it will be inserted into the existing blockchain and the interface objects are updated with the new information from that block. If there is any repeated transaction inside the received block, it will be automatically removed.

The proof-of-work implementation starts by requesting a timestamp and difficulty (for the proof-of-work) to the server and setting the proof-of-work flag equal to one. During its processing, the following steps are performed:

- A 32 bit integer value greater than zero (called nonce) is generated;
- A reward transaction is built. This transaction is responsible for awarding the client (in this case with one point by default) in the case of being the first of the entire network to submit a valid proof-of-work. This transaction seems to be a normal transaction apart from the fact that it is from the client for itself, and all the clients who are computing the

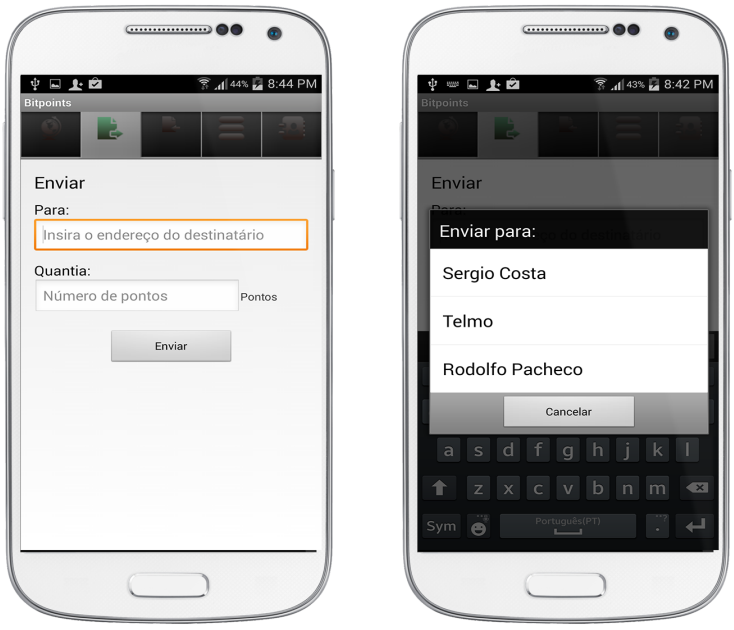
proof-of-work, have a reward transaction for them too;

- While transactions in *pendingTransactionsList* exist, they will be used to built a block and compute a hash value from it. This hash value must have certain number of initial bits equals to zero. This number of bits depends on the difficulty level defined by the server. Bitcoin protocol establishes its difficulty so that each block takes about 10 minutes to be calculated. In this particular scenario, in which there are only a few devices, 1, 9, 12 and 16 bits were used for the purpose of testing. The difficulty is sent by the Bootstrap in the message containing the timestamp. The difficulty is configurable and automatically adjusted to keep a constant rate of Bitpoints generation;

- When the nonce is found, the client connects to the server and sends the obtained proof-of-work, clearing the *pendingTransactionsList* list as well. While the nonce is invalid, it will be incremented until a valid value is found or until there is no more transactions in the *pendingTransactionsList*.

4.4.3 Perform Transaction Activity

The `Perform transaction activity`, whose screenshot is include in Figure 4.6, is the activity where the user is able to send points to others. It has a textbox sensitive to double tapping, which provides access to the address book, from which the user can select the destination. In the textbox related with the amount, the user is able to introduce the desired number of BTPs to be included in the transaction. After fulfilling all the required fields and clicking the submit button, the transaction process begins.



(a) Required fields. (b) Addresses list to choose.

Figure 4.6: Screenshot of the activity for performing transactions.

This process consists of two distinct parts: the transaction building process and the transaction sending process. The building process, which uses the `BitpointsUtils` class, is responsible for constructing a valid transaction starting from existing ones, as previously explained during the presentation of the Bitcoin protocol. During the sending process, the Bootstrap server is queried to find the destination point, using the `HandleConnections` class. Finally, the transaction is sent to the destination. With more detail, the transaction building process consists of the following steps:

1. Request the server timestamp to embed in the transaction;

2. Gather the unspent transactions (previous ones) and select one or more of them, until the desired amount of BTPs is obtained. While the value is below the target, transaction inputs will be built by adding more transactions. When the target value is achieved, the outputs are created. The new transaction can have one or several outputs. If the value of the transactions placed together is equal to the desired one, only one output will be created. If it exceeds the desired value, one output with the desired value to destination address and another one with the remaining points to the address of client itself will be created.

3. If the previous step has been successfully completed, the hash value (computed with the SHA 256) of the transaction will be calculated, and also included into the transaction.

4. Thereafter, in order to assure the transaction authenticity, a digital signature will be generated from the default private address, described with more detail in section 4.4.4.

At this point, the client has a data structure that is ready to be sent to the destination client. In the transmission process, the client generates a random port (not being used), where a socket will be listening for the remote node communication. Meanwhile, it tries to reach the Bootstrap server, sending it the preliminar transaction message: IP Address, port number and the destination public address. Once the Bootstrap discovers the destination peer, the latter will try to connect to the owner of the transaction, asking it for the transaction. The sender accepts the connection and sends back the transaction, thereby ending this process.

4.4.4 Personal Addresses Activity

The `Personal Addresses activity`, for which a screenshot is included in Figure 4.7, also called as the *private addresses activity*, is where the private addresses are generated. It is important to clarify that personal or private addresses, in this context, are key pairs (private and public keys) of the wallet owner. Public keys of other users are shown in public addresses activity.

As previously mentioned, a private address is exactly a key pair generated over ECC, where the

private key is used to digitally sign a transaction. The public key serves the purpose of verifying that digital signature and can be used like a mailing address, to where the transactions are sent. Only the client that owns the private key, corresponding to the public key to where the transaction was sent, can accept it.

It has already been said that a unique default key pair is generated in the client application during its first execution. This default address is represented in Figure 4.7 under the `SemLabel` name. Nonetheless, it is possible to generate a new key pair using the included button. A user can actually generate as many key pairs as she or he wants.

This activity presents a list of all available addresses. It is also recommended that a new address is generated per new transaction, in order to avoid any address tracking attempt, aiming to obtain the association between an address and its respective owner. After an address is generated, the interface does not provide an option to delete it, preventing the user from mistakenly or accidentally delete an important address, and also losing all the points associated with it. Even if the application is uninstalled, the file containing the keys will be kept for an eventual re-installation. Nonetheless, a user may diligently remove de application folder to delete their addresses.

(a) Private addresses of the client.

(b) Private address options.

(c) QR Code generated from the private address.

Figure 4.7: Screenshots of the activity with the features to handle the private addresses.

Another implemented functionality concerns the possibility to generate a QR code, so as to more easely send or share a public key. It is also possible to define the default address to be used in subsequent transactions. Both functionalities are accessed by tapping the address.

4.4.5 Transactions Activity

The `Transactions activity` presents a merely informative screen with the history of all transactions of a given wallet up to the current moment. As can be seen in Figure 4.8, there are three different types of transactions: the incoming, the outgoing and the reward transactions.

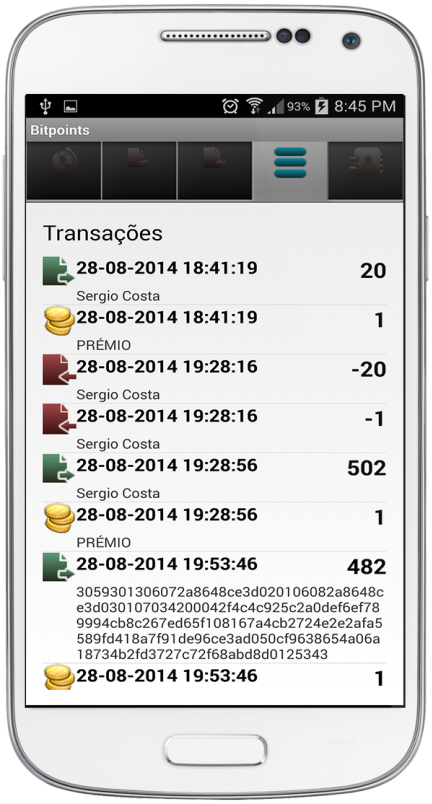


Figure 4.8: Screenshot of the activity presenting the list of transactions.

The incoming transactions, also referred to as inbound transactions, are transactions that someone sent to any address of the wallets of a given user. These include all transaction for which the public address of the user is in the outputs of some other transaction. It is not necessary to know an address in order to receive points from it.

The outgoing transactions, also known as outbound transactions, are those in which any address from a given wallet is included in the input of another transaction. To obtain the spent transactions, the client will first search for all inbound transactions, to then filter out the ones that were already spent. Finally, the award transactions are those that are offered to the clients who complete the proof-of-work first. These are generation points or generation transactions, and it means that they are not based on previous transactions. Their inputs are, for example, `"prevTx":"000...000"` and `"index":-1`.

4.4.6 Address Book Activity

Similarly to the activity concerning the private addresses, the address book consists of a list of public addresses of other users of the system. These addresses can be introduced in the addresses book manually or by reading QR codes. The QR code is recommended because the manually insertion is not practical and can originate errors. However, the manual insertion of 64 hexadecimal characters is provided, giving to the user the freedom to install the application to read QR codes or not.

Figure 4.9 presents the steps to add a new public address to the contacts list. By clicking the button QR Code, an interface to scan the code is spawn. Finally, the dialogue box with the address information is shown and the user is able to introduce a label to that address, followed by its insertion on the list. Unlike the private addresses, these can be removed by tapping them and selecting the delete option, available via dialogue box.

(a) Address book of the wallet.

(b) Reading a public address QR Code.

(c) Address form edition.

Figure 4.9: Screenshots of the activity with the features to handle the public addresses.

4.5 Privacy, Security and Performance Concerns

As previously mentioned, some attention was paid to the privacy, security and performance issues of the system. The privacy issues are partial solved by the mechanisms and concept behind Bitcoin, which guarantee that no association between an address and its owner can be done. Although the blockchain is public, no one knows the identity belonging to the addresses involved in the stored transactions. Notice that, nonetheless, since the communications are not encrypted (because it is not required by the protocol), it may still be possible for an eavesdropper to infer

that a user is using BTPs by monitoring the communication.

To complicate the association between an address and its user, an address generation mechanism was implemented. This means that a user can generate as many addresses as he or she wants, and to avoid the loss of points, there is no feature to remove private addresses. If the user decides to uninstall the application, the folder with the system files is kept on the device. When the user decide to reinstall the application, the old files are restored, instead of creating new ones.

For safety reasons, whenever a client or server receives a message (transaction or block), its digital signature and hash value is always computed and verified, and an invalid transaction or block is always discarded. Some critical fields, as the previous block hash field and the transactions inside the block, are also verified. Even the initial transactions, which are basically the offering of points to newcomers, are also verified. Notice that these transactions are necessary to make the system work properly.

Since the implementation is open source, the client can change the source code of the application and try to tamper the security mechanisms to bypass some of the verifications. Nonetheless, these changes are pointless because, unlike the Bitcoin protocol, all the messages are verified in a central entity before they are confirmed. First of all, all the initial reward transactions, generated to start and maintain the system, are all confirmed and inserted into a block with a client proof-of-work. Then, when a client tries to send a transaction to another, the message is broadcasted to all nodes and only the owner of the destination address (the private key corresponding to that public key) will accept the message. However, a bogus client can change the client implementation to accept multiple messages and request transactions that are not intended to it. The bogus client will receive the transaction and it will send it to the Bootstrap to be confirmed by someone else. Anyway, the real client will receive the transaction also, after being confirmed with the respective block, without the previous message.

A fake client can also try to forge and send a bogus block directly to the client. It is unlikely that a block will pass the verification, but if this occurs (for a perfect bogus block), this will unable the client to accept more blocks, because the next blocks will have an invalid previous block hash and no more blocks will pass the block verification. This client can be deceived and become useless for a while, but once it restarts the application, the correct chain will be synchronized again.

Another important aspect is the enforcement of block validation in the Bootstrap server. If an error occurs or for some reason a block is invalid, the unspent transactions are resent to the network, guaranteeing that no money is lost in the system.

Processing power is also an important issue, mainly when it comes to mobile devices and, as such, this factor was taken into account. The presence of the centralized server takes some of the computational burden from the devices, namely when it comes to find peers in the network. In order to keep the server running fast, it was decided to declare a data structure for storing references for the last 10000 confirmed transactions in memory, so as to quickly access them if necessary during validation. The data structure contains hash values and it was implemented so as to avoid to repeatedly read the blockchain.

Another concern was the memory consumption on the mobile devices. In the first trials, it was noticed that memory consumption was increasing smoothly but, after receiving a block, the proof-of-work quickly consumed almost the entire memory. The user was not able to use the device for anything else. After some research and experiments, it was concluded that there was no direct way to control the processing capacity of an application. The solution found to the problem was to put the thread to sleep for 1 millisecond between each nonce calculation. This method enables now the user to use its device, while proofs-of-work are being made.

Finally, one of the most important details that were taken into account was to ensure that the application was only used by mobile devices. Nowadays, a malicious user can easily run a mobile application in a desktop computer resorting, e.g., to an emulator and an Android OS. It is easy to understand why this is a problem, since a computer has higher processing power when compared to a smartphone or a tablet. The battery constraints are not applied to desktops either. Mobile devices would lose the mining process to the desktops. Since the smartphone has unique characteristics, an easy way to control this problem, would be to verify the device information where the application is running, like a valid Subscriber Identity Module (SIM) card serial number. The biggest problem is that there are good emulators, where the user can simulate device information as the SIM card serial number, International Mobile Equipment Identity (IMEI), International Mobile Subscriber Identity (IMSI), or even the Media Access Control (MAC) address [Luc13]. There are also many powerful mobile devices nowadays. As such, a potential solution to the problem consists in storing a list on the Bootstrap server containing the clients public keys and enforce a policy in which only a predefined number of points can be generated in a given time period. Since a client can generate infinite private addresses, a relation between the first address and the others needs to be created, impacting the privacy aspect.

4.6 Conclusions

The current chapter focuses on the presentation and description of the prototype developed during this masters programme. Some implementation details and decisions made during this phase are also discussed herein. These details summarize some of the most important features and way a functioning of the Bitpoints system. An analysis to the privacy, security and

performance issues was performed in the next-to-last section, along with the discussion of the measures taken to solve or minimize them. It was shown that many of those problems are solved by the Bitcoin mechanisms and concept. Nonetheless, some issues specific to the application scenario could only be tackled by leveraging the fact that a central server was implemented. The following chapter will discuss the testing of this prototype via a usability survey.

Chapter 5

Testing and Analysing the Prototype

5.1 Introduction

During the development of a user interface, the focus is on the potential end-users, for whom the system is to be designed to. According to Cooper [CRC07], even the most extraordinary application can fail if the end-users are not captivated by the product. During the development of a user interface, there are details that should always be taken into consideration, namely the appearance, ease of use, simplicity of the features and intuitiveness.

This chapter, and the results herein contained, reflect the importance of testing phase in the software development process. It describes the planning and delivery of a usability survey, which users had to anonymously answer while executing the developed prototype. The results of this survey are also presented in this chapter, along with some conclusions derived from them. The usability survey was performed to test the prototype and evaluate and assess its major flaws (e.g., interaction problems). It allowed, after analysing the results, the delineation of a guidelines for future work.

The next section will make a brief introduction to the survey. Then, section 5.3 focuses on the aspects analysed in this study and, in section 5.4, a few suggestions of the surveyed population are discussed. Finally, some conclusions are drawn.

5.2 Objectives

The survey elaborated for the purpose of testing the usability of the prototype can be consulted in Appendix A. It was delivered to a group of potential end-users from several professional areas. This usability test was elaborated using Google forms and disseminated via e-mail, social networks or even personally, by sending its Uniform Resource Locator (URL) to the users. The survey asked the users to install and test the prototype developed in this masters programme.

The survey is composed by four specific sets of questions: (i) questions to obtain some information about the respondent and its knowledge; (ii) questions to assess the mobile device used in the experiment; (iii) questions to evaluate the functioning of the prototype; and (iv) a few questions of personal and optional answer. The questions mentioned in last, were elaborated with the main purpose of providing a means for the respondent to include suggestions or criticize the developed application. This usability test was performed in order to evaluate the following

aspects:

- The culture of the respondents regarding the *Bitcoin* subject;
- Ease of downloading and installing the application;
- Ease of interacting with the user interface of the prototype;
- Ease of generating a new personal address;
- Ease of adding a new public address;
- Ease of performing a transaction to a public address;
- Evaluation of the understanding and ease of use of the provided features;
- Evaluation of the ease of navigation in the application and the understanding of its structure.

The survey assumed that an Android device with an active Internet connection was used. A QR code was initially provided to the user, so as to help him or her in the tasks of downloading and installing the application. The fact that the survey can be answered by people of different areas (not only in computer science) was taken into consideration. As such, all of the procedures that were asked to the respondents were carefully explained textually along with the questions. Because there are several versions of the Android OS (which could be problematic), the explanation was made so as to applied to the general case. The survey was written in Portuguese, because the audience was mostly constituted by Portuguese people. Below, the translation of part of the aforementioned explanation is included for exemplification purposes:

- After downloading the application, it will be stored in the `Downloads` folder, inside the `My Files` folder;
- To install the application, you can execute the `.apk` file and select the Package Installer option;
- If the application installation fails, you should go to Device Settings, then Security option, and make sure that the Unknown Sources checkbox is enabled.

Once the application is installed, the user has to follow a tutorial that helps understanding the prototype. After completing the tutorial and having the application synchronized with the central entity, an amount of 500 BTPs were automatically credited and confirmed into the user wallet for testing purposes. After that, the user was requested to generate a new personal

address and to add a public address to its contacts list. Later on, the respondent was requested to perform a transaction to an address given and to observe the changes in the transactions list, in order to have the most complete experience with the interface and all of its functionalities.

5.3 Survey Results

The survey was performed by a total of 34 anonymous respondents. As can be seen in Figure 5.1, the participants *ages* are comprised between 21 and 55 years, which provides a good distribution for this variable. The right chart in Figure 5.1 shows that from the total population, 23% were *female*, and 73% were *male* participants.

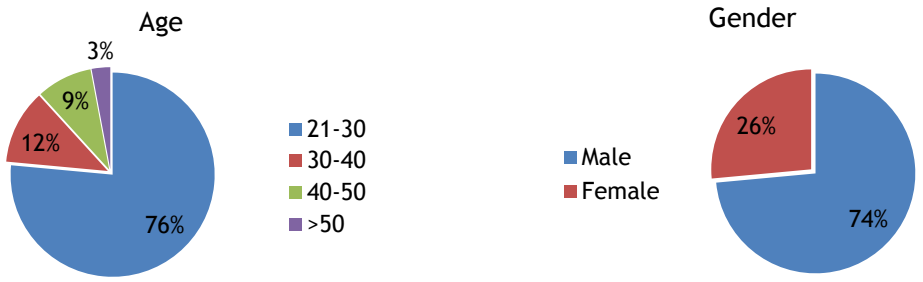


Figure 5.1: Pie charts compiling the results concerning age and gender of the participants.

The survey was distributed with the main concern of covering different *professional areas*, so as to obtain a potentially richer and statistically significant feedback. The several *professional areas* covered by the survey are represented in the left chart of Figure 5.2, with the largest slice referring to the computer engineering area. The chart on the right presents the *academic qualifications* of the population, with the biggest slice referring to individuals with a degree.

As previously mentioned, there are many versioning issues in Android smartphones, specially due to the OS fragmentation [Hil13]. Since its beta release in 2007, more than three different versions are released per year [Wes12]. This fact can possibly lead to programming inconsistencies, since it is hard to test and maintain applications on all available devices and OSs. Since Android OS is open source, and there are a broad number of manufacturers customizing their own versions, or installing alternative Android versions, like for example, the *CyanogenMod*, the manufacturers or the model of the devices are often related with incompatibility issues. In order to detect and debug technical issues or version incompatibilities, an effort was also made to distribute the survey to persons with different Android *versions* and mobile devices. The chart on the left side of Figure 5.3 presents the *manufacturers* of mobile devices that users claimed to use during the testing of the prototype. The chart on the right depicts the Android *versions* running on those devices.

The charts emphasize that different brands were used, with the largest slice belonging to *Samsung* and *Sony*. It should be noticed, however, that for each brand, several models were implicit,

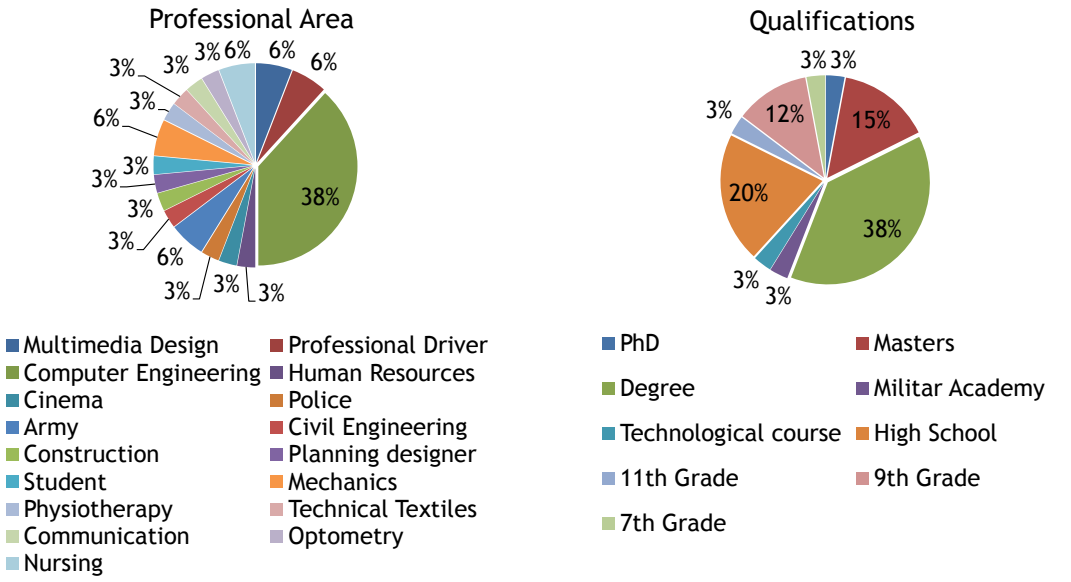


Figure 5.2: Pie charts compiling the results concerning professional areas and academic qualifications of the participants.

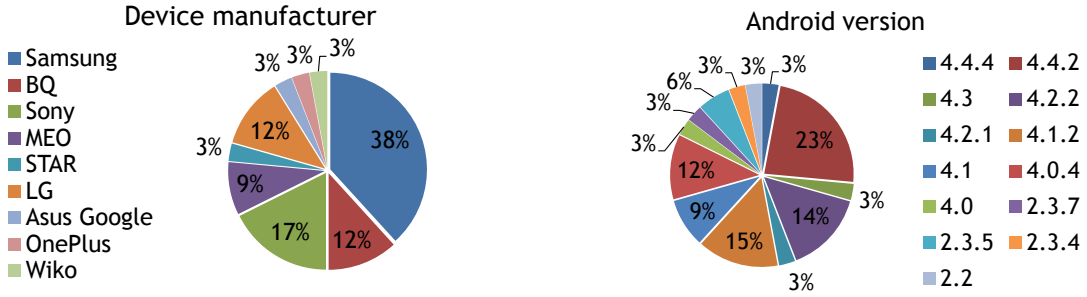


Figure 5.3: Pie charts compiling the results concerning device manufacturers and Android versions.

and thus not represented in the chart. The smartphone models used by the target population were the several *Samsung Galaxy* models (*S4*, *S*, *S3 Mini*, *Mini*, *Fresh*, *Star*, *Note II* and *Grand Neo DUOS*); the *Sony Xperia Go* and *Tipo*; *LG MAXIMO L3 II*, *MAXIMO L5 II*, *L40* and *L80*. *BQ* models are also used, namely *Aquaris E5* and *Aquaris 5.7*. Finally, a few devices like *One* (*One-Plus*), *N9000 Plus* (*STAR*), *Ozzy* (*Wiko*), *MEO A7* and *Meo A15* were used as well. In order to provide a different testing environment to the developed application, two people were asked to answer the survey with two different tablets: *Samsung SM-T110* and *Asus Google Nexus 7*. As can be concluded from the chart on the right of Figure 5.3, the Android versions running on the aforementioned devices ranged between 2.2 and 4.4.4, with the 4.4.2 and 4.1.2 being the most representative ones.

It is important to mention that the installation of the application failed in one case only, namely when a *Sony Xperia Arc S* with Android 2.3.4 was being used. It failed due to a compatibility error. In this case, another device was provided to the respondent so that the survey could be answered.

According to Figure 5.4, half of the population was already familiar with the Bitcoin concept. Even despite the aforementioned fact, 85% of the population considered that a non-governed currency was interesting (chart on right). In order to prevent random answers to the *yes or no* question (*Do you find the notion of having a non-governed currency interesting?*), a text field for justifying the answer was included.

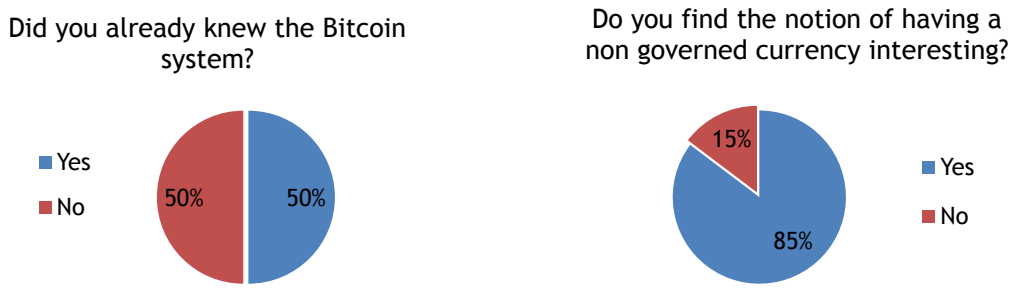


Figure 5.4: Pie charts compiling the results concerning question number 7 and 8 of the survey.

Most of the justifications accompanying the answers to question 8 concern protection and privacy, and the perspective of no one being able to eavesdrop or control what people do with their own money. The possibility of having a currency which is controlled in a decentralized manner is another reason included in the justifications. For most of the respondents, the fact that the crypto-currency is managed by the majority of well-intentioned users of the network, along with the fact that the ledger is public and it is possible to anonymously buy and sell services, seems really interesting. Some people are of the opinion that crypto-currencies could ease the payment of certain services or products mostly because it is not influenced by the government in any way. According to them, worldwide currency conversion would not be needed and some taxes can be avoided. Some respondents refer that communities are limited to the traditional coin and they may not know that a non-governed currency can easily become international and used in endless different services. Even though part of the population stated that they find this system interesting, they pointed some flaws too, namely that a decentralized coin can cause political disagreements between different governments (who is in favour and who is against of non-governed currency). There were also some answers disagreeing with the interest of digital currencies, mostly backed up by the fear of being robbed by hackers.

Subsequently, a few questions concerning the prototype and regarding its easiness of use, were made. Figure 5.5 presents the results concerning the success of the installation and execution of the application. Most of the respondents (59%) stated that the application was *very easy* to download, install and execute, while on the remaining population, 32% considered it *easy* to install. The remaining 9% considered that the application is *difficult* to setup and execute. However, it should be noticed that the 9% of the population is mainly constituted by older individuals or with less academic qualifications.

After installing the application and going through the tutorial, 76% of the respondents felt comfortable to use it, while 21% selected the *maybe* answer, as can be seen on the right chart of Figure 5.5.

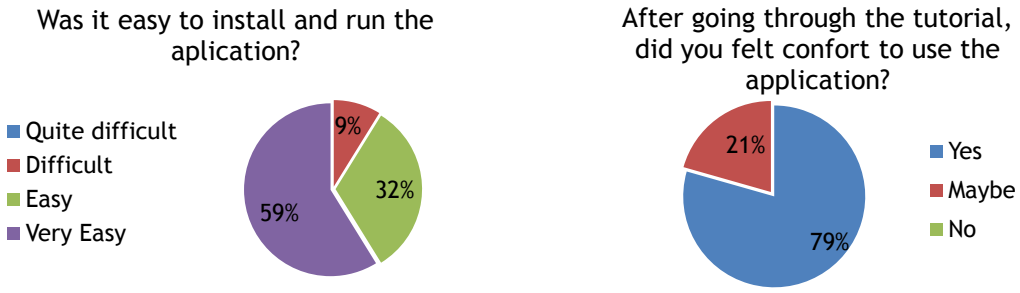


Figure 5.5: Pie charts compiling the results concerning question number 10 and 11 of the survey.

Once the application is running, the user is prompted to explore it, generate a private address and add a new public address by reading a QR code. The charts that evaluate the easiness of adding those addresses are represented in Figure 5.6. The chart on the left shows that 79% of the respondents answered that it was *very easy* to generate a new private address, and the remaining 21% answered that it was *easy*. In the chart on the right, the biggest slice (71%) corresponds to the participants that found it to be *very easy* to add a new public address. 26% stated that it was *easy*, while only 3% found it *difficult*.

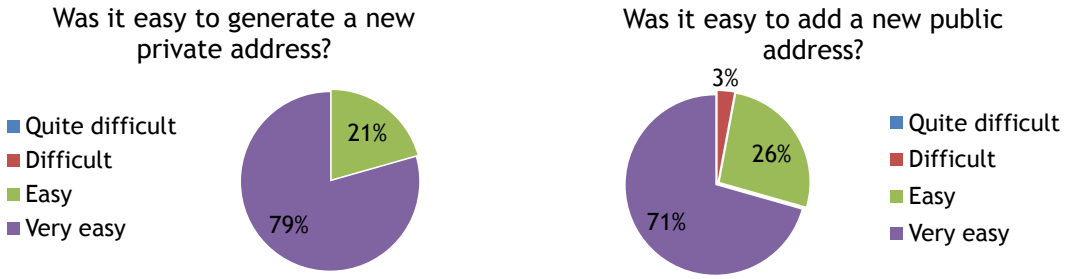


Figure 5.6: Pie charts compiling the results concerning question number 12 and 13 of the survey.

The participants were then invited to reflect on the two possibilities to add public addresses (manually or by reading a QR code) and then inquired of which option they prefer the most. The entire population claims that the best option is to add a public address by reading QR code) (shown on the left chart of Figure 5.7). Such answers were already expected, since each public address consists of a long string, which is difficult to insert manually or memorize. The chart on the right of Figure 5.7 concerns the answers related with the ease of performing a transaction using the application. Everyone agreed that it was easy.

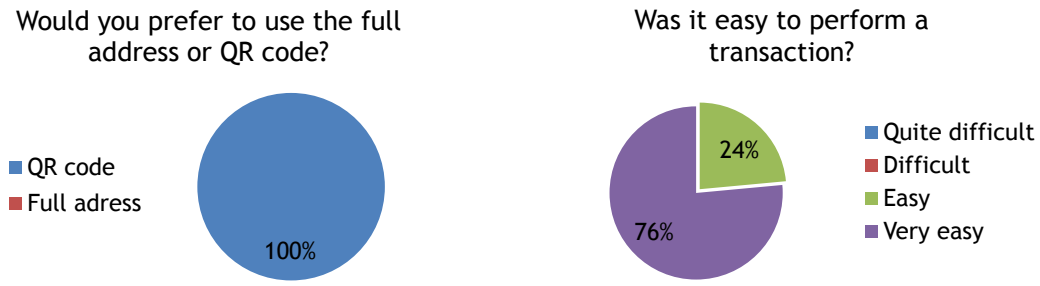


Figure 5.7: Pie charts compiling the results concerning question number 14 and 15 of the survey.

The next couple of charts (Figure 5.8) suggest that the prototype was easy to use and its functioning was easy to understand. 65% of the respondents found the application *very easy* to understand and the remaining 35% found it *easy*. Regarding the users opinion about the prototype navigation and structure, 74% found it *very easy* and 26% stated that it was *easy*.

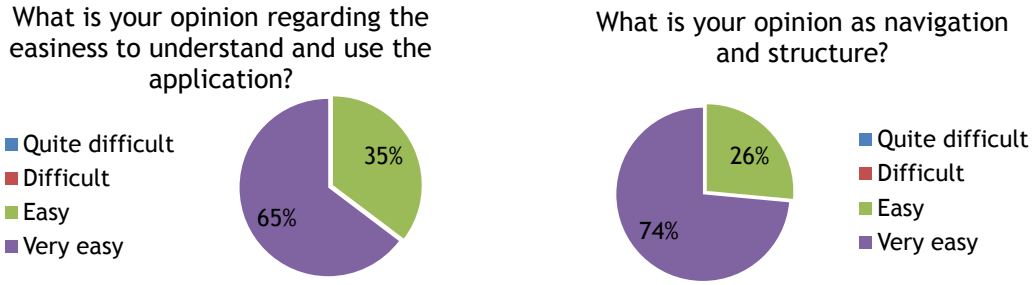


Figure 5.8: Pie charts compiling the results concerning question number 16 and 17 of the survey.

The charts in Figure 5.9 reflect the opinion of the population regarding the need of possessing some technical knowledge to use the application correctly (left chart), and whether or not the initial tutorial was helpful (right chart).

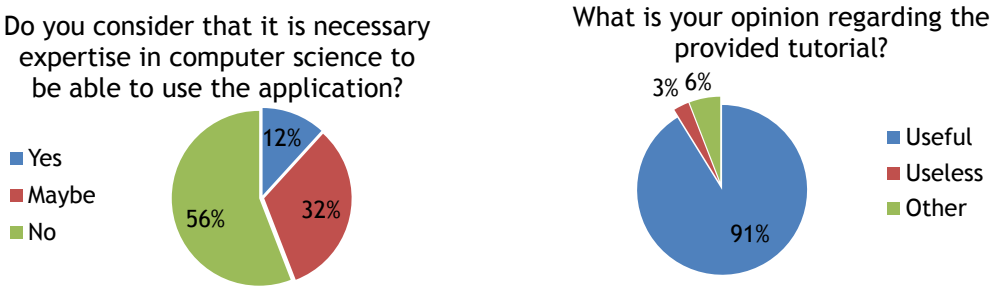


Figure 5.9: Pie charts compiling the results concerning question number 18 and 19 of the survey.

As can be seen on the left chart of Figure 5.9, the population is divided when it comes to the necessity of possessing some skills to operate the application. 56% claim that the user does not need to possess any type of expertise in computer science, while 32% state that the user might need some. 12% believe that some expertise is necessary. Regarding the initial tutorial, most

of the participants (91%) stated that it was *useful*. From the remaining 9%, 6% chose to provide a little more feedback to the author, by adding some thoughts in a dedicated text field, while the remaining 3% defended that the initial tutorial was *useless*.

After the questions concerning the application, the survey included three general opinion questions, whose results are depicted in Figures 5.10 and 5.11. These questions try to evaluate the attractiveness of having a loyalty card in a mobile device (chart on the left side of Figure 5.10), if the exchanging of points with other users would be useful or not (chart on the right side of Figure 5.10), and if the participant would prefer using this loyalty card in detriment of currently existing ones (Figure 5.11). Each one of the three aforementioned questions asks the participant to justify the answer in a text field.

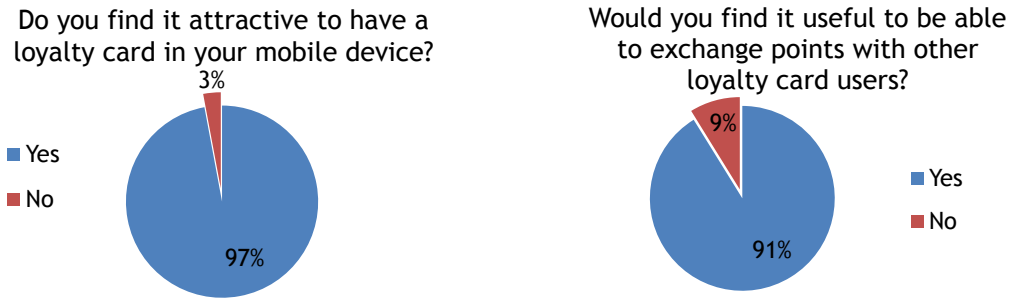


Figure 5.10: Pie charts compiling the results concerning question number 20 and 22 of the survey.

A few conclusions can be drawn from this part of the survey. Regarding the question associated with the chart on the left side of Figure 5.10, users generally say that it is very attractive to use a virtual loyalty card for several reasons, namely convenience of payments, and ease of use and performing transactions. A lot of participants also consider a virtual loyalty card better than the traditional plastic cards, specially due to the fact that usually, they have too many cards in their pockets, which could be avoided with a large adoption of these systems. It was also argued that it is easier for a person to forget a card somewhere than their personal phone. Another point, which was also mentioned by a few participants is that, with a virtual loyalty card, users can have easier access to the card information, and better ways of paying for services and products. Only one person, out of the entire population, claimed that virtual loyalty cards are not useful or attractive, because he or she feels that it might be hard to get used to these new technologies.

The chart on the right side of Figure 5.10 show that 91% of the respondents are of the opinion that it would be useful to be able to exchange points with other users. The justification for this result is maybe related with the possibility of the users to help family or friends to accumulate the necessary points to pay for some particular services or products, or just transfer points that they do not use. A certain number of users think that if they own these points, then they have the right to transfer them to others, since sometimes users do not care about the accumulated

points and they want, in occasional situations, to borrow or even trade them for real money.

The chart in Figure 5.11 compiles the results concerning the preferences of the population about the usage of an electronic loyalty card in detriment of physical ones, in case a big brand was promoting it. The biggest slice of the chart (about 76%) corresponds to positive answers. Only one respondent answered negatively, and the remaining respondents answered with *no opinion*.

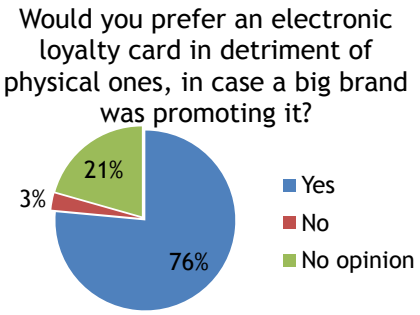


Figure 5.11: Pie charts compiling the results concerning question number 24 of the survey.

5.4 Interviewed Suggestions

One last and optional question was included with the main objective of gathering opinions and tips for improving the application in the future and to understand what the users want and need. After the careful analysis of the suggestions provided by the population, four main suggestions were identified:

1. Improve the initial tutorial, namely by adding more details and inserting the concept of crypto-currency and its connection to Bitpoints;

2. Improving the layout (enhance the interface colors and buttons);

3. Add some functionalities like, for example, associate a photo to an address;

4. Some messages to the user need polishing, so as the user interface (for example, fade the background, design a more interesting logo, etc.), and some information regarding the list of transactions must be included, namely the status, the block where it is inserted and the digital signature.

5.5 Conclusions

This chapter discussed the results of an electronic survey used to evaluate the prototype developed along this masters. The survey was delivered to a population constituted of 34 individuals, who answered 25 questions, while handling the application on their smartphones and tablets.

The participants were from several professional areas and had different backgrounds, with ages ranging from 21 and 55. Different mobile devices running several versions of the Android OS were used in this experiment, providing a good basis to obtain statistically meaningful results.

In general, most of the users found the prototype easy to handle and understand, although a big part of them did not knew Bitcoin and had never used similar crypto-currencies. Some respondents gave suggestions to improve the application aesthetics, which will be taken into account in the future.

The installation of the application was successful for all devices except one, due to an incompatibility error. Even though it was not possible to debug the problem for the specific equipment, it may be caused by the inclusion of a library to handle JSON data. Officially, there is no information regarding this incompatibility but there are some forums were issues in older Android OS versions or even with some device models were reported. This problem was left as a topic for future work.

This survey allows to induce that people would feel comfortable with this application and accept the concept on which it is based on, probably preferring a system similar to the proposed one, over the traditional ones. This seems very promising in terms of the objectives of this masters programme.

Chapter 6

Conclusions and Future Work

This chapter presents a synthesis of the main achievements of this masters programme and briefly describes several directions for future work. It is organized in two sections. Section 6.1, consists on presenting the main conclusions of the work and the suggestion for future research directions are described in section 6.2.

6.1 Main Conclusions

E-commerce has experienced a significant growth in the last decades and, until recently, it relied mostly on institutions that control all the involved transactions. However, with the evolution of cryptographic coins, alternatives to these centralized institutions are now viable. Some of these alternatives are self-contained and non-governed, being exclusively controlled by their users. Bitcoin is one of those crypto-currencies that offers secure exchange of virtual money with no need of central entity. It provides the properties of transaction irreversibility and verification. Although the system provides a public history that allows exploring all the transaction performed to date, it is fully anonymous and provides methods to ensure that there is no way to link users to transfers. Part of the research work described in this dissertation tackled the functioning of the aforementioned protocol (refer Chapter 2), as well as existing implementations of Bitcoin and virtual loyalty cards. The objective was to build a virtual loyalty card based on this already established and secure protocol.

It was shown that there are several available virtual loyalty cards available in the market nowadays, though most of them are limited or too simple. For example, they do not allow points to be exchanged between users and, as can be concluded from the survey described in Chapter 5, users of such cards agree in that this functionality would indeed be useful. Existing implementations are mostly focused on provide functionalities to connect Internet, consult promotions, products or balance, and generate bar codes to spend them. The application of crypto-currencies, namely Bitcoin, in the implementation of loyalty cards also has its limitations. One of the main advantages of this protocol could actually constitute a disadvantage in such systems, mostly because, in some specific scenarios, there is the need to centrally control the currency and the mining (rewards) process. The system described herein aims to merge some of the best features of both systems to deliver an improved electronic loyalty card.

As mentioned in Chapter 3, Bitpoints is a system designed, engineered and implemented from

scratch, inspired in Bitcoin, but with controlling entity. The system continues to maintain the privacy of transactions and users, but the existence of the central entity enables configuring some of the aspects of the system and controlling some of its functioning, namely rewarding.

The development of the prototype delivered with this dissertation proved to be a tougher challenge than initially thought. The choice to make an implementation from scratch was the factor that contributed to this challenge the most. The result is, nonetheless, a fully working prototype of a virtual loyalty card system for mobile devices. It allows the secure and anonymous exchange of points and supports rewards based on mining (the entity may also reward public addresses easily by transferring money). A central entity may control this system using a back-end on a bootstrap server. In contrast to other virtual loyalty cards systems, it works properly over a non-secure network, since the definition of the protocol guarantees its security without requiring, e.g., the encryption of the communication channels. Every transaction or block is beforehand digitally signed, and it is verified (by its digital signature and hash value) upon their reception, which enables discarding tampered messages.

Another difference to conventional loyalty systems is that Bitpoints includes mechanisms for guaranteeing the privacy of the user. For example, a user may generate as many public keys as he or she wants. The current version of the mobile application does not include any method to delete private addresses, to prevent the user from losing the points associated with that address. Bitpoints uses ECC mechanisms (refer Chapter 4) to generate the client key-pairs and digitally signing the messages, which are nowadays considered to be safer than RSA related primitives.

The introduction of a central entity (Bootstrap server) in the devised system was necessary for the specific application scenario. Bitpoints is more flexible than most physical loyalty cards. Moreover, contrarily to what happens with Bitcoin, this server side application prevents that two separate branches emerge from blocks arrived at the same time and with the same timestamp value. This occurs in Bitcoin because there are too many nodes broadcasting confirmed blocks at the same time. In Bitpoints, only the Bootstrap server broadcasts confirmed blocks to the network. Given that, when two distinct blocks arrive to the server at the same time and with the same timestamp, the Bootstrap automatically randomly selects one of them before broadcasting it. The invalid block is discarded from the system and any unconfirmed transactions are returned to the network.

Overall, most of the respondents of the survey (refer Chapter 5), found the prototype easy to use and understand, even though most of them had no prior expertise in such systems, namely Bitcoin and virtual loyalty cards. The respondents found the idea behind Bitpoints to be interesting and most of them would use this system as a loyalty card, although there were complaints about the aesthetics of the application. The survey conducted in the final phase of this masters

programme was relevant and very helpful regarding the improvement of Bitpoints system. It allowed gathering some of the changes that need to be made to improve the system.

Table 6.1 ends up this section with a comparison between the different virtual systems mentioned so far and Bitpoints.

Table 6.1: Comparison between Bitcoin, existing Virtual Loyalty Cards and Bitpoints.

Feature	Bitcoin	Existing Virtual Loyalty Cards	Bitpoints
Setup	Days	N/A	Secs
Unit Cap	≈21 Million	N/A	Controlled
Allows Exchange	Yes	No	Yes
Mining Reward	25 BTC	N/A	Controlled
Excessive Mining	Yes	N/A	No
Controlling Entity	No	Yes	Yes
Load on Device	Yes	No	No
Crypto Algorithms	ECC & SHA-256	No	ECC & SHA-256
Local Security	No	Yes	No
Network Security	No	No	No
Wallet Backup	Yes	No	No
Generation Blocks	Limited	N/A	Controlled

The developed prototype implements some of the best functionalities of the other two systems from the perspective of its specific application scenario, but the table is not presenting the general advantages or disadvantages of the systems. The aspects under comparison in the table can be further discussed as follows:

Setup - Typically, existing electronic loyalty cards do not have a synchronizing process. In the case of Bitcoin, synchronization may take several days to be concluded, because it operates worldwide. As for Bitpoints, in small and specific applications scenarios (e.g., stores and supermarkets), synchronization will be faster, since the number of users and the information will be smaller compared to the original Bitcoin;

Unit Cap - The unit cap aspect is again not applicable to virtual loyalty cards. Bitcoin has a limit of coins (approximately 21 Millions) by conception. On the other hand, Bitpoints does not define a limit by default and enables the central entity to control the reward and, as such, the amount of maximum points;

Allows Exchange - Both Bitcoin and Bitpoints enable the exchange of coins/points between users, but this functionality is not typically supported by existing virtual loyalty cards;

Mining Reward - Mining rewards are typical of systems supporting transactions between users.

As such, virtual loyalty cards do not have mining rewards. At the time of writing this dissertation, Bitcoin was providing an reward of 25 BTC. This value will halve every 210000 blocks. The Bitpoints rewards are controlled, and the administrator can change their value along the way;

Excessive Mining - A huge concern about e-currencies is the one of excessive mining, since the greater is the processing power of the device, the greater is the probability of confirming a block. Therefore, there are *pools* [Coi14] where a large number of users combine efforts and processing power in order to be more likely to confirm blocks. Usually, the rewards earned within the *pool* are distributed amongst its members. This implies that the most powerful devices are always more likely to earn rewards than the others, and thus the weaker ones will win fewer times. In order to decrease the gap between more powerful devices and the weaker ones, mechanisms to limit the rewards per device were implemented in Bitpoints. In this case, each device can only get 1 reward every 10 minutes. This time is configurable by the controlling entity;

Controlling Entity - As previously mentioned, similarly to existing virtual loyalty cards and unlike Bitcoin protocol, a central entity was implemented in Bitpoints, which allows control and configuration of the system;

Load on Device - Since Bitcoin does not have central entity, the computational load falls on every connected nodes. This is not true for Bitpoints, since the Bootstrap server is responsible for part of the load of the system, against the inevitable exception of proof-of-work;

Crypto Algorithms - Unlike the existing virtual loyalty cards, Bitcoin and Bitpoints use cryptographic algorithms in its definition. Primitives from ECC are used to compute digital signatures and the SHA-256 is used to compute hash values;

Local Security - Most of the Bitcoin wallets and the Bitpoints mobile application do not have local security mechanisms. However, this is an important feature that will have to be handled in the future (see below). Some physical and virtual loyalty cards do have security mechanisms. E.g., the *Continente* virtual loyalty card is protected by a Personal Identification Number (PIN) code that identifies its owner;

Network Security - As already described, Bitcoin and Bitpoints do not require mechanisms to protect communications over computer networks. All transactions are public and there is no risk associated with the manipulation of messages. However, the same does not apply to some of the existing virtual loyalty cards, threatening the privacy of users, since there are no methods to protect the information in case of interception [Nak09];

Wallet Backup - One of the most important and extra features that can be implemented in

e-currency systems concern backup and restore functionalities. Some implementations of Bitcoin provide this feature, though it is not described in the seminal work. Bitpoints should support such feature in the future;

Generation Blocks - The generation blocks are those that represent the beginning of the coin or, within the scope of this dissertation, of the points. These blocks, with a specific amount of coins/points, are generated and confirmed manually. In Bitcoin, these blocks were generated at the same time in 2009, and therefore there is no possibility to generate new blocks with the desired values. The developed prototype is a system that, controlled by the central entity, provides the functionality to generate new blocks to the first N clients that connect to network. This value is configurable also. This feature is not applicable in the context of the traditional virtual loyalty cards.

6.2 Directions for Future Work

This dissertation and the project it refers to aimed to development of a prototype of an application for mobile devices that implements a virtual loyalty card based on the Bitcoin protocol. The devised system is perhaps just the basis for several different research and development directions for future work. Below, some of those potential directions are briefly discussed.

Malware constitutes one of the most popular topics within the scope of mobile devices and platforms, namely Android OS. The number of samples is growing at an exponential rate, emphasizing that it is necessary to implement and further enhance security mechanisms, particularly for mobile applications where security and privacy are critical factors. According to [Kel14], from 2012 to 2013, Android malware rose from 238 threats to 804 new threats (97%), and the combined total of Apple iOS, BlackBerry OS and Microsoft Windows Phone was 0%. The remaining 3% came from Symbian OS as can be seen in Figure 6.1. It is necessary to device anti-virus and firewalls for these systems and also to make the code less prone to vulnerabilities. It is also necessary to assess if the implementation of the cryptographic primitives are not prone to side channel attacks and guarantee that the implementations used are secure regarding that aspect.

Some of the aspects that should be addressed in the near future concern the protection of the files representing the local wallet. The suggestion is to use a state-of-the-art symmetric key cipher to encrypt the aforementioned files. The encryption key should be derived from the combination of a large PIN or password with a nonce. The nonce may change each time the file is decrypted. An access control mechanism to start the mobile application can also be put in place using one of those credentials. Such mechanisms would protect the files against attackers with access to the mobile device.

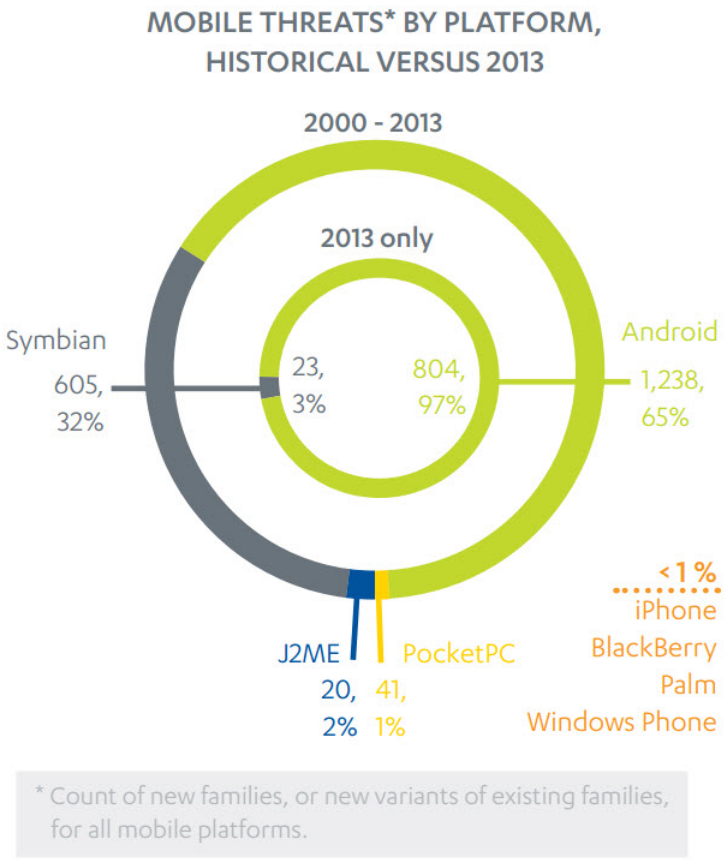


Figure 6.1: Mobile threats by platform.

The backup mechanism is another feature that should be implemented in the near future. The idea is to provide the means to create a compressed and protected backup of the private keys and sending it to remote location for storage and later retrieval, if necessary. The mechanism should guarantee that the archive can only be accessed by the rightful owner. Cryptography primitives should be useful once again. Once the Bitpoints wallet is locally secured, mechanisms to protect the communications over the network can be introduced. These mechanisms are not really necessary for protection of the system per se, but they should be at least useful for protecting the user from profiling attempts based on traffic monitoring.

A current concern about mobile devices is definitely its memory capacity. Although the memory of this type of devices is increasing every day, this aspect should be subject to further analysis in the future, mostly because the blockchain grows with time, sometimes erratically depending on the popularity of the protocol. It not handled correctly it may end up consuming a large slice of the device memory. A suggestion for fixing this problem is to implement a mechanism that stores only the references to transactions concerning the wallet of the user, and then perform transactions verification by using the Merkle trees.

Another feature that was not implemented due to lack of time concerns the means to allow offline transactions. Bitcoin implementations and existing virtual loyalty cards require an In-

ternet connection to perform any operations, for example, transactions or balance consulting. The system developed along this masters programme can be improved by adding some methods to save the current state of the wallet in order to allow the user to use it without an Internet connection and put all transactions in a waiting state until a connection is available. At this time the system will synchronize and verify the pendent transactions, and if they are valid, it will send them.

A subject that would be quite interesting to investigate concerns the usage of Bitpoints for gamification purposes. There could be rewards for using the application, and thereby increase the interest of the users in the system, or there could be rewards for challenges regarding products. Since the system is totally electronic and enables crediting points on the fly (e.g., by reading QR codes), setting up such challenges would be easy for the owners of the system. This type of mechanisms can be also used to direct advertisements to users or collect data from clients in order to improve the system services.

The impact of adding a central entity to a system inspired in Bitcoin needs to be further investigated also. The fact is that part of the success of Bitcoin lies in its fully decentralized mode of operation, and the existence of a central entity may be introducing new issues.

Finally, it should be mentioned that the interface should be improved before deployment. Such line of work is also a direct result of the survey conducted along the programme.

Bibliography

[And] Android. Google Mobile OS. http://www.android.com/. Last access: March 3, 2014. 25

[Boc13] Thomas Bocek. A P2P-based high performance key-value pair storage library. http://tomp2p.net/, April 2013. Last access: October 27, 2013. xiii, 41

[Bom13] Satish Bommisetty. Elliptic Curve Cryptography: A Case for Mobile Encryption. http://www.securitylearn.net/2014/02/28/elliptic-curve-cryptography-a-case-for-mobile-encryption/, February 2013. Last access: September 1, 2014. xxiii, 48

[But12] Vitalik Buterin. Bitcoin Wallet Reviews – Ease Of Use And Security. http://bitcoinmagazine.com/327/bitcoin-wallet-options/, March 2012. Last access: December 20, 2013. xii, 19

[CC] Douglas Crockford and Zach Carter. JSONLint: The JSON Validator. http://www.jsonlint.com/. Last access: October 14, 2013. xiii, 26

[Coi14] CoinDesk. What are Bitcoin Mining Pools? http://www.coindesk.com/information/get-started-mining-pools/, March 2014. Last access: September 21, 2014. 72

[CRC07] A. Cooper, R. Reimann, and D. Cronin. *About Face 3 - The Essentials of Interaction Design*. Wiley Publishing, Inc., illustrated edition edition, 2007. 59

[Cry14] CryptoEscudo. Uma moeda para o século XXI. http://cryptoescudo.org/, 2014. Last access: September 8, 2014. xii, 22

[Cur13] Currito. Currito - The Burrito, Evolved. http://currito.com/, 2013. Last access: May 9, 2014. xii, 19

[Dai98] Wei Dai. b-money. http://www.weidai.com/bmoney.txt, November 1998. Last access: October 19, 2013. 2

[Dar14] James K. Darlington. *The Future of Bitcoin: Mapping the Globgal Adoption orf Worlds's Largest Cryptocurrency Through Benefit Analysis*. PhD thesis, University of Tennessee, Knoxville, 2014. 1, 2

[Dev] Android Developers. Using Hardware Devices. http://developer.android.com/tools/device.html. Last access: October 19, 2013. xii, 26

[dJ] Jos de Jong. JSON Editor Online. http://jsoneditoronline.org/. Last access: October 15, 2013. xiii, 26

[Doc13] Oracle Java SE Documentation. Java Cryptography Architecture Oracle Providers Documentation for Java Platform Standard Edition 7. http://docs.oracle.com/javase/7/docs/technotes/guides/security/SunProviders.html, 2013. Last access: October 11, 2013. 48

[Dri13] Scott Driscoll. How Bitcoin Works Under the Hood. http://www.imponderablethings.com/2013/07/how-bitcoin-works-under-hood.html, July 2013. Last access: December 18, 2013. xii, 10

[Eis] Christian Eisenberg. Crypto Coins List. http://www.cryptocoincharts.info/coins/info. Last access: January 8, 2014. xii, 22

[Hil13] Jerry Hildenbrand. Inside the different Android Versions. http://www.androidcentral.com/android-versions, June 2013. Last access: September 12, 2014. 61

[Hos] Bitcoin Web Hosting. History of Bitcoin - The world's first decentralized currency. http://historyofbitcoin.org/. Last access: March 18, 2014. 2

[Hoy09] Jeff Hoye. A P2P-based high performance key-value pair storage library. http://www.freepastry.org/FreePastry/, March 2009. Last access: October 27, 2013. xiii, 41

[Jin13] David Jinkins. Bitcoin vs Local Currency. http://veryshuai.no-ip.org/blog/index.php/bitcoin-vs-local-currency/, April 2013. Last access: March 18, 2014. 1, 2

[Kel14] Gordon Kelly. 97% of Mobile Malware is on Android. This is the Way to Stay Safe. http://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/, March 2014. Last access: September 11, 2014. 73

[Lan13] Knut Landvik. How to work with Bitcoins. https://blog.ipredator.se/howto/bitcoins/howto-work-with-bitcoins.html, 2013. Last access: May 10, 2014. 20

[Len13] Jason Lengstorf. JSON: What It Is, How It Works, How to Use It. http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it/, 2013. Last access: October 14, 2013. 26

[Luc13] Andrei Luca. Android emulator patch for configurable IMEI, IMSI and SIM card serial number. http://www.blog.codepainters.com/2010/11/20/

android-emulator-patch-for-configurable-imei-imsi-and-sim-card-serial-number/, June 2013. Last access: September 2, 2014. 57

[Mer80] Ralph C. Merkle. Protocols for Public Key Cryptosystems. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 14-16, 1980*, pages 122–134, 1980. xii, 9

[Mic13] Michal. Mobile Market Share and Statistics for IOS, Android and Other. http://nomtek.com/mobile-market-share-and-statistics-for-ios-android-and-other/, April 2013. Last access: June 7, 2014. xxi, 42

[Nak09] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. Available from: http://www.bitcoin.org/bitcoin.pdf. xii, 1, 5, 11, 72

[Nex14] Virtual Next. Simple loyalty payment solution...that's not an app at all! http://virtualnext.com/, 2014. Last access: May 9, 2014. xii, 19

[Ora13] Oracle. Java.net - The Source for Java Technology Collaboration. https://java.net/projects/jxta, 2013. Last access: October 19, 2013. xiii, 41

[Ora14] Oracle. Java Cryptography Architecture Oracle Providers Documentation for Java Platform Standard Edition 7. http://docs.oracle.com/javase/7/docs/technotes/guides/security/SunProviders.html/, 2014. Last access: July 14, 2014. 48

[oST14] National Institute of Standards and Technology. Secure Hashing. http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html, July 2014. Last access: August 26, 2014. 9

[Pla14] Google Play. Barcode Scanner. https://play.google.com/store/apps/details?id=com.google.zxing.client.android&hl=pt_PT, May 2014. Last access: September 22, 2014. 37

[PP10] C. Paar and J. Pelzl. *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer-Verlag New York Inc, 2010. xii, 9

[RB14] Angela ROGOJANU and Liana Badea. The issue of competing currencies. Case study – Bitcoin. *Theoretical and Applied Economics*, XVIII(2014)(1(590)):103–114, 2014. Available from: http://EconPapers.repec.org/RePEc:agr:journl:v:1(590):y:2014:i:1(590):p:103-114. 8

[Sev03] Rosane Severo. O estigma do excesso de cartões de fidelidade. http://www.consultores.com.br/artigos.asp?cod_artigo=82, August 2003. Last access: April

3, 2014. xii, 8

[Tho13] Jessica Thornsby. Connecting Your Android Device to Eclipse. http://www.developer.com/ws/android/connecting-your-android-device-to-eclipse.html, February 2013. Last access: October 7, 2013. xii, 26

[Vau10] Serge Vaudenay. *A Classical Introduction to Cryptography: Applications for Communications Security*. Springer Publishing Company, Incorporated, 1st edition, 2010. xii, 9

[Wag13] Kurt Wagner. World's First Bitcoin ATM. http://mashable.com/2013/10/30/bitcoin-atm-2/, October 2013. Last access: December 2, 2013. 2

[Wes12] Luke Westaway. Android updates guide: All the features of every version. http://www.cnet.com/news/android-updates-guide-all-the-features-of-every-version/, July 2012. Last access: September 23, 2014. 61

Appendix A

Survey of Usability Test

Bitpoints Teste de Usabilidade

Estima-se que o tempo de preenchimento do formulário não será superior a 20 minutos.

O presente inquérito surge no âmbito de uma Dissertação de Mestrado em Eng.ª Informática da Universidade da Beira Interior e tem como objectivo avaliar a usabilidade de uma aplicação desenvolvida durante o programa deste mestrado.

A aplicação é uma interface para um sistema de pontos virtuais, emulando um cartão de fidelização, que possui como principais vantagens, para além das habituais a um desses cartões, uma troca segura de pontos entre diferentes utilizadores e possibilidade de minar pontos, isto é, ganhar pontos consoante o uso da aplicação.

Este inquérito pressupõe a disponibilidade de um dispositivo móvel Android com ligação à Internet e com um leitor de QR Codes instalado (recomenda-se o Barcode Scanner: https://play.google.com/store/apps/details?id=com.google.zxing.client.android&hl=pt_PT). Ser-lhe-á, por isso, pedido que instale uma aplicação no seu smartphone.

O download, instalação da aplicação e uma operação de transação deverá ser realizada a partir de QR Codes fornecidos.
Não há respostas certas ou erradas relativamente a qualquer um dos itens, pretendendo-se apenas a sua opinião pessoal e sincera.
Algumas questões possuem uma descrição de como realizar a tarefa.
Este questionário é de natureza confidencial e anónima e não deverá em momento algum do questionário escrever algo que o possa identificar mais tarde.

* Required

1. **Idade:** *

   .....

2. **Sexo:** *
   Mark only one oval.

   - ◯ Masculino
   - ◯ Feminino

3. **Área Profissional/Estudos** *

   .....

4. **Habilitações Literárias** *
   Mark only one oval.

   - ◯ Secundário
   - ◯ Licenciatura
   - ◯ Mestrado
   - ◯ Doutoramento
   - ◯ Other: .....

5. **Modelo do Smartphone:**
   Pode encontrar essa informação em:
   Definições -> Acerca do dispositivo

   .....

6. **Versão do sistema operativo Android:** *
   Pode encontrar essa informação em:
   Definições -> Acerca do dispositivo

   .....

7. **Já conhecia o sistema Bitcoin?** *
   Mark only one oval.

   - ◯ Sim.
   - ◯ Não.

8. **Acha interessante a existência de uma moeda alternativa não governada?** *
   Mark only one oval.

   - ◯ Sim.
   - ◯ Não.

9. **Pode especificar a razão da resposta anterior?**

   .....

   .....

   .....

   .....

   .....

Download da Aplicação

Agora vamos passar à avaliação do sistema desenvolvido.

1. Para fazer download da aplicação para o smartphone basta ler o QR Code acima ilustrado (deverá ter o QR Code reader instalado referido na introdução do questionário).

2. A aplicação ficará guardada na pasta "Downloads".

3. Para instalar a aplicação poderá executá-la e seleccionar a opção "Instalador de Pacotes".

No caso de a instalação falhar deverá permitir a instalação de aplicações com origem desconhecida em: Definicoes -> Seguranca -> Origens desconhecidas

4. Depois executar a aplicação será apresentado um pequeno tutorial para auxiliar na compreensão da mesma.

5. Depois da conclusão do tutorial é feita uma sincronização, e para efeito de testes serão creditados na sua carteira 500 pontos.

10. **Facilidade de instalação e execução da aplicação?** *
    Mark only one oval.

    - ◯ Bastante difícil.
    - ◯ Difícil.
    - ◯ Fácil.
    - ◯ Muito fácil.

11. **Após o tutorial apresentado no início da aplicação, sentiu-se realmente apto para utilizar o sistema?** *
    Mark only one oval.

    - ◯ Sim.
    - ◯ Não.
    - ◯ Talvez.

12. **Facilidade em gerar um novo endereço privado.** *
    Deverá, na opção "Endereços para Receber", seleccionar a opção gerar novo.
    Mark only one oval.

    - ◯ Bastante difícil.
    - ◯ Difícil.
    - ◯ Fácil.
    - ◯ Muito fácil.

Endereço Público (Sérgio Costa)

13. **Facilidade em adicionar um novo endereço público.** *
    Deverá, na opção "Endereços para enviar", ler o endereço fornecido a partir do QR Code ilustrado acima.
    Mark only one oval.

    - ◯ Bastante difícil.
    - ◯ Difícil.
    - ◯ Fácil.
    - ◯ Muito fácil.

14. **Prefere usar endereços por extenso ou a partir de código QR?** *
    Mark only one oval.

    - ◯ Endereço por extenso.
    - ◯ Código QR.

15. **Facilidade em efetuar uma transação de pontos?** *
    Deverá efetuar uma transação para o endereço fornecido e importado antes.
    Mark only one oval.

    - ◯ Bastante difícil.
    - ◯ Difícil.
    - ◯ Fácil.
    - ◯ Muito fácil.

Visão geral da aplicação.

Cada separador corresponde a uma funcionalidade da aplicação.
Após o teste de cada uma das funcionalidades classifique cada uma das seguintes questões:

16. **O que tem a dizer acerca da facilidade de compreender e utilizar a aplicação?** *
    Mark only one oval.

    - ◯ Bastante difícil.
    - ◯ Difícil.
    - ◯ Fácil
    - ◯ Muito fácil.

17. **O que achou da navegabilidade e estrutura da aplicação?** *
    Mark only one oval.

    - ◯ Bastante difícil.
    - ◯ Difícil.
    - ◯ Fácil.
    - ◯ Muito fácil.

18. **Considera que sejam necessários conhecimentos técnicos para poder utilizar a aplicação?** *
    Mark only one oval.

    - ◯ Sim.
    - ◯ Não.
    - ◯ Talvez.

19. **Depois de ter usado a aplicação pode dizer-se que o tutorial inicial** *
Mark only one oval.

- ◯ Foi útil.
- ◯ Não foi útil.
- ◯ Other: .....

20. **Acha útil possuir um cartão de pontos virtual inserido no dispositivo móvel?** *
Mark only one oval.

- ◯ Sim
- ◯ Não

21. **Pode especificar a razão da sua resposta anterior?**

.....

.....

.....

.....

.....

22. **Acha útil poder trocar pontos (por exemplo: Continente) com outras pessoas?** *
Mark only one oval.

- ◯ Sim
- ◯ Não

23. **Pode especificar a razão da sua resposta anterior?**

.....

.....

.....

.....

.....

24. **Se uma grande marca passasse a disponibilizar um cartão de fidelização como o que lhe foi apresentado, iria preferi-lo em detrimento dos que existem?**
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Sem opinião.

25. **Que alterações sugeria para esta aplicação?**

.....

.....

.....

.....

.....
