UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

HCE Solution For Offline Transactions

Author:
D H T Punchihewa

Supervisor:
P.V.K.G Gunawardana

A thesis submitted in fulfillment of the requirements for the degree of Master of Information Security

in the

University of Colombo School of Computing

September 5, 2018

Declaration of Authorship

I hereby declare that the thesis is my original work and it has been written by me in its entirety.
I have duly acknowledged all the sources of information which have been used in the thesis.
This thesis has also not been submitted for any degree in any university previously

---

Tharanga Punchihewa

September 5, 2018

Abstract

Electronic smart card payment are widely used in transport payment systems these days. During last few years most of transport operators have started to introduce smart card payments. Most of the existing systems accept offline transactions. The key feature of smart card payments is storing data in the smart card. Smart card has embedded integrated circuits which can be used to store data securely and to communicate with terminal application via Near-field communication(NFC).

The objective of this research is to create a proof of concept(PoC) to replace existing smart cards by Android HCE application. HCE assumes that any data stored on a android device is vulnerable and therefore restricts the storage of sensitive data to host or "cloud" databases. Usually HCE provides more simple and less secure way for payment transaction. So this research focus on increasing the security of the stored data in mobile application and make payments without connecting to back-end services.

I have selected Desfire EV1 as smart card to emulate on android device. Desfire EV1 card has complex file structure and security features with compare to other available smart cards. Server system is used to increase the security of emulated virtual card in Android device. HSM simulator is used to store keys and perform crypto operations.

We present full design and implementation of the system architecture and security architecture involving with the project. We also provide a comparison of implemented system with the existing smart card solution. Android base terminal was used to simulate this bus ticketing terminal application. This terminal application uses Desfire AV2 SAM to store master keys of the virtual card and perform crypto operations to prepare PICC APDUs.

Acknowledgements

I am using this opportunity to express my gratitude and deep regards to my supervisor Dr.P.V.K.G Gunawardana for his guidance, monitoring and encouragement given. throughout the course of this thesis.

I would also like to express my appreciation to my project proposal panel and interim panel Dr. Chamath Keppatiyagama and Dr. Ajantha Athukorala for their valuable advice and guidance.

I hereby thank all my family members for the support which they are giving me. I would say I'll never achieve this great goal without their support. Finally I would like to thank my work place SilverLeap Technologies (Pvt) Ltd for the experience I gained from there, which inspired me to do this research.

Contents

List of Figures

List of Tables

List of Abbreviations

AID	**A**pplication **I**d
APDU	**A**pplication **P**rotocol **D**ata **U**nit
API	**A**pplication **P**rogramming **I**nterface
CMAC	**C**ipher-based **M**essage **A**uthentication **C**ode
HCE	**H**ost **C**ard **E**mulation
HSM	**H**ardware **S**ecurity **M**odule
MNO	**M**obile **N**etwork **O**perators
NFC	**N**ear **F**ield **C**ommunication
PICC	**P**roximity **I**ntegrated **C**ircuit **C**ard
PoC	**P**roof of **C**oncept
POM	**P**roject **O**bject **M**odel
SAM	**S**ecure **A**pplication **M**odule
TA	**T**rusted **A**pplication
UID	**U**nique **I**dentification
USIM	**U**niversal **S**ubscriber **I**dentity **M**odule
WAR	**W**eb **A**Rchive

Chapter 1

Introduction

1.1 Overview

Most people have mobile phones today. According to statics of mobile phone usage, the number of mobile phone users in the world are expected to pass the five billion mark by 2019 [14]. Initially mobile phones were primarily used to get calls and send text messages. Latest mobile phones have more features such as internet access, Bluetooth, Camera, NFC etc. Smart phones allow installing third party applications on the mobile device. Therefore new features are used for various purposes in the third party applications.

One of the latest technologies in the mobile devices is Near-field communication (NFC). NFC is a set of communication protocols that enables two electronic devices to transfer data. NFC protocols establish a generally supported standard. Each NFC enable mobile phone can work in three modes such as NFC reader/writer, NFC peer-to-peer and NFC card emulation.

Many android mobile devices that offer NFC functionality already support NFC card emulation. Android 4.4 introduces an additional method of card emulation that does not involve a secure element, called host-based card emulation (HCE) [7]. HCE allows the emulation of a contact-less smart card.

1.2 Objectives

The main goal of the project is to create a proof of concept that existing near field communication (NFC) payment solutions can be replaced with host card emulation (HCE) android application which has additional security features. HCE assumes that any data stored on a mobile application is vulnerable. Usually HCE provides more simple and less secure way for payment transaction. Therefore this research focus on increasing the security of the stored data in a mobile application.

The objective of the project is to demonstrate the use of NFC enabled phones in HCE mode as a suitable option for travelers to use as secure, reliable and offering suitable performance in daily usage. The project will focus mainly on a stored value based use.

Other than the main objective of the research project, This application can be used as access control card, privilege card, one time ticket and ID card.

1.3 Motivation

Some countries wildly use smart cards for transport ticketing. In some countries only the card payments are accepted in the public transport system. Since cash tickets are not allowed non-nationals have to face difficulties when using public transportation system. Not only that but there also other disadvantages in card payments such as,

- Complexity of the card personalization.
  Subscriber has to go to purchasing centers to purchase card. Service provider has to personalize the card with subscriber personal information. Therefore service provider has to keep a separate terminal with personalization application.
- Complexity of the card top-up process.
  Subscriber has to go to top-up centers to top-up the card wallet. There is no way to top-up the card wallet themselves.
- No recovery process.
  If the card is lost, there is no way to recover card wallet. Card owner can not transfer the card balance to a new account and can not block his lost card by himself.

Most of the above issues can be addressed by using a card emulated mobile application.

1.4 Scope

In this POC I selected to emulate Desfire EV1 card. Desfire EV1 card has four types of commands namely security related commands, Desfire commands, application level commands and data manipulation commands. Each command type has list of commands. I have emulated a few selected commands from above list. Desfire EV1 card has three types of communication modes namely plain communication, secured by MACing, and full encrypted communication. Fully encrypted communication is selected to emulate the virtual card.

Management system and subscriber portal are used to manage the merchants, subscribers and terminal application.

1.5 Deliverables

As mentioned in the section 1.4, mobile application with emulated card is implemented. Sample bus ticketing POS application is implemented to verify the mobile application. This POS application supports both physical card and the virtual card of the mobile application. Back-end system is implemented to manage the mobile application and the POS application. HSM simulator module provides secure key storage and a set of cryptographic operations for the card and treminal personalization process.

1.6 Thesis Structure

The second chapter will be the related work studies. This chapter includes the details of the research and commercial projects which have been done by the others. The third chapter will describe the system architecture of the POC system. The fourth chapter will describe the security architecture of the POC system. The Fifth chapter will describe details of the the implementation of the POC. The sixth chapter will describe the evaluation of the research. Final chapter will describes about future works and the general discussion.

Chapter 2

Literature Review

2.1 Near Field Communication - NFC

NFC stands for Near Field Communication. Bluetooth and Wi-Fi seem similar to near field communication on the surface. All of these technologies allow to exchange data between two devices. NFC operates within a radius of about 4 cm and provides a wireless connection between two devices. NFC transmits or receives data via radio waves.

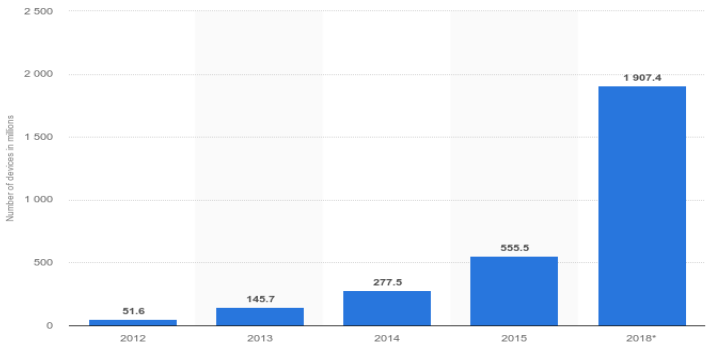


FIGURE 2.1: Number of NFC-enabled mobile devices ([15])

Above figure shows the statics of number of NFC enabled mobile devices from 2012 to 2018. Number of the NFC enabled mobile devices are growing. Looking toward the future, it's possible that NFC chips could be used to replace every smart cards.

All NFC device categories support three communication modes.

- NFC reader/writer

  This mode of operation within NFC near field communication allows applications to transfer data in an NFC Forum-defined message format. This communication mode is not secure.

- NFC peer-to-peer

  In Peer-to-Peer mode, two NFC devices can exchange data. Android Beam technology uses this mode for its operations. Peer-to-Peer mode is standardized on the ISO/IEC 18092 standard.

- Card Emulation Enables NFC-enabled devices to act like smart cards, This feature can be used to perform transactions such as payment or ticketing.

2.2 Host Card Emulation

2.2.1 What is HCE

Host Card Emulation(HCE) is a contemporary technology that used to emulate smart card on NFC enable devices. Smart card emulation is done using a software. In the context of mobile payments, it is used to complete transactions via NFC. In this software solutions sensitive data are stored in a secure element(SE). This SE can be a microchip that is embedded into the NFC device or cloud solution which is outside the mobile phone. In this kind of software there is no dependency with mobile operator.

2.2.2 Benefit of HCE

- Reduced complexity for application developers and opens up NFC capability for new applications.
- Easy and flexible card provisioning to mobile device.
- No dependency on MNO or SE owner.

  Payment providers can create NFC-enabled applications without seeking permission from the MNO or mobile phone manufacturer.

2.2.3 Limitations of HCE

- No hardware secured storage of data and credentials on device.

  If the android device is rooted, attacker can access all the data stored in the HCE application, including sensitive data such as keys, wallet information. Mitigating HCE related risks can be done in two different ways,

  - Provide more secure location for storing sensitive data.
  - Apply security mechanisms to make existing location more secure.

- Dependent on OS capability.

  The security of HCE is dependent on the security of the underlying Android OS. If there is a vulnerability in the android OS, enabling data within the HCE application to be accessed, replicated or manipulated.

- Alternate security considerations for HCE can increase back-end complexity.

2.2.4 Card Emulation with SE

A Secure Element (SE) is a microprocessor chip which can securely hosting applications and storing confidential and cryptographic data. The secure elements can exist either on the chip (embedded), or in SIMs, so it's like using a smart card in your smartphone. The card to be emulated is provisioned into the secure element on the device through an Android application. The NFC controller in the device routes all data from the reader directly to the secure element.

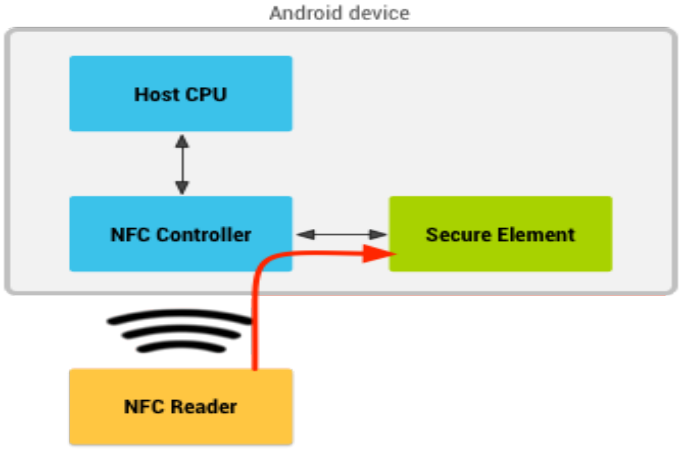


FIGURE 2.2: Card Emulation with SE

2.2.5 Card Emulation without SE

When an NFC card is emulated using host-based card emulation, 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.

2.2.6 Current Support for HCE

1. Google Android Kitkat4.4 and higher.
2. Blackberry OS 7 and higher.
3. Microsoft Windows 10.

2.2.7 Smart Card Solutions

Smart card has embedded integrated circuits and it can be used store data in securely and it communicates with terminal via NFC. In the micro payment and transport ticketing systems, Smart card is very popular. Visa and master has deployed new version in US in 2004-2006. But this solution is not compatible with existing fair collecting systems, though the MIFARE Standard card (Desfire EV1, Ultralight C, MIFARE Ultralight EV1, MIFARE Plus EV1,

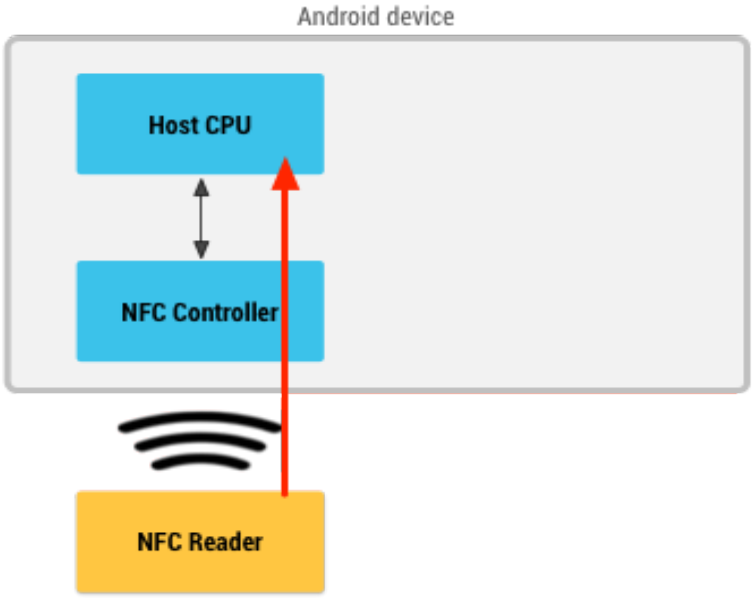


FIGURE 2.3: Card Emulation without SE

MIFARE Plus) from NXP Semiconductors has a considerable market share in the US and Europe[17].

Few of existing solutions which are used smart card are shown in table 2.1.

TABLE 2.1: Existing solutions [16]

Project	Locality	Usages
Freedom Card	United States	Travel card
Oyster card	United Kingdom	Travel card
Rabbit Card	Thailand	Micro Payment card
RTA	Dubai	Travel card,Parking card,Taxi Payment
Opal card	Australia	Travel card
Touch	Sri Lanka	Travel card, Fual card

2.3 HCE Solutions

Most of existing HCE applications support only online transaction. All of the existing application are using cloud based secure storage. They are not using

android device storage for the HCE application. Main disadvantages of the cloud based systems are shown below.

1. Transaction speed.

   Payment transactions require a quick response although the threshold may be somewhat lower (e.g. 400ms). HCE cloud-based NFC transactions face a considerable challenge here when compared to offline solutions.

2. Roaming and no data connectivity scenarios.

   Main disadvantage of the cloud based systems is internet connection should be there and it takes more time exchange commands between reader and the server. When the user has disabled data connectivity to avoid roaming charges, user can not perform the transactions.

2.3.1 Rambus Ticketing Solution

Ranbus[11] has used HCE technology to buy bus tickets. The Rambus HCE software enables mobile NFC transactions to be made by storing and accessing credentials in a remote environment rather than on the mobile device.

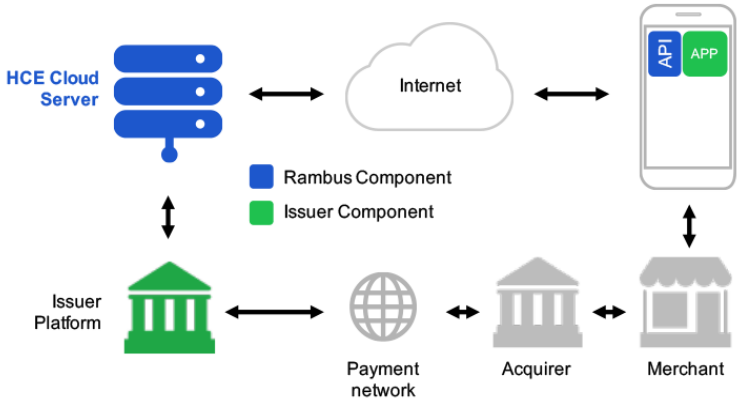


FIGURE 2.4: The Rambus HCE Solution Overview [13]

2.3.2 MasterCard HCE Pilot Project

MasterCard has announced the coming of new pilot programs in a move to support Host Card Emulation (HCE). The projects are aimed to enhance the payments experience on Android devices.

MasterCard Cloud Based Payments streamlines the deployment of mobile transaction services for financial institutions and providers. The new offering veers away from secure element (SE)-based technology, and relies on cloud processing coupled with HCE architecture [10].

2.3.3 Kazkommertsbank HCE Payment Solution

Customers of Kazkommertsbank (KKB) in Kazakhstan has launched a HCE application in partnership with Visa. Visa card issued by the Kazkommertsbank can be integrated with this solution to pay for goods and services. [9].

2.3.4 Fibank HCE Payment Solution

Customers of First Investment Bank (Fibank) of Bulgaria can now make host card emulation (HCE) based payments through its mobile banking application is using a digital card supplied by MasterCard. The service makes use of technology supplied by white label mobile wallet provider MeaWallet and HCE specialist SimplyTapp. Here also they allow only online transactions [3].

2.3.5 Commonwealth bank HCE Solution

Commonwealth Bank of Australia is working with MasterCard to use host card emulation to extend the availability of its mobile contactless payments service. G and D is the technology provider for the HCE service. Here also they allow only online transactions [4].

Chapter 3

System Architecture

3.1 Overview

The concept is essential to use a suitable equipped Near Field Communication(NFC) mobile phone in the place of a physical subscriber's card. All aspects of the architecture are implemented to enable smooth usage from a consumer perspective and minimal impact on the existing systems of transport application system.

Server side infrastructure is added to the ecosystem to provide remote management, provisioning and financial risk management, to handle high volumes and quantity of users.

The application can be downloaded from the subscriber portal. User can register to the system by registration form in android application. Then after the successful top-up transaction, HCE android application will be activated. This top-up can be done either at merchant top-up machine or android online fund transferring.

Trusted application (TA) is the android library which handle all security related functions and storing sensitive data in android HCE application. The card layout and the personalize data of the virtual card is loaded to the trusted application (TA) module of the mobile application. After the every successful transaction TA module is updated and save in securely.

The following diagram is a high level representation of the solution (Figure : 3.1).

3.2 HCE Ticketing Solution

This HCE ticketing solution offers a unique virtual card solution.That uses a TA to provide strong security for storing monetary value. This application delivers significant improvements in ease of use, flexibility and performance over other methods of secure NFC ticketing.

This solution consists primarily of:

1. System Administrator Portal
2. Subscriber Portal
3. UID assigning

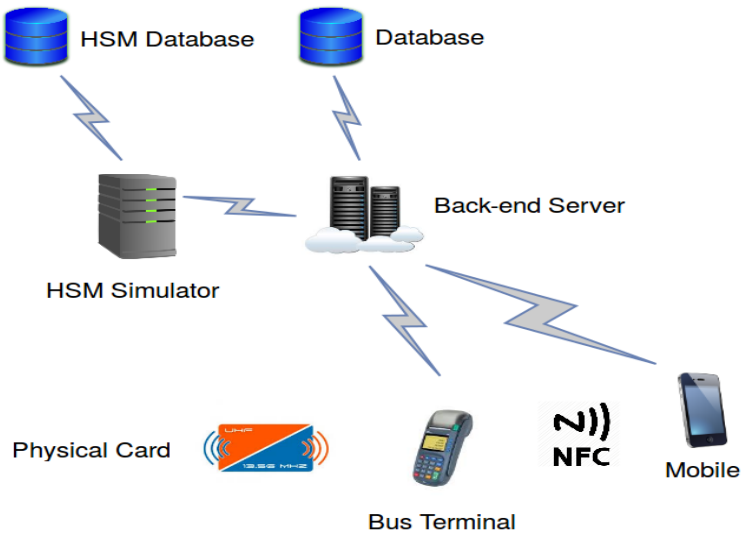


FIGURE 3.1: System Overview

4. HSM Simulator
5. Mobile Application
6. Terminal Application

For demonstration and proof of concept purposes I provided a validation terminal in order to fully verify the solution. This terminal application supports either HCE mobile application and the physical NFC card.

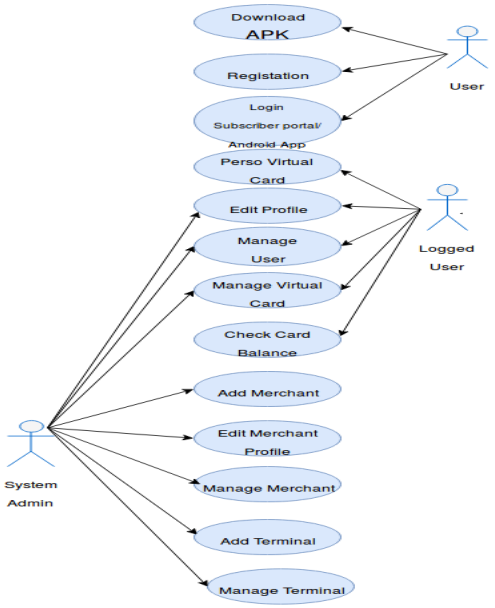


FIGURE 3.2: System Use Case Diagram

3.3 NFC Phones

HCE is a relatively new feature available on Android version 4.4 NFC mobile phones and above. It allows for contactless card emulation without the need for access to embedded Smartcard chips or NFC USIM card in the phone.

HCE allows APDU interactions with the Level 1 devices to be routed to an Android application rather than a smartcard in the form of an embedded secure element.

This approach has a number of significant advantages such as flexibility in software development, simplified card supply chain and speed of interaction. However it requires a new approach to security. So I have used TA to secure data of the virtual card.

The NFC application is a standard Android application that communicates with a Trusted application. TA module is packaged with the Android APK bundle.

When, the phone is used on a transport ticketing reader or Top-up reader the communication between the reader and the HCE Virtual Card TA and follows the reverse path for responses.

3.4 Server Software

This server software system consists of a set of modules that together handle all the remote management need for deploying and managing large numbers of mobile phones with HCE virtual cards. All the server software is developed in enterprise Java.

Functionally, the server side software will provide the following capabilities:

1. Creation, download and remote Management of HCE virtual cards into mobile phones
2. User registration and management
3. Secure Remote Top-up of HCE virtual in mobile phones

The following diagram is a high level representation of the server solution.(Figure - 3.3)

3.4.1 System Management Portal

System administrator portal is a web-based application designed to administrate components and resources that are involved in the HCE Application Management System. This portal has following functions.

1. User Management
   (a) New users can be created and permissions can be assigned for the system users.

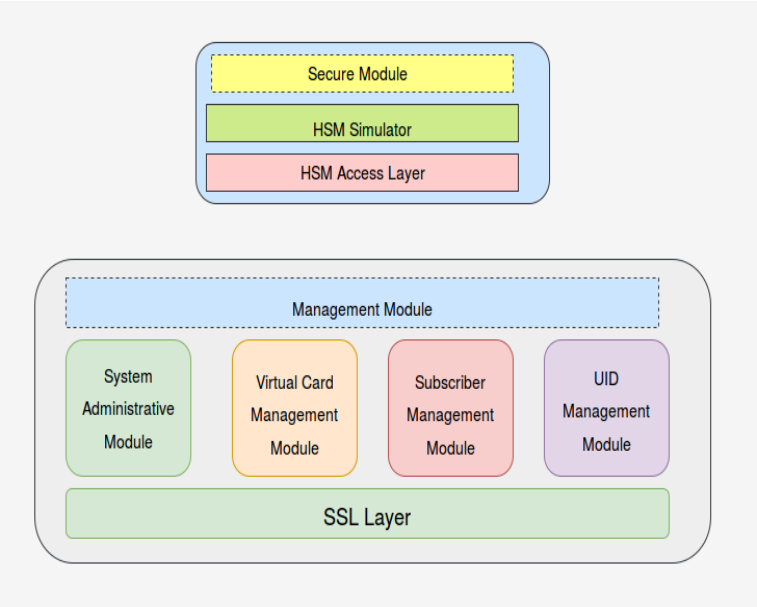


FIGURE 3.3: Server System Overview

(b) Existing users can be modified.

(c) Existing user's details can be viewed.

(d) Existing user can be deleted. This option terminate the all user access permission to the system.

(e) List view of the all system users.

2. Merchant Management

(a) Register new merchant.

(b) Existing merchant can be modified.

(c) Existing merchant's details can be viewed.

(d) Merchant can be deleted.

(e) List view of merchants. All above functions are allowed to do system admin and the system users.

3. Device Management

(a) Add merchant's device.

(b) Assign device to merchant.

(c) Revoke device from the merchant.

(d) Block device

4. Virtual Card Management

(a) Card Assigned Once subscriber registered to the system,Virtual card UID is temporally assigned to the registered subscriber.

(b) Permanently Assigned After the successful mobile application login,Temporally assigned virtual card UID is permanently assigned to the subscriber.

(c) Blocked After the virtual card blocked.Subscriber mobile application login is not allowed. So Subscriber can not use virtual card after the block.

5. Subscribers Management

   (a) Block subscriber's account.

   (b) View subscriber's profile.

6. Settlement Module

   Settlement module has following features.

   (a) View all transactions

   (b) Generate daily transaction report

   (c) Settle transactions

Login page (Ref - 3.4) and home page of the System Management portal (Ref -(3.5)) are shown below.

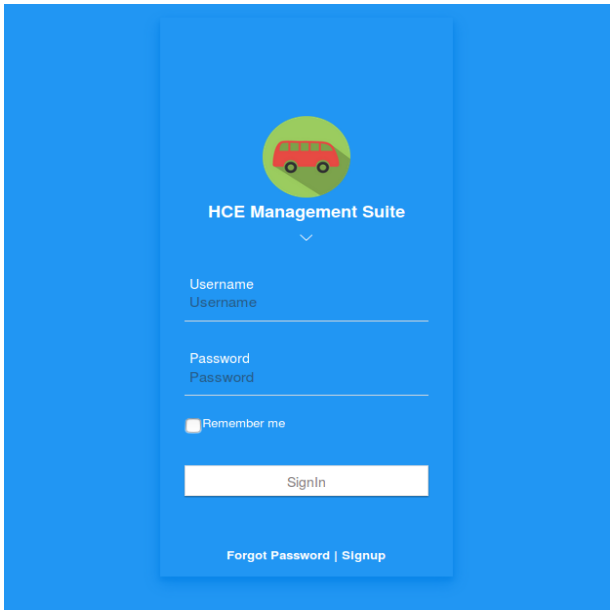


FIGURE 3.4: Login page of the management portal

Below services are implemented to manage android application.

1. Download merchant details to terminal

2. Upload transaction files to management system.

3. Download device details to terminal.

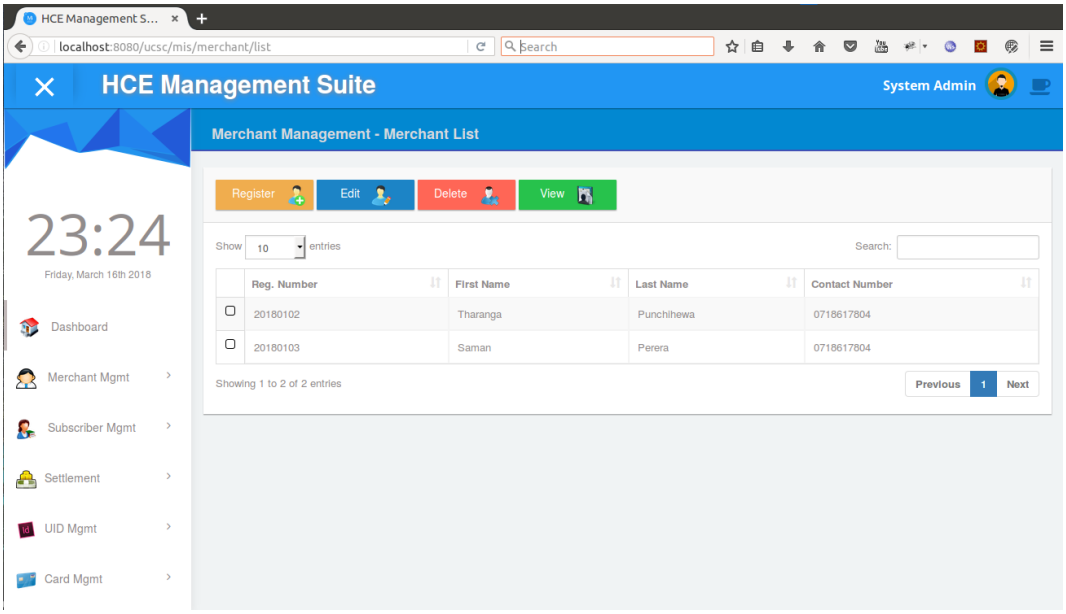


FIGURE 3.5: System Management Server - Merchant List

3.4.2 Subscriber Portal

Subscriber portal is a web based interface for subscriber interaction and account management. Each subscriber has a user account to access the subscriber portal. Subscriber can use following functions after the successful login to the portal.

1. Login
   Subscriber can be login to the subscriber portal with his user credential which is used to login mobile HCE application. After the successful login he can manage his account.

2. Edit user profile
   Subscriber can only register by HCE mobile application. In the registration screen system capture only some limited data. So after the successful registration subscriber can be login to the system and he can edit the his profile.

3. Check transaction details
   Terminal will upload all transaction to the management system. Subscriber can view his transaction details. History screen display below details.

   (a) Check-In/Check-out.
   (b) Pre Balance.
   (c) Post Balance.
   (d) Transaction Reference Number.
   (e) Transaction Time.
   (f) Merchant Name.

(g) Status

(h) Check last updated wallet balance

4. Manage Virtual Card
Subscriber can disable and enable the card any time and this information will be saved on the Management Module.

5. Below services are implemented to manage the android application.

(a) Subscriber login service

(b) Download user parameter service

(c) Upload user activity log

(d) Subscriber registration service

3.4.3 UID Management Module

Card unique identifications(UIDs) are generated by the back-end server. This 7 byte UID is assigned to the virtual card in the HCE application penalization process. This UID is used to generate diversified PICC keys of the virtual card. This UID is unique to the virtual card so unique PICC keys could be generated for each virtual card same as the physical card. In this module following functions are managed.

(a) Upload UID Bulk

(b) Assign UID to Virtual Card

(c) Delete UID

3.4.4 HSM Simulator

This module is a software base HSM simulator. This module will support for the secure key storage and services for the card penalization and online top-up. I used a desfire SAM(AV2) to simulate this module. In the card penalization process, This module is done following functions.

(a) Key diversification.

(b) Create full protection APDUs to send HCE mobile application.

(c) Use offline keys to secure server and Mobile application communication.

3.5 Mobile Application

Mobile application is a android application. Following functions are available in the mobile application.

1. Registration
   Subscriber can be registered to the HCE services by using mobile application.

2. Login
   Subscriber can be login to the subscriber portal and mobile application with his user credential. After the successful login, he can manage his account.

3. Check Transaction History
   Last ten transactions are list in the transaction which are performed by the virtual card wallet in history page.

4. Check Wallet Balance
   Balance of the virtual card wallet is displayed in the balance check screen.

5. Manage Virtual Card
   Subscriber can disable and enable the card anytime and this information will be saved on the Management Module.

6. Check Transaction History.
   Subscriber can check his transaction history which are perform by the virtual card. Last ten transactions are displayed in the transaction history screen.

Login screen (Figure - 3.6) , home page (Figure -3.7a) and card balance check screen (Figure -3.7b) of the android HCE mobile application are shown below.

3.6 Terminal Application

For HCE ticketing to work the bus terminal need to be modified to send an ISO SELECT AID command before sending other commands. The changes are comparatively small. This terminal application supports all functions for physical card as well as HCE mobile application. I have used a android base terminal to simulate this bus validator. This terminal application is managed by the management portal. This application has following functions.

1. SAM Activation

2. Check-In

3. Check-out

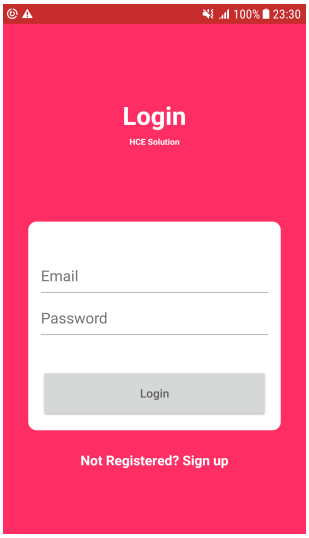
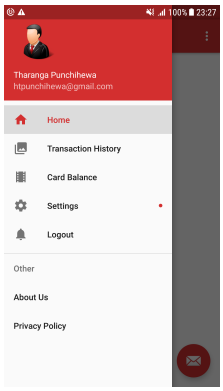
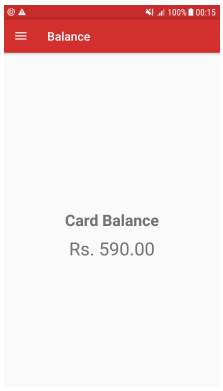


FIGURE 3.6: Mobile Application - Login Screen



(A) Menu And Home Page



(B) Card Balance Screen

FIGURE 3.7: Subscriber Mobile App Screens

4. Balance Check
5. Off-line Top-up

Screen of the terminal application is shown below.
APDU exchange sequence of the main usecases are shown below.(Figure : 3.9)

3.6.1 Software Changes on Terminal

NFC phones are designed to perform three types of basic NFC functions such as,

1. Card Emulation
2. Card reading
3. Peer to peer communication

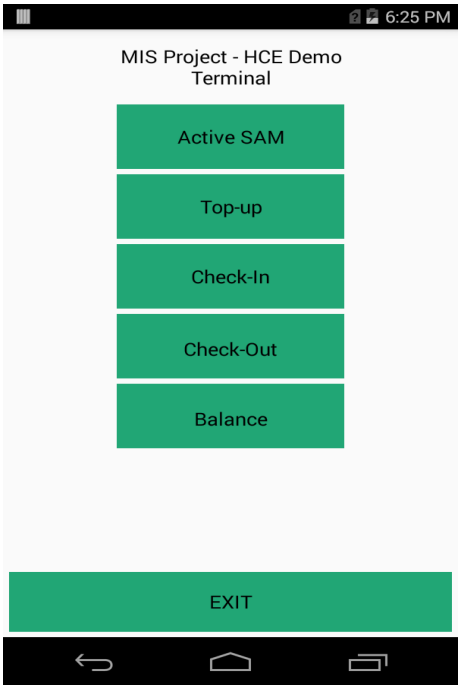


FIGURE 3.8: Terminal Application

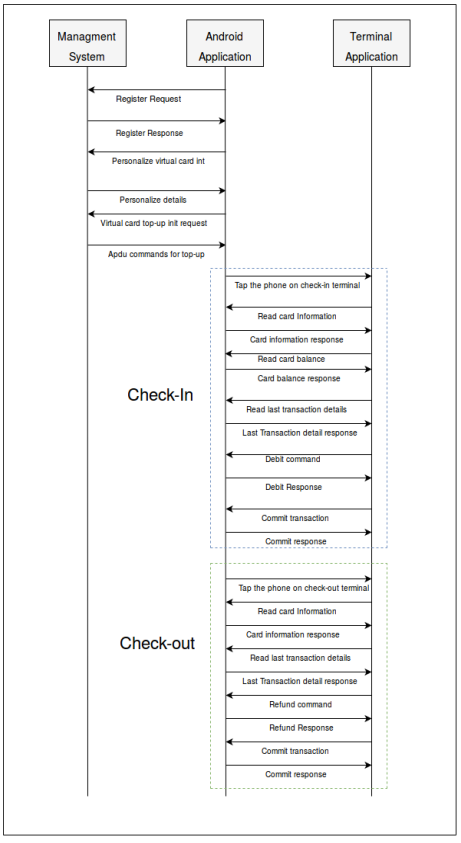


FIGURE 3.9: APDU Exchange Sequence

When emulating a card the fundamental design consideration is that the phone can emulate multiple NFC cards at the same time. These cards can

be of any type: payment, access control, transit, etc. Therefore there are specific mechanisms implemented to allow for this cohabitation of cards in the phone. Since the phone can conceivable be any card there is no specific UID that the phone can present to a reader, hence the use of Random UIDs in the anti-collision phase of the transaction set-up.

Selecting the HCE application

In order for the phone to know which application to present to the reader the reader must ask for the application explicitly. This is done using the ISO standard APDU, SELECT AID ("0xD2760000850100"), together with the Desfire Application AID. Note that this AID is the AID of the HCE Desfire card/application, not the individual files within the particular (This can be the standard Desfire AID defined by NXP or a pre-registered AID).

Once the phone has received this ISO SELECT AID the HCE application can be selected by the phone, and all subsequent APDU processing us unchanged from the existing standard card processing.

Read UID

The basis of the proposed approach is that when a mobile phone is placed in the field of an NFC reader the NFC reader will perform the standard ISO 14443 protocol steps and will arrive 'at the top' of the 4 layer stack with an active session with the NFC instrument, ready to issue a command APDU.

At this point the reader will not know anything about the instrument in the field other than the UID used in the anti-collision phase of the protocol stack (ISO 14443-level 3). So Terminal application should be changed to read UID from the GetVersion command. It gives the static UID and this UID can be used to key diversification process.

3.7 Use Cases

The following use cases will be supported in the POC project.

1. Download android application.
2. Subscriber registration.
3. UID assigning.
4. HCE Application Personalization.
5. Login to HCE Application.
6. Check Current Balance.
7. Check-In Functionality.
8. Check-out Functionality.

9. Top-up HCE Wallet.
10. Block HCE Wallet.

3.8 Tools and technologies used

1. Hibernate
   Hibernate is a persistence framework and it used to persist Java object in relational database. I have used hibernate version 4.0.2.RELEASE.

2. Spring
   Spring is an enterprise Java framework and I have used Spring version 4.2.0.RELEASE

3. MySql
   I have used MySql as database management system. The version of the MySql is 5.7.

4. Tomcat
   Apache Tomcat is used to deploy all the servers of this POC system. Build war file of each server module and copy those files into tomcat webapps folder. Version of the tomcat is 9.0.0

5. Maven
   Maven is to used to build java project.Maven allows a project to build using its pom file. All the dependencies are configured in this pom file.Version of the Apache Maven is 3.5.0

6. Gradle Gradle is used to build HCE android application and the terminal application. Gradle version 1.5.0 is used in this project.

Chapter 4

Security Architecture

4.1 Overview

The objective of the security architecture is to exhaustively specify the security features related to the HCE Virtual Card Solution. The solution depends on the HCE (Host Card Emulation) capability of modern NFC phones and plans to store the same physical card application structure on a DESFire emulation.

The following diagram is a very abstract representation of the overall architecture. (Figure - 4.1)

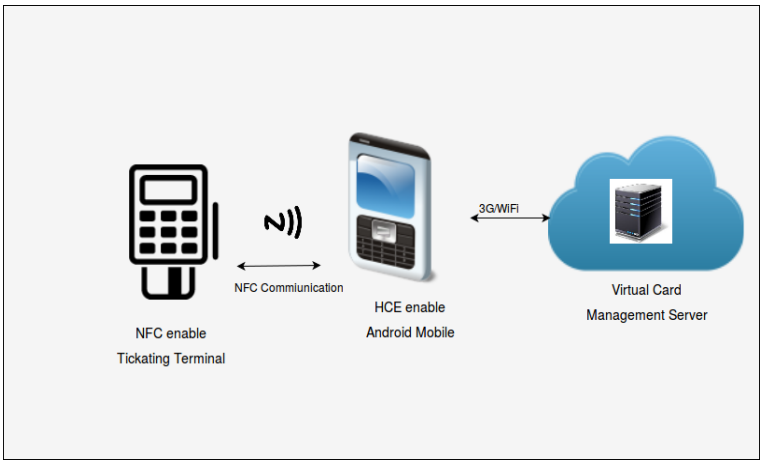


FIGURE 4.1: Abstract Representation of the Overall Architecture

4.2 Virtual Card

Objective of this project is emulate physical card in android device and this virtual card is based on DESFire EV1 physical card. Card layout structure of the Desfire EV1 and virtual card is shown below figure.

The following diagram is a very abstract representation of the card layout. (Figure - 4.2)

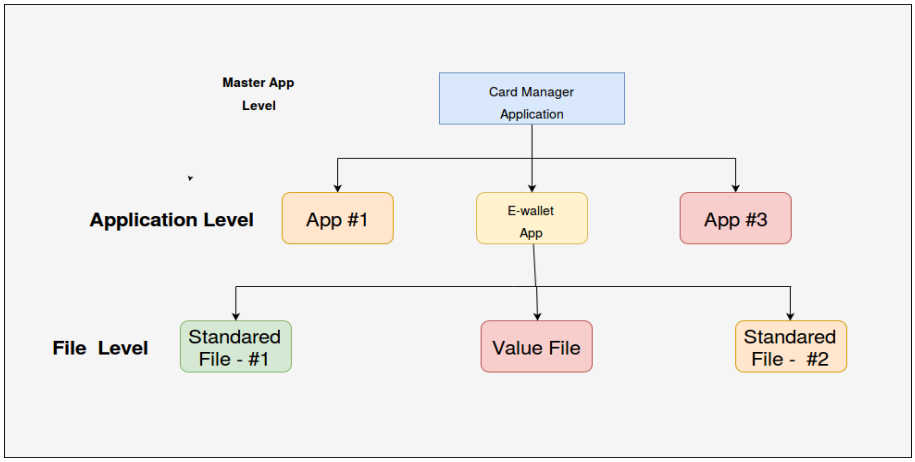


FIGURE 4.2: Abstract Representation of the Card Layout

1. Value File (0x03) (Credit/Debit/Limited Credit)
   (a) Read Key - 0x01
   (b) Write Key - 0x02
   (c) R/W Key - 0x03
   (d) Balance Read - Free
2. Standard File (0x04)(Card Information)
   (a) Read Key - 0x01
   (b) Write Key - 0x0E
   (c) R/W Key - 0x0E

   After the card personalization write permission is not allowed for this file.

TABLE 4.1: Card Information File Details

Data Field	Size(bytes)	Offset in Hex
Card Account Number	6	00-0x00
Card Type	1	06-0x06
Expiry Date	3	09-0x09
Status	1	10 - 0x0A

3. Standard File (0x05)(Transaction History)
   (a) Read Key - 0x01
   (b) Write Key - 0x02
   (c) R/W Key - 0x03

TABLE 4.2: Transaction History File Details

Data Field	Size(bytes)	Offset in Hex
Transaction Ref Number	6	00-0x00
Transaction Amount	3	06-0x06
Transaction Time	6	09-0x09
Status	1	15 - 0x0F
Transaction Type‘	1	16 - 0x10

4.2.1 Symmetric Key Diversifications

Key diversification is key feature of the Desfire EV1 card. Key diversification is deriving from the master keys by using unique in put of the card. By using this algorithm different values for each key can be generated. So if some keys of the card is broken the vulnerability is not effect to the other card in the field.

All the PICC keys in the HCE virtual card are diversified. In the personalization process all the diversified keys are generated by using HSM simulator which has save all master keys of the system. Master keys should be saved in the reader application. So secure storage should be used to store those master keys in the reader application. There I used Mifare SAM(AV2) to store master keys of the system in reader application.

There are two types of key diversification methods support by Mifare SAM AV2.

1. Based on classical encryption.
2. Based on CMAC calculation.

In this project I have used CMAC based key diversification method. Most of the current systems generate diversified keys by using one CMAC based calculation. It possible to use different sets of master keys by using key versions for the subscriber's card.

Shows the CAMC based key diversification algorithm as a blocked diagram below.(Figure - 4.3)

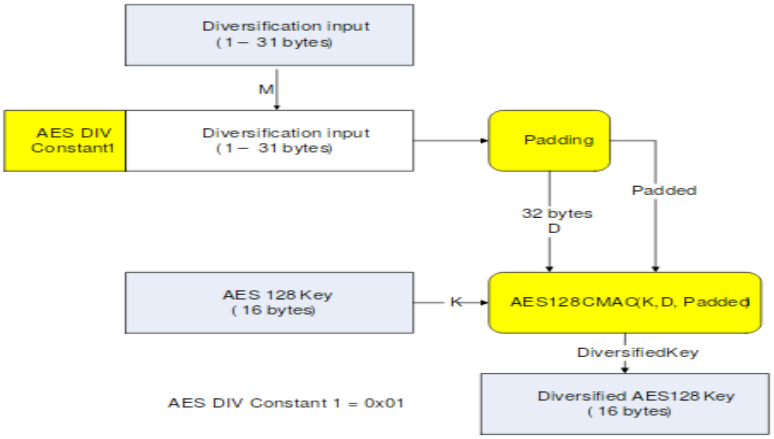


FIGURE 4.3: CMAC Based Key Diversification [1]

4.2.2 AES 3Pass Mutual Authentication

I have used AES keys in virtual card layout. In the AES authentication has multiple APDU exchanging steps between NFC card and the terminal application. After the successful authentication, session key is generated.

Sequence of the APDU exchanging of the AES authentication is shown below. [8]

1. The authentication process is started by the reader application. Reader sends command with key number to PICC card to initialize the process.
2. Once PICC receive the command, It generates a random number(16 bytes)(RandB) and encrypt generated random number by selected AES key. Then it returns encrypted data packet to reader.
3. After the reader receive the response it follows following steps.
   (a) Decrypt the received encrypted data and find the random number which is generated by the PICC.
   (b) Generate a 16 byte random number.
   (c) Rotate the generated random number by one byte.
   (d) Concatenates two random numbers and crate a new 32 byte array.
   (e) Then encrypt the created 32-byte byte array with selected AES key.
   (f) Then send the encrypted result to PICC.

4. After the PICC receive the response it follows following steps.
   (a) Decrypt the received encrypted data and find the two random number which is generated by the PICC and reader application.
   (b) Rotate the random number by one byte and compare with generated random number.
   (c) If comparison is success,Rotate the reader generated random number by one byte and encrypt it with selected AES key.
   (d) Sends the encrypted data to reader application.
5. After the reader receive the response it follows following step
   (a) Decrypt the received data with selected AES key.
   (b) Rotate the decrypted data by one byte.
   (c) Compare the generated random number and received random number. If both random numbers are equal then AES authentication is considered as successful.
6. After the successful authentication session key is generated as below. (Figure - 4.4)

$$\text{AES Session Key} = \text{RndA}_{(\text{byte 0-3})} + \text{RndB}_{(\text{byte 0-3})} + \text{RndA}_{(\text{byte 12-15})} + \text{RndB}_{(\text{byte 12-15})}$$

FIGURE 4.4: AES Session Key

Communication sequence of the AES 3Pass authentication is shown below. (Figure - 4.5)

4.2.3 External Interactions with virtual Card

HCE virtual card interacts with 3 main external entities and all communications are cryptographically protected using HSM simulator.

1. Android TA Application

   A Pre-perso key is used for authentication newly installed HCE virtual card instance before issuing diversified PICC keys. The HCE virtual card TA installation package is shipped with the pre-perso key injected to the DESFire application master key slot. Communication with the DESfire card during personalization is via secure channels.

2. HCE Virtual Card Management System

   HCE Virtual Card Management System interacts with HCE Virtual Card TA more frequently and such interactions are listed bellows,

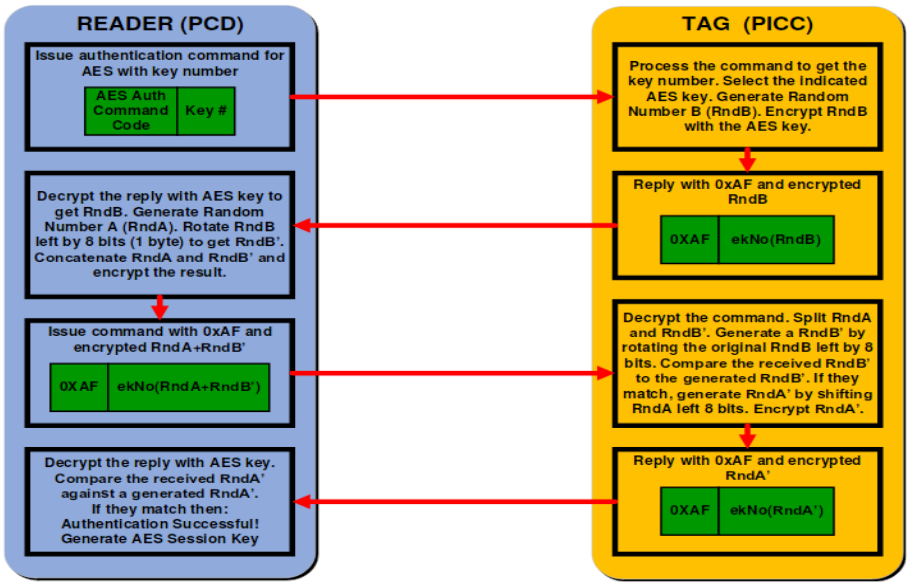


FIGURE 4.5: AES 3Pass authentication Communication Sequence

(a) One-time programming of Card Info such as UID.

(b) Card Information injection and diversified PICC key injection

(c) Manage card lifecycle

HCE Virtual Card Management System used standard DESFire AES 128 bit three-pass authentication before transmit any APDU to mobile application with pre-perso AES key. APDUs are interchanged between Management System and Virtual Card using a SSL protected communication channel to provide additional layer of security for the data-in-motion.

3. Acceptance Terminal

   These terminals are the existing card transaction acceptance terminals with a slight modification to support Virtual Cards (App Select command as in NFC-Forum ISO-DEP specification.)

4.3 Mobile Application

This mobile application is an Android 4.4 (Kit-Kat) or higher compatible application that hosts the virtual card functionality. HCE android application provides following protection profiles.

1. Root Detection
   If application is running on a rooted device sensitive information are not secure at all. So I checked the device is rooted or not before run the application. If the device is rooted then virtual card penalization is not allowed.

2. Anti-Debug

   Debug level of the android application has managed by using proguard rules.

3. Data Integrity Protection

   In the virtual card there are sensitive data. So I have protected integrity of those data by using CMAC calculation with offline key. In the initial step(ISO Application selection step) of the APDU exchange I have validated this CMAC value of the sensitive data. After the transaction new CMAC value is calculated and update the existing CAMC value.

4. Data security.

   I used SQLite database to store application data of the HCE android application. SQLite is not a secured database because SQLite database easily restore. So I encrypted all the data and generated CMAC with offline application specific key before insert to the database. This encrption key was keeped in a class veriable as string valus. We can use commercial obfuscator to obfuscate Andoid application. DexGuard [2] is a commercial obfuscator. Free version of the DexGuard tool is now available. It can additionally encrypt/obfuscate the strings and classes of the Android Application. Extracting the keys then takes even more time and expertise.

   Before encrypt the data I have concatenated offline counters which are sync with back-end management system to prevent database restoring. These counter are concatenate with transaction data and send updated counter to the management system. If the counter are duplicated system can detect it and can take necessary action to the virtual card.

   When HCE application receives first APDU from the terminal application(HCE application selecting command) HCE service reads data of the virtual card from the database. Then encrypted data is decrypted and validates the CMAC of the virtual card data. Using CMAC vaalidation, HCE application can keep the virtual card data in securly.

5. Code Obfuscation

   Any apk file can be easily decompiled by using revers engineering tools which are freely available. Then it can be easily transformed to malicious application thought the reverse engineering. So I have obfuscated android source code to make harder to reverse engineering. Android code obfuscation renames some Class names,variable names etc.

6. Use the lockscreen

   Android application forces to user to use android lockscreen functions. This feature can be used to improve security of the application.Use PIN, password or patten is the bare minimum level of the security that can be used in mobile devices.

7. Sign Android Application.

   Android application has digitally signed with a certificate.

4.4 Terminal application

This terminal application is an android compatible application that hosts the ticketing emulated solution. Terminal application provides following protection profiles.

1. Use Desfire AV2 SAM.

   All the master keys of the virtual card is stored in this SAM. SAM has personalized before use with terminal. In the personalization process all keys are injecting to SAM. There are offline keys in the SAM. Those keys are used to generate CAMC data for the transaction recode.

2. Use a diversified key for host key of the SAM.

   In the personalization process host key of the SAM has diversified. Diversification inputs are UID of the SAM. Therefore unique host key can be generated for each merchant SAM.

3. Before use the SAM it should authenticate with host authentication key.

   Before use the terminal application, It connects to the HSM simulator through the management server and get the diversified host key of the SAM. This key is store in the terminal internal memory. SAM can be activated by using this key.

4. Use CMAC protected communication with management portal to data exchange.

   SAM has some offline keys. After the transaction offline key should be activated and generate CMAC for the transaction data. Concatenate this CMAC value with the transaction data before send data to backend and backend server is validating this CMAC with transaction data.

Chapter 5

Implementation

5.1 Native Wrapped Command

In this mode, native commands are wrapped inside iso7816 style APDUs. I have used native wrapped command format to send APDU from terminal to virtual card because many readers are support to native wrapped commands. Format of the native wrapped command and response are shown below.

Command Format

CLA = 0x90
INC = Native Command
P1 = 0x00
P2 = 0x00
Lc= Length of the data
Data = Data

Response Format

SW1 = 0x91
SW2 =Native Response

5.2 Virtual Card Support APDU Commands

5.2.1 Read Card UID

Reading UID is required before performing transaction. Because all PICC keys of ticketing application is diversified with this UID. Details of the UID read command is shown below.

TABLE 5.1: Details of UID Read Command

Command	Virtual Card Success Response	Description
90000003000000	9100	Select Master Application
9060000000	91AF	Get Version Command
90AF000000	[7Bytes UID] 9100	Read Additional Frame

5.2.2 Read Card Information File

Card information related data are stored in a standard file. Before perform a transaction this file should be read and validate. Below table is describe the command and response of the virtual card.

TABLE 5.2: Details of Card Information Read Command

Command	Virtual Card Success Response	Description
905A000003010101	9100	Select Transport Application
903D0000070400000011000000	[11 Bytes Data] 9100	File Read Command

ISO wrapped command format of file read command is shown below.

CLA - 0x90

CMD - 0x3D

P1,P2 - 0x00 , 0x00

Lc- 0x07

File id - 0x04

Offset - 0x000000

Length - 0x110000

Lc - 0x00

5.2.3 Read Balance

Wallet balance is store in a value file. Value file of the virtual card has function such as GetValue, Credit, Debit and Limited Credit. Steps to read card balance is shown below.

TABLE 5.3: Details of Card Balance Read Command

Command	Virtual Card Success Response	Description
905A000003010101	9100	Select Transport Application
906C0000010300	[4-Bytes Value] 9100	Get Value Command

ISO wrapped command format of get Value command is shown below.

CLA - 0x90

CMD - 0x6C

P1,P2 - 0x00 , 0x00

Lc- 0x01

Data , File Id - 0x03

Lc - 0x00

Success Response [4 - Byte Data] 9100

5.2.4 PICC Authentication

PICC authenticate process is should be execute before send Credit, Debit and Limited Credit command. PICC key index is depend on the execution function.

ISO wrapped command format of init PICC authentication command is shown below.

CLA - 0x90

CMD - 0xAA

P1,P2 - 0x00 , 0x00

Lc- 0x01

(key Index) - 0x01

TABLE 5.4: Details of Card Balance Read Command

Command	Virtual Card Success Response	Description
905A000003010101	9100	Select Transport Application
90AA0000010100	[16-Bytes RandB'] 91AF	Init PICC Authentication
90AF000020 [32 Bytes Data]	[16 - Bytes RandA'] 9100	Second step of PICC Authentication Command

Lc - 0x00

Success Response [16 - Byte Data] 91AF

ISO wrapped command format of second step of the PICC authentication command is shown below.

CLA - 0x90

CMD - 0xAF

P1,P2 - 0x00 , 0x00

Lc - 0x20

Data (Encrypted Challenge) - 32 byte challenge

Le - 0x00

Success Response [16 - Byte Data] 9100

5.2.5 Credit

Credit command is used to top-up the virtual card. Before execute the credit command, Virtual card should be authenticated with 0x03 key.

ISO wrapped command format of the credit command is shown below.

CLA - 0x90

CMD - 0x0C

P1,P2 - 0x00 , 0x00

Lc 0x05

Data File ID - 0x03 , Credit Amount - 4byte- amount

Le - 0x00

TABLE 5.5: Details of Credit Command

Command	Virtual Card Success Response	Description
905A000003010101	9100	Select Transport Application
PICC Authentication With 0x03 Key		
900C00000503[4-Bytes]00	9100	Credit Command
90C7000000	9100	Commit Command

5.2.6 Debit

Debit command is used in the check-in function.Before proceed debit command, virtual card should be authenticated with 0x01 key.

TABLE 5.6: Details of Credit Command

Command	Virtual Card Success Response	Description
905A000003010101	9100	Select Transport Application
PICC Authentication With 0x01 Key		
900D00000503[4-Bytes]00	9100	Debit Command
90C7000000	9100	Commit Command

ISO wrapped command format of the debit command is shown below.

CLA - 0x90

CMD - 0x0D

P1,P2 - 0x00 , 0x00

Lc 0x05

Data File ID - 0x03 , Debit Amount - 4byte- amount

Le - 0x00

5.2.7 Limited Credit

Limited credit command is used in the check-out function. Before excecute limited credit command, virtual card should be authenticated with 0x01 key.

TABLE 5.7: Details of Limited Credit Command

Command	Virtual Card Success Response	Description
905A000003010101	9100	Select Transport Application
PICC Authentication With 0x03 Key		
901C00000503[4-Bytes]00	9100	Limited Command
90C7000000	9100	Commit Command

ISO wrapped command format of the Limited Credit command

CLA - 0x90

CMD - 0x1C

P1,P2 - 0x00 , 0x00

Lc 0x05

Data File ID - 0x03 , Refund Amount - 4byte- amount

Le - 0x00

5.2.8 Read Transaction History

Transaction history is saved in a standard file. This file is updating after the every successful transaction. In the check-out function, this file should be read and last transaction is validated.

Read Transaction History File

ISO wrapped command format of the file read command.

CLA - 0x90

CMD - 0x1C

P1,P2 - 0x00 , 0x00

Lc 0x05

TABLE 5.8: Details of Read Transaction History File

Command	Virtual Card Success Response	Descriptio
905A000003010101	9100	Select Transport Application
PICC Authentication With 0x01 Key		
903D0000070500000011 000000	[17Bytes Data] 9100	Read Transaction History File

Data File ID - 0x05 , Offset- 0x000000,Length-0x110000

Le - 0x00

Update transaction history file.

TABLE 5.9: Details of Write Transaction History File

Command	Virtual Card Success Response	Description
905A000003010101	9100	Select Transport Application
PICC Authentication With 0x02/0x03 Key		
90CD000010[16 - Byte Data]00	9100	Write Transaction History File
90C7000000	9100	Commit Command

ISO wrapped command format of the file write command.

CLA - 0x90

CMD - 0xCD

P1,P2 - 0x00 , 0x00

Lc 0x10

Data File ID - 0x05 , 16-Byte Data

Le - 0x00

5.2.9 Commit Transaction

Commit command is executed after the every file update in backup data file, value file or recode file. Commit transaction command is the last APDU of the transaction.

ISO wrapped command of the commit command is shown below.

CLA - 0x90

CMD - 0xC7

P1,P2 - 0x00 , 0x00

Lc 0x00

5.3 Terminal Application

5.3.1 Reload

The top-up function, reader application executes “select application command”. After selecting transport application of the card, reader application authenticates the card with 0x03 key. Sequence of the executing APDU and expected return status codes are shown in the table below.

TABLE 5.10: Steps of Reload Function

Description	APDU	Possible Responses From PICC
Select Master Application	00A4040007D276 0000850 10000	9000- SUCCESS, 6A84- Not a EV1 Card
Application Get Version Command	9060000000	[Data] 91AF - Additional Frame Request
Application Get Version Request Second Frame	90AF000000	[Data] 9100 - SUCCESS
Select Ticketing Application	905A00000301010100	9100 - SUCCESS, 91A0 - Invalid PICC Card
PICC Authentication With Key 0x03	90AA0000030300, 90AF000010[32byte]00	9100 - SUCCESS, 91AE - Authentication Faied
Read Transaction History File	903D0000070500000011 000000	[Data]9100 - SUCCESS, 91AE - No valid authentication, 91BE - Invalid File Size.
Credit Command	900C00000503[4-Bytes]00	9100 - SUCCESS, 91AE - No valid authentication, 91BE - Maximum limit Exceeded.
Update History File	90CD000010[16 - Byte Data]00	9100 - SUCCESS, 91AE - No valid authentication,91BE
Commit Transaction	90C7000000	9100 - SUCCESS

5.3.2 Check-In

When passenger gets into bus, he should tap his phone on check-in terminal to purchase a ticket. Terminal application will execute the debit function and charge the amount for last location of the current trip of the bus.

Before executing the debit function, reader application executes "select application command". After selecting transport application of the card, reader application authenticates the card with 0x01 key. Sequence of the executing APDU and expected return status codes are shown in the table below.

TABLE 5.11: Steps of Check-In Function

Description	APDU	Possible Responses From PICC
Select Master Application	00A4040007D276000 0850 10000	9000- SUCCESS, 6A84- Not a EV1 Card
Application Get Version Command	9060000000	[Data] 91AF - Additional Frame Request
Application Get Version Request Second Frame	90AF000000	[Data] 9100 - SUCCESS
Select Ticketing Application	905A00000301010100	9100 - SUCCESS,91A0 - Invalid PICC Card
PICC Authentication With Key 0x01	90AA0000030100, 90AF000010[32byte]00	9100 - SUCCESS,91AE - Authentication Failed
Debit Command	900C00000503[4-Bytes]00	9100 - SUCCESS, 91AE - No valid authentication, 91BE - Insufficient balance.
Update History File	90CD000010[16 - Byte Data]00	9100 - SUCCESS, 91AE - No valid authentication,91BE
Commit Transaction	90C7000000	9100 - SUCCESS

5.3.3 Check-Out

Passenger should tap his phone on check-out terminal when getting down from the bus. If he gets down before the last stop of the trip he will be refunded the additional amount charged at check in time. Here, limited credit

function is used to refund the balance.

Before executing the limited credit function, reader application executes "select application command". After selecting transport application of the card, reader application authenticates the card with 0x02 key. Sequence of the executing APDU and expected return status codes are shown in the table below.

TABLE 5.12: Steps of Check-Out Function

Description	APDU	Possible Responses From PICC
Select Master Application	00A4040007D2760000850 10000	9000- SUCCESS, 6A84- Not a EV1 Card
Application Get Version Command	9060000000	[Data] 91AF - Additional Frame Request
Application Get Version Request Second Frame	90AF000000	[Data] 9100 - SUCCESS
Select Ticketing Application	905A00000301010100	9100 - SUCCESS,91A0 - Invalid PICC Card
PICC Authentication With Key 0x02	90AA0000030200, 90AF000010[32byte]00	9100 - SUCCESS,91AE - Authentication Failed
Debit Command	900C00000503[4-Bytes]00	9100 - SUCCESS, 91AE - No valid authentication,91BE - Maximum limit Exceeded.
[2em] Update History File	90CD000010[16 - Byte Data]00	9100 - SUCCESS, 91AE - No valid authentication,91BE
Commit Transaction	90C7000000	9100 - SUCCESS

Chapter 6

Results and Evaluation

6.1 Overview

This research project has focused on emulating desfire EV1 card on android application. Application testing was facused on the functional testing and performance testing of the application. Performance of the HCE application was compared with the existing smart cards.

6.2 Testing Environment

When considering smartphone hardware, Android version, build version, firmware model and manufacture, there is vast permutation of devices in used today. Therefor it is more difficult to test each and every permutation. So three android devices were selected to test the HCE application. Two android phones from two different manufactures were selected. The other android phone was rooted phone.

TABLE 6.1: Test Device Profile

	Samsung I9505 Galaxy S4 [5]	LG G5 [6]	Samsung G935F Galaxy S7 Edge [12]
Model	GT-I9505	LG-H860	G935F
IMEI	356266053700677	358394071476571	358982072697144
Android version	5.0.1	6.0.1	6.0.1
Build number	LRX22C. I9505XXSHQA3	MMB29M	MMB29K.G
Rooted Phone	Yes	No	No

6.3 Results

6.3.1 Functional testing

Functional testing was done with above three phones. Rooted phone does not allow to do personalization of the HCE application. Therefor transaction couldn't do with the rooted phone. Functional testing results of the other two phones are shown in the table below.

TABLE 6.2: Functional Testing Result

	LG G5 [6]	Samsung G935F Galaxy S7 Edge [12]
Check-in	Success	Success
Check-out	Success	Success
Top-up	Success	Success
Balance Check	Success	Success

6.3.2 Performance testing

After first request is sent to HCE application, "response time" measures how long it takes for a response to return back from the HCE application. I calculated time between first APDU request and the last APDU response of the below functions for mobile application and the Desfire EV1 card.

1. Check-In Function.
2. Check-Out Function.
3. Top-up Function.
4. Balance Check Function.

The below statistics taken by after performing 30 transactions and values are derived by taking average of those transactions. All values are in milliseconds.

HCE APDU Execution Time Analysis

The below statistics taken by after performing 30 transactions and values are derived by taking average of those transactions. All values are in milliseconds. (Figure: 6.1)

TABLE 6.3: Transaction Time

	Desfire Ev1 (ms)	LG G5 (ms)	Samsung G935F Galaxy S7 Edge (ms)
Check-in	276	556	546
Check-out	265	610	613
Top-up	291	605	614
Balance Check	183	380	372

"Time taken for APDU execution at Android App" column shows the milliseconds laps between each APDU command receipt by the App and dispatch of the corresponding command response to the reader by the app.

Step	Time taken for APDU execution at Android App	Terminal's waiting time between APDU command and response	Difference
ISO Application Selection	14	106.9	92.9
Application Get Version Command	10	32	22
Application Get Version Request Second Frame	12	37	25
Select Ticketing Application	20	46	26
Init Auth with 0x01 Key	43	66	23
Final Auth Command	46	70	26
Debit Command	22	43	21
Update History File	20	32.9	12.9
Commit	92	120	28
Total	**279**	**553.8**	**276.8**

FIGURE 6.1: APDU Excecution Time Analysis

"Terminal's waiting time between APDU command and response" column shows the milliseconds laps between each APDU command dispatch by the Reader and receipt of the corresponding command response by the reader.

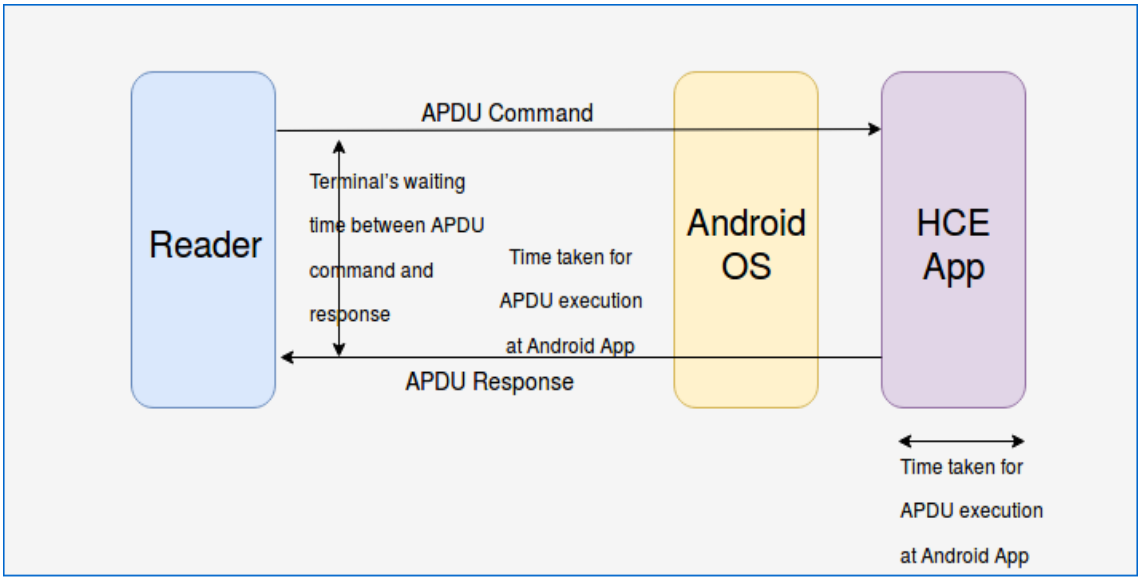


FIGURE 6.2: HCE APDU Execution Time Analysis

There is a time gap between terminal waiting time and the android application APDU execution time(Figure: 6.2). There for this time gap is taken by the android OS. Android NFC adapter takes 10-100ms to route the APDU from NFC adapter to HCE application. Therefor there is performance issue in the HCE application. This delay effect to total transaction time.

Chapter 7

Conclusion and Future Works

7.1 Overview

This research project has focused on emulating desfire EV1 card on Android application. More security can be added to the HCE Android application. Android application can be improved to use to make various payments.

7.2 Finding and Limitation

There are few difficulties encountered during this project. Finding full specification about Desfire EV1 card was most difficult but there are lot of information available on various sources.

7.3 Future Works

The area which this project has done,has potential scope for further study.

New payment methodologies can be introduce to virtual card payment other than the NFC terminal payment.

New Payment Methodologies

1. QR Payment
   QR code payment also kind of contactless payment method.QR based payment is going to both speed up the e-commerce payment experience and make it more secure. This feature can be implemented on the android application and same wallet can be used for the QR payment.

2. Utility Bill Payment
   HCE android application can be improved to pay utility bills. For this payment also HCE wallet can be used.

3. Money Transfer between Virtual Card Wallets.
   HCE android application can be improved to transfer the money between HCE virtual card wallet to wallet.

Improve the Support Commands

In this stage we have emulated most important functionalities of the Desfire EV1 card. Rest of the functionaries can be emulated.

In this POC, virtual card supports following Desfire commands.

1. Get Version.
2. Select Application.
3. Authenticate AES.
4. Get Key Settings.
5. Create Application.
6. Get Application IDs.
7. Read Data.
8. Write Data.
9. Get Value.
10. Credit.
11. Debit.
12. Limited Credit.
13. Write Recode.
14. Read Recode.
15. Commit.
16. Abort Commit.

Improve Functionalities of the HCE Application

In this project, mainly focus on transport ticketing solution. We can extend this project to support on following areas.

1. Access management.
2. Electronic toll collection .
3. Car parking system.
4. Loyalty programs.
5. Micro payments

7.4 Conclusion

The objective of the POC project is to create a proof of concept which is support to physical card payment slotions can be replaced with host card emulation HCE Android application. Most of the existing ticketing slotions are using only physical smart cards. Due to complexity of the physical card solutions, we proposed Desfire EV1 card emulated HCE ticketing solution. We have measured system performance qualitatively and quantitatively by comparing with Desfire EV1 card functions.

7.5 The problem

The problem domain of this thesis is to provide secure, efficient and costeffective alternative solution for the smart cards using for offline payments.

7.6 Solution

We have implemented HCE Desfire EV1 card emulated Android application to replace smart card from payment systems to mitigate complexity of the current transport industry.

Backend server system was proposed to manage the virtual card of the HCE application. HSM simulator was used to store master keys of the proposed system and perform crypto operations.

Bibliography

[1] *AN10922-Symmetric key diversifications*. [online]. Mar. 2017. URL: www.nxp.com/documents/application_note/AN10922.pdf.

[2] *DexGuard,Protecting Android applications and SDKs against reverse engineering and hacking*. [online]. Mar. 2017. URL: https://www.guardsquare.com/en/products/dexguard.

[3] *Fibank launches HCE in Bulgaria*. [online]. Apr. 2016. URL: https://www.nfcworld.com/2016/04/21/344205/fibank-launches-hce-bulgaria/.

[4] *GD provides commercial Cloud payment solution to Commonwealth Bank of Australia*. [online]. Mar. 2015. URL: https://www.iot-now.com/2015/03/24/31299-gd-provides-commercial-cloud-payment-solution-to-commonwealth-bank-of-australia/.

[5] *Gsmarena Samsung I9505 Galaxy S4*. [online]. Mar. 2017. URL: https://www.gsmarena.com/samsung_i9505_galaxy_s4-5371.php.

[6] *Gsmarena Samsung I9505 Galaxy S4*. [online]. Mar. 2017. URL: https://www.gsmarena.com/lg_g5-7815.php.

[7] *Host-based Card Emulation*. Host-based Card Emulation [online]. Feb. 2017. URL: https://developer.android.com/guide/topics/connectivity/nfc/hce.html.

[8] Ralph Jacobi and Josh Wyatt . *MIFARE DESFire EV1 AES Authentication With TRF7970A*. Texas Instruments, 2014.

[9] *Kazakhstan gets HCE payments*. [online]. Feb. 2016. URL: https://www.nfcworld.com/2016/02/08/342071/kazakhstan-gets-hce-payments/.

[10] *MasterCard Launches HCE Projects*. [online]. Feb. 2015. URL: https://paymentweek.com/2015-2-25-mastercard-launches-hce-projects-to-boost-international-mobile-payments-programs-6734/.

[11] *Rambus HCE Ticketing Solution*. Rambus HCE Ticketing App [online]. Feb. 2017. URL: https://www.rambus.com/security/smart-ticketing/hce-ticketing-app/.

[12] *Samsung Galaxy S7 edge*. [online]. Mar. 2017. URL: https://www.phonearena.com/phones/Samsung-Galaxy-S7-edge_id981.

[13] *The Rambus HCE Solution Overview*. Rambus HCE Ticketing App [online]. Feb. 2017. URL: https://www.rambus.com/security/payments/host-card-emulation/.

[14] *The Statistics Portal, Forecast of mobile phone users worldwide*. [online]. Mar. 2017. URL: https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/.

[15] *The Statistics Portal, NFC enabled mobile devices worldwide*. [online]. Mar. 2017. URL: https://www.statista.com/statistics/461494/nfc-enabled-mobile-devices-worldwide/.

[16] *Wikipedia - List of Smart Cards*. [online]. June 2018. URL: https://en.wikipedia.org/wiki/List_of_smart_cards.

[17] *Wikipedia - Smart Card*. [online]. June 2018. URL: https://en.wikipedia.org/wiki/Smart_card.