UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

NFC AND MOBILE PAYMENTS TODAY

André Filipe de Azevedo Figueiredo Cruz

MESTRADO EM SEGURANÇA INFORMÁTICA

Novembro 2011

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

NFC AND MOBILE PAYMENTS TODAY

André Filipe de Azevedo Figueiredo Cruz

Orientador
Norman Sadeh

Co-Orientador
Nuno Fuentecilla Maia Ferreira Neves

MESTRADO EM SEGURANÇA INFORMÁTICA

Novembro 2011

Resumo

NFC (Near Field Communication) e pagamentos móveis são duas áreas que se tornaram muito populares ultimamente, ambas duplicaram o seu índice de volume de pesquisas medido pelo Google Trends no último ano. NFC é uma tecnologia de comunicação sem fios já disponível em alguns telemóveis, sendo que mais estão anunciados para breve, e os pagamentos móveis são um serviço cuja utilização se espera que cresça a um ritmo bastante acelerado nos próximos anos. Este crescimento já foi previsto antes, e as expectativas saíram goradas, mas pensa-se que a NFC seja a tecnologia que vai trazer os pagamentos móveis às massas. Esta tese foca-se nestas duas áreas e em como a NFC pode ser útil num protocolo para executar pagamentos móveis nos dias de hoje. Para isto, um novo protocolo chamado *mTrocos* é apresentado. Este possui várias características desejáveis tais como anonimato, alta segurança, boa usabilidade, a não dependência de bancos ou instituições financeiras tradicionais, o suporte para micro-pagamentos e não requer nenhum hardware especial. O seu desenho é baseado no conceito de dinheiro digital e em protocolos de estabelecimento de chaves ad-hoc. Estes últimos são úteis visto que a NFC é um meio sem fios que não oferece nenhuma segurança de raiz para além do seu curto alcance. É detalhada uma prova de conceito da implementação usando um telefone com o sistema operativo Android e um leitor NFC de secretária, provando que ela funciona usando apenas hardware comum disponível actualmente. No entanto, a API (Application Programming Interface) de NFC do Android revelou-se limitada, o que influenciou o desenho do *mTrocos*, e o impediu de fazer uso apenas da NFC para a troca das suas mensagens. Como parte da avaliação do protocolo, foram feitos testes com utilizadores que mostram que o *mTrocos* é fácil de usar e que é indicado para o cenário pensado: máquinas de venda automática. Outra conclusão a que se pode chegar é que a NFC é uma tecnologia que melhora a experiência de utilização e que vai ser de grande utilidade para o crescimento dos pagamentos móveis.

Palavras-chave: NFC, Pagamentos Móveis, Android, Segurança

Abstract

NFC (Near Field Communication) and mobile payments are two areas that have received a significant amount of attention lately. NFC is a wireless communication technology already available on some mobile phones, with more to come in the near future, and mobile payments are a service whose usage is expected to grow at a significant rate in the coming years. This growth has been predicted before, and expectations have been let down, but NFC is thought to be the technology that will bring mobile payments to the masses. This thesis is focused on these two areas and how NFC can be of use in a protocol to conduct mobile payments. For this, a new protocol called *mTrocos* is presented that possesses several desirable characteristics such as anonymity, high security, good usability, unbanked, support for micropayments and no special hardware requirements. Its design is based on digital money concepts and ad-hoc key establishment protocols. The latter are useful because NFC is a wireless medium and offers no built-in security other than its limited range. A proof-of-concept implementation with an Android phone and a desktop NFC reader is detailed, proving that it works using only commodity equipment currently available. However, Android's NFC API (Application Programming Interface) was found to be limited, which influenced the design of *mTrocos*, preventing it from relying only on NFC for the exchange of the messages. As part of the protocol's evaluation, user tests were conducted which show that *mTrocos* is easy to use and that it is suited to the envisaged scenario: vending machines. Another conclusion is that NFC is a technology that improves the user experience and will be of great help for the growth of mobile payments.

Keywords: NFC, Mobile Payments, Android, Security

Acknowledgments

First I would like to thank SAPO, specially Eduardo Pinto, who came up with the idea for the topic, and Celso Martinho, for the support and needed equipment to conduct the research and development. Without them this work would not have been possible.

Next I would like to thank my advisor, Norman Sadeh, for his guidance and valuable insight which allowed me to achieve and surpass my goals.

I would also like to thank my co-advisor, Nuno Neves, for his feedback on the various phases of the project, which contributed to an overall better final product.

Finally, a special thank you to SONY, in the person of Stephen Tiedemann, who provided me with additional NFC readers which helped me overcome some technical problems I was having with the original ones.

Lisboa, November 2011

Dedicated to my family

Contents

1 Introduction **1**
1.1 Contributions . . . 3
1.2 Thesis Organization . . . 4

2 Related work **5**
2.1 Mobile payments . . . 5
2.1.1 Smartcard-based services . . . 6
2.1.1.1 GeldKarte . . . 6
2.1.1.2 Porta-Moedas Multibanco . . . 7
2.1.2 Credit card-based services . . . 8
2.1.2.1 Square . . . 8
2.1.3 Phone-based services . . . 8
2.1.3.1 Serve . . . 8
2.1.3.2 Google Wallet . . . 9
2.1.3.3 Venmo . . . 9
2.1.3.4 Bump . . . 10
2.1.3.5 PingPing . . . 10
2.1.3.6 Isis . . . 10
2.1.3.7 Zoosh . . . 11
2.1.4 Solutions proposed in the literature . . . 11
2.2 Near Field Communication . . . 13
2.2.1 Possible attacks . . . 14
2.2.1.1 Eavesdropping . . . 15
2.2.1.2 Data corruption, modification or insertion . . . 15
2.2.1.3 Man-In-The-Middle . . . 15
2.2.1.4 Relay . . . 16
2.2.2 Countermeasures . . . 16
2.2.2.1 Data corruption . . . 16
2.2.2.2 Eavesdropping, data modification or insertion . . . 16
2.2.2.3 Man-In-The-Middle and relay . . . 16

3 PoC mobile payments protocol — *mTrocos* **19**
3.1 Main characteristics . . . 19
3.2 Assumptions . . . 20
3.3 Design . . . 20

3.4 Design alternatives . . . 28
3.4.1 NFC only . . . 28
3.4.2 NFC + QR Codes . . . 30
3.5 Threat model . . . 32
3.5.1 Security threats . . . 32
3.5.2 Privacy threats . . . 35
3.6 Classification . . . 35

4 Implementation and evaluation **37**
4.1 Implementation . . . 37
4.1.1 Client . . . 40
4.1.2 Vending machine . . . 42
4.1.3 Operator . . . 44
4.2 Current Android NFC implementation pitfalls . . . 44
4.3 Evaluation . . . 48
4.3.1 Performance . . . 48
4.3.2 Hardware cost . . . 49
4.3.3 User tests . . . 49
4.3.4 Attained characteristics . . . 53

5 Conclusion **55**

A Users tests procedure **57**

Bibliography **59**

List of Figures

2.1 GeldKarte loading procedure 6
2.2 GeldKarte payment procedure 7

3.1 High-level description of *mTrocos* 21
3.2 Phases of payment operation 22
3.3 First phase of payment operation 22
3.4 Second phase of payment operation 24
3.5 Third phase of payment operation 24
3.6 Fourth phase of payment operation 25
3.7 Redeeming of *notes* 25
3.8 Description of the ad-hoc key establishment 27
3.9 QR Code example 30

4.1 Architecture of the prototype 37
4.2 Layout of the *Authorizer* activity 40
4.3 Layout of the *Handshaker* activity 41
4.4 Layout of the *Bluestarter* activity 41
4.5 Vending Machine UI 43
4.6 PN65N present on Nexus S (source: iFixit.com) 46
4.7 NFC subsystem hang 47
4.8 Android system crash 47
4.9 NFC service death 47
4.10 NFC stack deadlock 48
4.11 NFC subsystem crash 48
4.12 Answers to test questions 50
4.13 Android Bluetooth error 50
4.14 Verification string comparison results 51
4.15 The two different hypotheses to improve the test results 52
4.16 Verification string comparison results with implemented changes . . . 53

List of Tables

2.1 Comparison table of deployed online mobile payment solutions 5
2.2 Comparison of payment protocols in literature 11

4.1 Value of *mTrocos* system variables 39

Abbreviations

API	Application Programming Interface
B2B	Business To Business
B2C	Business To Consumer
CA	Certificate Authority
CPU	Central Processing Unit
CRL	Certificate Revocation List
DEP	Data Exchange Protocol
DH	Diffie-Hellman
DoS	Denial of Service
ECC	Error-Correcting Code
EMP	ElectroMagnetic Pulse
GMT	Greenwich Mean Time
HTTPS	HyperText Transfer Protocol Secure
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LCD	Liquid Crystal Display
LLCP	Logical Link Control Protocol
MIME	Multipurpose Internet Mail Extensions
MITM	Man In The Middle
MPASP	Mobile Payment Application Service Provider
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
NFCIP	Near Field Communication Interface and Protocol

NPP	NDEF Push Protocol
OOB	Out Of Band
OS	Operating System
P2P	Peer to Peer
PCD	Proximity Coupling Device
PDU	Protocol Data Unit
PKI	Public Key Infrastructure
PMB	Porta-Moedas Multibanco
POS	Point Of Sale
RF	Radio Frequency
ROM	Read-Only Memory
SD	Secure Digital
SDK	Software Development Kit
SMS	Short Message Service
SPP	Secure Simple Pairing
TCP	Transmission Control Protocol
TSM	Trusted Service Manager
TTP	Trusted Third Party
UI	User Interface
URI	Uniform Resource Identifier
URL	Universal Resource Locator
USB	Universal Serial Bus
UUID	Universally Unique ID

Chapter 1

Introduction

Two areas that have become very popular recently are mobile payments and NFC, as their search volume index on Google Trends shows. In the past year it has doubled for both of these terms and the trend is increasing. NFC is a wireless communication technology that is only available on a small number of mobile phones as of now, but this number will increase soon as multiple vendors have announced support for it in the upcoming models. Mobile payments are a service that has yet to see widespread usage, but this usage is expected to grow significantly in the years to come. This growth has been predicted before, and expectations have been let down, but NFC is thought to be the technology that will bring mobile payments to the masses [118]. With NFC available in the handsets, merchants will have more incentives to invest in PoS NFC technology and thus NFC will gain the momentum needed for broad adoption. This thesis is focused on these two areas and how NFC can be of use in a protocol to conduct mobile payments. It has long been touted as a technology that will be widely deployed on mobile phones and the perfect enabler for the adoption of mobile payments. Therefore, it is no surprise that a significant effort has been put into studying its possible applications on a mobile phone, as well as its security properties, vulnerabilities and countermeasures.

Imhontu and Kumah [50] give an overview of NFC and its various uses when implemented in mobile phones, while Ondrus and Pigneur conduct an analysis [92] that shows that NFC has great potential as a technology for future mobile payments and industry experts expect it to be successful. It is cheaper and more reliable than standard chip cards, providing much better performance than other mobile phone technologies that could be used instead [93]. Its low friction allows it to be much faster than other standard mobile payment solutions [70]. Haselsteiner and Breitfuß [47] enumerate several threats to the NFC protocol such as eavesdropping, data corruption, data modification, data insertion and MITM (Man In The Middle) attacks, and examine how the different communications modes of NFC, active or passive, impact its security. Their results will be discussed in Subsection 2.2.1. Madlmayr, Langer and Kantner [69] discuss NFC use cases for integrating this technology on devices and possible attacks on security and privacy. They conclude that NFC has issues regarding security and privacy that can be solved by employing a set of best practices in future devices:

- Adoption of a button so the user can turn off the NFC functionality.
- Discouraging the use of the unique ID present in contactless smart cards for the anti-collision phase, because they can be used for identification.
- Signing of tags and applications.
- Inclusion of an application to manage the secure elements of an NFC device.
- Using the security features of the secure element to secure the NFC P2P (Peer to Peer) connections.

Since there are already phones on the market that support NFC, such as the Nokia 6131 and 6212, there have also been studies on attacks that focus specifically on these mobile phones or use them to conduct attacks. Mulliner [76] performs a security analysis of a NFC-enabled phone, the Nokia 6131, in PCD (Proximity Coupling Device) mode of operation and looks at the security of several NFC-based systems currently deployed that use NFC tags. He discovers several bugs in the NFC implementation of the phone and problems in the handling of NDEF (NFC Data Exchange Format) [81] messages. Weiß [128] demonstrates a relay attack on NFC using the PCD and card emulation modes, while Francis et al. [34] demonstrate a relay attack on the NFC P2P mode using mobile phones as both the victims and attackers. This attack in particular is done using the software API available on the phones, no special hardware is necessary, and countermeasures for both of these attacks are presented. These countermeasures are based on location-aware and distance bounding protocols. Verdult and Kooman [126] analyze the more recent Nokia 6212 Classic and find several vulnerabilities allowing Bluetooth pairings and application installations to be done without user intervention through the P2P link.

Traditionally there were five types of payment systems, each with their own advantages and disadvantages: cash, checking transfer, credit cards, stored value and accumulating balance. The web has introduced new requirements for these systems in the form of P2P payments, B2B (Business To Business) payments and micropayments. Online payment systems are typically either adaptations of traditional offline payment solutions or entirely new solutions, and they usually take one of these forms: digital cash, online stored value systems, digital accumulating balance payment systems, digital credit accounts and digital checking. According to [106], mobile payment services will continue to grow faster in developing markets because there are less alternatives and regions exist where it can be the only way people have access to financial services and make payments, with the next trend being to capture the large unbanked population on these markets. In developed markets users are much more worried about security so they usually start by using features such as text alerts and balance checking, after that they may progress to bill payment, before moving on to P2P payments. Some vendors will merge, while at the same time new players will appear offering specialized and custom solutions, since payment systems have to obey local laws and market characteristics, and so the same solution cannot be applied everywhere. Regulators are expected to make their appearance but will probably settle on middle ground, opting not to stifle the development of these services.

In some markets such as India, regulators have given banks the central role in mobile payments and banking and this impedes market growth, favoring vendors with bank-grade solutions against telecom-grade solutions.

PayPal has been the leader in Mobile payments but its leadership is dependent on others, since it does not control the financial networks where the payments go through and does not control the devices the consumers use or the data networks where the transactions occur. Today, the companies that control them, such as American Express, are looking to enter the market as well [15].

This thesis will present a novel mobile payments protocol, *mTrocos*, that resorts to NFC. The protocol's main target is vending machines, allowing customers to use their mobile phones to pay for products. Their mobile account balance is used to finance their purchases so the operator serves as bank, completely sidestepping the traditional financial institutions and protocols. This way, using less intermediaries, less fees and less overhead are needed to conduct payment operations, which is a benefit for all actors of the protocol. It also possesses several desirable characteristics — anonymity, high security, good usability, support for micropayments and no special hardware requirements — that cater to the needs of future clients, merchants and operators. Its design is based on digital money concepts, the basis for the anonymity support, and ad-hoc key establishment protocols, which are useful because NFC is a wireless medium and offers no built-in security other than its limited range. A proof-of-concept implementation using an Android [43] phone and a desktop NFC reader is detailed, proving that it works with only commodity hardware currently available. However, Android's NFC API was found to be limited, which influenced the design of *mTrocos* preventing it from relying only on NFC for the exchange of the messages. As part of the protocol's evaluation, user tests were also conducted in order to ensure that all requirements were met and to request the users' opinion on using *mTrocos*.

1.1 Contributions

This work makes three contributions:

1. A new mobile payments protocol for use with vending machines is proposed, called *mTrocos*, that takes advantage of NFC and possesses the following characteristics: good usability, support for micropayments, unbanked, secure, anonymous and no special hardware requirements. It takes into account in its design the current state of the art on NFC security.

2. A complete implementation of the *mTrocos* protocol using a phone with the Gingerbread [41] version of the Android mobile operating system and a desktop NFC reader. As part of this implementation were developed an Android application, a vending machine simulator that includes an implementation of the NPP (NDEF Push Protocol) [42] protocol, and an operator simulator.

3. An evaluation of the *mTrocos* protocol performed with user tests, followed by improvements to the protocol, until a final version was reached that complies with all the stipulated requirements.

1.2 Thesis Organization

In Chapter 2, related work is presented regarding mobile payments and NFC. Both deployed solutions and proposed solutions for mobile payments are discussed, comparing them with *mTrocos* when enough information is available. Also, an overview of NFC is given, focusing on NFC security, possible attacks and countermeasures. In Chapter 3, the design of the new mobile payments protocol, *mTrocos*, is presented along with several design alternatives. In Chapter 4, the prototype implementation of *mTrocos* is detailed and evaluated. As a side note, the current limitations of the NFC API present in Android are also put forth, since they had an impact on the development of *mTrocos* and should be taken into consideration by anyone wanting to use NFC on that platform. Chapter 5 concludes this work by presenting a business perspective and a recommendation regarding the adoption of NFC.

Chapter 2

Related work

This chapter provides background and context information on mobile payments and NFC, and the necessary security considerations one should take into account in order to use NFC. A comparison is done among existing mobile payment solutions deployed today, some of the available new proposals, and *mTrocos*. In several cases it was difficult to find technical information on currently deployed solutions, so a best-effort attempt is done using the publicly available information. The current state of the art on NFC security is also presented, discussing the threat model of the NFC medium and giving pointers on how to overcome the security problems.

2.1 Mobile payments

A survey of the deployed mobile payment solutions was carried out. Below, each solution is explained with the information that was obtainable from their web sites. Table 2.1 provides a breakdown of current mobile payment solutions according to their characteristics. All of them operate online and most just extend the functionality of traditional credit cards to the mobile payment scenario. Over the years, several players have appeared, changing the landscape of the first reviews of mobile payment

Solution	Type
GeldKarte	Digital Cash
PMB	Digital Cash
Square	Digital Credit Account
Serve	Digital Credit Account
Google Wallet	Digital Credit Account
Venmo	Digital Credit Account
Bump	Online Stored Value
PingPing	Digital Accumulating Balance Payment
Isis	Digital Credit Account
Zoosh	Online Stored Value

Table 2.1: Comparison table of deployed online mobile payment solutions

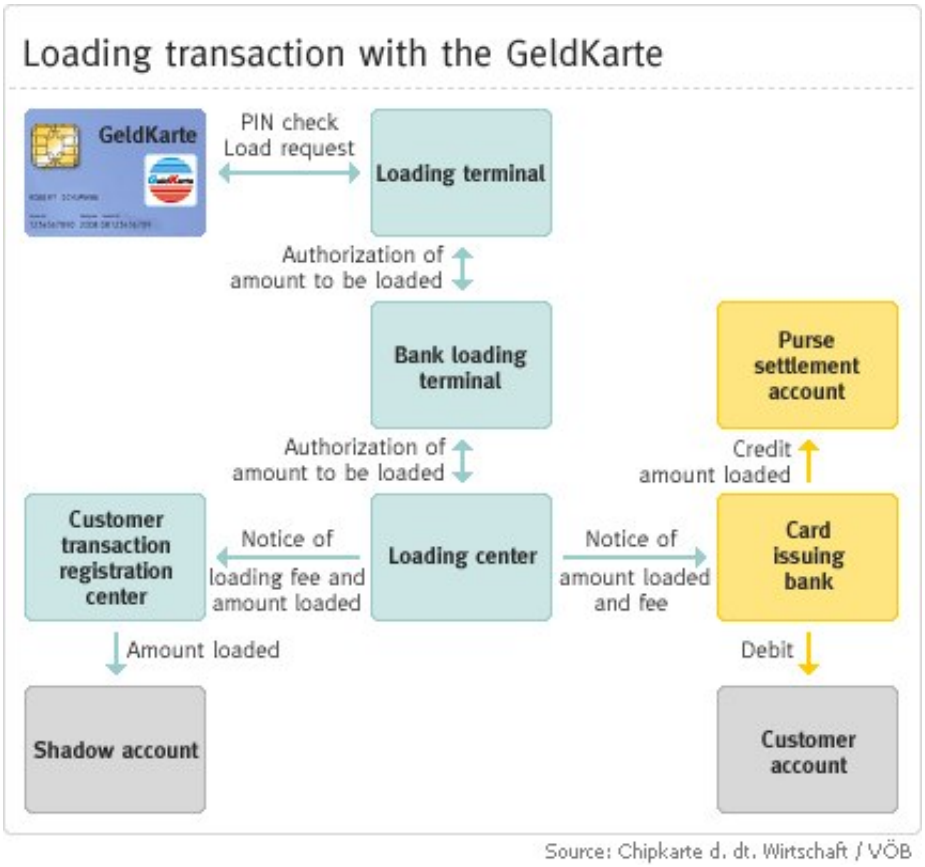


Figure 2.1: GeldKarte loading procedure

solutions [102]. In the following sections each solution is detailed in order to provide a clear idea of how the current players operate and their target.

2.1.1 Smartcard-based services

2.1.1.1 GeldKarte

The GeldKarte is a form of digital cash. It consists of a smartcard that can be loaded with a maximum of €200 [37] and that can be used for instance in public transportation, vending machines, public phones and online payments with a secure card reader. During the loading phase, depicted in Figure 2.1, the issuing bank is contacted so that the customer account can be debited with the correct amount, which is then put in a common purse settlement account at the issuing bank and also transferred to the microchip in the card. During the payment phase, depicted in Figure 2.2, no authorization system is used, the amount in question is deducted from the credit stored on the chip and credited to the merchant's chip that is in the terminal. This process is done offline, which minimizes the transaction costs and time needed for the operation. There is also the possibility to conduct payments online, via the Internet. During the settlement of the payment transactions the merchant's chip generates a cumulative total, encrypts all individual transactions, and contacts

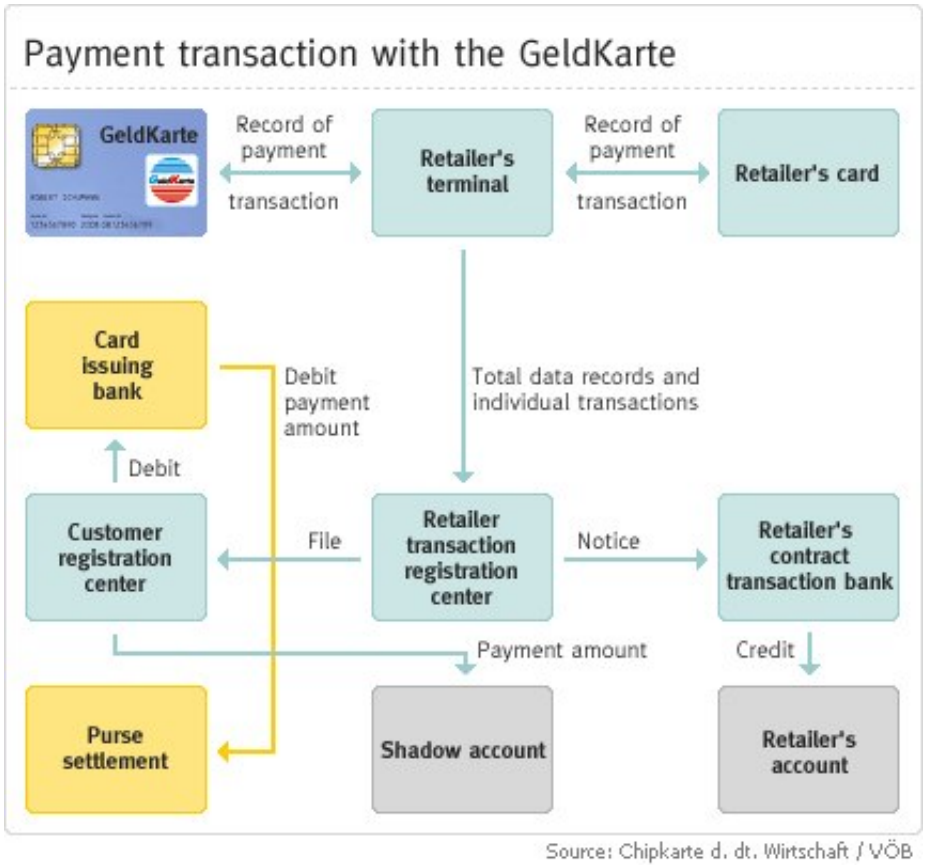


Figure 2.2: GeldKarte payment procedure

the transaction registration centre who in turn contacts the issuing banks. Since payments are cleared in totals, and not individually, it is hard for a bank to identify the customers responsible for the transactions [36].

The merchant cannot identify the customer, and the issuing bank does not know which customer did the transaction, but the customer registration centre keeps a shadow account linked to each card that can be used to recover the previous transactions and the current amount in the card. So it is possible to recover a customer's transactions if the customer registration center and bank exchange some information. The merchants pay a fee of 0,3% of the transaction value (min €0,01) [38].

Additionally, there are stand-alone cards, called *white cards*, which are not linked to any particular bank account so they cannot be traced to an individual[1]. There are also standalone GeldKarte terminals, which have only electronic purse functions.

2.1.1.2 Porta-Moedas Multibanco

In Portugal there was a system called PMB (Porta-Moedas Multibanco) [107] that was very similar to the white GeldKarte cards presented in Subsection 2.1.1.1. It used standalone smartcards that could be charged up to €200 on ATMs from a debit

[1]They can be charged with cash at a bank counter.

card, or on bank counters using cash. They provided anonymity because they were not linked to any individual or bank account. Payment operations were also done in a similar way to GeldKarte — the specified amount would be transferred from the chip in the PMB card to the merchant's chip, and later a reconciliation phase would happen between merchant's bank and the issuing banks. Here, all the accumulated transactions would be presented and the sum of money would be transferred to the merchant's bank account.

Since it was introduced in 1995 and discontinued in 2006 [18, 23, 108], this system did not have success on the portuguese market. Both banks and users blame the lack of adoption by merchants as the main reason why it failed, and when the Euro was introduced, all the cards had to be replaced and technology upgraded causing its adoption to come down to insignificant levels.[2] It is thought that the high equipment costs — a Multibanco terminal was needed — was the main reason that few merchants adopted it, but there are also those who say that since there were no intermediary taxes (i.e., no transaction fees) the banking system did not fully support it.

2.1.2 Credit card-based services

2.1.2.1 Square

Square [112] enables credit card usage through phones or tablets. Both iOS and Android operating systems are supported by Square. It has a reader, sent for free, that plugs in the audio jack and allows cards to be swiped. The square account and application that runs on the device are also free but the merchant pays 2,25% per swipe if the reader is used. After a payment operation, the funds are immediately sent to the merchants bank account.

There is also the Card Case feature [39] that enables users to pay just by saying their name at the checkout counter of their frequent merchants. This requires users to also download square's application and so requires a compatible device. The buyer's personal information is not shared by Square with the merchants.

There are several other services similar to Square, which aim to make use of traditional credit card payments through a mobile phone, such as Intuit's GoPayment [51], VeriFone's Payware Mobile [127] or Corduro [17].

2.1.3 Phone-based services

2.1.3.1 Serve

Serve [2] is a new service from American Express that allows customers to transfer money between them while online and use their mobile phones to make payments. Serve accounts are loaded from bank accounts, debit or credit cards or funds from another user, and can be accessed through Android or iOS applications, serve.com and Facebook. Customers are also provided with a reloadable pre-paid card, linked to their Serve account, which can be used as an American Express credit card and

[2] In October 2005 only 3754 PMB cards were active.

thus support transactions while being offline. Merchants, however, have to be always online during a transaction. In the case of P2P transfers they are also performed online, either through serve.com or mobile apps. No anonymity is provided but sub-accounts can be created, and limits may be imposed on the amount spent and how it is spent.

2.1.3.2 Google Wallet

Google is teaming with MasterCard and Citigroup to integrate technology in Android phones to enable them to be used for mobile payments. This system was announced on May 2011 [31, 44]. It will not charge a transaction fee but will instead focus on offering retailers more data about their customers, so that in turn they can buy targeted advertising from Google. The special cash-register systems from VeriFone Systems will be installed at merchants at Google's expense. An application that relies on NFC technology, Google Wallet, is used for the payments and for now it will only work on the Nexus S 4G. This system does not provide anonymity to customers. The mobile wallet will use a Citibank issued MasterCard credit card number and a virtual Google pre-paid MasterCard card. It can be used at any of the 124000 merchants that have the PayPass terminals from MasterCard, and there are also deals with retailers that will enable consumers to redeem discounts and participate in loyalty programs. Since the phone's network connection is not used but the PoS needs a network connection, the customer is offline and the merchant is online. Google refers to this Wallet as "basically the same" as a credit card [14].

2.1.3.3 Venmo

Venmo [124] is targeted at P2P transactions. It works as an Android or iPhone application and currently has no transaction fees, although businesses will have these fees when B2C (Business To Consumer) transactions are supported. It has a social component in that trust relationships can be built between users to bypass charge confirmations, as long as they are reciprocal. It also relies heavily on existing social networks by encouraging their users to broadcast Venmo operations. This is used not only to notify the recipient of the transaction but to increase awareness of the service in other users hoping that it will become viral. A public API is also provided to allow third party services to integrate with Venmo and provide new functionalities. Venmo accounts are linked to credit or debit cards to obtain the funds, and then the funds can be withdrawn by a bank transfer. It also produces statistics for users about their spending habits, and anonymous public data has been released [125] that reveals interesting trends. For example, one can see that most payments are for food, specially splitting meals.

2.1.3.4 Bump

Bump [9] is a technology that can be used to transfer information between two devices[3] that have been "bumped" together. "Bumping" means running the Bump application, choosing the option to bump on both devices, and finally physically bumping the phones together. The application will transmit the location, other physical properties and timing information to the Bump servers, which will try to determine the other device involved in the bump. Assuming the match is found, information can then be transmitted between the devices through the Bump servers. It is currently used to transmit contact information, photos, music, apps, calendars, invites and connect via Twitter, Facebook and LinkedIn. In addition, PayPal and ING Direct have also started using it for P2P money transfers, allowing easy transfers of funds between people physically near one another.

2.1.3.5 PingPing

PingPing [75] is a mobile micro-payment service that can be used via SMSs (Short Message Service) or NFC tags. In the case of SMSs, a customer can pay for online items, meals or bus tickets by sending an SMS to a provided number with a code given to him. A message confirmation is then received on the mobile phone that can be used as proof of payment. In the case of NFC tags, the phones are scanned to identify the specific customer, the merchant then has to go online to look up the customer information and balance in order to perform the transaction. No anonymity is provided because each NFC tag is unique, and as the limit for transactions is small, €25, no PIN or confirmation is used. P2P transactions are supported via the PingPing mobile application.

2.1.3.6 Isis

Isis [53] is a joint venture between AT&T, Verizon and T-Mobile. The original plan was to develop its own mobile payment service, but recently [89] it announced a change in the strategy where Isis will work with the current players and enable US merchants to provide NFC-based services to the clients of Isis's founding partners. Banks can also use other TSMs (Trusted Service Manager), but Isis intends to become the preferred TSM by providing a single integration point. The first Isis pilot will go live in 2012. The Isis mobile application is to come pre-loaded on compatible phones either in the SIM, or in the phone's hardware [129]. The app will not handle payment transactions, just store card data and the regular payment network is used.

Sprint is planning a similar solution [5], trying to beat Isis to the markets, as is Visa [35, 72]. In fact, recently [71], Visa partnered with Isis in order to allow customers of the Isis mobile wallet to use Visa debit, credit and pre-paid accounts.

[3] iOS and Android phones as of now.

Property	[48]	[65]	[122]	[3]	[60]	[64]	*mTrocos*
Anonymous				✓			✓
Unbanked	✓		✓	✓			✓
P2P			✓	✓			
Offline merchants			✓	✓			✓
Micropayments	✓		✓	✓			✓
No TTP			✓	✓	✓		✓
No special hardware						✓	✓

Table 2.2: Comparison of payment protocols in literature

2.1.3.7 Zoosh

Zoosh is a technology developed by Naratte [78] that aims to provide the same functionalities as NFC on phones that do not have NFC chips. It does this by leveraging the microphone and speaker found on every phone and SDKs (Software Development Kit) have been released for the Android and iOS platforms. Zoosh is still not used for mobile payments, but SparkBase has begun employing it to allow customers to redeem rewards and coupons, and track their account status with a merchant [111].

2.1.4 Solutions proposed in the literature

Over the last few years several authors have designed mobile payment protocols with different characteristics. This section reviews some of the most interesting proposals. As these proposals typically include substantial technical information, it is possible to perform a comparison between them and *mTrocos* with regard to the most important characteristics of a mobile payments solution (see Table 2.2).

Horn and Preneel's [48] protocol provides micropayments based on *ticks*, where every amount must be a multiple of 1 *tick*. These *ticks* are pre-images of an initial value under a one-way hash function that is committed at the beginning of the protocol. The user commits by signing this initial value and reveals the number of *ticks* needed to make a payment of a certain amount. This protocol is not anonymous, requires a TTP (Trusted Third Party) and a long-term private key stored in secure storage in the mobile phone.

Kungpisdan, Srinivasan and Le [65] present a banked mobile payment protocol. It assumes that a payment gateway is used, a TTP, and that shared keys exist between the payment gateway and the merchants, and between the issuer banks and the clients. Clients have to run a client registration protocol the first time they interact with a merchant in order to establish a shared key. One-way hash functions are used to derive subsequent keys in all cases. Using these shared keys, clients generate messages that merchants can decrypt, and that include other messages to be decrypted by the issuer bank. Merchants contact the payment gateway, which in turn contacts the issuer and acquirer banks. The payment gateway responds to the merchant with the result of the payment request. This protocol relies only on symmetric crypto, because it is

less computationally demanding than public key crypto operations.[4] It also does not support anonymity and relies on TTPs and tamper-proof storage for storing long-term keys on the mobile phone.

Van Damme et al. [122] discuss a mobile payment protocol that supports offline transactions and P2P. Their system is based on the notion of vouchers, which are generated by an issuer and sent encrypted to a secure element on the beneficiary's mobile phone using an SMS. P2P transactions can be done offline, between beneficiaries, while payment operations, between a beneficiary and an affiliate, can be done offline or online. The final phase, the clearance, happens between the affiliate and the issuer and is used to redeem the vouchers. This protocol relies on secure storage provided by a secure element and assumes the deployment of a PKI (Public Key Infrastructure) where phones and affiliates have key pairs signed by the issuing bank. This solution is in contrast with *mTrocos*, which does not require any of these.

mFerio [3] is a P2P payment protocol that relies on NFC. It is strictly P2P, in the sense that no infrastructure is needed and both parties are offline, and is designed to have better usability, security and auditing capabilities than cash-based transactions. Like *mTrocos*, it relies on the notion of digital cash for the money representation employed. However, mFerio relies on a secure element to achieve secure data storage for this digital cash, and although these chips are available in some mobile phones today, they might not be accessible to normal applications. As an example, the Android API does not give access to this secure element, even though the Nexus S phone has a chip with these capabilities. Therefore, a protocol that does not rely on the secure element has a higher probability of being usable today, which is exactly one of the goals of the *mTrocos* protocol. Another difference is that since mFerio is designed for transactions between two mobile phones, it assumes that users will clearly see anyone trying to intercept a transaction and will know exactly with which device they are communicating with. In a vending machine scenario these assumptions are serious vulnerabilities that can be exploited, and therefore the *mTrocos* protocol has to address these potential attacks.

Kadambi, Li and Karp [60] also describe a mobile payment protocol based on NFC that relies on traditional banking protocols, such as EMV [30] and credit cards. Using the secure elements in mobile devices to issue payment authorization tokens they have managed to prevent the transfer of sensitive information over public networks. This banked protocol requires merchants to be online during transactions, payments are traceable by banks, and requires secure storage such as a smartcard or a secure element in the mobile phone. A PKI also needs to be deployed where each client has at least one public-private key pair stored on the secure storage in the phone.

Kumar and Rabara [64] propose an architecture for mobile payments based on NFC, but they focus on macropayments while *mTrocos* is tailored to micropayments. A service-oriented architecture is presented that tries to integrate with the existing payment processing infrastructure and their goal is to improve the protection of user

[4]It is worthy of note that when this proposal was elaborated the CPUs (Central Processing Unit) present on mobile phones were much less powerful than the ones on today's phones, so public key crypto operations were too expensive.

credentials, more user control over the transaction, and support for both proximity and remote transactions over the web. Their architecture describes a banked macro-payments protocol which requires merchants to be online and relies on symmetric crypto and shared secrets with the MPASP (Mobile Payment Application Service Provider) server, a TTP.

Looking again at Table 2.2 we can now have a better picture of the described solutions and how they compare with each other. mFerio and Van Damme et al. are the solutions that share the most characteristics with *mTrocos*, but they both rely on special hardware, the secure elements. Since it is not guaranteed that an application running on a mobile phone will have access to this secure element, it is a requirement of *mTrocos* not to use it.

2.2 Near Field Communication

NFC is a wireless communication technology that operates in the unlicensed radio frequency of 13.56MHz. It has an operating distance of 4 cm and supports data rates that vary from 106 to 424 kbit/s. It relies on magnetic induction where a device, an initiator, generates a RF (Radio Frequency) field that can power a passive device, the target. These passive devices modulate the existing field to answer the initiator, and since they do not require a power source, they can have simple form factors such as tags or cards. It is also possible to use an active communication mode, when both devices are powered, for P2P communication. In this case, both the initiator and target devices generate their RF fields alternatively, deactivating them when waiting for data. There is also the card emulation mode, when a reader emulates a tag to another device. One of the advantages of NFC, is that it requires no setup or pairing. Frictionless interaction is possible just by bringing two compatible devices together.

Over the years several independent and incompatible tag technologies have been developed but, since 2003, standardization efforts have been made to ensure that compliant devices can communicate with each other. ISO/IEC 18092 / ECMA-340 [29] specify the interface and DEP (Data Exchange Protocol), and ISO/IEC 21481 / ECMA-352 [28] specify the mode selection mechanism. In 2004, the NFC Forum [88] was formed, whose goals are to advance the use of NFC by developing specifications, ensuring the interoperability among devices and educating consumers and enterprises about NFC. It already has 150 members among manufacturers, application developers and financial services institutions. NFC Forum compliant devices have to support ISO/IEC 14443 [55, 57, 58, 56] and four types of tags have been standardized: Type 1 [82], Type 2 [83], Type 3 [84] and Type 4 [87]. Most NFC readers today support all these standard types plus the proprietary MIFARE Classic [90]. MIFARE is a chip technology owned by NXP that is used for contactless smart cards or proximity cards. The MIFARE Classic card in particular is ISO/IEC 14443-A compliant and comes in three memory sizes: 320B, 1KB and 4KB. It uses a proprietary security protocol for mutual authentication and data secrecy called CRYPTO1. This protocol has however been broken [22], so its security has been rendered useless and therefore should not be used anymore.

A common data format, NDEF, was also developed. This format allows the transmission and storage of various kinds of data independently of the tag technology being used underneath, and it should be used when transmitting data to NFC devices. Another important standard for P2P communication between NFC devices is LLCP (Logical Link Control Protocol) [85]. It builds on ECMA-340 and allows the definition of service types and classes for bi-directional communications, resulting in a more robust environment for developing NFC-based applications.

2.2.1 Possible attacks

This subsection introduces some attacks to NFC communications and devices. Two interesting works on this area were performed by Mulliner [76] and Verdult and Kooman [126]. In their studies, they focused on the NFC stack of Nokia phones, and their main findings can be summarized in the following points:

- Using fuzzing techniques on NDEF and LLCP parameters it was possible to cause the crash of phones.
- By taking advantage of URI (Uniform Resource Identifier) obfuscation on Smart Poster tags, it was possible to trick users into visiting malicious URLs (Universal Resource Locator), make unintended phone calls and send SMSs.
- Through the "Content Sharing" feature present on Nokia phones, which is turned on by default and normally used to easily transfer passive content between devices, it is possible to upload a malicious application without user consent. It is also possible to enable Bluetooth and pair with a target device using a tag with a "Connection Handover" NDEF message. These two attacks combined can be used to transfer the whole contents of the phone through the Nokia PC Suite interface, or give the just uploaded application elevated access rights.

These attacks do not work against the Android NFC stack. Both NDEF and LLCP parameter fuzzing was tried with no success, Smart Poster URI obfuscation is correctly dealt with by the Android UI, and applications need permission to be installed and can only send messages when they are running in the foreground.

Haselsteiner and Breitfuß [47] enumerate a list of attacks and discuss their applicability. They also provide countermeasures when the threats prove to be real, as in the case of eavesdropping or data modification attacks, concluding that a secure channel needs to be established. They claim that since NFC is not vulnerable to MITM attacks, any key agreement technique without authentication can be used. However, it is assumed that a third-party cannot interpose himself between the two NFC communicating parties, blocking them from seeing each other, and allowing data exchanges only through the attacker. In the case of vending machines in public spaces, one cannot safely assume this, so the applicability of the attacks is revised below, taking this fact into account.

2.2.1.1 Eavesdropping

NFC is a wireless protocol and so it is susceptible to eavesdropping. It has a small range, but this varies with several factors such as the power used by the NFC devices, the NFC mode being used by the source, the physical barriers of the local environment, the quality of the attacker's receiver or antenna, but is mostly influenced by the NFC mode of the source. Therefore, it is not possible to give precise values to this range. However, in a normal scenario, a device operating in active mode can be eavesdropped up to a distance of 10cm, while a device operating in passive mode can only be eavesdropped up to a distance of 1cm. As a rough estimate, by varying the factors enunciated above, we can assume that a device operating in active mode can be eavesdropped up to a distance of 10m, while a device operating in passive mode can be eavesdropped up to a distance of 1m. In P2P mode, the two devices alternate between active and passive mode, active when transmitting data and passive when receiving, so this means that both devices can be eavesdropped up to 10m. Since the NFCIP (Near Field Communication Interface and Protocol) link has no underlying encryption, all communications are done in the clear.

2.2.1.2 Data corruption, modification or insertion

Data corruption, modification and/or insertion attacks can be performed in NFC communications. Data corruption requires only that the attacker transmit on valid frequencies at the right time, and this time can be calculated. Data modification can be hindered by the encoding method being used. For example, Manchester encoding makes it feasible to change all transmitted bits, while using Miller encoding makes the change feasible only for certain bits. Data insertion can be carried out as long as the answering device takes some time to respond, and meanwhile the attacker inserts his own response. If these two transmissions overlap, it will turn into a data corruption attack.

2.2.1.3 Man-In-The-Middle

Regarding MITM attacks there are two possibilities: either the two communicating devices continue to see each other when a third party enters the range, or the third party manages to block the communications between the original two devices.

In the first case, whenever an attacker tries to interfere with a transmission, the current source can detect it. Furthermore, if an attacker tries to send his own data to the destination, which is in passive mode, he will have to produce his own RF field. Since there is already an RF field being produced by the legitimate source these would have to be perfectly aligned, which is typically not feasible. If the P2P mode is being used then both the participants are using active mode to transmit their data. Here, the sources can detect interference produced by the attacker and interrupt the transmission, thus preventing the execution of a MITM attack.

In the latter case, a third party manages to block the communications between the original two devices and the attack turns into a relay attack (which has been proven to work both in PCD and P2P modes). As explained in the next subsection,

the relay attack can be used to perform a MITM attack because there is nothing to prevent a proxy from altering the payload of the messages it is relaying.

2.2.1.4 Relay

Relay attacks occur when two communicating devices think they are near one another and directly messaging each other but instead their data is being proxied and carried over a longer distance. Researchers have proven that this attack can be carried out against the NFC protocol. Weiß [128] has performed a successful attack against the PCD mode, by emulating a remote tag against a reader, using a mobile phone as a *mole* and a Beagle board as a *proxy*. Francis et al. [34] attacked the P2P mode of the protocol using four mobile phones. Two of the phones are attempting to communicate with one another through P2P and the other two phones act as *proxy* and *mole* relaying the signal between them. Both of these approaches used Bluetooth to transmit the signal between the *proxy* and *mole*. Since the anti-collision phase of the ISO 14443 specification has strict timing requirements, both the *proxy* and *mole* perform it locally and only relay the data packets between the two victims.

2.2.2 Countermeasures

This subsection introduces some of the countermeasures that can be employed to prevent the attacks to NFC devices.

2.2.2.1 Data corruption

Data corruption attacks are the hardest to defeat. They can easily be detected because a sender can monitor the RF field being generated, but it is very difficult to stop an adversary with the right tools to create collisions on the wireless medium. As there is no generic countermeasure, this results in a possible DoS (Denial of Service) opportunity.

2.2.2.2 Eavesdropping, data modification or insertion

Eavesdropping, data modification or insertion attacks can be prevented by establishing a secure channel prior to the transmission of the data. NFC per se does not offer this functionality, so this has to be implemented by an higher level protocol. A key agreement protocol, such as Diffie-Hellman, can be used to negotiate a shared secret that can then be used with a symmetric key algorithm, such as AES, to establish the secure channel.

2.2.2.3 Man-In-The-Middle and relay

Despite the small range of NFC, MITM and relay attacks are still possible and therefore a secure channel may end up being established with an entity that is not the one intended. To address this problem, location aware/distance bounding algorithms such as [8, 61, 100] are frequently suggested as a solution to address relay attacks.

These solutions might not be enough because an attacker already has to be close and the target can also be near, a situation which would not be detected. Also, these checks would need to be implemented at the physical layer, which involve changes to the NFC hardware. Roundtrip timing measurements at a higher level protocol such as LLCP might also not help in the P2P scenario because there are a multitude of factors that contribute to the response time, such as the OS (Operating System) of the device and other applications running in the background. LLCP even has a PDU (Protocol Data Unit) called *SYMM*, which is used when a peer has no information to send at that time, and so can prolong the response time of a request. In order to correctly mitigate MITM and relay attack threats, the user has to be assured of which other device he is communicating with, this is called "demonstrative identification". In Chapter 3 we discuss how to achieve this by leveraging the Diffie-Hellman approach with a String Comparison variant.

Chapter 3

PoC mobile payments protocol — *mTrocos*

Many times people have found themselves unable to buy a certain product from a vending machine, either because they did not have the exact change, or they did not have enough money on themselves. Some merchants have responded to this problem by allowing card payments, but this approach has its own set of limitations. For example, the merchants have to rely on third parties for the service availability and have to pay transaction fees, and the clients lose the anonymity afforded by traditional cash transactions. The goal of this work is to design a protocol that addresses all the aforementioned problems and uses only commodity hardware available for sale today. A possible use case is a client that wants to buy a certain product in a vending machine, and instead of using cards, coins or paper notes, employs his mobile phone for payment.

The chapter is organized in the following way: we start by presenting the new protocol's characteristics in Section 3.1 and this is followed by stating the fundamental assumptions the protocol makes in Section 3.2. After this, the design of the *mTrocos* protocol is introduced in Section 3.3 and the design choices are explained in Section 3.4. Finally, we elaborate on the threat model of the protocol in Section 3.5.

3.1 Main characteristics

The list with the main characteristics of a mobile payment solution was created taking in consideration previous studies [62, 98, 103] and the specific use case scenario mentioned above:

Good usability The protocol should be easy to use by anyone with a compatible mobile phone, without requiring any specific training. It should also be fast to avoid leading people to give up in the middle of a transaction.

Enable micropayments Most of the transactions carried out on vending machines are of small value, and therefore it is imperative that the transaction costs for merchants be minimal (otherwise they will not adhere). A small upfront cost is

allowed, since some equipment will have to be installed in the vending machines, but this cost has to be low.

Unbanked In order to simplify the protocol and remove intermediaries, the only three actors should be the telecom operator (who has a running account with a client), the client (who wants to pay for an item at a vending machine) and the vending machine merchant. This is to ensure that the transaction costs involved are kept at a minimum.

Secure No payment method will work if it is not secure. It will have to provide assurance to the merchants that they will receive the payments, to the clients that they are actually paying for the item they want and that the correct amount will be charged to their account, and to the operator that the money it is giving to a merchant is related to a transaction authorized by their client.

Anonymous In this age where privacy is becoming scarce, it is important to design new protocols that try to preserve anonymity from the start. Therefore this protocol should ensure that what is being bought and where, is kept private from the operator, and who is buying it is kept private from the merchant.

No special hardware requirements This protocol must be usable with off-the-shelf commodity hardware that is available for purchase today.

3.2 Assumptions

- There exists a secure source of randomness on both the vending machine and on the client.
- Vending machines, clients and operator have a clock that does not drift more than one day.
- If the DH-SC [10] protocol is used, users will compare the verification string presented in their mobile phones with the one presented by the vending machine, and cancel the protocol if these do not match.

3.3 Design

Figure 3.1 represents a high-level view of the *mTrocos* protocol. The client starts by obtaining the price of the product he intends to buy and then turns to the operator who produces a *note*. This *note* is then presented to the merchant, who accepts it and releases the product. Later, the merchant can redeem this *note* from the operator and obtain the funds. This redeeming operation should be done at the end of the day with all the *notes* received during that day, so as to amortize the costs.

From this simple diagram we can identify several issues that the protocol will have to address. First, the operator will have to provide a *note* whose authenticity has to

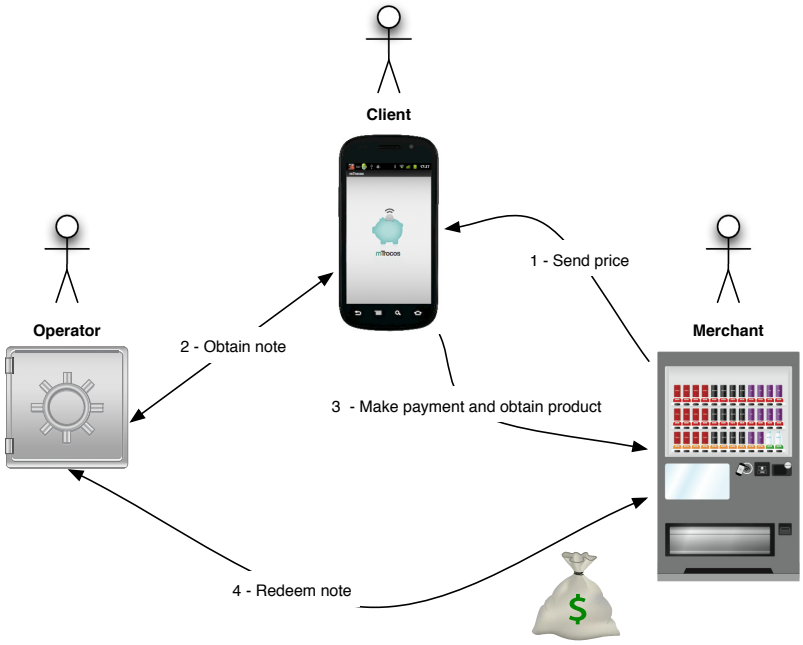


Figure 3.1: High-level description of *mTrocos*

be verified by the vending machine. However, if the operator generates this note alone or is able to see its contents, the operator can track it thereby eliminating the privacy we seek to maintain with the protocol. To resolve this issue the protocol uses the notion of *blind signatures* [11], which allows the operator to sign notes presented by the client, without seeing the content of those notes. This ensures that the payments are anonymous, the operator does not know where the note will be spent and for what, not even when a merchant tries to redeem it, and the merchant will have no additional information on the client.

Second, a merchant could in theory specify a certain unique price to make a client ask the operator for that unique amount, but this behavior can be seen by the client who can opt not to do business with this merchant. This scheme has similarities with [13, 12], but in *mTrocos* the client (mobile phone) is online during the transaction and the merchant (vending machine) is offline. Also, current mobile phones do not have a tamper-resistant secure element, or at least do not allow applications to access it, which prevents the phone form being used as a mobile wallet or possessing any kind of secret keys, so transactions have to be done with one of the parties being online.

Third, it is of course dangerous for the operator to sign something without seeing it, so a client will provide N blinded notes for the same amount but with random serial numbers. The operator will then chose one at random to keep blinded, ask for the blinding factors of the other $N-1$ notes, and will make sure that they all refer to the same amount. It can then sign the remaining blinded note and be confident that, with $1-\frac{1}{N}$ probability, it will have the same value as the others.

Figure 3.2 shows the payment operation and divides it into four distinct phases.

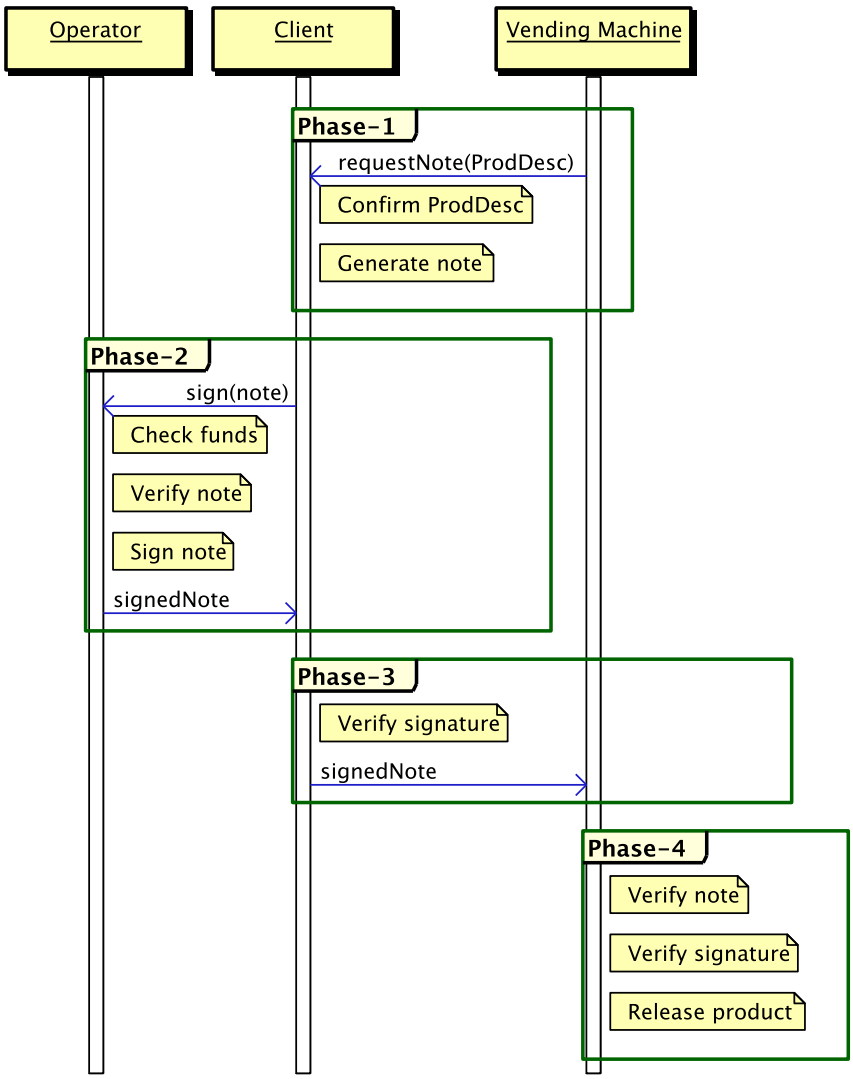


Figure 3.2: Phases of payment operation

Client	Vending machine
Pick 100 $U \in_U \{0,1\}^k$ Pick 100 serials $anon \leftarrow Anon(R, U)$ 100 $N \leftarrow Note(serial, price, anon, date())$ 100 $(N_B, BF) \leftarrow blind(N)$	$PD \leftarrow (price, desc)$ Pick $R \in_U \{0,1\}^k$ $\leftarrow PD, R$

Figure 3.3: First phase of payment operation

Each phase will now be detailed individually. The main steps of the first phase can be seen in Figure 3.3. In this phase the vending machine sends to the client the price and description of the product that was selected. Additionally, the vending machine also sends a *nonce*, R, that is to be reflected in the *note* in order to prevent replay attacks. After asking the user for confirmation of the description and price, the client creates the 100 notes, each one with a different serial number. A note contains the following fields:

Price This is the amount that the note is worth. It is of type *Float* and has to be higher than zero and smaller than the maximum note value configured for the protocol.

Serial Each note has a unique serial number in the UUID type 4 format [68].

Anon An hash of the nonce R chosen by the vending machine concatenated with a random nonce U picked by the client. U is different for each note even in the same protocol run. It ensures the vending machine that this note was generated for this request in particular but does not allow the operator to use it for tracking, since each note has a different *anon* and it is not possible to derive R from *anon* alone.

Date The creation date of the note, in the date escape format yyyy-mm-dd using the GMT (Greenwich Mean Time) timezone. Only the date is used, not the time, since storing a complete timestamp would make it easier for the notes to be correlated when they are returned to the operator. Vending machines will only accept notes generated on the the current day, and the operator will only accept notes generated V_{Max} days ago, where V_{Max} is the validity period of the notes in days. This is to ensure that the vending machine merchant will have enough time to redeem the note with the operator, and also helps manage the size of the spent notes database on the operator's side, since notes can be flushed when their validity period expires. An expiration date is not present in notes because these would have to be inserted by the client when generating them. It should not be up to the client to choose this date, so the value V_{Max} is a configurable parameter of the *mTrocos* system and known by the operator and merchant.

Each note is then blinded with a different blinding factor, BF, to produce the 100 blinded notes, N_B.

The second phase is displayed in Figure 3.4. A client possesses credentials that he uses to invoke the services of the operator and that enable the latter to identify the particular client performing the requests. The operator receives the 100 blinded notes and chooses one to keep blinded (this is the note that will be signed later). He informs the client of the note he has chosen and the client returns the 99 blinding factors corresponding to the remaining 99 notes, which are then unblinded. The operator performs the following set of checks on the unblinded notes:

Operator	Client
Pick G, $0 \leq G < 100$ $G \rightarrow$ $foreach\ (i \in [0,99] - G)\ \{$ $N_i \leftarrow unblind(N_{B_i}, BF_i)$ $Validate\ N_i\ structure$ $0 < N_i.price < P_{Max}$ $N_i.price \leq balance(client)$ $N_i.serial \notin spentDatabase$ $N_i.date = date() \pm drift$ $\}$ $[N_{BG}]_O^{-1} \leftarrow sign(N_{BG})$ $[N_{BG}]_O^{-1} \rightarrow$	$\leftarrow$ 100 N_B $\leftarrow$ 99 BF

Figure 3.4: Second phase of payment operation

Client	Vending machine
$[N_G]_O^{-1} \leftarrow unblind([N_{BG}]_O^{-1}, BF_G)$ $VerifySignature(N_G, [N_G]_O^{-1})$ $U, N_G, [N_G]_O^{-1} \rightarrow$	

Figure 3.5: Third phase of payment operation

- Ensures that all the notes are valid in terms of their structure. They must have correct values in all the fields described above.
- Ensures that they all have the same price and that this price is valid, as per the specifications above.
- Ensures that the client is allowed to request that amount.
- Ensures that they possess a valid serial number and that it is not present in the spent notes list.
- Ensures that the date on the notes is the current date, with a maximum drift of one day.

After all these checks are done, the operator will sign the remaining blinded note and send the signature back to the client. If any of these checks fail, the protocol is canceled and the user account should be suspended.

The third phase is presented in Figure 3.5. The client unblinds the blinded signature received from the operator to obtain a signed note N_G. After checking the validity of the signature — both clients and vending machines need to possess the operator's public key(s) so they can verify the signature on the notes — he sends the

Vending machine
$VerifySignature(N_G, [N_G]_O^{-1})$
$N_G.anon = Anon(R, U)$
$N_G.date = date() \pm drift$
$N_G.serial \notin localSpentDatabase$
Store N_G
Release product

Figure 3.6: Fourth phase of payment operation

Operator	Vending machine
	$\leftarrow N, [N]_O^{-1}$
$VerifySignature(N, [N]_O^{-1})$	
$N.date < date() + V_{Max}$	
$N.serial \notin spentDatabase$	
$N.price < P_{Max}$	
$spentDatabase \leftarrow spentDatabase + N.serial$	
Pay merchant	

Figure 3.7: Redeeming of *notes*

note, N_G, the signature, $[N_G]_O^{-1}$, and the corresponding U that was used to build the *anon* present on that specific *note* to the vending machine.

The fourth and final phase can be seen in Figure 3.6. When a vending machine receives a note it needs to perform a set of checks:

- Ensure that the operator's signature is valid on the note.
- Ensure that the *anon* field on the note corresponds to the hash of its chosen R plus U chosen by the client.
- Ensure that the note's serial number is not already present on its spent notes list.
- Ensure that the date on the note corresponds to the current date, with a maximum drift of one day.
- Ensure that the price on the note corresponds to the price of the product being bought.

After all these checks are done, the vending machine will store the note and release the product.

Another aspect that warrants an explanation is how merchants redeem the notes they receive from clients. Merchants will need to be registered with the operator and have unique credentials so that they invoke the operator's services with them. When the operator receives notes for redemption it will verify the credentials and perform a set of checks:

- Ensure that the signature is valid on the note.
- Ensure that the date on the note plus the validity period of notes is smaller than the current date.
- Ensure that the serial number is not present in the spent notes list.
- Ensure that the price on the note is below the protocol's configurable maximum value.

After all these checks are done, the operator will credit the merchants account with the amount equal to the price of the note and will add the note's serial number to the spent notes list. The operator periodically cleans the spent notes database removing entries which refer to notes whose expiration date has passed.

If a note is emitted, but for some reason the client is unable to send it to the vending machine, the software on the client will be able to send the note back to the operator who will store the serial number of the note and the ID of the client, and will credit his account when the validity period of the note expires. If the note is already on the spent notes list or a merchant redeems it during the validity period, the client is committing a fraud and should be suspended.

Another issue the protocol has to address is that eavesdropping the communications would undermine the privacy of the client, since an attacker would be able to track what product was being bought and possibly by whom. Therefore, the protocol requires secure channels both between the operator and the client, and between the client and the vending machine. Nevertheless, these two secure channels are different in their characteristics because for the communication between the operator and the client one can rely on a PKI to support authentication and the creation of temporary keys, but for the client and vending machine one needs to rely on an alternate solution. This is due to the fact that the number of operators will be relatively small, so it is feasible for a client to possess the public key certificates of all the operators. On the other hand, it would be very hard to maintain a list of vending machine certificates updated, so a CA (Certificate Authority) would have to be maintained per vending machine company that would sign the individual vending machine certificates. A client would then need the certificates of these CAs to verify the individual vending machine certificates. This approach has at least two problems:

- The effort to maintain this PKI would be high because certificates would have to be generated per machine and CRLs (Certificate Revocation List) would have to be kept up to date to deal with disclosures of private keys.
- It would still not be secure since a client would not know if it was talking with the correct vending machine. There is nothing to tie a validated certificate to a specific machine.

To solve this issue we resort to ad-hoc key establishment protocols, which attempt to verify the public DH (Diffie-Hellman) numbers used to establish a shared key. The chosen DH-SC protocol [10] does just that. The string comparison variant is used

Client		Vending machine
Given g^{X_A}		Given g^{X_B}
Pick $N_A \in_U \{0,1\}^k$		Pick $N_B \in_U \{0,1\}^k$
$m_A \leftarrow 1 \parallel g^{X_A} \parallel N_A$		$m_B \leftarrow 0 \parallel g^{X_B} \parallel N_B$
$c_A \leftarrow h(m_A)$		$c_B \leftarrow h(m_B)$
	$c_A \rightarrow$	
	$\leftarrow c_B$	
	$g^{X_A}, N_A \rightarrow$	
	$\leftarrow g^{X_B}, N_B$	Verify $c_A = h(1 \parallel g^{X_A} \parallel N_A)$
Verify $c_B = h(0 \parallel g^{X_B} \parallel N_B)$		
$i_A \leftarrow N_A \oplus N_B$		$i_B \leftarrow N_B \oplus N_A$
Verify OOB $i_A = i_B$		

Figure 3.8: Description of the ad-hoc key establishment

because studies with users have shown that it leads to a lower error-rate [121] with an appropriate UI (User Interface).

Figure 3.8 explains how the DH-SC protocol is used to verify the public DH numbers. This protocol relies on commitment schemes [21] to ensure that both parties commit to their public values before they are revealed, and cannot change them later. Each party chooses its public DH value and a nonce, and generates its commitment, c_A and c_B. These commitments are exchanged first, and only after that are the public DH values and nonces transmitted. An attacker cannot compute the correct commitments to send because the nonces are not yet publicly known. After the exchange, the decommitments are verified and an OOB (Out Of Band) comparison, $i_A = i_B$, takes place. If these match, both parties can generate a shared session key and be confident that they are talking with the correct entity.

If a number of protocol runs are cancelled by the user because the verification strings are different, an attack can be in place and these events should be reported by the client application to the operator, who can conduct further investigations. An hash of the public DH value concatenated with a random nonce and a fixed value is used to generate the commitments. The nonces used for the commitments are N_A and N_B, which are generated in the execution of the DH-SC protocol itself, and are reused. The DH-SC protocol uses these nonces to generate the verification strings that will be compared visually by the user. N_B, in particular, will serve an additional purpose: it will be used by *mTrocos* as the R value sent by the vending machine (see Figure 3.3).

The communication between the client and the operator can be protected by using a TLS [24] channel on top of the operator's data network. TLS can take advantage of the existing operator certificate for the establishment of the secure channel. The communication between the client and the vending machine is based on NFC in P2P mode. As for NFC, it is clear that although it has a very small range, this alone is not sufficient to ensure that no eavesdropping or MITM attacks will take place (recall Subsection 2.2.1). Akin to scams in ATM machines, it is possible to place a device

on top of the NFC antenna that blocks the signal to the outside and relay the signal through another antenna, this way performing a MITM attack. What this means is that NFC per se does not guarantee a secure channel, but the DH-SC protocol supports the computation of a shared key that can be used to build one.

As explained in Section 4.2, the Android NFC API has limitations that prevent the transmission of more than one message per LLCP session between the client and the vending machine. This poses some problems to the execution of the DH-SC protocol that requires several message exchanges. To work around this problem, NFC was used to exchange the data needed to establish a Bluetooth [6] connection on which to execute the rest of the protocol. This combines the advantage of the lower range and quickness (no pairing needed, a micro-interaction [66]) of NFC during key establishment, with the higher range, bandwidth and reliability of Bluetooth for the rest of the execution of the protocol. The increased range of Bluetooth also has benefits for usability, as the user can pull the mobile phone closer in order to confirm the purchase, and since everything is transmitted over a secure channel, security is not affected.

The `createInsecureRfcommSocketToServiceRecord()` method of the Android API, present in the `BluetoothDevice` class, allows Bluetooth connections to be established using the SPP (Secure Simple Pairing) mode, which bypasses the usual pairing step of requiring a user to enter a PIN code. This mode is vulnerable to MITM attacks but our overlying protocol, the variant of DH-SC, prevents them. Looking at Figure 3.8, the exchange of commitments will take place over NFC, the vending machine will also send its Bluetooth address and service UUID (Universally Unique ID), and then the client will have all the necessary information to establish the Bluetooth connection and continue the execution of the protocol.

3.4 Design alternatives

3.4.1 NFC only

A first design of the *mTrocos* payment protocol relied only on NFC communications (Bluetooth was not used in any part of the protocol). Due to the limitations discussed in Section 4.2, this would imply that the mobile phone would be the first to send a message, and could only send one, and a response could then be obtained from the vending machine. With these restrictions in place, the *mTrocos* protocol has to be changed in some parts, and the following protocol could be envisaged:

1. The client selects a product in the vending machine.

2. The client starts the Android application on the phone. At this time the application requests a signed blank cheque from the operator. There can be no value on this cheque because the mobile phone has no idea of what is being bought and how much it costs.

3. The mobile phone sends the signed blank cheque to the NFC peer in range.

4. The NFC peer in range, hopefully the vending machine in question, verifies the signature of the cheque and releases the product.

5. Later, the vending machine company will redeem these cheques with the operator, specifying the amount owed in each.

There are several problems with this approach:

No anonymity The operator is free to mark the cheques being issued, or save the serial number, and thus it will know how the cheque was used.

Double spending or double redeeming Once a client obtains a blank cheque, he is free to use it on several vending machines. These independent machines will have no way to tell if that cheque has already been used, and therefore it would be considered valid. On the other hand, a malicious vending machine company can try to redeem the same cheque several times, this way framing a client. The operator cannot distinguish between a situation where a client double spent and a situation of double redeeming on the part of the vending machine company.

Eavesdropping of cheques Since the NFC communication would be in cleartext, an eavesdropper can obtain the blank cheques and use them himself for subsequent purchases. Establishing a secure channel in this case would imply an existent PKI that would be preloaded on the mobile phone. A single public/private key pair would exist for this particular vending machine company, since it would be infeasible for a user to choose the right key for this particular vending machine, but the user would still need to choose the company. This would undermine the usability of the protocol and maintaing this PKI would be a very daunting task. The private key of the company would need to be on every vending machine, which meant that it would not stay private for long, and changing keys would involve changing the keys stored in the vending machines and in the mobile phones as well.

Amount specified by the vending machine The cheques themselves do not have a specific value, as the mobile phone does not know the value when the cheque is requested. Therefore, it is up to the vending machine operator to specify the correct value. This can lead to disputes when the client checks his balance, which would be difficult to solve because there is no way to tell who is telling the truth. The protocol could mandate that the vending machine response contain the signed price and description, and the client would need to forward this to the operator to activate the issued check. However, the client could just not forward the information since he already has the product. The inverse is also true, if the cheque was activated by default, the vending machine could just not send the response because it already had the blank cheque.

Relay attacks These attacks are similar to eavesdropping attacks in that an attacker has access to the communication, but establishing a secure communication channel is not enough to prevent them. An attacker may have disabled or

Figure 3.9: QR Code example

blocked the antenna in the vending machine and can relay the signal to another vending machine. The protocol is executed correctly, but the client has just bought an item on another vending machine chosen by the attacker. To prevent this, demonstrative identification has to be achieved, in order for the client to be sure that he is really talking to the vending machine in front of him.

For all these reasons the NFC only approach was abandoned. It cannot be made as secure as the *mTrocos* solution using NFC + Bluetooth and it has no usability advantage because users would need to choose the correct key to use to talk to the vending machine.

3.4.2 NFC + QR Codes

An alternative solution that could address some of the problems of the previous subsection could be based on NFC and QR Codes [54]. QR Codes have a very interesting property in that demonstrative identification is achieved: when a code presented by a vending machine screen is scanned, the user can be confident that the data received really came from that machine [73]. So an outline of a possible protocol would be:

1. The client selects a product in the vending machine, which then displays a QR Code.

2. The client starts the mobile application and scans QR Code. Since QR Codes can contain up to 4296 alphanumeric characters, it includes the product description and price, the public key of this vending machine and a nonce. An example

QR Code with this information is in Figure 3.9. Since there is demonstrative identification, the client can be sure that the received public key is really from the vending machine (no PKI needs to be deployed).

3. The client uses the application to confirm that this is the correct product and price.

4. Here the protocol follows the design of *mTrocos* as seen in Section 3.3 with respect to the communication between the mobile phone and the operator until a signed note is obtained.

5. The application instructs the client to put the mobile phone next to the vending machine so that the P2P NFC link can be established. The last message of the protocol is sent encrypted via NFC using the public key obtained from the vending machine.

6. The vending machine releases the product after checking the note.

This alternative protocol, compared with the NFC + Bluetooth approach, has the advantage that it does not depend on the user to ensure its security properties: it does not assume that the user will compare the verification string presented in the mobile application with the one presented by the vending machine. However, it has its own set of problems:

Higher cost Displays would have to be available at vending machines to allow the display of QR Codes with sufficient resolution. In the example in Figure 3.9, the QR Code contains 981 characters of text and was encoded with an ECC (Error-Correcting Code) level of L (7% error correction capability). This requires a version 19 QR Code and a 93x93 capable display. Doing a search on Alibaba [1] shows that the average price for an LCD (Liquid Crystal Display) module with at least 93x93 resolution and a viewing area of at least 10cm per side, so that readers can parse the QR Code, costs around $20-25, which is much more expensive than the Bluetooth module (see Subsection 4.3.2).

Vandalism Besides the added cost, using these larger screens on vending machines would make them more prone to vandalism. A cracked or painted screen would result in a physical DoS because it would not be possible to conduct payment operations.

Usability issues Making a client scan a QR Code with his phone has usability issues, compared with the NFC + Bluetooth solution.

For these reasons, the NFC + Bluetooth solution was used as the basis for the prototype.

3.5 Threat model

There are several scenarios that may pose a threat to the *mTrocos* protocol. Here they are detailed and it is explained how *mTrocos* deals with these situations, separating them into threats to the security of the protocol and threats to the privacy of the users.

3.5.1 Security threats

Double spending A client obtains a signed note from the operator and tries to use it to pay multiple items. To protect against this attack each iteration of the protocol involves the vending machine generating a nonce R, a 48 bit random number, which must be reflected in the signed note obtained from the client. The client will insert a hash of R concatenated with another 48 bit random number, U, generated locally (with a different U per note). This way the operator will never see the hash or the serial number that eventually reaches the vending machine making the payment untraceable. The client will send the corresponding U for the note that was signed and the hash of $R + U$ must be the one present in the note, which proves that the note was generated for this request. The client will not be able to reuse this note since on a subsequent purchase the R returned from the vending machine will be different. Brute-force attempts by the client to obtain a specific R can be quickly detected, and would be a daunting task because the vending machine also throttles the amount of times the protocol can be restarted to no more than once per second. Even if the client managed to brute-force several vending machines to obtain a particular R, it would take him more than a day, and vending machines are programmed to reject notes not emitted on the present day.

Double redemption A vending machine company tries to redeem the same note twice. This is easily detected by the spent notes list. Every note that is redeemed is inserted in the spent notes list for the duration of the validity period so any attempt to redeem a note twice will not succeed.

Erroneous note amount The client attempts to make the operator sign a note that has a different value than the one present on the unblinded notes. The chance of this attack succeeding is $\frac{1}{N}$, where N is the number of notes produced by the client and sent to the operator. In this prototype N is fixed at 100, which means that this attack has a 99% probability of being detected, and the client is suspended at the first detection, so the deterrence factor is high. Nevertheless, N can be adjusted to increase this probability further at the expense of bandwidth and computation time. Another deterrence factor is the maximum note value, which should be set to a low value because this protocol is designed for micro-transactions. This would lower even more the appealing factor of this attack.

Return spent note The client sends the note to the vending machine to make the purchase and then returns the note to the operator, claiming that the transac-

tion was not performed in order to be reimbursed. Since the credit will only be done when the validity period of the note expires, and only if no merchant redeems it in the meantime, this attack is prevented.

Rogue Android application An Android vulnerability could allow a malicious application to take control of the phone OS and monitor the *mTrocos* application, possibly obtaining the signed notes or the user PIN if the memory of the *mTrocos* application could be directly accessed during an execution of the protocol. Android applications run in separate sandboxes and, so far, the vulnerabilities found in applications have not led to exploitation of other applications precisely because the sandbox has not been breached. It remains to be seen if this sandbox will continue to withstand attacks. However, there is another vector of attack that is an application stealing the focus of the *mTrocos* application while it is on the login screen, this way capturing the user's credentials without him noticing it. This vulnerability exists in Android OS [104]. Stealing the credentials of a user allows an attacker to request notes on behalf of this user. Although there is a daily limit on the amount of money a user can request, if the attack goes unnoticed it can cause considerable damages to the user. One way to improve the detection of this attack is for the operator to notify the user whenever it receives a note request from him, for example, using an SMS. A user that is not executing a payment, and receives an SMS notifying him that he has requested a note, should assume that his credentials have been stolen, contact the operator and change its password.

DoS on the operator On each successful protocol run, the operator has to unblind 99 notes and sign 1 note. When a merchant tries to redeem notes all their signatures have to be verified. As these are expensive cryptographic computations, care must be taken to disallow clients and merchants from performing DoS attacks on the operator. As all these operations are authenticated, the deterrence factor is high because it is easy to identify who is responsible for the attack and block it. Nevertheless, it is important to consider what other limits could be in place. With regard to the client facing service, the client can request notes as long as enough balance exists to cover the amount and the daily maximum amount has not been exceeded. With a daily maximum of 100, and requesting 0.01 notes, a client can perform 10000 requests. Taking into account that, a 2048 bit RSA signing operation takes roughly 10ms on today's hardware, and the unblinding of a note, which is comparable to a 2048-bit RSA signature verify operation, takes roughly 1ms on today's hardware, a single client could use approximately 1090s of processing time of the operator per day ($10000 \times 1ms \times 99 + 10000 \times 10ms = 1090s$). A simple throttling mechanism can be put in place to limit a client to requesting no more than 10 notes per minute, since the average time the protocol takes to run is 20s (5s for the actual message exchanges and 15s for user interaction) this limit seems reasonable. A client that frequently comes up against this limit should be investigated. As for the merchant facing service, as long as valid notes are presented and since

verifying a signature is less expensive compared to generating one, no limits should be needed. The merchant is limited to redeeming notes he has received, so the limits on note generation described earlier take effect, and notes have a limited validity period so a merchant cannot stock up on a large numbers of notes to be able to cause problems when redeeming them all. Nevertheless, the operations performed by the operator are highly parallelizable, so it is easy to scale should the need arise.

DoS on vending machine storage Even though notes are stored in encrypted form in the vending machine and it is tamper-resistant to some extent, DoS attacks on the note storage should be considered. These do not allow an attacker to steal money, but can erase the digital notes from the storage medium. In the case of SD cards, which is the suggested storage medium, it may be possible to externally erase them through the use of a EMP (ElectroMagnetic Pulse) directed at the vending machine casing, for example, using a Taser. Studying these aspects is beyond the scope of this work.

Man-In-The-Middle and relay attacks An attacker manages to insert himself between the antenna of the mobile phone and the antenna of the intended vending machine, blocking the signal between both, but relaying the signal to another vending machine. Since vending machines are in public spaces, they are more subject to these kinds of physical attacks, as opposed to POS (Point Of Sale) machines which are in stores and more guarded by their owners, who have no incentive to engage in fraud against themselves. This attack attempts to make a client pay for an item on another vending machine. As part of the DH-SC protocol, a verification string is produced on both sides that must match. As seen in Figure 3.8, one of the steps of the protocol involves the user comparing the string displayed on the mobile phone with the one displayed on the vending machine. As it is infeasible for an attacker to produce the correct verification string on both sides, if it matches, it means that the client is talking with the correct machine and that no MITM attack is taking place. One of the assumptions of the *mTrocos* protocol, as seen in Section 3.2, is that users will indeed do the comparison and signal when the verification strings are different.

Data corruption Since the communication between the vending machine and the client is done using wireless protocols, NFC and Bluetooth, they are susceptible to jamming. While NFC requires this to be done at a very close range, Bluetooth jammers exists that have a range of 15m. *mTrocos* is thus susceptible to DoS attacks on this part of the protocol, which do not result in a breach of security, but can prevent a successful run of the complete protocol.

Data modification or insertion Since *mTrocos* establishes secure channels between the operator and the client, and between the vending machine and the client, it is not susceptible to attacks that attempt to alter the transmitted data.

3.5.2 Privacy threats

Tracking of notes Even though the operator is unable to see the note being signed, it can try to guess some of the contents to match them when the note is redeemed. For example, an unusual price value can be used to track the note. However, since the vending machine has no way to identify users, it would have to generate these unusual prices for every user, which would raise suspicions. The serial number of the note is completely random, so it is not possible to be used for tracking. The *anon* field, which is based on the R sent by the vending machine to the client, could be used for tracking because the blinded note would have it as well. To address this issue, R is hashed with a random value U different per note (as explained in Section 3.3). The date field is also a possible target because all the notes sent by the client in a request have the same value. As the number of users grows, the amount of notes generated on a single day will make it very difficult to track a specific note when it is redeemed later. However, if a merchant colludes with the operator, the vending machine can store the exact time when a note was used and the operator can store the note creation time (which should be more or less the same). If then the merchant supplies this time with the note in the redeeming phase, it can help the operator in matching notes with users. Again, as the number of users grows, this attack will be much harder to perform.

Eavesdropping Eavesdropping the communication between a client and the operator would reveal the PIN of the user and the signed note, but it would require breaking the encryption used in the TLS tunnel. This however, is very difficult to achieve as long as secure cyphers and hash algorithms are used. As for the communication between the client and the vending machine, it could reveal the signed note or result in a loss of privacy, but the DH protocol ensures that both parties compute a shared key that is unknown to a listening party. Since MITM attacks are prevented by the DH-SC protocol, the computed key is indeed safe.

3.6 Classification

Kreyer et al. [63] aggregated a set of properties to classify mobile payment systems, and grouped them into three categories: strategic questions, participants and operational issues. Below, the *mTrocos* protocol is examined in light of each of these properties.

Strategic questions

- Payment scenarios: stationary merchant. It is tailored to the vending machine scenario and not to P2P or online commerce.
- Payment heights: micropayments. Macropayments are not supported due to the limit on each note's value. Picopayments are not supported because the note value has to be a multiple of 0.01.

Participants

- Involved parties: customer, merchant and telco. No banks or special intermediaries are part of the payment operation.
- Receiver of customer data: none. The protocol ensures anonymity.
- Pre-registration needed: yes. Each user must already be a client of the operator and for this he must have previously registered himself.
- Technology required: special payment software. The user has to install the *mTrocos* Android application, or it comes pre-installed, in order to use the protocol. SMS, WAP or dual-card phones are not needed.

Operational issues

- Basis of payment: token-based. A form of digital cash is used, signed electronic notes, even though each customer has an account with the operator.
- Payment frequency: pay per product unit. A flat price is charged for goods or services and no provision is made to support subscription-based services or usage-based payments.
- Deduction time: pre-paid, post-paid. This will depend on the type of contract the customer has with the operator, which can be a pre-paid plan or a post-paid one.
- Method for settlement: telephone bill. Since it is the operator that is providing the service, it makes sense to use the normal settlement method, the telephone bill. For the user this is practical because he is already used to pay those bills.

Chapter 4

Implementation and evaluation

This chapter discusses the implementation and evaluation of the *mTrocos* protocol, as well as some considerations about the current state of the NFC API present in the Android OS.

4.1 Implementation

This section presents in more detail the implementation aspects of the *mTrocos* protocol, including: the format of the PDUs used in each of the communication channels; the size of the nonces required by the *mTrocos* protocol; the chosen crypto algorithms and values that were given to the configurable parameters.

Figure 4.1 illustrates the architecture of the developed implementation. Two laptops were used to simulate the operator and the vending machine. The mobile application can either use the 3G network, or an available WiFi connection to reach the operator's services. In both cases, a TLS tunnel is used to establish a secure channel. The laptop simulating the vending machine has two dongles, one for NFC and the other for Bluetooth.

On the Bluetooth connection between the client and the vending machine and on the TCP (Transmission Control Protocol) connection between the client and the operator, the PDUs will be formed by a 4 byte unsigned integer in network-byte order,

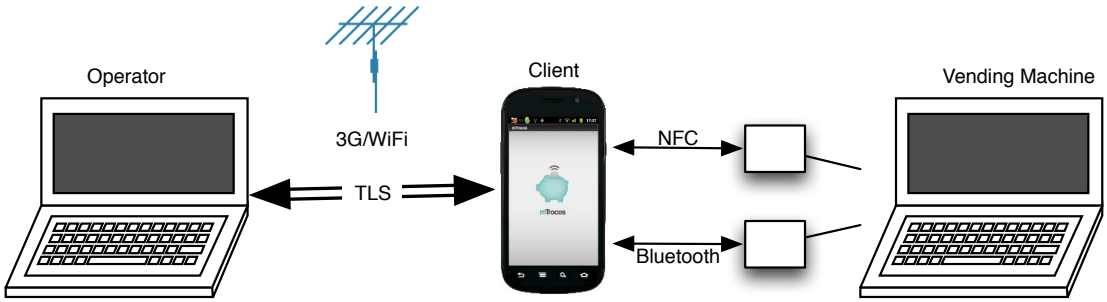


Figure 4.1: Architecture of the prototype

which represents the size of the payload in bytes, followed by the payload itself as a JSON (JavaScript Object Notation) [19] byte array. On the NFC connection, NDEF messages are exchanged, containing the MIME type *application/mTrocos* and one record with a JSON byte array. JSON is a text-based lightweight and compact way to represent data structures, which is available for most programming languages, and was used to encode the protocol messages. When binary data needs to be transmitted, such as nonces or public keys, they are encoded using Base 64 [59].

The crypto algorithms used in the protocol are the following. The secure channel is based on AES [79] with a 256 bit key — unbroken in its full 14 round version — in CBC [27] mode. For the hashes, the SHA-256 [80] algorithm — unbroken in its full 64 round version — is used. The nonces R and U have 48 bits. As for the DH parameters, a prime P with 8192 bits was chosen and for the generator G the value *2* was selected. In order to use the shared key computed by DH for the AES algorithm, which takes a 256 bit key, a SHA-256 hash is taken of the computed key. The operator's key is a 2048 bit RSA [101] key, which has been deemed secure until 2031 [4]. For the blind signatures, the code was initially developed from scratch in order to make sure that the mobile phone had the computing power needed to perform the calculations (see Subsection 4.3.1). After these tests, it was decided to change to the *RSABlindingEngine* of the BouncyCastle [116] library to generate the blind signatures. Relying on a tested public implementation of a cryptographic algorithm is generally a better idea, and this one has the advantage of generating standard signatures that can be verified by most crypto libraries. The BouncyCastle implementation signs a SHA-1 digest of the note, which is more secure since it is much harder to be tricked into signing something dangerous. However, one disadvantage is that, since the blinded data is already an hash of the original data, it is not possible for the operator to unblind a note given the blinding factor. The operator has to receive the original notes along with the requested blinding factors, so that he can blind the notes himself and compare with the blinded versions received earlier. If they match, it can be sure that the client is not lying and that these are the original notes that were received blinded. Another disadvantage is that these standard signatures contain some metadata, such as the ASN.1 DigestInfo structure, which can have parameters that could be employed for tracking. Studying this possibility will be left for future work.

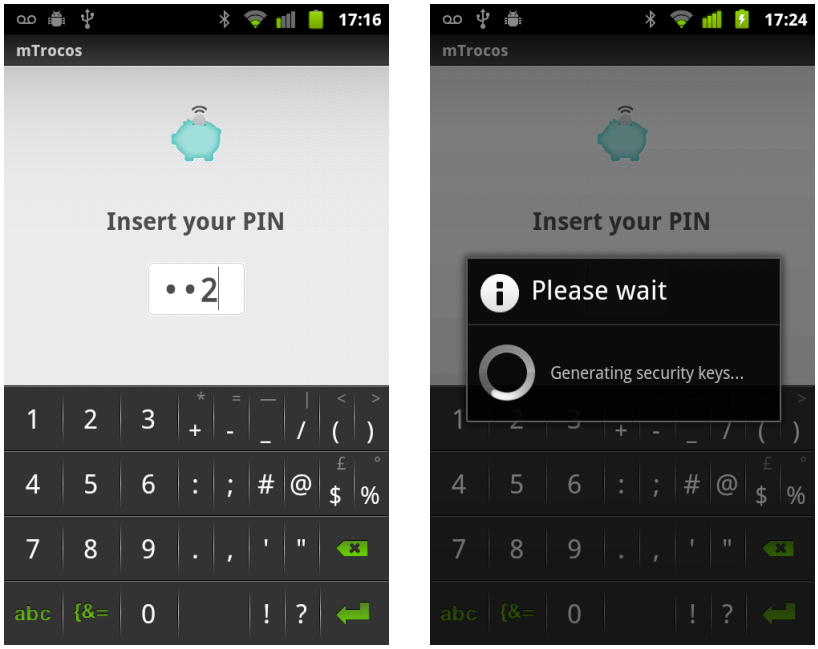
Table 4.1 shows the configuration values of the protocol that were used in the implementation. The maximum note value was set to 20, the validity of the notes to 7 days, and the daily maximum a user can spend to 100. As for the k value in the DH-SC protocol, a balance must be reached between security and usability. Higher values of k ensure that an attacker has a lower probability of breaking/influencing the communications. On the other hand, it will also lead to longer verification strings that must be compared by users, resulting in more user errors or users bypassing that step, which undermines security. The probability of an attacker succeeding against the DH-SC protocol is bounded by $n\gamma 2^{-k}$, where n is the number of vending machines tampered by one or more attackers and γ is the number of protocol runs monitored in each one or, in other words, the number of payment operations performed by users on a vending machine and seen by an attacker. Assuming that 2^{10} vending machines

System variable	Value
Note validity period (V_{Max})	7 days
Maximum PDU size	4,294,967,295 bytes
Secure channel crypto algorithm	AES/256/CBC
Hash algorithm	SHA-256
Nonce size	48 bits
DH prime size	8192 bits
DH generator value	2
Asymmetric crypto algorithm	RSA
Asymmetric crypto key size	2048 bits
DH-SC k value	30
Maximum note value (P_{Max})	20
Maximum daily value per user	100
Number of blind notes generated	100
Maximum clock drift	1 day
Maximum number of note requests	10 / user / minute

Table 4.1: Value of *mTrocos* system variables

are controlled by attackers, and that 2^{14} payment operations are performed at each machine, and with the value of k equal to 30, the probability of the protocol being successfully broken after attackers see 2^{24} runs is at most $n\gamma 2^{-k} = 2^{-6}$. In a worst case scenario where users are continually making payment operations on a vending machine and assuming that each payment operation takes on average 20 seconds, 2^{10} vending machines would have to be physically tampered for more than $2^{14} \times 20s = 91\, hours$ without anyone noticing it to achieve that probability of success. Additionally, this success would lead to a signed note of at most 20 in value. For all these reasons, the value of $k = 30$ was chosen because it is a good compromise between security and usability.

In order to produce readable text from the binary verification strings, so that it can be compared by users in the course of the DH-SC protocol, the Base 64 encoding with no padding is used. This encoding has a compact output and all the characters can be represented on the limited displays of current vending machines. Another option was to use the method described in [46], which relies on a pre-built dictionary of 2^{11} entries, to transform a binary string into 1-4 letter words. As an example, the same 30 bit data produces *K6irO* as Base 64 output and *BHOY GENT HURL* using the dictionary method. Since the strings to compare become much larger and this dictionary would have to be language dependent, without any significant gain in usability, the Base 64 approach was chosen. If one were to build a dictionary of 2^{15} entries, the transformation of the 30 bit data would produce just two words instead of three, but in the case of the English language the dictionary would require listing words of up to seven characters. Since this would have a negative impact on usability, this approach was not further investigated. Had the vending machines been equipped with higher quality displays, other visualization mechanisms such as hash



(a) PIN entry (b) Key generation

Figure 4.2: Layout of the *Authorizer* activity

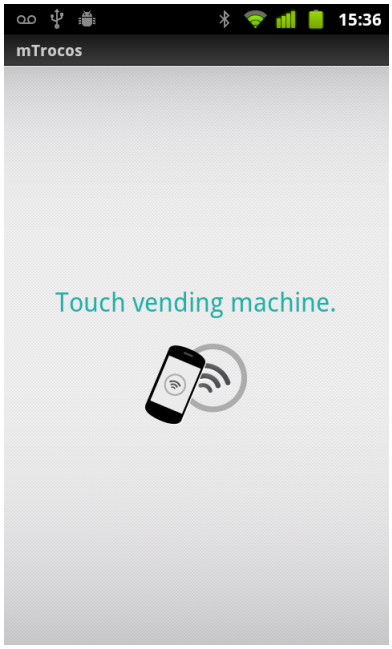
visualizations [97] could have been tried. These solutions could allow the comparison of the whole public DH numbers instead of just a smaller verification string. However, current vending machines are not capable of displaying these graphics.

4.1.1 Client

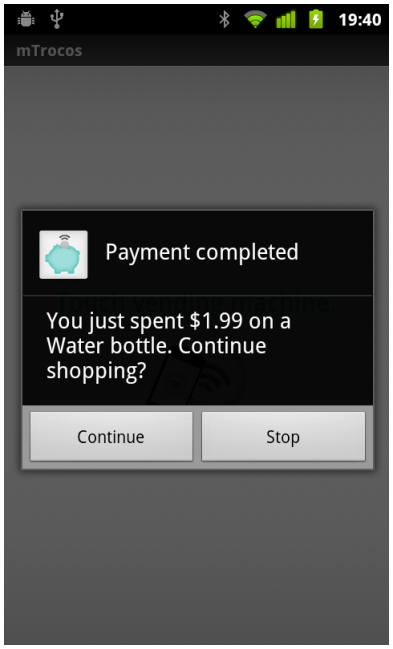
An Android client application was written to run in a Nexus S phone, which has NFC support. Android applications are written in Java [94] and organized in Activities. The *mTrocos* Android application consists of three activities:

1. *Authorizer* — Figure 4.2 shows the layout of this activity. It asks the user for a PIN on startup and verifies it with the operator. The PIN prevents someone unauthorized from using the *mTrocos* application on another person's mobile phone, and it will also be used to invoke the services of the operator later on. This activity also generates the key pair, random N_A and commitment values used in the DH-SC protocol during the NFC P2P exchange (see Figure 3.8). Both the verification of the PIN and the generation of the key pair are done by threads separate from the main application thread, in order not to block the UI.

2. *Handshaker* — Figure 4.3 shows the layout of this activity. It calls the method `NfcAdapter.enableForegroundNdefPush()` to register the initial commitment message. The Android OS will send it as soon as the NFC handshake is completed and the LLCP link is established. This activity also registers the in-

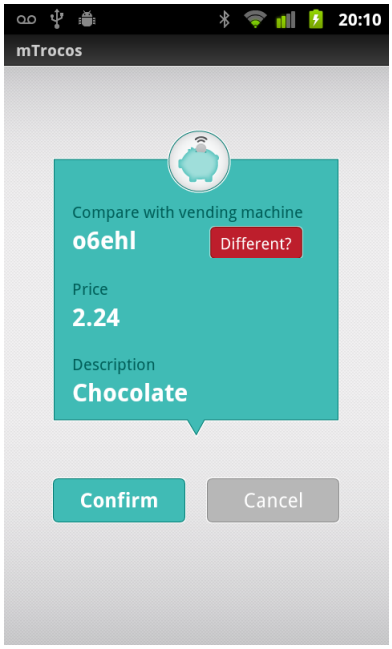


(a) Waiting for handshake

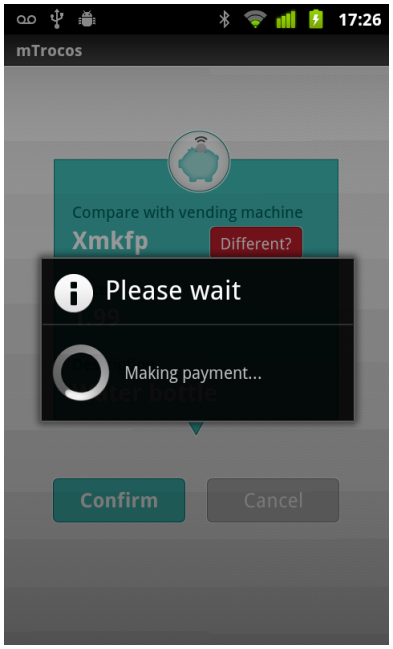


(b) Success message

Figure 4.3: Layout of the *Handshaker* activity



(a) Confirm details



(b) Executing payment phase

Figure 4.4: Layout of the *Bluestarter* activity

tent to receive NDEF messages with the MIME type *application/mTrocos* using the method `NfcAdapter.enableForegroundDispatch()`. These responses will contain the commitment of the vending machine and the needed information to establish the Bluetooth connection, such as the Bluetooth address and service UUID, and will be delivered to the *mTrocos* application as soon as the Android device receives them. This activity also shows the success dialog, when a payment is completed, asking the user if he wants to continue shopping or quit the application.

3. *BlueStarter* — Figure 4.4 depicts the layout of this activity. It establishes the Bluetooth connection and performs the rest of the protocol driving the *BluetoothService* class, presenting a UI when necessary. It turns on Bluetooth, if it is off, and turns it back off at the end in that case. It also establishes a TLS over TCP connection to the operator to request the notes. It exchanges the decommitments, verifies them, and asks the user for confirmation of the verification string and product purchase. Then, it generates the blinded notes and requests a signed note from the operator, sends it to the vending machine, and awaits confirmation. Both the communications with the vending machine and with the operator are handled in threads separate from the main application thread, in order to free the UI and interact with the user.

By design, the Android application does not start automatically when an appropriate NDEF message is received. This is for two reasons:

- The Android device needs to register the intent to send a message before the LLCP link is established, otherwise it will not be able to send a message later to that same P2P partner (as explained in Section 4.2).
- If the application started automatically, a rogue application could also register the same intent, which would cause the Android OS to ask the user for which application to use. If the user was tricked into choosing the rogue application, it could emulate the appearance of the *mTrocos* application and thus capture the PIN used to invoke the operator's services.

Since Android already comes with a crippled version of the BouncyCastle libraries, there were problems including a more recent version with the *mTrocos* application because the JVM's (Java Virtual Machine) class loader would get confused and try to load the wrong classes. To work around this problem, the variant SpongyCastle [120] was used, which is the same library but with a different name.

4.1.2 Vending machine

The vending machine simulator runs on a PC. It was written in Python [99] because there already existed an NFC library, *nfcpy* [119], that had a promising implementation of the LLCP protocol. It also supported desktop NFC readers with the PN533 [91] chip, such as the SCL3711 reader from SCM [105], that was going to be used

Figure 4.5: Vending Machine UI

in the prototype. Several USB (Universal Serial Bus) bugs were encountered, such as [67], which made development tricky, but a configuration was found that minimizes the occurrence of these errors. In the end, we switched to the Sony RC-S360 NFC reader [110], which is also based on the PN533, but is much more stable. With the LLCP implementation done, what was left to do was the development of the NPP support, which was then contributed to the project [20]. Bluez[7] was employed for the Bluetooth support, along with the corresponding Python bindings PyBluez [49]. A Bluetooth agent was also needed to perform the SPP pairing on the side of the vending machine, so the *simple-agent* that comes with the PyBluez package was adapted to this function. The Bluetooth dongle that was utilized was an IOGEAR USB 2.1 Bluetooth Micro Adapter GBU421 [52]. To implement the crypto operations — AES and DH — the OpenSSL [117] library was used with the corresponding Python bindings M2Crypto [95]. To provide a minimal graphical interface for simulating the vending machine UI, we resorted to the wxWidgets [109] library and the corresponding Python bindings wxPython [26]. Figure 4.5 depicts the product choosing screen.

At this moment, the prototype does not implement note storage, but in a real world scenario a vending machine could possess the public key of the merchant and encrypt the notes before storing them in a SD card. Vending machines are built to be tamper-proof, but even if an attacker could access the storage he would not be able to decrypt the saved notes. At most, he could access the spent notes database, which would have the notes serial numbers and dates. He could alter the stored merchant's public key to his own public key, so notes stored after that would be encrypted in a way that only the attacker could decrypt. In any case, this attack can be prevented because vending machines should at least detect tampering, and then either warn the owners or stop functioning.

4.1.3 Operator

The operator simulator runs on a PC and consists of a server that exposes a TCP endpoint. A server application written in Java using Apache MINA [115] was envisaged. MINA is an an event-driven network application framework that can serve a large number of clients simultaneously.

The application would also be connected to a storage database such as MySQL [77] or Cassandra [114] to store the spent notes database. A separate worker process can run everyday and clean the database by removing notes that are outside their validity period. By having a relatively small validity period, 7 days, we can prevent this database from growing too large.

There was also the option of implementing this service as a REST [32] web application over HTTPS (HyperText Transfer Protocol Secure), but since there are two steps involved, some state would have to be stored between the two HTTP requests, which would add complexity. However, if the operator already has a pool of application servers deployed, this can turn out to be a more attractive solution. In this case Memcached [74] servers can be used to store the blinded notes encrypted with the user's PIN, while waiting for the response from the Android application that contains the 99 blinding factors.

The simple TCP server approach, using a TLS filter that comes with MINA, was the approach chosen. The BouncyCastle library was used for the crypto computations. A protocol encoder/decoder was implemented along with the handler, which serves three operations: login, request note and receive blinding factors. For the prototype, a storage backend was not connected, so the user's credentials were hardcoded. Since the note redeeming and unspent note submission are also not implemented, a spent notes database was not needed. In a real world scenario, a user registration process would also be needed in order for users to have associated PINs. This process would also ensure that the same person could not register multiple accounts to try to brute-force the *mTrocos* protocol and obtain a signed note with a *price* value different than the one debited from his account. Even though there is a low maximum note price, 20, this is another deterrent.

4.2 Current Android NFC implementation pitfalls

The current implementation of *mTrocos* is based on Android version 2.3.4. This is a maintenance release that adds no new functionality to the NFC API and only provides bug fixes. In version 2.3.3 [40], the Android NFC API was augmented with support for reading and writing most standard tag types, transceiving raw ISO-DEP (International Organization for Standardization) data to compatible tags, support for NDEF and limited support for P2P. The functionality of card emulation was purposely left out from Android. Card emulation enables a reader to emulate a tag to other legacy readers, but since most hardware would not be able to emulate all of the tag types, it would be confusing for applications. Additionally, tags have very limited space and they would quickly become a contended resource by running applications.

NFC P2P in Android means establishing a LLCP link, in initiator or target mode, and transferring NDEF messages using the NPP protocol. The NPP protocol is relatively simple: it describes a service that registers itself under the name *com.android.npp* and that receives NDEF messages. In the current version 1.0, the server only receives one NDEF message per connection, with an action code of "0x01", and does not respond to the client. This means that in the normal case, where two devices want to exchange several messages in NFC P2P mode, they have to establish a new connection to each other's NPP service for each message they need to send.

As for sending NDEF messages in Android applications, the foreground activity can register one message to be pushed to the other device when the LLCP link is established, but this has several problems:

- Only one message can be sent per LLCP link establishment. Whenever a LLCP link is brought up, the *NdefPushClient* Android service checks to see if there is a message to transmit; in the affirmative case, the service sends the message and then exits. If another NDEF message is registered for transmission this will only happen after the current LLCP link is torn down and a new one is brought up. Therefore, the protocol is not suitable for challenge-response scenarios between two Android devices, since it is not possible to first receive a message and only after send a response. This is in line with the Android developers' idea of NFC being used for *instant gratification* and *zero-click interactions*. They do not want the user to have to choose any option and an action should immediately take place, also taking advantage of NFC's speed.

- Applications are not notified when a message has been sent. This coupled with the point above, means that two Android devices communicating in P2P mode cannot know if their message was received, unless some other communication channel is used, such as Bluetooth. In fact, using NFC for exchanging the needed information to quickly setup a Bluetooth connection is suggested by the Android developers as a best practice, and standards exist for this procedure both from the Bluetooth side[1] and from the NFC side [86].

- The NDEF message has to be built when the Activity starts, in its `onCreate()` method and registered with the `NfcAdapter.enableForegroundNdefPush()` call, even if it will only be sent some time later when the state of the application can already be different. It would be better if a callback method could be defined, such that it could return an up-to-date message when it is actually about to be transmitted. This functionality will be present in the next Android version 4.0 [96].

As for receiving NDEF messages, an application can signal in the Android manifest file the intent to be notified when a tag is discovered or a NDEF message is received. This intent can be specific to a certain type of NDEF messages, all NDEF messages, or tags with a certain technology or group of technologies. When dispatching these notifications, Android will check for registered applications in that order,

[1] Simple Pairing with the OOB association model.

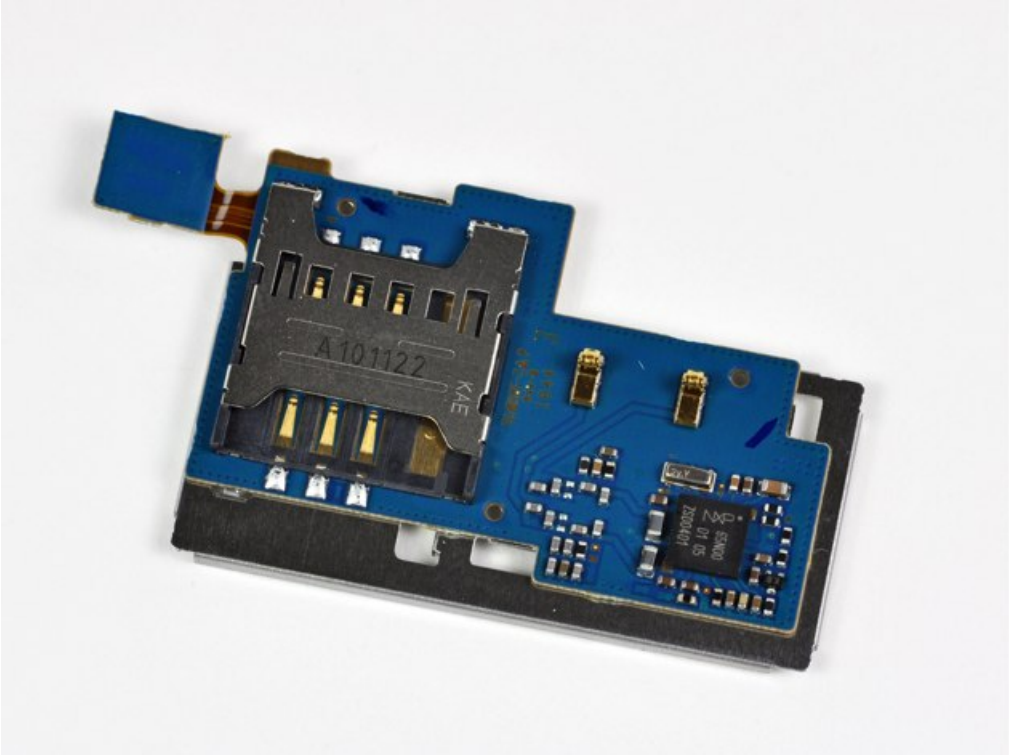


Figure 4.6: PN65N present on Nexus S (source: iFixit.com)

and if multiple applications are found, a dialog is shown to the user to choose which application he wants to use. The foreground application can prevent this dialog from appearing by using `NfcAdapter.enableForegroundDispatch()` and again specifying what it is interested in, this way Android will immediately dispatch these notifications to this application, and not to others which may also be interested in them. While an Android device is not able to send more than one message when a LLCP link is established, it is able to receive multiple messages. So if the other peer is not an Android device, it can use the NPP protocol to push multiple messages in the same LLCP link, although each in a separate connection, and the Android peer will correctly receive them.

Some Android handsets, such as the Nexus S [45], contain a secure element chip. In the case of this phone it has a PN65N chip from NXP that combines a PN544 NFC controller and an embedded SmartMX secure element (see Figure 4.6). This secure element is a tamperproof device that can store sensitive data, such as private keys, and that can also be used for card emulation. Google Wallet [44] employs this secure element to store the credit card details and it is of great use for mobile payments solutions that intend to run on mobile phones. However, the Android API does not provide access to the secure element functionalities and the reason is that the secure element is a very limited resource that cannot hold a large amount of data and so would be highly contended by applications. Also, in order to access the secure element, applications would need to authenticate themselves to it and if the number of failed authentication attempts exceeded a predetermined value, some secure elements

```
D/NfcEnabler( 1345): Setting NFC enabled state to: true
D/NFC JNI ( 1256): Start Initialization
W/NFC JNI ( 1256): phLibNfc_Mgt_ConfigureDriver() returned 0x0032[
    NFCSTATUS_ALREADY_INITIALISED]
E/NFC JNI ( 1256): phLibNfc_Mgt_Initialize() returned 0x0032[
    NFCSTATUS_ALREADY_INITIALISED]
D/NFC JNI ( 1256): Terminating client thread...
W/NfcEnabler( 1345): Error setting NFC enabled state to true
```

Figure 4.7: NFC subsystem hang

```
I/DEBUG   (12683): *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
I/DEBUG   (12683): Build fingerprint: 'google/soju/crespo:2.3.4/GRJ22/121341:user/
    release-keys'
I/DEBUG   (12683): pid: 114, tid: 13696  >>> system_server <<<
I/DEBUG   (12683): signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr deadd00d
I/DEBUG   (12683):  r0 fffffe84  r1 deadd00d  r2 00000026  r3 00000000
I/DEBUG   (12683):  r4 800a5600  r5 00448030  r6 800a5600  r7 0077da08
I/DEBUG   (12683):  r8 00000000  r9 00000000  10 4bfa6d50  fp 00000000
I/DEBUG   (12683):  ip 800a570c  sp 4bfa6cb0  lr afd191d9  pc 80046240  cpsr 20000030
```

Figure 4.8: Android system crash

erase themselves or execute an auto-destroy operation. For these reasons, it is not expected that the Android API will allow access to the secure element in the future. There are references to the secure element in the source code [113] and rebuilds of the Android source code have been made that expose the secure element functionality, but this requires users of these applications to install modified ROMs (Read-Only Memory) of the Android system, something which is not viable for most users.

During this investigation, we came across several implementation bugs on Android's NFC stack. Some have been corrected and the fixes will appear in the next version [33], while others remain unsolved because no reference to them was found on Android's issue tracker. The consequences ranged from a simple connection retry, to a restart of the NFC service, to a complete restart of the mobile phone. However, since some of the errors resulted in *SIGSEGV* exceptions, the possibility of causing memory corruption and buffer overflows through the NFC interface cannot be ignored, which can lead to security vulnerabilities. Nevertheless, these errors were sporadic and could not be reproduced reliably. Examples of log output when these errors manifested themselves can be found in Figures 4.7, 4.8, 4.9, 4.10 and 4.11.

```
E/NFC (17923): NFC service dead - attempting to recover
E/NFC (17923): android.os.DeadObjectException
E/NFC (17923): at android.os.BinderProxy.transact(Native Method)
E/NFC (17923): at android.nfc.INfcAdapter$Stub$Proxy.disableForegroundDispatch(
    INfcAdapter.java:549)
E/NFC (17923): at android.nfc.NfcAdapter.disableForegroundDispatchInternal(NfcAdapter
    .java:530)
E/NFC (17923): at android.nfc.NfcAdapter.disableForegroundDispatch(NfcAdapter.java
    :518)
E/NFC (17923): at pt.sapo.p2ptester.P2PTest.onPause(P2PTest.java:50)
```

Figure 4.9: NFC service death

```
E/NFC JNI (  208): phLibNfc_SE_SetMode() returned 0x00ff[NFCSTATUS_FAILED]
E/NFC JNI (  208): emergency_recovery: NFC stack dead-locked, please show to npelly
D/NFC JNI (  208): Discovered P2P Target
E/NFC JNI (  208): emergency_recovery: NFC stack dead-locked, please show to npelly
W/ActivityManager(  109): Timeout of broadcast BroadcastRecord{40a702a0 com.android.
    nfc.action.INTERNAL_TARGET_DESELECTED} - receiver=android.os.BinderProxy@40566ea8
    , started 10027ms ago
W/ActivityManager(  109): Receiver during timeout: BroadcastFilter{40762960
    ReceiverList{40765300 208 com.android.nfc3/1025 remote:40566ea8}}
I/Process (  109): Sending signal. PID: 208 SIG: 9
I/ActivityManager(  109): Process com.android.nfc3 (pid 208) has died.
I/ActivityManager(  109): Start proc com.android.nfc3 for restart com.android.nfc3:
    pid=24702 uid=1025 gids={}
I/ServiceManager(   69): service 'nfc' died
```

Figure 4.10: NFC stack deadlock

```
D/NFC-HCI (17250): phOsalNfc_RaiseException() called
E/NFC-HCI (17250): HCI Timeout - Exception raised
E/NFC-HCI (17250): Force restart of NFC Service
I/DEBUG   (17249): *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
I/DEBUG   (17249): Build fingerprint: 'google/soju/crespo:2.3.4/GRJ22/121341:user/
    release-keys'
I/DEBUG   (17249): pid: 17250, tid: 17259  >>> com.android.nfc3 <<<
I/DEBUG   (17249): signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr deadbaad
I/DEBUG   (17249):  r0 00000027  r1 deadbaad  r2 a0000000  r3 00000000
I/DEBUG   (17249):  r4 00000001  r5 00000000  r6 80549348  r7 44a5ce4c
I/DEBUG   (17249):  r8 ffffff44  r9 44a5cea8  10 00000311  fp 44a5cdd4
I/DEBUG   (17249):  ip afd46668  sp 44a5cd98  lr afd191d9  pc afd15ca4  cpsr 60000030
```

Figure 4.11: NFC subsystem crash

4.3 Evaluation

The evaluation of the *mTrocos* protocol starts with an analysis of the performance on a mobile phone, which is a platform with limited computational power. Next, the cost of the hardware needed for deploying *mTrocos* on vending machines is approximated. This is followed by a series of tests with users in order to collect information about the user experience with the protocol. After that, the results of the tests will be analyzed and the characteristics presented in Section 3.1 will be revisited, to ensure that *mTrocos* fulfills them.

4.3.1 Performance

One of the concerns while developing *mTrocos* was related to the computational power needed to perform the calculations in the mobile phone. Since asymmetric crypto is used, which is more computationally expensive than symmetric crypto, it could be the case that it would take too long for the mobile phone to respond, and thus degrade the user experience. It was therefore imperative to test the time needed to perform the blind signature operations. The blind signatures code was initially developed from scratch, because it would be easier to pinpoint the causes of delays. The following times were obtained for the generation of 100 blinded notes (a 2048 bit RSA key is used in the blinding step):

- Generating 100 blinding factors — $<$ 300 ms
- Generating 100 anonymizers (U) — $<$ 1 ms
- Generating 100 notes[2] — $<$ 100 ms
- Blinding 100 notes — $<$ 350 ms

From these values it is apparent that the 1GHz Cortex A8 Hummingbird processor available in the Nexus S is capable of performing the needed computations for the *mTrocos* protocol.

4.3.2 Hardware cost

Each vending machine will need to have NFC support. According to Digikey [25], a PN544 NFC chip from NXP costs $8.3 per unit and each vending machine will need to have one. By looking at vending machine prices displayed in Vending World [123], they vary between $800 and $2000, so the price of the NFC chip does not increase the cost of a vending machine by a large margin. Even comparing just with the the typical coin and bill payment modules, these vary between $80 and $300, so an NFC payment module can be made at a lower cost and, since it has no moving parts, has lower maintenance. If the vending machine does not provide digital storage for the notes, an additional component is required for this function, but a simple SD card reader/writer and a SD (Secure Digital) card will fit the purpose. A search on Alibaba [1] for Bluetooth adapters revealed the average price to be $1-8, depending on the quantity bought, so it also does not seem like an impediment.

4.3.3 User tests

The goals of these tests are to:

- Ensure that users are able to perform the payments operations.
- Assess the usability of the protocol with real users.
- Ensure that users actually compare the verification strings, which is an assumption of the protocol and very important for its security properties.
- Collect the users' opinions about the protocol and their inclination to use it.

These tests were done using ten subjects, four who were native english speakers, and six who were native portuguese speakers. The test procedure given to them can be seen in Appendix A. They were to perform a payment operation five times, and in the end answer four questions. In the fourth iteration the verification strings were different in one character, and in the fifth iteration they were completely different.

[2]Includes an SHA-256 operation per note.

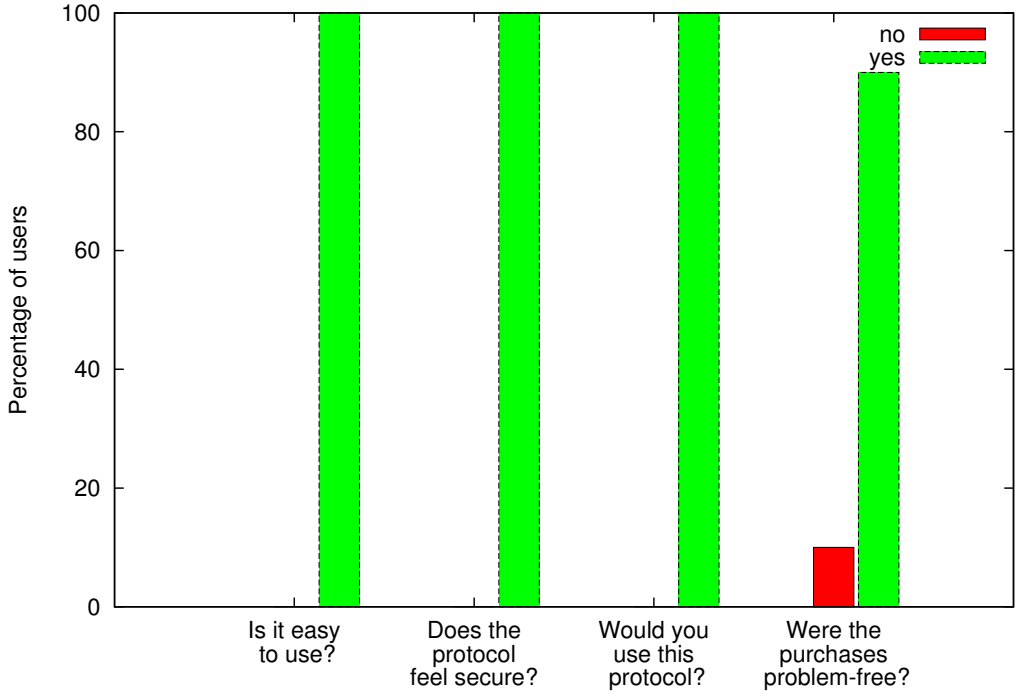


Figure 4.12: Answers to test questions

```
I/BluetoothEventLoop.cpp(  109): agent_event_filter: Received method org.bluez.Agent:
    OutOfBandAvailable
E/BluetoothEventLoop.cpp(  109): onCreateDeviceResult: D-Bus error: org.bluez.Error.
    AlreadyExists (Device already exists)
E/BluetoothEventLoop.cpp(  109): onDiscoverServicesResult: D-Bus error: org.bluez.
    Error.InProgress (Discover in progress)
D/BluetoothService(  109): Cleaning up failed UUID channel lookup: 00:02:72:20:A4:32
    a35f6a2f-3e07-4644-924a-2b4461d3730b
E/BluetoothService( 5457): unable to connect()
E/BluetoothService( 5457): java.io.IOException: Service discovery failed
```

Figure 4.13: Android Bluetooth error

Figure 4.12 shows how users responded to the four questions presented to them. As we can see, all users find the protocol easy to use, perceive it as being secure and show intention of using it when deployed in real systems. Even though all the users completed the test procedure, part of them had problems during the execution of the protocol. These problems were failed Bluetooth connection attempts that happened after the NFC exchange, when the application on the mobile phone tries to connect to the Bluetooth service on the vending machine. When this connection failed, users had to retry the operation. There are several possible causes for these failures, ranging from the well-known Wi-Fi interference with Bluetooth[3] to incompatibilities between the Bluetooth adapters, to problems in the Android Bluetooth stack itself (see Figure 4.13). The study of these Bluetooth problems is beyond the scope of this work but,

[3]They both share the same spectrum.

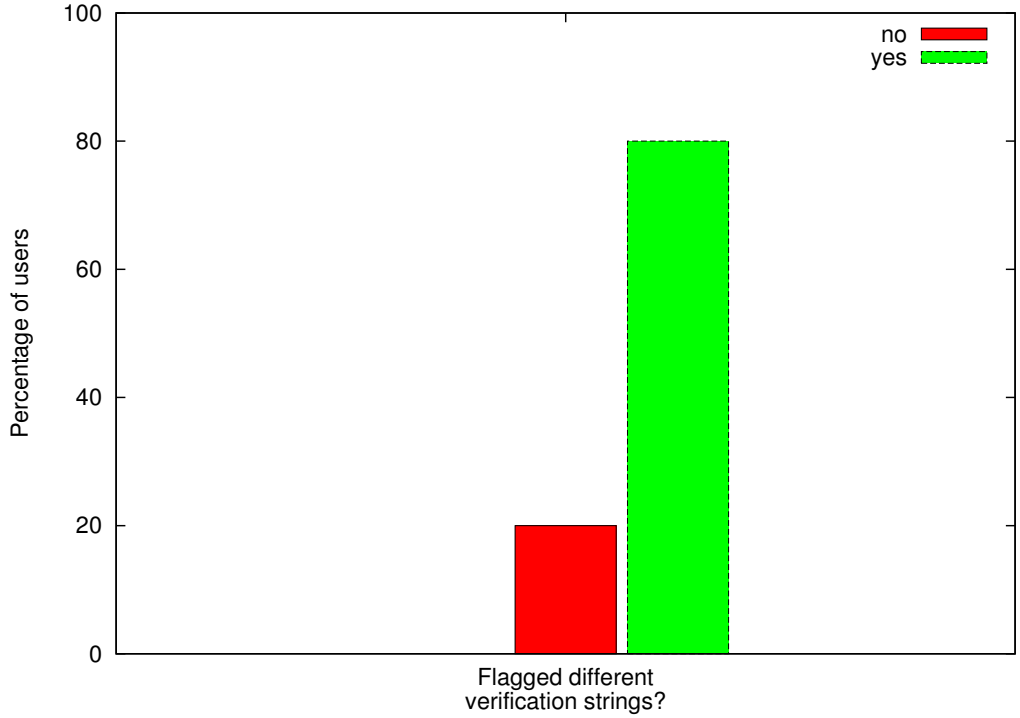


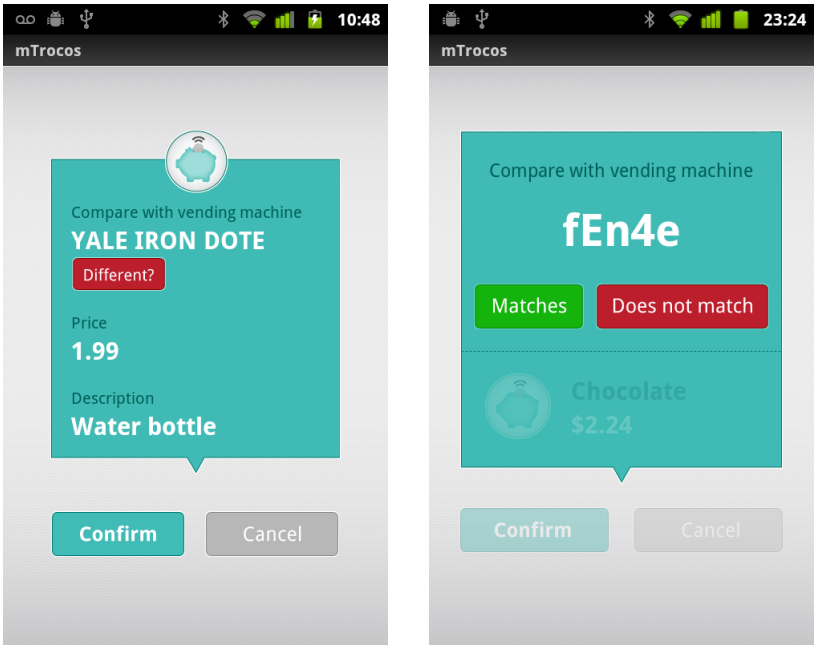
Figure 4.14: Verification string comparison results

nevertheless, it is something to take into account when evaluating this protocol for a production scenario.

Figure 4.14 represents the percentage of users that correctly flagged the different verification strings. These results are worrying because 20% of the tested users went ahead with the purchase operation even though the verification strings were different, which could be an indication that an attack was taking place. This high percentage undermines the security of *mTrocos*, since it violates one of the assumptions. This means that an attacker can succeed not only by guessing the nonce generated by the vending machine, but by attacking a user that ends up not comparing the verification strings. While the first has a very low probability, as seen in Section 4.1, the latter has a 20% probability, which is too high even if the other 80% would flag it and the attack was signaled. There can be two causes for people not doing the comparison: either the comparison operation is complicated or users do not even realize that they have to do it. In order to distinguish these two possibilities two more tests were conducted.

To test the hypotheses that the comparison operation is complicated, the five letter verification string was substituted by three words from a 2048 word dictionary,[4] to see if easing the comparison operation improved the results. Figure 4.15a shows an example of this in use, where the three word verification string is used.

[4]Each word represents 11 bits, so 3 words are enough for the verification strings because they had 30 bits.



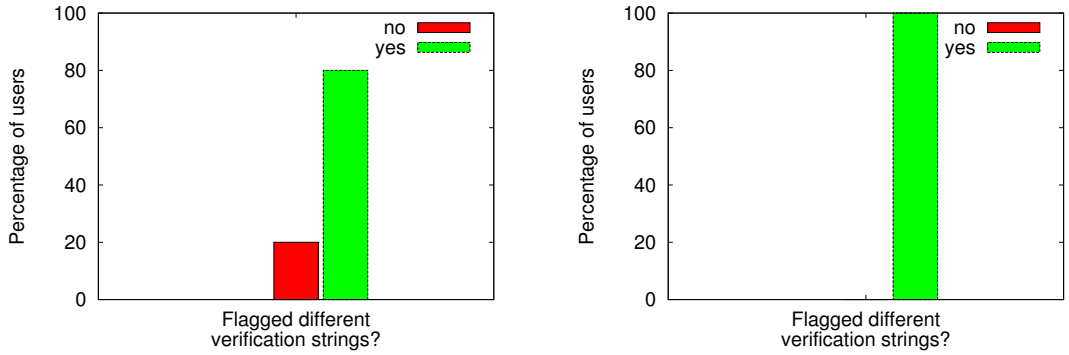
(a) Using dictionary words for the verification string

(b) Two-step process to emphasize comparison

Figure 4.15: The two different hypotheses to improve the test results

To test the hypothesis that users do not even realize that they have to do the comparison, the UI of the mobile application was altered in order to give more emphasis to the verification string and the comparison operation. This would, hopefully, catch the attention of users and compel them to perform the comparison. Figure 4.15b shows the design of the UI, where the process was transformed into two steps in a single screen: first the user is required to compare the verifications strings and press the corresponding button, then the bottom buttons become enabled for him to confirm the price and description. Although this has an impact on usability, it may be a needed compromise to achieve the desired level of security.

Two new groups of users were tested, this time the only focus of the test was to check if they flagged different verification strings. The first test group, consisting of ten subjects who were native portuguese speakers, tried the dictionary approach. Figure 4.15a shows the results of this first test. Afterwards, a second test group, consisting of nine subjects who were native portuguese speakers, tried the two-step approach. The results of the second test are shown in Figure 4.15b. As we can see, the dictionary approach did not bring any advantage, but the improved UI and the two-step process did have a noticeable positive impact: all the differences were correctly flagged. It is of note that, since the users who comprised the test group were native Portuguese speakers with good English skills, the results may have been different regarding the dictionary approach had they been native English speakers. However, other studies [121] have already determined that the UI plays a very important role in the error rate of these pairing methods, and again we see that here. With the new



(a) Using dictionary words

(b) Using the two-step process

Figure 4.16: Verification string comparison results with implemented changes

UI in place, the user tests part of the evaluation was completed with positive results.

4.3.4 Attained characteristics

Recalling the characteristics presented in Section 3.1, they will now be listed again and it will be explained how the *mTrocos* protocol achieves them:

Good usability This was achieved by minimizing the number of steps required of the user during a payment operation and designing an appropriate UI. The results of the user tests have shown that this was achieved.

Enable micropayments Micropayments depend on low per-transaction fees. *mTrocos* does not require merchants to be online, it is the customer that is online and contacts the operator, so merchants do not incur in any cost per payment operation. It is up to the operator to charge customers or not for the invocation of its service, but it is preferable that this is kept free of charge. Instead, it is encouraged that operators charge fixed periodic fees. For the operators, *mTrocos* is a differentiating service to attract more customers, still, there is money to be made when it is used by pre-paid customers who have to top up their accounts before they can use the money to conduct payments. The *float*, the interval of time between the money arrives at the operator and it is redeemed by a merchant, can be used by the operator to perform overnight investments.

Unbanked The traditional banking systems and protocols are not used by the *mTrocos* protocol.

Secure The DH-SC protocol allows the establishment of a secure channel between the client and the vending machine, SSL ensures the security of the communications between the operator and the client, and RSA signatures ensure that a note is valid.

Anonymous Blind signatures ensure that a note cannot be traced back to the customer that requested it.

No special hardware requirements An Android phone already available for sale is used, more phones with NFC support have also been released meanwhile, and a normal Android application was developed that can be installed on these phones.

Chapter 5

Conclusion

The NFC technology has evolved in recent years, from a state where multiple incompatible and proprietary solutions existed, to today where most of them are standardized and compatible layers such as NDEF have been developed. The NFC Forum has been busy designing standards to make NFC applicable to a myriad of new situations, so it seems unlikely that we will once again return to the previous state of multiple incompatible and proprietary[1] solutions. The mobile phone industry has noticed this trend and new phones have appeared that integrate the NFC technology.

The current NFC API found in the Android mobile operating system was analyzed and a few limitations were discovered. Developers should be aware of them, as they make it impossible to implement challenge-response type protocols using NFC. Since the NFC standard does not enforce any security properties, it is vulnerable to the typical security problems of a wireless communication technology such as eavesdropping or man-in-the-middle-attacks. In this work the necessary countermeasures have been enumerated to prevent these attacks from succeeding.

A market survey of the mobile payments area was also done and the current players were presented, detailing them and comparing them with each other though a comparison table. Most of them are in the category of digital credit accounts, which seek to extend the usage of traditional credit cards to mobile payments. This analysis was important in order to help define the characteristics a mobile payments protocol should have in order to succeed, namely: good usability, enable micropayments, unbanked, secure, anonymous and no special hardware requirements.

A new mobile payments protocol is presented, *mTrocos*, that takes advantage of the NFC technology and possesses a number of desirable characteristics not found in the other analyzed protocols. It is suited to the vending machine scenario and several alternative designs are also discussed, along with the reasons for not choosing them. A prototype of the whole protocol was developed, which proved the applicability of *mTrocos* using the Android mobile operating system. This prototype also allowed the execution of user tests and a complete evaluation of *mTrocos*, leading to the conclusion that it is a viable solution and that it can be implemented today. The use of NFC has brought added value to the protocol in that it allows for quick and

[1] And possibly insecure as in the case of Mifare.

easy pairing between the mobile phone and vending machine. If the price of vending machine displays capable of presenting QR Codes comes down to acceptable levels in the future, then the NFC + QR Codes approach of *mTrocos* also becomes a viable solution, with the added advantage that it does not rely on users for enforcing its security. It also does not use Bluetooth, which was found to be a source of implementation problems. It remains to be seen if scanning QR Codes would pose a usability problem and their reliability. A recent study [16] shows that on June 2011 alone, 14 million american customers have scanned QR Codes, which seems to indicate that users are well capable of performing the operation.

As NFC is starting to become standard on new phones and critical mass of users is reached, mobile operators have the opportunity to enter the mobile payments market by deploying protocols that do not involve banks, such as *mTrocos*. However, they are not alone, since the typical players are also maneuvering to tap this market and soon a lot of mobile payment options will be available to consumers. As for merchants, NFC will be just another technology that they will embrace, since it is cheap to support. Unbanked protocols like *mTrocos* are appealing to them because they support micropayments due to the non-existent transaction fee.

Appendix A

Users tests procedure

This appendix contains the procedure given to users during the user tests conducted in the context of the evaluation of the *mTrocos* protocol. No additional instructions were given to them.

mTrocos

1 Overview

mTrocos is a payment protocol that runs on mobile phones and can be used to buy products on vending machines. As part of this study we request that you help us evaluate this protocol. To this end, you will be given a mobile phone and will be asked to pretend that you are purchasing items from a real vending machine. For the purpose of this study, the vending machine will be simulated on the laptop screen and you can interact with it using the trackpad and the white sensor at its side.

2 Test procedure

You will be asked to execute the protocol 5 times to attempt to buy 5 products. In the end a simple questionnaire will be presented with 4 questions. There are only 2 different products available on the simulated vending machine, so you will buy each one of them alternatively starting with the water bottle. Each time you initiate a new transaction assume that you just arrived at a new vending machine.

2.1 Steps

1. Start application *mTrocos*, which is on the home screen of the phone.
2. Insert your PIN, which is "1111".
3. Choose a product on vending machine.
4. Place the back of the mobile phone against the vending machine, the white sensor, until the vending machine displays a message to continue the protocol on the phone.
5. Wait for the payment details to appear on the phone.
6. The display will show 3 pieces of data:
   (a) A 5 letter word.
   (b) The price.
   (c) The product description.
7. You should check the price, product description and make sure that the 5 letter word matches the one displayed on the vending machine. If they were not to match this would be an indication that you were interacting with a rogue sensor, in which case you would want to abort the transaction by clicking the "Different" button.
8. If all is well, press the "Confirm" button and wait for the protocol to finish.
9. When the success message is presented, click on "stop" to signal that you do are finished with this vending machine.

3 Questions

1. Is it easy to use?
2. Does the protocol feel secure?
3. Would you use this protocol?
4. Were the purchases problem-free?

Bibliography

[1] Alibaba.com. Manufacturers, suppliers, exporters and importers from the world's largest online B2B marketplace-alibaba.com, June 2011. URL: `http://www.alibaba.com/`.

[2] American Express. What is serve, June 2011. URL: `http://www.serve.com/about.html`.

[3] R. Balan, N. Ramasubbu, K. Prakobphol, N. Christin, and J. Hong. mferio: the design and evaluation of a peer-to-peer mobile payment system. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 291–304. ACM, 2009.

[4] E. Barker, W. Barker, and W. Burr. Recommendation for key management – part 1 : General. *NIST Special Publication 800-57*, 1(1/3):1–143, May 2011. URL: `http://csrc.nist.gov/groups/ST/toolkit/documents/SP800-57Part1-Revision3_May2011.pdf`.

[5] Bloomberg. Sprint plans tap-and-go payments in 2011, ahead of rivals. Business Week, May 2011. URL: `http://www.businessweek.com/news/2011-04-04/sprint-plans-tap-and-go-payments-in-2011-ahead-of-rivals.html`.

[6] Bluetooth SIG. *Bluetooth Specification Version 2.1+ EDR*. Bluetooth, SIG, 2007.

[7] BlueZ Project. BlueZ - official Linux Bluetooth protocol stack, June 2011. URL: `http://www.bluez.org/`.

[8] S. Brands and D. Chaum. Distance-bounding protocols. In *Advances in Cryptology EUROCRYPT 93*, pages 344–359. Springer, 1994.

[9] Bump Technologies. The bump app for iPhone and Android, May 2011. URL: `http://bu.mp/faq`.

[10] M. Cagalj, S. Capkun, and J. Hubaux. Key agreement in peer-to-peer wireless networks. *Proceedings of the IEEE*, 94(2):467–478, 2006.

[11] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto*, volume 82, pages 199–203, 1983.

[12] D. Chaum and S. Brands. Minting electronic cash. *IEEE Spectrum*, 34(2):30–34, 1997.

[13] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology CRYPTO 88*, pages 319–327. Springer, 1990.

[14] CNN. Google wallet 'basically the same' as credit card, exec says, May 2011. URL: `http://edition.cnn.com/2011/TECH/mobile/05/26/google.wallet.mastercard/index.html`.

[15] CNNMoney. Your mobile phone is becoming your wallet, May 2011. URL: `http://money.cnn.com/2011/01/19/technology/mobile_payments/index.htm?iid=EAL`.

[16] comScore. 14 million americans scanned QR codes on their mobile phones in june 2011, August 2011. URL: `http://www.comscore.com/Press_Events/Press_Releases/2011/8/14_Million_Americans_Scanned_QR_or_Bar_Codes_on_their_Mobile_Phones_in_June_2011`.

[17] Corduro. Corduro FAQ, June 2011. URL: `http://www.corduro.com/faq/`.

[18] Correio da manhã. Porta moedas multibanco: Uma aposta falhada, January 2003. URL: `http://www.cmjornal.xl.pt/detalhe/noticias/outros/domingo/porta-moedas-multibanco-uma-aposta-falhada`.

[19] D. Crockford. Introducing JSON, May 2011. URL: `http://www.json.org/`.

[20] A. Cruz. NPP support for nfcpy, June 2011. URL: `https://code.launchpad.net/~andrefcruz/nfcpy/npp-1.0`.

[21] I. Damgård. Commitment schemes and zero-knowledge protocols. *Lectures on Data Security*, pages 63–86, 1999.

[22] G. de Koning Gans, J. Hoepman, and F. Garcia. A practical attack on the MIFARE classic. *Smart Card Research and Advanced Applications*, pages 267–282, 2008.

[23] Diário de Notícias. SIBS e bancos acabam com porta-moedas multibanco, January 2006. URL: `http://www.dn.pt/inicio/interior.aspx?content_id=635135&page=-1`.

[24] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176. URL: `http://www.ietf.org/rfc/rfc5246.txt`.

[25] Digi-Key Corporation. RFID ICs | Digi-Key, June 2011. URL: `http://search.digikey.com/scripts/DkSearch/dksus.dll?keywords=pn544`.

[26] R. Dunn. wxpython, a blending of the wxwidgets c++ class library with the python programming language, July 2011. URL: `http://www.wxpython.org/`.

[27] M. Dworkin. Recommendation for block cipher modes of operation, methods and techniques. *NIST Special Publication 800-38A*, pages 1–66, 2001.

[28] ECMA. *ECMA-352: Near Field Communication Interface and Protocol-2 (NFCIP-2)*. ECMA (European Association for Standardizing Information and Communication Systems), Dec. 2003. URL: http://www.ecma-international.org/publications/standards/Ecma-352.htm.

[29] ECMA. *ECMA-340: Near Field Communication — Interface and Protocol (NFCIP-1)*. ECMA (European Association for Standardizing Information and Communication Systems), Dec. 2004. URL: http://www.ecma-international.org/publications/standards/Ecma-340.htm.

[30] EMVCo. About EMV, June 2011. URL: http://www.emvco.com/about_emv.aspx.

[31] Engadget. Live from the google wallet press event!, May 2011. URL: http://www.engadget.com/2011/05/26/live-from-the-google-wallet-press-event/.

[32] R. Fielding and R. Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150, 2002.

[33] S. Fonteneau. Avoid missing NFCIP inbound frame, May 2011. URL: http://android.git.kernel.org/?p=platform/external/libnfc-nxp.git;a=commit;h=7b40e6d4b45110797cdea7d451d313af891bf6e3.

[34] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. Practical NFC peer-to-peer relay attack using mobile phones. *Radio Frequency Identification: Security and Privacy Issues*, pages 35–49, 2010.

[35] B. Gajda. Real innovation in mobile payments. Visa Blog, May 2011. URL: http://blog.visa.com/2010/11/18/real-innovation-in-mobile-payments/.

[36] GeldKarte. Details behind loading and payment transactions, May 2011. URL: http://www.geldkarte.de/_www/en/pub/geldkarte/press/technical_procedure.php.

[37] GeldKarte. Loading and unloading your geldkarte, May 2011. URL: http://www.geldkarte.de/_www/en/pub/geldkarte/geldkarte_users/loading_and_unloading.php.

[38] GeldKarte. Welcome to geldkarte.de, May 2011. URL: http://www.geldkarte.de/_www/en/pub/geldkarte/geldkarte_users.php.

[39] Gizmodo. Square iPhone-based mobile payment system kills cash registers and wallets, May 2011. URL: http://gizmodo.com/5804589/square-introduces-iphone+based-mobile-payment-system.

[40] Google. Android 2.3.3 platform | android developers, May 2011. URL: http://developer.android.com/sdk/android-2.3.3.html.

[41] Google. Android 2.3.4 platform | android developers, July 2011. URL: http://developer.android.com/sdk/android-2.3.4.html.

[42] Google. *Android NDEF Push Protocol Specification (NPP) - Version 1.* Google, http://source.android.com/compatibility/ndef-push-protocol.pdf, February 2011.

[43] Google. Android.com, July 2011. URL: http://www.android.com/.

[44] Google. Google wallet - make your phone your wallet, June 2011. URL: http://www.google.com/wallet/.

[45] Google. Tech specs – Nexus S, May 2011. URL: http://www.google.com/nexus/tech-specs.html.

[46] N. Haller. The S/KEY One-Time Password System. RFC 1760 (Informational), Feb. 1995. URL: http://www.ietf.org/rfc/rfc1760.txt.

[47] E. Haselsteiner and K. Breitfuß. Security in near field communication (NFC). In *Proceedings of Workshop on RFID and Lightweight Crypto (RFIDSec06)*, volume 11, 2006.

[48] G. Horn and B. Preneel. Authentication and payment in future mobile systems. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors, *Computer Security — ESORICS 98*, volume 1485 of *Lecture Notes in Computer Science*, pages 277–293. Springer Berlin / Heidelberg, 1998.

[49] A. Huang. Pybluez is an effort to create python wrappers around system Bluetooth resources to allow python developers to easily and quickly create Bluetooth applications, June 2011. URL: http://code.google.com/p/pybluez/.

[50] E. Imhontu and Y. Kumah. A survey on near field communication in mobile phones & PDAs. Technical report, School of Information Science, Computer and Electrical Engineering, Halmstad University, 2010.

[51] Intuit. GoPayment, May 2011. URL: http://gopayment.com/.

[52] IOGEAR. Iogear - gbu421 - Bluetooth 2.1 USB micro adapter, May 2011. URL: http://www.iogear.com/product/GBU421/.

[53] Isis. Welcome to Isis, June 2011. URL: http://www.paywithisis.com/.

[54] ISO/IEC. *ISO/IEC 18004:2006 Information technology – Automatic identification and data capture techniques – QR Code 2005 bar code symbology specification.* International Organization for Standardization, 2006. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=43655.

[55] ISO/IEC. *ISO/IEC 14443-1:2008 Identification cards — Contactless integrated circuit(s) cards — Proximity cards - Part 1: Physical characteristics*. International Organization for Standardization, 2008.

[56] ISO/IEC. *ISO/IEC 14443-1:2008 Identification cards — Contactless integrated circuit(s) cards — Proximity cards - Part 4: Transmission protocol*. International Organization for Standardization, 2008.

[57] ISO/IEC. *ISO/IEC 14443-1:2008 Identification cards — Contactless integrated circuit(s) cards — Proximity cards - Part 2: Radio frequency power and signal interface*. International Organization for Standardization, 2010.

[58] ISO/IEC. *ISO/IEC 14443-1:2008 Identification cards — Contactless integrated circuit(s) cards — Proximity cards - Part 3: Initialization and anticollision*. International Organization for Standardization, 2011.

[59] S. Josefsson. The Base16, Base32, and Base64 Data Encodings. RFC 4648 (Proposed Standard), Oct. 2006. URL: `http://www.ietf.org/rfc/rfc4648.txt`.

[60] K. Kadambi, J. Li, and A. Karp. Near-field communication-based secure mobile payment service. In *Proceedings of the 11th international Conference on Electronic Commerce*, pages 142–151. ACM, 2009.

[61] C. Kim, G. Avoine, F. Koeune, F. Standaert, and O. Pereira. The swiss-knife RFID distance bounding protocol. In P. J. Lee and J. H. Cheon, editors, *Information Security and Cryptology — ICISC 2008*, pages 98–115. Springer-Verlag, Berlin, Heidelberg, 2009.

[62] C. Kim, M. Mirusmonov, and I. Lee. An empirical examination of factors influencing the intention to use mobile payment. *Computers in Human Behavior*, 26(3):310–322, 2010.

[63] N. Kreyer, K. Pousttchi, and K. Turowski. Characteristics of mobile payment procedures. Technical report, University Library of Munich, Germany, 2002.

[64] S. Kumar and S. Rabara. An architectural design for secure mobile remote macro-payments. *Journal of Next Generation Information Technology*, 1(2), 2010.

[65] S. Kungpisdan, B. Srinivasan, and P. Le. A secure account-based mobile payment protocol. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2 - Volume 2*, volume 1 of *ITCC '04*, pages 35–39, Washington, DC, USA, 2004. IEEE Computer Society.

[66] B. Lam. P2P micro-interactions with NFC-enabled mobile phones. In *ACM Symposium on User Interface Software and Technology*, 2011. To appear.

[67] Launchpad. Scl3711 reader disappears from usb bus after running llcp example, May 2011. URL: https://bugs.launchpad.net/nfcpy/+bug/788658.

[68] P. Leach, M. Mealling, and R. Salz. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122 (Proposed Standard), July 2005. URL: http://www.ietf.org/rfc/rfc4122.txt.

[69] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger. NFC devices: Security and privacy. In *The Third International Conference on Availability, Reliability and Security*, pages 642–647. IEEE, 2008.

[70] M. Massoth and T. Bingel. Performance of different mobile payment service concepts compared with a NFC-based solution. In *2009 Fourth International Conference on Internet and Web Applications and Services*, pages 205–210. IEEE, 2009.

[71] J. McCarthy. Isis now includes Visa. Visa Blog, July 2011. URL: http://blog.visa.com/2011/07/19/isis-now-includes-visa/.

[72] J. McCarthy. Visa advances next-gen payments solutions. Visa Blog, May 2011. URL: http://blog.visa.com/2011/05/11/visa-advances-next-gen-payments-solutions/.

[73] J. McCune, A. Perrig, and M. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. *Security and Privacy, IEEE Symposium on*, pages 110–124, 2005.

[74] Memcached. memcached - a distributed memory object caching system, May 2011. URL: http://memcached.org/.

[75] Mobile-For. Ping ping, June 2011. URL: http://www.pingping.be/wp/about/.

[76] C. Mulliner. Vulnerability analysis and attacks on NFC-enabled mobile phones. In *2009 International Conference on Availability, Reliability and Security*, pages 695–700. IEEE, 2009.

[77] MySQL. MySQL :: The world's most popular open source database, May 2011. URL: http://www.mysql.com/.

[78] Naratte. Zoosh, June 2011. URL: http://www.naratte.com/.

[79] National Institute of Standards and Technology. *FIPS 197: Advanced Encryption Standard (AES)*. National Institute for Standards and Technology, 2001. URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[80] National Institute of Standards and Technology. *FIPS 180-3: Secure Hash Standard (SHS)*. National Institute for Standards and Technology, 2008. URL: http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

[81] NFC Forum. *NFC Data Exchange Format (NDEF), Technical Specification - Version 1.0*. NFC Forum, 2006.

[82] NFC Forum. *Type 1 Tag Operation Specification, Technical Specification - Version 1.0*. NFC Forum, 2007.

[83] NFC Forum. *Type 2 Tag Operation, Technical Specification - Version 1.0*. NFC Forum, 2007.

[84] NFC Forum. *Type 3 Tag Operation, Technical Specification - Version 1.0*. NFC Forum, 2007.

[85] NFC Forum. *NFC Logical Link Control Protocol (LLCP), Technical Specification - Version 1.0*. NFC Forum, 2009.

[86] NFC Forum. *Connection Handover, Technical Specification - Version 1.2*. NFC Forum, 2010.

[87] NFC Forum. *Type 4 Tag Operation, Technical Specification - Version 2.0*. NFC Forum, 2010.

[88] NFC Forum. NFC forum : Home, May 2011. URL: `http://www.nfc-forum.org/`.

[89] NFC World. Isis sets out its new NFC strategy, May 2011. URL: `http://www.nearfieldcommunicationsworld.com/2011/05/24/37652/isis-sets-out-new-nfc-strategy/`.

[90] NXP. mifare.net :: Home, May 2011. URL: `http://www.mifare.net/`.

[91] NXP. Pn533 user manual rev 03, May 2011. URL: `http://www.nxp.com/documents/user_manual/157830_PN533_um080103.pdf`.

[92] J. Ondrus and Y. Pigneur. An assessment of NFC for future mobile payment systems. In *Proceedings of the International Conference on the Management of Mobile Business*, pages 43–, Washington, DC, USA, 2007. IEEE Computer Society.

[93] J. Ondrus and Y. Pigneur. Near field communication: an assessment for future payment systems. *Information Systems and E-Business Management*, 7(3):347–361, 2009.

[94] Oracle. java.com: Java + you, May 2011. URL: `http://www.java.com/`.

[95] OSAF - Open Source Applications Foundation. Me too crypto, June 2011. URL: `http://chandlerproject.org/bin/view/Projects/MeTooCrypto`.

[96] H. J. Pelly N. How to NFC. Google I/O 2011, May 2011. URL: `http://www.google.com/events/io/2011/static/presofiles/how_to_nfc.pdf`.

[97] A. Perrig and D. Song. Hash visualization: A new technique to improve real-world security. In *International Workshop on Cryptographic Techniques and E-Commerce*, pages 131–138. Citeseer, 1999.

[98] K. Pousttchi. Conditions for acceptance and usage of mobile payment procedures. In *The Second International Conference on Mobile Business*, pages 201–210. Citeseer, 2003.

[99] Python Software Foundation. Python programming language – official website, May 2011. URL: `http://www.python.org/`.

[100] K. Rasmussen and S. Capkun. Realization of RF distance bounding. In *USENIX Security 2010: Proceedings of the 19th USENIX Security Symposium*, 2010.

[101] RSA Laboratories. *PKCS #1 v2.1: RSA Cryptography Standard*, Apr. 2002.

[102] N. Sadeh. *M-commerce: technologies, services, and business models*. John Wiley & Sons, April 2002.

[103] P. Schierz, O. Schilke, and B. Wirtz. Understanding consumer acceptance of mobile payment services: An empirical analysis. *Electronic Commerce Research and Applications*, 9(3):209–216, 2010.

[104] S. Schulte. Security advisory twsl2011-008: Focus stealing vulnerability in Android. Technical report, Trustwave's SpiderLabs, August 2011. URL: `https://www.trustwave.com/spiderlabs/advisories/TWSL2011-008.txt`.

[105] SCM. Scl3711 contactless reader and nfc enabling accessory, May 2011. URL: `http://www.scmmicro.com/fileadmin/products/datasheets/Dat_SCL3711_e.pdf`.

[106] S. Shen. Competitive landscape: Mobile payment vendors, worldwide, 2010. *Gartner Group*, 2010. URL: `http://www.gartner.com/DisplayDocument?doc_cd=174110`.

[107] SIBS. Os serviços multibanco (porta moedas multibanco), October 2000. URL: `http://web.archive.org/web/20001015022842/http://www.sibs.pt/pt/porta_moedas.html`.

[108] SIBS. Perguntas frequentes, October 2006. URL: `http://web.archive.org/web/20060101034006/http://www.sibs.pt/perg_freq.php`.

[109] J. Smart. wxwidgets - cross-platform GUI library, July 2011. URL: `http://www.wxwidgets.org/`.

[110] Sony. Contactless IC card reader/writer felica port(stick type) - RC-S360/S RC-S360/SH, June 2011. URL: `http://www.sony.net/Products/felica/business/data/RC-S360_E.pdf`.

[111] SparkBase. Sparkbase first to integrate zoosh NFC into retail transactions, June 2011. URL: http://sparkbase.com/2011/06/sparkbase-first-to-integrate-zoosh-nfc-into-retail-transactions/.

[112] Square. Square help, May 2011. URL: https://help.squareup.com/.

[113] The Android Open Source Project. Source code of com.android.nfc.NativeNfcSecureElement.java, June 2011. URL: http://android.git.kernel.org/?p=platform/packages/apps/Nfc.git;a=blob_plain;f=src/com/android/nfc/NativeNfcSecureElement.java.

[114] The Apache Software Foundation. The apache cassandra project, May 2011. URL: http://cassandra.apache.org/.

[115] The Apache Software Foundation. Apache MINA - welcome to apache MINA project, July 2011. URL: http://mina.apache.org/.

[116] The Legion of the Bouncy Castle. Bouncy castle, June 2011. URL: http://www.bouncycastle.org/.

[117] The OpenSSL Project. OpenSSL: The open source toolkit for SSL/TLS, June 2011. URL: http://www.openssl.org/.

[118] The Wall Street Journal. Mobile wallets poised for european takeoff, May 2011. URL: http://online.wsj.com/article/SB10001424052748703652104576121952411914010.html.

[119] S. Tiedemann. Python module for near field communication, May 2011. URL: https://launchpad.net/nfcpy.

[120] R. Tyley. Spongy castle - a repackage of bouncy castle for Android, June 2011. URL: https://github.com/rtyley/spongycastle.

[121] E. Uzun, K. Karvonen, and N. Asokan. Usability analysis of secure pairing methods. *Financial Cryptography and Data Security*, pages 307–324, 2007.

[122] G. Van Damme, K. Wouters, H. Karahan, and B. Preneel. Offline NFC payments with electronic vouchers. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, pages 25–30. ACM, 2009.

[123] Vending World. Vending machines, refurbished and used, vending machine business, June 2011. URL: http://www.vendingworld.com/.

[124] Venmo. Venmo help, June 2011. URL: https://help.venmo.com/.

[125] Venmo. Venmo trends in social payments, 2010, May 2011. URL: http://blog.venmo.com/post/3291380355/venmo-trends-in-social-payments-2010.

[126] R. Verdult and F. Kooman. Practical attacks on NFC enabled cell phones. In *Near Field Communication (NFC), 2011 3rd International Workshop on*, pages 77–82. IEEE, 2011.

[127] VeriFone. PAYware mobile, May 2011. URL: `http://www.paywaremobile.com/en/features`.

[128] M. Weiß. Performing relay attacks on ISO 14443 contactless smart cards using NFC mobile equipment. Master's thesis, Technische Universität München, May 2010. URL: `http://www.sec.in.tum.de/assets/studentwork/finished/Weiss2010.pdf`.

[129] E. Woyke. How AT&T, T-Mobile and Verizon's mobile payment app will work. Forbes, May 2011. URL: `http://blogs.forbes.com/elizabethwoyke/2011/05/17/how-att-t-mobile-and-verizons-mobile-payment-app-will-work/`.