

University of Moratuwa
Department of Computer Science and Engineering



CS 4202 – Research and Development Project

Final Year Project Report

Contactless Enabled Payment Solution

Project Group 08

P.M.I. Chamara (110067U)

S.A.C. Dulanga (110143B)

P.H.T. Imesh (110228P)

J.M.R. Ruparathna (110486D)

Internal Supervisor

Dr. H.M.N. Dilum Bandara

External Supervisor

Mr. Rohan Kumara, CAKE LABS.

Coordinated By

Dr. Malaka Walpola

THIS REPORT IS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF BACHELOR OF SCIENCE OF ENGINEERING
AT UNIVERSITY OF MORATUWA, SRI LANKA.

15th of May, 2015

Declaration

We, the project group 08 (P.M.I. Chamara, S.A.C. Dulanga, P.H.T. Imesh, J.M.R. Ruparathna under the supervision of Dr. H.M.N. Dilum Bandara and Mr. Rohana Kumara) hereby declare that except where specified reference is made to the work of others, the project “Contactless Enabled Payment Solution” is our own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgement.

Signatures of the candidates:

1. P.M.I. Chamara (110067U)
2. S.A.C. Dulanga (110143B)
3. P.H.T. Imesh (110228P)
4. J.M.R. Ruparathna (110468D)

Supervisor:

.....

(Signature and Date)

Dr. H.M.N. Dilum Bandara

Project Coordinator:

.....

(Signature and Date)

Dr. Malaka Walpola

Abstract

Project Title: Contactless Enabled Payment Solution
P.M.I. Chamara (110067U)
S.A.C. Dulanga (110143B)
P.H.T. Imesh (110228P)
J.M.R. Ruparathna (110468D)
Internal Supervisor: Dr. H. M. N. Dilum Bandara
External Supervisors: Mr. Rohana Kumara

Contactless payments are the latest trend in retail payment applications. They are payment transactions that require no physical contact between the consumer payment device and the point-of-sale (POS) terminal. Contactless transactions became very popular due to the speed and convenience of payments to the consumers and also due to the increase of sales volumes to merchants.

Cake Labs is one of the leading POS solution providers for restaurant industry in the USA. *Cake POS* is their main product, which includes a hardware terminal and software for managing orders, issuing bills, handling payments, and obtaining sales statistics.

Cake POS works with Cake Labs Wallet, which is a magnetic stripe card designed by Cake Labs to perform payments and transactions at their POS terminals, which acts as a wallet. In the USA many retail payment applications now support contactless payments. Following its competition, Cake Labs wants to extend their POS and Wallet cards to support contactless payments. This will enable their partners to enjoy the benefits on contactless payments. However, the integration of contactless payments should be accomplished in such a way that it negates the impacts due to security issues such as being vulnerable to spoofing attacks and eavesdropping. Therefore, a significant amount of research is required to select an appropriate contactless card, compatible reader, and design and develop a secure communication and data retention strategy.

We address this problem by researching and developing a complete contactless enabled solution to facilitate contactless payments in Cake POS. The proposed solution to be implemented while addressing known and anticipated security issues present in existing contactless payment solutions. Specific contributions to include integrating a suitable contactless reader with necessary security options with the POS system, identifying a suitable contactless-enabled card with necessary security options, developing mobile application(s) for smartphones which can be used as virtual wallet cards and developing a software development kit / library (SDK). The proposed SDK is to support multiple contactless cards and readers.

Acknowledgement

First and foremost we would like to express our sincere gratitude to our project supervisor, Dr. H.M.N. Dilum Bandara for the valuable guidance and dedicated involvement at every step throughout the process.

We would also like to thank our external supervisor Mr. Rohana Kumara (VP. Engineering, Cake Labs) for the valuable advice and the direction given to us regarding the project.

In addition, we would like to thank Mr. Nuwan Janaka (Software Engineer, Cake Labs), for the support, encouragement and insightful comments.

We would like to express our warm gratitude to Dr. Malaka Walpola for coordinating the final year projects.

Last but not least, we would like to express our greatest gratitude to the Department of Computer Science and Engineering, University of Moratuwa for providing the support for us to successfully finish the project.

Table of Contents

List of Figures	viii
List of Tables	x
List of Abbreviations	xi
1 Introduction.....	1
1.1 Background.....	1
1.2 Problem Statement	1
1.3 Proposed Solution	2
2 Literature Survey	4
2.1 Near Field Communication.....	4
2.1.1 NFC Standard and Protocols.....	4
2.1.2 Classifications of NFC Devices	5
2.1.2.1 Active vs. Passive Devices.....	5
2.1.2.2 Initiator vs. Target Devices	5
2.1.3 Operating Modes of NFC devices.....	5
2.1.3.1 Reader/Writer Mode	5
2.1.3.2 Peer-to-Peer Mode	6
2.1.3.3 Card Emulation Mode	7
2.1.4 NFC Protocols Stack, Interface, and Specifications	8
2.1.5 NFCIP-1	8
2.1.6 NFCIP-2	8
2.1.7 Protocol Layer.....	8
2.1.7.1 NDEF and RTD	11
2.1.8 Operation of NFC.....	11
2.1.9 NFC Software Stack.....	13
2.1.9.1 LibNFC	13
2.1.9.2 Open NFC	13
2.1.9.2.1 Open NFC Software Stack	14
2.1.9.2.2 Open NFC Interfaces	14
2.1.9.2.3 Open NFC Security Stack	15
2.2 Card Types.....	16

2.3	NFC Coupling Devices (Readers and Writers).....	17
2.3.1	Comparison of existing NFC Coupling Devices.....	18
2.4	Security	19
2.4.1	Threats.....	19
2.4.1.1	Eavesdropping.....	19
2.4.1.2	Data Corruption.....	19
2.4.1.3	Data Modification	19
2.4.1.4	Data Insertion.....	20
2.4.1.5	Man-in-the-Middle-Attack.....	20
2.4.1.6	Data Stealing	20
2.4.1.7	Relay Attack.....	21
2.4.1.8	Malware	21
2.4.1.9	Android Beam.....	21
2.4.1.10	Spoofing.....	21
2.4.2	Solutions	21
2.5	Android NFC Stack.....	22
2.6	Payment Card Industry Guidelines, Recommendations and Best Practices	23
2.6.1	PCI PA-DSS Requirements	23
2.6.2	Contactless and Mobile Payments Council - Implementation Best Practices.....	24
2.6.3	Micro Payments	24
2.7	Related Products	25
2.7.1	EMV Technology.....	25
2.7.1.1	Security Mechanisms used by EMV	25
2.7.1.2	Contactless Card Technology and EMV	26
2.7.1.3	NFC Mobile Payment and EMV	26
2.7.1.4	Proposed Solution and EMV.....	26
2.7.2	Google Wallet	26
2.7.3	Apple Pay.....	27
3	Design	28
3.1	Overall System Architecture of the solution.....	28
3.2	Mobile Application	29

3.3	ResmArc SDK	30
3.3.1	Communication Module	31
3.3.2	Card monitoring and processing	31
3.3.3	Communications / APDU	32
3.3.3.1	APDU Commands with Application.....	33
3.3.4	Authentication Module	33
3.3.5	Web Service Request Module.....	33
3.4	NFC Card and Reader selection.....	33
3.4.1	NFC Reader selection	33
3.4.2	NFC Card selection.....	35
3.4.2.1	MIFARE CLASSIC	35
3.5	NFC Reader Library and Drivers.....	36
4	Implementation	38
4.1	Languages, Tools and Technologies.....	38
4.2	Implementation of the Android mobile application	38
4.3	Implementation of the ResmArc SDK.....	40
5	Results.....	41
5.1	Testing of the modules	41
5.2	Results.....	41
6	Conclusion	43
6.1	Problems and challenges faced	43
6.2	Future work.....	43
	References.....	44

List of Figures

Figure 1.1 Targeted project deliverables, which include four main modules; NFC card, reader, mobile application and libraries for the framework	2
Figure 2.1 Evolution of NFC technology [5]	4
Figure 2.2 NFC objectives [7]	5
Figure 2.3 Reader/Writer mode [5]	5
Figure 2.4 Usage scenario of NFC shopping: reader/writer mode [5]	6
Figure 2.5 Peer-to-peer mode [5]	6
Figure 2.6 Generic usage model of peer-to-peer mode [5]	6
Figure 2.7 Card emulation mode [5]	7
Figure 2.8 Generic usage model of card emulation mode [5]	7
Figure 2.9 Relationship between LLCP and OSI reference model [5]	9
Figure 2.10 NFC protocol stack [14]	10
Figure 2.11 NDEF structure [15]	11
Figure 2.12 Protocol stack for each operating mode [16]	11
Figure 2.13 NFC communication model	12
Figure 2.14 NFC communication model internal architecture [17]	12
Figure 2.15 Modulation spectra showing load modulation [6]	13
Figure 2.16 Components of the Open NFC security stack [23]	16
Figure 2.17 Internal hardware of NFC tag [24]	16
Figure 2.18 NFC USB reader DL533N OEM [33]	18
Figure 2.19 OpenPCD RFID reader [34]	18
Figure 2.20 Man-in-the-middle setup	20
Figure 2.21 Relay attack scenario [38]	21
Figure 2.22 Android NFC stack with different libraries	22
Figure 3.1 Overall System Architecture	28
Figure 3.2 NFC card emulation without a secure element.	29
Figure 3.3 Pyscard Architecture [47]	31
Figure 3.4 APDU Command Format [49].	32
Figure 3.5 APDU Response Format [49]	32
Figure 3.6 APDU Commands with Wallet Card	32

Figure 3.7 Application Select Command	33
Figure 3.8 ACR1252U USB NFC Reader	34
Figure 3.9 MIFARE DESFire EV1 NFC Card	35
Figure 3.10 MIFARE Classic NFC Card	35
Figure 4.1 Wallet app main screen.....	39
Figure 4.2 Challenge Response Authentication Protocol	39
Figure 4.3 Otto Event bus	40

List of Tables

Table 2.1 Benefits of various operating modes of NFC	7
Table 2.2 Interaction with the service provider	8
Table 2.3 Modulation and coding schemes based on device type and data rate	12
Table 2.4 Open NFC interfaces summary	14
Table 2.5 Summary of NFC tag types.....	17
Table 2.6 Comparison of exiting NFC coupling devices	18
Table 2.7 PCI DSS requirements	23

List of Abbreviations

ACL	Access Control List
AES	Advanced Encryption Standard
AID	Access Identifier
APDU	Application Protocol Data Units
API	Application Programming Interface
ASK	Amplitude Shift Keying
CDA	Combined Data Authentication
CVM	Cardholder Verification Methods
DDA	Dynamic Data Authentication
DES	Data Encryption Standard
ECMA	European Computer Manufacturers Association
EMV	Euro-pay MasterCard Visa
ESE	Embedded Secure Element
ETSI TS	European Telecommunications Standards Institute Technical Specification
FSK	Frequency Shift Keying
FWT	Frame Waiting Time
HAL	Hardware Abstraction Layer
IC	Integrated Circuit
IEC	International Electro-technical Commission
ISO	International Organization for Standardization
JCOP	Java Card Open Platform
JIS	Japanese Industrial Standard
LED	Light Emitting Diode
LLCP	Logical Link Control Protocol
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
NFCIP	Near Field Communication Interface and Protocol
OEM	Original Equipment Manufacturer
PAN	Permanent Account Number
PCB	Printed Circuit Board

PCI DSS	Payment Card Industry Data Security Standard
PCI PA-DSS	Payment Card Industry Payment Applications-Data Security Standard
PDA	Personal Digital Assistant
PIN	Personal Identification Number
POS	Point of Sale
P2P	Peer to Peer
RF	Radio Frequency
RFID	Radio Frequency Identification
RTD	Record Type Definition
SDA	Static Data Authentication
SDK	Software Development Kit
SEAC	Secure Element Access Control
SNEP	Simple NDEF Exchange Protocol
SSL	Secure Sockets Layer
UICC	Universal Integrated Circuit Card
USA	United State America
USB	Universal Serial Bus
URI	Uniform Resource Identifier
HCE	Host-based Card Emulation
AID	Application Identifier

1 Introduction

1.1 Background

Contactless payments are the latest trend in retail payment applications. Contactless payments are payment transactions that require no physical contact between the consumer payment device and the point-of-sale (POS) terminal. In such a transaction, the consumer holds the contactless card or device in close proximity (typically less than 1cm) to the merchant's POS terminal and the payment account information is communicated wirelessly (via radio frequency (RF)). There is significant momentum in the marketplace to support the contactless payments to charge, credit, debit, prepay, and use as gift cards in both consumer and corporate card programs. Contactless payments provide significant benefits for all stakeholders [1] including the following:

- Speed and convenience of contactless payments for consumers
- Increase transaction volumes by capturing transactions typically made using cash for financial institutions
- Ability for financial institutions to differentiate themselves with innovative new form factors to enhance loyalty and retention

Contactless payments allow issuers to penetrate the cash payment market, enjoy increased customer transaction volume, and improve customer retention and loyalty. Retailers realize benefits due to shorter transaction times, increased revenue, improved operational efficiency, and lower operating costs. Consumers enjoy the convenience of hands-free payment, the ability to pay for multiple services using one device, and the security of not having to display a card for payment [2]. According to a Smart Card Alliance Contactless Payments Council White Paper [3] Chase (one of the four largest banks of the USA) has reported that with contactless payments time at the POS is reduced by 30 - 40% and an American Express study has found contactless transactions to be 63% faster than cash and 53% faster than using a traditional credit card. Research also shows that consumers generally spend more per transaction when they use contactless payments, e.g., Chase reported a 20 - 30% increase over cash purchases [3].

Contactless payment applications are particularly attractive to retail segments where speed and convenience of payment are essential (e.g., quick service restaurants, gas stations, convenience stores, parking facilities, transit services, entertainment venues, and unstaffed vending locations). Today almost all of the major payment brands such as American Express, MasterCard and Visa support contactless payments.

There are several concerns among the consumers regarding contactless payment's security over a magnetic stripe based payment system. The primary difference is that the contactless payment device (i.e., card or mobile device) uses RF technology to send payment account information to the merchant's POS terminal instead of requiring the payment card's magnetic stripe to be physically read. While these could open up opportunities for eavesdropping, data corruption, data insertion, and cloning, contactless payment devices are designed to operate at very short ranges (typically less than 10cm from the POS), and can include additional security elements to further enhance the security.

1.2 Problem Statement

Cake Labs is one of the leading POS solution providers for restaurant industry in the USA. *Cake Labs POS* [4] is their main product, which includes a hardware terminal and software for managing orders, issuing bills, handling payments, and obtaining sales statistics. Cake Labs POS works with Cake Labs Wallet, which is a magnetic stripe card designed by Cake Labs to perform payments and transactions

at their POS terminals, which acts as a wallet. Cake Labs Wallet cards are issued per customers' request at Cake Labs partner restaurants. Each wallet card has a unique token number stored in the magnetic stripe which is used to link with customer information stored in the Cake Labs backend databases. POS allows customers to top up the wallet using a method of preference such as cash, credit/debit cards, or using another Cake Labs Wallet card. Currently there is no limit on the amount a customer can top up the wallet. Cancelling of wallet can also be done at POSs and the balance can be redeemed as cash.

In the USA many retail payment applications now support contactless payments. Following its competition, Cake Labs wants to extend their POS and Wallet cards to support contactless payments. This will enable their partners to enjoy the benefits on contactless payments such as increase of sales volumes, speed up of transactions, and improvements in customer acquisition and retention. However, the integration of contactless payments should be accomplished in such a way that it negates the impacts due to security issues such as being vulnerable to spoofing attacks and eavesdropping. Therefore, a significant amount of research is required to select an appropriate contactless card, compatible reader, and design and develop a secure communication and data retention strategy.

Objective of this research is to develop a complete contactless enabled solution to facilitate contactless payments in Cake Labs POS. The proposed solution to be implemented while addressing known and anticipated security issues present in existing contactless payment solutions. Specific contributions to include integrating a suitable contactless reader with necessary security options with the POS system, identifying a suitable contactless-enabled card with necessary security options, developing mobile application(s) for smartphones which can be used as virtual wallet cards and developing a software development kit / library (SDK). The proposed SDK is to support multiple contactless cards and readers.

1.3 Proposed Solution

The proposed solution is to be a complete contactless enabled solution that will enable contactless payments within Cake Labs POS systems and Wallet. NFC (Near Field Communication) is selected as the contactless solution, which is a wireless technology for very-short-range communication with added support for secure communication. More details on NFC is presented in the following chapter.

Figure 1.1 shows an overview of the proposed solution. An NFC reader, which is identified as suitable and fit for the requirements, will be integrated to the existing Cake Labs POS terminals giving them the ability to detect the presence of an NFC card or smartphone, and then initiate a transaction with it.



Figure 1.1 Targeted project deliverables, which include four main modules; NFC card, reader, mobile application and libraries for the framework

For the integration and functionality an NFC framework with a set of capabilities such as contactless and secure data transfer, user and card authentication, and authorization will be developed. The

proposed NFC framework will be compatible with the Linux platform as the Cake Labs POS runs on the Linux platform. This project will also use a wallet card which can be used with security mechanisms such as challenge response. Solution also includes a mobile application for NFC-enabled smartphones. This mobile application may be cross platform to support multiple mobile platforms such as Android and Windows mobile.

2 Literature Survey

Contactless payments have been identified as an emerging trend in mobile payment industry. This literature survey provides a comprehensive review about contactless payment technologies. Review begins with an introduction and evolution of Near Field Communication (NFC) along with standards and protocols of NFC stack. Then types of NFC communication, internal operation of NFC, operating modes, NFC device classifications and NFC software stack are discussed in the Section 2.1. Different types of NFC cards, readers and their comparison are presented in Section 2.2. NFC security related issues and possible solutions are discussed in Section 2.4. In section 2.5 Android NFC stack is presented. Payment card industry guidelines, recommendations, and best practices are presented in Section 2.6. Final section discusses existing NFC payment solutions.

2.1 Near Field Communication

2.1.1 NFC Standard and Protocols

Near Field Communication (NFC) is a set of standards or protocols to communicate between two devices by either touching or bringing into close proximity. NFC has been evolved from a combination of wireless identification and interconnection technologies as illustrated in Figure 2.1. NFC is mainly aimed for mobile or handheld devices to allow simplified transactions, data exchange, and wireless connections between two devices [5].

The communication protocols of such devices are based on RFID (Radio Frequency Identification) technology and standards, which combines the interface of a smartcard and a reader into a single device. This allows two-way communication between endpoints, where earlier systems were one-way only. These standards are defined and extended by the NFC Forum [5].

NFC operates within the globally available and unlicensed radio frequency band of 13.56 MHz, with a bandwidth of 14 kHz. Since the operating frequency is very low, it can be easily integrated into portable devices without the need of much battery power. NFC has a typical working distance with compact standard antennas up to 10cm and typically supported data rates range from 106, 212 to 424 Kbps.

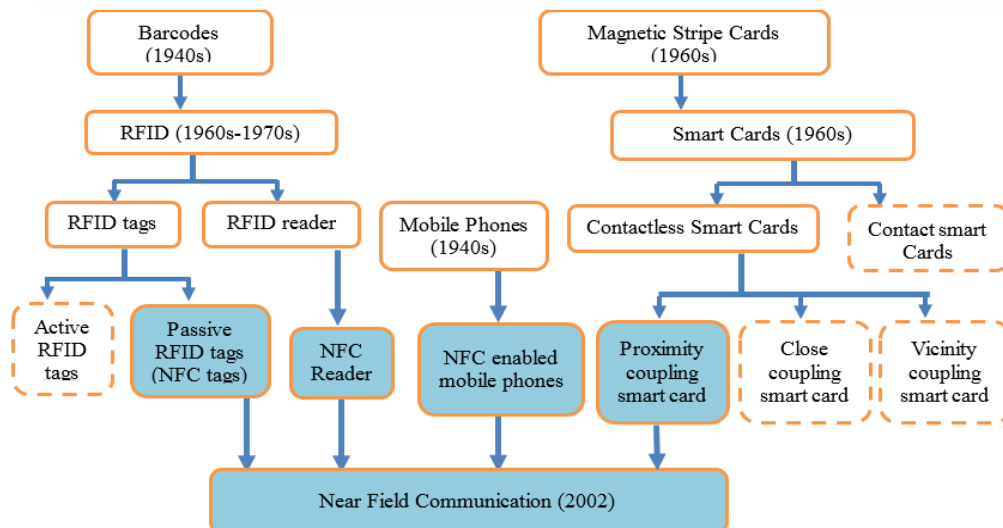


Figure 2.1 Evolution of NFC technology [5]

NFC is based on and compatible with the smart card infrastructure based on ISO/IEC 14443 A and 14443 B standards as well as with the Sony FeliCa card. For the exchange of information between two NFC devices, a new protocol was developed which is defined in ECMA (European Computer Manufacturers Association) 340 and ISO/IEC 18092 [6] standards. Currently major smart phones and

handheld device manufactures like Apple, Samsung, Google, Sony, and Nokia have introduced NFC facilities in their handheld devices.

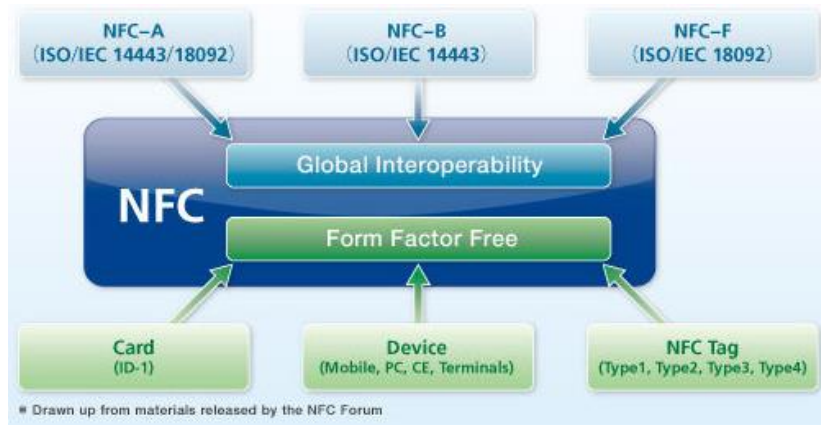


Figure 2.2 NFC objectives [7]

As shown in Figure 2.2, NFC technology has two main objectives:

1. To provide global compatibility with international communication standards
2. To allow different form factors for devices implementing NFC compared to existing contactless terminals

2.1.2 Classifications of NFC Devices

NFC devices can be classified based on two parameters. First is the energy supply, which results in active and passive devices. Second is based on who initiates the communication.

2.1.2.1 Active vs. Passive Devices

An *active device* is one that is powered by a power source (e.g., battery) so that it generates its own electromagnetic (EM) field. Alternatively, a passive device does not have an integrated power source. In NFC, the energy to the passive device is supplied by the EM field generated by the active device.

2.1.2.2 Initiator vs. Target Devices

NFC always occurs between two parties, where one party is called the *initiator* and the other is called the *target*. The initiator is the one that initiates the communication; the target responds to the request that is made by the initiator. An initiator always needs to be an active device, because it requires a power source to initiate the communication. The target, on the other hand, may be either an active or a passive device. If the target is an active device, then it uses its own power source to respond. If it is a passive device, it uses the energy generated from the initiator's EM field.

2.1.3 Operating Modes of NFC devices

2.1.3.1 Reader/Writer Mode

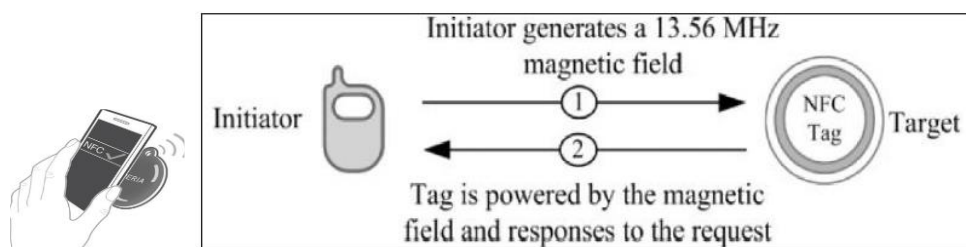


Figure 2.3 Reader/Writer mode [5]

In reader/writer operating mode, as illustrated in the Figure 2.3, an active NFC enabled device such as a mobile phone initiates the wireless communication, and can read and alter data stored in NFC Forum-mandated tags. This enables the mobile user to retrieve the data stored in the tag and take appropriate actions afterwards. The reader/writer mode's RF interface is compliant with ISO/IEC 14443 Type A and Type B. NFC Forum has standardized tag types, operation of tag types and data exchange format between components. The reader/writer operating mode usually does not need a secure area. The process consists of only reading data stored inside the passive tag and writing data to the passive tag. Generic usage scenario of reader/writer mode in NFC shopping is shown below in Figure 2.4.

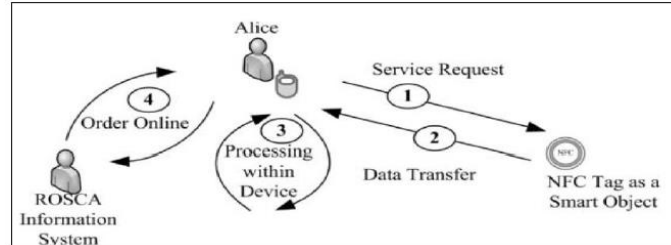


Figure 2.4 Usage scenario of NFC shopping: reader/writer mode [5]

2.1.3.2 Peer-to-Peer Mode

In peer-to-peer mode, two NFC enabled devices establish a bidirectional connection to exchange data. They can exchange data such as virtual business cards, digital photos, and any other kind of data. Peer-to-peer operating mode's RF communication interface is standardized by ISO/IEC 18092 as NFCIP-1. Due to the low transfer speed of NFC, if large amounts of data need to be sent, peer-to-peer mode can be used to create a secondary high speed connection (handover) like Bluetooth or Wi-Fi.

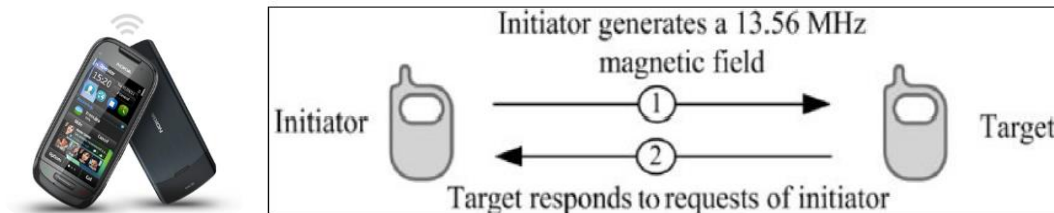


Figure 2.5 Peer-to-peer mode [5]

This mode has two standardized options: NFCIP-1 and LLCP. NFCIP-1 takes advantage of the initiator-target paradigm in which the initiator and the target devices are defined prior to starting the communication. Whereas in LLCP communication the devices are identical. After the initial handshake, the decision is made by the application that is running in the application layer.

Both devices are in active mode during the communication in peer-to-peer mode. Data are sent over a bi-directional half duplex channel which means when one device is transmitting, the other one has to listen and should start to transmit data after the first one finishes, obtaining maximum possible data rate of 424 kbps [1]. Generic usage model of peer-to-peer mode is shown below in Figure 2.6.

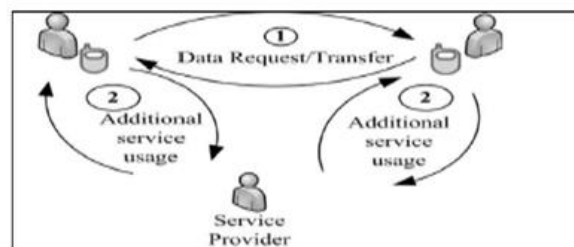


Figure 2.6 Generic usage model of peer-to-peer mode [5]

2.1.3.3 Card Emulation Mode

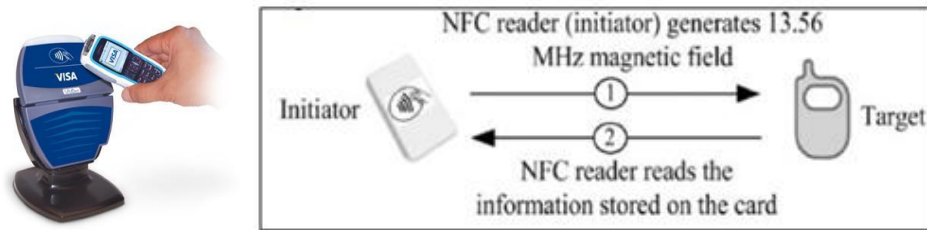


Figure 2.7 Card emulation mode [5]

In card emulation mode, the NFC enabled mobile phone acts as a contactless smartcard. NFC enabled mobile phone either emulates an ISO 14443 smart card or a smart card chip integrated in the phone is connected to the antenna of the NFC module. As the user touches his/her mobile phone to an NFC reader, the NFC reader initiates the communication.

In this mode, the NFC device appears to an external reader much the same as a traditional contactless smart card. This enables contactless payments and ticketing by NFC devices without changing the existing infrastructure [6]. Mobile devices can even store multiple contactless smart card applications. Examples of emulated contactless smart cards are credit card, debit card, and loyalty card. Figure 2.8 shows the generic usage model of card emulation mode.

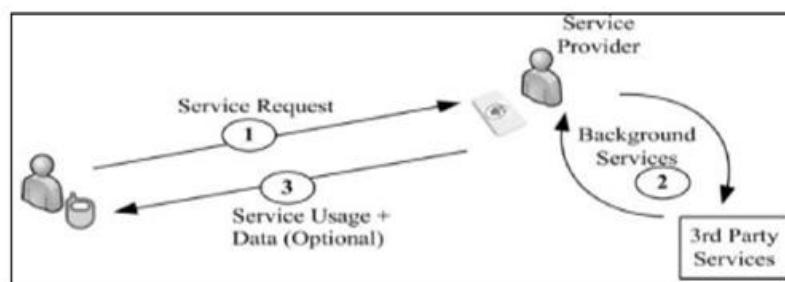


Figure 2.8 Generic usage model of card emulation mode [5]

In card emulation mode, the user interacts with an NFC reader, generally using his/her mobile phone as a smart card. The NFC reader is owned by a service provider which is possibly connected to the Internet as well. The user makes a request to service provider by touching the smartphone with the NFC reader.

Required data are transferred from the phone to the service provider through NFC reader. The service provider runs the required backend service after getting the data. NFC Ticketing usage model using card emulation mode is depicted in Figure 2.8.

Table 2.1 Benefits of various operating modes of NFC [8]

Reader/Writer Mode	Peer-to-Peer Mode	Card Emulation Mode
<ul style="list-style-type: none"> Increases mobility Decreases physical effort Ability to be adapted in many scenarios Easy to implement 	<ul style="list-style-type: none"> Easy data exchange Device pairing 	<ul style="list-style-type: none"> Physical object elimination Access control

Table 2.2 Interaction with the service provider [5]

Operating Mode	Initiator	Target	Connecting to Service Provider	Awareness of Service Provider
Reader/Writer	Mobile device	NFC tag	Through internet	Yes
Peer-to-peer	Mobile device	Mobile device	Through internet	Yes
Card emulation	NFC reader	Mobile device	Through NFC reader	No

The prominent mode of NFC is card emulation mode, because NFC yields two big improvements in this mode: elimination of a physical object and providing access control using a mobile device. Commercially available applications for payment, electronic key, ticketing, etc., generally use the card emulation mode [9].

2.1.4 NFC Protocols Stack, Interface, and Specifications

NFC Interface and Protocol (NFCIP) is standardized in two forms as NFCIP-1 and NFCIP-2. NFCIP-1 defines the NFC communication modes on the RF layer and other technical features of the RF layer. NFCIP-2 supports mode switching by detecting and selecting one of the communication modes.

2.1.5 NFCIP-1

NFCIP-1 is defined in ISO/IEC 18092, ECMA 340, and ETSI TS (European Telecommunications Standards Institute Technical Specification) 102 190 [10]. This standard defines two communication modes as active and passive. The standard specifies modulation schemes, coding, transfer speeds (106/212/424 Kbps), and frame format of the RF interface, as well as initialization schemes and conditions required for data collision control during initialization. Furthermore, it defines a transport protocol including protocol activation and data exchange methods.

2.1.6 NFCIP-2

NFCIP-2 is specified in ISO/IEC 21481, ECMA 352 and ETSI TS 102 312. It specifies the communication mode selection mechanism and is designed not to disturb any on-going communication at 13.56 MHz for devices implementing ISO/IEC 18092 (i.e., NFCIP-1), ISO/IEC 14443 or ISO/IEC 15693. For example, long range-vicinity communication and RFID tags [11]. Although all of the ISO/IEC 18092, ISO/IEC 14443 and ISO/IEC 15693 standards specify 13.56 MHz as their working frequency, they may specify distinct communication modes.

2.1.7 Protocol Layer

As illustrated in the Figure 2.10 there are six divisions in the protocol layer with four major protocol types, namely Type 1 (Topaz), Mifare Classic, Mifare Ultralight and LLCP (P2P).

- Type 1 (Topaz) – Type 1 tags use a format sometimes called the Topaz protocol. It uses a simple memory model which is either static for tags with memory size less than 120 bytes or dynamic for tags with larger memory. Bytes are read/written to the tag using commands such as RALL, READ, WRITE-E, WRITE-NE, RSEG, READ8, WRITE-E8 and WRITE-N8.
- Mifare classic – MIFARE classic tags are storage devices with simple security mechanisms for access control. They use an NXP proprietary security protocol for authentication and ciphering. This encryption was reverse engineered and broken in 2007 [12] [13] .
- Mifare ultralight – These tags are similar to Topaz tags. They have a static memory layout when they have less than 64 bytes available and a dynamic layout otherwise. The first 16 bytes of memory contain metadata like a serial number, access rights, and capability container. The rest is for the actual data. Data is accessed using READ and WRITE commands.
- LLCP (P2P) – LLCP enhances the basic functionalities provided by NFCIP-1 protocol as well. LLCP provides five important services, namely connectionless transport; connection oriented

transport; link activation, supervision and deactivation; asynchronous balanced communication; and protocol multiplexing [13].

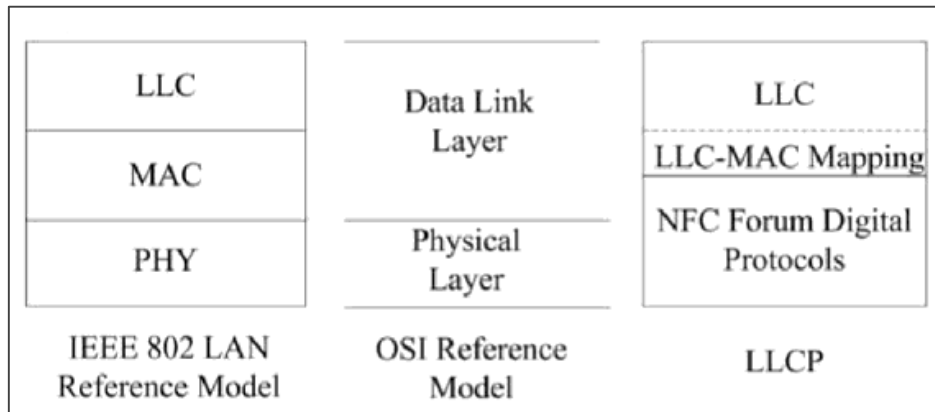


Figure 2.9 Relationship between LLCP and OSI reference model [5]

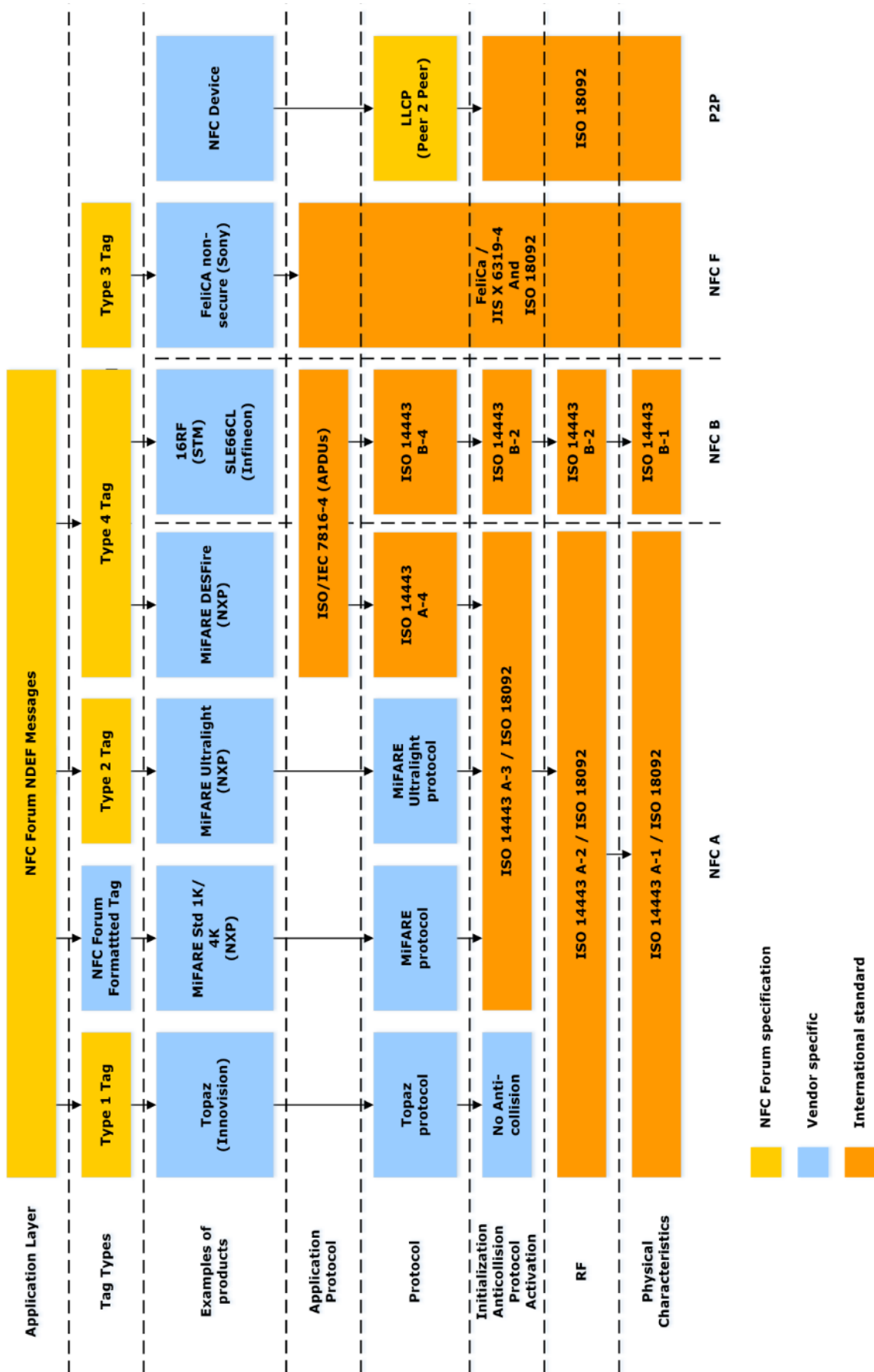


Figure 2.10 NFC protocol stack [14]

2.1.7.1 NDEF and RTD

NDEF specification is a standard defined by the NFC Forum [15]. NDEF is a data format to exchange information between NFC devices. An NDEF message is exchanged when an NFC device is in the proximity of an NFC Forum mandated tag (message get exchanged over the NFC Forum LLCP).

NDEF is a binary message format that encapsulates one or more NDEF records (i.e., application defined payloads) into a single message as depicted in Figure 2.11. A record is the unit for carrying a payload within an NDEF message. Each record consists of a payload up to $2^{32}-1$ octets in size. Records can be chained together to support larger payloads as well.

NDEF records are of variable length with a common format. Record type names (RTD) are used by NDEF applications to identify the semantics and structure of the record content. NFC Forum defines various record types for specific cases; smart posters, URIs, digital signature, and text [15]. Figure 2.12 illustrates a summarized view of the protocol stack of NFC based on different operating modes.

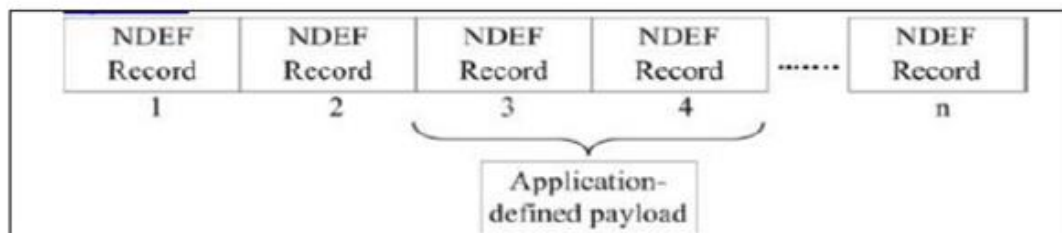


Figure 2.11 NDEF structure [15]

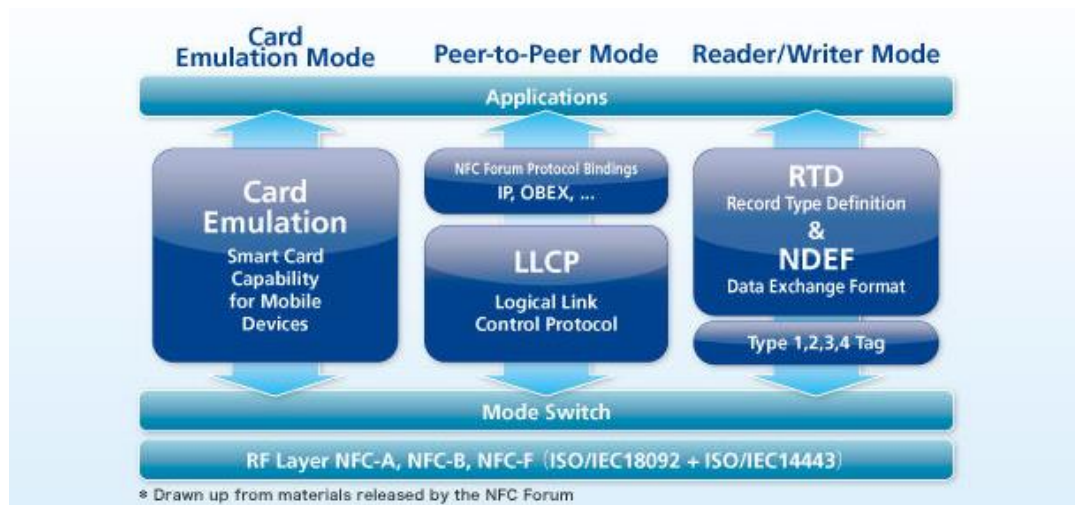


Figure 2.12 Protocol stack for each operating mode [16]

2.1.8 Operation of NFC

Typical NFC communication model has two communicative terminals. The Initiator is the one who wishes to communicate and starts the communication. The Target receives the initiator's communication request and sends back a reply.

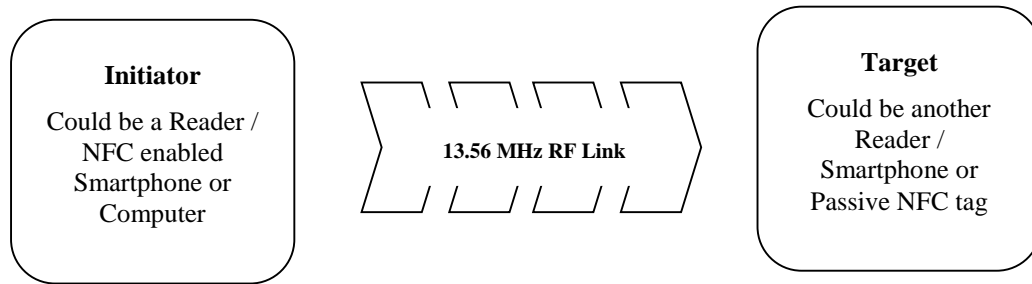


Figure 2.13 NFC communication model

NFC devices communicate via magnetic field induction, where two loop antennas are located within each other's near field, effectively forming an air-core transformer. The reader continuously generates an RF carrier signal (at 13.56 MHz), watching always for modulation to occur. Detected modulation of the field would indicate the presence of a tag. A tag enters the RF field generated by the reader. Once the tag has received sufficient energy to operate correctly, it divides down the carrier and begins clocking its data to an output transistor, which is normally connected across the coil inputs. The tag's output transistor shunts the coil, sequentially corresponding to the data which is being clocked out of the memory array. Shunting the coil causes a momentary fluctuation (dampening) of the carrier wave, which is seen as a slight change in amplitude of the carrier. The reader's peak detector hardware detects the amplitude modulated data and processes the resulting bit stream according to the encoding and data modulation methods used [12].

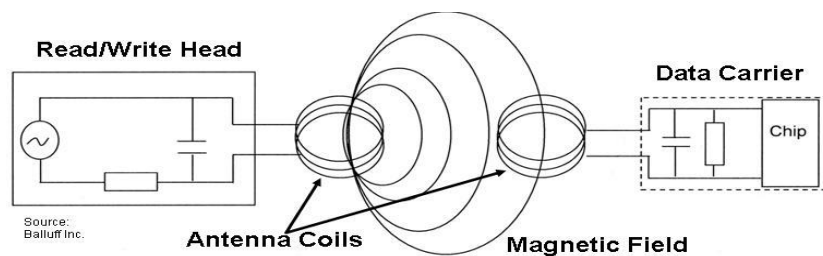


Figure 2.14 NFC communication model internal architecture [17]

NFC employs two different coding to transfer data. If an active device transfers data at 106 Kbit/s, a Modified Miller coding with 100% modulation is used. In all other cases Manchester coding is used with a modulation ratio of 10%, as summarized in Table 2.3.

Table 2.3 Modulation and coding schemes based on device type and data rate

Speed	Active Device	Passive Device
106 kbps	Modified Miller, 100% ASK	Manchester, 10% ASK
212 kbps	Manchester, 10% ASK	Manchester, 10% ASK
424 kbps	Manchester, 10% ASK	Manchester, 10% ASK

Due to the coupling of the coils of initiator and target, a passive target also affects the active initiator. A variation in the impedance of the target causes amplitude or phase changes to the antenna voltage of the initiator, detected by it. This technique is called load modulation. Load modulation is carried out in target mode using an auxiliary carrier at 848 kHz which is modulated by the baseband and varies the impedance of the target device.

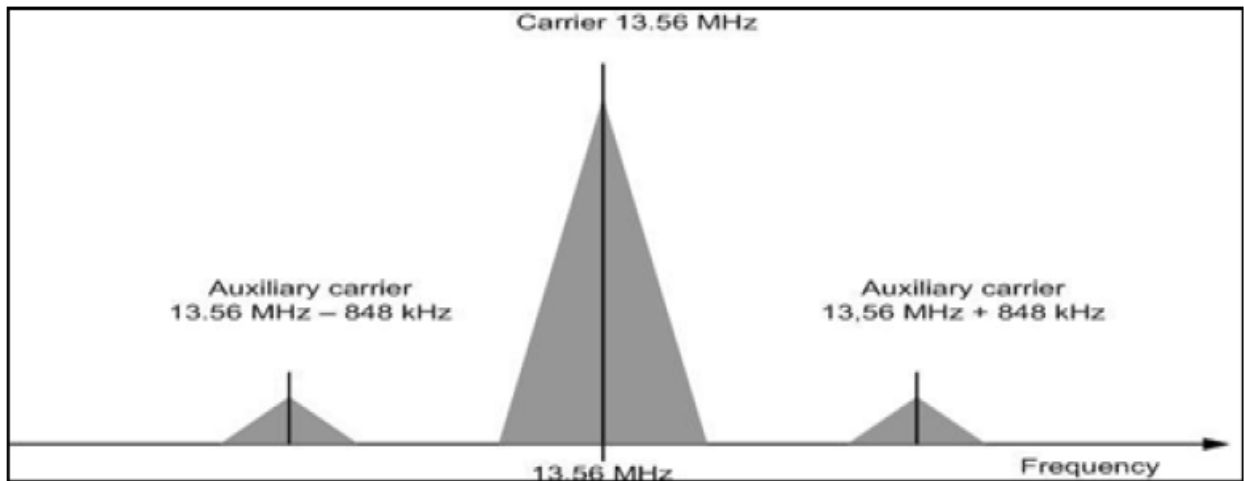


Figure 2.15 Modulation spectra showing load modulation [6]

2.1.9 NFC Software Stack

2.1.9.1 LibNFC

LibNFC is a low-level NFC SDK and Programmers API released under the GNU Lesser General Public License. It supports all major operating systems including GNU/Linux, Mac OS X and Windows. This library supports various NFC hardware devices: dongles, flat and OEM devices. The library currently supports modulations for ISO/IEC 14443 A and B, FeliCa, Jewel/Topaz tags and Data Exchange Protocol (P2P) as target and as initiator [18].

Libnfc provides the following features for easy NFC application development:

- Complete coverage of low-level PN53x chipset commands
- Written in pure and plain C for portability and speed
- Released under the open source GNU Lesser General Public License
- Supports ISO14443-A/B, ISO18092 (p2p) and JIS X 6319-4 modulation
- Implements NXP MIFARE Classic and Sony FeliCa protocol
- Can transform your NFC device into a initiator (reader) or target (tag)
- Allows complete control of the transmitted data frames
- Accepts transmission of incorrect or empty CRC information
- Provides a way to send arbitrary and incorrect parity bits
- Supports sending of incomplete byte-frames (e.g. four bits error message)
- Allows configuration of advanced chip registers and modulation settings

2.1.9.2 Open NFC

Open NFC [19] is an open source project on implementing a software stack associated with NFC functionalities on top of NFC controller chipsets. It supports a wide variety of NFC tags and protocols through both high level and low level interfaces. If someone is hoping to develop applications for NFC, this open source project is a great place to start. Following are some of its main features:

- Supports different NFC modes of operation: reader mode, card emulation mode, and peer-to-peer mode
- Supports connection handover, including Bluetooth pairing and Wi-Fi pairing
- Supports card emulation on secure elements (UICC or other secured chipset), and integrates a security stack to filter access to these elements

- NFC hardware-independent stack, based on an NFC Hardware Abstraction Layer (HAL). NFC HAL modules for some NFC chipsets are provided, including an NFC simulator software
- Portable code, with several reference porting (“Open NFC Editions”). The stack comes in three different architectures to better fit the different environments: monolithic, user-driver, or client-server [20]

Minimum requirements to install and work with Open NFC stack include:

- Roughly 128 KB RAM for heap and stack
- About 500 KB for the code
- A timer with a resolution of at least 1ms

2.1.9.2.1 Open NFC Software Stack

Following components are available in the Open NFC software stack [21]:

- Open NFC core module:
 - Written in C language, portable and hardware independent module.
 - Contains the logic necessary to handle the different NFC protocols and modes of operations.
 - Can be transparently compiled as one module (library) in monolithic porting, or as two modules in client-server porting or user-kernel porting.
- Editor-specific bindings
 - These are the supporting modules for communicating with the native open NFC API.
 - These bindings are dependent on the open NFC editions, not the core Edition.
- NFC HAL module
 - Used to establish communication between the core module and the lower layer modules.
 - This lower layer is responsible for managing the specific hardware of the target platform.
 - NFC controller itself, and the other hardware components involved in the communication.

2.1.9.2.2 Open NFC Interfaces

Open NFC interfaces [22] can be classified at different levels, beginning from high-level interfaces for simplified NFC applications and low-level interfaces for complicated NFC applications with fine tuning. Table 2.4 list a summary of interfaces.

Furthermore, Open NFC supports various protocols and tags in different operating modes, including NFC-forum compliant tags, Mifare tags, Felica tags and ISO 1444-3/4, 7816-4, 15693, 18092 standards [22].

Table 2.4 Open NFC interfaces summary

High-Level Interfaces	NDEF Messages	These functions allow easy handling (parsing and building) of the message format defined by the NFC Forum.
	Bluetooth and Wi-Fi Pairing	Support connection handover in just a few lines of code.
	Read/Write to Any Tag	Enable easy and transparent access to tags of different technologies.
	P2P	Peer-to-peer functions allow easy exchange of information in different modes over the LLCP exchange. Integrated SNEP functions are also available.
	Virtual Tags	This feature greatly simplifies the task of emulating an NFC Tag.

Intermediate Level Interfaces	Card Emulation	This allows fine control over the events and data exchanged when the NFC device is behaving like a tag.
	Protocol-Level Data Exchanges (“raw”)	This enables the application to fully specify the data exchanged with the cards, while taking care of some specificities like management of the checksums.
Low-Level Interfaces	NFC Controller	Allows direct (but filtered) access to NFC controller hardware to retrieve or set some internal parameters, including RF control, power modes, etc.
	Security Stack	Provides a secured framework to limit access to some AIDs on secure elements to identified applications only.
	Secure Elements	Allows access to a secure element through a special pipe.

2.1.9.2.3 Open NFC Security Stack

Open NFC includes a security stack (see Figure 2.16) designed to protect the access from applications to the secure elements like UICC (Universal Integrated Circuit Card) and ESE (Embedded Secure Element). This is needed because NFC hardware provides a new channel to communicate with these elements. A malware application could therefore potentially send destructive commands to the secure elements; for example failing several authentication attempts, resulting in locking the secure element. The security stack simply enforces an access policy to the secure element.

The Open NFC stack attempts to load a specific applet from the secure element(s) it finds. If the applet is not available in the secure element, the former behavior is used as fallback, as if no Security Stack was available. If the applet is found, its contents is verified and used. The applet expected by Open NFC contains schematically a white list of applications (on the phone) that can access applets and specific commands in the secure element. The Open NFC stack enforces this policy and rejects commands from unauthorized applications.

Following sequence of events happening when an application wants to communicate with a secure element through Open NFC security stack:

1. Open NFC stack reads the applet (if available) from the secure element and builds an internal representation of the Access Control List (ACL) it contains. Note that to exchange data with the secure element, some porting function may be required.
2. When an application wants to “talk” with the secure element, it must first call the *WSecurityAuthenticate()* function. This allows the Open NFC stack to map the application with its ACL entries.
3. When the application opens afterwards a connection to the secure element or exchanges data, the Open NFC stack enforces the policy read from the applet and blocks the calls that are not allowed in that file.

A Java application called AC File Generator Tool is included in the Open NFC core module delivery to ease the creation of the contents file for the applet [23].

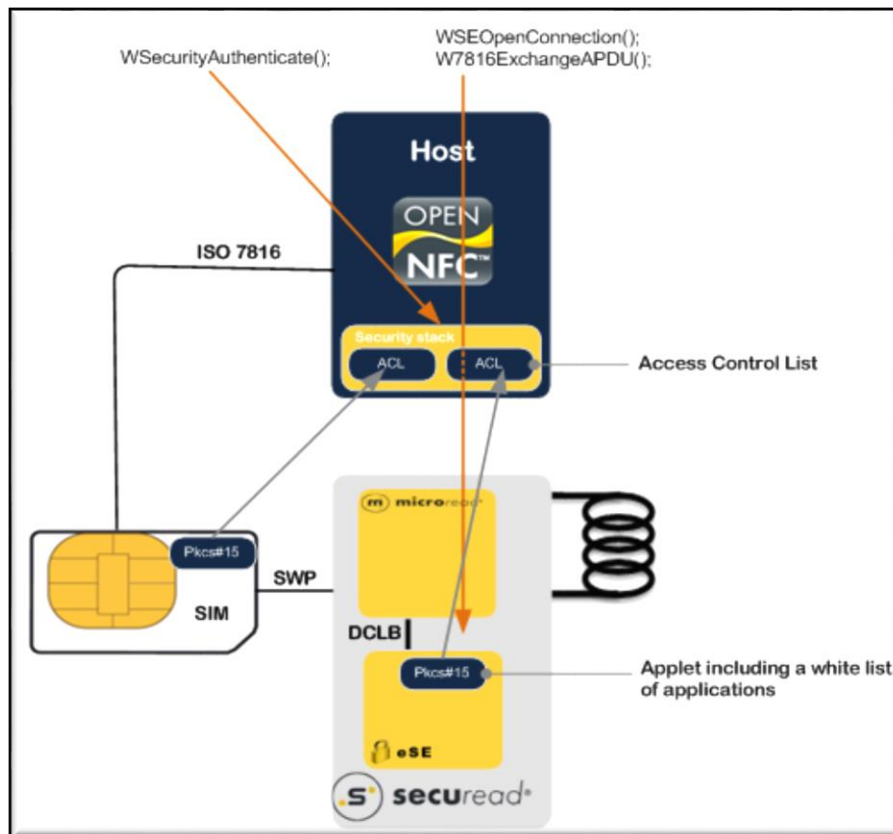


Figure 2.16 Components of the Open NFC security stack [23]

2.2 Card Types

An NFC tag is an RFID device incorporating a silicon memory chip connecting to an external antenna. Tag does not have its own power source (i.e., passive). The passive tag absorbs a small portion of the energy emitted by the reader, and starts sending modulated information when sufficient energy is acquired from the RF field generated by the reader. Data modulation is accomplished by either direct modulation, FSK or Phase modulation.

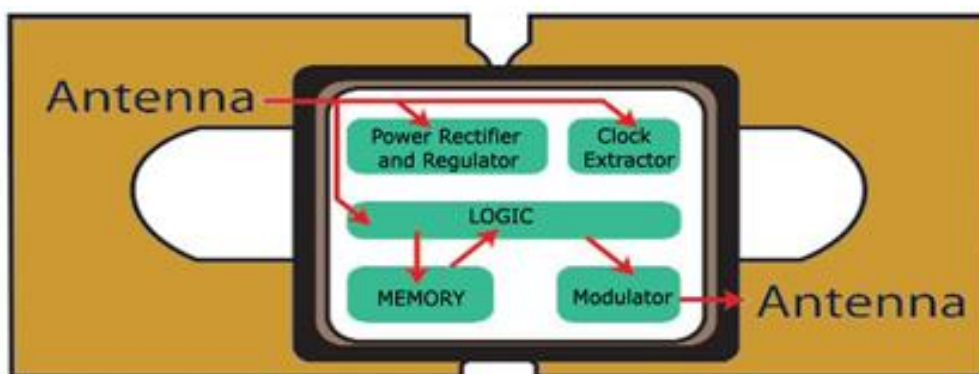


Figure 2.17 Internal hardware of NFC tag [24]

NFC Forum mandated several tag types and those are summarized in Table 2.5. It shows that only difference between type 1 tag and type 2 tag is memory capacity. Type 3 tag is different from type 4 tag in terms of memory capacity and data transferring speed. Mifare Classic is not an NFC forum compliant tag, although reading and writing of the tag is supported by most of the NFC devices as they ship with an NXP chip. Detailed specifications of each tag type available at [25].

Table 2.5 Summary of NFC tag types

Type	Standard	Capabilities	Memory	Speed	Cost	Examples
1	ISO/IEC 14443A	Read Only Read and Write Collision protection	96 B - 2 KB (expandable) Can be write protected	106 kbit/s	1.25 \$	Innovision Topaz [26]
2	ISO/IEC 14443A	Read Only Read and Write Collision protection	48 B - 2 KB Can be write protected	106 kbit/s	1.35\$	NXP Mifare Ultralight [27]
3	(JIS) X 6319-4	Read Only Read and Write (pre-configured at manufacture)	Up to 1 MB	212 kbit/s	1-2 \$	Sony Felica [28]
4	ISO/IEC 14443 (A and/or B)	Read Only Read and Write (pre-configured at manufacture)	Up to 32 KB	106 kbit/s	2 \$	NXP DESfire [29], NXP SmartMX [30] with JCOP

2.3 NFC Coupling Devices (Readers and Writers)



NFC coupling devices are always called readers, even if they can write. They are usually microcontroller-based (e.g., NFC enabled phones) with an integrated circuits that is capable of generating radio frequency at 13.56 MHz. It also includes other components such as encoders, decoders, antenna, comparators, and firmware designed to transmit energy to a tag and read data back from it by detecting the backscatter modulation. The reader continuously emits RF carrier signals, and keeps observing the received RF signals for data. No standard API, hardware, and/or protocols seem to be available for NFC readers. Hence, most applications are written to vendor provided, device-specific APIs. There is however one exception, readers for smartcards with RF-interface use (PC/SC) standards.

PC/SC is the de-facto standard to interface Personal Computers with Smart Cards (and smartcard readers/writers of course), and is available on most operating systems, including Windows and Linux [31]. This standard only applies to CPU contact cards. Version 2.0 also dictates PIN pad to card communications. Apple, Oracle-Sun, Linux and Microsoft all support this standard [32].

Variety of readers are available for 13.56MHz frequency. They can be classified into mainly two categories as active and passive readers. *Active readers* run the RFID protocol stack on the microcontroller (in the reader itself). In *passive readers*, the RFID protocol runs on the host system. Entire software protocol is implemented on host (typically on top of a PC or embedded OS). RFID chip is just a dump transceiver device (sends and receives frames and does not care about the protocol). Two of the well-known examples are DL533N OEM Reader [33] (see Figure 2.18) and OpenPCD (see Figure 2.19). Table 2.6 presents a comparison of the two readers.

2.3.1 Comparison of existing NFC Coupling Devices

Table 2.6 Comparison of exiting NFC coupling devices

Device	DL533N OEM	OpenPCD
Manufacturer	Digital Logic Ltd	The OpenPCD project
Model	DL533N	OpenPCD 2 RFID Reader
Operating Frequency	13.56 MHz	13.56 MHz
Interface Type	USB	USB
Supported NFC Tag/Card Types	<ul style="list-style-type: none"> • MIFARE® Classic • MIFARE DESFire • MIFARE Plus • MIFARE Ultralight • Sony Felica 	<ul style="list-style-type: none"> • MIFARE® Classic • MIFARE DESFire • MIFARE Plus • MIFARE Ultralight • Sony Felica
Dimensions(WxDxH)	95 x 63 x 11 mm	85 x 53 x 15 mm
Transmission Rate	Up to 848 kbit/s	Up to 424 kbit/s
Contactless (RFID) Smart Card Interface / Protocols	<ul style="list-style-type: none"> • ISO 14443 A • ISO 14443 B • MIFARE • FeliCa 	<ul style="list-style-type: none"> • ISO 14443A • ISO 14443B (FeliCa) • ISO 15693 • Mifare and ICODE
OS Support	<ul style="list-style-type: none"> • Windows® • Mac OS X • Linux® 	<ul style="list-style-type: none"> • Linux 2.4/2.6 Kernel (no driver needed) • Microsoft Windows XP/Vista/7 • Mac OS X (Leopard, Lion etc. - no driver needed)
Special notes	<ul style="list-style-type: none"> • Uses NXP PN533 transceiver module • Firmware support for libnfc 	<ul style="list-style-type: none"> • Uses NXP PN533 transceiver module • Firmware support for libnfc • Open source product for hardware, software and firmware. • PCB design Schematics and Layout are available free and open source
Figure	 <p>Figure 2.18 NFC USB reader DL533N OEM [33]</p>	 <p>Figure 2.19 OpenPCD RFID reader [34]</p>
Price	51 \$	52 \$

According to the comparison among available NFC coupling devices (reader/writers) DL533N OEM is more convenient as an NFC reader, because it has the highest transmission rate and multiple OS supports for fare price.

2.4 Security

As many applications of NFC deal with sensitive data such that cash, credit card details, usernames, and passwords, NFC cards and readers are becoming popular targets for various forms of attacks. They are vulnerable to security attacks includes eavesdropping, data corruption, unauthorized modification, and relay attacks. Next, we discuss both the threats and potential solutions to mitigate those threats.

2.4.1 Threats

2.4.1.1 Eavesdropping

Wireless communication obviously has the issue of eavesdropping, and this is common to NFC as well. When two devices communicate via NFC, an attacker can use an antenna to receive the transmitted signals. If attacker has the required knowledge on how to extract the transmitted data out of the received RF signal, he/she can read transmitted data. In eavesdropping whether attacker is able to retrieve a usable RF signal depends on how close the attacker is. Distance to attacker is determined using the following parameters [36]:

- RF filed characteristic of the given sender device (i.e., antenna geometry, shielding effect of the case, PCB (Printed Circuit board), and environment).
- Characteristic of the attacker's antenna (i.e., antenna geometry, possibility to change the position in all three dimensions).
- Quality of the attacker's receiver.
- Quality of the attacker's RF signal decoder.
- Setup of the location where the attack is performed (e.g., barriers like walls or metal, noise floor level).
- Power sent out by the NFC device.

When a device is sending data in active mode, eavesdropping could be done up to a distance of about 10m. Whereas when the sending device is in passive mode, this distance is decreased to about 1m.

2.4.1.2 Data Corruption

Instead of just listening, an attacker can also try to modify the data which are transmitted via the NFC interface. In the simplest case, the attacker can just disturb the communication such that the receiver is unable to understand the data sent by the other device. This can be achieved by transmitting valid frequencies of the data spectrum at correct time. The correct time can be calculated if the attacker has a good understanding of the used modulation scheme and coding. This attack is not too complicated, but it does not allow the attacker to manipulate the actual data and it is basically a Denial of Service (DoS) attack [36].

2.4.1.3 Data Modification

If attacker wants the receiving device to actually receive some valid, but manipulated data he/she could use data modification. The feasibility of this attack highly depends on the applied strength of the amplitude modulation. This is because the decoding of the signal is different for 100% and 10% modulation. In 100% modulation the decoder basically checks the two half bits for RF signal on (no pause) or RF signal off (pause). In order to modify the data, attacker fills up a pause in the modulation with the carrier frequency and generate a pause of the RF signal, which is received by the legitimate

receiver, to make the decoder to understand a one as a zero. In 10% modulation the decoder measures both signal levels (82% and full) and compares them. In case they are in the correct range the signal is valid and gets decoded. An attacker could try to add a signal to the 82% signal, such that the 82% signal appears as the full signal and the actual full signal becomes the 82% signal. This way the decode would decode a valid bit of the opposite value of the bit sent by the correct sender.

Modified Miller encoding with 100% ASK this attack is feasible for certain bits and impossible for other bits (an attacker cannot change a bit of value 0 to a bit of value 1, but can change a bit of value 1 to a bit of value 0), but for Manchester coding with 10% ASK this attack is feasible on all bits [36].

Also an attacker can modify the data already stored in NFC tag even when tag is not communicating. This attack overwrites the data of a tag or physically replaces it. Overwriting can be done easily if the original tag is a writeable tag without any security measures (or these measures are broken). The aim of this attack is to falsify the original tag, e.g. for phishing purposes [37].

2.4.1.4 Data Insertion

An attacker can insert messages into the data exchange between two devices. This is only possible, in case when the answering device takes a very long time to answer. The attacker could then send his data earlier than the valid receiver. This would only be successful if the transmission is finished, before the answering device starts with its answer. Otherwise the message would be corrupted [36].

2.4.1.5 Man-in-the-Middle-Attack

In the classical man-in-the-middle attack, two parties which want to talk to each other are tricked into a three party conversation by an attacker. As seen in Figure 2.20 Alice uses active mode and Bob would be in passive mode, Alice generates the RF field and sends data to Bob. In case the third-party Eve is close enough, she can eavesdrop the data sent by Alice. Additionally she must actively disturb the transmission of Alice to make sure that Bob does not receive the data. But this can also be detected by Alice. In case Alice detects the disturbance, Alice can stop the key agreement protocol. If Alice does not check for active disturbance and so the protocol can continue.

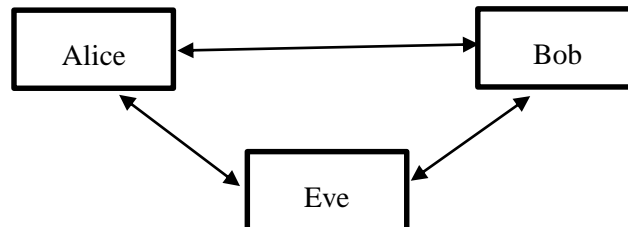


Figure 2.20 Man-in-the-middle setup

In the next step Eve needs to send data to Bob. That is already a problem, because the RF field generated by Alice is still there, so Eve has to generate a second RF field. This causes two RF fields to be active at the same time. It is practically impossible to perfectly align these two RF fields. Thus, it is practically impossible for Bob to understand data sent by Eve [36].

2.4.1.6 Data Stealing

An attacker can simply use NFC reader to steal data from NFC enabled applications but the NFC reader must be brought into a convenient distance from NFC tag. If there is no security implementation in NFC application, then attacker can read directly. Even though there is any security mechanism there is a chance to decrypt the data stored in NFC cards. Moreover attacker can clone NFC tag by writing the stolen data into another NFC tag.

2.4.1.7 Relay Attack

As shown in Figure 2.21 an attacker forwards the request of the reader to the victim and relay back its answer to the reader in real time in order to carry out a task by pretending to be the owner of the victim's smart card. This attack focuses on the extension of the range between the NFC token (e.g., a card) and the reader. Two NFC enabled devices are necessary to perform this with one acting as a reader and another acting as a card emulator. The access victim system will not be able to detect the attack because it will think a card is actually in front of it. In the attack scenario the attacker holds the NFC reader near the victim's card and relays the data over another communication channel to a second NFC reader placed in proximity to the original reader that will emulate the victim's card [38]. This enables the attacker to mediate all the communication between the reader and victim's card.

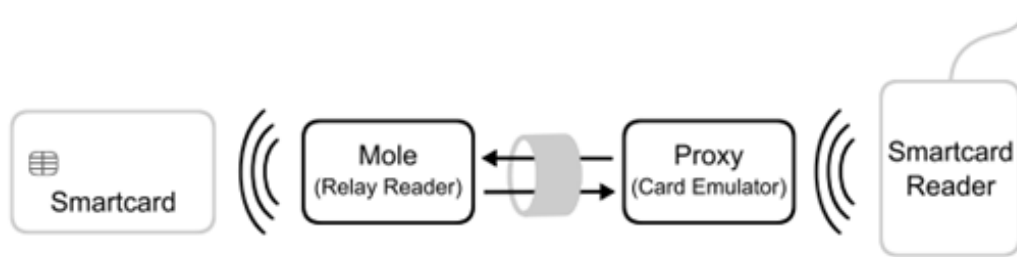


Figure 2.21 Relay attack scenario [38]

2.4.1.8 Malware

Mobile malware is also starting to become an issue. The malware could easily sniff sensitive information such as credit card data stored or used on the NFC device and forward this to the attacker over an NFC channel or the web [39].

2.4.1.9 Android Beam

Android Beam can be used to pass information between devices or from a tag to a device. The information that can be passed includes contacts, URLs, applications, etc. There is no confirmation required on the receiving side and the device runs the associated application automatically. This opens a whole new can of worms as malicious applications could be transferred to devices without the user requiring to confirm the transfer. Attacker could also transfer a malicious URL and either trick the user into clicking it or exploit a browser bug to visit the malicious website and download malicious content [39].

2.4.1.10 Spoofing

In a spoofing attack, a third-party pretends to be another entity to induce a user to tap its device against the tag. This is possible if an attacker compromised an NFC tag (e.g., a smart poster) with a malicious tag that could force a user to execute a malicious code, aided by the fact that some mobile devices are configured to execute commands received from NFC tags automatically [38].

2.4.2 Solutions

While the attack potential of NFC cards is high simple cryptographic and authentication solutions can address most of those issues. Following is a list of potential solutions to mitigate such attacks:

- NFC devices can detect eavesdropping and data corruption attacks because they can check the RF field continuously, while they are transmitting data. If an NFC device performs this checking, it will be able to detect the attacks because power which is needed to corrupt the data is significantly higher than the power which can be detected by the NFC device.

- Establishing a secure channel such as SSL which the transmitted data are encrypted, between two NFC devices is the best approach to protect against eavesdropping and any kind of data modification attack. Due to the inherent protection of NFC against man-in-the-middle attacks it is rather easy and straightforward to setup a secure channel. For this, Hybrid cryptosystem concept can be used. Hybrid cryptosystem is a sum of public key cryptosystem for the key encapsulation and symmetric key cryptosystem for the data encapsulation. A standard key agreement protocol like Diffie-Hellmann based on RSA or Elliptic Curves could be applied to establish a shared secret between two devices via key encapsulation. The shared secret can then be used to derive a symmetric key like 3DES or AES, which is then used for the secure channel providing confidentiality, integrity, and authenticity of the transmitted data [36].
- Relay attack is constrained by a timing issue. Because of the physical distance between the two NFC devices, the packets that are relayed will take longer to be transferred to the destination. RFID technology has some constraints on the time range between a challenge and response, named FWT; exceeding this limit will cause the failure of the attack. Principal countermeasures to prevent relay attacks are:
 - Faraday cages – this is the simplest measure. It consists in shielding the card at the user’s side with a box that is called a Faraday cage.
 - Signing of the data would result in more security, but a determinant factor is the computational power of the cards used and the ability to verify the signer is a reasonable time.
 - Adoption of distance bounding protocols of the RFID system, so that the reader knows whether the card is presented inside the electromagnetic held, or a relay attack is being performed [38].
- The principal countermeasure against spoofing attack is to properly configure the device to prompt a message before executing commands through NFC (e.g., opening a URL).

2.5 Android NFC Stack

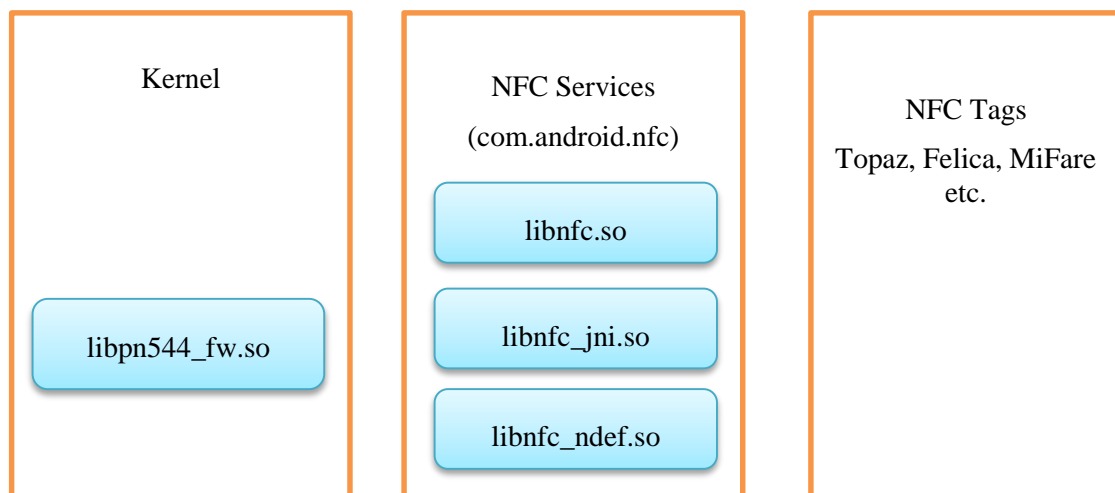


Figure 2.22 Android NFC stack with different libraries

Android NFC stack is a framework of APIs which are based around an NFC Forum standard called NDEF (NFC Data Exchange Format) which support NFC technology. Android NFC stack allow users to share small payloads of data between an NFC tag and an Android-powered device, or between two Android-powered devices via NFC technology.

There are three major components of the Android NFC stack, namely the *Kernel*, *NFC Services* and the *tag* (or device) itself. The Kernel consists of the NFC driver called “libpn544_fw.so”, which will respond and interact with NFC signals. NFC services available in a typical Android device is called as *com.android.nfc*. It relies on three main driver components: *libnfc.so*, *libnfc_jni.so* and *libnfc_ndef.so*. These components are divided on the basis of the classification of NFC data (JNI, NDEF or any other). Therefore, when any NFC enabled tag (or other NFC enabled device) is brought close to an Android NFC device, the kernel component *libpn544_fw.so* calls the NFC services. Once the NFC services are called, they receive the NFC data and store the information by dividing it into suitable categories. Most important library module is the *libnfc_ndef.so* driver component, which is responsible for the NDEF data communication between the NFC tag and Smartphone.

2.6 Payment Card Industry Guidelines, Recommendations and Best Practices

Since Cake Labs POS is primarily used by the restaurants based in the USA, payments card related rules and regulations applicable to the USA need to be considered as well. We also present general guidelines and best practices.

2.6.1 PCI PA-DSS Requirements

PCI PA-DSS is the Payment Card Industry (PCI) Payment Applications-Data Security Standard (PA-DSS) [40]. It consists of 14 requirements to ensure vendors provide products which support merchants’ efforts to maintain PCI DSS compliance and eliminate the storage of sensitive cardholder data. It is important to distinguish between mobile payment applications used to initiate payments and those mobile payment applications used by merchants for payment card acceptance and processing. The PA-DSS program addresses payment applications used to accept and process payment for goods and services. Applications used for payment-initiation—for example, those downloaded by consumers onto their mobile phones and used for consumers’ personal shopping—are seen as similar to the payment card in a consumer’s wallet and PA-DSS is not applicable for these mobile applications.

Table 2.7 only includes PA-DSS requirements which applies for the contactless payment implementation of the POS.

Table 2.7 PCI DSS requirements

Requirements	Description
Do not retain full track data, card verification code or value or PIN block data	<ul style="list-style-type: none"> Delete sensitive authentication data stored by previous payment application versions Delete any sensitive authentication data (pre-authorization) gathered as a result of troubleshooting the payment application.
Protect stored cardholder data	<ul style="list-style-type: none"> Securely delete cardholder data after customer-defined retention period. Mask PAN when displayed so only personnel with a business need can see the full PAN. Render PAN unreadable anywhere it is stored (including data on portable digital media, backup media, and in logs). Protect keys used to secure cardholder data against disclosure and misuse. Implement key-management processes and procedures for cryptographic keys used for encryption of cardholder data. Provide a mechanism to render irretrievable cryptographic key material or cryptograms stored by the payment application. Securely delete cardholder data after customer-defined retention period. Mask PAN when displayed so only personnel with a business need can see the full PAN.

	<ul style="list-style-type: none"> • Render PAN unreadable anywhere it is stored (including data on portable digital media, backup media, and in logs). • Protect keys used to secure cardholder data against disclosure and misuse. • Implement key-management processes and procedures for cryptographic keys used for encryption of cardholder data. • Provide a mechanism to render irretrievable cryptographic key material or cryptograms stored by the payment application.
Provide secure authentication features	<ul style="list-style-type: none"> • Use unique user IDs and secure authentication for administrative access and access to cardholder data. • Use unique user IDs and secure authentication for access to PCs, servers, and databases with payment applications.
Protect wireless transmissions	<ul style="list-style-type: none"> • Securely implement wireless technology. • Secure transmissions of cardholder data over wireless networks. • Provide instructions for secure use of wireless technology.
Facilitate secure network implementation	<ul style="list-style-type: none"> • Use only necessary and secure services, protocols, components, and dependent software and hardware, including those provided by third parties.

2.6.2 Contactless and Mobile Payments Council - Implementation Best Practices

The Contactless and Mobile Payments Council is one of several Smart Card Alliance technology and industry councils. There are several contactless payment system implementation guidelines discussed in one Smart Card Alliance Contactless and Mobile Payments Council White Paper [41]. Among the guidelines those which are applicable to this context are as follows:

- Contactless reader should be integrated with the POS system software so that the subsystems are synchronized.
- It is strongly recommended that the POS application software not require a sales clerk to take a “special” action (e.g., press a key) to start a contactless transaction. The contactless transaction should follow the same transaction flow as a magnetic stripe transaction. Once a card is swiped or tapped, the POS behavior should be the same.
- If a transaction is initiated with a contactless payment device, the POS system software should not ask for the last four digits of the card number. Some contactless payment devices (e.g., NFC enabled mobile phone, stickers, key fobs) will not have the card number embossed or printed on the device.
- The POS system should be implemented so that it knows if the contactless reader is disconnected or not functioning.
- If the contactless reader uses a USB port, a USB driver with dynamic port allocation must be used so that the POS system can find the reader if it is disconnected and reconnected.

2.6.3 Micro Payments

American Express, Discover Network, MasterCard, and Visa, in concert with their issuers, have implemented new programs and rules designed to encourage adoption of contactless payments. These programs are typically aimed at transaction scenarios where the transaction value is low (typically below \$25) and cash is the predominant form of payment. While the specific programs differ by payment brand and may be specific to any given issuer and/or merchant implementation, these programs may include the following features:

- The consumer signature is not required.

- A customer receipt is only required upon cardholder request.
- The merchant retains full chargeback protection for transactions that meet the program requirements.

Contactless payment can also be used for transactions greater than \$25. For example, pharmacy chains accept contactless payment and frequently have higher valued transactions. For these transactions, a signature is most likely required by the merchant to maintain full chargeback rights. Some merchants do extend the benefit of “no signature” for larger-sized transactions to their customers. However, these merchants typically would have made the business decision to take on the additional risk that may be associated with these transactions [1].

2.7 Related Products

2.7.1 EMV Technology

EMV (Europay MasterCard Visa) is a global and open standard set of specifications for smart card, credit card, debit payment cards and acceptance devices. The EMV specifications were developed to define a set of requirements to ensure interoperability between chips based payment cards and terminals. EMV chip cards contain embedded microprocessors that provide strong transaction security features and other application capabilities.

Today lot of countries are moving to EMV enabled, chip-base payment card and terminal systems. Reason for this move includes increased security and reduce fraud resulting from counterfeit, lost and stolen cards. EMV also provides interoperability with the global payments infrastructure. Consumers with EMV chip payment cards can use their card on any EMV-compatible payment terminal. EMV technology supports enhanced cardholder verification methods and, unlike magnetic stripe cards, EMV payment cards can also be used to secure online payment transaction.

2.7.1.1 Security Mechanisms used by EMV

EMV secures the payment transaction with enhanced functionality in following three areas:

- Card authentication – Card is authenticated during the payment transaction, protecting against counterfeit cards. Transactions require an authentic card validated either online by the issuer using a dynamic cryptogram or offline with the terminal using Static Data Authentication (SDA), Dynamic Data Authentication (DDA) or Combined DDA with application cryptogram generation (CDA). EMV transactions also create unique transaction data, so that any captured data cannot be used to execute new transactions.
- Cardholder verification – Authenticating the cardholder and protecting against lost and stolen cards. Cardholder verification ensures that the person attempting to make the transaction is the person to whom the card belongs. EMV supports four Cardholder Verification Methods (CVM): offline PIN, online PIN, signature, or no CVM. The issuer prioritizes CVMs based on the associated risk of the transaction (e.g., no CVM is used for unattended devices where transaction amounts are typically quite low).
- Transaction authorization – Using issuer-defined rules to authorize transactions. The transaction is authorized either online and offline. For an online authorization, transactions proceed as they do today in the U.S. with magnetic stripe cards. The transaction information is sent to the issuer, along with a transaction specific cryptogram, and the issuer either authorizes or declines the transaction. In an offline EMV transaction, the card and terminal communicate and use issuer-defined risk parameters that are set in the card to determine whether the transaction can be authorized. Offline transactions are used when terminals do not have online connectivity (e.g., at a ticket kiosk) or in countries where telecommunications costs are high.

2.7.1.2 Contactless Card Technology and EMV

Contactless EMV transactions use the ISO/IEC 14443 protocol for communication, with EMV Cooperation defining the EMV Contactless Communication Protocol Specification that is common for all payment brands. EMV has also published specifications for contactless POS readers that work with the payment brands' contactless applications. The EMV specifications provide a basis for contactless EMV payments, but do not specify all payment application functionality. Payment brands can implement contactless payment for EMV transactions to function in both offline and online transaction environments and to leverage the EMV cryptogram security function to validate the authenticity of the card and the transaction.

2.7.1.3 NFC Mobile Payment and EMV

EMV has been active in defining the architecture, specifications, requirements and type approval processes for supporting EMV mobile contactless payments. This effort has been critical in supporting the launch of NFC mobile contactless payment in Europe, which uses an EMV-based payments infrastructure [42].

2.7.1.4 Proposed Solution and EMV

Proposed solution plans to implement security features that are in line with EMV card's security level. Today 2.37 billion chip payment cards are being used in the world [42]. In 2013 United States also migrated to EMV chip technology. American Express, Discover, MasterCard and Visa have all announced their plans for moving to a chip-based payments infrastructure in the USA. Hence, Cake Labs' main market being the USA, the proposed NFC-based payment solution must be based on EMV chip enabled contact less cards.

2.7.2 Google Wallet

Google wallet helps both customers and merchants by allowing a faster, easier way to pay for their products. User begin by connecting all their credit cards, debit cards, and other money cards to the application. The application is equipped with a mechanism that, utilizes NFC technology to communicate with a nearby NFC tags to complete payments - or as Google tells it, "you'll only have to tap your smartphone on a store's credit card processor and you're good to go". At this time, the Google Wallet app is only available in the USA.

Security implementations of Google Wallet includes the following:

- Google took extra safety measures to ensure the protection of user credit cards, debit cards information through a four-digit Google Wallet PIN number that prevents anyone from accessing user accounts without authorization.
- Another safety step that Google has created is allowing consumers to remotely disable accounts online that prevents the phone from making any purchases but does not cancel the cards on user account.
- NFC system that is used to process transactions in partner with Google Wallet never receives any of user credit card information. The way it works is Google actually pays the store for user product and then processes user transaction within its own system. Therefore, person standing next to you in line will never be able to have access to your card numbers.
- Google Wallet also creates immunity to those running the application to what is known as "click-fraud" by blocking transactions that may occur from advertisements trying to charge user for accessing their web site because Google Wallet would simply ignore the request [43].

2.7.3 Apple Pay

People can use their iPhone, Apple Watch, or iPad to pay in a simple, secure, and private way using Apple pay. When today paying in stores happens in one natural motion and there is no need to open an app or even wake phone display because of innovative NFC antenna in iPhone 6. To pay, the your just needs to hold iPhone near the contactless reader with user finger on Touch ID. User does not even need to look at the screen to know whether payment information was successfully sent. A subtle vibration and beep let user know.

Security implementations of Apple Pay include the following:

- Every time user hand over their credit or debit card to pay, their card number and identity are visible. With Apple Pay, instead of using user actual credit and debit card numbers when user add their card, a unique Device Account Number is assigned, encrypted, and securely stored in the Secure Element, a dedicated chip in iPhone, iPad, and Apple Watch. These numbers are never stored on Apple servers. And when a user makes a purchase, the Device Account Number, along with a transaction-specific dynamic security code, is used to process user payment. So user's actual credit or debit card numbers are never shared by Apple with merchants or transmitted with payment.
- If user's iPhone or iPad is ever lost or stolen, user can use "Find My iPhone" to quickly put their device in Lost Mode to suspend Apple Pay, or user can wipe their device clean completely. User can also stop the ability to make payments from credit and debit cards on their iPhone, iPad, or Apple Watch with iCloud. Just log into iCloud.com and click on Settings [44].

3 Design

3.1 Overall System Architecture of the solution

Our solution mainly consists of four parts,

- Mobile Application
- SDK for the Reader
- NFC Reader
- NFC Card

The mobile application (as known as wallet application) works as a virtual wallet card. It is currently designed for Android smartphones. Any smartphone which has NFC capabilities can use the application. When the user brings the mobile device closer to the NFC reader, application communicates with the reader via the NFC communication module. Also it connects to a set of web services for user authentication purposes. The SDK handles the communication between POS API and the NFC reader, as well as the communication between the reader and the card / wallet application. The POS request handler essentially manages the data transfer to the POS terminal and data transfer back to the NFC reader.

Also there is a security module inside the SDK which handles all security related features such as mutual authentication between the reader and card/app and data encryption. Finally the POS API connects with pre-configured web services to store and read data in the backend. Figure 3.1 will illustrate the high level view of the system architecture.

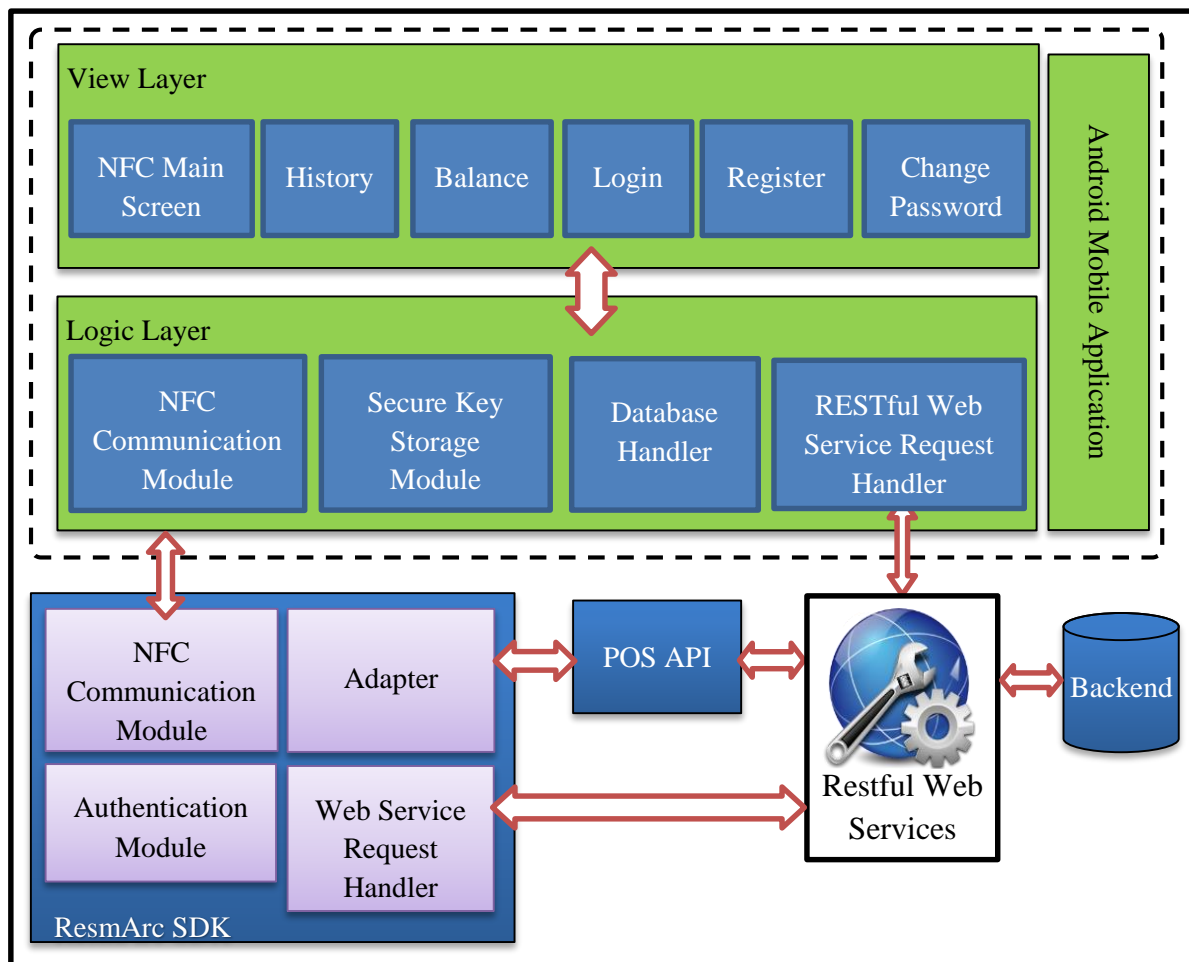


Figure 3.1 Overall System Architecture

3.2 Mobile Application

The mobile application basically consists of four major components apart from the user interface module. They are namely, NFC communication module, secure key store module, database handler and web service consumer module. NFC communication module is the heart of the application and is responsible for all data transfer with the NFC reader. NFC communication is based on the method called *Android Host-Based Card Emulation*.

Most of the devices runs on Android that has NFC capabilities already support NFC card emulation. In most cases, the card is emulated by a separate chip in the device, called a *secure element*. After Android 4.4 (*KitKat*), Google introduces an additional method of card emulation that does not involve a secure element, called *host-based card emulation (HCE)*. This allows any Android application to emulate a card and talk directly to the NFC reader. The data is routed to the host CPU on which Android applications are running directly, instead of routing the NFC protocol frames to a secure element [45]. Figure 3.2 illustrates how this works.

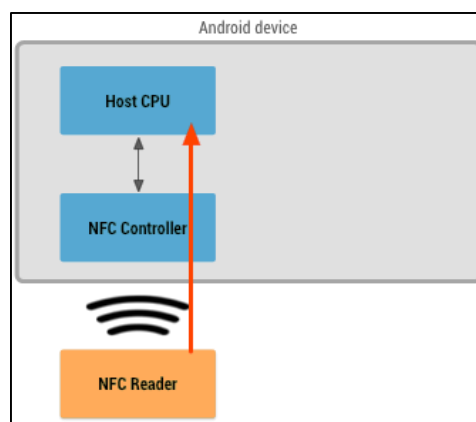


Figure 3.2 NFC card emulation without a secure element.

Host Card Emulation mechanism is primarily a service which runs in the background. And a supporting user interface is provided to the user to give feedback and notifications. When the user runs the application and taps the device to the NFC reader, the Android system will select the host card emulation service centered on a custom Application ID (AID), as defined in the ISO/IEC 7816-4 specification. Data are transferred over the network as application-level packets known as Application Packet Data Units (APDU). The communication protocol between the Android device and the NFC reader is half-duplex, meaning the NFC reader sends a command APDU and it will wait for the device to send a response APDU in return.

When the user installs the app and runs it for the first time, he needs to register in the system. After he successfully creates an account, he can login to the application. This user login happens only in the initial run, then a flag will be saved in the app so that the user does not need to give credentials every time he uses the app. To prevent unauthorized uses of the app, it is secured with a four-digit PIN number. User will be asked to specify the PIN number after the first login. After the PIN number is saved securely, user need the PIN to use the application.

The secure key store modules is responsible for encrypting and securely storing the PIN number in the device. This is based on the Android Key store system API. It gives the functionality of storing cryptographic keys in a container to make it more difficult to extract from the device. Once keys are in the key store, they can be used for cryptographic operations with the key material remaining non-exportable. Moreover, it offers facilities to restrict when and how keys can be used, such as requiring

user authentication for key use or restricting keys to be used only in certain cryptographic modes. Android Key store mitigates unauthorized use of key material outside of the Android device by preventing extraction of the key material from application processes and from the Android device as a whole. Secondly, Android Key Store mitigates unauthorized use of key material on the Android device by making apps specify authorized uses of their keys and then enforcing these restrictions outside of the apps' processes [46]. Initially, PIN number will be encrypted using RSA public and private key algorithm and then stored securely. When the user enters the PIN again, stored PIN will be decrypted using securely stored keys and compared with the entered PIN. If it is a match, user will be allowed to enter to the application.

When the user creates an account in the backend, a unique *token* will be created and passed to the device over SSL secure connection. This token is essential for all NFC based transactions. This token will not be communicated in plain text or encrypted as it is during transactions. Instead, we use a *challenge response protocol* to transfer the token to the reader. When the user brings the device closer to the reader, reader will first send an APDU containing an activate command for the HCE service in the device together with AID and a random number. Then the application will compute the hash value of the concatenation of received random number and token. Then the app will send the device MAC address together with the hash value and a success message. Then the reader will get the token from the backend using the device's MAC address and computes the hash value of sent random number and token using the same hash function. Then the reader compare two hash values. If they did not match, reader will send a fail message to the device, otherwise the transaction will continue with the backend and reader will send updates back to the device. Transaction could be a top-up, a payment or an update of balance. After the transaction, its type, amount and the balance will be sent to the device and finally the device will send a success message to the reader. The application will update the balance and show a notification to the user.

The application will store the most recent balance and a history of payments inside a SQLite database, so that the user can view the balance and his payment history anytime. Since important data are stored inside the device, if the user clears the data for some reason unintentionally, those data will be lost. To prevent that, the data clear feature has been disabled for the app and if someone tries to clear data of the app through application manager, an Activity will be launched with a notification saying that the data cannot be cleared.

The web service consumer module of the app is responsible for all the direct communication with the backend. These communications are done mainly through a set of RESTful web services. These web services includes, user registration, login, password change and password reset. All the service requests are converted to JSON objects using a JSON Parser and then fed to the web services. We used an event bus to manage communication among the modules of the applications. Each network response will be posted to the event bus and corresponding registered listener will get the response without errors.

3.3 ResmArc SDK

ResmArc SDK is developed in python in top of *pyscard* library [47], a wrapper library for open source implementation of *pcsc-lite* library [48]. Its main modules are Communication Module, Adaptor, Authentication Module, and Web Service Request Module.

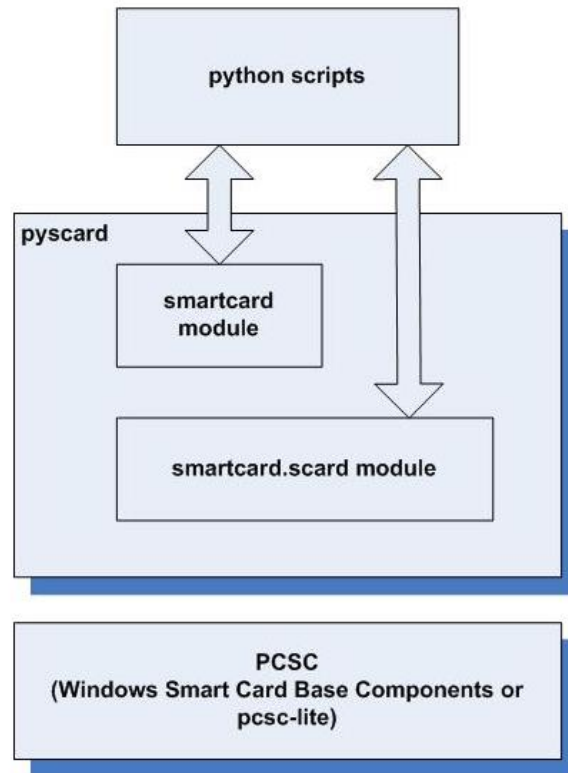


Figure 3.3 Pyscard Architecture [47].

- Easily adaptable to any reader compliant with PC/SC specifications
- Can be integrated conveniently with any POS terminal
- Supports mutual authentication through challenge response mechanism
- Support HCE (Host Card Emulation) based applications
- Ability to detect card type as process accordingly

3.3.1 Communication Module

Communication module is the main module and its functionalities include Card Monitoring and Processing, Identifying card types and Doing communications related to Authentication, Read Binary and Write Binary.

3.3.2 Card monitoring and processing

This done using automatic PICC polling. Whenever the reader is connected to the PC, the PICC polling function will start the PICC scanning to determine if a PICC is placed on/removed from the built-in antenna. In the SDK an Observer Observable model is used for this monitoring and processing activity. *CardMonitor* will notify its observers when a card is detected and *CardObserver* can do processing accordingly.

3.3.3 Communications / APDU

Communications are done through APDUs according to ISO 7816-4 Specification. APDU (Application Protocol Data Unit) format is as following. APDU commands are always referred to as command-response pair.

Code	Name	Length	Description
CLA	Class	1	Class of instruction
INS	Instruction	1	Instruction code
P1	Parameter 1	1	Instruction parameter 1
P2	Parameter 2	1	Instruction parameter 2
Lc field	Length	variable 1 or 3	Number of bytes present in the data field of the command
Data field	Data	variable=Lc	String of bytes sent in the data field of the command
Le field	Length	variable 1 or 3	Maximum number of bytes expected in the data field of the response to the command

Figure 3.4 APDU Command Format [49].

Code	Name	Length	Description
Data field	Data	variable=Lr	String of bytes received in the data field of the response
SW1	Status byte 1	1	Command processing status
SW2	Status byte 2	1	Command processing qualifier

Figure 3.5 APDU Response Format [49].

Load Authentication Keys APDU Format (11 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Load Authentication Keys	FFh	82h	Key Structure	Key Number	06h	Key (6 bytes)

Authenticate Data Bytes (5 bytes)

Byte1	Byte 2	Byte 3	Byte 4	Byte 5
Version 01h	00h	Block Number	Key Type	Key Number

Read Binary APDU Format (5 bytes)

Command	Class	INS	P1	P2	Le
Read Binary Blocks	FFh	B0h	00h	Block Number	Number of Bytes to Read

Figure 3.6 APDU Commands with Wallet Card

Update Binary APDU Format (Multiple of 16 + 5 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Update Binary Blocks	FFh	D6h	00h	Block Number	Number of bytes to update	Block Data (Multiple of 16 bytes)

3.3.3.1 APDU Commands with Application

Android application works in host card emulation mode and therefore communicating with the application is a bit different from communication with the card. When communicating with the application the reader first has to issue application select command proving the proper AID (Application ID) for the application. Then only we can communicate with the application.

CLA	INS	P1	P2	Lc	Data	Le
00h	A4h	04h	00h	07h	D2760000850101h	00h

Figure 3.7 Application Select Command

3.3.4 Authentication Module

Authentication are done in two ways. With the mobile application we use challenge response authentication and with wallet card we use RSA asymmetric key encryption. For CRA we use the *pycrypto* library for python which include and implementation of SHA-512 which is a variant of SHA-2.

3.3.5 Web Service Request Module

Web service request module is used to communicate with the backend databases. JSON and requests python libraries are used for this implementation.

3.4 NFC Card and Reader selection

3.4.1 NFC Reader selection

After a lot of research about existing NFC readers, we selected ACR1252U USB NFC Reader, which is an NFC Forum-certified PC-linked reader, developed based on 13.56 MHz contactless technology. It has a SAM (Secure Access Module) slot which can be used together with a SAM card for key diversification and mutual authentication, providing high-level security in contactless transactions. Post-deployment firmware upgrade is also supported, eliminating the need for additional hardware modification.

ACR1252U is capable of the three modes of NFC, namely: card reader/writer, card emulation and peer-to-peer communication. It supports ISO 14443 Type A and B cards, MIFARE, FeliCa, and ISO 18092-compliant NFC tags. It also supports other NFC devices with an access speed of up to 424 Kbps and proximity operating distance of up to 50mm (depending on tag type used). Compliant with both CCID (chip card interface device) and PC/SC (Personal Computer/Smart Card), this plug-and-play USB NFC device allows interoperability with different devices and applications. The ACR1252U comes with an optional stand to hold the smart card reader at an optimal angle, so that users can tap contactless cards or NFC-enabled devices onto the ACR1252U with ease [50].



Figure 3.8 ACR1252U USB NFC Reader

ACR1252 Features and Specifications

- USB 2.0 Full Speed Interface
- CCID Compliance
- USB Firmware Upgradeability
- Smart Card Reader:
 - Read/write speed up to 424 kbps
 - Built-in antenna for contactless tag access, with card reading distance of up to 50 mm (depending on tag type)
 - Supports ISO 14443 Type A and B cards, MIFARE, FeliCa, and all 4 types of NFC (ISO/IEC 18092) tags
 - Supports MIFARE 7-byte UID, MIFARE Plus and MIFARE DESfire
 - Built-in anti-collision feature (only 1 tag is accessed at any time)
 - One ISO 7816-compliant SAM slot (Class A)
- NFC Support:
 - NFC Reader/Writer Mode
 - Peer-to-Peer Mode
 - Card Emulation Mode
- Application Programming Interface:
 - Supports PC/SC
 - Supports CT-API (through wrapper on top of PC/SC)
- Peripherals:
 - User-controllable bi-color LED
 - User-controllable buzzer
 - SAM Slot
- Operating System support

- Win 2000, Win XP, Win Vista, Win 7, Win 8, Win 8.1, Win Server 2003, Win XP x64, Win Vista x64, Win 7 x64, Win 8 x64, Win 8.1 x64
- Linux
- Mac
- Dimensions: 98.0 mm (L) x 65.0 mm (W) x 12.8 mm (H)
- Weight: 81.0 g

3.4.2 NFC Card selection

We have selected MIFARE DESFire EV1 as the most suitable NFC enabled card for the solution since it is based on open global standards for both RF interface and cryptographic methods. It features an on-chip backup management system and mutual three pass authentication, allowing it to hold up to 28 different applications and 32 files per application. MIFARE DESFire EV1 Fully compliant with ISO/IEC 14443A (part 1-4) using optional ISO/IEC 7816-4 commands, the selectable cryptographic methods include 2KTDDES, 3KTDDES and AES128 [51].



Figure 3.9 MIFARE DESFire EV1 NFC Card

Also we have considered about MIFARE Classic NFC cards due to its low cost and easy to program, configure and use. MIFARE Classic cards are fully compliant with ISO/IEC 14443 Type A, it is available with 1 KB and 4 KB memory, and 7 byte or 4 byte identifiers. MIFARE Classic is a product family of contactless smart card ICs used in public transport, access management, loyalty cards and many more [49].



Figure 3.10 MIFARE Classic NFC Card

3.4.2.1 MIFARE CLASSIC

Memory Organization

Mifare classic 1k has fixed memory structure. It has been divided into 16 sectors with each sector having 4 blocks and each block with size of 16 bytes. Forth block in each sector is called sector trailer and it contain the keys for each sector.

Sector	Block	Byte Number within a Block																Description
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
15	3	Key A						Access Bits				Key B						Sector Trailer 15
	2																	Data
	1																	Data
	0																	Data
14	3	Key A						Access Bits				Key B						Sector Trailer 14
	2																	Data
	1																	Data
	0																	Data
:	:																	
:	:																	
:	:																	
1	3	Key A						Access Bits				Key B						Sector Trailer 1
	2																	Data
	1																	Data
	0																	Data
0	3	Key A						Access Bits				Key B						Sector Trailer 0
	2																	Data
	1																	Data
	0	Manufacturer Data																Manufacturer Block

001aan011

Accessing Mifare Classic Blocks

To access Mifare classic blocks first the reader must call the authenticate command proving the security key of the sector of the block or blocks it wants to get access to. Then the read can go on to read binary blocks if the authentication is successful.

Changing Security Keys

To change security key for a sector the reader must first authenticate itself using the current key and then do write command to change the for the sector trailer block.

3.5 NFC Reader Library and Drivers

ACR1252 NFC Reader comes with a set of drivers and libraries for each operating system it supports. From those drivers, the Linux distribution is free and open source to use. Therefore we chose the Linux distribution of drivers and libraries to work with. Drivers supports ACS CCID smart card readers. This

library provides a PC/SC IFD handler implementation and communicates with the readers through the PC/SC Lite resource manager. [50]

4 Implementation

4.1 Languages, Tools and Technologies

We chose Linux Ubuntu 14.04 as our main development operating system since the drivers and libraries that we had for the NFC reader are supported for Linux. And for the SDK development, we used Python language because its simplicity and less coding requirement. And we were able to integrate NFC reader libraries easily with Python. We used, *PyCharm* IDE which is *Jet Brain*'s official IDE for Python.

As for the Android mobile application, the code is in Java language and we have used several open source APIs to facilitate various functionalities,

- Android HCE (Host Card Emulation)
- Android Key Store System APIs, Android
- Crypto APIs
- Apache REST APIs
- JSON APIs

Also we have used the Otto event bus framework in the app to manage network responses without any collisions. Also we used SQLite database to store payment history and other data inside the app.

Moreover, we have used the REST for communication with the web services, and GIT for version control of our project.

4.2 Implementation of the Android mobile application

As discussed in the design part, the Android application is implemented according to the part of an architecture diagram illustrated in figure 3.1. The application can be deployed to the Google Store and users can download it from the store. Once they download the application, they need to register for an account in the backend. The app uses REST API for connect with web services to facilitate those functionalities. Users can register, login, logout, change password and reset forgotten password. There are separate REST services for all these functionalities. All the web services which connect to the app do the data transfer via SSL secure tunnel to ensure security.

App does not save user credentials inside the device, instead it uses a PIN number to secure the app from unauthorized access. Therefore the user needs to specify a four digit PIN number at the initial login. When the user register for an account, the mobile device's unique MAC address will be sent to the server and that address will be used to identify the device uniquely. So one device can be used for only one account in the backend.

Moreover, a unique *token* will be created for each device and will be sent to the device over the SSL connection after initial login. This token is essential in every NFC based transaction with the reader. Therefore the token and the PIN number are encrypted using Android Key store mechanism and saved in app's shared preferences. The Android Key store system provides cryptographic keys for encryption purposes and store them in a container to make it more difficult to extract from the device. Therefore even somebody try and get the encrypted token and PIN number from the device, he would not be able to extract the key to decrypt them. Here we used the RSA/ECB algorithm with PKCS1Padding for key generation.

Android key store API provides the standard *KeyStore* and *KeyPairGenerator* or *KeyGenerator* classes introduced in Android 4.3 (API level 18). Using these classes, new private keys and secret keys can be generated easily. For signing data, the *KeyStore.Entry* from the key store can be used with the *Signature APIs*, such as *sign()* and for fetching data, *verify(byte[])* method can be used.

All the NFC related communication of the app will be based on the Android Host-based Card Emulation. Communication mechanism is implemented as described in the section 3.2. NFC reader initiates the communication by sending an activation signal to the application. A HCE service which is implemented in the app will be selected and triggered. There is a background service component that handles the NFC transactions inside the app. User needs to tap the NFC button on the app before bringing the device closer to the reader, if the NFC functionality is turned off in the device, user will be notified to turn on the functionality. If the mobile device is rooted, the NFC functionality will be turned on programmatically without the user's effort.

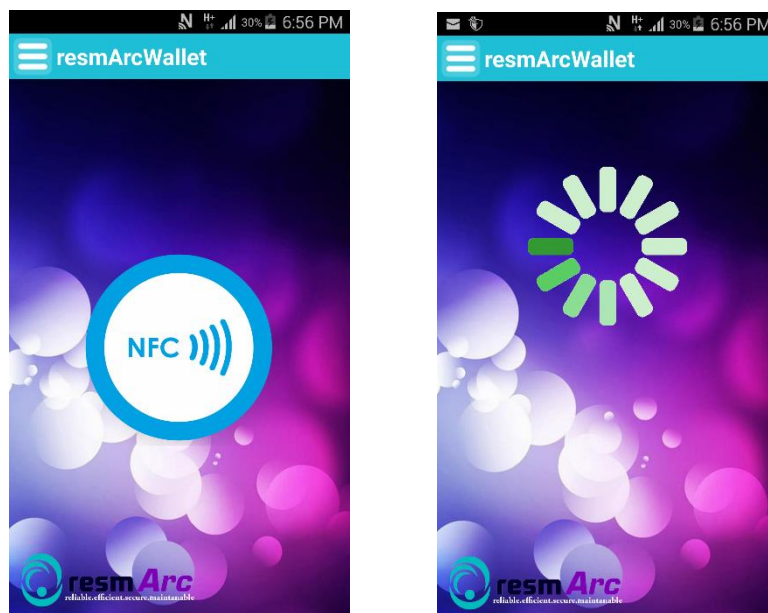


Figure 4.1 Wallet app main screen

Each transaction process between the reader and the device is secured via challenge response protocol. Therefore the token will not be communicated in plain text or encrypted as it is. Always the token will be encrypted together with a random number before transmitting over the network.

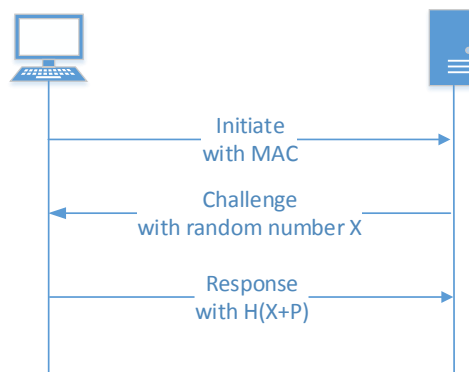


Figure 4.2 Challenge Response Authentication Protocol

As illustrated in the figure 4.2, the challenge-response authentication (CRA) is a method for proving the identity over an insecure medium without giving any information out to eavesdroppers that may enable them to identify.

CRA fundamentally depends on the existence of "one-way hashes". A one-way hash is a function that takes an input, and returns a "hash value". Finding the input of the function from the hash is "computationally infeasible", which, in the case of a decent hash function, means you'll be crunching numbers until the next Stone Age.

We have used SHA-512 algorithm as the one way hash function for this purpose. For a moment, if we think the hash function as $h()$, when the device connects to the reader, the reader makes up a random value, X . The reader sends X to the device. The device sends the server $h(X+T)$, where T is the token and $+$ represents concatenation. The reader computes $h(X+T)$ as well, and checks to see if the data from the app matches the computed value. If so, the app must know the token.

After a transaction completed successfully, the user will be notified with the current balance. And all the transaction history will be saved to a SQLite database inside the application. User can view the history of the each transaction together with the type of transaction (Pay, Update, and Top-up), amount and Date/time. We have disabled the "data clear feature" of the app because if the user try to clear data, all the essential data including the pin number, token, login status and transaction history will be removed. Therefore the user cannot clear application data and he will be notified the consequences if he try to clear app data. But when the user logs out from the app, all history details and PIN number will be removed and he needs to specify a new PIN number when login.

This application requires internet connection only in the initial user registration and login process. Hence the application can work offline unless the user wants to log out, re-login or change password. When handling web service responses, the Otto event bus framework has been used to avoid collisions among the network responses. Each network response will be posted to the event bus and corresponding registered listener will get the response without errors as illustrated in the figure 4.3.

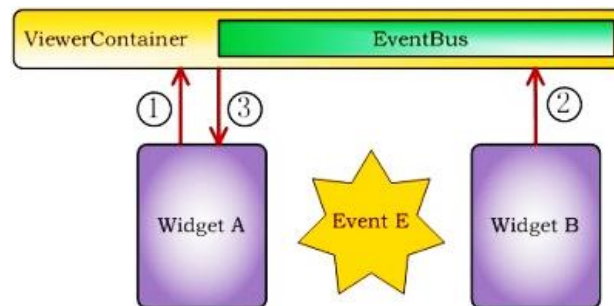


Figure 4.3 Otto Event bus

4.3 Implementation of the ResmArc SDK

ResmArc SDK was implemented as discussed in the design section. We used a python socket client at the SDK side and java socket server at the POS side for the integration process.

5 Results

5.1 Testing of the modules

The two major components of the system are mobile application and the SDK, these two components have been tested separately at various stages in the development process. In the mobile app, unit tests were carried out to ensure the code quality and standards. And also in the SDK, several tests have been carried out to reflect various use cases such as NFC reader - card communication and NFC reader – mobile device communication. When the integration of SDK with the POS API, several integration tests were done to ensure the proper communication between the two parties.

5.2 Results

Android Application

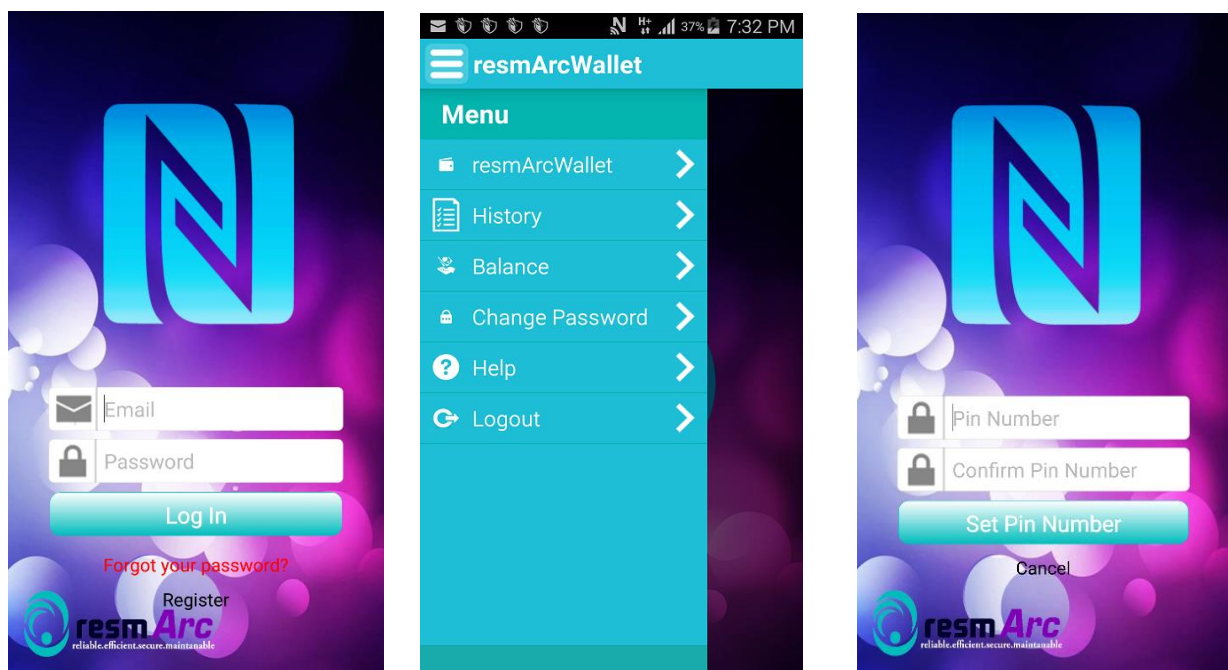
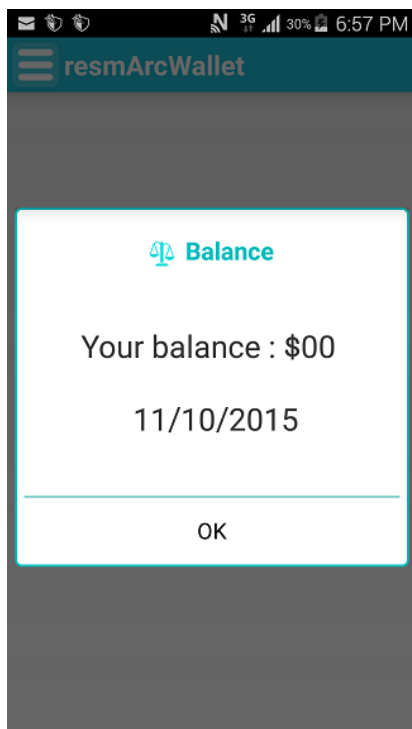


Figure 5.1 Mobile app screenshots, Login screen, Drawer and Pin enter screen



Figure 5.2 Main NFC Screen



resmArcWallet		
	Paid	90.00
	3/11/2015	
	Paid	80.00
	4/11/2015	
	Updated	30.00
	5/11/2015	
	Paid	20.00
	6/11/2015	
	Updated	10.00
	7/11/2015	
	Paid	5.00
	8/11/2015	
	Top-Up	200.00
	9/11/2015	
	Paid	50.00
	10/11/2015	
	Updated	155.00
	11/11/2015	

Figure 5.3 Balance and History

6 Conclusion

6.1 Problems and challenges faced

- Android application is only supported in devices with Android 4.4 and upwards
- SDK does not yet have the functionality to store and retrieve keys in/from Secure Access Module
- Integrating the reader with the POS API was challenging task
- We did not have backend support, therefore we had to write our own web services for testing purposes

6.2 Future work

- Need to upgrade the cards from Mifare Classic to Mifare Desfire
- SDK needs to be customized so that it is compatible with any POS terminal
- Mobile application needs to be extended to iOS and Windows mobile platforms
- Extend the system to the Sri Lankan market as well

References

- [1] Smart Card Alliance. (2007, Feb.). *Contactless Payments: Frequently Asked Questions* [Online]. Available: <http://www.smartcardalliance.org/publications-contactless-payments-faq/>.
- [2] Smart Card Alliance, "Contactless Payment and the Retail Point of Sale: Applications, Technologies and Transaction Models," New Jersey., Mar. 2003.
- [3] Smart Card Alliance, "The What, Who and Why of Contactless," New Jersey., Nov. 2006.
- [4] Cake Labs. (2015). *Cake POS* [Online]. Available: <http://www.cakelabs.lk/product-suite/cake-pos/>
- [5] V. Coskun, K. Ok, and B. Ozdenizci, "Near Field Communication (NFC): From Theory to Practice," 1st ed. New York: Wiley, 2012.
- [6] NFC Forum, "NFC Analog Technical Specification Version 1.0," July 2012.
- [7] Sony. (2015). *Felica: Contactless IC card technology* [Online]. Available: <http://www.sony.net/Products/felica/NFC/index.html>.
- [8] E. Rukzio, K. Leichtenstern, V. Callaghan, P. Holleis, A. Schmidt, and J. Chin, "An Experimental Comparison of Physical Mobile Interaction Techniques: Touching, Pointing and Scanning," in *UbiComp 2006: Ubiquitous Computing*, California, USA, 2006, pp. 87-104.
- [9] B. Ozdenizci, M. Aydin, V. Coskun, and K. Ok, "NFC Research Framework: A Literature Review and Future Research Directions," in *14th IBIMA International Business Information Management Conf*, Istanbul, Turkey, 2010, pp. 2672-2685.
- [10] V. Coskun, K. Ok and B. Ordenizci, "Current Benefits and Future Directions of NFC Services," in *Proc. International Conference on Education and Management Technology (ICEMT)*, Cairo Egypt, 2010, pp. 334–338.
- [11] K. Nohl and D. Evans. (2008). *Reverse-Engineering a Cryptographic RFID Tag* [Online]. Available: <http://www.cs.virginia.edu/~evans/pubs/usenix08/mifare.html>.
- [12] P. Sorrells, "Passive RFID Basics," Microchip Technology Inc., 1998.
- [13] K. Nohl. (2007). *Mifare: little security, despite obscurity* [Online]. Available: https://www.blackhat.com/presentations/bh-usa-08/Nohl/BH_US_08_Nohl_Mifare.pdf.
- [14] AllBits Knowledge Hub. *NFC* [Online]. Available: <http://www.allbits.eu/internet-of-things/wireless-connectivity/nfc/>.
- [15] NFC Forum, "NFC Data Exchange Format Technical Specification 1.0," July. 24, 2006.
- [16] Sony Coperation. (2015). *NFC Forum Specifications* [Online]. Available: <http://www.sony.net/Products/felica/NFC/forum.html>.
- [17] M. Sippel. (2010, Aug. 10). *Choosing RFID For Industrial Applications* [Online]. Available: <https://rfidspec.wordpress.com/2010/08/02/choosing-rfid-for-industrial-applications-part-3/>.
- [18] Libnfc. (2015, April 19). *Libnfc* [Online]. Available: <http://nfc-tools.org/index.php?title=Libnfc>.
- [19] Inside Secure. (2015, July 31). *OpenNFC Developpe Site Home* [Online]. Available: <http://open-nfc.org/wp/>.
- [20] Inside Secure. (2015, July 31). *Open NFC Developer Site* [Online]. Available: <http://open-nfc.org/wp/>.

- [21] Inside Secure. (2015, July 31). *General principles* [Online]. Available: <http://open-nfc.org/wp/home/documentation/general-principles/>.
- [22] Inside Secure. (2015, July 31). *Features* [Online]. Available: <http://open-nfc.org/wp/home/features/>.
- [23] NFC Forum. (2015). *NFC Forum Technical Specifications* [Online]. Available: http://members.nfc-forum.org/specs/spec_list/.
- [24] Inside Secure. (2015, July 31). *Security Stack* [Online]. Available: <http://open-nfc.org/wp/home/documentation/security-stack/>.
- [25] R. Moscatiello, “Basic Concepts in RFID Technology,” 2007.
- [26] S. Clark. (2010, June 16). *Innovision launches 512 byte NFC tag* [Online]. Available: <http://www.nfcworld.com/2010/06/16/33937/innovision-launches-512-byte-nfc-tag/>.
- [27] NXP Semiconductors. (2015). *MIFARE Ultralight* [Online]. Available: http://www.nxp.com/products/identification_and_security/smart_card_ics/mifare_smart_card_ics/mifare_ultralight/.
- [28] SONY. (2015). *FeliCa* [Online]. Available: <http://www.sony.net/Products/felica/business/products/RC-S966.html>.
- [29] NXP Semiconductors. (2015). *MIFARE DESFire* [Online]. Available: http://www.nxp.com/products/identification_and_security/smart_card_ics/mifare_smart_card_ics/mifare_desfire/.
- [30] NXP Semiconductors. (2015). *SmartMX contact interface controllers* [Online]. Available: http://www.nxp.com/products/identification_and_security/smart_card_ics/smartmx_contact_interface_controllers/.
- [31] NFC Forum. (2015). *Tag Type Technical Specifications* [Online]. Available: <http://nfc-forum.org/our-work/specifications-and-application-documents/specifications/tag-type-technical-specifications/>.
- [32] SpringCard. (2004). *Introduction to PC/SC Development* [Online]. Available: <http://www.springcard.com/en/solutions/pcsc-sdk.html>.
- [33] Smart Card Standards. (2010). *Smart Card Standards* [Online]. Available: <http://www.smartcardbasics.com/smart-card-standards.html#pcsc>.
- [34] Digital Logic Ltd. (2015). *NFC USB reader DL533N OEM* [Online]. Available: <http://www.d-logic.net/nfc-rfid-reader-sdk/products/nfc-usb-card-size-reader-dl533n-oem>.
- [35] OpenPCD. *OpenPCD 2 RFID Reader for 13.56MHz* [Online]. Available: http://www.openpcd.org/OpenPCD_2_RFID_Reader_for_13.56MHz.
- [36] E. Haselsteiner and K. Breitfuß, “Security in Near Field Communication (NFC),” Gratkorn, Australia, 2006.
- [37] L. Church and M. Moloney, “State of the Art for Near Field Communication: security and privacy within the field,” Ireland, 2012.
- [38] Infosec Institute. (2013, June 18). *Near Field Communication (NFC) Technology, Vulnerabilities and Principal Attack Schema* [Online]. Available: <http://resources.infosecinstitute.com/near-field-communication-nfc-technology-vulnerabilities-and-principal-attack-schema/>.

- [39] A. Pal. (2012, Nov. 01). *Near field communication – the security risks* [Online]. Available: http://www.cso.com.au/article/440741/near_field_communication_security_risks_/.
- [40] Payment Card Industry Security Standards Council, “Payment Application Data Security Standard,” Oct. 2014.
- [41] Smart Card Alliance Contactless and Mobile Payments Council, “Issuer and Merchant Best Practices: Promoting Contactless Payments Usage and Acceptance”, New Jersey, Sep. 2009.
- [42] Smart Card Alians. (2015, April). *EMV FAQ* [Online]. Available: <http://www.emv-connection.com/emv-faq/>.
- [43] Google. *Google Wallet* [Online]. Available: <https://www.google.lk/wallet/>.
- [44] Apple. *Apple Pay* [Online]. Available: <https://www.apple.com/apple-pay/>.
- [45] Google. Android HCE [Online]. Available: <http://developer.android.com/guide/topics/connectivity/nfc/hce.html>.
- [46] Google. Android KeyStore [Online]. Available: <http://developer.android.com/training/articles/keystore.html>.
- [47] Sourceforge. (2014). pycard - python for smart cards[Online]. Available: <http://pycard.sourceforge.net/>.
- [48] MUSCLE. (2015, December). PCSC-Lite [Online]. Available: <https://pcsc-lite.alioth.debian.org/>.
- [49] CardWerk. Interindustry Commands for Interchange [Online]. Available: http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_5_basic_organizations.aspx
- [50] ACS. ACR1252U USB NFC Reader III [Online]. Available: <http://www.acs.com.hk/en/products/342/acr1252u-usb-nfc-reader-iii-nfc-forum-certified-reader/>.
- [51] NXP. MIFARE DESFire [Online]. Available: http://www.nxp.com/products/identification-and-security/smart-card-ics/mifare-smart-card-ics/mifare-desfire:MC_53450.
- [52] NXP. MIFARE Classic [Online]. Available: http://www.nxp.com/products/identification-and-security/smart-card-ics/mifare-smart-card-ics/mifare-classic:MC_41863.
- [53] CCID.[Online]. Available: <http://pcsc-lite.alioth.debian.org/ccid.html>.