การพัฒนาระบบควบคุมการเข้าถึงโดยใช้การสื่อสารแบบสนามใกล้ (เอ็นเอฟซี)

นายจัน ดาราลี ชิน

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2558

DEVELOPMENT OF AN ACCESS CONTROL SYSTEM USING NEAR-FIELD COMMUNICATION (NFC)

Mr. Chan Daraly Chin

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2015

Thesis Title	DEVELOPMENT OF AN ACCESS CONTROL SYSTEM USING NEAR-FIELD COMMUNICATION (NFC)
By	Mr. Chan Daraly Chin
Field of Study	Electrical Engineering
Thesis Advisor	Associate Professor Watit Benjapolakul, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of the Faculty of Engineering

(Associate Professor Supot Teachavorasinskun, D.Eng.)

THESIS COMMITTEE

.....Chairman

(Associate Professor Lunchakorn Wuttisittikulkij, Ph.D.)

.....Thesis Advisor

(Associate Professor Watit Benjapolakul, Ph.D.)

.....Examiner

(Assistant Professor Chaiyachet Saivichit, Ph.D.)

.....External Examiner

(Chaiyaporn Khemapatapan, Ph.D.)

จัน ดาราลี ชิน : การพัฒนาระบบควบคุมการเข้าถึงโดยใช้การสื่อสารแบบสนามใกล้ (เอ็นเอฟซี) (DEVELOPMENT OF AN ACCESS CONTROL SYSTEM USING NEAR-FIELD COMMUNICATION (NFC)) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร.วาทิต เบญจพลกุล, 57 หน้า.

Host-based Emulation Card (HCE) ถือเป็นเหตุผลหลักสำหรับการพัฒนาแอปพลิเคชันการสื่อสารแบบสนามใกล้ (เอ็นเอฟซี) ของระบบปฏิบัติการแอนดรอยด์ เนื่องจากสามารถใช้แทน card emulation โดยการใช้ Secure Element (SE) เมื่อนักพัฒนาจำเป็นต้องเจรจาต่อรองกับผู้ผลิต SE ซึ่งช่วยเปิดโอกาสให้กับสถาบันการศึกษาในการพัฒนาในด้านดังกล่าวเพื่อนำมาใช้ประโยชน์ ทั้งนี้ HCE สามารถรองรับมาตรฐานโพรโทคอล ISO / IEC 7816-4 ที่ใช้งานกันอย่างแพร่หลายในตลาด อย่างไรก็ตาม นักพัฒนายังคงยากที่จะพัฒนาแอปพลิเคชัน บนมาตรฐานนี้เนื่องจากซอฟต์แวร์บุคคลที่สามมักจะจำกัดให้ใช้งานได้บนผลิตภัณฑ์ของตนเองเท่านั้น นอกจากนี้ ไลบรารีเอ็นเอฟซีในปัจจุบัน เช่น NFC shield ยังรองรับเฉพาะโพรโทคอล ISO 14443-4 เท่านั้น

แอปพลิเคชันแอนดรอยด์ที่รองรับเอ็นเอฟซี ถูกพัฒนาขึ้นเพื่อปูทางให้กับนักพัฒนา โดยจำลองเป็นเอ็นเอฟซีสมาร์ตการ์ด ซึ่งใช้ HCE แทนรหัสผ่าน 4 หลักของระบบการควบคุมการเข้าถึง โดยรหัสผ่านทั้งหมดจะถูกปรับเปลี่ยน เข้ารหัส และบันทึกไว้ในหน่วยความจำภายในของโทรศัพท์เสมือนเป็นระบบที่แยกจากกัน (isolation system) ได้ตลอดเวลา ในการทดสอบแอปพลิเคชันดังกล่าวขณะพัฒนา ไลบรารีเอ็นเอฟซีแบบโอเพนซอร์สจะถูกขยายออกบนไลบรารีมาตรฐาน ISO ที่มีอยู่เดิม เพื่อช่วยในการติดตามทุกขั้นตอนของแอปพลิเคชัน HCE โดย ACR122U NFC reader จะถูกใช้เพื่ออ่าน HCE ในการทดสอบนี้ ซึ่งเป็นลักษณะเดียวกับวิธีที่ ACR122U ทำงานบนแพลตฟอร์มวินโดวส์ผ่าน USB ซึ่งยังเป็นไปไม่ได้ในปัจจุบัน สุดท้าย การพัฒนาไลบรารีเอ็นเอฟซีสำหรับ NFC shield จะอ้างอิงตามมาตรฐาน ISO / IEC 7816-4 โดยใช้ข้อกำหนดตาม NFC Forum Type 4 เพื่อสนับสนุนการใช้งานบอร์ด Arduino Uno เป็น NFC reader จากนั้น ไลบรารีที่ถูกสร้างขึ้นสำหรับ NFC shield จะถูกทดสอบด้วยแอปพลิเคชันแอนดรอยด์ โดยในงานวิจัยนี้จะประกอบด้วยสามการทดลอง การทดลองแรกของแอปพลิเคชันแอนดรอยด์ จะเป็นการสำรวจข้อมูลที่จัดเก็บอยู่ในระบบแยกของแอนดรอยด์ (Android's isolation system) การทดลองที่สองจะเป็นการตรวจสอบว่าแอปพลิเคชันสามารถทำงานได้ถูกต้องบนโพรโทคอลมาตรฐานด้วยไลบรารีโอเพนซอร์สแบบขยาย (extended open source library) โดยใช้ ACR122U และการทดลองสุดท้ายคือการตรวจสอบไลบรารีที่สร้างขึ้นของ NFC shield สำหรับ Arduino กับแอปพลิเคชัน HCE โดยสองการทดลองสุดท้ายแสดงให้เห็นว่าข้อมูลเอาต์พุตที่ได้เหมือนกับรหัสผ่านของแอปพลิเคชันแอนดรอยด์

ภาควิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2558

5670541621 : MAJOR ELECTRICAL ENGINEERING
KEYWORDS: HOST-BASED CARD EMULATION / NEAR-FIELD COMMUNICATION (NFC) / ACCESS CONTROL SYSTEM / ANDROID / ARDUINO

CHAN DARALY CHIN: DEVELOPMENT OF AN ACCESS CONTROL SYSTEM USING NEAR-FIELD COMMUNICATION (NFC). ADVISOR: ASSOC. PROF. WATIT BENJAPOLAKUL, Ph.D., 57 pp.

Host-based Card Emulation (HCE) is main reason for developing NFC application of Android OS. It can replace the card emulation by using Secure Element (SE) when developers need to negotiate with SE manufacturer. This is more open for academia to develop in this field and take its benefits. HCE can support ISO/IEC 7816-4 protocol standard which is commonly used in the market. However, developers are still difficult to develop their application over this standard since third-party software is limited only to their products. Moreover, NFC library such as NFC shield currently supports only ISO 14443-4 protocol.

To pave the ways for developers, NFC-enable Android application is developed to emulate as NFC smartcard using HCE instead of 4 digits passcode of access control system. All passcodes are modified, encrypted and saved in internal phone storage as isolation system anytime. To test this application while developing, open source NFC library is extended over existing ISO standard's library to help to track all procedures of this HCE application. ACR122U NFC reader is used to read HCE in this test as well as how ACR122U works on Windows platform over USB which it is not currently feasible. Lastly, developing the NFC library for NFC shield is based upon the ISO/IEC 7816-4 using NFC Forum Type 4 Tag Specification to support Arduino Uno board as NFC reader, then this built library for NFC shield is tested with Android application. There are three experiments in this work. The first experiment of Android application is to discovery data which storing in Android's isolation system. The second experiment is to validate whether the application works correctly over protocol standard with extended open source library using ACR122U. The last experiment is to validate the built library of NFC shield for Arduino with HCE application. These last two experiments show the output data is the same as the passcodes of Android application.

Department:	Electrical Engineering	Student's Signature	.....
Field of Study:	Electrical Engineering	Advisor's Signature	.....
Academic Year:	2015		

ACKNOWLEDGEMENTS

First of all, I would like to express my heartfelt gratitude to my thesis adviser, Assoc. Prof. Dr. Watit Benjapolakul, for his advice and valuable guidance throughout my study and research. I would like to thank to Asst. Prof. Dr. Lunchakorn Wuttisittikulkij for being the chairman of this thesis, and giving useful comments. I am also thankful to Asst. Prof. Dr. Chaiyachet Saivichit and Dr. Chaiyaporn Khemapatapan for being internal and external committee, respectively, for his suggestions.

I also would like to extend my thanks to friends, seniors and juniors in Telecommunication research laboratory for their kindness and sharing their experience during my research here.

I am particularly indebted to AUN/Seed-Net, JICA project for financial support during my stay in Thailand. I also would like to my acknowledgement to Institute of Technology of Cambodia (ITC) where giving me a chance to study here.

Lastly, I am highly indebted and thoroughly grateful to my parents and my brother. They have sacrificed everything for me. Without them, I cannot stand up here today.

CONTENTS

Page

Page

Page

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Introduction

People today carry their usual keys (physical keys, access cards, credit cards ...) to everywhere. At least two physical keys and an electronic card are taken every time. They keep these keys in their pockets or wallet. Therefore, they need more space to keep those things. In addition, they also carry the essential devices (smartphones, tablets, notebooks) in daily life.

Nowadays, smartphones are not just only for calling but for other purposes such camera, recorder, music player, game station, internet explorer and so on. In recent years Near Field Communication (NFC) have been integrated into smartphone. NFC which is in group of wireless technology is one of the most promising technologies. In short-range communication channel, NFC performs the data transfers with speed up to 424 kbps that operates on 13.56 MHz frequency and range of communication is maximum 4 centimeter between two devices. Since NFC started launching, there are two important components in its application. One is active NFC device for starting communication, and the other one is passive device such as tags or NFC cards which can store information as memory. This popular technology is mostly used in access control which is the selective restriction of access to a place or other resources. Smartphones today becomes more essential in a part of human life since it includes necessary multifunction. Many researchers try developing application to operate embedded sensors in smartphone for serving life style. Since NFC-enabled smartphone was released by Google and Samsung in 2010, the researchers were attracted and started taking its advantages. In mid-2011, Google wallet is a new mobile payment system developed by Google that allows the users to store their credit cards, debit cards or gift cards on mobile phone. NFC plays the main function in this system to make it efficient. Just tapping the smartphone on PayPass-enabled terminal at checkout in market, so it is unnecessary for users to take and keep any personal cards

in their pocket. Moreover, it indicates that smartphones technology are moving fast to replace physical world by virtual information.

1.2 Objectives

The intention of this thesis is to create an offline Android application which can use NFC-enabled Android smartphone as a single card emulator (Host-based Card Emulation or HCE) with key management (4 digits passcode) for NFC access control system; to pave the way for both debugging Android HCE on 64 bits Windows platform; and to distribute open library for Arduino to support NFC Forum Type 4 Tag by implementing ISO/IEC 7816-4 protocol.

1.3 Scope of thesis

This research focuses on, developing an application of NFC emulation card integrated with key access management (without server generated) for NFC access control system via host-based card emulator in smartphone, testing application using open library to debugging in Window platform over USB, and developing a library for Arduino to communicate with HCE over ISO/IEC 7816-4 The research will cover and consider the following issues:

1. To Develop an application for NFC-enabled phone with Android OS version 4.4 (KitKat) or upper.
2. To emulate an NFC card emulation by Host-card emulation mode (HCE) based-on *ISO/IEC 7816-4* (Identification cards – Part 4: Organization, security and commands for interchange).
3. To implement the service for a *NFC Forum Type 4 Tag Operation Version 2.0 to* transmit data from android application through embedded NFC chip of Android smartphone by standing on ISO/IEC 7816-4.
4. To modify and save passcodes in credential storage as data for HCE to use.
5. To configure, to extend, and to build the work from NFC open library to test HCE application to support Window platform 64-bit using NFC Reader over USB.

6. To develop a new library for Arduino to communicate with Android application (NFC Forum Type 4 Tag Spec) over NFC Shield Version 2.0.

1.4 Expected outcomes and contributions

After completing this research, we can expect the benefits over existing technology as below:

1. To replace and reduce carrying physical keys and the NFC tags and card by NFC smartphone.
2. To Emulate NFC phone as real NFC cards or tags without assistance from computer while the existing work needs the assistance (Computer) to generate key code
3. To hide 4 digits passcode for access control system using NFC and to access multi-system of access control.
4. To pave the way for developers testing and debugging Android HCE on Windows platform over USB.
5. To be the forerunner of a library for NFC Shield over ISO/IEC 7816-4 in order to apply in real system.

1.5 Organization of dissertation

In this section, it presents the rest of this thesis arrangement. **Chapter 2** describes the background of Android OS, NFC, and NFC-enabled phone, NFC Forum including the literature review. The detail of how to implement the Host-based card emulation through NFC Forum type 4 tag operation version 2.0 over ISO/IEC 7816-4, store credentials in internal phone in secured way in **Chapter 3**. **Chapter 4** talks about the test-setup, and result from the experiment and discussion. The conclusion of this dissertation, as well as the future work, are in **Chapter 5**.

Chapter 2

BACKGROUND AND LITERATURE REVIEW

2.1 Background

2.1.1 Android operating system

Android is the name of an operating system (OS) for mobile device. It is based on Linux kernel and is developed by Google. Android OS nowadays is a popular OS among three OSs (Android OS, iOS and Window phone) [1]. It is designed for touchscreen mobile device such as tablets and smartphones, and has the largest installed user base. This OS has direct manipulation interface using touch gestures such tapping, swiping and pinching. Not only are Android used on smartphones or tablets, but also televisions, wristwatches, digital cameras, and other electronic devices. Android is owned by Google under open source licenses and proprietary software including proprietary components for services. It is popular with information technology companies because it requires customizable, low-cost, and optimized operating system for high technology devices.

Until now the versions of Android OS is updated by starting from Android 1.0 to Android 5.0 - Lollipop as well as 5.0.2 with a few bug fixes, and Android 6.0 - Marshmallow (current version) [2].

2.1.2 Near-Field Communication (NFC)

Near–Field Communication or NFC, a contactless technology developed by NXP Semiconductor and Sony, is known as communication for short distance. NFC is the main point which is considered as the key to develop the application. It has been implemented in many information systems such as public transport, payment, ticket, access control systems [3], [4]. The main functions of NFC are information storing, readable and writable. NFC are divided into two main hardware; NFC device (Active) and NFC tags (Passive). NFC device is used for reading/writing data from/to NFC tags or NFC device. NFC tags, however, are used as information storage.

2.1.3 NFC-enabled phone

In 2010, Nexus S is the first Android NFC phone made by cooperation between Google and Samsung and it was the first NFC-enabled phone used in NFC payment system. After seeing its current advantages and application, new generations of smartphone made by many different manufacturers are embedded with NFC inside even though it is currently low functioning. NFC enabled smartphone can be operated in three different modes. These operating modes are defined by NFC-forum [5], [6], [7]; Reader/Writer (R/W), Peer-to-Peer (P2P), and Card Emulation (CE):

- Read/Write mode (R/W), NFC device can read and/or write passive NFC tags.
- Peer-to-Peer mode (P2P), allowing two active NFC devices to exchange data together.
- Card emulation mode (CE), NFC device itself acts as an NFC card.

2.1.3.1 Read/Write mode

In reader and writer mode, the NFC devices (Active tag) has ability to read and write NFC Forum-mandated tag types (Passive tag) such as NFC smart poster tag. Reader/Writer mode is to enable NFC device to read data stored in NFC tags or cards on RF interface.

2.1.3.2 Peer-to-Peer mode

In this Peer-to-Peer (P2P) mode, Two NFC devices (Only active devices) can exchange information bi-directionally at link-level with speed up to 424Kbit per second. Peer-to-Peer mode is standardized on ISO/IEC 18092 that specifies, coding, modulation schemes, transfer speed, and the RF interface's frame format. It is also initialization scheme and conditions required to control data collision during initialization. Furthermore, this ISO standard defines a transport protocols, protocol activation, in particular, data exchange method. P2P is mostly used in device pairing, networking,

and data transfer operation such as Android Beam function that allows to share files between two NFC-enabled Android devices by holding them close together.

2.1.3.3 Card emulation mode

There are two different types for card emulation mode [8], [9]; one is a secure element (SE) and the other is a host card emulation. Card emulation with a secure elements or SE, means that NFC module consists of two parts; an NFC controller and a secure element. NFC controller is used for the communication and the secure element is used for encrypting and decrypting secure information.

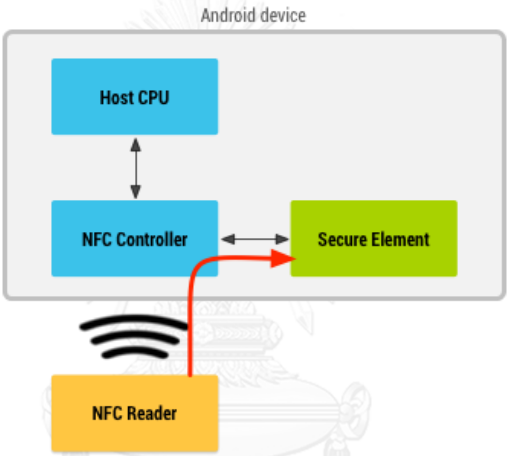

Figure 1: Routing data with a secure element [10]

Moreover, the SE can be considered as a smart card in phone. When NFC card emulation is provided by using a secure element, through an application to be emulated, the card is provisioned into the SE on the device. Then, when the user taps the device on NFC reader in Figure 1 [10], all data from the reader to SE are routed directly by the NFC controller in the device [8]. The SE and NFC reader start communicating with each other and Android application is not involved in the transaction at all. After the transaction completed, the Android application can ask directly to SE for the transaction status and notify the user. In NFC phone, SE can be embedded in SIM card, SD card or NFC chip [9].

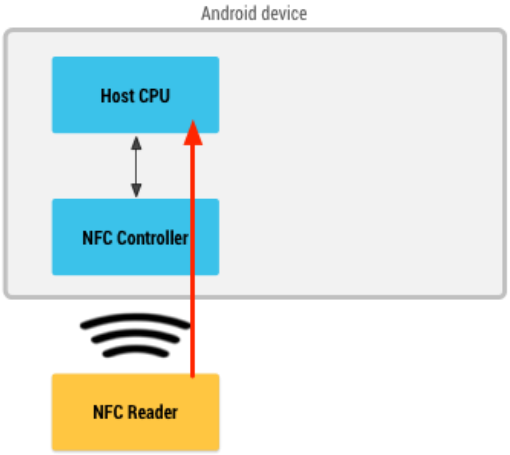

Figure 2: Routing data without the secure element [10]

Card emulation without a secure element or Host Card Emulation (HCE) in Figure 2 was introduced in Android 4.4. HCE does not rely on a secure element. This technique emulates the card by allowing application to talk directly to NFC reader. When HCE emulates the NFC card in Android phone, the NFC controller routes the data from the NFC reader directly to host CPU on which Android application is running [10].

2.2 Literature review

The study of current benefits and future directions of NFC service for mobile application was started by Ok, K. et al [5]. By standing on that statement, they conducted NFC application's literature review, application prototypes, and studies from academy and industry. On their overview on application, they present the application examined in survey table by classifying them based on each operating modes (R/W, CE, and P2P). An application named "Electronic key" in the table in [5] was in class of CE in order to eliminate carrying contactless smart key and physical key. However, when they investigated CE mode, they found out that this mode did not support mobility. It can be considered as mobile phone challenge and future developments issue to be solved and to be researched. Many researches for access control works on R/W mode and P2P mode such as Saminger, C. et al [11], Nasution S.M. et al [12], Teck, T. et al [13] and Pascal, U. [14]. Logical link control protocol (LLCP in P2P) was used by Hung, C.-H., Y.-W. Bai, and J.-H. Ren [15] to design of a door lock control based on NFC of a smartphone as keypad to open the door and implementation. They also proposed another design and its implementation of door

system based on NFC technology of a smartphone using a single button [16]. Being quite similar to this work, John et al [17] have proposed One-Time password (Via message) to access the door with ID and pin.

The biometric access control also is used to unlock the door [18]. They proposed to scan fingerprint via camera phone and send it (P2P) to the system. Moreover, another technique from Teh, P. et al [19] proposed NFC smartphone as a medium (P2P) to send stego-photo, a photo (object) that hided secret or important information in itself, of user (Information hiding) to unlock the door. Stego-photo, is the name after that object, had existence of the information, and is invisible. The server will extract the passcode from selected stego-photo and it will succeed if the passcode is matched in system. The existing embedded SE in NFC Android phone is the main purpose of payment system to ease the burden of researcher. The researcher starts work on NFC enabled smartphone with the secure element beyond payment system but it can be for access control system [20]. Practical attack scenarios on SE-enabled mobile device presents the security level of SE [21]. Christoph, B. et al [22] presented access control technique to unlock the car door by Android based smartphone with secure element in micro SD card. Anwar, W. et al proposed secure element and mobile trusted module as alternative SE access control for NFC enabled Android smartphone [23]. Their aim is to store secured data into an embedded secure element in NFC mobile phone. Furthermore, in the market, products of HID Global (Hughes Identification Devices) and ASSA ABLOY (August Stenman Stenman August, Ab Låsfabriken Lukkotehdas Oy), are using the secure element as the main point for access control. Secure element is controlled by mobile operator or handset manufacturer. This statement makes the difficulties for application developer and the end-user, since the developer needs to negotiate with device manufacturer or service provider and end-user needs to change or add external hardware such as SIM card or device [24]. To reduce any difficulties and use NFC enabled smartphone as a real card, Android developer releases Android OS version 4.4 KitKat (Application Programming Interface or API level 19) which has NFC host card emulation without secure element to enable smartphone to replace the NFC smart cards and tags. Saparkhojayev, N et al [25] and an Android application of Telcred product [26] are using this idea in their work to emulate the smartcard. But in

their Android application, they have no rights to select and save their own keys. The keys was received from the generation of server.

2.3 Problem statement

After observing the literature review, we can see that the researchers try developing Android's application for their own NFC access control system and they used many techniques (Peer-2-Peer or Read/Write mode) that all are not the official emulated cards or tags.

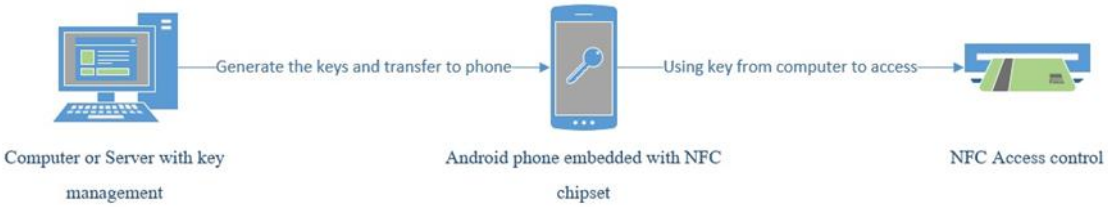

Figure 3: Generating keys from computer and transferring to Android phone

Even though Host card emulation (HCE) was released to solve the negotiation problems (Card emulation with a secure element), and the smartphones are small computing and mobile devices. However, emulation of NFC cards or tags in smartphone still needs the assistance from third-party (Computer) to generate keys for them in Figure 3 and the users have no right to use or change their own passcode immediately. Most of their works that using Card Emulation implemented over medium protocol, not the high standard provided.

On the other hand, many types of NFC reader today in the market are used with their own software to read/write the data to NFC cards and tags. While the Android HCE application is being developed over ISO/IEC 7816-4, we certainly need to execute and to debug this application on the real hardware with other trusted third-party software in order to make clear that it does work. However, those hardware and software are proprietary products and all of it are limited to their rights. One way to run and test Android HCE application we have to consider open source NFC library such as Google Chrome Application NFC Library [27] or NFC Tool Developer [28]. Their purpose are to contribute constructively by opening the library for any researcher and developers who are working with NFC hardware on a very low level. Google Chrome App NFC

Library (For Chrome OS) plays well with Chrome books connected with ACR122U and it has some problem with driver for Ubuntu. This Chrome Application supports with ACR122U and SCL3711 only and Window OS is unsupported for this version. This is the problem for the most of developers who use Window and Linux OS. NFC tool developer also released main NFC library to compile for many different OSs such as GNU/Linux, Mac OS as well as Window OS. But following the released Lib-NFC for window version, it currently supports only communication over UART not USB. By seeing these problems, we have some questions as follow:

1. Can we develop application for the version of Android 4.4 as smart cards or tags for access control system through NFC-enabled phone over ISO/IEC 7816-4?

2. How can we create and manage keys management in this application to give user the rights to manage those keys by themselves (moving the keys management from computer to the mobile application) to avoid assist from computer (server) in Figure 4?

3. How is the keys protected is protected from attack without privilege.

4. How can we test and debug this application in Window environment with NFC reader over USB?

5. Is this application able to talk with NFC Shield V2.0 over Arduino board?

2.4 Research proposal and its application

The target of this research is following the current statement of problem and solving those problems. So there are some different points from most existing research:

1. Using new NFC card emulation (host card emulation) which a few researchers are following.
2. Using medium protocols such as ISO 14443A [29], that is not high standard protocol for Android's HCE.

3. Giving user the right to manage the keys in Android smartphone that previous work used computer to manage all access keys and generate then transfer to the phone.
4. High performance x64 computers for Window users are not able to support NFC test using open library.
5. No supported ISO/IEC 7816-4 protocol library for academic researches such Arduino board.

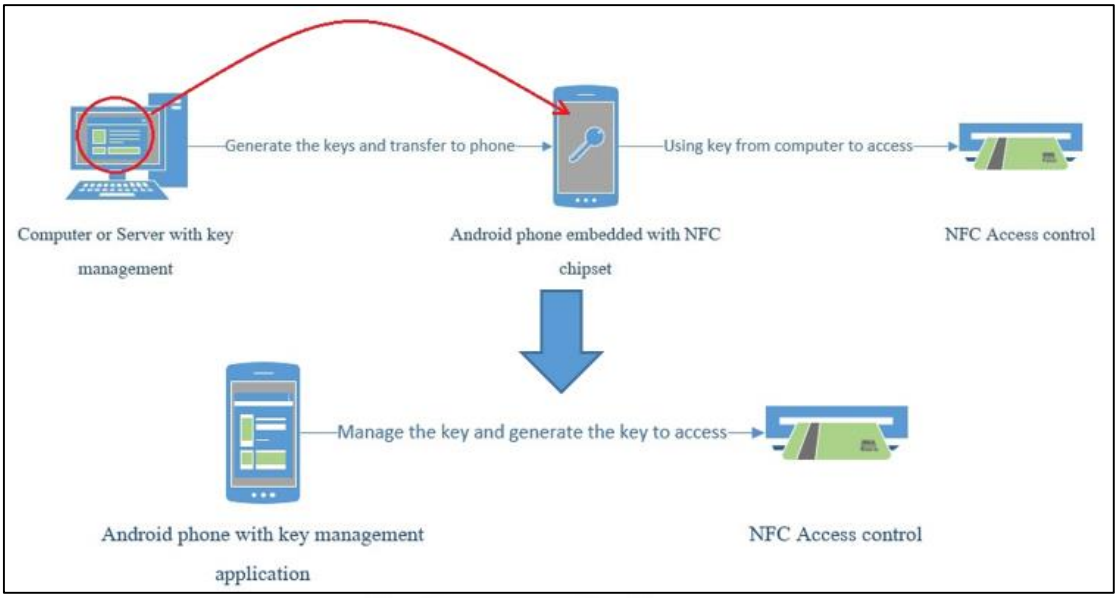

Figure 4: Proposal of the application development

This work is designed “NFC Unlocker” application in Android smartphone (Android 4.4 with embedded NFC chipset) and a library of Arduino support ISO/IEC 7816-4 protocol with an AID in order to apply in 4 digits passcode for access control system. This android application can store all 4 digits passcodes of each access control system. Users save passcodes in offline key management in application and then use it with careless of selecting the correct passcode if the correct one is saved (in Figure 5). The matched passcode is automatically unlock the door.

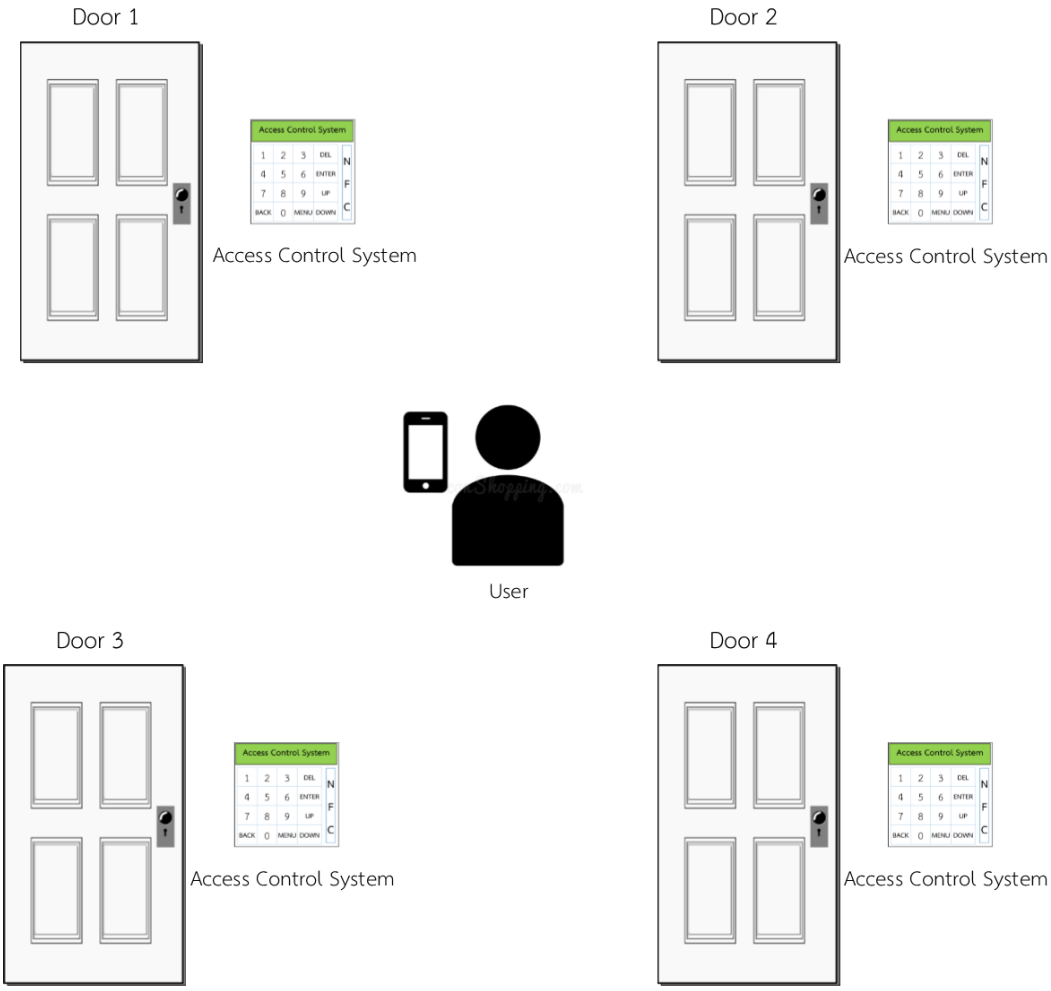

Figure 5: Applying application in the access control system

Chapter 3

NFC-ENABLED ANDROID SOFTWARE AND HARDWARE DEVELOPMENT

As what we described in the objectives above, our purpose is to develop an Android application in Android OS 4.4 (API 19) with the keys management that are different from most previous works which did not use official card emulation from Android and the user cannot manage their own keys with assist from computer and extending work from open NFC library to test APDU of application, next building a new library support last standardized protocol, then showing a method to test NFC application on Windows platform over USB that currently support only UART. The final step of this development, we test it with NFC shield using Serial communication (Serial Com). From the start point to the end point in the methodology we have several steps below and we will describe each step as follow:

1. Android IDE: Choose a development to develop an application having user interface.

2. NFC protocol support and service selection: Select an NFC protocol and service for the development.

3. Key management: Manage the password by user and encryption.

4. Application security: Increasing the application security when application is launched and resumed Credential data storage.

5. Extending work from open NFC library to test application on real hardware.

6. Preparing the test-setup for Android application and ACR122U over USB using Windows platform

7. Building a library for NFC shield for PN532 chipset over ISO/IEC 7816-4 protocol.

3.1 Overview of architecture

The architecture of Android application and computer (Windows platform) is illustrated in two different mediums. NFC reader is used in application development to test and debug it process in Figure 6.

Figure 6: Android application with NFC reader using Window platform

After successful development of Android application and development of library for Arduino, experiment of Arduino with Android smartphone is demonstrated in Figure 7.

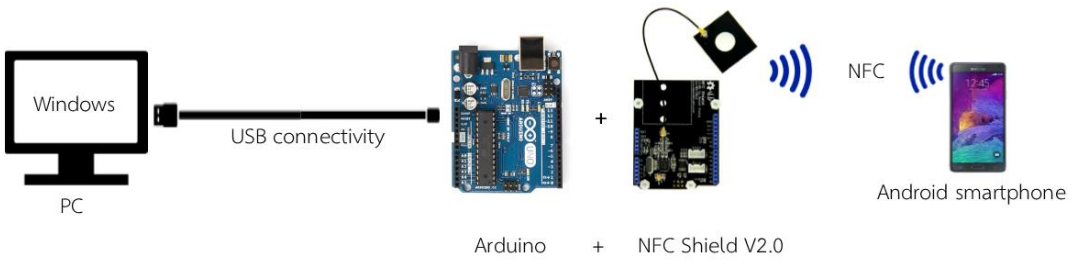

Figure 7: Android application with Aruido + NFC Shield V2.0

3.2 Android Application development

3.2.1 Android IDE

Android IDE (Integrated Development Environment) is a software application such as Android Studio in Figure 8 that provides comprehensive facilities to computer programmer for software development. By default, after installing Android IDE, developer needs to install Android SDK manager (Software Development Kit) that can download and install any tools, platforms, and other components into IDE. SDK is like a plugin for Android IDE. Android application are developed in Java programming language by using Software Development Kit.

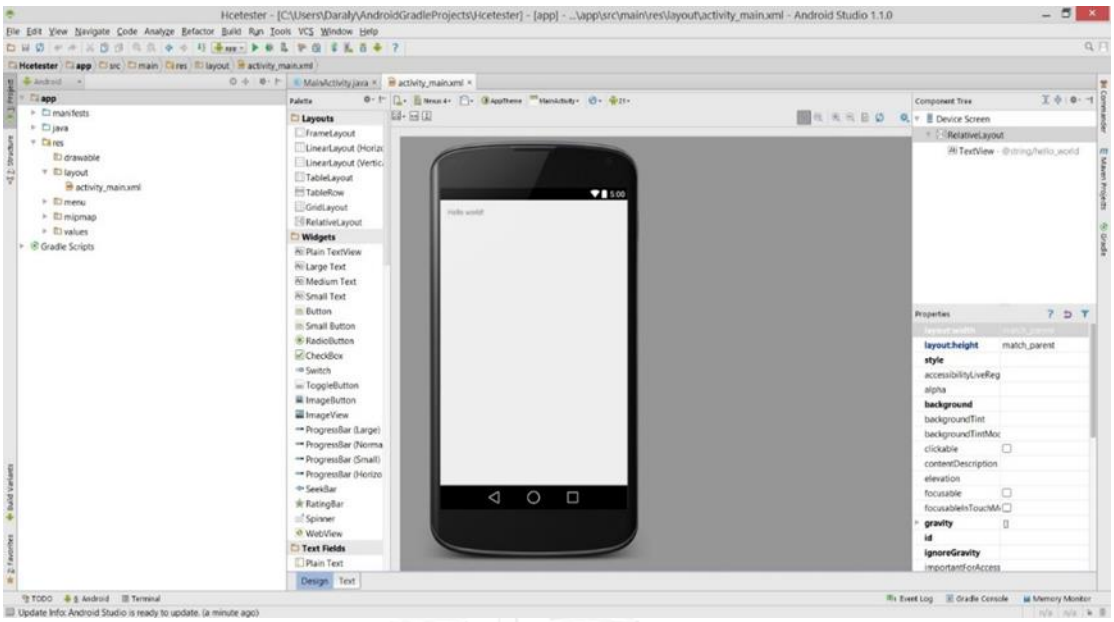

Figure 8: Android Studio IDE 1.1.0

3.2.2 User interface (UI)

User interface (UI) is one of the main parts of the application. To talk with application, users need the medium interaction to communicate with each other. It is difficult for end-user using dialect of shell scripts (Command-line interpreter) in Figure 9 unless they are advanced programming.

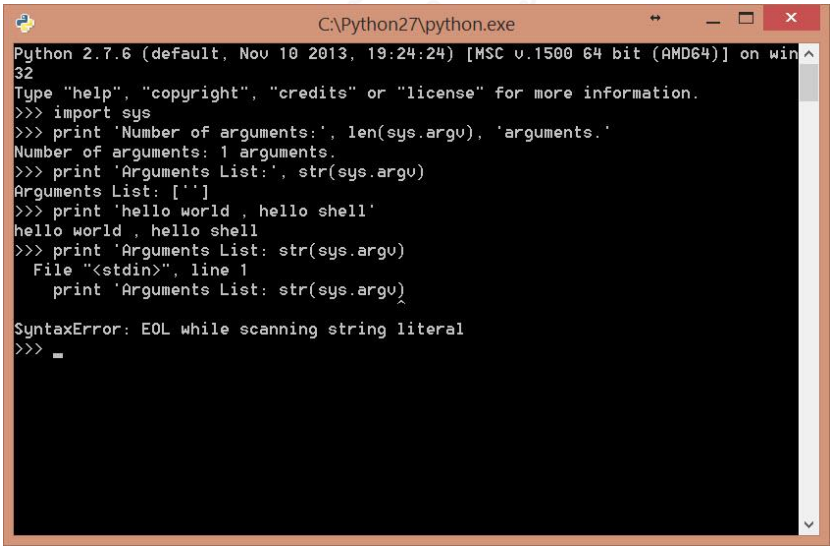

Figure 9: A sample of Python 2.7 command-line

UI is everything that users can see and interact with, in particular the use of control, manage, modify, or input to software. Everything that the application can do is shown

on the user interface. In this application, user interface is plays its role as NFC card and key management for users as a sample in Figure 10.

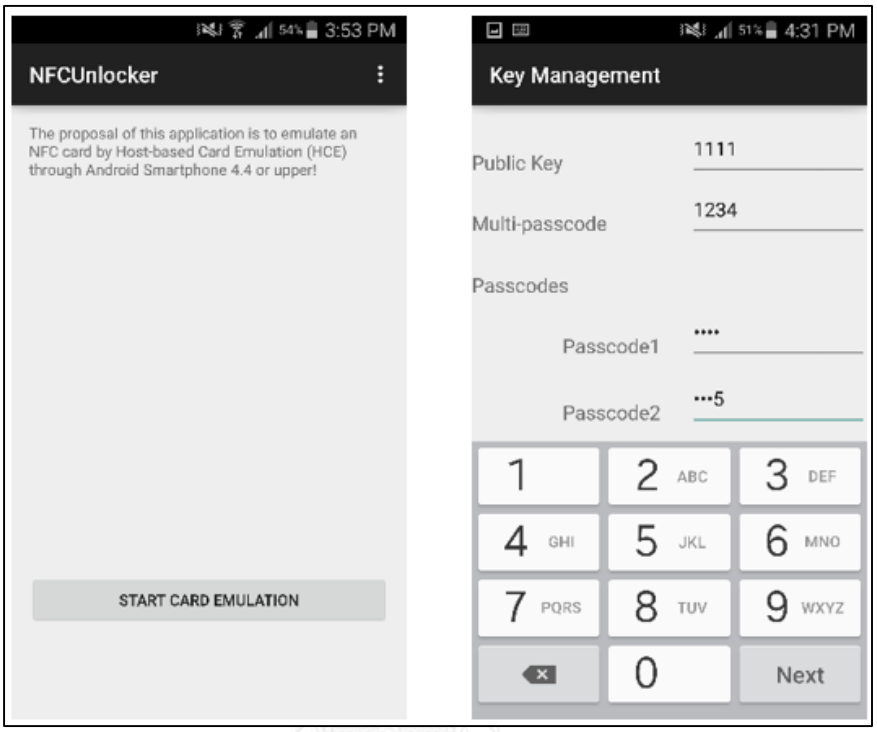

Figure 10: A sample of UI of an Android application

3.2.3 NFC protocol support and service selection

NFC technology standard offers support of many different protocols and different types of NFC cards and tags. For Android 4.4, it can support only a few protocols but these are in common and are supported by many NFC readers in the market today such as ISO 14443-4 [30], ISO 7816-4 [31] in Table 1, including Android NFC device functioning as reader themselves [10]. Android 4.4 can support emulating smart cards which based-on the NFC Forum (ISO-DEP specification in ISO/IEC 14443-4) and Application Protocol Data Units (APDUs) that is defined in ISO/IEC 7816-4 specification). Emulating ISO-DEP is mandated only on top of the NFC-A technology (defining in ISO/IEC 14443-3 Type A) and can support optionally for NFC-B technology (defining in ISO/IEC 14443-4 Type B).

Table 1: Protocol stack of Android's HCE

ISO7816-4: Card organization and structure
ISO14443-4: Transmission protocol
ISO14443-3 Type A: Activation and anti-collision
ISO 14443-2: RF signal interface
ISO14443-1: Physical layer

3.2.3.1 HCE Service (Host Card Emulation)

The HCE architecture in Android is based on Android service components. A key advantage of the service in Android is that it can run in background without any user interface (UI). However, this key should be disabled in this application to avoid access without authorization. Another key for HCE service follows the specification.

3.2.3.2 Service selection

This service is selected for the reader and card. When the user taps the device to an NFC reader, the Android system needs to aware of which HCE service the NFC reader really wants to talk to. This is where the ISO7816-4 specifications [31] come in by following NFC forum technical specifications. It is the way to select application, centered on an AID (Application IDentification) which consists up to 16 bytes.

3.2.3.3 AID group and categories

In several cases, an HCE service may need to register multiple AIDs to implement an application, and it needs to be sure that it is the default handler for all these AIDs (as opposed to some AIDs in the group going to another service). Each AID group can be

associated with a category. This allows Android to group HCE service together by category, and that in turn allows the user to set defaults at the category level instead of the AID level. Android 4.4 supports two categories: CATEGORY_PAYMENT (covering industry-standard payment application) and CATEGORY_OTHER (for all other HCE applications) [10].

3.2.3.4 Key management

Multi-passcode is the main key in this purpose. It is a key to be verified by the NFC reader. If it is matched with access control system then the system starts searching in the transmitted data for the passcode. If it is found, the system returns for allowing. Otherwise, it is prohibited.

The user can create and modify their password anytime in key management. Each passcode is contained with two inputs, one is for passcode input to unlock the access control, and the other can be for description that remarks the password in Figure 11.

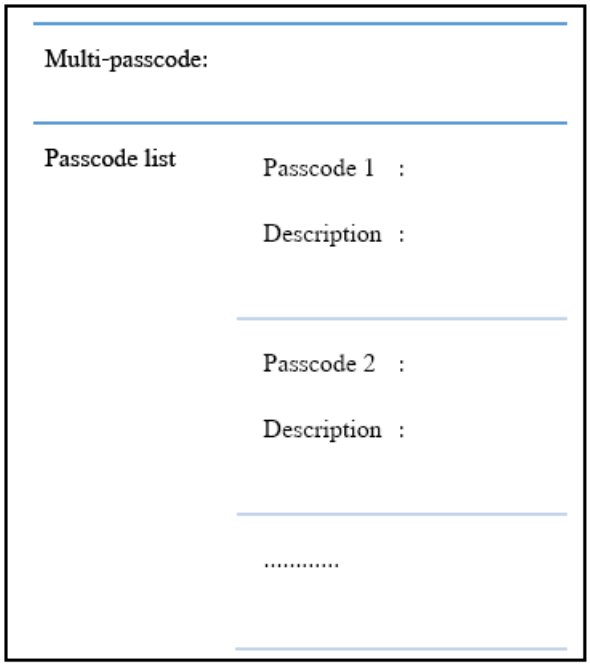

Figure 11: Passcode list

3.2.4 Access passcode

The keys which is created and modified are access codes. Each access passcode has 4 digit which is the most common key code for access control such as door's access control, ATM or SIM's PIN. To inhibit from eavesdropping, each NFC access control prefers 4 digits (in Figure 12) access control but we can replace pressing these passwords on the input-keypad by just tapping the phone on the control system that makes anyone not able to see it.

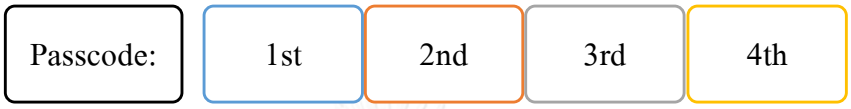

Figure 12: 4 digits of a passcode

3.2.4.1 Formation

Formation is an aspect after the passcodes are saved. Those passcodes are formatted as a text line and encrypted and stored in credential data storage as describe in Section 3.2.4.2 for transmitting to NFC reader:

Key format:

(Header**)-(**Multi-passcode**)-(**Passcode1**)-(**Passcode2**)-(**...**)-(**PaacodeN**)-(**Footer**)**

3.2.4.2 Credential data storage

The phones actually have two storages; internal storage which is internal phone's storage, and external storage that can insert and remove to/from device such as SD memory card. The internal phone's storage which is chosen from these two storages is safer to prevent the data lost or damage. Android has two others ways to store data in the device [32], those are SharedPreferences and databases. SharedPreferences allows the application to store the pair of name and value (primitive data types only). Android's database includes a SQLite to create and manage structured data more complex than name/value of SharedPreferences. Both of these ways have isolation system based on underlying Linux permissions. From Section 3.2.3.4, the data contains two inputs, name and value. The name refers to Multi-passcode and Passcode, and the value denotes the integer value of those name. So Android's SharedPreferences

is good choice to store data in device for easy access. All passcodes are saved as a file in the device's filesystem.

All stored data in credential storage are secured as the filesystem. Application will set by default within the application's data directory in filesystem set. To access them, filesystem permissions allows only the UID with which the specific application runs. In addition, advanced programming users are able to hack to the Android system such using running two applications with the same UID or etc. After all, this application works securely in isolated mechanisms since it supports only Android 4.4 and upper, which means that this version come up with Security-enhanced (SE) Android [33] – was develop by the NAS (National Security Agency) and Red Hat for Linux server to provide security of kernel-level which is able to be used as a firewall of application-level and was developed as mandatory access control system (MAC) to improve the security on Linux server. As a result, MAC sets the enforcement form the kernel to access data. So even the users with the administrator's privileges cannot access to the private data in Android devices. SE Android prevents credential data from unbridled access that makes the malwares cannot bypass security features unless Android phones are rooted to modify SE Android policy.

3.2.5 Application security

Moreover, to protect itself in case of theft, this application needs user to use a password before it is launched. Furthermore, in some cases of pausing, although the activity is resumed, the user still needs to input the correct password to launch it. The application password is stored in credential data as same as access passcode.

3.2.6 Cryptography

Talking about cryptography, they will think of encryption which simply refers to converting a plaintext to ciphertext (scrambled form) for preventing attackers who try to deduce the message from the encrypted form. In this section, Advanced Encryption Standard 256 bits algorithm (AES-256), one of 3 AESs (128, 192 and 256 bits) from high to higher, is applied for the private key with public key for advance security encrypted/descripted data established by United State National Institute of Standards

and Technology (NIST) and recently it was used in “NFC-enabled Access control and Management system” [25].

3.2.7 Proposed method

The aforementioned Section 3.2.3, 3.2.4 and 3.2.5, the basic of HCE service implementation is illustrated such as key management and security activity with encryption.

3.2.7.1 Implementing Host ADPU service

This service is a component that handles all transactions of NFC. It allows an application to emulate an NFC card using HCE.

First of all, checking the feature of NFC Host Card Emulation by using the <*uses-feature*> tag in the application manifest to declare that this application uses the HCE feature for the reason that every last models of smartphone neither have NFC embedded chipset nor support HCE.

Secondly, extending the service implementation by using HostApduService class. The processCommandApdu() method is called when the ADPU is defined in ISO/IEC7816-4 called. This ADPU are used to be the application-level packets to exchange between the HCE service and Reader in halt-duplex. The Reader typically transmits the first AID to the smartphone at “SELECT APDU” state. In this state, APDU contains AID will extract the AID and resolve it to the service, and next forward ADPU to the service. ISO-DEP protocol in this ISO/IEC will be initiated by the reader. RATS (Request for Answer To Select) command will be sent and its response, the ATS, will be generated by NFC controller. But the service implementations are required to meet the requirement of NFC Forum for this response to be able to count on those parameters setting in accordance with NFC Forum.

3.2.7.2 Key management and security activity

The process of this activity is to be able to modify multi passcode and the passcodes. In the key management activity, it has the input the integer number boxes for editing and pairing with its names. The Multi-passcode and passcode can be modified there.

If it is “No”, the activity close with do nothing. But, if yes, everything are saved, this activity will be closed, and the process in Figure 13 happening in the meanwhile. A method, the concatenating method, is called. It will sort the multi-passcode and the passcodes that is gotten from the edit text object. The output will be as sequence like the Section 3.2.4.1 and then the encrypting method catches this output. The process of encryption starts converting the concatenated passcodes using the symmetric-key encryption, referring to the same key for both encryption and decryption, to scrambled text. Similarly, password for protecting application is stored in the same encrypting way as the passcodes after saved or exit if it is cancelled. When secondary activity is called, the preferences method, which is used to get the data from the storage, starts decrypting the encrypted passcode and then loads to each text boxes, respectively. Likewise, the password activity of the application does. The whole interaction of key management and password activity with the storage structure is demonstrated in Figure 14.

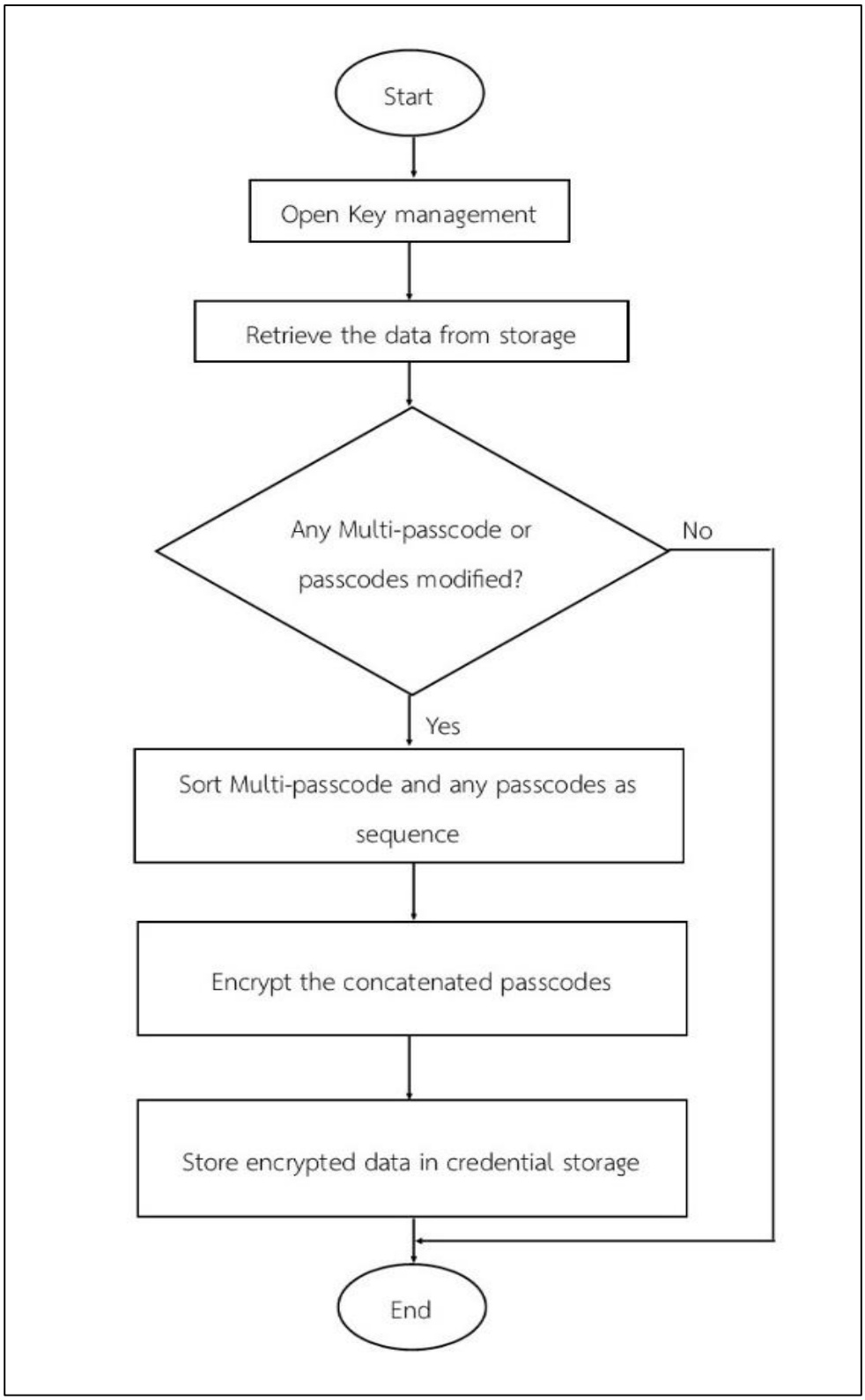

Figure 13: Flowchart of key management process after saved

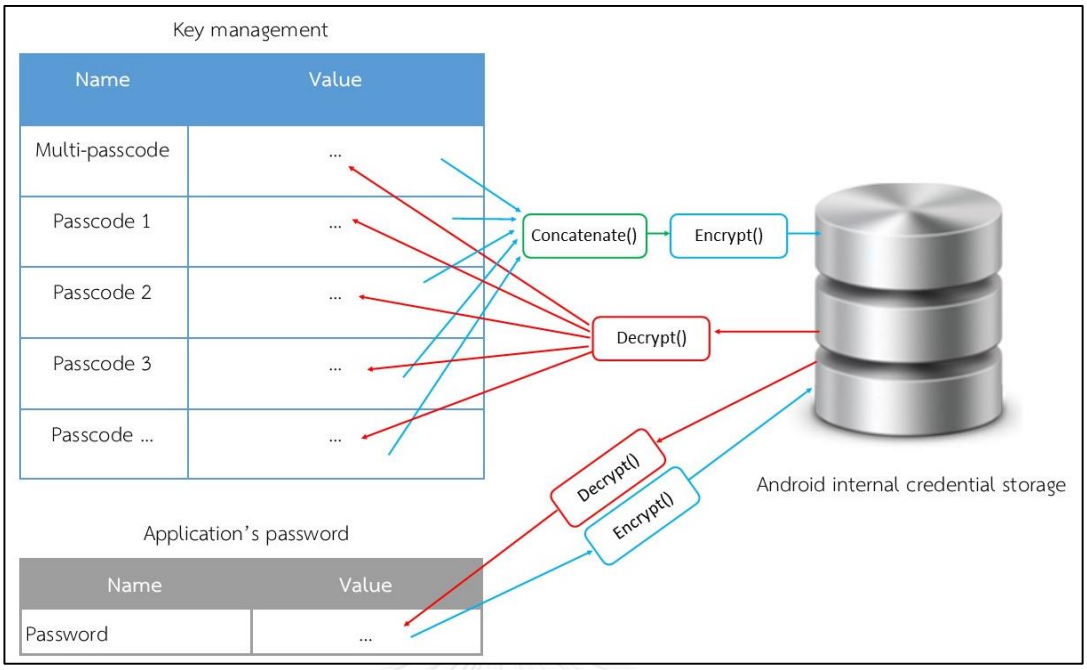

Figure 14: Application credential storage structure

3.2.7.3 Application structure

Combination of Section 3.2.7.1 and Section 3.2.7.2, its structure is showed in Figure 15.

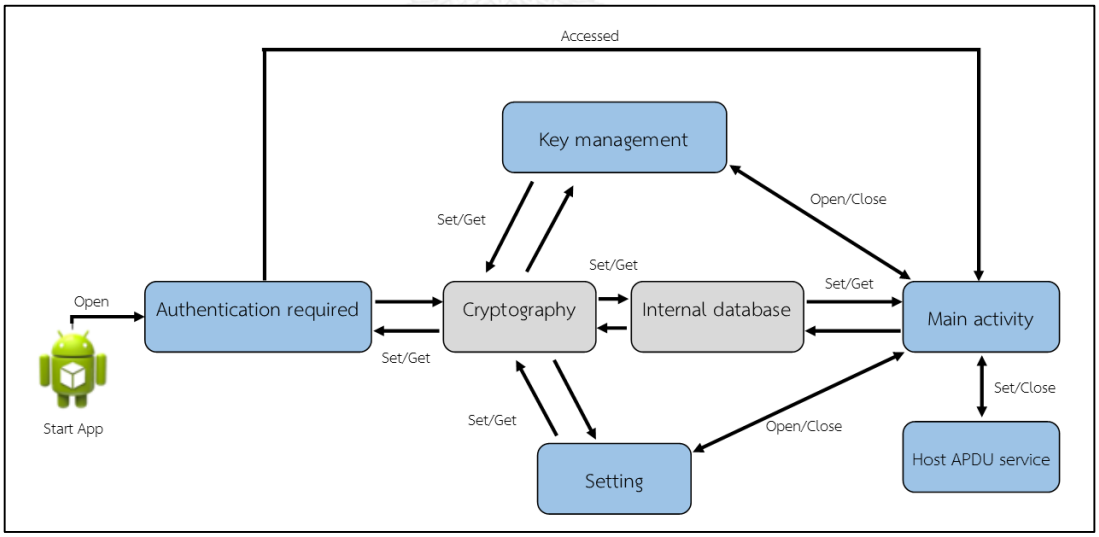

Figure 15: Application structure

Authentication is required to check for authorized users who can use this application. After the right users accessed, Main activity is main interface to manage other activity such as Key management, Setting and Card emulation. Set/Get function is used for read and write the value in internal storage.

3.3 Hardware development

3.3.1 Developing a library of NFC shield for Arduino

Currently, The NFC's library for Arduino to communicate with Android's HCE (ISO/IEC 7816-4) is not existed and NFC Shield V2.0's library cannot support ISO/IEC 7816-4. In this part we, in brief, describe algorithm to develop Type 4 Tag Specification [34] library for the Shield as reader side to support ISO/IEC 7816-4. For writer side, it is not excluded in account of having HCE application. This part addresses basically how the reader talk to the card emulation.

To read to NDEF (NFC Data Exchange Format), the reader needs to follow the below step according to the specification [34] and ISO/IEC 7816-4 [31] using the Application Protocol Data Unit message Command-Response pair, Application Protocol Data Unit (C-APDU) in Table 2 and Response – Application Protocol Data Unit (R-APDU) in Table 3.

Table 2: C-APDU Format

CLA	INS	P1	P2	Lc (optional	Data (optional)	Le (optional)
Class byte	Instruction byte	Parameter byte 1	Parameter byte 2	Length command	Data bytes	Length expected

Table 3: R-APDU Format [34]

Response body (optional)	SW1	SW2
Data bytes	Status Word 1	Status Word 2

NDEF Tag Application select procedure in Figure 16(a). This procedure is to select the NDEF Tag Application by sending the Select command to the passive device (emulated card) and waiting its response. The Application ID from reader must be as same as the card. If not, the APDU select procedure fails and the next process certainly aborts, then the device starts implementing both Mapping Version 1.0 (in Type 4 Tag specification 1.0) and Mapping Version 2.0 (Type 4 Tag specification 2.0 [34]).

Table 4: NDEF Tag Application select – C-APDU [34]

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

- Application ID or AID is set by develop for both the reader and card according to Section 3.2.3.2 and 3.2.7.1. In this application, data of AID is F0394148148100.

Table 5: NDEF Tag Application select – R-APDU [34]

	Data	SW1	SW2
Completed	File control information MAY be returned	90h	00h
NDEF Tag App not found	-	6Ah	82h

Presuming that the NDEF Tag Application succeeds then the next procedure occurs. The Figure 16(b) illustrates Capability Container select procedure. The procedure is to select the CC file usning C-APDU which the parameter of the Select command is set to select by EF (the Elementary File) in
Table 6. After sending this command, the card responds with the command in Table 7 to the reader to confirm the process as below.

Table 6: CC select command – C-APDU [34]

CLA	INS	P1	P2	Lc	Data	Le
00h	A4h	00h	0Ch	02h	E103h	Not present

Table 7: CC select command – R-APDU [34]

	Data	SW1	SW2
Completed	Not present	90h	00h
CC not found	Not present	6Ah	82h

The Figure 16(c) represents the CC file read procedure. Afer completed previous process successfully, this procedure is to read the data from CC file. It read the CC file (15 bytes of CC file in Table 5 in [34]) using the ReadBinary command in Table 8 with zero offset in the ReadBinary and response in Table 9. If the CC length is less than 0Fh or read access witout condiction is not granted, the file is not valid. The valid range of Le is from 01h to FFh.

Table 8: ReadBinary command – C-APDU [34]

CLA	INS	P1	P2	Lc	Data	Le
00h	B0h	Offset	Offset	Not present	Not present	Length Le

Table 9: ReadBinary command – R-APDU [34]

	Data	SW1	SW2
Completed	Content read	90h	00h

The NDEF file select procedure is demonstated in Figure 16(d). Read sends the NDEF select command to the card with a detail in Table 10 and then waits the response in Table 11 as below.

Table 10: NDEF select – C-APDU [34]

CLA	INS	P1	P2	Lc	Data	Le
00h	A4h	00h	0Ch	02h	File ID	Not present

Table 11: NDEF select – R-APDU [34]

		Data	SW1	SW2
Completed		-	90h	00h
NDEF not found		-	6Ah	82h

Next procedure is to read the NDEF Length (NLEN) in Figure 16(e). The condition of Type 4 Tag is applied in this state (detailed condition provided in [34]). To read the NLEN of NDEF file, using the ReadBinary command In Table 12, is starting from offset zero, and the response is corresponding to Table 13.

Table 12: NLEN read command – C-APDU

CLA	INS	P1	P2	Lc	Data	Le
00h	B0h	Offset	Offset	Not present	Not present	Length Le

Table 13: NLEN read comman – R-APDU

	Data	SW1	SW2
Completed	Content read	90h	00h

The last procedure is the NDEF data read in Figure 16(f). NLEN read uses the ReadBinary command and response for the process. Similarly, the NDEF data read uses this command to read the NDEF message and it stats at offset 02h of the file.

Table 14: Data read command – C-APDU

CLA	INS	P1	P2	Lc	Data	Le
00h	B0h	Offset	Offset	Not present	Not present	Length Le

Table 15: Data read command – R-APDU

	Data	SW1	SW2
Completed	Content read	90h	00h

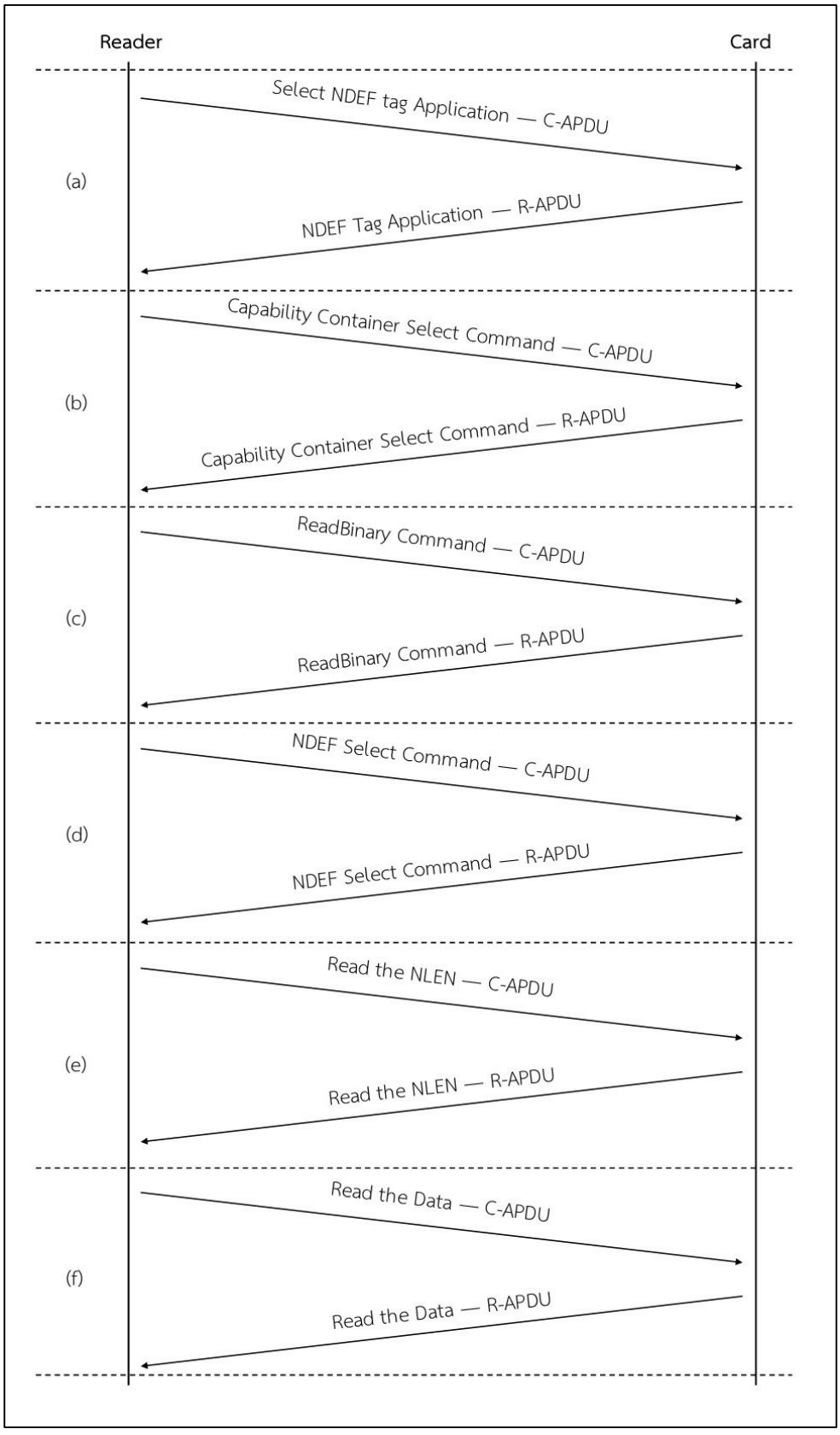

Figure 16: APDU procedure

3.3.2 Extending work from libnfc open library

The purpose of this work is to test ADPU of card and detect each step of the thrown data from HCE service by extending work [35] from open library to establish a test-setup for ACR122. The work from libnfc library using APDU exchange is not enough for reading entire HCE's data but it can complete only 3 steps in Figure 16 from (a) to (c) in Section 3.3.1 that is from the "NDEF App Tag select" to the "CC file read". To extend the current work, this Section shows the groundwork for the "NDEF select" to the "Data read" as below and these steps are represents in Figure 17.

- Declaring the library from stdlib.h, string.h and nfc/nfc.h.
- Manually setting AID as same as HCE application's AID.
- Manually setting the C-APUD command, size of memory, file ID to transmit as mentioned in Section 3.3.1.
- Using the Card Transmit method to determine that the response.
- Validating the response for each process.

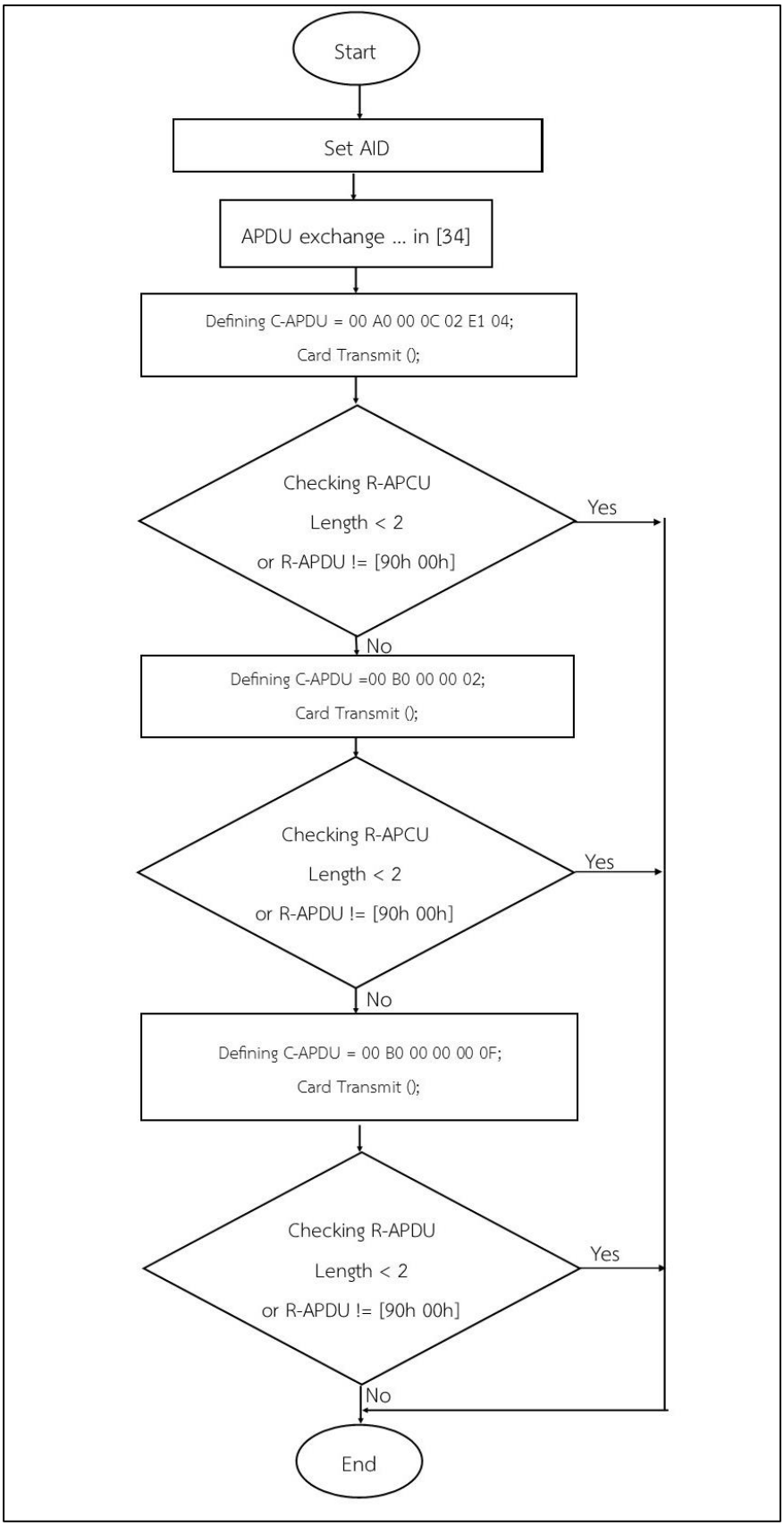

Figure 17: Flowchart of extended APDU exchange from libnfc

Chapter 4

EXPERIMENT AND DISCUSSION

In this section, we set up test to deploy the secured data in the Android internal data storage, and emulate the output data of Android HCE application over Type 4 Tag Specification 2.0 using ISO/IEC 7816-4 with both ACR122U Reader and NFC Shield V2.0.

4.1 Credential data in internal storage

This section figures the stored data out in Android database. After compiled and running to the Android phone, the credential data is stored in internal storage, located in "\data\data*app_name*", is ideal for safety in unrooted Android phones. In contrast, rooted Android phone was mentioned in Section 3.2.4.2, it means that this phone is give users being the super users. In other words, it gives all uses gain full access to their phone without restriction. In fact, Google and Android smartphone manufacturer do not recommend the end-users to root their phone for the reason that it is harmful to the security system. In this case, this application uses the encryption to help stolen credential data. In Figure 18, it shows the rooted Android 4.4 was accessed to Android security system and duplicated the private data from this application. Hacker, however, cannot be deduced the meaning of those passcodes in Figure 19.

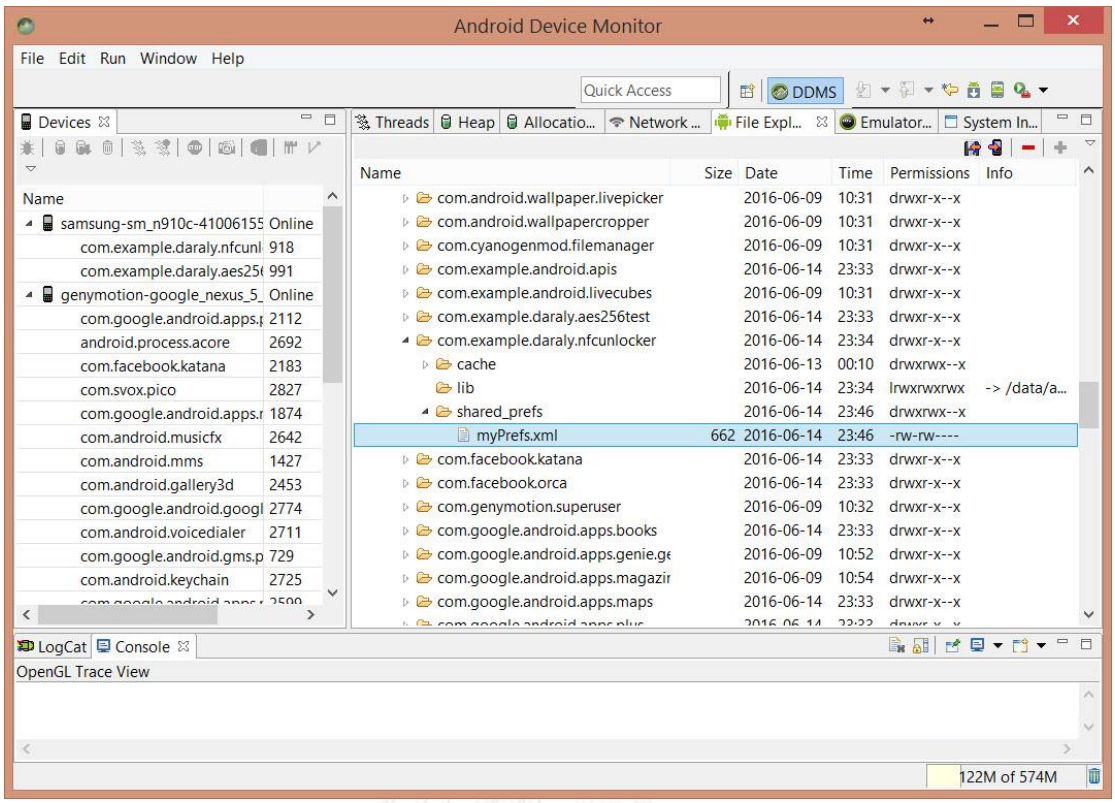

Figure 18: Android device monitor in rooted smartphone

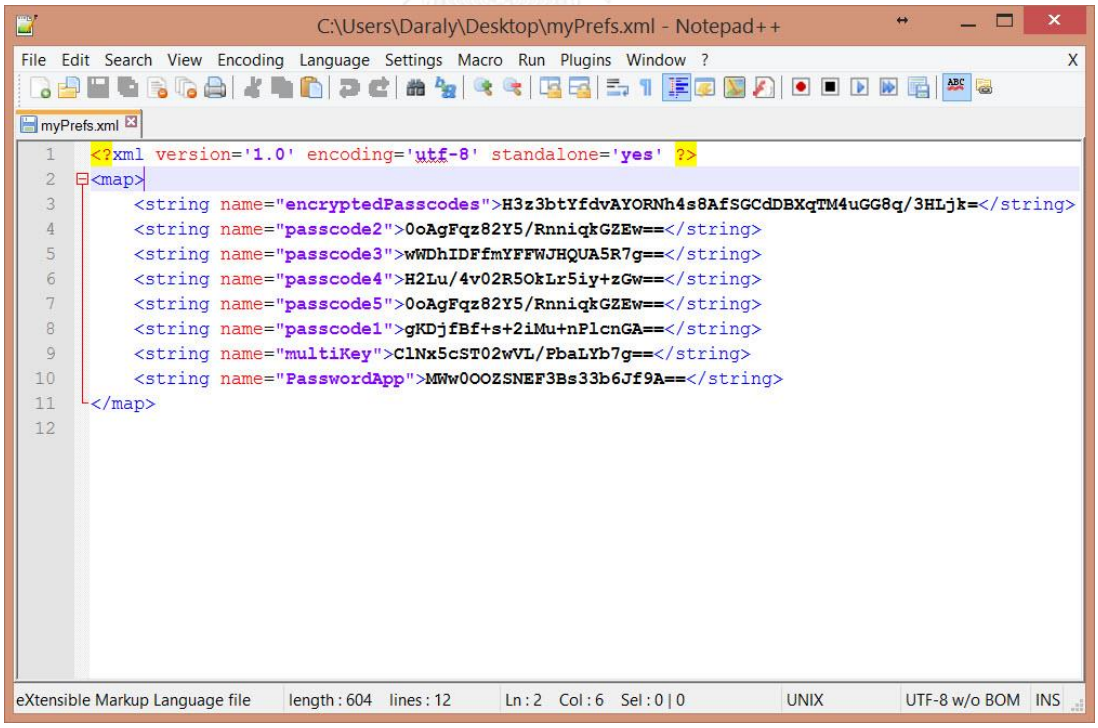

Figure 19: Access the stored encrypted data in rooted Android smartphone

4.2 Test-setup with ACR122U Reader

In this test, we use hardware as below:

Devices	Specifications
Laptop	- Model: Sony VAIO F2 - Processor: Intel i7 2720QM CPU @ 2.20GHz - OS: Window 8.1 - System type: 64-bit Operating system, x64-based processor - USB: Version 3.0
Smartphone	- Samsung Galaxy Note 4 (SM-910C) - Operating system: Android 5.0.1 - NFC chipset embedded
NFC Device	- ACR122U - Firmware 1.6

As described in Section 2.3, we will do experiment in 64-bit Window platform to test HCE application. This experiment follows some instructions of Lib-NFC for window which currently only support over UART [19] to communicate with NFC reader over USB by installing, configuring and using libnfc on Window. What we have to do is to install below software which are used in this experiment:

Software used	Version
1. TDM-GCC MinGW Compiler [36]	5.1.0 (x64)
2. libusb-win32-bin [37]	1.2.6.0
3. PCRE [38]	7.0
4. CMake [39]	3.5.2 (x86)
5. Doxygen [40]	1.8.11
6. libnfc [41]	1.7.1

All software used, add them in one directory: example “C:\Tools\”

Step 1: Installing TDM-GCC

- Create: Create a new TDM-GCC installation.
- Setting it as MinGW-w64/TDM64 Experimental (32-bit and 64-bit).

- Select the type of install: Recommended, All packages.

Step 2: Fetching and unzip to the same directory.

Step 3: Installing PCRE with full installation.

Step 4: Installing CMake and add CMake to the system PATH.

Step 5: Installing Doxygen.

Step 6: Unpacking libnfc to Tools directory.

Step 7: Configuring libnfc.

- To avoiding a bug in Window, we have to modify a file “CMakeLists.text” by writing one line in libnfc.

From

```
MACRO (GET_CURRENT_YEAR RESULT)
  EXECUTE_PROCESS(COMMAND "cmd" " /C date /T" OUTPUT_VARIABLE ${RESULT})
  STRING(REGEX REPLACE ".*(..)/(..)/(....).*" "\\3" ${RESULT} ${${RESULT}})
ENDMACRO (GET_CURRENT_YEAR)
GET_CURRENT_YEAR(CURRENT_YEAR)
```

To

```
MACRO (GET_CURRENT_YEAR RESULT)
  EXECUTE_PROCESS(COMMAND "cmd" " /C date /T" OUTPUT_VARIABLE ${RESULT})
  STRING(REGEX REPLACE "\n" "" ${RESULT} ${${RESULT}})
  STRING(REGEX REPLACE ".*(..)/(..)/(....).*" "\\3" ${RESULT} ${${RESULT}})
ENDMACRO (GET_CURRENT_YEAR)
GET_CURRENT_YEAR(CURRENT_YEAR)
```

- Modifying a file in “libnfc\libnfc\nfc-internal.c” by changing value from false to true of “allow_intrusive_scan”.

```
  // Set default context values
  res->allow_autoscan = true;
  res->allow_intrusive_scan = true;
#ifdef DEBUG
  res->log_level = 3;
#else
```

Step 8: Rename the a file “README.md” to “README”.

Step 9: Modifying a library file in “libnfc\libnfc\drivers\acr122_ubs.c” by removing a line of “usb_reset(data.pudh) to solve the problem of reset while it is processing.

```
        continue;
    // Reset device
    usb_reset(data.pudh);
    // Retrieve end points
    acr122_usb_get_end_points(dev, &data);
    // Claim interface
```

Step 10: Adding some PATH into the “System Variables” in “Computer Properties> Advances system settings > Advanced Tab> Environment Variables”

PATH: “C:\Tools\MinGW64\bin; C:\Tools\doxygen\bin; C:\tools\MinGW64\x86_64-w64-mingw32\lib32; C:\tools\MinGW64\x86_64-w64-mingw32\include; C:\tools\CMake 2.8\bin; C:\tools\GnuWin32\bin; C:/Tools/libnfc;”.

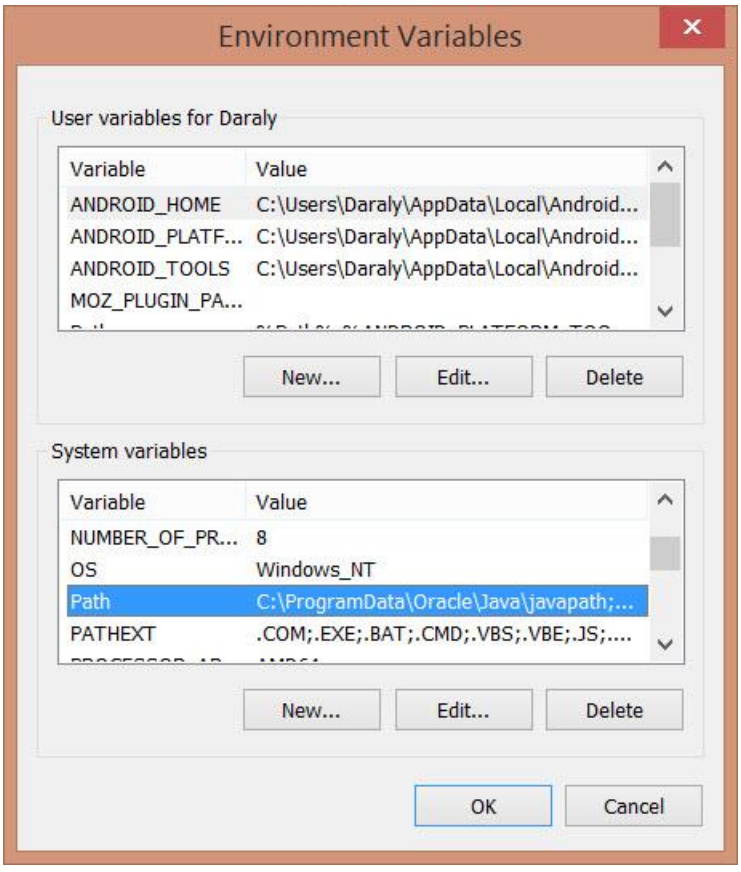

Figure 20: Set system variables

Step 11: Open the cmake-gui.

- Selecting the source directory to libnfc.
- Creating a build directory and set it as where to build the binaries in Figure 21.

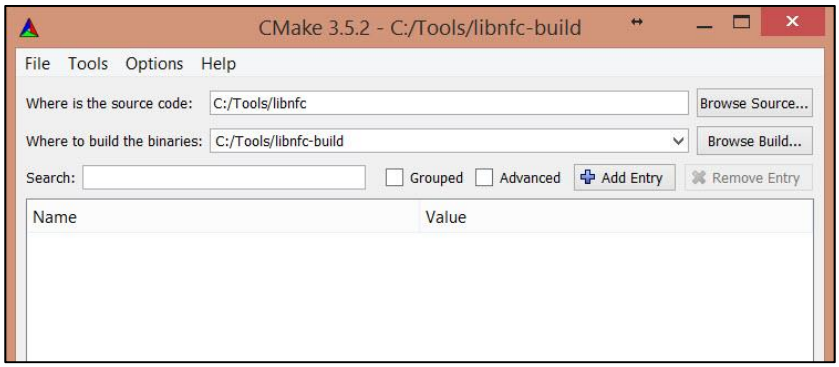

Figure 21: Setting the build directory for CMake

- Clicking on the “Configure” and specify the generator for the project with “MinGW Makefile” using default native compilers.

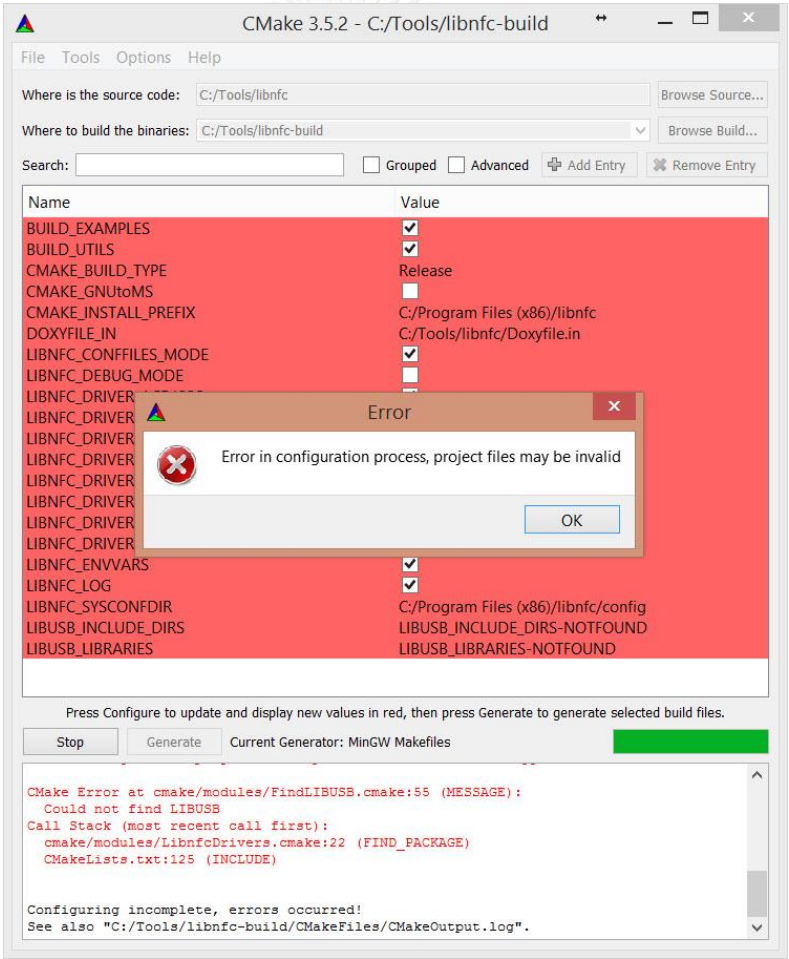

Figure 22: Editing the entry of cmake

- Some errors exactly occurs. To fix these, editing the entry again by browsing path for “LIBUSB_INCLUDE_DIRS” to directory of “\libusb-win32-bin-

1.2.6.0\include" and path for "LIBUSB_LIBRARIES" to a file "\libusb-win32-1.2.6.0\lib\gcc\libusb.a".

- Adding new path by naming "LIBNFC_ROOT_DIR" with value "\libnfc"
- Make sure that LIBNFC_DRIVER_ACR122_USB is ticked as Figure 23.

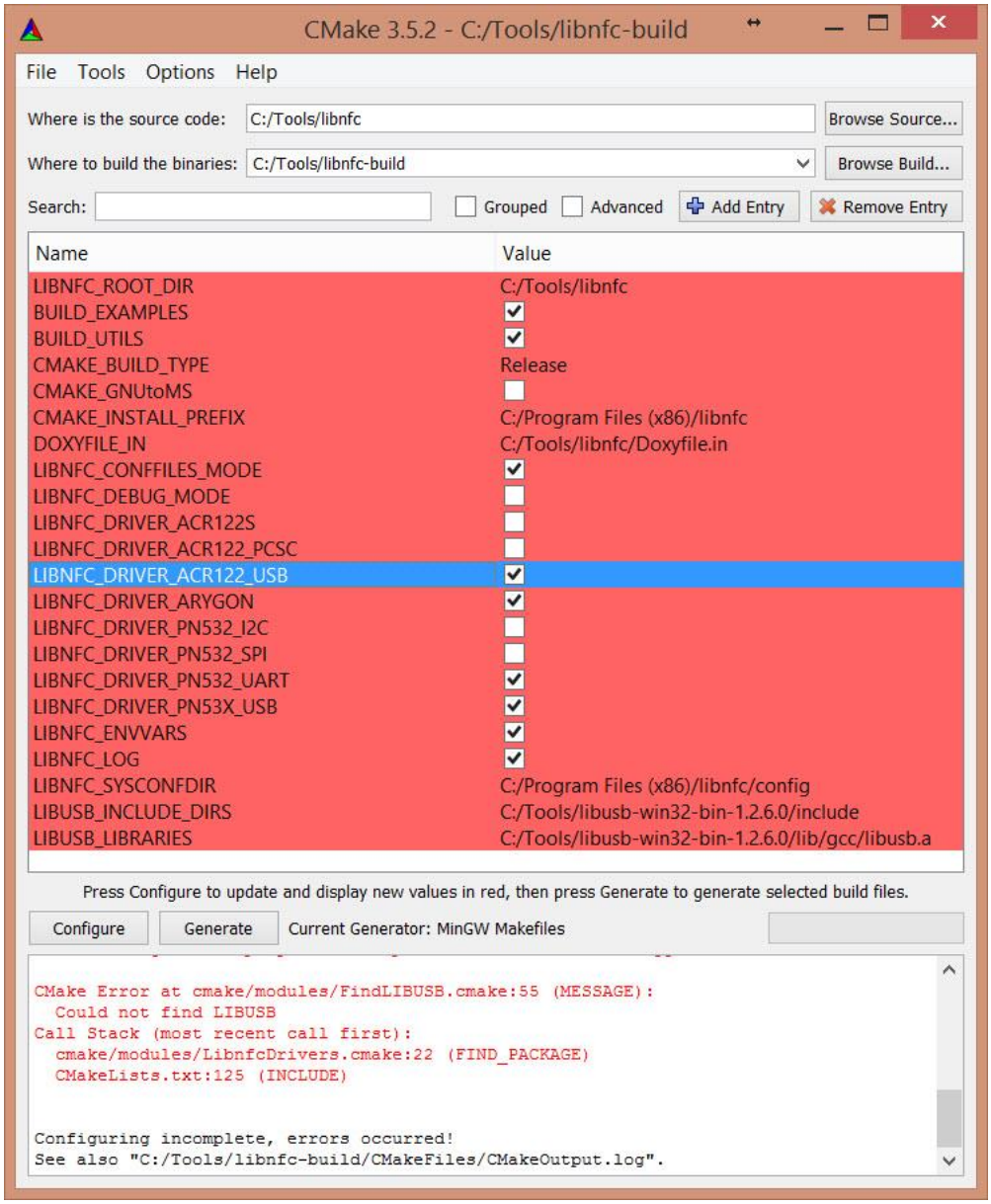

Figure 23: Fixing errors in cmake-gui

- Clicking on "Configure" button again. It has an error with LATEX but it is not matter. Ignore it and click on "Generate".

Step 12: Copying a file "apdu_extended_read.c" (mentioned in Section 3.3.2 to "\libnfc\examples\" in order to compile with others C files. Then modify

CMakeList.txt in the same directory of example to point this file to execute as Figure 24.

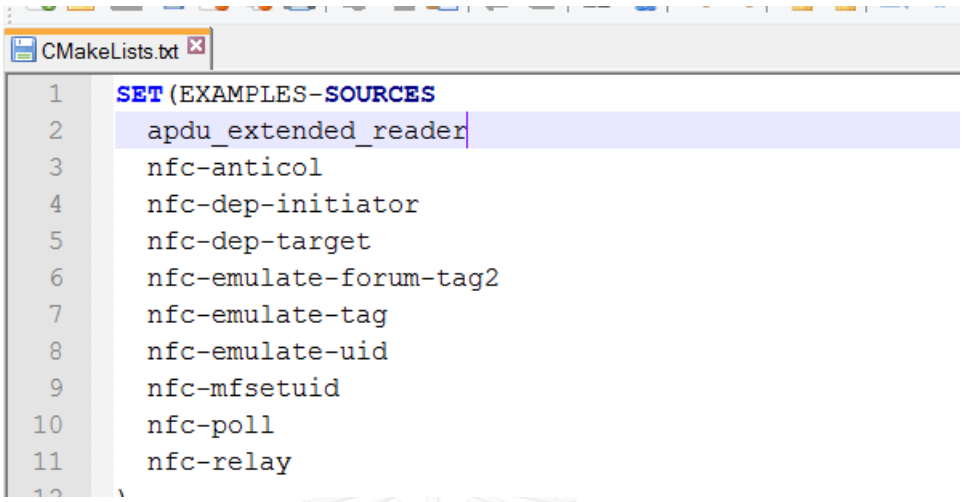

Figure 24: Adding adpu_extended_reader to CMakeLists.txt

- Starting to build and execute files with mingw32-make command at libnfc directory in cmd.exe and type “mingw32-make”
- After builded all files without any errors, now copy a file from “\libnfc-build\libnfc\libnfc.dll” to “\Windows\SysWOW64\”
- Creating an inf driver for ACR122 by execute a file from the “\libusb-win32-bin-1.2.6.0\bin\inf-wizard.exe”. Plug-in the ACR122 into the computer via USB port in order to inf-wizard may detect the device. If the NFC reader does not connect to computer so it does not have in the list as Figure 26. After completely made, this driver will be an unsigned driver.

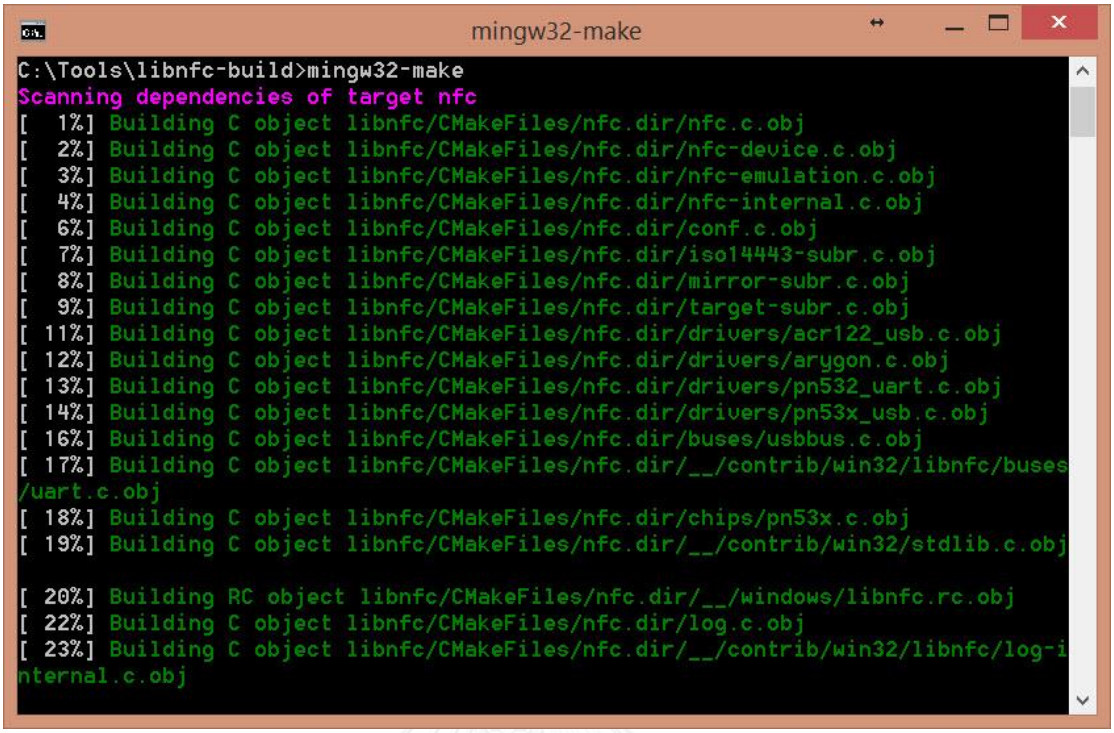

Figure 25: Building the files in cmake

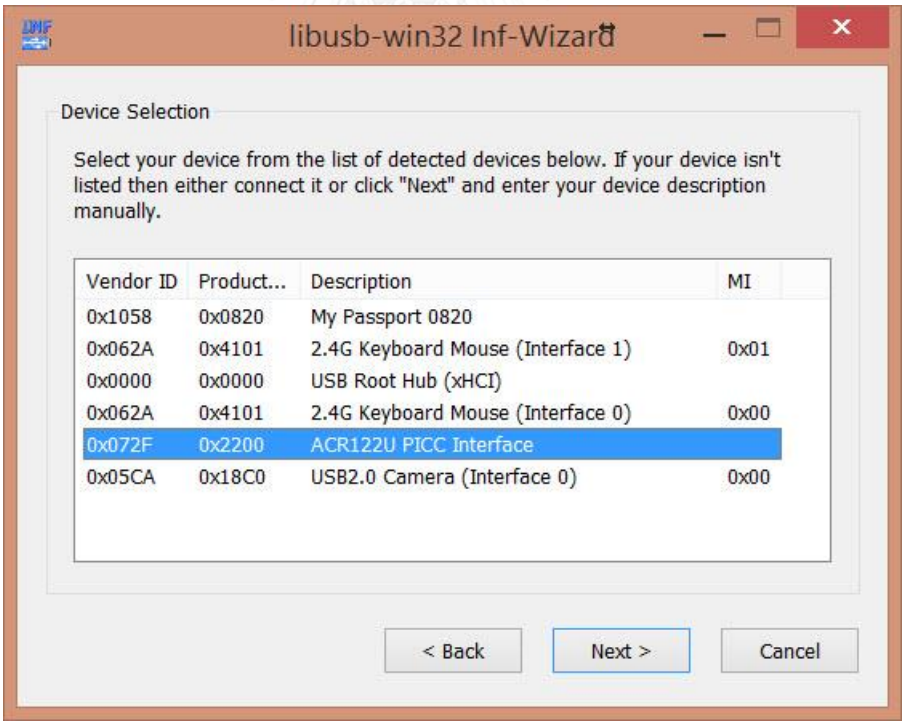

Figure 26: Creating the inf driver file by inf-wizard

- 64-bit Windows cannot be installed the unsigned drivers. This is a matter of above inf ACR122 driver. To fix it, users need to disable the driver signature verification on the Windows first. This trick work for Window 8

and 10. Hold down the “SHIFT” key which clicking the Restart button. After rebooted, the Boot Manager shows up. Go to “Troubleshoot> Advanced options> Startup Settings”. Then click on Restart button to return to edit the “Startup Settings” and then disable the driver signature enforcement.

- After disable the driver signature, now install inf file of ACR122 driver by right click on ACR122U_PICC_interface.inf and then getting installed and agreed with unsigned driver warning installation message.

Step 13: Updating driver in device manager (DM). By default, Window installs a driver for ACR122 and it names Microsoft Usbccid Smartcard Reader (WUDF) as Figure 27. If users have install a driver from the ACS website (Advanced Card Systems), please remove and roll back driver from window. Update then WUDF to ACR122U interface as Figure 28 and the driver becomes libusb-win32. as Figure 29.

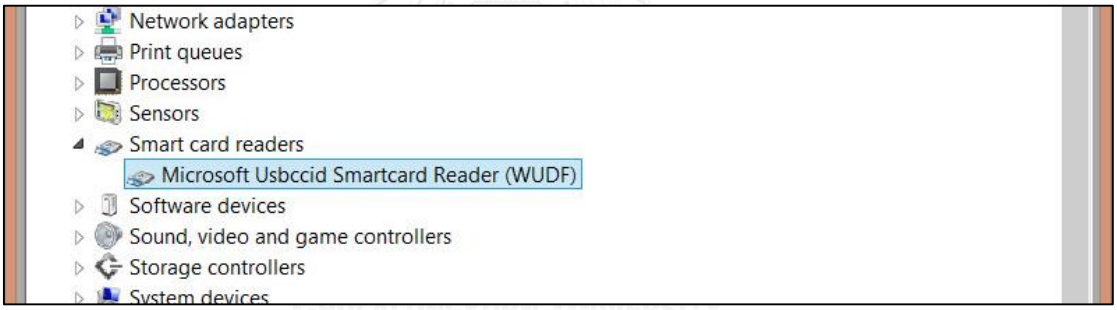

Figure 27: Microsoft Usbccid Smartcard Reader driver in DM

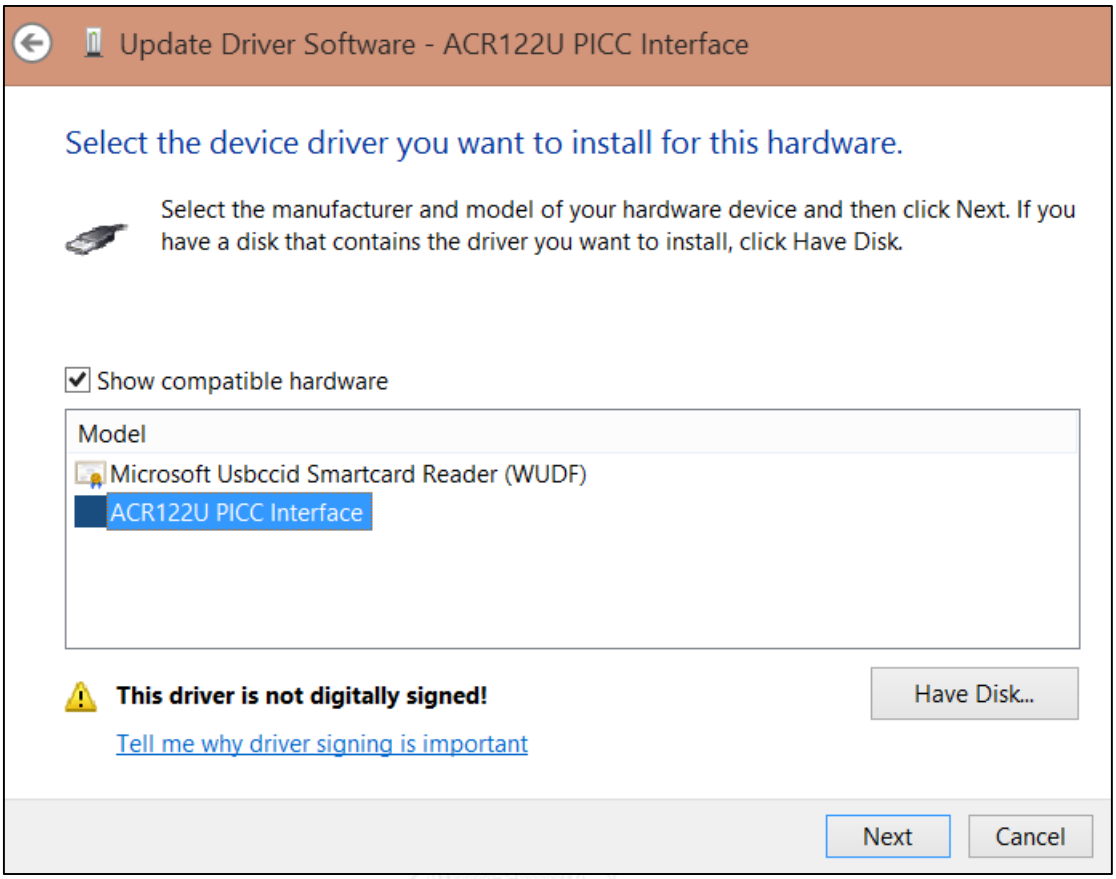

Figure 28: Updating from WUDF to ACR122U PICC Interface

Figure 29: ACR122U PICC Interface from libusb-win32

- Finally, the test-setup for ACR122 is completed. Next, we may run the file "apcu_extended_reader" from \"libnfc-build\examples\" in command prompt.

The result is shown in Figure 30. From this illustration, it was observed that the extended work from libnfc open library may process following the specification in [34] using ISO/IEC 7816-4 [31]. Firstly, apdu_extended_reader checks the reader over USB port. If NFC reader is compatible with it, a message shows the name of device and it

jumps to detect a cards. At this state, reader is waiting for a card (Type 4 Tag). Secondly, tapping the Galaxy Note 4 phone, while running the NFC application Figure 31, then reader detects and shows the next result (in Section 3.3.1):

a. First of all, NDEF Tag Application select procedure starts and sends C-APDU [00, A4 04 00 07, F0, 39, 41, 48, 14, 81, 00, 00], then the card respond [90, 00]. With this response, it means this procedure is successful then go to next step.
b. Secondly, the CC file select process by sending a C-APDU [00, A4, 00, 0C, 02, E1, 03] and its response is [90, 00].
c. Next step, CC file read procedure starts by transmit C-APDU [00, B0, 00, 00, 0F]; and the card replies that [00, 0F, 20, 00, 3B, 00, 34, 04, 06, E1, 04, 00, 32, 00, 00, 90, 00].
d. Thereafter, [00, A4, 00, 0C, 02, E1, 04] is sent and [90, 00] is back.
e. NDEF Length state sends C-APDU and gets a data [00, 33, 90, 00].
f. Lastly, after NDEF data using ReadBinary C-APDU [00, B0, 00, 00, 0F}, the gotten data is [00, 1F, D9, 01, 18, 02, 54, E1, 04, 42, 4D, 30, 6D, 6E, 79, 4D, 48, 66, 77, 66, 2B, 48, 4F, 6E, 45, 61, 32, 6D, 42, 77, 77, 3D, 3D, 90, 00].

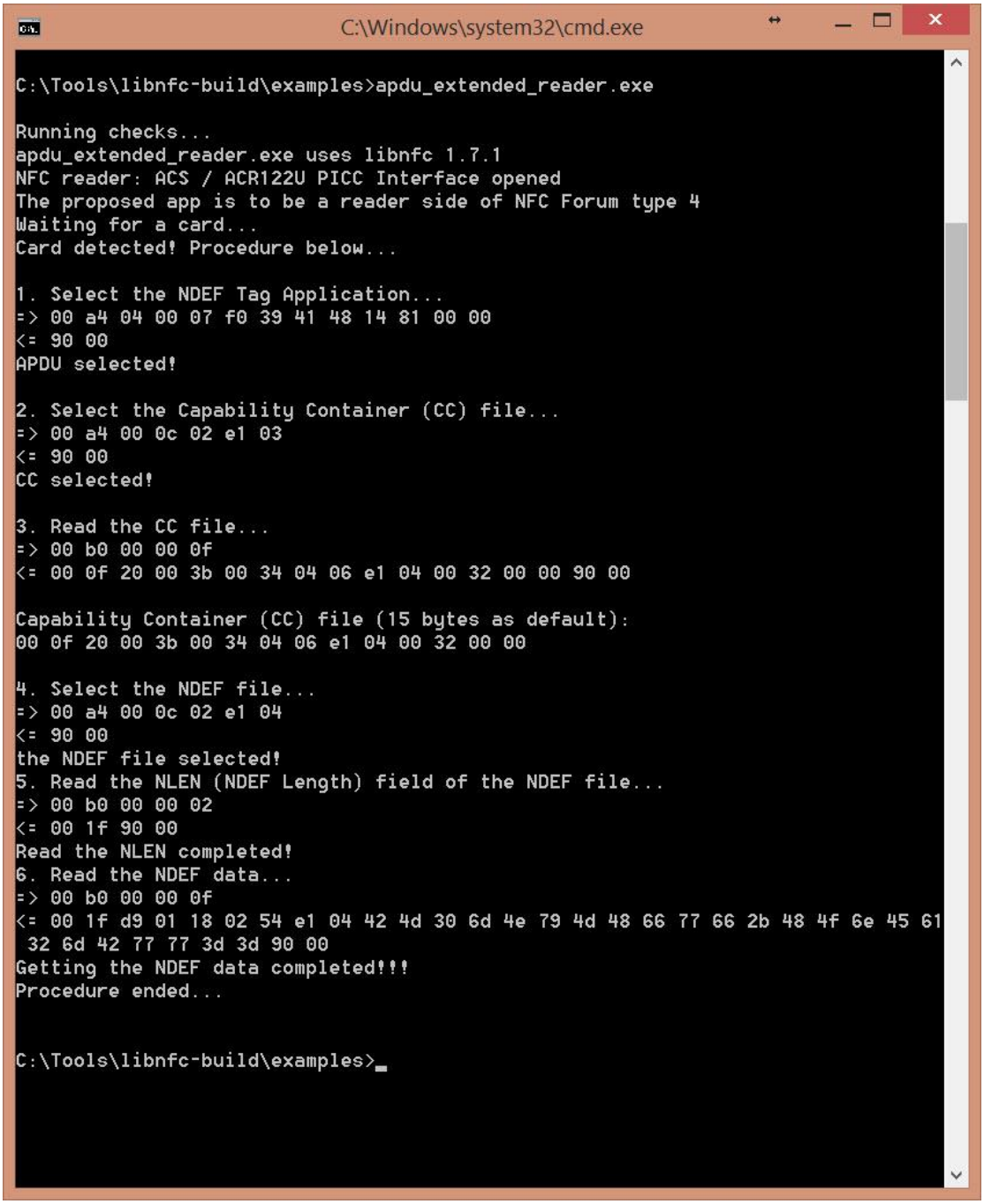

Figure 30: Result of Android app with apdu_extended_reader via usb on 64-bit windows

The first 2 response's characters of NLEN read and NDEF data read is [00, 1F] refers to the size of the data (size is 31 in decimal). After the File ID [E1, 04] in char [7] and char [8] and before the end of value of [90, 00] is the main encrypted passcodes in the Android application. As a result, this application is developed successfully how to move the key management to Android smartphone without a server generating a

passcode or key. In addition, the test-setup on Windows platform over USB is a way for other researchers using Windows to develop NFC on

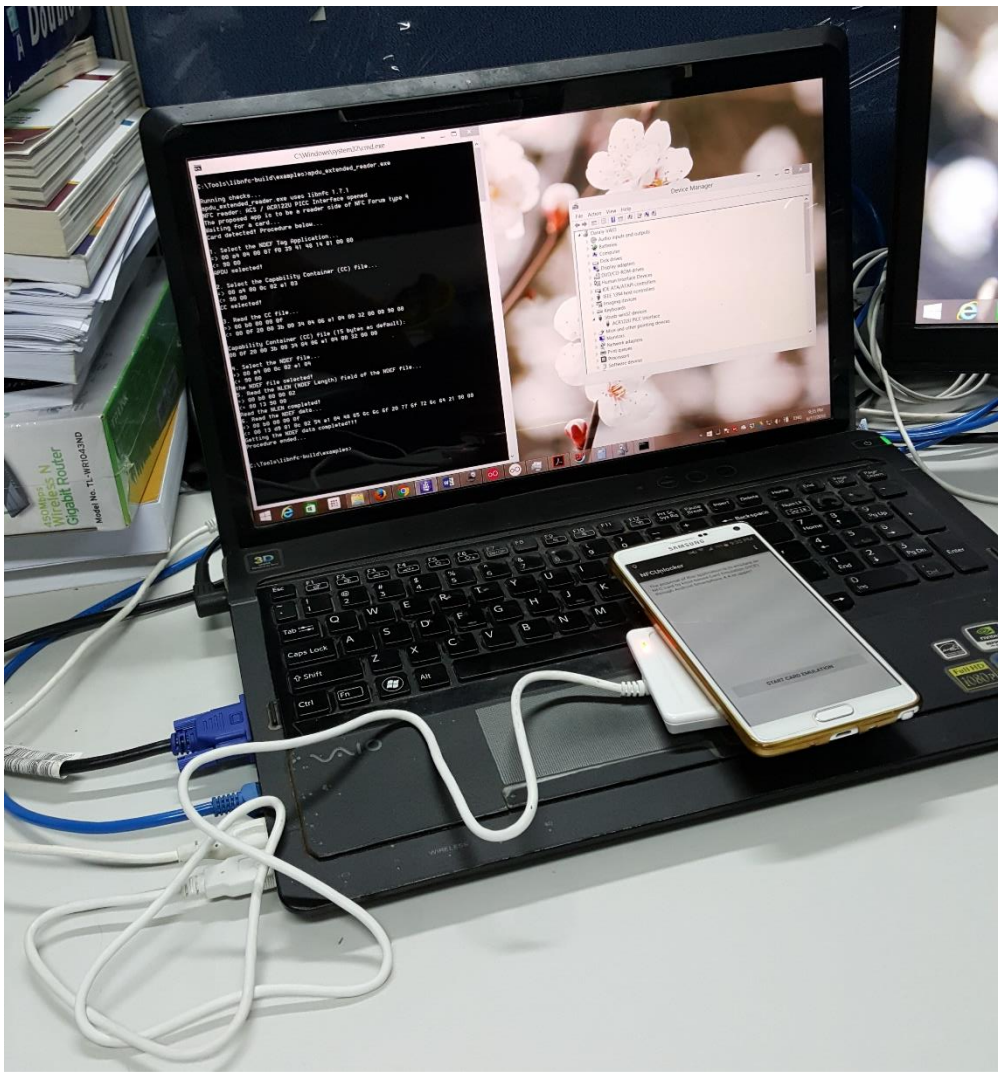

Figure 31: Testing ACR122U with Android applciation using USB port

4.3 Test-setup with NFC Shield using Arduino

Before this experiment occurs, make sure that Android application side works successfully while running HostApduSevice by using ISO/IEC 7816-4. According to Section 4.2 the Android NFC app is effective. Then this Section proves the test result of the NFC application with NFC shield connecting to Arduino Uno (in Figure 32) using developed library in Section 3.3.1.

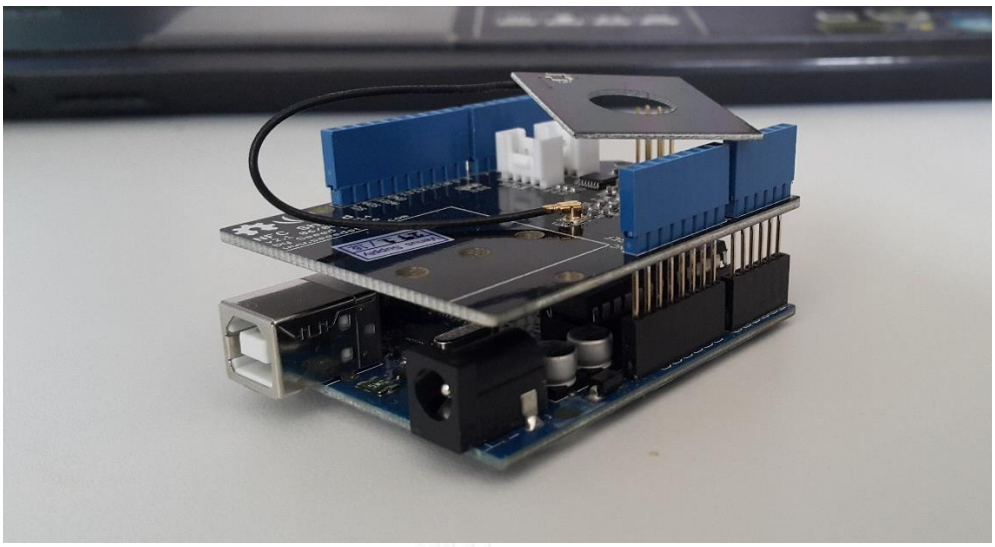

Figure 32: Arduino Uno with NFC Shield V2.0

Connect the Arduino board to computer via USB and then go to Arduino application and upload the developed library from Section 3.3.1. Through the Serial Com. From tool of Arduino program, it show the chipset type of NFC shield, the firmware of the shield and waiting for a card. Tap the Android smartphone running NFC application to the antenna of the shield then getting the result on computer. This library is built to get the data and return the main data as output in Figure 33. After got the response, the extract data from the NDEF data is [42, 4D, 30, 6D, 4E, 79, 4D, 48, 66, 77, 66, 2B, 48, 4F, 6E, 45, 61, 32, 6D, 42, 77, 3D, 3D] in hexadecimal. This data is the encrypted passcodes from the Android HCE application to NFC shield's library using Type 4 Tag Specification over ISO/IEC 7816-4. Thus, the built library is effective to work for Arduino board with NFC shield and based on high standardized protocol.

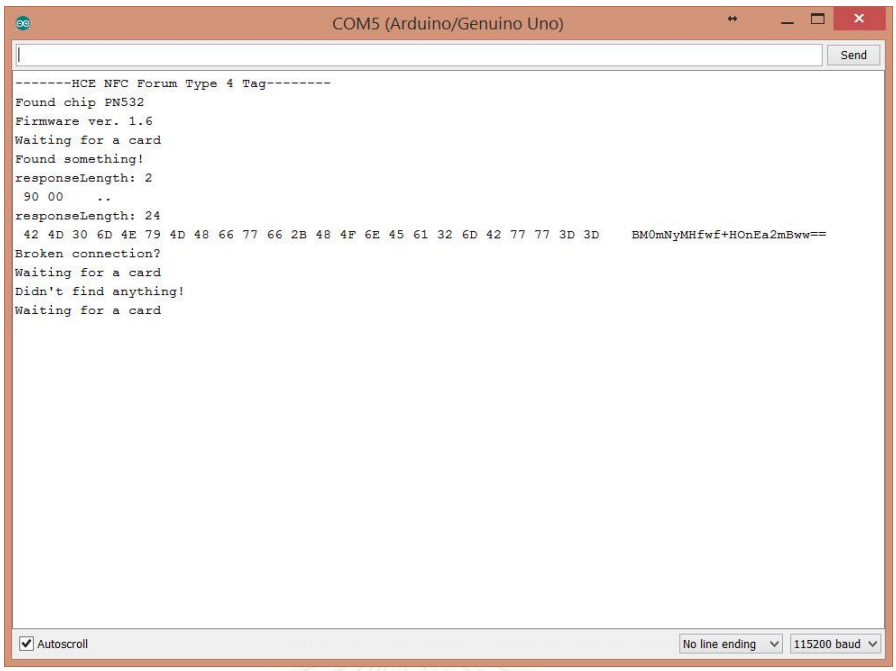

Figure 33: Result of Android application talking with NFC shield through Serial com.

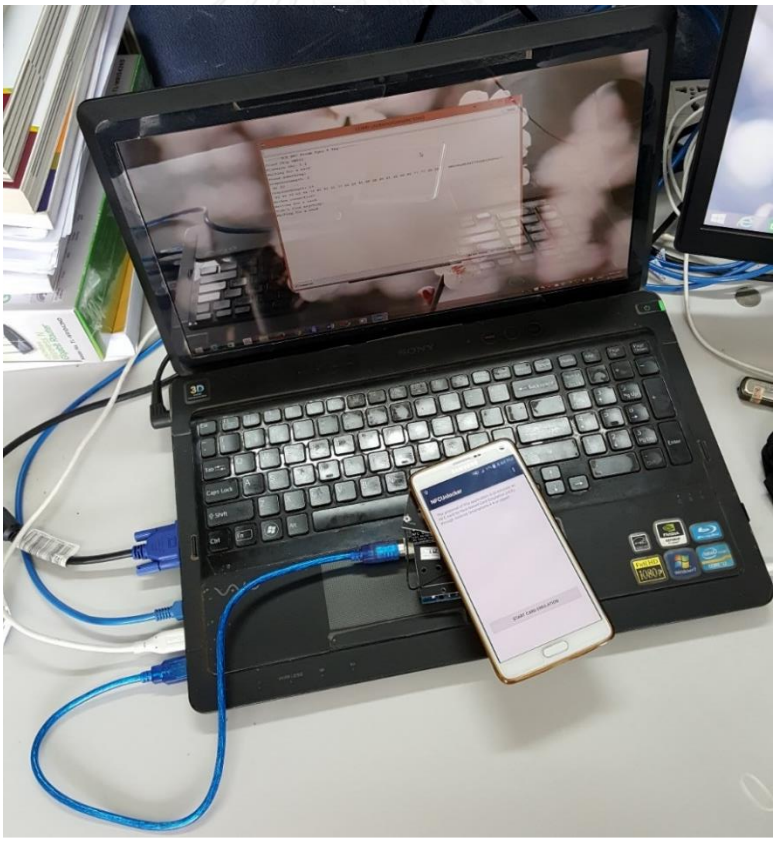

Figure 34: Android application with NFC shield using developed library

4.4 Comparison of process time and analysis

Sending and receiving data over the built library are not enough to perform its effect, but the duration of processing time of APDU is also important. This duration is for the developers to compare their method with other existing methods and to know that which one is optimized and better. Starting from an emulated card detecting, the time is counted to the all procedure of the exchanging data finished and 100 times of test have done. The Figure 35 is demonstrated the duration of each time testing built ISO 7816-4 library for Arduino. The maximum duration is 130ms and its minimum is 78ms. From this result, the average spending time is only 104.35ms.

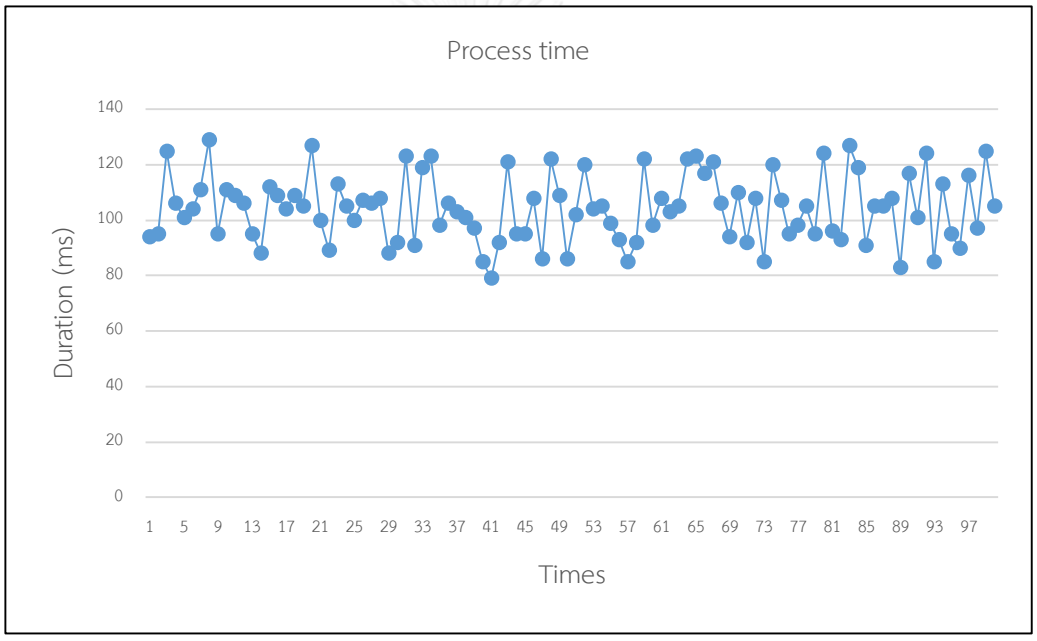

Figure 35: The respond time of APDU exchange over ISO/IEC 7816-4

From [29] using HCE over ISO 14443A for Arduino Uno, they attempted 15 testing times with their implementation of HCE application. The gotten average result is 261.33ms in the test.

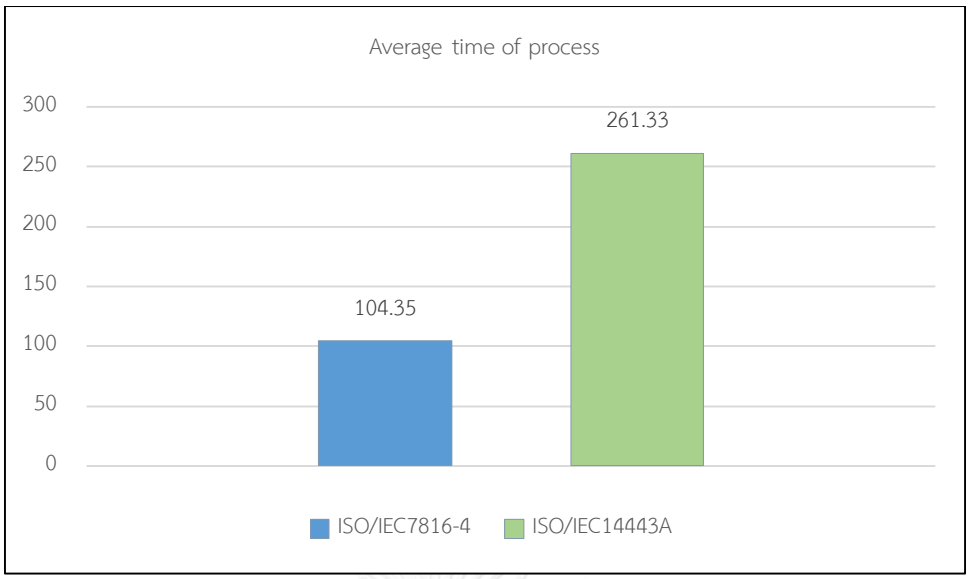

Figure 36: Average time of APDU between ISO7816-4 and ISO 14443A

Analysis of these two result from built ISO 7816-4 library and ISO14443A for Arduino are the same hardware used but Android phones are different. The process time of the built library spends less than current ISO 14443A library. It has two factors, external factors and internal factors. For external factors, it has position, distance, or obstacles. And the other factors, it has CPU's clock, firmware, delay of application, or interference signal. It also means that this built library is optimized and faster than current existing ISO 14443A library of Arduino. It is very effective and significant for developer using last ISO standard in their work. In these 100 times testing, no failure of the APDU detection occurred because the AID of smartcard is defined as constant value and it is cannot be changed after the application executed.

Chapter 5
CONCLUSION

The main objectives of this research has 2 parts. First part is to develop Android NFC emulation card application without a server and storing the all the passcodes in credential internal storage based on security-enhanced Android and additional encryption. Second part is to set the scene for other researchers using these test-setup for Windows platform over USB, extended work for APDU from open NFC library, and develop a library for NFC shield over ISO/IEC 7816-4 to test and debug in real NFC experiment. The implication of the developed library is also to add more facilities for users considering Type 4 Tag specification library over ISO/IEC 7816-4 for Arduino as the NFC reader for any purpose.

Hence, the signification of Host-based Card Emulation is deployed. For example, in the experiment in Chapter 4, the Android application have developed using Android HCE to emulate an NFC card and store the private data in their own way for easy access, and they have modified the credential data many times without matters. Unlike the most previous work using a secure element, the developers had to negotiate the manufacturer first before modifying and accessing the secure data. Also the reader side, reader had to get the privilege to gain the data from the secure element.

To enhance this work better with current completed work, in the future work, it is recommended that application security using password should be replaced by cloud logging seeing as hackers could decrypt the scrambled data to original version. In case of stolen phone, users are capable to erase the private data before the phone is attacked into the security-enhanced mechanism. Moreover, NFC library would be open source to all researchers and developers assess and improve to get stable version. Since the current library is support only reader mode and using some specifications of the ISO/IEC7816-4 protocol, emulating the card and P2P mode should be considered.

REFERENCES

[1] (12 Jan). *Smartphone OS Market Share, Q3 2014*. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[2] (15 Jan). *Android history*. Available: http://www.android.com/

[3] (15 Jan). *NFC-Forum*. Available: http://nfc-forum.org/

[4] Ed. (19 Jan). *NFC Implementation*. Available: http://www.nfcnearfieldcommunication.org/

[5] K. OK, V. COSKUN, M. N. AYDIN, and B. OZDENIZCI, "Current Benefits and Future Directions of NFC Services," presented at the International Conference on Education and Management Technology (ICEMT), 2010.

[6] S. Burkard. (2012, 15 Jan). *Near Field Communication in Smartphones*. Available: www.snet.tu-berlin.de/fileadmin/fg220/courses/WS1112/snet-project/nfc-in-smartphones_burkard.pdf

[7] A. Dudwadkar, A. Gore, T. Nachnani, and H. Sabhnani, "Near Field Communication in Mobile Phones," *International Journal of Engineering and Advanced Technology (IJEAT),* vol. 3, pp. 309-313, 2013.

[8] (10 Nov). *Near Field Communication*. Available: https://developer.android.com/guide/topics/connectivity/nfc/index.html

[9] L. Zhang. (2013, 01 Feb). *NFC Application Development on Android* with Case Studies*. Available: https://software.intel.com/en-us/android/articles/nfc-application-development-on-android

[10] (10 Nov). *Host-based Card Emulation*. Available: https://developer.android.com/guide/topics/connectivity/nfc/hce.html

[11] C. Saminger, S. Grünberger, and J. Langer, "An NFC ticketing system with a new approach of an inverse reader mode," presented at the International Workshop on Near Field Communication (NFC), 2013.

[12] S. M. Nasution, E. M. Husni, and A. I. Wuryandari, "Prototype of train ticketing application using near field communication (NFC) technology on android device," presented at the System Engineering and Technology (ICSET), 2012.

[13] T. Y. Teck, P. Sebastian, and V. Asirvadam, "Card Emulator for Door Access Using Android Platform," presented at the Control System, Computing and Engineering (ICCSCE), 2013.

[14] P. Urien, "A Secure Cloud of Electronic Keys for NFC Locks Securely Controlled by NFC Smartphones," presented at the Consumer Communications and Networking Conference (CCNC), 2014.

[15] C.-H. Hung, Y.-W. Bai, and J.-H. Ren, "Design and Implementation of a Door Lock Control Based on a Near Field Communication of a Smartphone," in *International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, Taipei, 2015, pp. 45 - 46.

[16] C.-H. Hung, Y.-W. Bai, and J.-H. Ren, "Design and implementation of a single button operation for a door lock control system based on a near field communication of a smartphone," in *5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, Berlin, 2015, pp. 260 - 261.

[17] J. Jacob, K. Jha, P. Kotak, and S. Puthran, "Mobile attendance using Near Field Communication and One-Time Password," in *International Conference on Green Computing and Internet of Things (ICGCIoT)*, 2015, pp. 1298 - 1303.

[18] M. O. Derawi, S. McCallum, H. Witte, and P. Bours, "Biometric access control using Near Field Communication and smart phones," presented at the International Conference on Biometrics Compendium (ICB), 2012.

[19] P.-L. Teh, H.-C. Ling, and S.-N. Cheong, "NFC Smartphone Based Access Control System Using Information Hiding," presented at the IEEE Conference on Open Systems (ICOS), 2013.

[20] HIDGlobal, "NFC and Smartphone Technology Drive New Opportunities for Smart Access Control," HID Global2013.

[21] M. Roland, J. Langer, and J. Scharinger, "Practical attack scenarios on secure element-enabled mobile devices," presented at the International Workshop on Near Field Communication (NFC), 2012.

[22] Christoph Busold, Ahmad-Reza Sadeghi, C. Wachsmann, A. Dmitrienko, H. Seudié, M. Sobhani, *et al.*, "Smart Keys for Cyber-Cars: Secure Smartphone-

based NFC-enabled Car Immobilizer," in *Proceeding of CODASPY '13*, 2013, pp. 233-242.

[23] W. Anwar, D. Lindskog, P. Zavarsky, and R. Ruhl, "An Alternate Secure Element Access Control for NFC Enabled Android Smartphones," *International Journal for Information Security Research (IJISR),* vol. 3, pp. 391-399, March/June 2013 2013.

[24] Consult-Hypersion. (15 Feb). *Host card emulation - why it matters*. Available: https://www.chyp.com/assets/uploads/Documents/2013/11/hce.pdf

[25] N. Saparkhojayev, A. Nurtayev, A. Dauitbayeva, and G. Baimenshina, "NFC-enabled Access Control and Management System," presented at the Web and Open Access to Learning (ICWOAL), 2014.

[26] (2014, 01 Mar). *Telcred - affordable and secure access control with NFC*. Available: http://www.telcred.com/index.php

[27] Google. (09 September 2015). *Chrome App NFC Library*. Available: https://github.com/GoogleChrome/chrome-nfc

[28] (25 Feb 2016). *NFC Library Development*. Available: http://nfc-tools.org/index.php?title=Libnfc:Windows

[29] R. S. Basyari, S. M. Nasution, and B. Dirgantara, "Implementation of Host Card Emulation Mode Over Android Smartphone as Alternative ISO 14443A for Arduino NFC Shield," in *Internaation Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, 2015.

[30] ISO/IEC, "International Standard ISO/IEC14443-4," in *Part 4: Transmission protocol* vol. 2015, ed, 2001.

[31] ISO/IEC, "International Standard ISO/IEC7418-4," in *Part 4: Organization, security and commands for interchange* vol. 2015, ed, 2005.

[32] J. Six, *Application Security for the Android Platform*: O'Reilly Media, 2011.

[33] (25 April 2016). *Security Enhancements in Android 4.4*. Available: https://source.android.com/security/enhancements/enhancements44.html

[34] "Type 4 Tag Operation Specification 2.0," vol. 2015, ed: NFC Forum, 2011.

[35] (15 Octorber 2015). *Libnfc:APDU*. Available: http://nfc-tools.org/index.php?title=Libnfc:APDU_example

[36] *TDM-GCC bundle*. Available: http://tdm-gcc.tdragon.net/

[37] *libusb*. Available: https://sourceforge.net/projects/libusb-win32/

[38] *Perl Compatible Regular Expressions (PCRE)*. Available: http://www.pcre.org/

[39] *CMake*. Available: https://cmake.org/

[40] *Doxygen*. Available: www.doxygen.org/

[41] *libnfc*. Available: https://github.com/nfc-tools/libnfc

APPENDIX

Appendix

List of Publications

Chan Daraly Chin and Watit Benjapolakul, “NFC-enabled smartphone application development to hide 4-digits passcode for access control system,” *in International Electrical Engineering Congress 2016 (iEECON 2016)*, Chiang Mai, Thailand, and published in Journal of Procedia Computer Science, vol. 86(2016), pp. 429-432.

VITA

Chan Daraly Chin was born in 1990 in Phnom Penh, Cambodia. He got his bachelor's degree in Electrical and Electronic Engineering from Institute of Technology of Cambodia in 2013. Currently he is a master's degree student at Electrical Engineering Department, Chulalongkorn University. His research interests include: Mobile Application Development, Network Protocol, and Telecommunication Networking.