

A study on the security aspects and  
limitations of mobile payments using Host  
Card Emulation (HCE) with Near Field  
Communication (NFC)

Shana Micallef

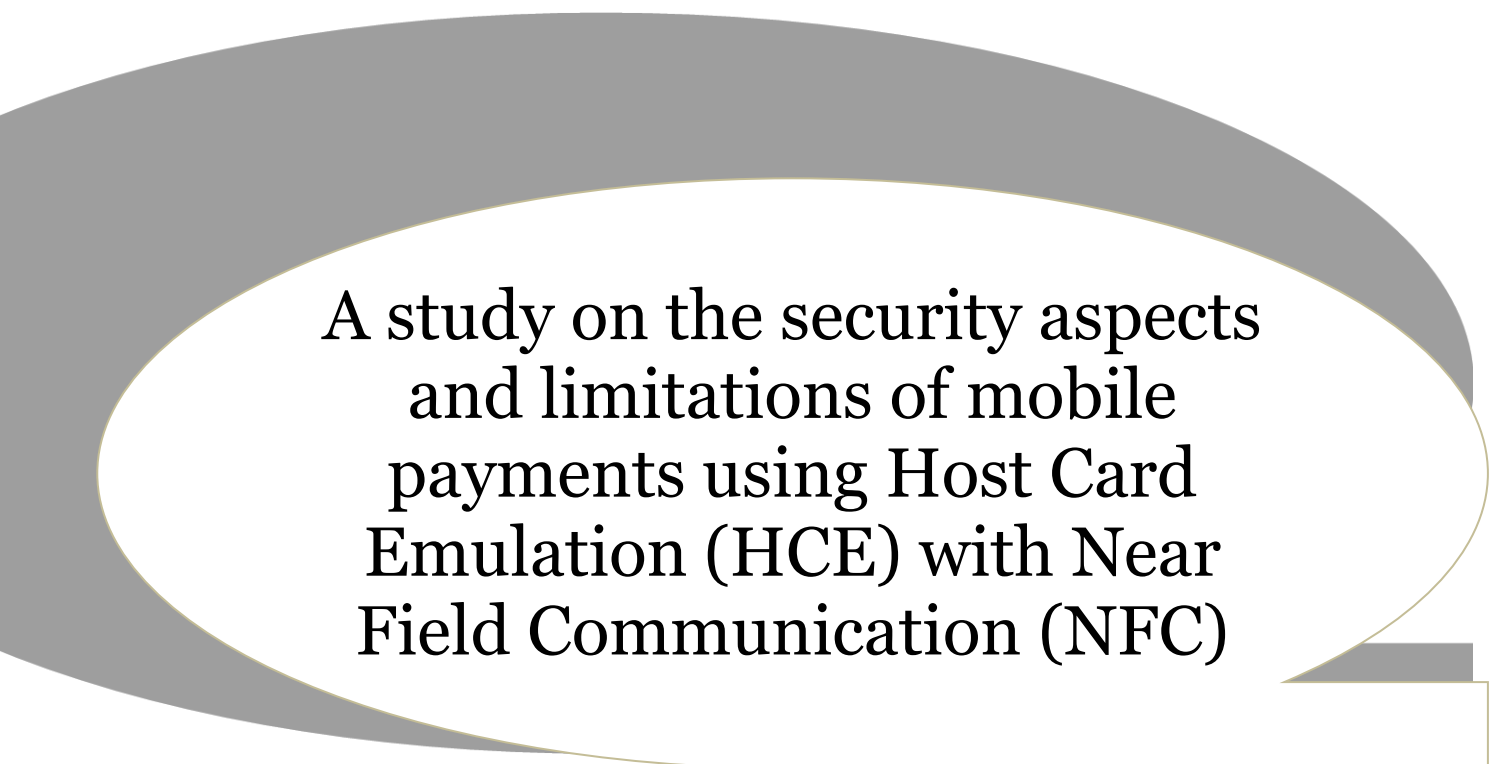
Technical Report

RHUL-ISG-2018-6

5 April 2018



Information Security Group  
Royal Holloway University of London  
Egham, Surrey, TW20 0EX  
United Kingdom



A study on the security aspects  
and limitations of mobile  
payments using Host Card  
Emulation (HCE) with Near  
Field Communication (NFC)

**Author Name:** Shana Micallef  
**Supervisor Name:** Professor Kostantinos Markantonakis  
**SRN:** 130268415  
**Date:** March 2017

**Declaration:**

Submitted as part of the requirements for the award of the MSc in Information Security of the University of London.



# Table of Contents

Abbreviations.....	3
Acknowledgements .....	4
Executive Summary.....	5
1 Introduction.....	6
1.1 Introduction.....	6
1.2 Motivation.....	7
1.3 Objectives.....	7
1.4 Structure of the project.....	8
2 An overview of contactless payments.....	9
2.1 History of contactless payments.....	9
2.2 Contactless payments standards and specifications.....	9
2.2.1 Near Field Communication .....	10
2.2.2 Tokenisation.....	10
2.2.3 Authentication and Verification Methods .....	11
2.2.4 Transaction Data Encryption .....	12
2.2.5 Card Scheme HCE specifications .....	13
2.3 Mobile devices and Host Card Emulation .....	13
2.3.1 Near Field communication .....	14
2.3.2 Host Card Emulation architecture .....	14
2.3.3 Models of HCE.....	16
2.4 Stakeholders roles in a Mobile Payment Ecosystem.....	18
3 An analysis of attacks on traditional and HCE based contactless payment schemes.....	23
3.1 Literature review of attacks on traditional contactless payments .....	23
3.1.1 Attacks at the Physical Layer .....	23
3.1.2 Attacks at the Application Layer.....	29
3.1.3 Attacks on Cryptography and Key Management .....	32
3.2 Literature review on attacks of mobile payments using Host Card Emulation (HCE) .....	34
3.2.1 Attacks at the Operating System/Kernel Mobile Device.....	34
3.2.2 Attacks on Secure Memory Areas .....	40
3.2.3 Attacks on Consumer Device Cardholder Verification (CDCVM) methods .....	41
3.2.4 Attacks on Tokenization and its Infrastructure.....	43
3.3 Review of existing HCE Implementations.....	44
3.3.1 Device/OS Provider HCE Model.....	45
3.3.2 Issuer HCE Model .....	47
3.3.3 Summary on the Advantages and Disadvantages .....	48
4 Modeling an HCE Payment Transaction.....	50
4.1 Review of modelling tools and their application in modelling similar payment transactions ...	50
4.2 Methodology .....	52
4.2.1 Acquiring reference documentation.....	52
4.2.2 Developing the model.....	56

4.3	Development of an HCE Payment Transaction Model .....	57
4.3.1	Architecture .....	57
4.3.2	Data and Structures.....	58
4.3.3	State Enumeration .....	59
4.3.4	Connective Junction .....	60
4.3.5	Payment Transaction Process .....	60
4.3.6	Token Requestor Process .....	62
4.3.7	Account Management Process.....	64
5	Security Analysis of an HCE Payment Transaction .....	72
5.1	Security analysis of customer verification and authentication techniques when employed in HCE payments .....	72
5.1.1	Authentication vs Verification.....	72
5.1.2	Verification of the POS during a two tap transaction.....	72
5.1.3	Issues of Pre-Authentication.....	75
5.1.4	Sharing the verification database between Mobile Device and Payment Application .....	75
5.1.5	Shift of control from Issuer to Wallet App Owner .....	76
5.2	Security analysis of the tokenization process used during an HCE payment transaction .....	76
5.2.1	Token Generation.....	77
5.2.2	Device authentication during enrolment.....	78
5.2.3	LUK location risk analysis .....	78
5.2.4	Token to Device Mapping.....	79
5.2.5	LUK Supply and Demand Synchronisation .....	80
5.2.6	Issues of Tokenization in MSD Mode.....	80
5.3	Analysis of the impact in operating cryptographic functions and storing cryptographic keys, required during an HCE payment process, on a mobile device.....	81
5.3.1	Preventing access to the LUK through the TEE .....	81
5.3.2	Offline Data Authentication (ODA) RSA Key Storage .....	82
5.3.3	Issues of using LUKs for authentication to the TR and the VISA Cloud Platform .....	83
5.4	Other generic issues to the use of mobile phones for payments .....	83
5.4.1	Phishing and social engineering .....	84
5.4.2	Installation of rogue and malware applications.....	84
5.4.3	Reverse engineering of payment application and mobile operating system .....	84
5.4.4	Denial of service attacks and data connectivity compromise .....	84
5.4.5	Problems related to the use of biometrics .....	84
5.4.6	Privacy.....	84
5.5	Summary of Analysis and Main Findings .....	85
6	General Conclusion.....	88
6.1	Summary .....	88
6.2	Areas of Further Research .....	90
7	Bibliography .....	91
8	Appendix .....	100
8.1	Appendix 1 – History of Contactless Payments Timeline .....	100

## Abbreviations

Acronym	Description
ARPC	Authorization Response Cryptogram
ARQC	Authorization Request Cryptogram
ASK	Amplitude Shift Keying
ATC	Application Transaction Number
ATM	Automated Teller Machine
CDCVM	Consumer Device Cardholder Verification Method
CVE	Common Vulnerabilities and Exposures
CVM	Cardholder Verification Method
DDOL	Data Authentication Data Object List
DoS	Denial of Service
E2EE	End-to-end Encryption
EMV	EMVCo
ENISA	European Union Agency for Network and Information Security
fDDA	Fast Dynamic Data Authentication
FIDO	Fast Identity Online Alliance
FWI	Frame Waiting Time Integer
FWT	Frame Waiting Time
HCE	Host Card Emulation
HF	High Frequency
LUK	Limited Use Key
mBanking	Mobile Banking
MHz	Megahertz
mS	Milliseconds
MSD	Mag-Stripe Mode
NFC	Near Field Communications
NIST	National Institute of Standards and Technology
OMACP	OMA Client Provisioning
OS	Operating System
OTP	One Time Password
P2PE	Point-to-Point Encryption
PAN	Payment Card Number
PCI PTS	Payment Card Industry PIN Transaction Security
PCI SSC	Payment Card Industry Security Standards Council
PIN	Personal Identification Numbers
POS	Point of Sale
qVSDC	Quick Visa Smart Debit/Credit
RF	Radio Frequency
RFID	Radio Frequency Identification
SE	Secure Element
TCP	Transmission Control Protocol
TEE	Trusted Execution Environment
TR	Token Requestor
TSP	Token Service Provider
TTQ	Terminal Transaction Qualifier
UDK	Unique Derivation Key
UHF	Ultra-High Frequency
UN	Unpredictable Number
URL	Uniform Resource Locator
uS	Microsecond
WIFI	Wireless Fidelity

## **Acknowledgements**

I would like to thank my supervisor, Professor Kostantinos Markantonakis for his support, guidance and review of the progress of my work in completing this project.

I would also like to thank my husband, David, who supported me throughout these years while reading for my MSc. Moreover, he helped me proof-read this document.

Finally, I would like to thank Conrad Micallef, Charles Grech and Wilhelm Attard, for their technical support throughout the development of this project.

## Executive Summary

The payments industry has evolved from magnetic stripe cards to smart cards and eventually to contactless cards payments. Now the industry is making another step in its evolution by replacing the card with a mobile phone. Initially a solution based on a secure element in the mobile phone was proposed but Issuers did not like the reliance on MNOs and eventually HCE was proposed as a solution. In HCE, an App, termed a wallet is responsible for communication with a POS during a transaction and for storing the payment credentials. To limit risk in HCE, payment credentials are exchanged for tokens that have limited use, a process known as tokenization.

In recent years, a number of wallet applications have been introduced by different stakeholders, including Android Pay deployed by Google and Microsoft Wallet deployed by Microsoft. However, Issuers can also develop their own wallet app. Other than for some minor specifics, such as exchanging credentials with tokens, HCE uses the same infrastructure used by contactless cards. A transaction executed by a contactless card is nearly identical to a transaction done through HCE. Most card schemes support payments with HCE and also provide services such as tokenization.

HCE is attractive to consumers because it allows for quick and convenient way to make a payment. It is faster as cardholder verification can be done on the mobile device using biometrics instead of PINs. It is also more user interactive and HCE wallet can provide other services that a physical card cannot, e.g. providing a history of payments. The benefits of HCE is not only limited to cardholders but also to merchants and Issuers. Issuers can deploy loyalty programs through HCE without possible additional marketing costs.

While convenience is a necessity in today's world, HCE comes with its own set of risks. Given the use of mobile devices, HCE could be exposed to a number of threats and vulnerabilities such as malicious attacks from malware running on the mobile device. Thus the industry needs to be more pro-active to ensure that the same level of security provided by a Secure Element is achieved.

In this regard, the aim of this project was to evaluate current implementations from a security perspective. To achieve this, a finite state machine model of an HCE wallet application, based on the requirements provided by VISA and in line with EMV's specifications was developed. The model was then studied for security risks with particular emphasis on customer verification and authentication methods, tokenization and the impact in operating cryptographic functions and storing cryptographic keys, required during an HCE payment process. Other generic issues in the use of mobile phones for payments when employed in HCE payments were also outlined.

As a result of the risks identified, the main findings of this project can be summarized into the following points:

- A safe storage for limited use keys and other sensitive data is required on the mobile device. Shifting the data to the cloud does not necessarily eliminate the risk of theft of such keys.
- The generation of tokens (i.e. LUKs) should occur on the mobile device rather than on the cloud. Considering all the risks involved between doing this process in the cloud and on the mobile device, the latter is a safer approach.
- There is a strong need for a method/technique that can be used to uniquely identify a mobile device. The technique should be difficult to tamper with.
- A supply and demand synchronization mechanism between token generation and token usage is required. While tokens are of limited use, a better way of managing them is required.



# 1 Introduction

## 1.1 Introduction

Payment technologies have come a long way since the first payment cards were introduced. Until the introduction of EMV standards, credit or debit card transactions used a magnetic stripe to record account data. Whilst mag-stripe was simple and cheap they were easy to clone. As a result, EMV chip and pin cards and EMV contactless technology have been introduced to replace the magnetic stripe. Both technologies allow the card to make cryptographic operations making it possible to implement countermeasures against cloning, unauthorised use and many other possible fraud attacks.

HCE is essentially an extension to the EMV contactless technology. EMV contactless cards use Near Field Communication to communicate with the payment terminal. HCE is a technique whereby a mobile device with embedded NFC hardware emulates a contactless card. An app, normally referred to as a 'digital wallet' manages the payment credentials and emulates the card/s when the mobile is tapped on a POS. The advantages and disadvantages of using an emulated card on a mobile device instead of a card will be dealt further in this study.

The launch of digital wallets, by device/OS vendors, including Apple Pay in 2014, Android Pay and Samsung Pay in 2015, and Microsoft Wallet in 2016 have generated a lot of interest to the public in general and other market players in the industry such as banks and payment schemes. Each entity is free to utilize different options when it comes to implementing the digital wallet in the mobile device. Apple have chosen to deploy NFC payments using an embedded secure element (SE), to store the credentials. Samsung Pay chose to implement NFC with a TEE and Android Pay and Microsoft Wallet use NFC with host card emulation (HCE). The difference in these implementation is mostly related to where and how the payment tokens and cryptographic keys are stored.

Consumers are also provided with the option of having multiple digital wallets on the same mobile device. Such possibility offers a fast and convenient solution to both the consumer and the merchant. Cardholder verification can be implemented through the fingerprint sensors or biometric hardware on the mobile device instead of entering pins on payment terminals thereby adding to user convenience. It reduces queuing in the store, cash management and handling. There is also the possibility of delivering promotions and/or integrating loyalty opportunities in the mobile payment experience, making HCE a marketing platform. There is little difference in the transaction experience at the POS between contactless cards and cards emulated on the mobile device from a usability perspective to the consumer.

Despite such significant opportunities and the increase of mobile payment transactions, the perception in the marketplace is that mobile payments are risky. This is evident in the ISACA's 2015 Mobile Payment Security study which shows that 87% [1] expected to see an increase in mobile payment data breaches during 2016, and only 23% [1] of cybersecurity experts believe that mobile payments keep personal information safe.

Given HCE is still being rapidly evolving, early HCE implementations may not provide the same level of security provided by EMV cards. Furthermore, limited number of specifications/recommendations exist, that are specific to HCE. These are issued by the different card payment schemes, and are constantly being updated. Most of the options and issues exclusive to HCE, are related to where sensitive data is stored. In HCE, payment credentials and cryptographic keys could become accessible in the mobile's operating system if not securely protected. Hence, vulnerabilities such as rooting the mobile device, introducing malware to the mobile device and lost or stolen devices can make it easier for fraudsters to gain access to sensitive information stored in the wallet app and use it for fraudulent reasons. For these reasons new

techniques, such as Tokenization have been introduced to combat such risk. Such process will also be analyzed within this project.

It is difficult to identify all risks within such a large eco-system and hence the main aim of this project is to model the software that emulates a contactless card, known as the HCE wallet app, as a finite state machine. Such a model will be able to provide a point of reference for the different states that exist, in a payment transaction. Security risks, at each state, are outlined for different stakeholders in the eco-system. The model outlines how sensitive data is transformed during the various stages of a payment from the moment the mobile is tapped on a payment terminal until the actual payment is processed by the Issuer. In the end, the aim is to outline risks in the ecosystems and possibly come up with countermeasures or better implementation methods.

## **1.2 Motivation**

The reason why I have decided to explore HCE technology comes from an initiative in the Bank where I am currently employed. The Bank has started an evaluation process to implement HCE and provide it as an added value service to its clientele. It would allow the bank to diversify in its digital payments services offered and furthermore leveraging the potential of mobile devices, the bank can deliver its own marketing to its clients. The Bank also sees it as a viable alternative to payment cards and as an alternative to cash payments.

Despite these benefits, it is clear that HCE deployments pose several risks and as such my interest in evaluating it further. HCE opens up many possibilities for security threats, data privacy issues and many fine line discussions when dealing with industry payment implementations. As part of my roles and responsibilities as an IT auditor, I am responsible to prove assurance to management, board of directors and audit committee that with every new process, procedure and initiative, any significant, both present and potential risks that emerge are identified, monitored and managed effectively. Unmanaged risks can give rise to security breaches, and privacy risks which can result in financial losses and market reputation issues for the Bank. Hence, through this project I expect to have a deeper understanding of HCE, identify its risks and propose countermeasures and improvements. I expect this topic to be very interesting and challenging.

## **1.3 Objectives**

The Objectives of this project are the following:

- i.** Review relevant existing literature on NFC technology and HCE payment implementations.
- ii.** Model the HCE payment transaction process flow as well as the tokenisation process using formal methods. Following the modelling process, a set of threats and attacks that could exist during an HCE payment transaction, will be identified and discussed.
- iii.** Analyze the security aspects and weaknesses within the elements that make up HCE payment scheme and EMV payment tokenization infrastructure.
- iv.** Conduct a high level risk assessment on the identified threats and how certain controls can be applied to mitigate certain risks.

## 1.4 Structure of the project

This document is structured as follows:

Chapter 2, titled "An overview of contactless payments", provides an overview on the history of contactless technology and specifications used by the payment industry. This chapter will also briefly highlight the stakeholders' role in the mobile eco system as well as some facts on NFC and HCE architecture. This chapter is an introduction to the first objective.

Chapter 3, titled "An analysis of attacks on traditional and HCE based contactless payment schemes", will involve an analysis of attacks on contactless payments including traditional and HCE. Some commercial implementation will be identified along with their associated advantages and disadvantages. Thus, this chapter addresses the first objective.

Chapter 4, titled "Modelling an HCE Payment Transaction" will include reviewing modeling tools including their characteristics and limitations and their application in modeling similar payment transactions by other academics or researchers. Following this, the HCE process and tokenization processes will be modeled using a selected modeling tool.

Chapter 5, titled "Security Analysis of an HCE Payment Transaction", analyses from a security perspective. As this title reads, this addresses the third objective.

Chapter 6, titled "Conclusion", sums up the project by summarizing the main findings and provides a reference for further research on the subject. This chapter addresses the fourth objective. Wrapping up this chapter by outlining areas of further research.

## 2 An overview of contactless payments

Radio Frequency Identification (RFID) [2] is technology that uses close range wireless communication to transmit a unique 'id' related to a particular object or entity. A small tag, sticker or card would house a chip with a unique or programmable id value. Such a tag would then be used to replace a physical key, a passport or contactless payments to name a few. The specifications of the communication between the tag and reader are defined in ISO-14443 [3]. The main difference between a tag and a contactless card is that the chip in a contactless card is capable of some computation (e.g. generating a cryptographic signature) rather than just transmitting back a Unique ID.

A report [4] by Visa Europe issued in April 2016 states that there are around 3 million terminals (POS) with NFC contactless capabilities and around 130 million contactless cards. The report states that between June 2014 and June 2015, around 1.4 billion transactions were made using contactless cards summing up to a total of £14 Billion in transactions [4].

According to the UK Card Association [5], in December 2015, 12% of the total transaction made using credit and debit cards used contactless technology. The UK had 81.5 million contactless cards issued and during 2015 a total of 1 billion transactions were made using contactless cards for a total of £7.75bn. An increase of 233% in spending was recorded over the year in 2015.

In the near future, mobile payments will be a significant payment medium potentially overtaking any type of smart cards [6]. The next section will cover a brief summary of timelines outlining the early inventions of contactless technology including first patents awarded the evolution and development of NFC.

### 2.1 History of contactless payments

The industry has evolved from contactless EMV card payments, to mobile NFC payments using secure elements to HCE payments. In 1997 the first patent for RFID was awarded [7]. This served as the baseline for the invention of the NFC. Also within the same year, Mobil Oil Corp provided a key fob to be used by its customer to pay for fuel in its US gas stations.

Around 2004, the NFC forum was formed by Nokia, Philips and Sony with the purpose to advance the use of NFC technology and to create standards and interoperability in the mobile industry [8].

In 2007, Barclaycard introduced the first contactless cards in the UK [7]. Following this period, Google launched Google Wallet which enabled Android users to pay the participating retailers with their mobile phones, simply tapping the phone on any contactless enabled terminal at checkout. This system was initially implemented using an SE-based model.

In 2013, Google released the first HCE architecture in Android 4.4 KitKat and subsequently launched Android Pay. Later in 2014, Google stopped supporting the SE version [9]. Within the same year, Apple Pay was also introduced in the market. Next Visa, MasterCard, Android Pay, Samsung Pay and Microsoft Wallet also launched their contactless mobile payment platform [10] [11] [12] [13].

A more detailed historic audit trail on contactless payments can be located in Appendix 1.

### 2.2 Contactless payments standards and specifications

Standard bodies and industry organisations such as Smart Card Alliance [14] and EMVCo [15] have created a set of standards and guidelines to ensure interoperability between different equipment and processes within the contactless payments industry. The following sections provide an overview of the different standards pertaining to HCE.

## 2.2.1 Near Field Communication

1. ISO/IEC 14443:
  - Is the international Standard for proximity contactless smartcards. This standard defines the physical layer of NFC communication. There are four parts that comprise the Standard:
    - Part 1: ISO/IEC 14443-1 [16] defines the physical characteristics of the cards.
    - Part 2: ISO/IEC 14443-2 [17] specifies the radio frequency power and signal interface.
    - Part 3: ISO/IEC 14443-3 [3] defines the initialization and anti-collision measures used in the protocol.
    - Part 4: ISO/IEC 14443-4 [18] specifies the transmission protocol requirements. It includes special provisions required for a contactless environment including activation and deactivation.
  - Contactless payment transactions between an NFC-enabled mobile phone and a POS reader use the standard ISO/IEC 14443 communication protocol. This standard is also currently used by EMV contactless credit and debit cards.
  - Furthermore, the ISO 14443 standard is available in two variants (type A and B) that differ in the way they handle radio frequency power and signal interface [19].
  - NXP's MIFARE [20] smart card uses ISO 14443 part 1 - 3 type A and is a proprietary transmission protocol instead of ISO 14443-4 [19].
2. ISO/IEC 18092 [21]:
  - Defines the possible communication modes for NFC devices operating at 13.56MHz. These are the Active (both devices are powered) and the Passive mode (i.e. Initiator is powered, target is not powered).
  - Also referred to as NFCIP-1 (Near Field Communication - Interface and Protocol Specification) [22].
  - It is based on ISO/IEC 14443 but utilizes a different command protocol to replace Part 4 of ISO/IEC 14443 [22].
  - There are three modes of operation defined in ISO/IEC 18092: Read/Write, Peer-to-Peer and Card Emulation. For the scope of this project, it is the Card-emulation mode that applies.
3. ISO/IEC 7816 [23]:
  - ISO 7816, specifically part 4 of the standard, defines the 'application layer' protocol used in a contactless transaction. This protocol is used for contact and contactless smart cards and can also be used with NFC. It specifies the application protocol data units (APDU) which are type of a message format that is used between a reader and smart card [19].

## 2.2.2 Tokenisation

Industry bodies such as the American National Standards Institute (ANSI) Accredited Standards Committee (ASC X9) [24], EMVCo [25], Payment Card Industry Security Standards Council (PCI SSC) [26] and National Institute of Standards and Technology (NIST) [27] have developed tokenisation specifications [28] to be used by the payment card industry to replace sensitive card data with a token and mitigate associated risks for merchants, acquirers, payment card Issuers, and mobile and digital payment providers.

The aim of tokenisation is to replace the static PAN in a card with a token, normally called a tokenized PAN or tPAN. A token is a substitute value used instead of the PAN. The token is a 13 to 19 digit value that, like the PAN, has to pass the basic rules of validation including the Luhn Algorithm [29]. The tokens are generated within a BIN range that has been dedicated for tokens. During a transaction, the token simply replaces the PAN, hence the existing payment infrastructure (e.g. POS terminals, etc.) does not require any changes to work with tokenization.

The following is a list of existent standards, guidelines and specifications that define tokenization and its implementation in a payment transaction:

- EMVCo, EMV Payment Tokenisation Specification – Technical Framework, Version 1.0 [25]: This framework provides guidance regarding the building and maintenance of token requester APIs, token vaults, token storage and security, token provisioning platforms and token registries. A draft version of version 2 of the EMV Payment Tokenisation Specification has been released in February 2017, however at the time of reporting, access to such document was available to subscribers only.
- PCI DSS Tokenization Guidelines [26]: is a set of guidelines for developing, evaluating and implementing a tokenization system.

Card Schemes are also releasing good practices documents and API specifications that define how other entities in the payments industry can communicate with their tokenization services. Some of these documents are being outlined in Section 4.2.1.2.

Tokenization is a process that is optional. Issuers are free to use their own security schemes that do not rely on a tokenization based infrastructure. Instead they may use static PANs, dynamic tokens or a combination of both for individual transactions. Tokenisation will also be covered in greater detail in Sections 3.2.4 and 4.3.6.

### **2.2.3 Authentication and Verification Methods**

There are various stages where verification and/or authentication of the cardholder is required in HCE. When a user enrolls a new card in the wallet the application has to verify the cardholder and during a payment the cardholder needs to be verified to authorize a payment. In card based payment methods, this was achieved through a signature or PIN.

With the introduction of mobile devices for payment EMVCo introduced the concept of Consumer Device Cardholder Verification Method (CDCVM) [30]. The concept behind CDCVM is that the mobile device is used to provide cardholder verification using new methods of verification such as fingerprint and biometrics. EMVCo does not declare which methods can be used and how such methods are implemented. The methods used are subject to availability across the supported mobile devices and approval by the Issuer and/or card scheme.

On 12<sup>th</sup> July 2016, EMVCo and the FIDO Alliance [31] signed a memorandum of understanding in order to collaborate together to develop an open, interoperable authentication standard [32]. FIDO Alliance specification is an open standard to enable simple and secure user authentication across different online and mobile services. FIDO does not define any particular method of verification/authentication but provides a protocol whereby any method of verification, can be used to provide authentication. A typical example for the use of FIDO in HCE would be in the communication between the mobile device and a TSP. The TSP would support FIDO and hence different devices using different methods of verification would be able to authenticate to the TSP through the FIDO protocol. FIDO uses public key cryptography techniques to provide an authentication framework. During registration, the client generates a set of keys and sends the private key to the server. During authentication, the client proves to the server that it is in the possession of the private key by signing a challenge. The signature is only provided if the verification method used (e.g. fingerprint) is approved. The FIDO specification version 1.0 was released in December 2014 [33] to enable two factor authentication. Subsequently in June 2015, this specification document was extended to support NFC [34].

Different methods of verification are already being used in the industry for CDCVM. Android Pay and Samsung Pay use fingerprint biometric authentication with FIDO as the underlying protocol [35]. Apple has developed a similar method known as Touch ID which is integrated in iOS devices. On 5<sup>th</sup> January 2016,

EyeVerify, Inc. announced its eye-based mobile biometric authentication technology, Eyeprint ID. EyeVerify claim their technology is fully compliant with the FIDO 1.0 Universal Authentication Framework (UAF) standard [36]. Another proposed type of biometric authentication is selfie [37]. This is being acknowledged by FIDO alliance, MasterCard and HSBC. The selfie approach typically identifies distinctive features of a face. It was reported in the media that MasterCard will incorporate the use of selfie [38], customers would require to blink for the app to ensure the image is live when customers carry out any purchases or withdrawals. This prevents someone from downloading an image for example from social network sites and use it for authentication. HSBC is also allowing customers to take a selfie to open bank accounts [39].

It is the author's opinion that in the foreseeable future, we might also see mobile device authentication incorporating other types of methods such as behavioural biometrics (e.g. metrics capturing application use, keystrokes such as keys pressed, touch gestures) and physiological (e.g. iris and face, recognition) to make the payment experience more convenient and faster.

Other standards and protocols such as Biometric Open Protocol Standard (BOPS), Trusted Execution Environment (TEE), Trusted Mobile Zone (TMZ), and Secure Enclave Processor (SEP) provide features and functions required to achieve secure authentication. [40].

## **2.2.4 Transaction Data Encryption**

Both point-to-point encryption (P2PE) and end-to-end encryption (E2EE) solutions encrypt payment data at point-of-interaction with the payment type (i.e. Swipe/MSR, Dip/EMV, Tap/NFC). In a P2PE solution, the data is encrypted/decrypted at each stakeholder (e.g. merchant, Issuer etc.) and thus prevents third-parties from accessing data while it is transferred from one end system or device to another. In an E2EE solution, the cardholder data is encrypted at the point of entry when tapping the mobile device with POS and decrypted only at the intended destination, from one endpoint to another endpoint.

The originator is responsible for encrypting the data for the receiver in both methods. Under the P2PE standard, only the transaction processor or other third party is allowed to perform key management and data is decrypted in a Hardware Security Module (HSM). Whilst in E2EE management of the encryption key management can be carried out by any party that has an endpoint such as a merchant or a service provider.

The latest version of the standard titled PCI Point-to-Point Encryption Solution Requirements and Testing Procedures Version 2.0 was published in June 2015 [41] by PCI SSC. Despite that E2EE may represent an acceptable trend such method has not yet been recognised by any standard body in the payment industry due to the following risks:

- The lack of definition of the end points, in reference to PCI DSS:
  - If E2EE encryption is provided between the terminal and the payment network or acquirer then the card or mobile device is still susceptible to a range of attacks such as skimming, eavesdropping during a transaction etc. [42].
  - Having E2EE encryption between the card/mobile device and the payment network creates an issue of key management since it would entail management of keys for every card or account in the industry. Improper key management could become a new source of data compromise.
- The security of the endpoints - For E2EE to be secure, it is important for both endpoints to be secured. Example POS terminals, especially network connected terminals, are suspect to being breached.

### **2.2.5 Card Scheme HCE specifications**

The EMVCo contactless standards do not provide any specifications regarding HCE. From an EMV point of view, there is no difference between a contactless card and an HCE device during a payment transaction. EMVCo, at the time of writing has published, in December 2016, a specification entitled "EMV Mobile Payment - Software-based Mobile Payment Security Requirements Version 1.0" [43]. The objective of this specification is to provide guidance and generic security requirements for Mobile Wallet applications that provide payment transaction capabilities without the use of a secure element. This is the only specification provided by EMVCo pertaining to HCE.

The EMV contactless specifications [15] that HCE wallet App developers have to take into consideration when developing applications are:

- EMV Contactless Book A - Architecture and General Requirements.
- EMV Contactless Book B - Defines the entry point requirements (e.g. Application selection, etc.).
- EMV Contactless Book C - Kernel Specifications - Differs between the different type of cards and relates mostly to the POS requirements. Used by HCE developers when developing communication with the POS.
- EMV Contactless Book D- Contactless Communication Protocol.

These 4 books provide the specifications and requirements for contactless transactions. The same specification and requirements apply to HCE.

The payment card schemes have chosen to define their own specifications and requirements for HCE wallet app developers that intend to develop wallets for their contactless card types. A list of some of the specifications provided by VISA pertaining to HCE can be traced in Section 4.2.1.1.

Other card schemes such as MasterCard also provide similar specifications and requirements. The MasterCard and Visa implementations are both compliant with EMV standard to ensure compatibility with installed EMV contactless POS terminals. These specifications were used in Section 4 for designing and implementing the HCE model.

## **2.3 Mobile devices and Host Card Emulation**

A payment conducted with HCE requires a mobile device with NFC capabilities. The availability of NFC-enabled mobile devices continues to grow, with NFC included in over 690 devices [44]. It has been estimated that more than 100million people [45] around the world will use an NFC handset to make a purchase during 2016. The value of transactions conducted via NFC handsets will grow from US\$30bn in 2016 to US\$45bn in 2017, up to US\$240bn in 2021 [45].

There are many markets players in the industry launching their own payment services such as Samsung Pay, Apple Pay, Android Pay and Microsoft Wallet. There are also banks and major companies launching their own mobile payment app. NFC and HCE payments will continue to strengthen when more commercial implementations of HCE enabled payments will be rolled out. HCE is currently supported by Android OS, version Kitkat 4.4 and higher, Blackberry 7OS and higher, and recently Windows version 10 OS.

In this section, first an introduction on NFC is provided specifically the different modes available and those used for HCE. Next the architecture that makes HCE possible in a mobile phone is introduced and finally different models of HCE are provided.



### **2.3.1 Near Field communication**

NFC is a communication technology based on close magnetic field as a medium for transmission. Unlike most wireless communication technologies that rely on electromagnetic radiation (i.e. radio waves), NFC relies on electromagnetic induction. This is why it can only operate at very short range, typically a few centimeters. NFC inherited electromagnetic induction communication from RFID. Electromagnetic induction allows a receiver to 'harvest' a limited amount of energy from the transmission thereby a receiver does not require any battery to operate. This is why contactless credit/debit cards do not require a battery to operate.

The basic architecture of both an NFC reader and smartcard is made up of an antenna, a transceiver chip and controller. In a reader the controller is normally separate from the transceiver but in most tags and smartcards the controller and transceiver is incorporated into one chip. As from 2011 mobile device manufacturers such as Samsung, Nokia, HTC and BlackBerry started to integrate NFC chips and antennas in mobile devices, (further detail found in Appendix 1). In the mobile device the NFC transceiver is connected to either a secure element or to the host controller. Since a mobile device already contains a battery, a mobile device can act as a reader and also as a smartcard.

NFC works at 13.56 MHz and its interface specifications is provided by ISO/IEC 18000-3 [46]. The communication channel provides a data rate of up to 424kbit/s (~50KB/s). In a typical NFC transaction a reader initiates the transaction by generating an RF field which is detected by the target such as a smartcard or a mobile device. When the target is not powered (e.g. smartcard), NFC is said to be working in a Passive mode while if both the reader and the target are powered (as in the case of HCE) then NFC is said to be working in an Active mode. In passive mode, only the initiator (i.e. reader) generates the field while the target communicates by varying the amount of power it harvests from the electromagnetic field (i.e. load). This allows for a full duplex communication channel between the initiator and the target. In an active mode, both devices generate their own magnetic field. Amplitude Shift Keying (ASK) modulation is used on both sides and to ensure that no collisions occur. The device receiving data switches off its field while the other is transmitting. Data in NFC is transmitted according to the NFC Data Exchange Format (i.e. NDEF) protocol [47].

Furthermore, the standard identifies three possible operating modes or scenarios. These are reader/writer, peer-to-peer, and card emulation. The reader/writer mode is used with contactless cards and smartcards in passive mode, the peer-to-peer mode is mostly used to transfer files and information (e.g. a business card) between mobile phones while card emulation is the mode used during HCE. Card emulation can be used for other purposes apart from HCE such as when a mobile device is used to emulate an access smartcard.

### **2.3.2 Host Card Emulation architecture**

Host card emulation (HCE) is an on-device technology that enables NFC enabled mobile devices to emulate a payment card in order for a user to perform contactless payments. Prior to HCE, the NFC transceiver was connected directly to the SIM card in a mobile device known as a Secure Element. The payment application containing the payment credentials (i.e. secret cryptographic keys) reside on the secure element. The mobile device's processor does not take part in the payment transaction and all sensitive communication is done between the NFC and the SE. The fact that the SE (SIM) is owned by the mobile operator has created some barrier as no access is provided by the SE to host third party applications. Access to store or use the SE must be requested from the mobile network operator (MNO) and/or trusted service managers. Due to these dependencies, Google implemented a software based payment card emulation solution, (i.e. HCE) based only software [9]. In contrast with SE based systems, HCE eliminates the need for a hardware based SE as the mobile device operating system (the device host) hosts the payment application.

Card emulation is based on ISO-IEC 14443-4. The HCE communication process between the NFC Reader and the mobile device is described in Figure 1:

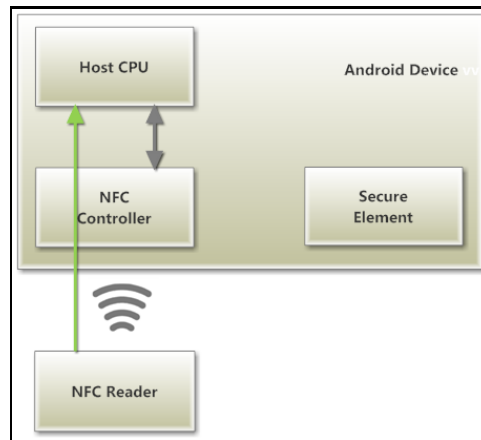


Figure 1 – Android HCE [48]

The HCE architecture consists of:

- i. **Host CPU:**  
This is the location where consumer applications are running on the main processor and serves as a user interface for functions such as selecting a card and initiating a transaction.
- ii. **NFC controller:**  
This is the contactless hardware front-end which acts as a gateway between the reader and the CPU on the device.
- iii. **NFC reader:**  
This is a type of electronic-transaction terminal where signals can be sent and received for contactless payment systems. A contactless symbol present on the reader indicates compliance with EMV Contactless Communication Protocol.

When a user taps a mobile device on a POS terminal both devices will go through a “handshake” procedure. During this initial setup, the POS will send the Applet ID (AID). The mobile device uses the AID to determine where to route the incoming transaction. When a user installs a wallet app and adds a card, the app will ‘register’ a service with that AID and inform the OS that any ‘transactions’ related to that AID should be routed towards its service. Every card may support more than one AID normally referred to an AID group. These AID groups are publicly known but new AIDs can be registered (for new cards) and the whole procedure is defined in the ISO/IEC 7816-5 [49] specification. It is vital that card Issuers follow such specification as it will avoid collisions with other applications.

HCE Implementations vary between different mobile platforms (e.g. Android versus Windows Phone) but in essence they all follow the same procedure. The actual service runs in the background and does not need any user intervention to execute, except in cases where the user is required to provide some form of authentication or verification. Android devices do not allow transactions to occur when the screen is off while Windows Phone allow transactions to occur even if the screen is off. This possess a risk since an attacker can read sensitive data, through NFC, from the mobile device even when the device is in a pocket or handbag simply by bringing a NFC reader device close to the victim’s mobile device without the owner being aware of such theft.

The software used by the mobile phone to make a payment transaction takes the form of a service running in the background. A mobile device can have more than one card type (e.g. Visa, MasterCard, etc.) associated with it hence a number of these service can be running in the background. When the mobile is tapped on a POS reader, the device's OS receives a specific number (i.e. AID) from the terminal instructing it to call a particular service, according to card type and pass the NFC communication to that particular service. The service is expected to implement the transaction processing according to the kernel specification of the card it is developed for. More information on the different kinds of implementations for this background service are provided in the next section.

### 2.3.3 Models of HCE

All payment cards store some form of data. Starting from simple mag-stripe card, to chip-and-pin and contactless cards, they all hold data related to an account/cardholder such as PAN, name, expiry, CVC etc. The data varies between different types of cards but essentially, such data should be kept as secure as possible. Furthermore chip-and-pin and contactless cards not only store data but they also have cryptographic capabilities whereby they can generate signatures, encrypt data, etc. Therefore, an HCE service should be able to store data securely and operate cryptographic functions on that data. Based on these two requirements, several models of implementation have been proposed and below is an overview of the different possibilities.

The different implementations are made up of 'modules'. Some modules are on the actual mobile device while others exist on the cloud.

**i. User Data:**

Credentials e.g. PAN, expiry, cardholder name and master keys, used to create the cryptogram (i.e. a digital signature). Note that the card will never communicate via NFC the keys used to generate the application cryptogram, but it will only communicate the cryptogram. In a traditional card based system, the credentials were stored on the magnetic card itself.

**ii. Service applet (agent):**

The service that is responsible for communicating, through the NFC controller with the POS. It is responsible for sending the credentials to the POS and generating the application cryptogram or dynamic CVV required during an online transaction. The applet is also referred to as the mobile wallet application.

**iii. HCE API:**

This is more commonly known as Hardware Abstraction Layer (i.e. HAL) or driver. It is an intermediate layer residing in the mobile OS. Its function is to provide a universal API (i.e. set of functions) to mobile app developer irrespective of the NFC hardware being used. This allows apps to be developed that work on different NFC hardware.

Several different models have been proposed by industry experts. A brief explanation can be traced in Figure 2.

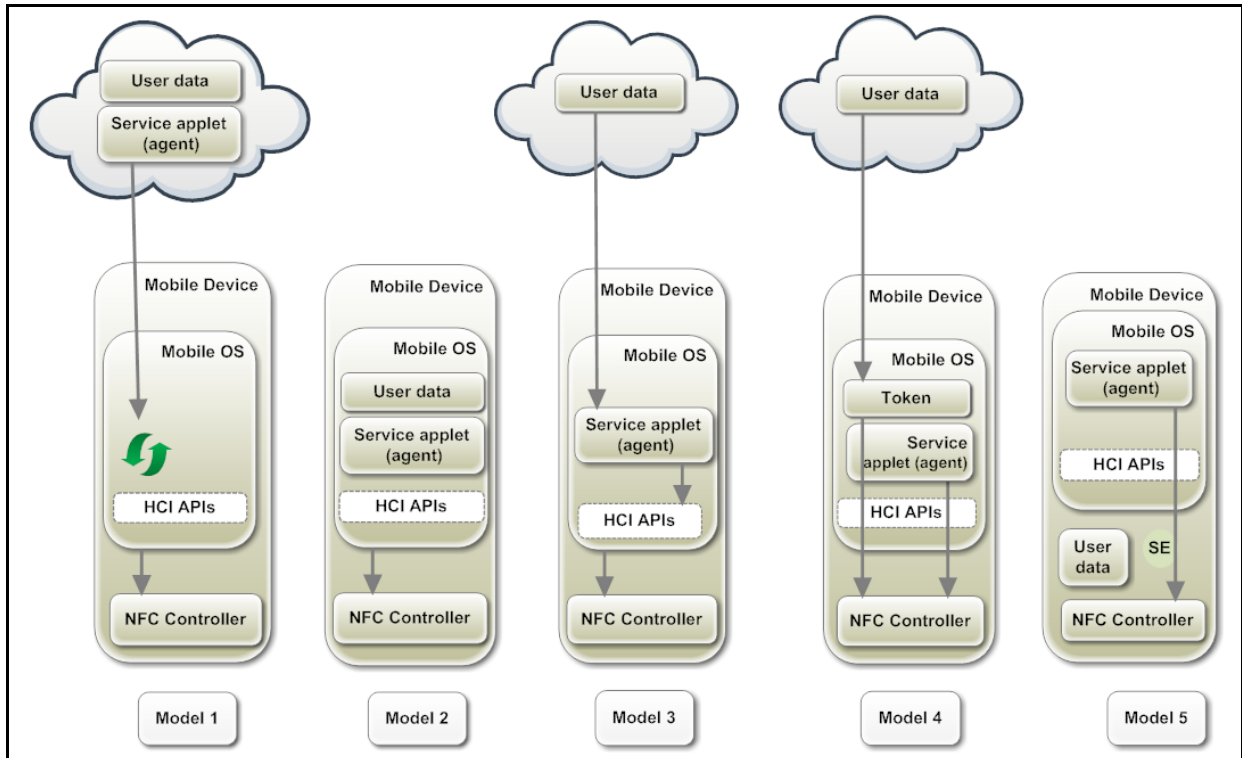


Figure 2 - Models as proposed by Industry Experts [48]

**Model 1:**

In Model 1, both the user data and the applet are stored in the cloud. The mobile device, reads the transaction data from the POS, and sends this data to a cloud applet. This in turn obtains the user data from the cloud and generates the cryptogram. This is sent to the mobile device which in turn communicates it to the NFC controller finally reaching the POS via NFC. Note that the device still has a small ‘dumb app’ whose role is to send and receive data between the HCE API and the cloud.

Such model requires that an internet connection is available for every transaction. As a result a user might experience network latency. Furthermore, this would make it more difficult to identify relay attacks based on time delay as this model requires a longer time window to accommodate for network latencies.

**Model 2:**

Similar to Model 1, both user data and applet are stored on the device itself. It is very insecure as the data is stored on the mobile which is vulnerable to attacks and could also be physically stolen.

**Model 3:**

User data is stored in the cloud and the applet resides on the mobile device OS which is beneficial as no performance and network issues are present. It simply means that the PAN and the keys are not stored on the device but retrieved from the cloud when in need. Another strategy could be to download or cache the data from the cloud before the transaction. That way it would further reduce network latency and it will be transparent to the POS. Such model requires that an internet connection is available for every transaction.

**Model 4:**

User data is stored in the cloud along with a set of tokens. Tokens are random numbers, replacing a traditional PAN that expire and become useless after some set time according to some rules. The cloud keeps a mapping of PAN to Tokens. In this case, the user data (i.e. PAN) is never sent to the device but a

token is sent instead. Later during the transaction the token is 'detokenized' by the payment infrastructure. An internet connection is required to download the tokens (i.e. token replenishment) but this does not need to be during the transaction as tokens can be cached.

#### **Model 5:**

The user data is stored in the secure element, (SIM card or specialised hardware) on the phone. The service applet can communicate with the NFC controller but it can never read the user data from the SE. When, user data is sent during the payment transaction, the SE will communicate directly with the NFC controller and pass this data directly to POS reader. More importantly the cryptogram is generated in the SE, thus the key is never sent. This way, if a mobile device is compromised (malware, etc.) it would be very difficult that the user data is stolen as this is simply not available at the OS level.

With this model, Issuers have to depend on either MNOs (e.g. as in the UICC-based SE case) or the manufacturers (as in the embedded SE case). This makes it difficult and expensive for actors in an NFC ecosystem to interact efficiently.

Although Models 1, 2, 3 are rarely used in today's current implementation they are options for implementation. Model 4 and Model 5 are the closest commercial implementation in the current markets.

## **2.4 Stakeholders roles in a Mobile Payment Ecosystem**

The mobile payment ecosystems is made up of many stakeholders, each assuming different roles and responsibilities. The major stakeholders that make up the HCE Payments Industry are:

### **i. Customers/CardHolder:**

- Authorised users that hold a debit, credit or charge card issued by Issuers and are able to carry out an HCE NFC payment using their mobile device. A wallet is required to carry out such task. This can be downloaded from an app store, which is normally managed by the OS provider for the mobile device. The app is published by the payment application provider typically by the Issuer.

### **ii. Merchant:**

- Any business entity such as retailers, transportation providers and vending machine providers that are authorised to accept mobile payment for the goods and services.

### **iii. Acquirer:**

- These may be a bank, or a third party payment processor/s that process the payment transactions for the merchants, via the payment networks through to the Issuer. Typically, the relationship with the merchant is negotiated and translated in a financial contract/service agreement. The acquirer bank may also supply a payment processing terminal known as contactless POS terminal.

### **iv. Payment Network:**

- An entity that manages and facilitates the flow of transactions, the clearing and settlement of card payment transactions between acquirers and Issuers.
- Examples of payment network brands include Visa, MasterCard and American Express.

### **v. Token Requestor:**

- Acts as an intermediate entity between cardholders and Token Service Provider (TSP).
- Collect payment card details from the cardholders and submit the information in a token request to the TSP.

- Requests payment tokens on behalf of the cardholders upon registering with a TSP and provisioning them in a mobile device or remote location in a cloud.
  - The following are the typical parties that can act as token requestors – digital wallet providers, Issuers, card-on-file merchants, acquirers, acquirer processors and payment service providers operating on behalf of merchants.
- vi. Token Service Provider (TSP):**
- The TSP can be an issuing bank, payment network or independent third entity that provides tokenisation services and payment tokens to token requestors.
  - Responsible for payment token generation and issuance and managing the payment token's lifecycle.
  - The TSP maintains the mapping between card numbers and expiry dates, PAN and other payment card confidential data and payment tokens and their respective expiry dates.
  - It oversees the operation and maintenance of token vault stored in a PCI-compliant environment, deployment of security measures and controls and the registration process of allowed token requestors [1].
  - Responsible to perform due diligence and registration functions for Token Requestors and assigns them with a unique token requestor ID.
- vii. Issuer:**
- Is a bank that issues payment cards to a customer and are responsible for billing and payment of transactions from the respective customer's account.
  - Issuers may also publish their own mobile payment application on the app store.
  - Issuers can setup a relationship with TSP for provisioning of tokenisation services. However, they are also free to use their own security schemes such as use of static PANs, dynamic data (e.g. session keys) for individual transactions.
- viii. Mobile Wallet App Developer:**
- A mobile wallet is an app that can hold one or more cards. Issuers may develop their own mobile wallet app or else they could accept that their card be used in other wallet apps. Typically mobile device OS vendors provide a wallet that could 'host' many cards from different vendors (e.g. AndroidPay). Therefore the developer of the mobile wallet app could be an Issuer, a third-party working for the Issuer, mobile device OS vendor or a third-party with no direct connection to Issuers or mobile device OS vendors.

Figure 3 illustrates the involvement of every stakeholder during the life cycle of a payment transaction.

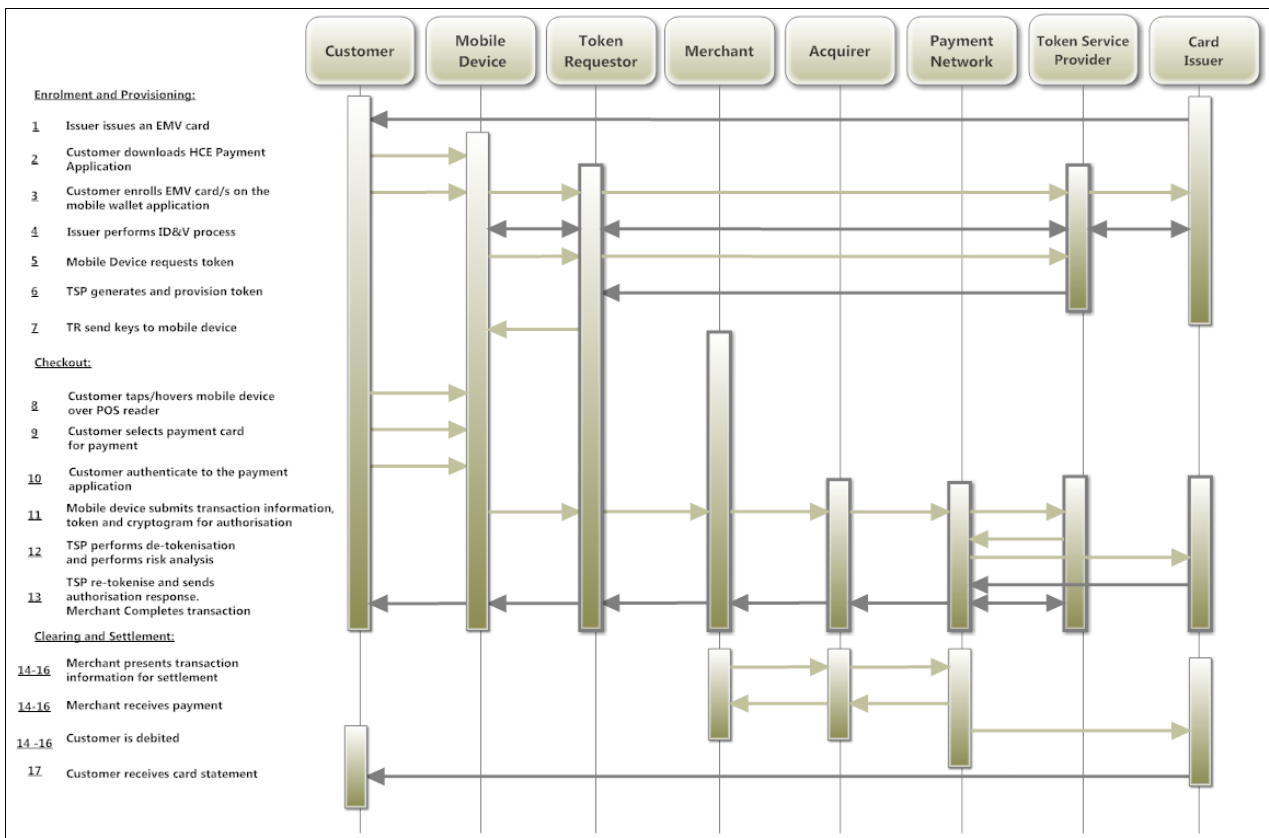


Figure 3 – Credentials and Payment transaction life cycle

## Enrolment and Provisioning

1. The process starts with the card Issuer sending an EMV card to the customer or equivalent credit card number through a secure web-based service such as internet banking. A payment account for the credit/debit card is linked with the EMV card at the issuing bank.

Typically the payment card is provisioned with application identifiers known as AID's, customer specific data, specific transaction rules such as cardholder verification method, online/offline authorization and authentication and other security information according to the Issuer's policy.

2. Since the cardholder will pay using the mobile device, the cardholder must ensure that the mobile device supports near field communication and downloads a mobile wallet application (e.g. Android Pay) or a proprietary HCE mobile application (e.g. Barclaycard app).
3. Cardholder then registers the EMV card information in their mobile wallet application residing in the mobile device operating system, either by using the camera to capture the card information or entering it manually. Multiple payment cards can be maintained on the mobile wallet application.
4. Issuer then verifies the cardholder's identity and mobile device before provisioning the actual payment credentials to the mobile device. Different methods of ID&V<sup>1</sup> can be used. One typical method is for the Issuer to send an OTP to the cardholder's registered mobile phone number.
5. The mobile device will request a token to be generated and provisioned. The request is made to the token requestor, which in turn requests the token from the TSP.
6. The TSP generates a tPAN (Tokenized PAN) and a tUDK (Tokenized Unique Derived Key) and sends it to the token requestor (TR).
7. The token requestor (TR) generates a set of LUK (Limited Use Keys) using the tPAN and tUDK and sends the tPAN and the LUKs to the mobile device.
  - a. The tPAN eventually replaces the card primary account number (PAN) and expiration date with numeric codes of the same length with a randomly generated token. Separate ranges of numeric codes are allocated so that no payment token can be reversed engineered to find the related PAN number. At most, a mobile wallet usually has five to fifteen stored LUKs on it at one time [1]. A token can be restricted to be valid depending on the risk appetite defined by the Issuer e.g. can be valid for a single transaction, changed at set intervals.
  - b. Tokens may be generated and downloaded to the device 'on-the-fly' when a user is making a payment or else they can be downloaded and stored in a secure location on the mobile device called the TEE and used later during a payment transaction.
  - c. Mapping a token to the original payment card information (PAN) is limited to the TSP or to the issuing bank. The mapping is stored in a TSP token vault.

## Checkout

8. The cardholder hovers or taps the mobile NFC device with the merchant's contactless POS terminal.
9. If more than one card of the same type have been enrolled the mobile device will ask the card holder to select which card is to be used for payment.
10. Optionally, the cardholder is asked to authenticate to the mobile device (e.g. through a fingerprint). Note this processes can happen later in a two tap approach or other methods can be used for authentication such as entering a PIN on the POS.

---

<sup>1</sup> Identification and Verification of Customer (ID&V) is the processes of verifying the cardholder with the issuer



11. The mobile device generates the required data and sends the data to the Merchant's POS:
  - a. **QVSDC mode** – in this case the mobile device generates an application cryptogram based on the transaction data provided by the POS. The cryptogram is generated using the LUK (provisioned earlier) as the key.
  - b. **MSD mode** – in this case the mobile device generates a signature (dynamic CVV) using the ATC (as dynamic data).
12. The POS forwards the transaction data to the acquirer, then to the payment network. At this point, the payment network detects the transaction is using a token and asks the TSP to de-tokenize the transaction. The TSP maps tPAN to its PAN and sends the PAN back to the payment network. The Payment network forwards the transaction (detokenized) to the card Issuer for authorization.
13. The Card Issuer generated an authorization response and sends it to the payment network. The payment network asks the TSP to re-tokenize the transaction data. The payment network then forwards the data back to the Merchant's POS. Merchant completes the transaction based on the authorization response provided.

### **Clearing and Settlement**

14 -16. This leads us to the "clearing stage" which involves the merchant depositing the transaction purchases information to the acquirer. The transaction is routed to the respective payment network. Example visa transactions to the Visa network, and so forth. The payment network pays the acquirer and in turn pays the merchant for the goods or services originally purchased by the cardholder. The payment network then passes the transaction to the card Issuer and the cardholder's account is debited. Throughout the transaction process the appropriate interchange fees are deducted by the respective stakeholders.

17. Cardholder then receives the bank statement summarising all the financial transactions.

### **3 An analysis of attacks on traditional and HCE based contactless payment schemes**

This chapter will be covering relevant existing literature encompassing attacks on both traditional and HCE based contactless payment implementations. The traditional attacks will include attacks at physical, application and key management and cryptography. The literature review with respect to attacks relating to HCE will cover attacks at Operating System, secure memory areas, consumer device cardholder verification and tokenisation and its infrastructure.

#### **3.1 Literature review of attacks on traditional contactless payments**

Contactless technologies including RFID and NFC have been used for payments and other uses for quite some time. The use of these technologies for payments has driven the interest on studying the weaknesses associated with these technologies and their implementation in the payments industry. This section provides an overview of the research done with regards to contactless payments.

##### **3.1.1 Attacks at the Physical Layer**

###### **3.1.1.1 RFID Skimming**

RFID skimming is also known as NFC Skimming or Electronic Pickpocketing. RFID skimming occurs when an unauthorised RFID reader interacts with the card or similar device without the owner's knowledge for the purpose of stealing information. The aim of which is normally to be able to replicate the information onto a fraudulent card for theft or malicious reasons. The skimming process was common in magnetic swipe cards where 'skimming devices and a camera' are installed in ATM machines to capture card data from the magnetic stripe when customers insert their card into ATMs. While the process and the hardware for RFID skimming is different than ATM skimming, the aim remains the same, stealing information stored on the card to make fraudulent transactions on a victim's account.

Kirschenbaum and Wool [50] showed the development of a stand-alone portable RFID skimmer that successfully read the contents from an ISO14443 RFID tag from a distance of approximately 25cm using a lightweight 40cm-diameter copper-tube antenna. With such a distance, the attacker does not need to touch or bump into the victim but merely stay close. Using hobbyist electronic supplies and tools, the authors managed to build a skimmer with a total budget of approximately \$100. The authors outlined that additional controls such as physical shielding inside a Faraday cage, cryptographic application-level access controls, and an actuator (switch to activate the RFID) are needed to prevent simple RFID tags being skimmed. Despite that this study was conducted on RFID tags, contactless cards use the same physical layer and thus are also susceptible to the same kind of attack.

Complementing this section, two cases were traced where card skimming thieves stole credit card information to make fraudulent purchases from people in a public place using RFID wireless technologies from RFID enabled credit card carried in pockets and purses as follows:

- i. In February 2016, a scammer photographed in Russia by a man named Paul Jarvis showed on Facebook, how a payment device (i.e. POS) was used to read contactless cards in the victims' pockets and bags without the victims knowing, whilst traveling on the train [51]. The scammer can then use the money transferred from these victims to make purchases of less than a certain value without needing the victim's signature or PIN.
- ii. In October 2015, Mr. Perez reported that a thief had bumped into him on the train and the thief managed to steal £20 through an unauthorized contactless payment [52].

One important aspect worth mentioning in NFC skimming is the fact that, in developing 'skimming' devices, contrary to commercial approved readers, the attacker can transmit signals at higher levels than that mandated in ISO 14443 and therefore can achieve higher distances [50].

Heydt-Benjamin et al, [53] used an off the shelf ISO14443 reader to skim credit cards and steal information such as cardholder name, number, and expiration date stored on the card. The authors make also reference to the fact that banks normally distribute contactless cards using regular mail and therefore side road mail-boxes can be an ideal 'scenario' to mount a skimming attack prior to the card is distributed to the owner. This attack is known as the Johnny Carson attack since the attacker can determine the 'contents' of the card without actually opening the envelope it was posted in.

Alfaraj [54], further developed the work of Kirschenbaum and Wool [50], by improving the antenna design. The author managed to achieve a skimming distance of 1.2 meters under ideal conditions. One important consideration is the fact that in all of these cases the authors used ISO14443 cards which are passive devices, meaning they need to be powered by the reader's H-Field signal output. HCE mobile devices do not have this requirement and hence the distance and the success of mounting a skimming attack might be higher. On the contrary, most mobile phones with HCE implementations, turn off their NFC capabilities when the screen is off [55] thereby limiting 'skimming' attacks only when the mobile device is being used by the user. However, such implementation is different in Windows based mobile phones where the phone can make NFC payments when the screen is off [56]. Furthermore, mobile phone payment apps can implement some form of notification (sound, pop-up, etc.) event which is triggered whenever a payment transaction is made. In a skimming attack, the event would serve as an alarm for the customer.

Finally, another type of skimming could be done on mobile phones in HCE mode. Without using any eavesdropping, an app first emulates a POS and allows someone to tap a card or an HCE device to the device running the app and then the data extracted is stored in the device's memory. Later, the device changes its function from a POS to a card (HCE) and when it is tapped on a real POS it provides the data extracted previously from the card or HCE device. SpotMe app [57] has been developed around this principle. Despite that the app is not intended for malicious reasons as it is aimed at friends who simply would like to 'pre pay' an item for another friend, the logic behind it could be used by an attacker to skim and replay an offline payment transaction.

In addition to RFID skimming, RFID technology is also susceptible to eavesdropping.

### **3.1.1.2 Eavesdropping**

Eavesdropping is the process of using a device to 'listen' on a communication event that is happening, between a card and a reader at a certain distance. Contrary to a skimming attack, the attacker does not need to provide power to the card and the card is unaware that the attack is being mounted. However, the attacker has a very limited window of time to mount the eavesdropping attack as one has to wait for the victim to use (tap) the card with the reader for a payment. A typical contactless transaction takes around 500mS [58] and the attacker has to 'capture' the signal during this small period of time. This means the attacker has to be at the right place and at the right time to mount the attack.

Hancke [59] claims a system with specialized RF equipment will almost certainly be able to eavesdrop from further away than an attacker with standard equipment such as commercial NFC Readers. The author conducts a proof-of-concept eavesdropping attack against HF RFID devices using the ISO 14443A/B and ISO 15693 standards. During a transaction, a reader (master) communicates with the card (slave) to exchange data over the NFC channel. This communication is split into 2 channels. The forward channel is when the reader sends data to the card, and the backward channel is when the card is sending data to the reader. In most studies, the channels are treated separately, but most of the time, the backward channel is the most important. It is also the most difficult to eavesdrop as the card is a passive device and hence its

signal output strength is less than that of the reader. Hancke [59] used an RFID reader, RFID token and Active h-field antenna. The author positioned the reader and the token at a practical distance and then placed the antenna at the same height level. Throughout the test, the antenna was kept at a fixed position while increasing the distance from the antenna to the token and reader in steps. Using an oscilloscope and Matlab software, the data signals from the receiver were captured and recorded. The author then demodulated the data and recovered the original data sent. The author noted that the result of such experiment is affected by a number of factors such as environmental conditions e.g. noise, RF equipment, barriers, amplitude modulation, which eventually during the analysis stage have to be accounted for. The author reported eavesdropping distances of between 1m and 3m, successfully recovering both the forward and backward channel. A distance of 5m was achieved but reading the forward channel only.

It is important to note that for HCE, the concept of forward and backward channel does not exist, it exists in Peer-to-Peer mode which operates under ISO 18092 [21]. Despite this, the claim of 5m by Hancke [59] as a possible distance for eavesdropping on HCE based transaction still holds. This is because, based on ISO14443, which is the standard in which HCE operates [60], both the reader and the device use the same modulation effectively creating a 'forward' channel for both reader to HCE device and HCE device to reader.

Diakos, Brown, Wesemeyer and Briffa [61] proved how reliable information from an ISO 14443 Type A device could be recovered by an eavesdropper through analysing the frame error rates over various distances up to 100cm using a covert antenna and low-cost electronics. Unlike the previous studies mentioned above, the authors did not use any special antennas but instead used a small loop-antenna which is easily concealed (i.e. by wearing it) and a shopping trolley. These antennas would not raise suspicion at a POS station. The authors proved that eavesdropping distances between 20cm to 90cm were achievable. Having a shopping trolley 90cm away from a POS reader is not uncommon in a supermarket. One key finding in this research was the fact that the power output from readers (termed H Field strength) was higher for some readers compared to others and the fact that in an HCE transactions, both the reader and the device are active devices (powered), the possibility of eavesdropping further increases.

In another study, Brown, Diakos and Briffa [62] carried out a similar study using a 5m wire, worn on the body and using a shopping trolley as antennas. Distances of between 90cm and 40cm were achieved. The study also showed the importance of background noise which serves as a 'protective layer' to eavesdropping of NFC devices.

Contrary to the mentioned authors, Pfeiffer, Finkenzeller and Biebl [63] took a completely theoretical approach to analyse much of the claims presented in their study. The author modelled most of the components related to an eavesdropping attack and calculated several theoretical distances at which eavesdropping could occur in different conditions. The authors stated that 1m to 3m distances are achievable in typical 'business' conditions while higher distances are only achievable in 'RF Noise free' environments which are impractical in reality.

### **3.1.1.3 Relay Attacks**

Given that eavesdropping is possible at lengths which are practical enough to be used in a typical payment scenario (e.g. near a counter in a shop) and skimming is also possible, the ability to mount a relay attack is trivial. The concept behind the relay attack relies on a fast digital communication to a device which can emulate a card/token. Given this project focuses on HCE, it is worth mentioning any HCE enable device can be used to mount such an attack.

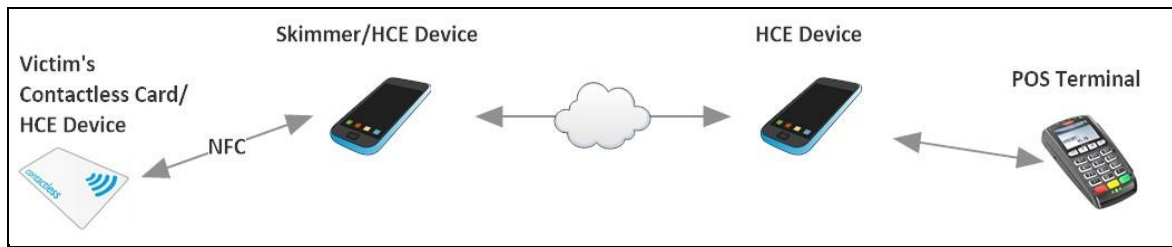


Figure 4 – Relay attack process flow diagram

The process to mount a Relay attack goes as follows:

1. A person holding the skimmer/mobile device approaches the victim. The victim would have the card/mobile device held in a pocket or wallet.
2. Another person holding another HCE device approaches a POS reader, orders an item and proceeds to pay with his HCE device. The reader makes a series of communication requests (i.e. data such as PAN, etc.) via NFC to the HCE device.
3. The HCE device then, communicates via a high-speed connection with the skimmer/HCE device held by the other person.
4. The skimmer makes the payment request to the card held by the victim, without the victim's knowledge, and communicates the reply back to the HCE device.
5. The HCE device forwards the reply back to the POS via NFC. The POS is fooled into thinking it is communicating with the victim's card.

The attack in Figure 4 is possible as no PIN or any form of authentication is being used and hence the victim is unaware of the attack. If a PIN or any other form of authentication is requested then this attack would not work. In the UK, the limit is currently set to £30 and therefore it makes this attack possible for purchases smaller than this limit.

Another limitation to such an attack is the latency introduced by digital communication. In the diagram above, when the HCE devices communicate, the latency could be higher than that allowed under the ISO14443 for example due to network latency, and therefore the transaction would be refused. ISO/IEC 14443-4 specifies a Frame Waiting Time (FWT) of 500uS to 5s [64]. It is up to manufacturer to choose the respective FWT. 500uS would certainly limit the attack possibility in certain architecture, especially software based, while 5s, would practically allow any architecture to work.

Hancke [64] describes a system with an ISO14443 transceiver<sup>2</sup> at both ends (i.e. POS terminal and victim contactless card) and a wireless data link between them. This type of link introduces virtually zero latency and is transparent to the reader. The author successfully mounted a relay attack having the victim 50m away from the POS reader.

Francis, Hancke, Mayes and Markantonakis [65] propose a system similar to that shown in Figure 4 but using mobile phones (i.e. SE based) on both ends with a Bluetooth connection between the mobile devices. At the time of the research the mobile phones only supported SE based. This limited the success of the attack to a certain extent, as the attacker is 'bound' by the operating system and the hardware implementation. Example, some data flows directly from the NFC controller to the SE and vice versa. Therefore the attacker would not have access to the data at the application layer to be 'relayed'. If an attacker would be able to obtain full control of the processor of the mobile device, then it would be possible to communicate directly with the NFC controller.

<sup>2</sup> A transceiver is a device comprising of both a transmitter and a receiver that are combined in a single package and share common circuitry.

Roland, et al, [66] took a different approach towards a relay attack. The authors installed a Google Wallet app on an Android phone. The authors proved that, with certain privileges provided to the app, the app was able to access information from the google wallet applet on the secure element. The app then opened a TCP connection to a remote PC which had a USB (i.e. ACR 122U) NFC transceiver connected to it. The remote PC then had a software that would receive the information from the app running on the mobile Android device and then execute a transaction through the USB NFC. The authors notified Google about this issue and claimed that this vulnerability was fixed in a new version of Google Wallet.

Issovits and Hutter [67] propose a similar study but instead of just having a basic relay infrastructure, the authors exploited a vulnerability in the ISO/IEC 14443 standard which is the Waiting Time Extension (WTX) command. This command can be sent to the reader to request an extension to the delay time in order to relay the data and the transaction would still be considered as valid. The authors also claimed that the ISO 14443 does not enforce systems to have countermeasures against relay attacks such as time limits that make mounting a relay attack harder for the attacker. As part of their study, the authors have also proposed a number of possible countermeasures to limit relay attacks while still being compliant with the respective ISO standard.

In a more recent study (2015), Van Den Breekel [68] utilized HCE without the SE to implement a relay attack. The author used two smartphones connected to each other using WIFI to implement the relay attack. The author also developed an 'optimized' version of the attack. In the optimized version, some communication terminal requests (e.g. AID, get processing options command) was cached. When information is cached, the use of the WIFI channel is minimised and therefore limiting the delay/latency that could occur due to the WIFI channel. The optimized relay attack added less than 200mS to the non-relayed (normal) transaction time. This is less than the variance observed when comparing the transaction time taken by different card scheme cards. Thus if timing was to be used as a monitoring basis to detect relay attacks it would fail to detect such an attack. Hence, the author claims that countermeasures based on 'delay sensing' are not effective against such an attack.

Vila and Rodriguez [69] also propose a system similar to Van Den Breekel. The authors also used two Android phones in HCE mode to mount the attack. The authors proposed a 'conceptual' distributed attack where one could use a network of dishonest readers, these being Android mobile devices with malware software that relays card data to one central reader upon detecting a contactless card in the vicinity. Given the maximum of 5 seconds time windows provided by ISO 14443-4 the authors claim the attack could be mounted even by using internet (3G/GPRS) communication between the mobile phones. In fact, the authors claim to have run the attack when the reader was in New York relaying information back to the other HCE mobile phone in Madrid (Spain). EMVCo defines the maximum time allowed using the FWT (Frame Waiting Time) parameter and recommends an FWT max of 37mS [70]. The authors recommended that a FWT of 37mS should be a mandatory requirement [69] rather than a recommendation to reduce the risk of a relay attack.

#### **3.1.1.4 Data Corruption, Modification and Insertion**

While many of the attacks are developed for financial gains, some attacks could have a different aim. Data corruption is one type of attack, where the aim is to interfere, disturb or block a transaction from taking place, commonly known as a DoS attack. A data modification attack consists of an attacker capturing data and manipulate it when being transferred across NFC enabled devices but such data still remains valid. Such attacks could prevent users from making a payment in places such shopping establishment or in a subway station to cause chaos. In NFC, these type of attacks could be split into two NFC Jamming and NFC Zapping.

#### **3.1.1.4.1 NFC Jamming**

NFC card and readers use air as a channel in which a magnetic field is used to transmit data. If that channel is in some way disabled, then cards and readers will not be able to communicate. In NFC, the typical transaction occurs between a POS reader and a card. The reader is an active powered device while the card is a passive device. Under this scenario, the reader modulates the signal by varying the amplitude of the signal. The card, as a passive device is not powered and thus harvests power from the reader. In fact, the card modulates its signal response by varying the load depending on how much energy it harvests [71].

Oren, Schirman, and Wool [72] developed a device that would transmit a signal at 14.408 MHz, at upper frequency channel in NFC used by the card to load modulate its signal. This signal would interfere and essentially corrupt the data that the card would send back to the reader. The authors used commercially available antennas to transmit the jamming signal and corrupt transmissions from 2 meters away from the POS reader.

Gummeson, et al, [73] in their research developed a device that uses the 'jamming' principle to block malicious behaviour according to defined blacklisted rules, as to protect a mobile phone from external NFC threats. The device consist of a small card/sticker that is attached at the back of the device. The device harvests power from the NFC being transmitted and is programmable to stop/block only the NFC transactions the user wants to block

Armourcard [74], another commercial product on the market that claims to protect against skimming attacks using NFC Jamming. Armourcard claims [75] the device has a touch interface which is used to indicate if the signal should be blocked or not. The device senses NFC signals and when no touch event is registered a 'blocking' signal is generated to block the skimming attack.

#### **3.1.1.4.2 NFC Zapping**

Another method of attack is to totally disable the chip on the card. This process is known as zapping. Since contactless cards do not have a battery, the cards have to harvest power from the 13.66MHz signal provided by the reader. Oren, Schirman, and Wool [72] explain a method of 'over powering'. The device using a simple camera flashlight connected to an antenna. The flashlight electronics generate a high voltage (>250VDC) and when this is discharged into the antenna held near a contactless card, the magnetic field generated is large enough to damage the electronic circuit in the card rendering the card faulty and unusable. The same principle can be applied on a mobile phone with the intention to damage the NFC hardware on the mobile device.

Whilst NFC Jamming and NFC Zapping can be considered as a form of data modification attack, the actual modification effects the whole transmission thus resulting in blocking the whole transmission (i.e. denial of service).

#### **3.1.1.4.3 NFC Data Modification**

It is more difficult for an attacker to be able to modify or change the data during a transaction. NFC uses two different encoding schemes: Miller encoding with 100% Amplitude Shift Keying (ASK) and Manchester coding. Haselsteiner and Breitfuß [76] claim that data modification on Miller encoding is only possible on certain bit combinations while using Manchester coding data modification is possible on all bits. Since this project focuses on HCE, where both the reader and the HCE device would be using active mode, the reader and the mobile device will be using Miller encoding with 100% ASK hence only certain bit sequences combination can be modified.

Data insertion in NFC is only possible if the device (e.g. smart card) responding to a message from the reader (POS) provides a delay, enough, for some other rogue device to reply 'instead' of the legitimate

device. If both devices transmit together the data will overlap and hence it will be corrupted. Hence such an attack can only be feasible if the smart card or device being used has a long delay before it responds to the message sent by the reader. Given that most NFC payment transaction are normally a tap event (very short event) such delays do not normally occur [76].

As at October 2016, no research could be found indicating a practical implementation of data modification and or insertion at the bit level during NFC communication.

### **3.1.2 Attacks at the Application Layer**

After looking at various ways of how data can be attacked and extracted at the physical layer due to weaknesses during the communication process, the application layer will be the next layer that will be analysed for possible attacks. Attacks in the application layer relate to weaknesses in the standards or specifications used in the contactless payments infrastructure or weaknesses in the implementation.

#### **3.1.2.1 Cloning**

While cloning a contactless card is classified as a weakness at the physical layer. The success to mounting the cloning attack is because the information provided by the card is 'static' and thus it could be copied and 'replayed'. The consequences could be avoided if the application layer provides some form of protection against this attack. Despite that countermeasures have been developed to limit the inherent risks, loopholes remained and thus attacks could be mounted.

Roland and Langer research [77] focused on cloning a contactless card and exploiting the design flaw in the CVC3s range due to problems with the unpredictable number. The authors noted that the unpredictable number was made up of a 4-byte field, theoretically providing a range of  $2^{32}$  possibilities. But, due to the way it was implemented, using BCD and the Mag-Stripe Protocol, the practical range was reduced to 1000 possible options thus not benefiting from the full range of a 4 byte field. Using BCD the range was reduced from  $0-2^{32}$  down to a number between  $0-99999999$  (2<sup>27</sup>). Furthermore, due to requirements in Mag strip protocol not all the numbers in this range could be used and in essence only 1000 numbers in that range were valid possibilities. Therefore, the authors developed an app on an Android phone that eavesdrops on a contactless card and forces it to generate the CVC3 for all the 1000 possible numbers in a minute. An ideal scenario for such task to take place is in a crowded area such as in a bus. Once the CVC3 are generated, a clone card is created by entering the details into the new card collected from the mobile app along with the table of CVC3 responses. The card is then used to make purchases and when the terminal asks for an ARQC, the card would simply lookup the CVC3 stored in the table and provide the response based on the unpredictable number provided. The only limiting factor is the ATC which would make the whole table obsolete after one transaction.

Emms and Moorsel [78] proved how a contactless card could be used to make purchases online. While this is not a direct 'Application Layer' defect, it is still considered as a weakness due to the way of how merchants handle card not present transactions. The authors' methodology was to install a hidden NFC reader near a legitimate POS reader. While a customer is paying for goods using a chip-and-pin terminal, the NFC reader would read the contents of the card (i.e. cardholder PAN, name, issue date and expiry) and send them to the attacker. The CVC was read using a camera also placed strategically near the POS. The attacker would then make purchases online where 3D secure is not being utilized. The authors claimed, at the time of writing, that there were a number of online retailers which were still not using 3D secure technology.

While the attack above can be mounted in a card-not-present transaction, when the card is present, the EMV protocol has in place several mechanisms at the application layer to protect against a skimming and replay attack. The EMV standard protects against a replay using a transaction counter (ATC<sup>3</sup>), stored on

---

<sup>3</sup> A 16 bit number stored by the card and incremented on each transaction



the card and against a pre-play attack using an 'unpredictable number' provided by the terminal or ATM. The card when asked to generate the ARQC uses these numbers, in a signature and hence cloning a card would require knowledge of these numbers. If the unpredictable number is truly unpredictable then an attacker can never clone a card.

As Bond, Choudary, et al, presented in their study [79] a weakness resides in the specifications of EMVCo [80] which states that in 4 consecutive transactions the terminal/ATM should provide 4 unique numbers. This is a very weak test and ATM developers favoured a simple counter rather than a truly random generator to generate the unpredictable number. The authors outlined the analysis a transaction log file containing the unpredictable number provided by the customer's Bank from an ATM, which clearly showed that a simple time bound counter was being used. The authors then described an attack scenario where an attacker would predict the unpredictable number and then use a modified POS terminal (in a mafia shop). When a customer pays for an item at the shop with his/her card, the POS would provide the 'predicted' unpredictable number and a large value to the card. The card will then generate the ARQC which is stored in the POS. The attacker then collects the ARQC and the pin, also entered by the customer on the rogue POS, and creates a clone of the card, visits an ATM (which will use the predicted number) and uses the cloned card to withdraw money. . The downside to this attack is the fact that the transaction amount is entered as part of the ARQC and the cloned card is only used once. Notwithstanding, the POS at the mafia shop would make the card sign whatever amount it wants.

One key 'risk limiting' feature within EMV is configuring transactions limits which do not require a PIN or a signature to authorise payment. In the UK, a cap was set at £30 at the time of writing to prevent fraudulent activity should a card be stolen. But Emms, Arief, et al, in their research [81] found another weakness, in the application layer, providing them with an opportunity to mount an attack by bypassing the transaction limit and steal considerable amount of money. The authors found that if the Visa contactless cards are presented with a transaction involving a foreign currency other than the base currency the cards would sign any amount up to 999,999.99 without requesting a pin or making the terminal go online. Secondly, the authors also noted that there is no requirement in EMV protocol for POS terminal to authenticate itself to the card and thus any rogue POS can be used with the card. In their attack scenario, an attacker would use an Android mobile phone with a special app. The app would allow the attacker to enter a specific amount and currency which will be sent to the victim's card to generate the signature. When it comes in contact with any contactless card through NFC, the App would ask the card to sign a transaction in foreign currency. The details are stored in the mobile phone and later transferred to a rogue merchant. The merchant would simply pass these transactions through a rogue POS that would be programmed to pass these transactions to a bank. The merchant would then collect the money in his/her account. If the card could in some way authenticate the POS then the card would have never provided the signature on the transaction when in contact with a rogue POS, an Android mobile phone in this case.

### **3.1.2.2 Point of Sale (POS)**

POS hardware has over the years evolved from simple POS Terminal systems having a stripe and keypad up to PCs commonly used in large retail stores. Today POS systems have much more functionality rather than payment application/function only. These include systems with ERPs, cash registers and barcode/RFID scanner. In such systems, the POS would be running an operating system such as Windows or Linux.



**Figure 5 - Simple Terminal with 'closed hardware'**



**Figure 6 - POS Terminal/PC running Windows OS**

Different types of attacks on the POS are possible. We can classify these in two categories:

#### **3.1.2.2.1 Physical tampering**

In this kind of attack, the POS's hardware is modified in a way where the attacker would be able to gain information or control from the POS. For example, a wireless chip might be added to transmit details to a wireless receiver. Another example could be a special chip or module installed in the POS that would bypass certain functionality in the POS such as making the POS authorize a transaction even if the actual authorization fails. To countermeasure these attacks, POS hardware is designed to be tamperproof [82] with special seals used to determine if the POS is tampered. Nonetheless, technicians might gain access during maintenance or servicing and attacker might even find ways of bypassing the tamper proof hardware.

#### **3.1.2.2.2 Software/Firmware tampering**

Software and firmware attacks modify or install new software and firmware on the POS. Obviously physical or remote access to the POS is required to mount such an attack. Such an attack can happen throughout the whole supply chain and hence POS hardware should be closely guarded even during supply and installation. In 2008 [83] several chip and pin readers installed around Europe were said to have been tampered in such a way that the POS reads and forwards card data to attackers that eventually created clones of the cards and made payments through these cards. The report states the POS might have been tampered during manufacturing or supply. Software and firmware attacks can also be mounted through remote access of the POS, especially POS hardware making use of operating systems such as Linux and Windows. The attacks vary from installation of malware, RAM scrapping, etc. The end aim of the attacker would be to either steal card data from the POS or modify the operation of the POS for the attacker's benefit.

In November 2009, Visa issued an alert entitled "Targeted Hospitality Sector Vulnerabilities" [84] and in 2013 "Preventing Memory-Parsing Malware Attacks on Grocery Merchants" [85]. In both of these cases Visa acknowledged an increase in attacks on POS using specialized software. The attackers either used debugging tools capable of accessing volatile memory or memory parsing malware [86].

#### **3.1.2.3 Payment Infrastructure**

Up to now, the attacks considered have been based on communication between the POS and the contactless device and/or the POS itself but the payment infrastructure also consists of stakeholders and payment data that flows through all the stakeholders that are subject for attacks.

Albeit, the security between the stakeholders has increased, the research on attacks on these stakeholders is still relevant to ensure a secure infrastructure. In 2005, CardSystems solutions, a credit card processing company was breached and over 40 million card data was stolen [87]. The company said the attackers got access to a file which was holding transaction data that was not secured properly. The company also issued

a statement that it was not compliant to the guidelines as required by card schemes for storing transaction data.

In 2008, Heartland Payment System suffered a data breach [88]. The investigations led to a weakness in one of the web-based services through which the attackers mounted an SQL injection attack and managed to get access to sensitive card data which was 'moving' through their network.

Unprotected data, whether in transit, information exchanged during communication or at rest such as data stored in plaintext in a database, can be a target for the attackers. While these type of breaches are not 'directly related' to HCE, the fact that third party providers and/or Issuers can act as a Token Service Providers, their role is added to the payment infrastructure model and thus increase the chance of being attacked.

### **3.1.3 Attacks on Cryptography and Key Management**

During a payment transaction, the payment card or device communicates with the POS to send payment account information data related to the transaction. During this exchange, several cryptography mechanisms are used to authenticate the card in both online or offline transactions. Below is an analysis of the authentication mechanisms used during an EMV contactless transaction.

#### **3.1.3.1 Offline Authentication**

##### **3.1.3.1.1 Static Data Authentication**

As its name implies SDA is a 'static' form of authentication whereby the aim is to provide data-origin authentication. This means that the static data on the card was actually created by the Issuer. This prevents an attacker from creating a counterfeit card under an Issuer's name or modifying the data. However, this does not limit an attacker from copying the card contents and cloning it on another card. Section 3.1.1 identifies a number of ways of how such attacks could be carried out.

SDA is based on public cryptography and in its simplest form is an RSA signature [89], using the private key of the Issuer, on the card's data (e.g. PAN, etc.). The actual cryptogram is personalized and written to the card upon production. This means that with every transaction, the same signature will be applied. However the cryptography of SDA itself relies on RSA signatures. RSA is considered secure as long as the correct key size is observed [90]. EMVCo, does not provide any guidance or mandatory requirements as to the size of the key but the guidelines state that the Issuer shall ensure that the keys used "should be adequate for the planned lifetime" [91]. This paper claimed that in 2014, some CA public keys used on SDA cards were still using 1024 key size even when many experts and companies in the security field were recommending a minimum size of 2048 bytes [92].

Coron, Naccache and Stern [93], showed how it was possible to create a forged existential signature over ISO 9796-1 and 2, which are used in EMV. Following this study, ISO 9796-1 was withdrawn and ISO 9796-2 was modified to ensure this attack could not be used against the standard. Further on in 2009, the same authors published another study this time refining the algorithms used in their previous research to prove an attack on the 'modified' ISO 9796-2 using an Amazon rented server. The authors proved that for a cost of \$800 (in 2009) they were able to create a signature on a message for a single modulus (key). While this applied to ISO 9796-2, EMV provides further restrictions and thus the authors concluded that an attack on EMV to create an existential signature would be unfeasible at a cost of \$45,000. Both Visa and MasterCard have mandated for SDA removal, starting October 2015 [94].

### **3.1.3.1.2 Dynamic Data Authentication**

DDA is an improved version of SDA, where apart from signing the static data on the card, the card has to prove knowledge on the private key that is matching the public key of the certificate. To do this a message is sent to the card, known as DDOL (Data Authentication Data Object List) containing at least a terminal generated nonce and some dynamic data provided by the card (e.g. a nonce). The terminal generated nonce ensures that the card is actually generating the signature, rather than replaying one that was pre-generated. DDA uses RSA signatures and hence the same factors mentioned above for SDA apply.

### **3.1.3.1.3 Combined Data Authentication**

The main difference between DDA and CDA is the fact that in CDA, no additional messages (i.e. extra terminal to device requests) are required as the actual authentication is part of the transaction itself. In CDA the card signs dynamic application data made up of a card-generated nonce, CID, cryptogram, Transaction Data Hash Code (TDHC) and a nonce generated by the terminal. All this is supplied to the card during the transaction (terminal generated nonce is supplied within the GENERATE AC command and hence there are no additional messages/steps to be done for authentication. The basics of the cryptography are the same as that used in DDA and SDA (RSA). The same factors as those mentioned for SDA apply for CDA.

Within EMV, the same key pair used for generating the signature and for encrypting the PIN for online cardholder verification. Degabriele et al, [95] presented a study showing how, using a partial decryption oracle, a forged signature can be generated on a chosen message. The attack is based on the knowledge about the 'structure' for encrypting the PIN for the card to verify. Specifically, the pin is required to have a padding of 0x7F and based on the algorithm presented by Bleichenbacher [96] and the algorithm provided in [95] by Degabriele et al, and given access to a cryptographic oracle that can provide if a certain ciphertext is valid or not, it is possible to forge a signature. The attack uses a wedge (i.e. a device) between a card and the terminal to mount the attack, result would be to manipulate the messages flowing between them. The whole attack is possible only because the same key pair is used for signature in CDA and to encrypt the PIN for verification. The authors provide a theoretical attack on forging the signature in an offline CDA authentication. The attack mentioned in the research was described for a chip and pin system but the same attack would work with a contactless transactions especially in a relay situation.

### **3.1.3.2 Online Authorization**

Both the card and the terminal can decide to request a transaction to be authorized online. The factors that are taken into consideration vary between different schemes and Issuers. One example could be a transaction amount limit whereby transaction above that limit would have to be done using online authorization. Transactions are required to go online when the ARQC is requested. The ARQC is an encrypted hash of specific data 'tags'. Both the type of hash or mac used and the encryption are selected by the Issuer. Some Issuer can request offline authentication cryptograms and ARQC, for example MasterCard PayPass can request CDA and ARQC [97]. The ARQC generated by the card is sent to the Issuer for authentication. Typically the card would have a Card Key which is based on a master key, held by the Issuer and its PAN. The Issuer hence can generate the ARQC and verify it matches with the received ARQC from the card. The contents of the ARQC are specified using the CDOL1 tag list and can vary between card schemes.

While access to the secret key in the card is difficult, it is very important that the protocol used contains 'features' that limit relay and pre-play attacks. As shown in Section 3.1.2.1 when the unpredictable number was actually predictable, attackers were able to demonstrate how the card itself, using NFC, could be used to generate the ARQCs without the need for the attacker to know the secret key [79].

## 3.2 Literature review on attacks of mobile payments using Host Card Emulation (HCE)

As usage of Host Card Emulation is increasing, such technology can become a prime target for attackers by exploiting the threats and vulnerabilities in the design and application of the technology such as the mobile app and OS and tokenisation. Absence of proper security measures and controls can lead to attackers stealing payment credentials to make fraudulent transactions.

The analysis in this section has been split into attacks and vulnerabilities related to the OS/Kernel of the mobile device, secure memory areas, cardholder verification and tokenization.

### 3.2.1 Attacks at the Operating System/Kernel Mobile Device

Host card emulation emulates what a chip in a contactless card would do in software. While the mobile device, might have hardware vulnerabilities (e.g. tampering), most of the related work and literature, on weaknesses and vulnerabilities, available is focused on the software running in the device. Since mobile phones are connected to the internet, weaknesses in software can be exploited by attackers to steal information, create backdoors, etc. At the time of writing the three major mobile operating systems in the market are Android, Windows Phone and iOS. Both Android and Windows Phone have implemented HCE while iOS (Apple) use an SE model. In essence all the mobile operating systems are vulnerable to attacks but the fact that Android has an open source model has allowed researchers to study and scrutinize the security of the OS easier and in much more detail.

For the purpose of this project, the analysis in this section will be focused on Android. With time, it is being noted that the vulnerabilities are increasing and growing exponentially as per reference observed in the Common Vulnerabilities and Exposures database (CVE). Figure 7 represents this detail.

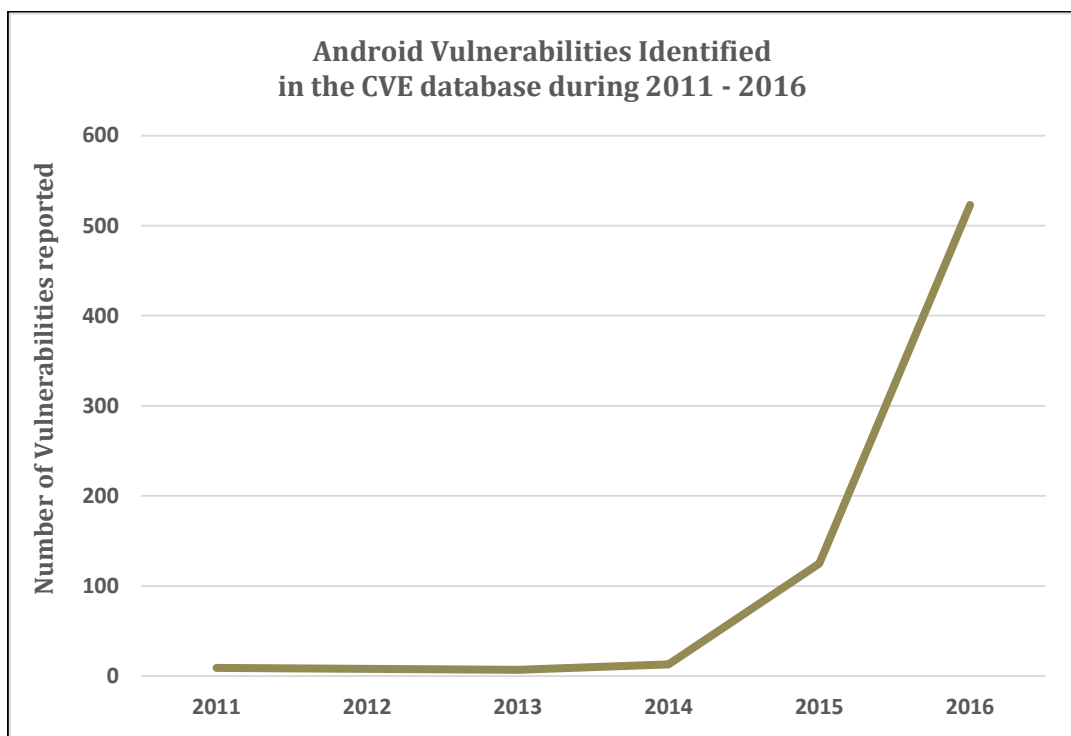


Figure 7 - Vulnerability Statistics reported in CVE database [98]

When researchers identify a vulnerability, it is reported to Google, acknowledged and eventually fixed through patches and updates to the system or the app where the vulnerability is found. The different type of vulnerabilities currently identified in the Android OS is being summarised in Figure 8. It is to be noted that the respective vulnerability may be classified under more than one type of vulnerability.

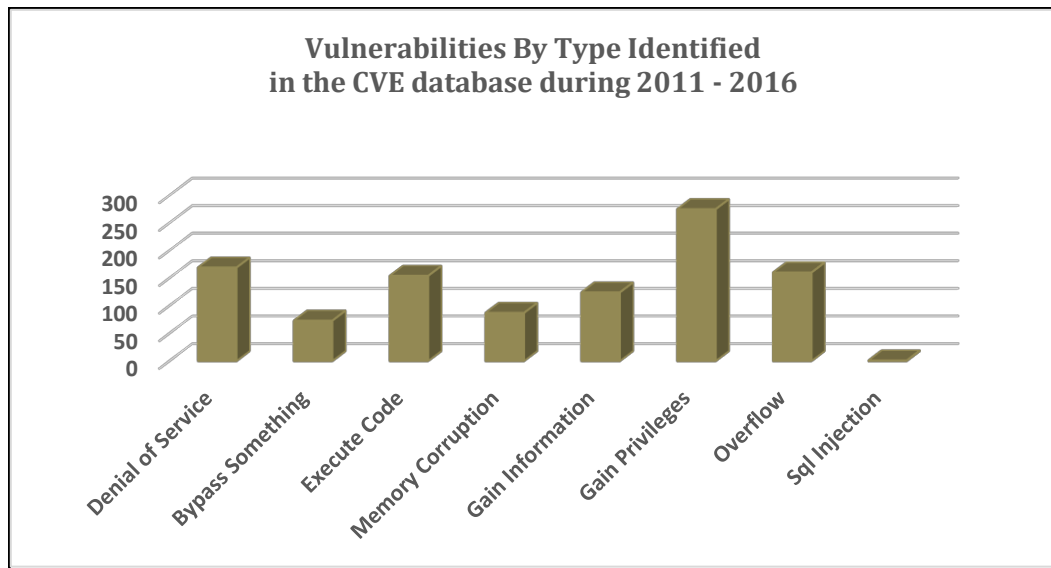


Figure 8 - Vulnerability Statistics reported by Type in CVE database [98]

The Android OS has been built to provide different level of permissions to applications thereby limiting what an application can do and access. Typically applications are run independently in a sandbox. Thus under normal circumstances an attacker would have a limited set of actions. But if 'Root' access privileges are gained, the attacker would have full access to do anything and access everything in the OS including the device's applications, data and resources. Following such access, attacks can be mounted.

Hei, Du and Lin [99] show a typical example of such an event. The authors show a vulnerability in a sub-section of the kernel leading to a buffer overflow and subsequently gaining root privilege by running specific code. Figure 9 identifies known vulnerabilities that managed to obtain root access level within the Kernel.

Nickname	CVE or ID	Release (platform)	Cause of Vulnerability	Vulnerable component		
				Kernel	Driver	Daemon
asroot	2009-2692	08/2009 ( $\leq 2.2$ )	Null pointer dereference	socket	-	-
exploid	2009-1185	07/2010 ( $\leq 2.1$ )	Incorrect input validation	-	-	udev
RAtC	2010-EASY	10/2010 ( $\leq 2.2$ )	Incorrect error handling	-	-	abdb
Zimperlich	2010-EASY	12/2010 ( $\leq 2.2$ )	Incorrect error handling	-	-	zygote
KITNO	2011-1149	01/2011 ( $\leq 2.2$ )	Incorrect sharing of resources	-	-	init
psneuter	2011-1149	01/2011 ( $\leq 2.2$ )	Incorrect sharing of resources	-	-	init
GingerBreak	2011-1823	04/2011 (2.1 - 2.3.3)	Incorrect input validation	-	-	vold
Zergcrush	2011-3874	10/2011 (2.2-2.3.6)	Buffer overflow	-	-	vold
levitator	2011-1350,1352	11/2011 (2.3-2.3.5)	Improper bound check	-	PowerVR	-
mempodroid	2012-0056	01/2012 (4.0-4.0.4)	Improper permission check	mem_write	-	-
bin4ry	OSVDB 94059	09/2012 (4.0-4.0.4)	Symlink attack	-	-	abdb
diaggetroot	2012-4220,4221	11/2012 (2.3-4.2)	Integer overflow	-	diagchar	-
-	2013-2094	06/2013 (2.2-4.3)	Integer overflow	perf	-	-

Nickname	CVE or ID	Release (platform)	Cause of Vulnerability	Vulnerable component		
				Kernel	Driver	Daemon
FramaRoot	2013-6282	04/2014 (2.x-4.x)	Missing checks	get/ put_user	-	-
TowelRoot	2014-3153	06/2014 (4.0-4.4)	Use-after-free	futex	-	-
GiefRoot	2014-4321,4322	12/2014 (4.0-4.4)	Missing checks	-	camera	-
PingPongRoot	2015-3636	08/2015 (≥4.3)	Use-after-free	net	-	-

**Figure 9 – Past Vulnerabilities in the Android Platform [100]**

As shown in Figure 9, the attacks were concentrated against the Kernel, Drivers and Daemons. The reason is that these three types of processes are run with root privilege within the OS, therefore exploiting one of them would mean the attacker would be able to run code at root level. Once these attacks have been acknowledged, patches will be released to fix these issues.

Once an attacker has gained root privileges, there are many ways in which an HCE payment process could be affected. Some examples include:

- An attacker could corrupt the AID table, thereby routing a transaction to another service.
- An attacker can corrupt the ARP table and set-up a Man-In-The-Middle attack forcing the mobile to pass through a ‘false’ router or gateway, change values in memory or even change the execution position (Program Counter) of another app.

Zimperium, a vendor of enterprise mobile security, described a vulnerability which was eventually fixed where an attacker could install and run a malicious app without the user interaction [101]. The only prerequisite for the attacker is the mobile number of the victim. The attack is based on a framework known as StageFright within Android. The framework is used for displaying media content such as videos. The framework contained an issue with a length field used in ‘.mp4’ movies which the framework treated as 32 bit. Zimperium explained that when sending a 64bit size field it will overload the buffer and create a buffer overflow attack and subsequently making the processor execute code within the media content itself. The attacker is only required to send an MMS to the victim with the movie as part of the message. The StageFright is called upon automatically when the victim’s device receives the message to be able to display a preview of the movie on the notifications screen. This means that without the victim taking action, the attacker manages to gain access to the device. Privilege escalation can be used by attackers to mount or aid in other attacks such as stealing information.

### 3.2.1.1 Denial of Service

This type of attack is aimed at denying the mobile phone from providing certain services. For example an attacker can deny the use of NFC in a phone thereby disabling the possibility of an HCE payment. A typical vulnerability of this type is identified in CVE-2016-7990 [102]. This vulnerability exploits a special protocol named OMA Client Provisioning (i.e. OMACP). OMACP is used to provision settings to a mobile phone via a WAP message sent directly to the phone. The user would visit a site, request the provision and a WAP message is sent as a message to the device. Upon clicking and accepting the message certain settings on the phone are automatically changed. In CVE-2016-7990 a vulnerability was identified in Samsung Galaxy S4 and S7 devices where an integer overflow occurs when parsing an OMACP message which leads to a heap corruption thereby creating a Denial of Service of certain services on the phone.

### 3.2.1.2 Execute Code

In this type of attack, the aim of the attacker would be to execute certain malicious code for various intentions. The software could be code that would already be existing on the mobile device or otherwise injected to the phone during the attack. Vulnerability CVE-2016-7990, mentioned above in the DoS attack can be used to mount such an attack. In this vulnerability the heap, which is the memory allocated to an

App including the code it will execute, is corrupted. If the heap is corrupted then an attacker can potentially change the execution code of an App thereby making it execute malicious code.

### **3.2.1.3 Memory Corruption**

Some of the vulnerability types can be used to mount a memory corruption attack. The OS should ensure that when RAM is assigned to an application other applications should not be able to access it and more importantly should not be able to corrupt it. If memory is corrupted it could lead to other types of vulnerabilities such as DoS (the app could stop working if the memory content is corrupted), Code execution, etc. A typical example is described in CVE-2016-0705 [102]. In this vulnerability a special condition known as a 'double free' occurs. The vulnerability was found in code used in OpenSSL 1.0.1 and 1.0.2. A double free condition occurs, in C, when the function "free()" is called twice on the same pointer. In the first call the condition works as intended but in the second call the pointer, passed as an argument, would have been set to null. According to the C11 standard [103], this leads to 'undefined behaviour'. In CVE-2016-0705 this condition leads to a memory corruption and eventually to a DoS of certain services due to the memory corruption that occurs.

### **3.2.1.4 Bypass procedures**

This type of attack is used to bypass certain procedures in the mobile device. For example, it could involve an attacker bypassing certain access procedures to gain access to information that would otherwise be unavailable to the attacker.

An example of a vulnerability of this type is identified in the CVE database as CVE-2015-8890 [102]. This vulnerability was found in the driver that handles partitioning in the Android system. Researchers found that the system was not validating the GUID Partition Table. This allows an attacker to create an SD or MMC card with a carefully crafted Partition Table to bypass access limitations i.e. gaining access and even the possibility to write data in areas where the OS does not provide access. Note that no root access is required to mount the attack. All that is required from the attacker is physical access to the phone to install the malicious SD or MMC card.

### **3.2.1.5 Gain Information**

This type of attack focuses on exposing certain information. For example, in an HCE wallet application, it could expose the value of a Key if the key is used in the application's memory (i.e. RAM). A typical example of this case has been identified in CVE-2016-2419 [102]. This vulnerability was found in the Android's media server. The vulnerability is based on an uninitialized data structure which can expose certain data until it is actually initialized by the media server software. When a software is allocated a piece of RAM for a variable or data structure, the area would still contain the previous value and it is up to the software to initialize (clear) the structure. If initialization is not done then the software can potentially expose the RAM contents that would have been left by the previous software occupying that part of the RAM. Hence through this weakness an attacker could expose the contents of the RAM of another piece of software.

### **3.2.1.6 Trusted Execution Environment (TEE)**

With the many different types of vulnerabilities that exist it would be impractical to store highly sensitive data and cryptographic keys through the 'standard' ways provided by the OS. For this reason, a more 'isolated hardware' solution has been proposed by different manufacturers. The solutions vary but in essence, the aim is to have a hardware isolated section (secure world) in the processor which is separate from the regular OS software such as the kernel and other applications (normal world).



### 3.2.1.6.1 GlobalPlatform's TEE Specifications

The 'secure world' environment is normally called the TEE (Trusted Execution Environment) which is a secure area in the main processor in which data can be stored and processed securely. Different hardware manufacturers have come up with different solutions making it difficult for software vendors to develop and support the different types of configurations of TEE available hence industry associations such as GlobalPlatform [104] have come up with specification with the aim of enhancing interoperability of software between different hardware manufacturers. GlobalPlatform has been responsible for standardization of the TEE Features on behalf of the industry. The first specifications were published in July 2010 [105]. The following is a list of some of the most important specification pertaining to HCE:

- TEE Client API Specification v1.0 – Communication between trusted applications in the TEE and regular application in the OS.
- TEE Internal Core API Specification v1.1.1 – Secure data storage, operating cryptographic functions, etc.
- TEE Sockets API Specification v1.0 – provides standards to enable trusted applications to directly make use of internet communication, bypassing the need to send data through applications at the OS level.
- Trusted User Interface API Specification v1.0 – provides standards on how trusted applications can access system hardware resources such as displaying text and graphics and allowing users to perform actions such as entering PINs etc.

### 3.2.1.6.2 ARM Trustzone

ARM [106] is by far the most popular mobile processor architecture and is used in over 95% [107] of mobile phones on the market. In the ARM architecture the TEE is implemented under the name TrustZone. The technology was added to ARMv6 processors upwards. The concept behind TrustZone, as shown in Figure 10, is that an arm processor is able to run two operating systems, Secure OS and Normal OS [106] at the same time using a single core of operation. With TrustZone normal applications operate in a 'normal' mode, the kernel operates at 'system' mode and the trusted application operates in 'monitor' mode. Hence even if an attacker manages to obtain access to a rooted application, the attacker would still be unable to access the protected parts of the trusted applications. The way trustzone is implemented allows developers to have access to hardware, e.g. accessing PCI-E address space, therefore trusted applications can be provided the ability to access certain hardware directly.

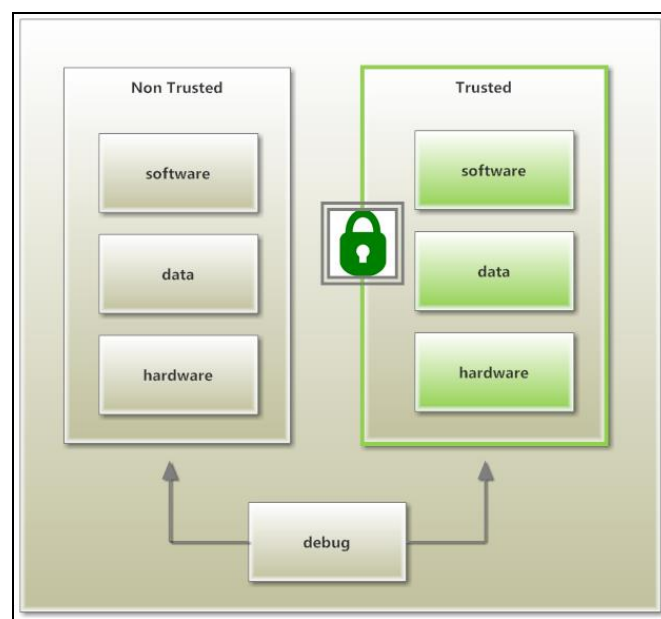


Figure 10 – ARM Trust Zone Technology [106]

### 3.2.1.6.3 Intel SGX

Intel's solution to create a TEE in its processor is based on Intel's Software Guard Extensions as shown in Figure 11. IntelSGX is made up of a set of extensions to the Intel architecture that aim to provide two key assurances to software running under IntelSGX: Integrity and Confidentiality. These assurances are provided even when normal software, including the privileged OS kernel, are compromised.

Confidentiality in IntelSGX is provided by isolating the code and data of a trusted application from the outside environment (normal software including the operating system and the hardware devices). To achieve this, a trusted application is first loaded in an 'enclave' and from then on all the code and data in the enclave become inaccessible to the regular environment (OS and regular applications) [108].

The second assurance property is data Integrity. Integrity in IntelSGX is provided through software attestation. Upon development of an application, the developer creates a cryptographic hash of the enclave when the application is loaded in the enclave. This hash is provided to the attestation service provider in the form of a certificate. When a user loads a trusted application in the user's computer (i.e. inside the processor's enclave), IntelSGX creates a hash of the enclave and verifies the hash with that stored in the attestation service. If this value matches then the attestation service can 'attest' to the software's integrity thereby providing assurance of the integrity of the software.

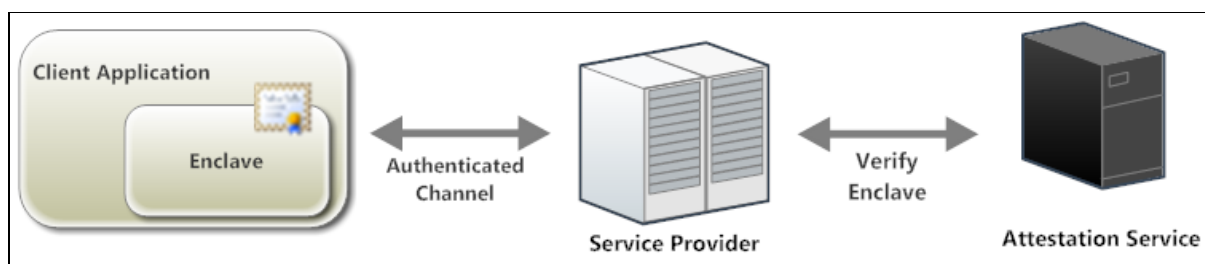


Figure 11 - Intel SGX Remote Attestation feature [109]

### 3.2.1.6.4 Implementation of the TEE in Android

In Android, the TEE is known as Trusty [110]. This is an open source set of software components that make up the TEE. The key components that make up the TEE are the Trusty OS, drivers that handle communication between the Android kernel and the trustlets, and a set of libraries (API) to facilitate communication between the non-secure world and the trustlets running in the Trusty OS.

There are two methods of how the TEE could be used as a secure memory storage for HCE wallet apps.

- The first method uses the TEE to store a 'Master' key and a trustlet to operate that master key. The actual data (e.g. credentials or tokens) would still be stored in the non-secure OS but they are encrypted in the TEE with the master key also stored in the TEE. This is how the hardware-backed Android keystore works.
- The other method uses the TEE to store its own keys. This means that a trustlet would have to be developed to handle the key storage and communication to and from the wallet app in the non-secure OS. Samsung Pay uses this model to store payment tokens and cryptographic keys in the TEE in the mobile device [111].

### 3.2.2 Attacks on Secure Memory Areas

One key requirement for every HCE app is a secure memory area to store and process payment applications and account information. This could either be volatile or non-volatile memory. Whenever data is stored on the mobile device, it is subject to attacks with an aim of stealing or modifying the data. In Android, an app, like an HCE Service, runs in a Sandbox [112]. This is an 'isolated' environment from any other App and only processes with root privileges would have access to this environment. Non-volatile memory for an app, keys and other security assets, by default, also has this isolated property. Files stored on internal storage of the mobile device are only accessible to the app. Access controls can be set to the files to be readable by other apps.

While it is safe to assume that an attacker would be able to access any data when gaining root privileges, other vulnerabilities have been found, where the attacker does not need root access. MWR Labs found a vulnerability in the Google Admin app, where a malicious app residing on the same device read data from within the sandbox of the Google Admin app [113]. The Android OS provides for inter-app (i.e. interacting with other apps) communication through a system of 'intents'. MWR Labs found that, by passing an intent for the ResetPinActivity within Google admin and passing a file url that is writable from within the malicious app, such file is loaded in a webview (running in Google Admin). Eventually, using HTML, the malicious app manages to read files that are within the Google Admin sandbox. This is not a 'sandbox' flaw but a flaw in Google admin that exposed its own sandbox.

Android provides a secure mechanism to encrypt the data being stored and a 'keystore' [114] to store keys for an app securely. The mechanism is called a keystore. The mechanism is implemented on Android using a KeyMaster service [115] which is split between a service running in the Android OS non-secure world and a trustlet running in the TEE as illustrated in Figure 12. Any application that needs to use the keystore will first use the Inter Process Communication to communicate with KeyMaster service (i.e. running in the secure world) which in turn communicates with the trustlet.

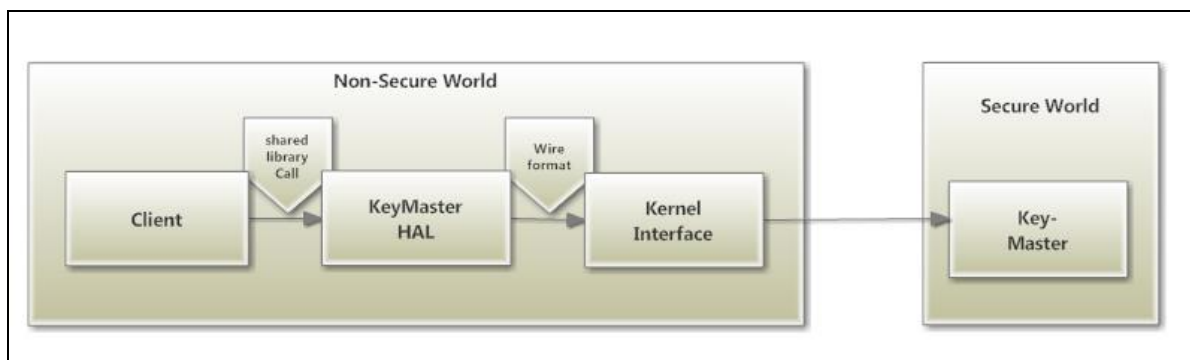


Figure 12 – Access to Keymaster [115]

The actual key is not stored in the TEE but it is stored in the app's sandboxed storage. The key is encrypted using a device specific key which is stored in the TEE. Since the keys are bound to the device a malicious app with root access, can use these keys to sign or encrypt the data [116].

For example consider a payment application that requires a signature on a transaction. A malicious app with root access copies the 'honest' payment application's key file to its own sandbox. Subsequently, the attacker signs a fraudulent transaction with that key and it would still be valid as the key is device binding. The app, however cannot find out the value of the key.

Not all mobile device vendors provide support for TEE based hardware keystore and in such case a software based keystore is used. In such case the keys are also stored in the application's sandbox. The

keys are encrypted with the device lock screen pin if this is activated or no encryption is used if the lock screen pin is deactivated [116]. If a lock screen pin is used it is only 'accessible' when the device has been unlocked, (i.e. when the user has been authenticated) [114]. Just like a hardware supported keystore, the key is device bound since the lock screen pin is device specific. Hay and Dayan [117] found a vulnerability in the keystore that could be potentially used to leak the device lock screen password and hence it could be used to leak the keys of other applications if root access is provided. The vulnerability was related to a buffer overflow in the keystore related to one of the functions 'KeyStore::getKeyForName'. This function calls another function 'encode\_key' and if a large key name is provided the encode\_key function will end up with a buffer overflow since no bound checking was implemented in encode\_key function. Once the buffer overflow occurs the attacker could expose the device's lock credentials. The credentials can then be used to extract the keys of other applications.

### **3.2.3 Attacks on Consumer Device Cardholder Verification (CDCVM) methods**

#### **3.2.3.1 Introduction to CDCVM**

In an EMV payment transaction, the terminal, based on a risk decision mechanism, decides whether a cardholder verification is necessary. This is necessary to ensure the authenticity of the person attempting to make the transaction.

In a contact based transaction this is normally in the form of a PIN or signature on the terminal but with contactless, specifically HCE based payments the terminal has the option to opt for Consumer Device Cardholder Verification Method (CDCVM) [30]. CDCVM options include PIN or biometric such as fingerprint entered on the mobile phone. Similarly to Apple Pay [118], which uses Touch ID or the passcode on the iOS device as the 'verification' of the cardholder identity for purchases Android Pay supported in Android 6.0 Marshmallow [119] launched the option to verify a payment transaction using the fingerprint scanner sensor, instead of a PIN. In the future, other verification methods may be supported like pattern and vein recognition. In fact, the FIDO Alliance is working with EMVCo in developing specifications for secure authentication and authorization protocols for the mobile ecosystem to reduce reliance on passwords and properly address these vulnerabilities [120].

The EMVCo specifications do not mandate what type of 'verification' method should be used. Both Visa and MasterCard support CDCVM in their specifications [118] but mostly leave it up to the Issuer in terms of methods to be used. For example, MasterCard leaves it up to the Issuer to select the method but suggest that the actual PIN of the card should not be entered on the mobile device if the device is not certified to the PCI PTS [121].

The sections below outline different attacks that have been identified in relation to CVM methods.

##### **3.2.3.1.1 PIN/Password Authentication**

PIN authentication is already used as a cardholder verification option in both contactless payments and chip-and-pin/EMV cards. With the addition of CDCVM, Issuers are able to provide on-device PIN or any passcode bound to the user. One consideration that Issuers have to consider about CVM is the fact that the whole contactless experience is about speed and convenience. Considering the fact that mobile device users are already providing a pin to unlock their mobile device and in some cases to open the wallet, another pin would mean that a payment could easily result in 3 PIN entries for a simple purchase. Most probably the value of the pins would also be different. This would be a problem from a usability point-of-view especially given the fact that most contactless schemes are marketed as 'tap and go'. Hence Issuers need to find a balance between usability and security. Furthermore the problems with knowledge-based factors, like PINs and password is that such factors rely on memory. People tend to forget them, use weak passwords or pins (i.e. 1234, 0000) or reuse them for multiple accounts.

The PIN, used for CVM, does not need to be the same as that used on the terminal as long as the Issuer has some way of verifying it. While one might argue that a 4 digit PIN entered on the mobile phone should provide the same security as that offered with a 4 digit pin entered on the terminal, one has to consider other factors when entering the pin on the mobile phone. Most of the attacks presented in literature are 'side channel' attacks using 'sensors' on board the mobile device to try and determine the PIN being entered. Simon and Anderson [122] studied how the front facing camera and the microphone on the device could be used to guess the pin being entered on the mobile device. The authors developed an application on a device to prove that after 5 attempts the software was able to guess the correct pin 50% of the time from a set of 50 pins. Similarly, Spreitzer [123] described a method using the ambient light sensor on the device to determine the PIN. The author describes how specific variations in the light sensor output are generated when a user clicks a different number on the touchscreen of the mobile device. The method guessed the right pin from a random set of 50 pins within the first 10 guesses.

Other methods have been proposed to lift pins or patterns entered on a keyboard. Aviv et al, [124] presented a study on 'Smudge Attacks' where a set of cameras were used to determine the oily residue left by the fingers as they touch or move on the mobile phone's screen. The authors claimed that they were able to identify a pattern in different lighting conditions and camera setups. Out of all the possible setups tried, the authors identified a pattern partially 92% and fully 68%.

### **3.2.3.1.2 Fingerprint Verification**

One of the most common attack used against fingerprint recognition for unlocking the mobile and executing a payment is a spoofing attack. A copy of the fingerprint image is obtained in some way and then used to provide verification. The methods to obtain a fingerprint vary. They can be stolen from polished surfaces such as mobile screens or even from a waving hand photo. It can then be re-created using conductive ink on AgIC paper, using latex milk or white wood glue. As an example Krissler, a German hacker, showed how using commercial software VeriFinger [125] and a set of high resolution pictures he managed to recreate the fingerprints of Germany's Defence Minister [126].

The second type of attack focuses on the fingerprint verification process within the operating system on the mobile device. A malicious app is presented to the user to enter his fingerprint. The app records the fingerprint image and sends it to the attacker. This is a type of phishing attack on the verification process. It could also be used to fake an actual verification on a legitimate app by for example presenting a fake lock-screen to which the user presents his fingerprint.

Another type of attack is related to fingerprint data. The Android framework, as explained in Section 3.2.1.6, provides a TEE where both the verification process and the data could be stored and processed. Thus, even in case of elevated root privileges by an attack, the data of the fingerprint should not be compromised if properly implemented by the device vendor. Using the Android framework, device vendors can 'lock' the fingerprint sensor so that it is only accessed by the TEE but device vendors do not always implement such locking mechanism and hence an attacker with root access is able to load the finger print sensor driver into the kernel and access the sensor directly. The attacker would present a false lock screen and then harvest the output from the fingerprint presented by the user. Such attack can be mounted remotely to harvest finger prints on a large scale. One such case was identified by researchers in the HTC One Max, the fingerprint data (known as template) was saved in `"/data/dbgdraw.bmp"` with a Linux permission set to 0666 which means it is world-readable and hence any process, including unprivileged ones can get access to the template [127]. Similar attack also affected Samsung Galaxy S5.

It is worth mentioning that fingerprints, unlike passwords, cannot be replaceable. In case a password is compromised then the victim can remedy the situation by changing the password but the fingerprint is something that is bound to the person 'for life'. Nowadays fingerprints are used for access control,

passports in addition to verify the cardholders. Any compromise of fingerprint has a ripple effect on other applications where fingerprint is used and thus a new form of authentication is required.

### 3.2.3.1.3 General Attacks on Biometric Verification

Figure 13 represents the different points of architecture of a biometric verification system that are vulnerable to be attacked.

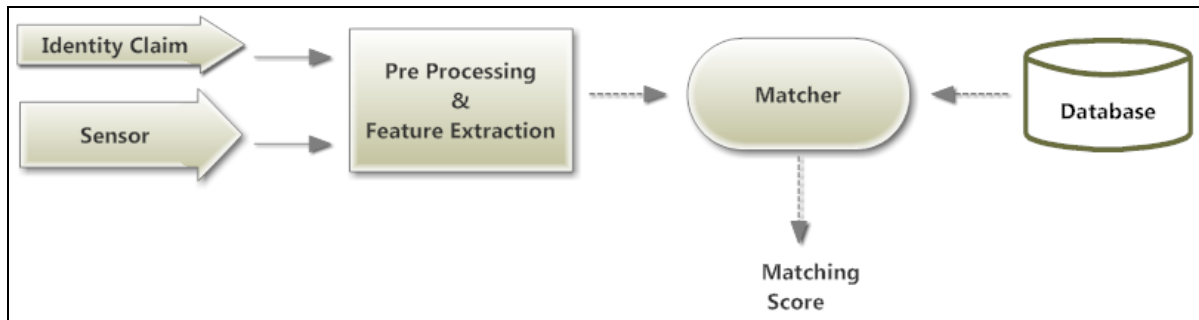


Figure 13 – Generic architecture of a biometric verification system [128]

There are two types of attacks on biometric verification system (i.e. direct and indirect). Direct attacks target the sensor in order to fraudulently access the system and thus such attacks can be carried out without any knowledge about the system. A fingerprint spoofing attack is one such kind of a direct attack. Note that biometric authentication methods rely on parts of our body which can easily be ‘captured and copied’ (spoofed). Unlike a password or pin, there is no secrecy to guard body features.

Indirect attacks focus on the ‘inner’ processes of the authentication method including pre-processing, feature extraction, the database and the matching process. To be able to carry out the attack the attackers need to have some knowledge on the inner workings of the systems and also access to the parts of the system such as the database or matcher. To achieve such an attack the attacker would rely on some malicious software such as a trojan horse or malware. A typical indirect attack would for example intercept the communication between the database and the matcher with the intention of manipulating the data to the attacker’s advantage.

### 3.2.4 Attacks on Tokenization and its Infrastructure

The tokenization infrastructure is meant to deal with issues relating to static payment data. Static data, such as an account number (PAN), once stolen can be re-played or used elsewhere. Tokens, on the other hand are ‘random’ numbers that represent static data, but are meant to be used once and then discarded thus becoming dynamic. The obvious advantage is the fact that if stolen, the token, will expire and can only be used once thus will have little or possibly no value. A TSP provides mapping between a token and its represented static data. In case of payment transactions, the token normally represents a PAN and its expiry, apart from other data. Assuming that tokens are truly randomly generated and are reasonably hard to steal, than tokens would provide protection against replay attacks.

With a random token, a method is required to map the token back to a PAN, a process known as de-tokenization. While this ensures that a token is useless if stolen (attacker is unable to determine the PAN) it creates a highly centralized infrastructure as all ‘trusted’ entities have to rely on the TSP to de-tokenize the token. These kind of tokens are known as irreversible tokens but the PCI guidelines for tokenization [26] allow the use of reversible tokens. Such tokens are cryptographic tokens which are reversible only through the knowledge of a key. This way, trusted entities within the infrastructure can share the ‘de-tokenization’ key and that way the infrastructure is not centralized. This comes at a disadvantage as an attacker might be able to de-tokenize the token if it is not cryptographically secure or if a key is obtained.

Mendoza [129] presented a study on Samsung Pay using Visa Token Service framework infrastructure. In his study, he found a flaw related to the implementation of the actual Samsung Pay App. The attack, described by the author, is mounted on a contactless transaction using Mag Stripe Mode. In this mode the card generates a signature known as dynamic CVV but this signature does not contain any dynamic data provided by the terminal [130]. The signature only contains the card's ATC. This means an attacker can simply obtain a dCVV and re-use it as long as the card is not used elsewhere. The author showed a conceptual attack using social engineering to 'steal' a token (using specialized hardware acting as a POS Terminal collecting the tokens which are then used on another device to make payments.

MalcomVetter [131] provided an overview of side channel attacks that an attacker could run on tokenization which, while not revealing all the data, but revealing important properties that could eventually lead to other attacks. One of the issues with tokenization is the fact that a mobile device would have to communicate via internet, to a TSP. This opens up a window of opportunity for an attacker who has access to the data traffic (e.g. on a WiFi network). The author explains two kinds of observations, one based on timing and the other based on the HTTP header. MalcomVetter found how the tokenisation service response time for a new credit card versus a known card had a different response time when communicating with the TSP side. Similarly, the author also discovered that upon analysing the different HTTP headers, one can determine valuable data.

At the time of writing (i.e. October 2016) no attacks were found against TSPs. Notwithstanding, every TSP is subject to an attack especially attackers would be reluctant in attacking the database where the mapping PAN-to-token are stored. Standard Bodies such as, EMVCo, PCI DSS, X9.119-2 are promoting and issuing requirements and guidelines for building a secure tokenization system and defining controls to prevent potential attacks against tokenization implementations.

### **3.3 Review of existing HCE Implementations**

In this section, two mobile contactless deployment payments models implemented in the industry that use HCE will be discussed. Such section will be as an extension to Section 2.4, where the lifecycle processes of the payment transaction was described. Other variants models may exist but the focus will be limited to the ones described below:

- Device/ OS Provider Wallet Model
- Issuer HCE Model

### 3.3.1 Device/OS Provider HCE Model

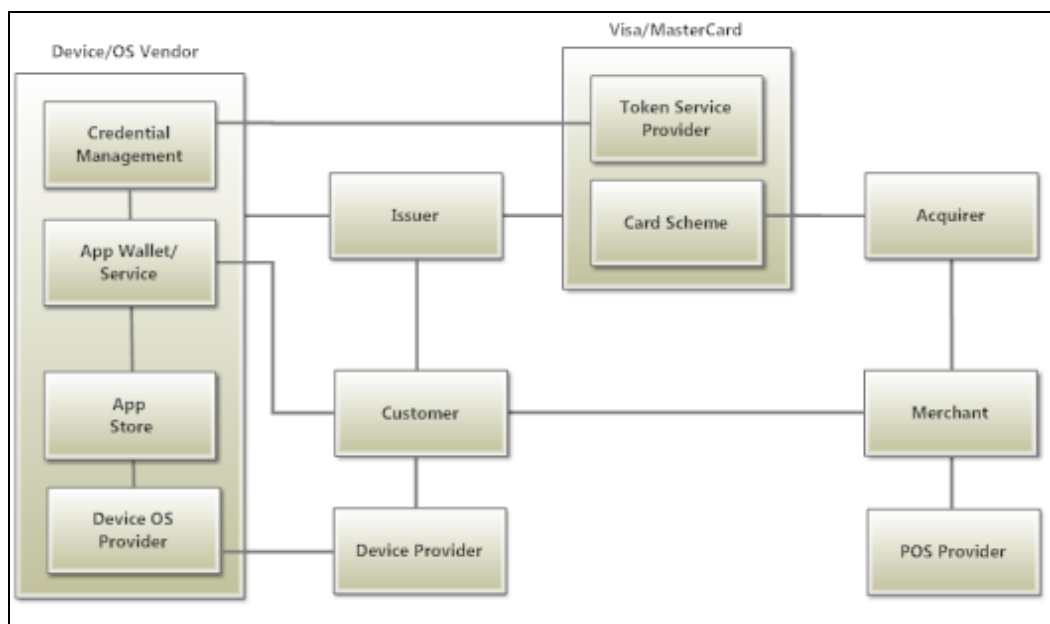


Figure 14 – Device/OS Provider Wallet Model

In this model of implementation, the Device OS provider is responsible for developing and maintaining the wallet app. The app is tightly integrated with the services provided by the Issuers such as the tokenization infrastructure. The concept behind this implementation is that one wallet is shared between many Issuers rather than having an app per Issuer. It also makes it easier for Issuers that wish to provide customers the possibility to use HCE as they do not need to invest in their own app development.

Two implementations fall under this category. These include Android Pay deployed by Google and Microsoft Wallet deployed by Microsoft. Both Android Pay and Microsoft Wallet support both MasterCard and Visa card schemes. The explanation in this section will be only limited to Android Pay due to the maturity and popularity in the market by the banks and retailers of Android Pay and the very limited research and documentation available, at the time of writing on Microsoft Wallet.

The process starts by the OS Vendor, Google in this case, developing the wallet. The wallet is developed according to the recommendations of the cards it will carry. In the case of Android Pay, Google made use of the card scheme's infrastructure for tokenization but there is no limitation on using own (i.e. OS Vendor) implementations as long as they are in line with the recommendations of the card scheme. The next step is for an Issuer to show interest in Android Pay via a card scheme and an agreement is formalised with Google [132]. Google will then proceed in integrating the Issuer in the wallet app, a set of cloud services to administer HCE payment card details including cardholder identification and integration with card schemes TSP for tokenisation. Other processes are also integrated such as Issuer cardholder identification and verification on registration.

Visa uses a system of Limited Use Keys (LUKs) whereby a key is used for a limited period and then changed. The key management system handles when to change the keys based on a set of thresholds. These include time, transaction amount and the number of transactions. MasterCard uses a system of single use keys (SUKs) [133]. A set of SUKs are downloaded on the device and the each key will be used once on the device. Additional keys are downloaded from MasterCard's cloud management system when the other set is used up. The SUKs are combined with a Mobile PIN to make up a session key. This key in turn is used to generate the online cryptogram used during the online authorization process.



The customer downloads the app from Google Play Store, log in using their Google account information, then registers the Issuer's card in Android Pay and accept the terms and conditions for adding a card to a digital wallet. It is to be noted that multiple cards from the same Issuer or different cards from other Issuers can be registered. Typically, a one-time verification passcode, consisting of a unique series of numbers and/or letters will be sent via email or text by the Issuer to confirm cardholder's identity to complete the enrolment. The cardholder authenticates with the Issuer via the card scheme TSP.

The mobile device must support the Android OS running Android 4.4 KitKat or higher. Authenticating the transaction in Android can be performed using fingerprint scanner, PIN code, pattern or a password. Furthermore, Issuers can also issue their own HCE credentials for their apps without the need of involving Android Pay. However, such credentials will not be valid for Android pay – in app.

At this stage the customer is ready to make purchases with the HCE mobile device. A payment token is generated by the card scheme TSP and is passed to Google's cloud server to be provisioned to the mobile device. A limited number of tokens are stored locally to be used when no internet connection is available. When the customer wishes to make a purchase the Android device is unlocked (using the owner's preferred method) and presented (tapped) on the POS. During the purchase, cardholder verification is performed using fingerprint scanner, PIN code, pattern or a password.

The payment credentials are forwarded from the POS to the acquirer, card network and Issuer. The Card network TSP de-tokenises the PAN and forwards it to the Issuer for transaction authorisation. This is then repeated backwards to complete the transaction. Typically a message will appear to confirm the payment has been sent and the terminal will confirm if the transaction has been successful.

Current Issuer's supporting Android Pay in the UK industry include: Bank of Scotland, First Direct, Halifax, HSBC, Lloyds Bank, M&S Bank, MBNA, Nationwide Building Society, NatWest, Santander and Ulster Bank [134].

### 3.3.2 Issuer HCE Model

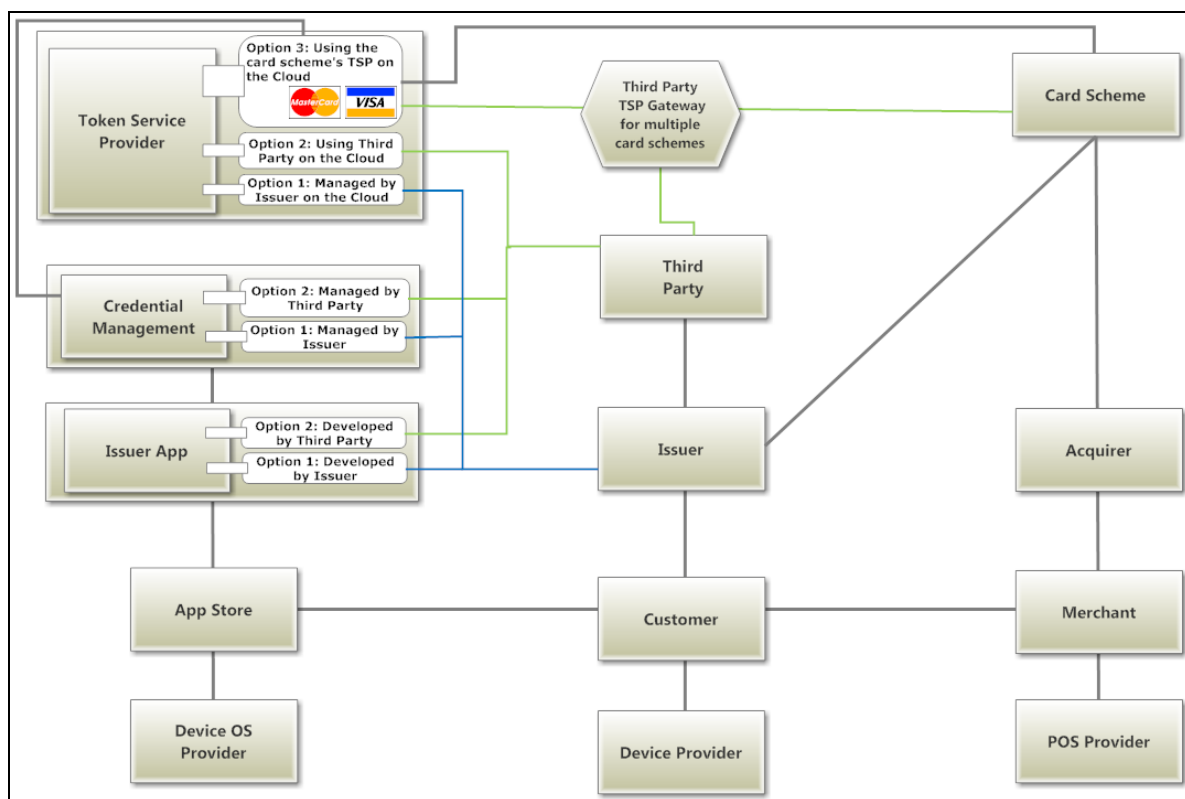


Figure 15 - Issuer HCE Model

In this model, the Issuer is responsible for developing the mobile wallet application. While this means more investment from the Issuer, it also means the Issuer gets to decide which TSP to use, what type of credential management to use and has full control on the wallet App. As shown in Figure 15, the Issuer can be responsible for the whole deployment solution. In this model the Issuer would develop the wallet application and also provide services such as credential management and Tokenization by acting as the TSP.

Issuers may also opt to outsource the above mentioned services to a third party. With respect to the tokenisation, there are 3 options. The role of TSP can either be managed by the Issuer, or by a third party or else using the card scheme tokenisation services platform offered by the respective card schemes such as Visa Token Service (VTS). If multiple card schemes are to be provided, Issuers can opt for the service of a third party gateway, integrating various card schemes into one common interface to reduce complexity for the Issuer.

Parties having the role of a TSP, have to follow adherence to EMV Payment Tokenization Specification Technical Framework, EMVCo TSP registration (i.e. not applicable to Issuers on-premises solutions) and hold a valid PCI-TSP certification [135].

Both the Issuer's and third party approach requires that the card scheme whose brand is used will need to be certified. If more than one card scheme brand is used, then the Issuer needs to individually certify the HCE solution to ensure compatibility and compliance with the card schemes specifications.

An implementation that was fully developed by the Issuer is the Barclaycard app for Android. It was reported in the media that Barclaycard was the first Bank in the UK to launch its own contactless payments app from any NFC on any Android device running version 4.4.2 KitKat or above [136]. Barclaycard's wallet

app is available on Google Play and currently accepts Visa Barclaycards as well as Barclays debit cards. Despite that no public information was traced specifically on Barclaycard tokenization, in an online article, it was published that Barclays, which is the parent company of Barclaycard, has built its HCE and tokenization technology in-house [137].

An example of a third party provider offering HCE solution with Secure Element in the Cloud including Tokenisation Management, Gateway and Token Service Provider is Rambus Bell ID. Rambus Bell ID is certified by Visa, MasterCard and American Express [138]. Their solution leverages HCE to emulate an EMV mobile payment via a remote SE, and provides the functionality to complete a payment transaction using a standard EMV contactless payment terminal. Public available information shows that ANZ Bank have implemented their HCE solution compatible with Android mobile phone which is available on Google Play Store. ANZ goMONEY WALLET supports a range of ANZ Visa payWave credit or debit cards, and ANZ American Express contactless credit cards [139]. Their Mobile Pay app can also be used to withdraw money at any contactless enabled ANZ ATM [139].

### 3.3.3 Summary on the Advantages and Disadvantages

Both deployment models described above has its own advantages and disadvantages. This section will briefly go through them. In the banking and finance industry having a number of deployment options leads to more competitive pricing and innovative strategies. As at November 2016, no transaction fees were being incurred by cardholders when using any of the services by the two deployment models.

A common characteristic is that these models rely on technology factors which are deemed to fail. A case in point is when the mobile device is out of battery, or has connection issues with Issuers or TSP network. Thus, backup methods need to be in place such as still issuing physical contactless cards to cardholders.

The table in Figure 16 summarizes some advantages and disadvantages between the different models:

Criteria	OS vendor wallet (e.g. Android Pay)	Issuer Wallet	
		Own Infrastructure & Development	Third-party Infrastructure & Development
<b>App/Wallet Customization (branding)</b>	Limited	Yes	Yes
<b>Certification expenses</b>	None	High	Medium
<b>Setup/Deployment Costs</b>	Low	High	Medium
<b>Running Costs</b>	Low	Medium	High
<b>Per transaction fees</b>	Possible (currently none)	None	Possible (based on contract signed with TSP)
<b>Multiple card schemes</b>	Yes	Possible but complex	Yes
<b>Issuer Marketing</b>	Not Possible	Yes	Yes
<b>Cardholder Verification</b>	Chosen by OS Vendor	Chosen by Issuer	Chosen by Issuer

Figure 16 – Summary of the advantages and disadvantages between the different models

When an Issuer does not operate its own solutions, monthly or per transaction fees will be incurred if TSP platform card scheme or TSP third party is being used. However, this might be a viable solution for small Issuers due to the cost involved for development and certifying the TSP platform as well as other certifications such as PCI DSS, ISO27001, card scheme specifications, etc.

Despite such complex task, large Issuers may prefer to have the setup in-house as it would be costly to pay the fees associated with TSP, credential management and per transaction charges associated with the infrastructure.

Issuers have to also consider security issues and the cost to maintain the infrastructure up to standard. This means updating the wallet app with new updates, adapting to new versions of the OS, etc. On the other hand companies such as Google and Microsoft in collaboration with card schemes can take all the steps and actions to offer better security when using their platforms due to the available resources to implement security features and are able to provide instant security vulnerabilities updates to ensure their reputation is not jeopardised. Due to restricted budgets in the cards market, Issuers might not be in position to implement top end security features and controls or otherwise might not have the right or in-house skills to do so.

When an Issuer does not operate its own solutions, monthly or per transaction fees will be incurred if TSP platform card scheme or TSP third party is being used. However, this might be a viable solution for small Issuers due to the cost it involves in for development and certifying the TSP platform as well as other certifications such as PCI DSS, ISO27001, card scheme specifications, etc. Despite such complex task, large Issuers may prefer to have the setup in-house as it would be costly to pay the fees associated with TSP, credential management and Google transaction (currently no fees being charged).

When using an OS vendor model, Issuers are limited in what content they can push towards their cardholders. Merchants affiliated with the OS vendor can have their campaigns delivered through the mobile wallet app but Issuer cannot leverage the HCE mechanism to do this in an OS Vendor model. In an Issuer model, Issuer are free to implement their own content delivery mechanism. They can provide other services such as offers, promotions and integration with mBanking, ATM withdrawals being offered by the bank through the Issuer's app.

Using Android Pay, Issuers may have an impact on their branding and independence. Issues that may occur within Android Pay will have a ripple effect on the Issuer as customers will associate the issue with the Issuer. Another disadvantage, of using Android Pay is that Google might filter information on payment transactions or any other relevant information and share it with merchants or other third party companies for commercial and advertising purposes. Greater control can be exercised on transaction management for Issuers with their own solution. Furthermore, Issuers have the full power to decide which options they will support, basing their choice on the services that meet their requirements best and their risk appetite.

During the life cycle of HCE payment process, cardholders might need some form of customer support. This can be easily provided by the Issuer, whilst using Android Pay, customers might end up ping ponging from various stakeholders. In terms of maturity, there is a greater chance that Android Pay to be used rather than the respective Issuer's own HCE solution, due to the branding advantages Google possesses in the industry and worldwide. In fact, it is the author's opinion that as at November 2016, only very few Issuers have implemented their own HCE and tokenisation infrastructure in-house.

## 4 Modeling an HCE Payment Transaction

The aim of this section is to develop a model of an HCE payment transaction that will aid in the security analysis done later in Section 5 of this project. First the modelling tools available and their features vis-à-vis the requirements of this project are evaluated. Next the methodology explaining the modelling processes is provided and eventually an explanation of the developed model is provided.

### 4.1 Review of modelling tools and their application in modelling similar payment transactions

A state machine consists of inputs, some intermediate processing and outputs, typically following a sequential system. The machine ‘transitions’ from one state to another either by default or based on some condition. This type of modelling has been used to model many types of systems including computer systems, industrial processes and network protocols. Finite state machine analysis is also a common tool for security analysis. For example Mitchell, Shamtikov and Stern [140] used finite state analysis to model and determine issues in the SSL 3.0 protocol. Marrero, Clarke and Jha [141] proposed methods to develop a state machine model verification tool to determine issues within a security protocol.

The authors’ objective of using a state machine model is to gather all the states that the transaction could fall into and see how an HCE Payment Wallet transitions between states. This simplification provides a window for analysis which essentially could outline security risks which are otherwise difficult to observe using other ways. This process is more commonly known as ‘state machine verification’ or ‘model checking’. The tools used to ‘verify’ and/or ‘check’ a state machine model will exhaustively search for all possible execution traces (i.e. set of sequential transitions). Such search will provide all possible states and traces the machine can run into and this could outline issues in the implementation or programming of the machine under study.

For this purpose, five tools for the modelling of a state machine are reviewed. The tools are: Fizzim [142], Mobius [143], UModel [144], QM [145] and Stateflow – part of Matlab – Simulink [146]. A brief overview will be outlined.

#### i. Fizzim

Fizzim is free open source tool under the GNU public license which is specifically targeted for the engineering community to be used for developing code for systems by modelling them graphically as state machines. The latest version of the tool traced was version 5.20 released on the 5<sup>th</sup> August 2016 and is supported on Windows, Linux, Apple and any other software that supports Java.

The tool offers the option to intended users to setup their states information through the GUI interface and the software itself generates code. Other functionality available include Mealy and Moore outputs, transition priority and automatic grey coding.

#### ii. Mobius

Möbius is a software tool for modelling systems using a finite state machine approach. This tool was created as part of a research project at University of Illinois at Urbana-Champaign. The aim of the tool is not to generate/develop code for a machine but rather to study the reliability, availability and performance of computer and network systems. The tool allows model creation through a graphical interface but it also provides the ability to write complex processing code during a state through programming languages such as C++. The software also provides the ability to study a system’s behaviour under different condition provided as inputs to the model. Mobius is free for Academic use but requires a licence for commercial purposes.

### **iii. UModel**

UModel is part of a suite of software developed by Altova. UModel is a tool for developing software through UML diagrams. State Machine diagrams are one of the types of UML diagrams and UModel allows developers to develop software by modelling it as a UML State diagram. The tool takes an object oriented approach since the output of the tool is object oriented code such as Java or C#. To access this software tool a license is required. Various licenses exist including Enterprise and Professional editions.

### **iv. QM**

QM (QP Modeler) is a similar tool to UModel for the development of code through UML statechart diagrams. The tool is tailored for developing code for embedded systems. The tool is provided as freeware but it is not open source.

### **v. Stateflow (Matlab and Simulink)**

Stateflow is a tool developed by MathWorks within the Matlab software suite. Particularly Stateflow is part of Simulink. Stateflow is a tool used to model a system via state machine diagrams and state charts. Unlike other tools, Stateflow uses an enhanced state machine diagram capable of modelling hierarchy, parallel processing and history. Stateflow also provides the ability to automatically generate state transition tables. These are similar to truth tables and shows which inputs make a machine go into which state.

Stateflow does not generate any code but it allows a simulation of the model to be run. Moreover, as it is part of Simulink it allows the use of other tools within Simulink and Matlab to be used in the model. MATLAB is a well-known tool by both the industry and academics. A comprehensive support services, tutorials, documentation, for using the tool is also offered by the company publicly on their website.

Simulink also provides a tool known as Simulink Verification and Validation [147]. This tool can be used to run coverage analysis to exhaustively search for all the possible execution traces in the model (known as 'coverage').

From all the tools analysed it was clear that the tools were either geared towards software development or modelling for systems analysis. Fizzim, UModel and QM are aimed at software development while Mobius and Stateflow's are more geared towards system analysis. Ultimately, the choice was reduced to either Mobius or Stateflow.

Both Mobius and Stateflow provide the tools and features required for the project's scope but Stateflow was the preferred tool due to the following points:

- It has a large support community.
- It allows the use of Matlab and Simulink tools to be added as part of the model.
- It allows for parallelism which is a requirement for this project as will be explained further on in this section.
- It provides a model verification tool.

## **4.2 Methodology**

### **4.2.1 Acquiring reference documentation**

Prior to starting the documentation a good introduction was acquired by reading the paper “EMV in a nutshell” [148]. Such paper covers the key processes of EMV Contactless payment transaction between the card and the terminal and the similarities to contactless payments using NFC mobile phones. The authors in this paper analysed both the Visa and MasterCard payment brands.

Such paper served as a good start to help me understand the relative processes and mechanisms such as data authentication, cardholder verification methods, the transaction, cryptogram generation and key infrastructures.

Eventually, the focus was narrowed to the Visa payment scheme, given that the author had limited access to Visa’s material and none to MasterCard. The model in this project is based on the following documentation:

#### 4.2.1.1 Transaction Payment Process

#	Document Name	Availability <sup>4</sup>	Abbreviation (in Model)
1	<p><b>Visa Cloud-Based Payments Contactless Specification - Visa Supplemental Requirements Version 1.7 - May 2016 [130]</b></p> <p><u>Scope of Document:</u> This is the backbone document that was used to build the model. This document provides detailed specification for a Mobile Application to communicate with a contactless reader over NFC, while supporting the Visa payWave transaction flow irrespective of the application's location (i.e. SE or cloud). It also provides the 'mobile specifications' where HCE is covered for a Visa payWave transaction.</p> <p>This document was used for the following processes:</p> <ol style="list-style-type: none"> <li>i. <u>Transaction Requirements:</u> <ul style="list-style-type: none"> <li>- account enrolment, account parameters (e.g. account parameters replenishment requests and responses);</li> <li>- qVSDC transactions data elements and requirements (i.e. APDU commands, PPSE<sup>5</sup>, AID, GPO, CVM processing, offline data authentication);</li> <li>- Read Records<sup>6</sup>;</li> <li>- Transaction Verification Log<sup>7</sup>.</li> </ul> </li> <li>ii. <u>Cryptograms and Signature:</u> <ul style="list-style-type: none"> <li>- Building Cryptograms and signature for Mag-Stripe Mode(MSD) and qVSDC contactless options;</li> <li>- Provisioning and algorithms of the cryptograms.</li> </ul> </li> </ol>	Private	VCBPCS

<sup>4</sup> Documents which are not publicly available were accessed through the official site of VISA which is accessible to its members. The author's workplace is a member of VISA.

<sup>5</sup> The Mobile Application receives a select command containing the information "2PAY.SYS.DDF01"

<sup>6</sup> The read record command reads a file record in a linear file in response to the GPO

<sup>7</sup> After executing a successful transaction, the mobile application stores information about the transaction in this log. The information consists of: mobile device UTC timestamp, unpredictable number received from the reader, application transaction counter (ATC) tag and the type of transaction whether MSD or qVSDC



#	Document Name	Availability <sup>4</sup>	Abbreviation (in Model)
2	<b>Visa Mobile Contactless Payment Specification (VMCPS) - Visa Supplemental Requirements Version 1.4.3 - May 2015 [149]</b>  <u>Scope of Document:</u> This document was used to understand the concepts of qVSDC Transaction Flow (i.e. Pre-Tap or 1 <sup>st</sup> Tap) <sup>8</sup> , Issuer Update Processing (i.e. 2 <sup>nd</sup> Tap) <sup>9</sup> , External authenticate command and Issuer Script processing flow.	Private	VMCPS
3	<b>Transaction Acceptance Device Guide (TADG) [150]</b>  <u>Scope of Document:</u> This document was used to understand the concepts of CDCVM request and response.	Public	TADG
4	<b>EMV Contactless Specifications for Payment Systems (Book C-3) - Kernel 3 Specification - Version 2.6 - February 2016 [151]</b>  <u>Scope of Document:</u> This document was used as a reference together with VCBPCS and VMCPS. The document provides the expected behaviour from a POS when an HCE device communicates with the POS.	Public	EMV_K3
5	<b>Implementing an HCE Service - <a href="https://developer.android.com/guide/topics/connectivity/nfc/hce.html">https://developer.android.com/guide/topics/connectivity/nfc/hce.html</a> [55]</b>  <u>Scope of the link:</u> This link was used to determine how the reader and mobile respond to an AID command, how an HCE service is started and how data flows to and from the NFC controller.	Public	AndroidOnline

<sup>8</sup> Initial physical presentation of the consumer device to the contactless payment reader

<sup>9</sup> Second physical presentation of the consumer device to the contactless payment reader to perform an Issuer Update (reset of application counters and possible issuer scripting)

#### 4.2.1.2 Tokenisation Process

#	Document Name	Document Source	Abbreviation (in Model)
1	<b>Visa Token Service - Introduction for Issuers - Version 3 - November 2016</b> [152] <u>Scope of Document:</u> This document was used to understand the cardholder verification process. Visa uses an authentication mechanism called step-up authentication. The mechanism supports different method such as a one-time-password generated by Visa's payment token service <sup>10</sup> .	Private	VTSII
2	<b>Visa Token Service - <a href="https://developer.visa.com/products/vts/reference">https://developer.visa.com/products/vts/reference</a></b> [153] <u>Scope of Link:</u> This consists of Visa Token Service APIs published on Visa Developers website for (e-Commerce and m-Commerce) and mobile contactless. This information was used to understand the device enrolment process and how tokens are provisioned and/or replenished and the type of cryptograms applied. Also, information on the Token Lifecycle processes (i.e. Resume, Delete, Active and Suspend) was used for reference.	Public	VisaDeveloperOnline
3	<b>EMV Payment Tokenisation Specification - Technical Framework - Version 1.0 - March 2014</b> [25] <u>Scope of Document:</u> This document was used for gathering insight on the Tokenisation process in general.	Public	EMV_PTS
4	<b>Information on limited use keys (LUK):</b> <a href="http://blog.simplytapp.com/2014/09/apple-pay-and-android-payment-eco.html">http://blog.simplytapp.com/2014/09/apple-pay-and-android-payment-eco.html</a> [154] <a href="http://developer.samsung.com/tech-insights/pay/token-handling-by-samsung-pay">http://developer.samsung.com/tech-insights/pay/token-handling-by-samsung-pay</a> [155] <u>Scope of the Links:</u> These links were used to understand principles of token storage location and how cryptograms (master keys (MDK), Unique derived Key for Issuer per card (UDK) and dynamic keys (LUKs) are applied.	Public	LUK_SA_TAPP

The abbreviations mentioned in the tables above have been used as part of the comments in all the three Stateflow charts and in their respective Matlab '.m' script files.

<sup>10</sup> Visa is a Token Service Provider in this context

## **4.2.2 Developing the model**

### **4.2.2.1 Conceptual Architecture**

When all the documentation was gathered, the first process in developing the model was to define a conceptual architecture. The mobile device is considered as a finite state machine but due to the capabilities of the mobile device and the way HCE payment wallets are implemented, different configurations are possible. The scope of the model will be limited to the processes involving the HCE payment wallet. The most important points considered in developing this conceptual model are:

- The mobile device is capable of multi-threading (multi-processing). This essentially means that the mobile device can have more than one 'machine' running simultaneously.
- For security reasons, some processes, such as the generation of the cryptogram or LUK does not occur in the HCE payment wallet App. This process occurs in the cloud or in a TEE. This process is still considered part of the model as it is seen as an extension of the HCE payment wallet App.
- The 'independent' processes or threads share the same data or database. Synchronization between the processes or threads are outside the scope of this model.
- The processes in the mobile device could be affected by other processes running in the mobile device which are not related directly with the HCE payment wallet. For example, the user could lock the mobile during a payment transaction. These are outside the scope of the model.

### **4.2.2.2 Defining States and their Transitions**

Once a conceptual architectural was developed (further information can be located in Section 4.3.1) the next step was to study in detail the documentation on each of the processes identified. These processes resemble a machine with a finite set of states hence the next step was to define all the possible states within the 'machine'. The machine is essentially executing a different processes in each state defined such as generating a cryptogram or waiting for a request from the POS.

The next task prior to developing the model in Stateflow was to identify the transitions between the states. Transitions are conditions that occur and take the machine from one state to the other. Some states have default transitions (i.e. the state would transition to another state after a specific process is completed, and other states can have multiple transition possibilities based on different conditions and priorities).

### **4.2.2.3 Defining Input and Output data**

The final task was to define the input and the output data. The model uses different kinds of data. Part of the data is 'standard' data according to EMV or Visa specifications. The structure of this data was preserved as much as possible to these specification as to make it easier for analysis. The other data resembles conditions or settings within the mobile device. The data was grouped into 'structures' which was then routed into the model as a signal bus.

The actual model was developed using the graphical tools provided by Simulink, specifically the tools within Stateflow. Similar to how state machines are represented, in Stateflow, states are defined as boxes. Processing done within a state is declared as Matlab code within the state box and transitions are modelled by drawing lines from one state to the other. Conditions for the transitions are also entered as part of the line in Matlab code.

Stateflow defines different kinds of data representation, but for the scope of this project three types were used; input, output and local. Data which is 'fed' into the model is registered as input. Local data represents structures which the model alters but do not provide data to external entities while output data is provided

to external entities such as data provided to other processes. The data structure and contents was defined in Matlab '.m' script files.

To ensure that the model works a sequence of simulations were run. Different variables were initialized to ensure that all states are executed at some point.

### 4.3 Development of an HCE Payment Transaction Model

#### 4.3.1 Architecture

After analysing the documentation made available by EMV and Visa it was concluded that an HCE payment wallet can be modelled as three state machines (i.e. processes) which can run in parallel. Given a mobile device is capable of multi-threading or running multiple processes in parallel, such a configuration can be achieved. The three processes identified are:

- **Payment Transaction** – responsible for NFC communication with the POS.
- **Token Requestor** – responsible for the provision of tokens on the mobile device from the Token Service Provider. Also responsible for the replenishment of limited use keys on the mobile device related to a token in use.
- **Account Management** – responsible for the provision/replenishment of account parameters such as limited use keys and the overall management of an account.

The Payment Transaction and the Account Management processes are part of the HCE payment wallet. The Token Requestor process could also be part of the HCE payment wallet app but as a good practice this is not recommended as the token requestor is in possession of the Unique Derived Key (UDK). This is an important key as it is used to generate the Limited Use Keys (LUKs) used by the Payment Transaction process to generate the cryptogram [154]. Figure 17 below shows at a high level the generation of the tokenized PANs (tPANs), the UDK (tUDK) and the LUKs. This shows the importance for storing the UDK in a secure environment.

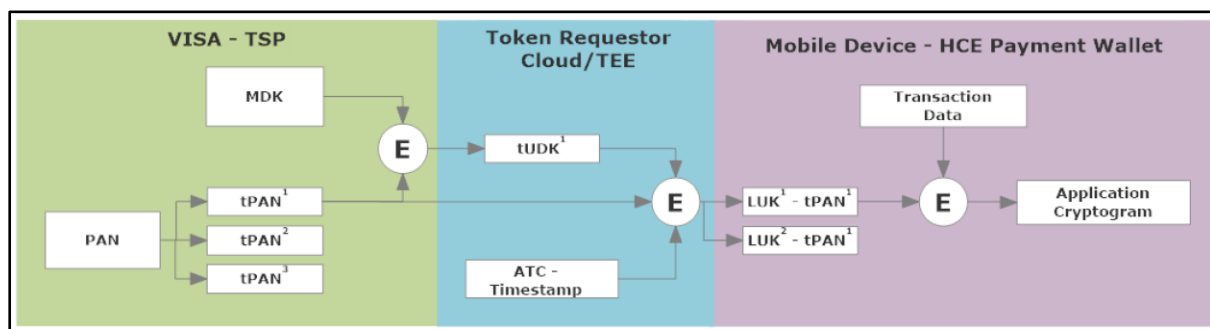
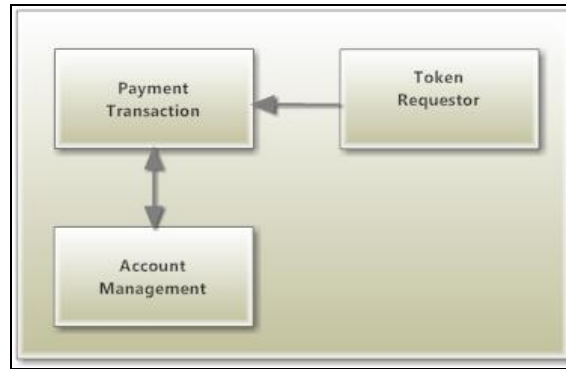


Figure 17 – Generating the Application Cryptogram<sup>11</sup>

Hence the Token Requestor process is normally located either in the TEE of a mobile device or in the cloud, provided by the wallet provider or a third party. Both the TEE and the cloud provide a 'safer' environment when compared to an HCE service App hosted in a non-secure environment in the mobile device.

<sup>11</sup> E in the diagram resembles some form of encryption algorithm. (e.g. Triple DES)



**Figure 18 - Model Architecture**

As shown in Figure 18, the three processes above were modelled as three separate Stateflow charts which are capable of running (or simulated to run) in parallel. Data between the processes flows through a series of Simulink busses. Each chart (process) has its own set of data inputs, outputs and local data structure. The data from one chart is not accessible directly from within another chart. But Simulink busses are used to make data available between the charts.

The Token Requestor process forwards the LUKs to the Payment Transaction process. The payment transaction process provides the verification logs to the Account Management process and the Account Management provides the account parameters to the Payment Transaction process.

Figure 25 shows further detail on the Simulink busses (i.e. signals) connecting the three processes together.

### **4.3.2 Data and Structures**

Data in Simulink ‘travels’ by means of signals. A signal resembles a variable or a condition in the mobile wallet app. When signals are grouped together they form a Bus. A Bus resembles a structure of data or, to an extent, a class since the signals in a Bus resemble the variables or properties of a class or data structure. Certain busses in the model also feature ‘nested busses’ which is again similar in notion to that of a nested class or nested structure. For the purpose of this project the term data structure is used but essentially, in Simulink, this is modelled as a Simulink Bus.

One important concept which differentiates a signal from a variable is the fact that a signal is expected to vary in time. This means that a signal is made up of a 2 dimensional array comprising of a value and a time value.  $[[Value^1, Time^1], [Value^2, Time^2], [Value^n, Time^n]]$ .

In the model developed, all the data that is fed into the model is considered as ‘constant’ and, as such, the data is only initialized once prior to running the model. The data is then fed into the stateflow chart as input data. Data and data structures that the model will process and vary during runtime is copied, during the initialization state (or in other specific states if required), into a local data structure or an output data structure. Input data structures are read only while local and output data is read/write.

Figure 19 summarises the bus names and whether these sweep data as an Input, an Output or Local data.

Annotation: I = Input, O= Output, L = Local

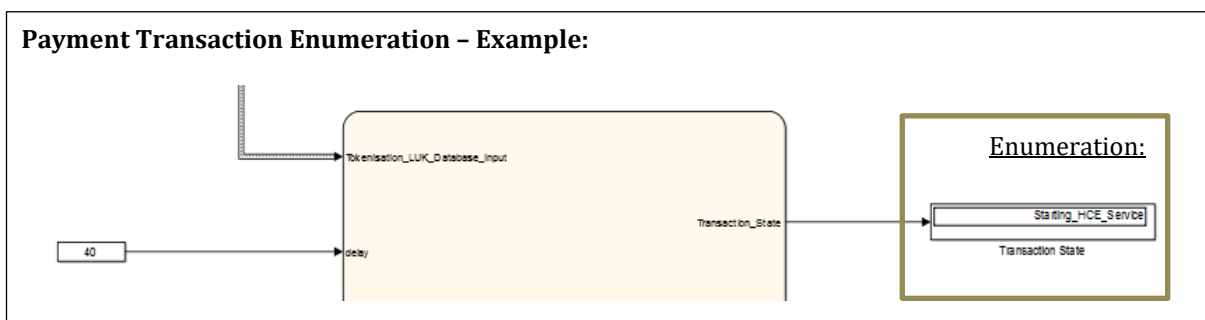
Bus Name	Description	Payment Transaction	Account Management	Token Requestor
Account_Data	A structure holding all the data related to an account (card).	I,L		I, L
Mobile_Device	A structure holding data related to the mobile device, its conditions, configuration, etc.	I, L		I, L
PDOL	An EMV standard structure. The structure is data provided by the POS to the mobile device.	I		
Issuer_Update	A structure holding data used during issuer update processing. An Issuer_Update structure is tied to an account	I,L		
Transaction_Verification_Log	A Visa standard used by the mobile device to hold a log of all the transaction. The structure is then used as a decision mechanism to update account parameters.	O	I	
LUK_Database	A structure holding the Limited Use Keys of a specific token.	I		O
Account_Parameters	Used to store keys related to an account along with thresholds and configuration for risk management. The structure is essentially an extension to Account_Data structure.	I	I,L,O	
Token_Data	A structure that holds the data related to tokenisation for a particular account			L

Figure 19 - Input-Output Model

### 4.3.3 State Enumeration

In order to have an audit trail in which state position each of the three processes operate, an enumeration class has been created for each of the three processes. The class simply defines each of the state in an enumeration. Enumeration is used to facilitate the understanding of the respective process to a user.

During the execution of the model by the compiler, each chart has an output displaying the enumeration as a means of visual to a user showing the state each process is currently in. The enumeration for each state process is being shown in Figure 20.



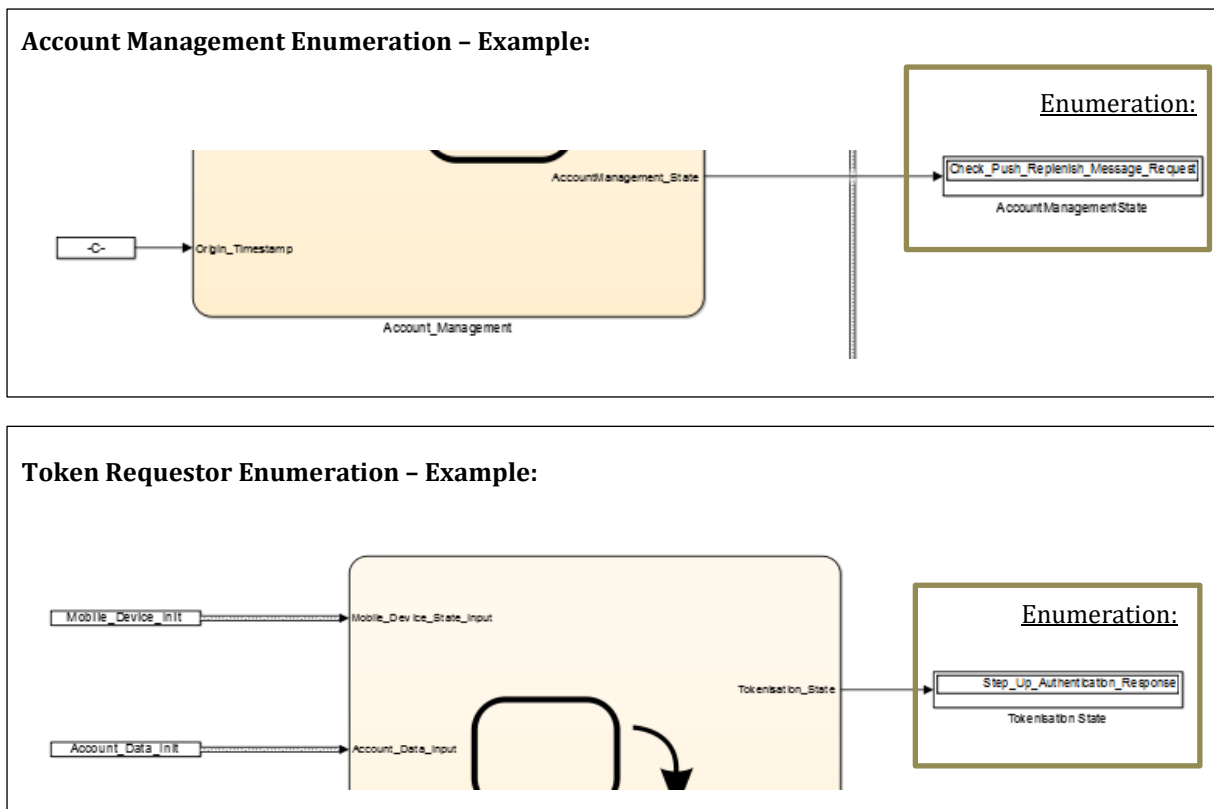


Figure 20 – Current State Output

#### 4.3.4 Connective Junction

The connective junction is used to split the flow of a transition into different paths. This is similar to an “If then else” decision. When a machine is required to transition from one state to multiple states the connective junction is used to decide (based on conditions) to which state it will transition. Transitions also have priorities. From within a connective junction conditions are evaluated according to the priority (defined in numbers with 1 being highest priority).



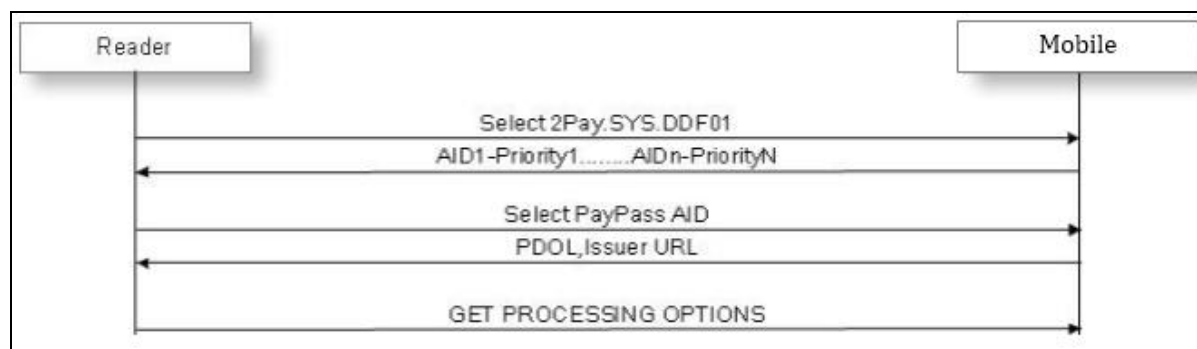
Figure 21 – Connective Junction Example

#### 4.3.5 Payment Transaction Process

The payment transaction process handles all the processing done by the mobile device from the point when the mobile is tapped on a POS until the transaction is complete whether approved or declined, including the possibility of a 2<sup>nd</sup> tap used for Issuer update processing. The total transaction time shall not exceed the 500 milliseconds (i.e. 0.5 seconds) [130].

The Payment Transaction Process chart has 37 finite states representing all the states the mobile device could be in during a payment transaction. This payment transaction process, throughout execution, is also responsible for updating counters and the verification log used in the account management model.

The main 'flow' of this process starts with the user tapping the mobile phone on the POS. The mobile then communicates, via NFC, with the POS up until when the POS sends the PDOL data. An explanation of the sequential steps can be traced below in Figure 22:



**Figure 22 - Initiation of Payment Transaction**

The POS will then provide a specific command (i.e. GPO) which sends the flow to either a payment or a second tap (i.e. External Authenticate command/Issuer Update command). If a payment is being processed then the flow is split again between an MSD payment and a qVSDC payment. These are the 2 options supported for a Visa transaction. Further information on the differences between these 2 options can be traced in [148]. The GPO command through the Terminal Transaction Qualifier (TTQ) tag indicates which mode is to be selected.

After this stage, the POS and the Mobile device will agree on a Cardholder Verification method and the payment transaction will proceed with the mobile generating the signature for ODA<sup>12</sup>, the IAD<sup>13</sup> and the Application Cryptogram. This data is then transferred to the POS through a series of read records initiated by the POS. If the POS and the mobile device agree to use CDCVM<sup>14</sup> then the mobile must ensure that the cardholder has been already verified and if not ask the user to go through the CDCVM of choice (e.g. fingerprint). The model does not cover which method and how such method is implemented but one state is declared where CDCVM occurs.

The model supports cryptogram generation for both with and without tokenisation. The algorithm used to generate the cryptogram and the data used is outlined in Figure 17. The difference between using tokenisation and not using tokenisation is in the LUKs used. In case tokenisation is used then the LUK is obtained from the token requestor while if no tokenisation is used then the LUK is obtained from the Account Management process through the Visa Cloud Platform. Furthermore if no tokenisation is used then the PAN is used instead of the tPAN.

At the end of a transaction the payment transaction process will update the respective counters and transaction verification log. As part of the processing within the payment transaction process, this model will also issue a command to the Token Requestor process requesting to replenish the LUKs with every transaction. In a practical implementation, a set of keys are downloaded for a token and these are replenished once depleted, but in this model it is assumed that only one key is downloaded, stored and replenished. This does not affect the model from a security analysis point of view. At this stage, the transaction is now complete.

For a detailed analysis refer to Figure 26, which illustrates a diagram of the Payment Transaction process.

<sup>12</sup> Offline Data Authentication

<sup>13</sup> Issuer Application Data

<sup>14</sup> Consumer Device Cardholder Verification Method



#### 4.3.5.1 CDCVM

The CDCVM process starts after the mobile device receives the GET PROCESSING OPTIONS command. The mobile application uses the PDOL data obtained from the terminal and its internal preference settings to decide on using CDCVM for Cardholder Verification. If this is the case, the mobile checks to see if CDCVM has been already performed. If CDCVM has already been performed then the mobile will update the CTQ and CVR data structures and progress with the transaction, eventually generating the cryptogram and providing the data through a series of read records commands.

If CDCVM is not yet performed the mobile device responds to the GPO command with a 6986 message indicating to the terminal that the mobile device is not a ready to pay state. At this point the NFC communication with the terminal is terminated and the mobile application instructs the user to perform a CDCVM (e.g. fingerprint scan). When CDCVM is ready, the mobile application asks the user to re-tap the phone on the terminal and the whole process will start again from the beginning. This time when the mobile receives the GPO command, it will decide to use CDCVM but it would have been provided and hence it will progress with the transaction instead of asking the user to do a CDCVM.

#### 4.3.5.2 Issuer Update

Issuer update is performed in a 2<sup>nd</sup> tap. After a payment is made, the terminal informs the cardholder to tap the mobile for a second time<sup>15</sup>. The first part of the 2<sup>nd</sup> Tap is the same process as that used during a payment transaction but instead of issuing the GPO command the reader issues the EXTERNAL AUTHENTICATE command. The Mobile device then verifies that this is the first external authenticate request received after the payment and generates a cryptogram from the ARPC and ARQC in the last payment. This cryptogram should match the ARPC sent by the Issuer during the External Authenticate process. In this operation the mobile device authenticates the Issuer, ensuring that the request is coming from the genuine Issuer rather than a fraudster trying to run a script in the mobile wallet application.

If the authentication fails the Mobile Device responds to the reader with a 6985 or 6300 response indicating that this is not the first external authenticate request or the Issuer Authentication failed respectively.

Finally, if the authentication of the Issuer is successful the application counters are reset and if provided, the Issuer script is run or interpreted. The Issuer script is a set of commands, set by the Issuer, to provide the ability to the Issuer to update and modify the parameters of the mobile application.

#### 4.3.6 Token Requestor Process

The Token Requestor is typically situated in a TEE or in the Cloud. However, other locations have also been specified in the EMV Payment Tokenisation Specification [25].

The Token Requestor manages tokens on behalf of the HCE wallet App. It is important not to mix up the Token Requestor with the Token Service Provider. A Token Requestor is unable to generate tokens and it can only obtain Tokens from the TSP. The functions of the token requestor processes in the model are listed below:

- Provision new token/s for an account upon enrolling an account with the TSP.
- Safely store the UDK in a token vault for a token and generate LUKs with that UDK for particular tokens.
- Download the LUKs on the Mobile Device according so some risk rules e.g. setup a rule limiting the maximum number of tokens that can be added to different devices for the same card/PAN.

---

<sup>15</sup> This is different from the 2<sup>nd</sup> Tap used during CDCVM

- Manage the state of the token for cases such as when a device is lost, card/account no longer valid, payment card expired and is replaced.
- Request additional cardholder verification, through the step-up authentication mechanism, prior to activating a token or at some point defined by a specific risk management ruleset.

The Token Requestor makes use of some account data, which is passed securely from the Mobile Device to the Token Requestor. The provisioned token corresponds to a valid PAN. Once account data is available, the token requestor will ask the Token Service Provider to provision a token and its UDK. These are stored securely in a token vault. The token requestor will then generate LUKs and download the LUKs onto the mobile device. It will also manage the state of the token including the possibility of suspending the token if the device is stolen, deleting the token or resuming it. A token is put into a suspended state if there is a possibility of finding a lost or stolen device without the token being compromised. If the device is declared permanently lost then the token is deleted. If the device is found then the token is resumed. Figure 27 illustrates a diagram of the Token Requestor process.

The model allows for only one LUK to be provisioned on the mobile wallet (in the payment transaction process). The model replenishes the LUK after each transaction. While this configuration is valid, typically TSPs provide a combinations of counters and limits as an indication to replenish the LUK. Figure 23 below consists of a LUK Configuration provided by VISA [29] for Android Pay. Such configurations vary according to the respective country. In such a configuration the LUK has to be replenished after 15 days or after 15 transactions, whichever comes first.

	LUK Parameters	Issuer available values for current	STIP Values for Current	Issuer available Valid values for previous	STIP Values for Previous	Comments
Must be the same across Wallets	TTL	15 days				Time to live in days after which replenishment will be triggered from the device
	Number of Transactions (NOT)	15 transactions				NOT after which replenishment will be triggered from the device

Figure 23 – LUK Configuration provided by VISA for Android Pay [29]

#### 4.3.6.1 Enrolment

The enrolment process starts by first enrolling the device. During this processes the TR register the device, specifically the wallet application, and from then onwards it will use this information to authenticate the mobile device when it attempts to communicate with the TR. The next step is for the user to enrol a PAN (credit card). This is done through the Enrol\_PAN state at which point the mobile device will initiate the step-up authentication mechanism. The model supports 2 types of mechanisms:

- Using the mobile banking app as authentication.
- Through an OTP provided by the Issuer.

If the mobile banking app is used, then the application transfers to the mobile banking app and the response from the mobile banking app will stipulate whether the card holder has been verified or not. If an OTP password is used then the TR will ask the Issuer to issue a OTP to its cardholder. The application then provides a way for the cardholder to enter the OTP. If the OTP matches the OTP sent by the Issuer then the cardholder is verified. If all this is successful then the TR is provided with the token data from the TSP. This includes a tPAN and its UDK. The tPAN is passed over to the device while the UDK is safely stored in the token vault.

#### **4.3.6.2 LUK Provisioning on the Mobile Device**

Once a PAN is enrolled and a token (tPAN and UDK) is obtained, the next step is to provision the Limited Use Keys for that particular tPAN. Note that, in terms of HCE, the token stored in the TR is a tPAN and UDK, while the tokens provisioned on the mobile device are tPAN and LUK.

The provisioning starts by the mobile device indicating to the TR that a token is required. This will put the TR into the provision token state. The TR will generate a LUK using the UDK and the tPAN and send the LUK along with other information to the mobile device. The TR also provides mechanisms to manage the token after it is issued including the option to Delete, Suspend and Resume a token.

#### **4.3.7 Account Management Process**

The Account Management process handles the replenishment of keys required when tokenisation is not used. Tokenisation is not a 'mandatory' requirement, however it is considered as a good practice by industry experts to be implemented. If tokenisation is not used then LUK are issued for a PAN through the VISA Cloud-Based Payments Platform.

Visa provides guidelines and methods to provision LUKs used to generate cryptograms during transactions. These LUKs are 'tied' to the PAN of a specific card and hence if a device is lost or stolen a user cannot simply change the PAN as is done with a tokenized PAN hence the only option is to reissue a new card. The LUKs do not need to be replenished with every transaction and specific risk management is applied and based on certain thresholds, these keys are replenished, depending on the rules configured.

The Account Management process uses the transaction verification log provided by the Payment Transaction to run the rules as part of risk analysis process and decide on when to replenish the keys. The keys are then provided back to the Payment Transaction process to generate the cryptogram. The "where" and the "how" these LUKs are stored are outside the scope of this model but in practical implementations these should be stored in a TEE environment.

##### **4.3.7.1 LUK Management**

The main function of the Account Management is to replenish LUKs when these expire. As shown above this is based on a number of factors including time, transactions, total amount of transactions etc. This risk analysis logic was developed in a Matlab function (i.e. Check Account Parameters Limits), which was appended using a connection point to this model. This function is fed data from the Account Management process and provides back a Boolean signal, indicating, when the LUKs require replenishment. The Boolean signal is used to transition the model into the replenishment state. This is being illustrated in Figure 24.

When new LUKs are required, the mobile device will go into the replenish account parameters state where it will communicate with the VISA Cloud-Based Payments Platform to download new LUKs. During this interaction it will provide the Transaction Verification Log, which is compiled at the end of each payment transaction. This transaction log and the LUK itself is used to authenticate the mobile device with the VISA Cloud-Based Payments Platform. If replenishment is successful the mobile device clears the counters used in the risk analysis above and the new LUK will be used to generate the cryptograms during payment transactions.

Furthermore, the Account Management process is being illustrated in Figure 28.

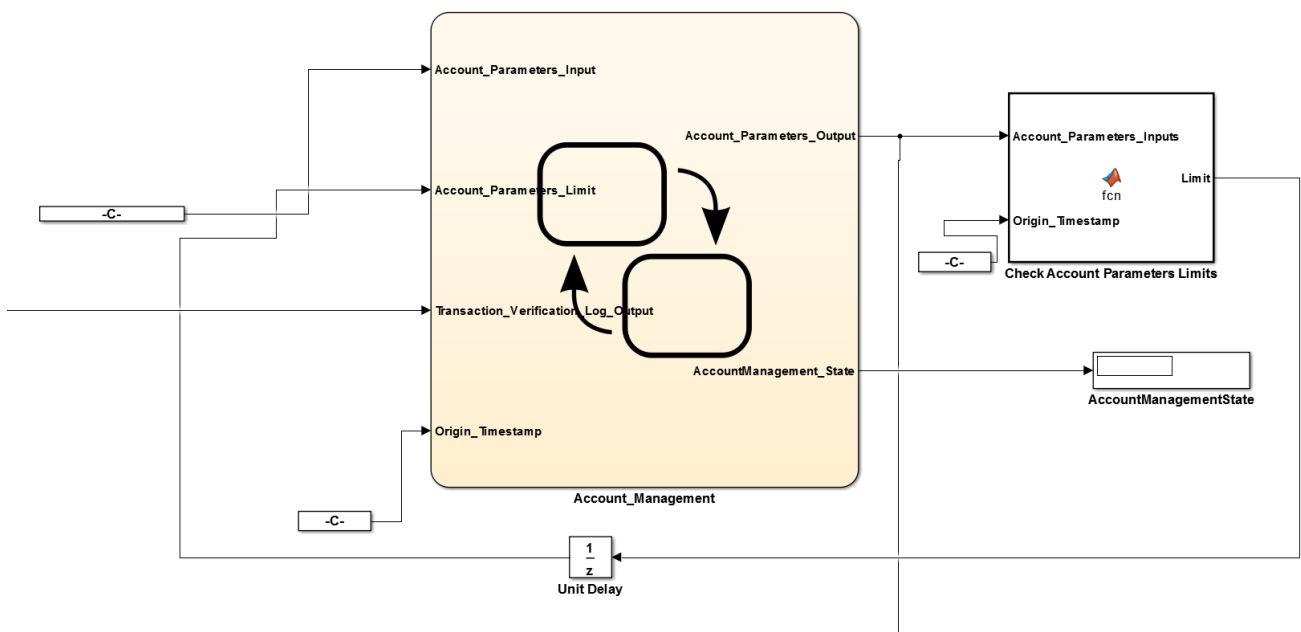


Figure 24 - Limit Checking as a MATLAB Function in Simulink

Figure 25 – Mobile Device Model Chart

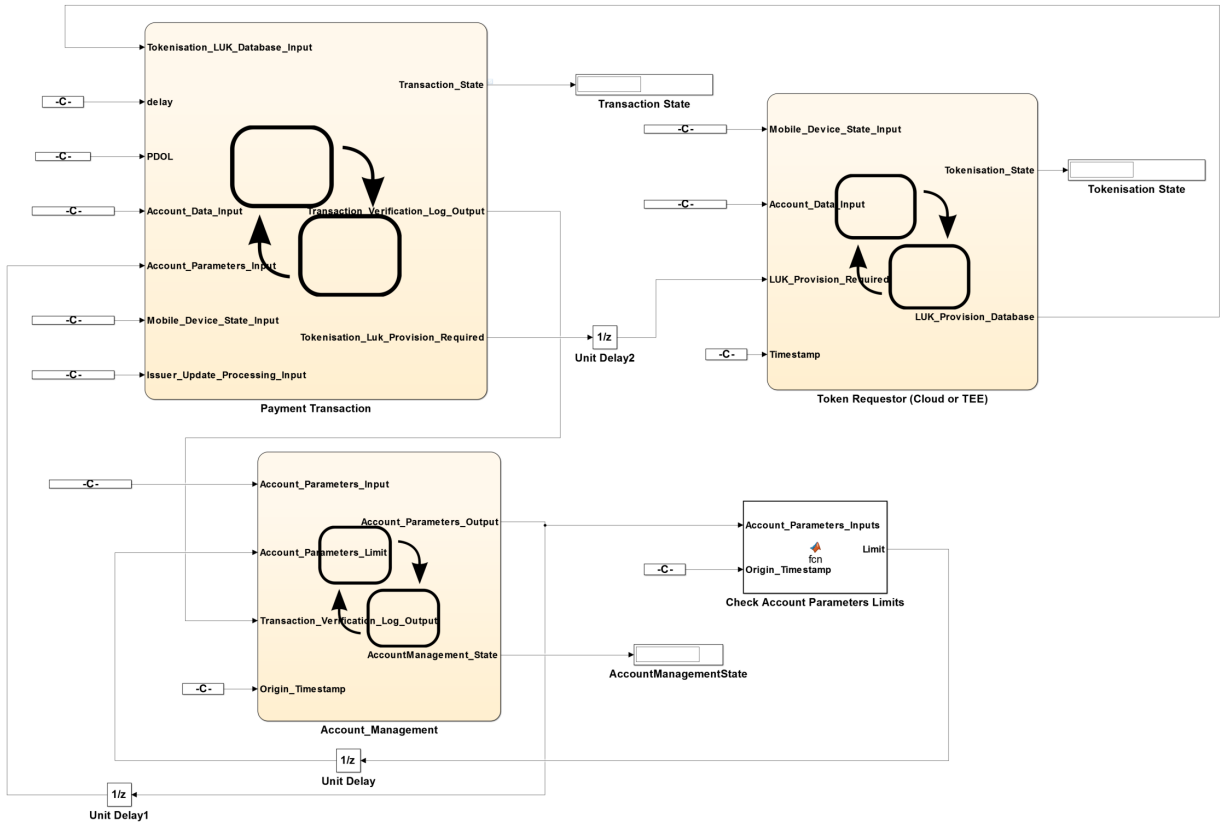
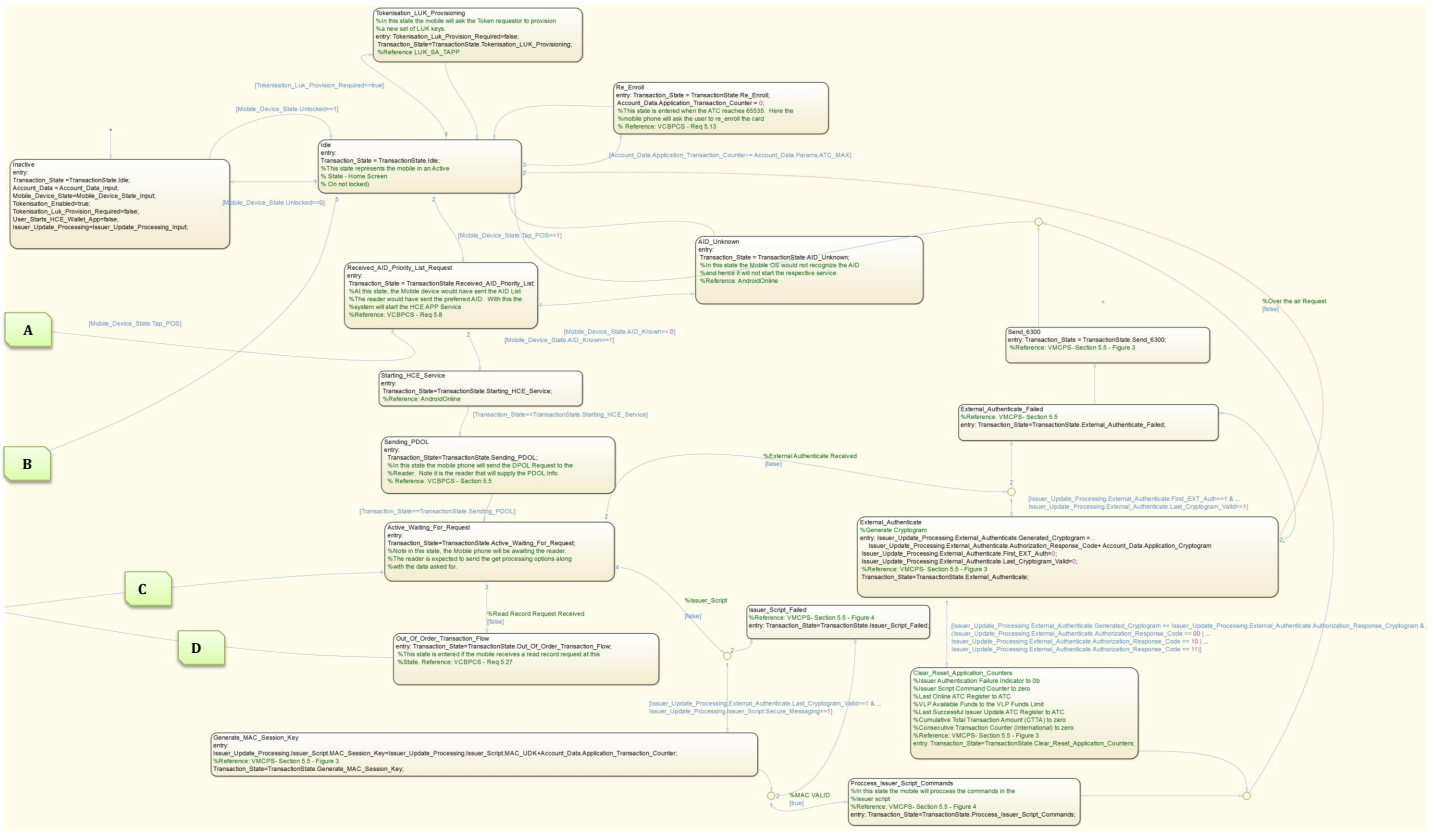


Figure 26 - Payment Transaction Process



Note: The connectors are only used in the word document due to the limitation that the model does not fit into one page

Figure 26 - Payment Transaction Process (continued)

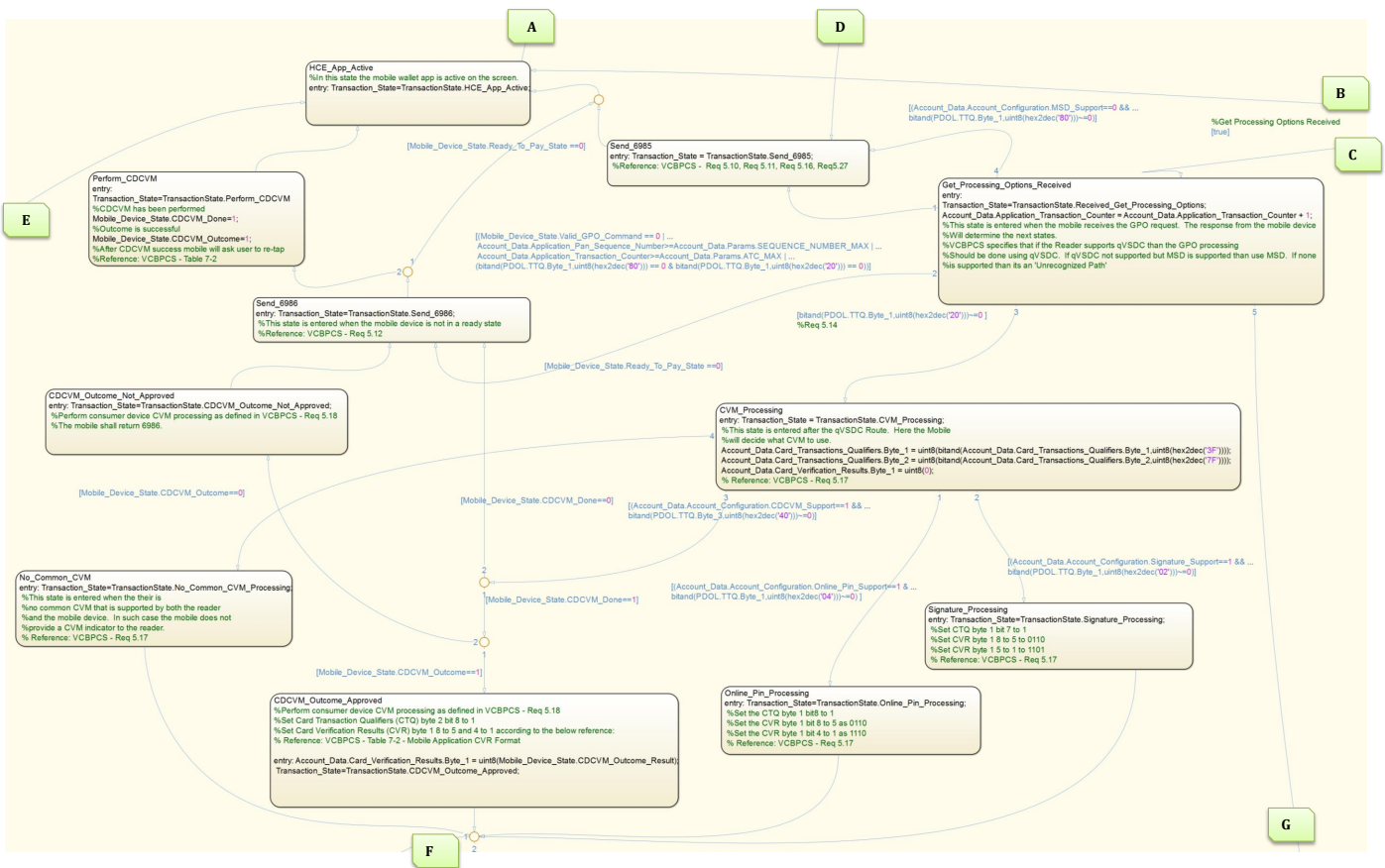


Figure 26 –Payment Transaction Process (continued)

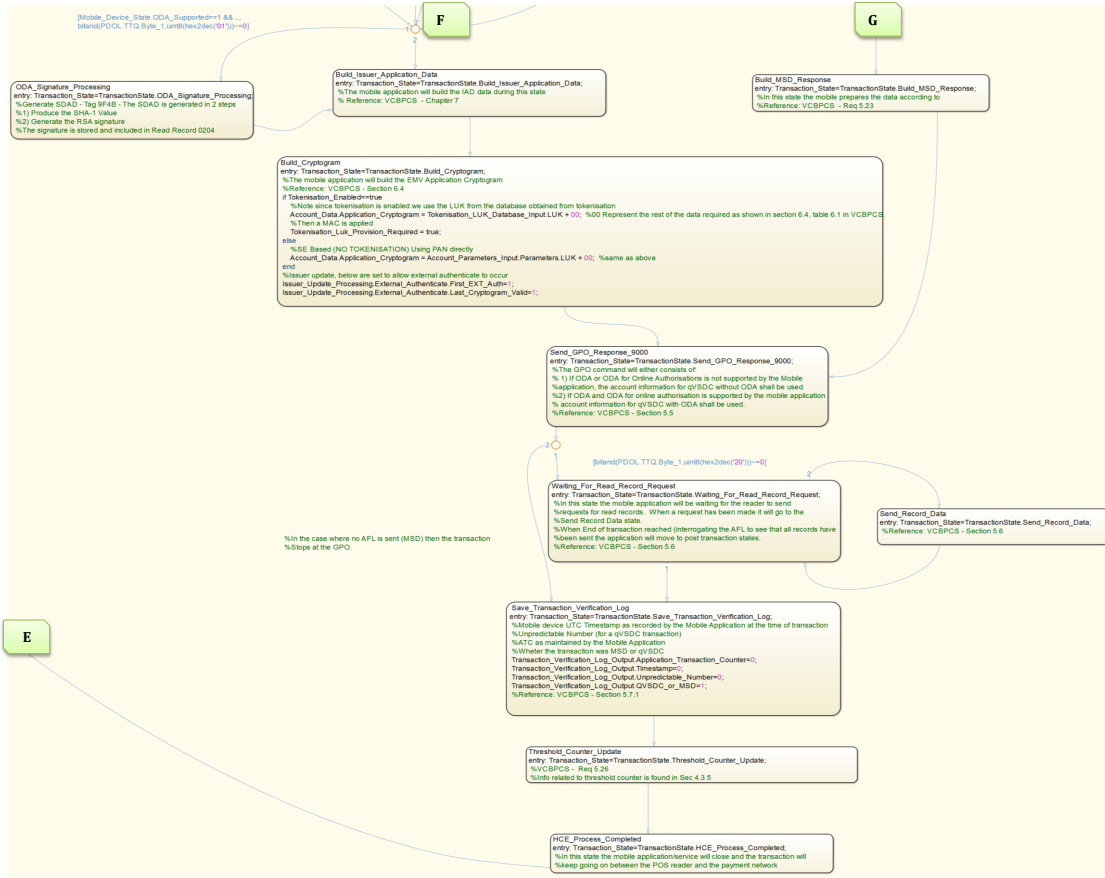




Figure 27 - Token Requestor Process

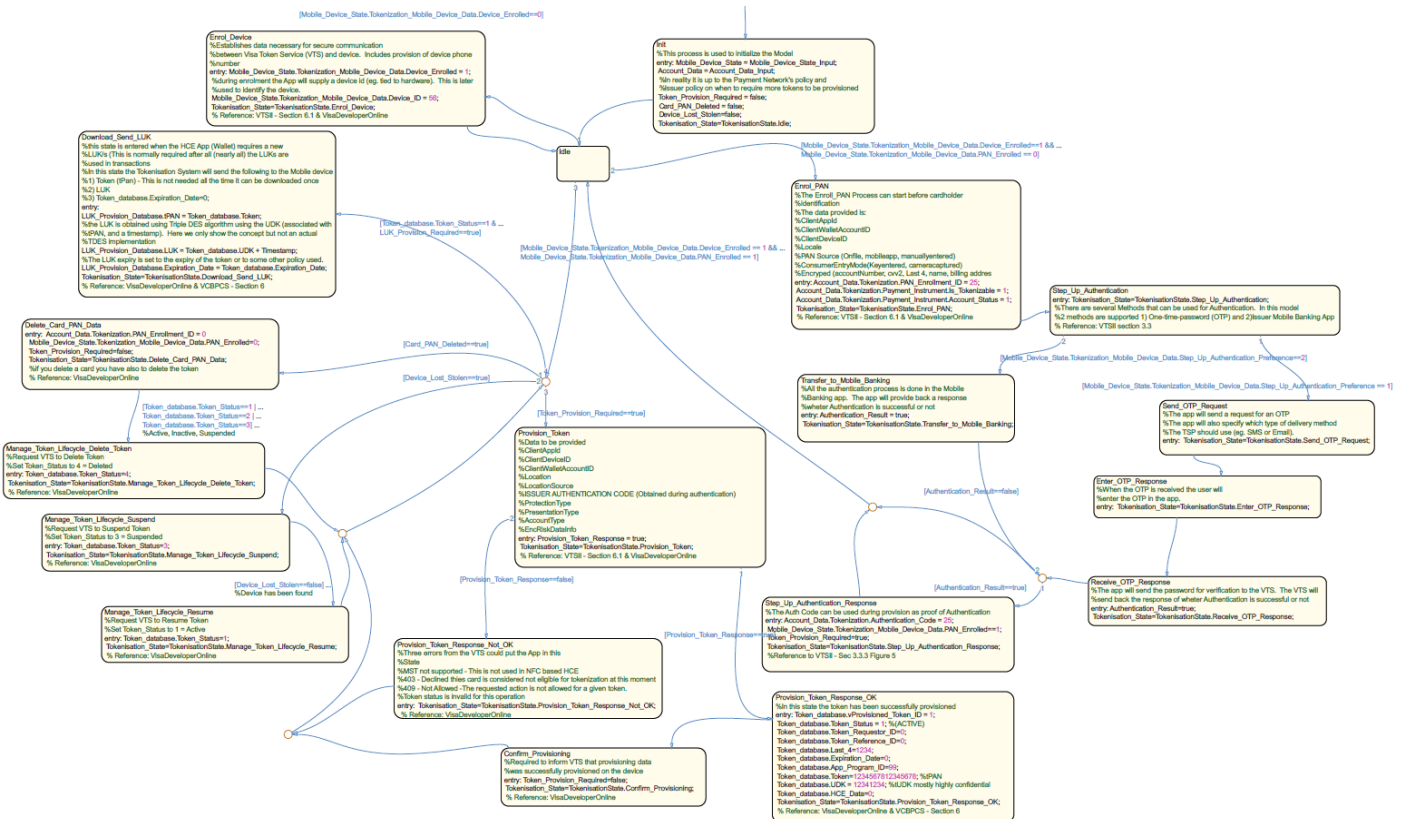
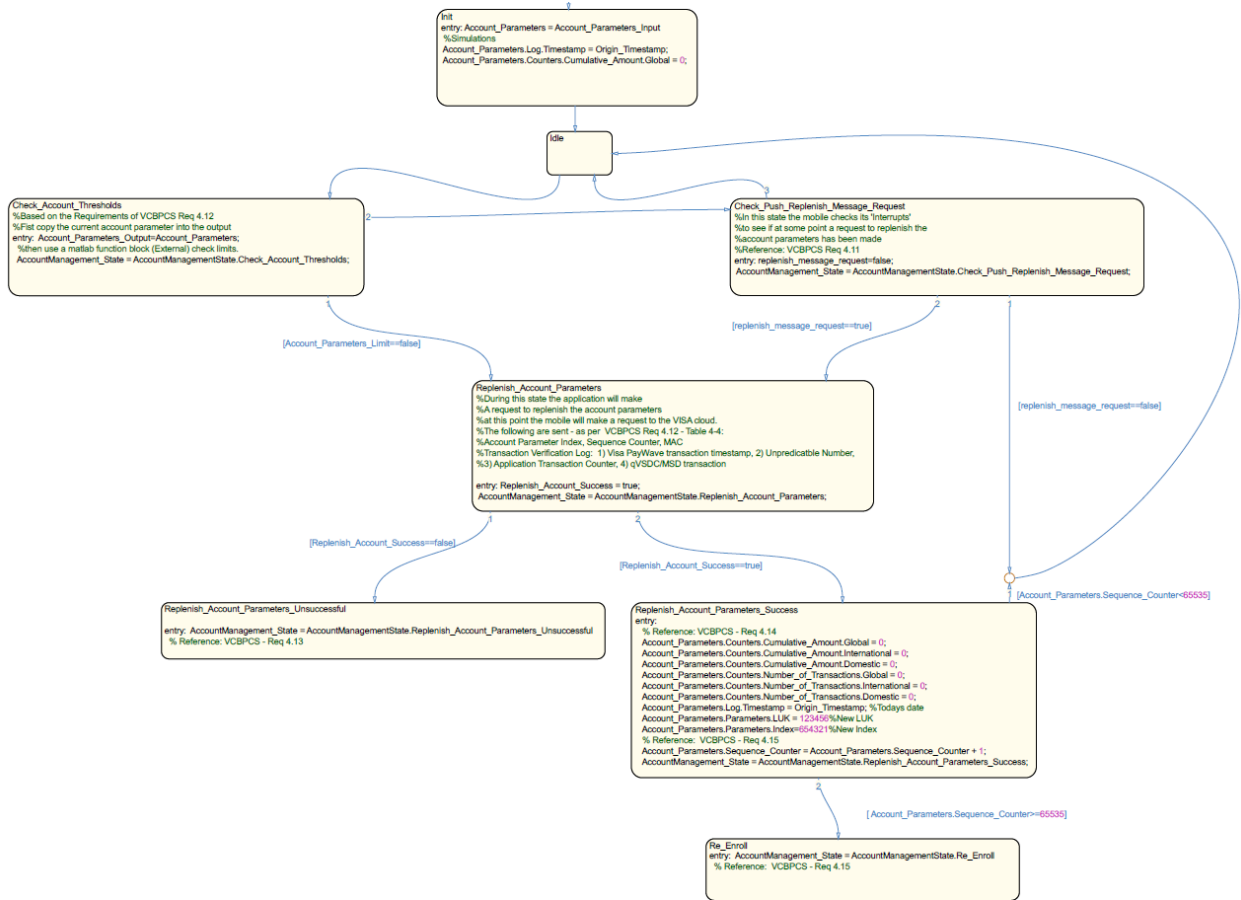


Figure 28 – Account Management Process



## 5 Security Analysis of an HCE Payment Transaction

In this chapter, an analysis of the models developed in Chapter 4 was performed. The purpose of the analysis entails studying the model states thoroughly and trying to determine risks associated with the following areas: customer verification and authentication methods, tokenization and the impact in operating cryptographic functions and storing cryptographic keys, required during an HCE payment process, on a mobile device. Following the identification of the risks, possible countermeasures, where possible, were identified. A summary of the high level risk assessment is provided in Section 5.5.

### 5.1 Security analysis of customer verification and authentication techniques when employed in HCE payments

In Section 3.2.3 several weaknesses in cardholder verification methods have been identified. This section provides an analysis of possible risks and weaknesses in the implementation of these methods used during HCE payment transactions and account management procedures. While traditional CVM methods are still possible with HCE this analysis was focused on CDCVM whereby the mobile device is used to provide cardholder verification independent of the POS for high value transactions.

The acceptance of CDCVM goes hand in hand with its ability to provide the same fraud prevention provided by traditional methods like PIN authentication. It has been calculated that \$0.027 per transaction are lost to fraud in signature based transactions while only \$0.005 per transaction are lost to fraud in PIN based transactions [156]. This is why PIN authentication gained significant acceptance and hence it is important, for CDCVM to provide, at least, the same fraud prevention capabilities of PIN based authentication to gain acceptance by all stakeholders in the industry.

#### 5.1.1 Authentication vs Verification

In security analysis, verification and authentication are marginally different. Verification is obtained by verifying something the cardholder has, for example a national identity card or a finger print. Authentication takes verification to the next level by challenging the cardholder in providing information that the cardholder only should know, example a pin or password. To an extent this is the reason why PIN (i.e. authentication) provides better fraud prevention than signature (i.e. verification).

CDCVM allows different methods to be used. The list below identifies and classifies some of these methods under authentication or verification:

- Fingerprint – the fingerprint is something the customer has hence it falls under verification.
- Password/Pin – something the customer knows hence it fall under authentication.
- Selfie Biometric – something the customer has hence it falls under verification.

It is the author's opinion that this could be 'a step backward' when dealing with fraud since some of the methods are verification rather than authentication. In Section 3.2.3, it has already been outlined that several of the methods above have been shown to be weak when compared to a PIN or Password. Furthermore, it is also important to note that a password/PIN can be changed in case it is breached but a fingerprint or selfie is something the cardholder cannot change.

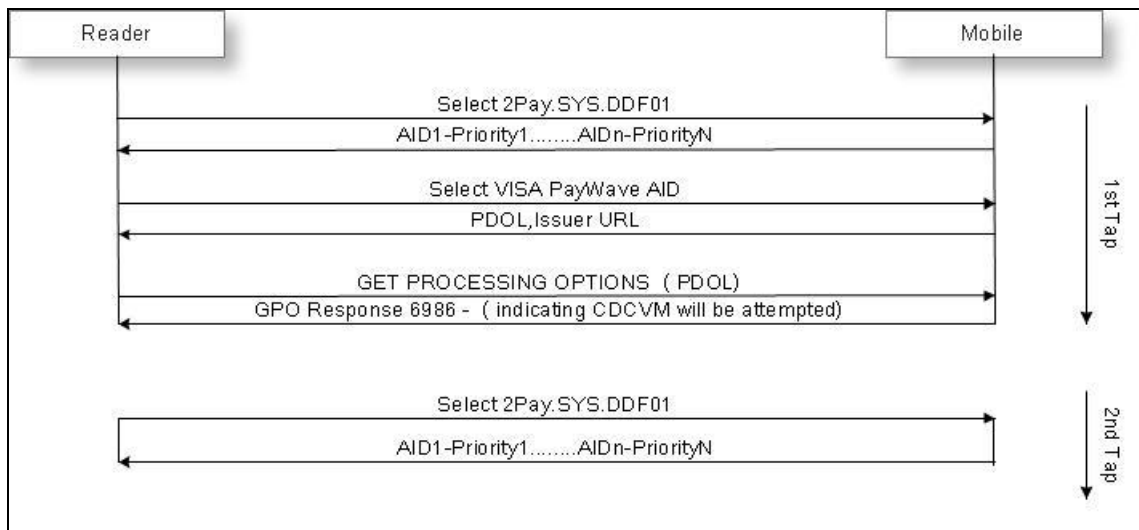
#### 5.1.2 Verification of the POS during a two tap transaction

To utilize CDCVM, a two tap approach can be used. The cardholder taps the mobile on the POS at which point the POS and the mobile device decide that CDCVM will be used and the mobile will ask the cardholder to go through the CDCVM process. After successful verification, the customer will re-tap the mobile device to proceed with the payment. The analysis below identifies a weakness which can be exploited by a fraudster to aid in bypassing the authentication mechanism.

**1st Tap**

Figure 29 below shows the communication between the mobile device and the reader up to when the reader sends the Get Processing Options command to the mobile device. The following is a step by step process of the communication that occurs in the 1<sup>st</sup> Tap:

1. The mobile device receives the AID from the reader and starts the HCE Wallet App.
2. The mobile device request the PDOL content it requires from the reader.
3. The reader passes the PDOL data through the Get Processing Options.
4. The mobile device will determine, based on the PDOL, that the transaction will be authenticated using CDCVM. The mobile then checks to see if it has already obtained a CDCVM from the user:
  - a. If it has obtained a CDCVM, the transaction will proceed as usual with the mobile generating the cryptogram and eventually sending the cryptogram along with other data through a series of read record commands.
  - b. If it does not yet have a CDCVM approval, it will send a 6986 status response to the get processing options command [150], indicating to the reader that the mobile device is not in 'ready to pay' state. The mobile device will indicate to the user to make a CDCVM and afterwards ask the user to re tap the mobile device on the POS.



**Figure 29 - Payment transaction flow using the Two Tap approach**

It is important to note that up to the GPO command the mobile device would have only received the PDOL data from the POS. The PDOL content is shown below in Figure 30. The mobile device is not required to request all the 'Tags' but for the purpose of this analysis we assume all the data is requested.

Tag	Length	Data Element Name
9F66	4 bytes	Terminal Transaction Qualifiers (TTQ)
9F02	6 bytes	Amount, Authorised
9F03	6 bytes	Amount, Other
9F1A	2 bytes	Terminal Country Code
95	5 bytes	Terminal verification Results (TVR)
5F2A	2 bytes	Transaction Currency Code
9A	3 bytes	Transaction Date
9C	1 byte	Transaction Type
9F37	4 bytes	Unpredictable Number

**Figure 30 - PDOL Content [130]**

## **2<sup>nd</sup> Tap**

When the mobile is presented the second time on the POS (after a successful cardholder verification) the transaction will follow a standard flow [150]. The mobile device can verify that the PDOL content is the same as the original/initial tap, received in Figure 29 above, before proceeding with the transaction. All the data, including the amount authorized, should not change in the second tap but it is important to note that the unpredictable number (i.e. UN) will change as otherwise this would make the UN predictable and prone to other forms of attacks such as relay attacks.

### **Possible Risk:**

Consider this scenario. Let us assume that a fraudster manages to get hold of a stolen mobile device. Consider that the CDCVM in the mobile device uses a fingerprint authentication method and the fraudster has a way of replicating the fingerprint of the legitimate owner [126]. It would not be ideal for the fraudster to use this method in a shop as the shop attendant would easily spot the attempted fraud. Therefore the fraudster needs to 'execute' the authentication method prior to visiting the shop.

1. The fraudster visits the target shop and uses another rogue phone to buy a low value item. The transaction is approved but the rogue phone records (i.e. sniffed) all the NFC communication. Hence the fraudster knows the PDOL data that a particular POS in the shop will provide apart from the UN, the amount authorized and the date.
2. The fraudster creates a reader device using a laptop and NFC transceiver that is able to mimic the reader at the shop. The reader will provide a copy of the PDOL recorded previously with a particular amount and date. The date is adjusted to the date the fraudster will visit the shop and the amount is set to a high value item in that shop.
3. The fraudster will put the stolen device on the rogue reader at which point the stolen mobile device will ask for a CDCVM. The fraudster can now utilize a known method to circumvent the CDCVM method at his own pace and in a private setting (e.g. in a car). For example, a fingerprint could be lifted off a personal item, replicated and then carefully re-introduced.
4. After successful CDCVM the fraudster will immediately visit the shop and make the transaction on the targeted POS for exactly the high value amount provided in the initial PDOL. The mobile would be in a state where the CDCVM is provided and approved, the amount and values of the PDOL match and hence the mobile device will assume this is the 2<sup>nd</sup> Tap.

Thus this shows us that the mobile device is unable to identify that the transaction had started on a rogue POS and completed on another legitimate POS. Since the PDOL and the amount matches, the mobile will proceed with the transaction. Obviously this attack relies on the fact that the CDCVM is circumvented in some way but the fraudster has all the privacy and time available to apply the technique while not in the shop. For example, the fraudster can use a photo to provide CDCVM for a selfie biometric in a car while, if the fraudster was in a shop this would immediately be spotted as fraud. Furthermore, the fraudster can try the CDCVM more than once until it is successful on the rogue reader if the mobile device does not have a retry limit.

### **Possible Countermeasure:**

For security purposes, to solve this issue, another unpredictable number (i.e. not related to the UN that is used to generate the cryptogram) should be used and provided by the POS to identify the transaction. This way the fraudster would need to know this number prior to initiating the transaction. Using the UN used for the generation of the cryptogram should be avoided as this would introduce a possibility for mounting relay attacks since the attacker would have a large time window where the UN could be relayed to obtain a signature/cryptogram. Furthermore a time limit should be introduced such that a second tap will occur within that time limit (e.g. 1 minute). This means the fraudster would have to provide a CDCVM within 1 minute of the initial tap.

### 5.1.3 Issues of Pre-Authentication

With CDCVM, the wallet developer can provide the option to the cardholder to provide a CDCVM authentication before the mobile is tapped on the reader [150]. This means the mobile device would be authenticated and be in a state to pay (up to a certain limit e.g. €100) prior to the amount of the transaction being known and the transaction would occur in a one tap. If the amount of the transaction is higher than the limit then a two tap approach will be used.

**Possible Risk:**

Given that a mobile device has been pre-authenticated it is subject to a relay attack even on high value transactions. Since no feedback or authorization is required from the cardholder during the payment transaction phase, a relay attack can be mounted without the cardholder providing authentication, as it would have already been provided. The technical difficulties to achieve a relay attack still apply (e.g. timing) but in a card based transaction a relay attack was only possible for transactions that did not require cardholder authentication. The only requirement for the fraudster is to have access to a mobile phone in a 'ready to pay' state. This is essentially an open cheque up to a specific amount.

**Possible Countermeasure:**

To mitigate this risk, pre-authentication should not be allowed at all, this way, during a transaction, the user is required to provide authentication, hence eliminating the possibility of a relay attack.

**Possible Risk:**

With a pre-authentication the 'normal' flow is altered and in this case the cardholder, can authorize without even knowing the amount. Therefore the cardholder is expected to look at the reader prior to tapping the mobile device to be sure that the amount one is paying for is the correct amount. There is a risk that the user overlooks this step or simply ignores the reader and simply taps the mobile on the reader without checking the correct transaction amount.

**Possible Countermeasure:**

To mitigate this risk, wallet developers should need to implement a 2<sup>nd</sup> tap approach even in a pre-authentication. The user would first provide authentication (CDCVM), tap the mobile devices (i.e. 1<sup>st</sup> Tap) on the POS and then present the amount to the user on the screen. The user makes sure the amount is correct and re-taps (i.e. 2<sup>nd</sup> tap) the device on the POS. This would ensure that the cardholder is aware of the amount one is about to pay for.

**Possible Risk:**

Pre-authentication can also cause issues for non-repudiation. If a cardholder provides pre-authentication then the cardholder cannot be held accountable for the purchase, which happens afterwards. The cardholder can claim s/he was not aware of the transaction, because he would have simply provided authentication beforehand.

### 5.1.4 Sharing the verification database between Mobile Device and Payment Application

In many cases, the mobile device may be using the same verification method to authenticate the user (e.g. Apple's Touch ID [118]) for the purpose of unlocking the phone. Since mobile phones can be used by more than one person (e.g. mother and child) it is common for the user to enroll another person into the fingerprint database (or database of a particular method).

**Possible Risk:**

If the payment application shares the same database than the other person other than the owner can also authenticate as a cardholder and make a transaction. This can create an issue of unauthorized payments (i.e. accountability) and also an issue of non-repudiation.

**Possible Countermeasure:**

The Issuer should explicitly state in its terms and conditions that the cardholder using a particular device should not enroll other users into the database. Furthermore the cardholder is held accountable for any transaction performed on the mobile device.

### 5.1.5 Shift of control from Issuer to Wallet App Owner

Prior to the introduction of mobile devices, the Issuer had full control over the card as the cards were personalized and configured with the Issuer's specific rules despite performed by the 'card perso bureaus'. Thereby the Issuer had control over which authentication method as well as the cardholder verification method priorities to be used with a certain payment card. Under the HCE model, the control over which authentication method is used is passed over to the owner of the wallet application<sup>16</sup>. In certain cases, the app is developed and maintained by the Issuer but in other circumstances the app could be developed and maintained by other entities (e.g. Google in case of Android Pay).

As shown in the model in Section 4.3.5, the reader indicates to the mobile device which method the reader supports through the PDOL – TTQ – Byte 1. The mobile device then provides its chosen method through the CTQ and CVR bytes.

**Possible Risk:**

This could lead to a situation where app developers, downgrade the authentication method used for simplicity (e.g. mobile pattern over PIN) and Issuers are unable to 'enforce and prioritize' the use of better methods. Note that during a transaction the reader is unaware of the priorities of the Issuer hence it cannot enforce certain method over another.

**Possible Countermeasure:**

The Issuer still has the final say and can choose to decline a transaction if a certain authentication method is used. However, this may impact the financial institution business revenues. If possible, an agreement should be reached between the wallet provider and Issuer to include the responsibilities, liabilities and prioritized preferred methods.

It is also recommended that payment schemes such as VISA should be able enforce a prioritized system of CDCVM methods based on the risk appetite of the Issuer. The app developer, subject to supporting certain methods, should be required to prioritize one method over the other to minimize risk for the Issuer.

## 5.2 Security analysis of the tokenization process used during an HCE payment transaction

The analysis of the tokenization process in this project has been split into different sections. The first section deals with Token Generation specifically the generation and provision of LUKs on the mobile device. The advantages and disadvantages of storing the TR in the cloud versus the TEE are analysed. Next the enrolment process is analyzed, specifically how the Issuer authenticates the mobile device. In the next section, the risk analysis processes of the TSP was studied. During risk analysis, the TSP calculates a value known as the risk assurance level and in this section possible improvements have been provided to the risk analysis calculation.

---

<sup>16</sup> An Issuer can choose to decline a transaction with a certain method but if the issuer supports two methods A and B the issuer cannot impose A having a higher priority over B.

Tokens used for contactless payment transactions cannot be used elsewhere (e.g. in online payments). Hence the next section dealt with further improving the token to device mapping to further ensure that tokens assigned to a device are used within that particular device to further limit the portability of the token. The subsequent section was dedicated to supply and demand synchronization between the mobile device and the token requestor and finally in the last section issues pertaining to the use of tokenization when making payments in MSD mode were identified and possible countermeasures were provided.

### **5.2.1 Token Generation**

As shown in Figure 17, the TSP generates the tPAN and the tUDK. These are transferred to the token requestor which in turn generates the LUK (Limited Use Keys) for the tPAN. This is why the system is labelled as a 'static token' (i.e. tPAN) with dynamic cryptographic keys (i.e. LUKs).

The Token Requestor is a separate entity from the HCE wallet process and can either be located in an environment such as the cloud or in the TEE of the mobile device. From a model point of view there is no difference whether the TR is stored in the TEE or the cloud.

#### **5.2.1.1 Token Requestor – Cloud**

When the TR is in the cloud, a secure link between the TR and the mobile Wallet App is required. This is particularly important during enrolment. VISA does not specify how the PAN is entered but methods include using the camera of the phone by taking a photo of their card or type their card details on the application or direct enrolment through their mBanking [157]. Using the camera and the touchscreen to capture the card details are potentially risky as the camera is directly accessible at an application layer. Other 'security related' sensors, such as the fingerprint sensor, are not available at the application layer but are only accessible by the kernel [158]. Given that VISA does not specify the method and how such method is implemented, TRs are inclined to shift as much processing to the cloud as possible. This makes it easier for the TR to upgrade and maintain the system. For example, if the PAN is enrolled from a camera capture then it could be possible that the image is sent over the secure link and processed at the TR's cloud versus processing the image on the mobile device. This further outlines the importance of the secure link between the mobile device and the TR.

#### **5.2.1.2 Token Requestor – TEE**

When the TR is situated in the TEE then a secure connection is required between the TR (i.e. mobile device TEE) and the TSP (i.e. VISA or any entity acting as the TSP). This connection is far more 'risky' than the connection between the TR and the mobile device in the Cloud scenario but in this case the TSP declares the type of communication to be used hence this is tightly 'defined and controlled' by the TSP. The reason this connection is risky is because the TSP will pass the tUDK, which if compromised can be used to generate LUKs which would compromise the whole tokenisation process.

The connection between the TEE and the HCE Wallet app can be considered as secure until the mobile device is not compromised (e.g. device is rooted). The information contained in the TEE is still secure but the communication between the TEE and the Wallet app would be vulnerable. Hence, communication between the TEE and the Wallet App should be secured to ensure that no 'eavesdropping or malware' process could steal the LUKs transferred between the TEE and the Wallet app. To further ensure security some implementations, such as Android Pay, are designed to not allow payments if the device is rooted [159].



## 5.2.2 Device authentication during enrolment

When a PAN is enrolled within a wallet, the Issuer or even a TR [157] could set provisioning rules to verify the cardholder. VISA calls this process 'Step-Up Authentication' [157] and provides several examples of how this can be done including using an OTP and using the mobile banking app of the Issuer. This process is outlined in the model in the Token Requestor Process and shown in Figure 27. During this process, the cardholder verification 'duty' is passed from the Issuer to the mobile Wallet app. Once a PAN is enrolled with the TSP and verified, the wallet will verify the cardholder during the payment using the 'identity data' provided during enrolment with the TSP.

### Possible Risk:

The issue with certain methods is that the mobile wallet is not an integral part of the authentication. Consider a situation where a fraudster has access to a credit card and the mobile device of the cardholder. The fraudster can use his/her phone to enrol the card. His wallet will instruct the TSP to send an OTP for enrolment and thereby the TSP will inform the Issuer to provide the OTP to the cardholder based on his/her registered mobile device phone number. The SMS is received on the cardholder's mobile device and the fraudster simply copies the OTP on his device. The PAN is now enrolled and from then onwards s/he is in a position to make payments through tokens that are mapped on the PAN of the victim. This risk becomes a major problem if CDCVM is provided and the cardholder enters his/her fingerprint during enrolment thus the fraudster can simply enrol his/her fingerprint during enrolment. The main issue of this risk is the fact that the issuer is using the mobile device number of the cardholders as a means of verification but in the end the issuer is not verifying that the card is enrolled on a device with that mobile phone number.

### Possible Countermeasure:

To mitigate this risk, the mobile device should become an integral part of the authentication, specifically the sim card. If the Issuer is verifying the cardholder using the cardholder's mobile phone number, which is globally unique, then the mobile device using that sim card should be the only device that is allowed to enrol that particular PAN and subsequently receive the tokens (LUKs) for that PAN. The following process identifies a possible improvement over the current step-up authentication:

1. The cardholder starts the enrolment on the Wallet HCE App.
2. The App, reads the mobile phone's SIM phone number and sends the number to the TSP. Subsequently the TSP sends this number to the Issuer for verification.
3. The Issuer sends back an OTP but the OTP is a key, which will only work on a device having a sim card with the cardholder's phone number recorded also with the Issuer.
4. This OTP key 'checking' happens once with every transaction and the software checks that the phone has network connectivity to ensure that the sim card has not been cloned.

This ensures that the scenario identified does not happen, and secondly, the Wallet verifies that the phone being used for the transaction belongs to the cardholder (i.e. carries a SIM with the cardholder's phone number).

If the mobile banking app is used as an authentication mechanism, then this is considered as a safe method, being that the mobile banking app communicates directly with the HCE Wallet App, as long as the Issuer provides a secure way to ensure the mobile banking app is only used and installed on the cardholder's device.

## 5.2.3 LUK location risk analysis

Once the LUK is 'replenished' on the mobile device it is either stored within the HCE wallet App's memory or in the TEE. Storage in the TEE is more secure than the wallet application but not all devices provide a TEE or certain devices might not be trusted by the Issuer or simply the developer of the wallet app decides not to use this storage mechanism.

**Possible Risk:**

One issue that was noted during the development of the model is the fact that the TR is requested to provide information about how the token (i.e. tPAN and tUDK) will be stored [153] (e.g. in the cloud or TEE) but there is no requirement for the TR to provide information on where the LUKs for that token will be stored eventually. Hence a TR can specify the cloud as storage during the provisioning of the token but eventually the LUKs are stored in the wallet application. This means that, during a transaction, the location of the LUK cannot be factored in during risk analysis by the card network or TSP.

**Possible Countermeasure:**

During enrolment of a PAN on the mobile device, the mobile wallet app provides information to the TR of where it intends to store the LUKs (i.e. App or TEE). The TR will then pass on this information to the TSP and hence the TSP would be in a position to make certain risk analysis based on the storage mechanism used when receiving a payment from that PAN. This is in line with EMV Payment Tokenisation Specification's Assigned Assurance Level value<sup>17</sup>. The EMV specification refer to the storage of the Token at the TR since it is the TR that communicates with the TSP. This should be updated to also include consideration about the storage of the LUK on the mobile device.

## 5.2.4 Token to Device Mapping

During a transaction, the LUK is used as the key to generate the transaction cryptogram which is passed over for authorization. The TSP will 'de tokenize' the cryptogram and eventually pass it over to the Issuer for authorisation. This process is no different from a regular contactless transaction but it does not prevent a fraudster from stealing the token (tPAN and LUK) and using it elsewhere for low value payments that do not require online authorization.

**Possible Risk:**

The issue with this process comes from the fact that EMV systems have been designed for card payments and HCE has been 'fitted' into the infrastructure. Tokens are in no way 'tied' to a device, during a transaction. Even if a unique deviceID is passed to the TR during enrolment, it is not passed to the POS during a transaction and furthermore the TSP does not have information that ties a token with this deviceID. This information is held by the TR and hence the TSP can only provide assurance of a valid token but cannot provide assurance that the token originated from the cardholder's device which was originally the receiver of the LUK from the TR. If a fraudster manages to access the storage within the wallet app and 'steal' the LUKs, then one is in a position to make a payment from another device and the TSP is unable to identify such fraud.

**Possible Countermeasure:**

To mitigate this risk the TSP should have access to a unique deviceID from the TR. This will allow the TSP to verify that the token came from the cardholder's device. The deviceID could be made up of an unpredictable number that is passed as part of the cryptogram<sup>18</sup>/dynamic CVV<sup>19</sup> data. This way a fraudster would be unable to identify it. Hence during a payment transaction, the TSP would pass the deviceID stored in its database to the Issuer and the Issuer can verify that the deviceID used to generate the cryptogram/dynamic CVV is the same as that provided by the TSP. This ensures that the token is mapped to the device and in case the token was stolen or moved, it would be unusable, subject that the deviceID, a unique number, is not easily guessed or stolen. Currently the EMVCo specification allow for tokens to be

---

<sup>17</sup> The Assigned assurance level consists of a baselines that is an integer value between 0 and 99 that representing the confidence level of the payment token to PAN to the cardholder and the cardholder's account to determine how trustworthy the payment token is [171]

<sup>18</sup> Applies for qVSDC

<sup>19</sup> Applies for MSD

mapped to a particular method of payment (e.g. a specific token range can only be used for contactless transactions) but the token, once provisioned on a device is not mapped to that particular device.

### 5.2.5 LUK Supply and Demand Synchronisation

Based on the model developed, the mobile device can request any amount of LUKs from the TR. The VISA tokenisation specifications and requirements there is no requirement from the TR to provide information to the TSP on the amount of LUKs issued and the TSP is not required to provide information to the TR about the LUKs that have been used up [152]. The impact is that the TR is not aware of how much of the LUKs issued have been used.

#### **Possible Risk:**

This could lead to a large amount of LUKs being issued by the TR stored on a mobile device increasing the risk and the cost of such risk should these LUKs be leaked or compromised. After all the concept of a LUK is only valid if the actual keys are limited. This risk can arise from different situations:

1. Given that mobile wallet app developers want to provide the best experience to the cardholder, the developers would want to have a large pool of LUKs to ensure that the user has a LUK available even during long periods without internet activity.
2. If, in the future, a fraudster is able to overcome the cryptographic measures used within EMV, then the fraudster would be in a position to obtain a large pool of tokens and keys to mount the attack.

#### **Possible Countermeasure**

To mitigate this risk, a synchronisation mechanism is required between the Wallet app, the TR and the TSP. The TR would replenish a certain amount of LUKs to the mobile device and when the TSP informs the TR that a certain amount of LUKs have been used the TR will allow the mobile device to request more keys. This synchronisation mechanism allows the TR to have assurance that the device stores a limited amount of LUKs. The Visa Cloud-Based Payments platform, which is used in case no tokenisation is used, allow the mobile device to upload the Transaction Verification Log. Such log is updated after each transaction. This log and the ATC can be used to ensure that certain amount of LUKs can be used. Hence this strategy can be implemented with the TR, subject that the TR can verify that the ATC is valid through the TSP.

### 5.2.6 Issues of Tokenization in MSD Mode

While VISA is trying to phase out MSD mode it is still, at the time of writing, supported and used by many Issuers especially in areas where many terminals do not yet support qVSDC mode. In MSD mode, tokenisation does not solve the replay attack due to way the dynamic CVV is generated [130]. This issue has been outlined by S. Mendoza [129].

In MSD mode, the mobile device will prepare all the data required and send the data to the terminal during the READ RECORD series of commands. All the data is static data except for the dynamic CVV in case of MSD verification value. The algorithm to generate this verification value is detailed in [130]. The algorithm is based on an encryption, using Triple DES and the LUKs as encryption key. The data encrypted is static data and the Application Transaction Counter (i.e. ATC).

#### **Possible Risk:**

This means that a fraudster, equipped with a mobile device acting as a reader, can simulate a POS terminal and thereby collect both the tPAN and the dynamic CVV for a specific ATC value created with a valid LUK for the tPAN. The fraudster can then visit a shop, make a purchase and pay using the values collected with his mobile device. Note that the amount of the purchase is not part of the dynamic CVV hence the fraudster can make any purchase amount as long as it is under the stipulated limits for MSD mode. The transaction will be made online, since all MSD mode transactions have to be made online [160] but all the information will be valid and hence it will not be detected as fraud subject that the fraudster is able to provide a valid

signature (cardholder verification). The ATC will not be good for the next transaction and hence the stolen dynamic CVV will only be good for one transaction.

The main issues that make such an attack possible are:

- The mobile device cannot authenticate the POS reader to ensure that it is truly legitimate.
- The dynamic CVV algorithm is not a 'challenge-response' algorithm. If the algorithm were to include the unpredictable number (UN) provided by the terminal then the dynamic CVV produced will be tied to that UN.

**Possible Countermeasure:**

To solve this issue, the dynamic CVV algorithm should include an unpredictable number provided by the terminal. Such a countermeasure is costly for card schemes as it would entail that all current contactless cards that support MSD would have to be updated or changed to support the new algorithm.

Another countermeasure could be, a challenge response mechanism can be used to enable the mobile phone to verify that the POS is legitimate. The mobile provides a challenge to the terminal, the terminal signs the challenge with its private key and sends the signature and the certificate of the key pair. The mobile verifies the certificate with the CA and then verifies that the POS is in possession of the private key by using the provided public key of the terminal. This way the mobile device is assured that the terminal is legitimate. If a rogue POS or reader is used the mobile device would not provide the dynamic CVV and thus decline to transact.

It is also recommended that MSD mode is phased out and qVSDC mode is used instead.<sup>20</sup>

### **5.3 Analysis of the impact in operating cryptographic functions and storing cryptographic keys, required during an HCE payment process, on a mobile device**

Unlike physical cards, mobile devices are subject to attacks that can lead to leaks of highly sensitive data such as cryptographic keys. HCE Wallet developers have to account for such risk and use adequate procedures to ensure that sensitive data is stored safely and used in a manner where it does not make it easy for an attacker to access them. In this section, first an analysis of the importance of the TEE for storing and operating cryptographic functions is provided. The next section focuses on the implication of storing the RSA Key used during ODA and eventually in the last section an analysis of the issues in using LUKs for authentication with online services is performed.

#### **5.3.1 Preventing access to the LUK through the TEE**

In the current system, as shown in Figure 29, the cryptogram generation is done after the GPO command is received from the POS reader. At this point the mobile device has all the information required to build the cryptogram. Currently, the cryptogram is generated in the actual HCE payment wallet (i.e. build cryptogram state in Figure 26) based on the specification set out in VMCPs [130].

**Possible Risk:**

This means that the LUK is used in the HCE Wallet App to generate the cryptogram and hence if the mobile phone is compromised then the LUKs stored in the App's key store would become accessible.

**Possible Countermeasure:**

To mitigate the risk, a requirement should be set such that, the cryptogram generation mechanism should be shifted from the Wallet App into the TEE environment. The LUKs are stored in the TEE and the HCE

---

<sup>20</sup> At the time of writing this report, VISA had started the phase out of MSD, but MSD was still being used.

wallet application would provide the data obtained about the transaction from the POS to the TEE. The TEE would generate the cryptogram and return back the generated cryptogram. This means that in case of a compromised phone, the fraudster would be unable to get access to the LUK even if the phone is rooted but it would only allow the fraudster to make use of such cryptogram generation mechanism. Hence the fraudster would have to transfer the transaction data, most importantly the unpredictable number provided by the terminal, onto the device, generate the cryptogram and transfer back the data towards his/her device creating a relay attack scenario.

This raises the debate whether the TR should reside in the Cloud or in the TEE. Certainly in this case, storing the TR in the TEE would be a better option as essentially the LUKs do not need to ever leave the TEE environment while if the TR is in the cloud, the mobile device would need to transfer the LUKs from the cloud and store them in the TEE. Notwithstanding, the mobile device would still need to download and store the UDK from the TSP in case the TR is situated in the TEE but this would only occur once during enrolment and subsequently when tPAN expires whilst if the TR is in the cloud it would have to transfer LUKs every time they are depleted.

### **5.3.2 Offline Data Authentication (ODA) RSA Key Storage**

Another factor worth considering in this analysis is the Offline Data Authentication signature generated during a transaction which requires ODA support. ODA is common in fast throughput scenarios such as at transit gates and theatres. VISA in VMCPs defines it by stating: “ODA requires that the Mobile Application contain an RSA key to generate the fDDA signature – The certificate and signature information is then used by the reader (POS Terminal) to verify that the Mobile Application is genuine” [130].

#### **Possible Risk:**

Since the aim of ODA is to allow the reader to authenticate the mobile application, the RSA key cannot be ‘tokenized’. Hence the RSA key will not change during the lifetime of the application and it should be protected with the highest protection possible. If a mobile device does not have a TEE or the wallet does not support the TEE, then the RSA key has to be stored in the application’s memory. This is a high risk as it would be accessible in a compromised phone. Note that, although payment accepted with ODA is limited to low value transaction, a large scale breach involving multiple mobile phones would result in huge losses.

#### **Possible Countermeasure:**

To minimize this risk, Card Schemes should require that the RSA key is stored in the TEE thereby reducing the risk of the key being known in case a mobile phone is breached. Currently the specifications [149] (i.e. requirement 4.4) declare the storage and acquisition of the RSA Key as ‘out of scope’ and allow the mobile wallet developer to choose the storage mechanisms as long as it is deemed secure.

If TEE storage is used then the SHA-1 value can be generated in the Wallet Application but its value is then passed to the TEE to generate the signature. Note that Android’s hardware backed keystore [114] already provides support for RSA signature generation.

#### **5.3.2.1 Further Risk Analysis in ODA**

Visa extended the Issuer Application Data with the following criteria: information about the cryptogram, the type of CVM used for authentication, the Digital Wallet Provider ID and the consumer’s device state. The consumer device state (CVR Byte 5) is used to show the following information in VMCPs [130]:

- The mobile device is in debug mode.
- The mobile device is rooted.
- Whether the mobile application is hooked (e.g. using Substrate framework).
- If any code modification occurred.
- Whether the mobile phone has data connectivity.

- Whether the phone is genuine or an emulator.

**Possible Risk:**

A mobile device that has been rooted or hooked has a high probability of malware installed on the device. Currently the IAD data is passed over to the Issuer but the terminal does not run any risk analysis on this information during an offline transaction.

**Possible Countermeasure:**

The IAD information can also be added to the ODA signature and hence the terminal would then be in a position to run risk analysis on whether it should approve or reject a particular offline transaction. Payments made from a device that is rooted or hooked should not be accepted.

### **5.3.3 Issues of using LUKs for authentication to the TR and the VISA Cloud Platform**

With or without tokenization the mobile device still requires some form of interaction with a cloud platform to download the Limited Use Keys. In case tokenization is not used than the LUKs will be downloaded from the Visa Cloud-Based payments platform or any TSP cloud platform. If tokenization is used then the LUKs will be downloaded from the Token Requestor. In both cases, these platforms need to authenticate the mobile device in some way. This authentication is crucial as essentially it is the master key to obtain LUKs for a particular PAN or tPAN.

For interaction with the Visa cloud-based payment platform a cryptogram is generated and used as the authentication mechanism. The cryptogram is generated by applying a MAC over account parameters including the Sequence Counter and the transaction log. This MAC is then encrypted, using a Triple DES algorithm with the Limited Use Key as the encryption key [130]. This means that an attacker with access to the LUK, Sequence Counter and the transaction log can impersonate the mobile and gain access to the VISA cloud platform and thereby having the possibility to generate and download new LUKs. The same risk applies when authenticating to a TR.

**Possible Risk:**

If a fraudster has access to a LUK or any key used to authenticate the mobile to the TR then the fraudster would be in a position to authenticate to the TR and request more LUKs. Note that the term 'Limited Use' relates to the fact that the LUK will expire in time or after a certain amount of transactions. If with such a key further keys can be downloaded then its value, in terms of risk is very high and rather than a limited use key it would be considered as a 'Master Key' to obtain further keys.

**Possible Countermeasure:**

This point leads to the need for a safe and secure storage of such keys both at rest and in transit. It should be a requirement that any key used to authenticate to online platforms such as the VISA Cloud Platform or a TR should be stored in the TEE. If a TEE is not available or not supported then Whitebox Cryptography could be used to hide these keys in the wallet application's memory.

## **5.4 Other generic issues to the use of mobile phones for payments**

In addition, to the risks identified as part of the security analyses there are also other risks. These risks are not based on the model but are generic issues that arise from using the mobile phone for payment applications. The following risks are not based on any particular implementation but might be used by fraudsters as an aid to activate the risks mentioned in Section 5.

### **5.4.1 Phishing and social engineering**

An attacker through social engineering, attempts to lure a cardholder with the aim to disclose sensitive payment related data or to convince the cardholder to replace a legitimate application with a fake one.

### **5.4.2 Installation of rogue and malware applications**

The installation of malware and rogue application can assist fraudsters in mounting remote attacks. Techniques such as memory scraping and remote rooting provide the fraudster with the ability to copy and upload areas of memory in the mobile phone that can later be analysed with the intention to steal sensitive information such as PANs, tokens and LUKs.

### **5.4.3 Reverse engineering of payment application and mobile operating system**

Applications are downloaded from app stores in the form of a package. Most importantly the package contains the compiled application. Attackers can use special tools to decompile the application with the intention to understand how the application works and in which place in memory it is storing the sensitive information including hard-coded passwords, mobile pattern and fingerprint authentication template database and PINs. This is one of the reasons why a TEE should be set as a mandatory requirement for payment applications.

### **5.4.4 Denial of service attacks and data connectivity compromise**

In Section 3, several studies were presented on DDOS attacks on the NFC interface. It was shown that these can only be mounted at close range and thus the risk emanating from a DDOS attack is unlikely to occur. Notwithstanding, if the HCE payment wallet application makes use of tokenisation, particularly using a Cloud TR, a DDOS can be mounted on the TR and hence cardholders would be unable to replenish tokens and LUKs. This may also effect the cardholder ability to enrol.

### **5.4.5 Problems related to the use of biometrics**

The software capturing the selfie may struggle under certain conditions such as in poor lighting ambient, face is positioned in a tilted position, facial expressions, age factors such as wrinkles or when objects partial cover the face such as when wearing sunglasses, hats, scarves and makeup. Such factors also lead to misidentification of the cardholder and thus resulting in denied access to the mobile device.

### **5.4.6 Privacy**

The collection of biometric data could also give rise to privacy issues especially if the image collected would be used for other than the intent purpose (i.e. for authentication), for example, abusing of such data, to present marketing material associated with that particular image. In addition, the collection of millions of individuals information maybe a goldmine for attackers. Thus, proper measures such as encryption and/or anonymize the data should be applied. This also leave a room for discussion to define the responsibility of the biometric service whether it is the Issuer, the mobile device manufacturer or the user.

The industry as well as cardholders should ensure that robust standards, specifications and best practices are in place to support HCE and the mobile payment ecosystem in order to mitigate against any inherent risks. The standards and specifications shall cover security controls and measures in the areas such as communications between the payment application and the cloud, device, and application and payment credentials to protect the confidentiality, integrity and/or availability of information. A risk assessment should be performed on a continuous basis by all the mobile payment ecosystem parties to prevent any risks emanating from unintentional acts or deliberate actions.

## 5.5 Summary of Analysis and Main Findings

In HCE, particularly with the use of tokenisation and CDCVM, cardholder verification responsibility is shifted from the Issuer towards the mobile device. Most of the methods used for verification such as fingerprint and biometrics have been proven, by researchers to be weak. Notwithstanding these methods provide more convenience and ease of use for the card holder and thus issuers and card networks are still in favour of using these methods, striking a balance between security and convenience. As shown during this analysis, it is important to minimize loopholes to reduce the possibility of fraud. Using the current methods it has been shown that a fraudster can attempt fingerprint spoofing prior to visiting a shop making it impossible for the shop attendant to spot the fraud. Furthermore, the methods have to support the typical payment flow whereby a cardholder provides authentication after the payment is known rather than at the beginning of a transaction ensuring that the customer is aware of the amount s/he will be authorizing.

Some of the CDCVM methods share the identification templates with the mobile device (e.g. fingerprint template database is used to unlock the device and to make a payment). Some mobile phone users might enrol the fingerprint of more than one user, example enrolling a child to be able to unlock and use the mother's phone. This raises issues for non-repudiation and accountability of a transaction. Issuers should ensure that the payment application accepts the fingerprint of the cardholder only rather than allowing all the users enrolled in the mobile phone's fingerprint database to authorize during a transaction.

The use of HCE shifts control over the ID&V method used from the Issuer towards the mobile wallet developer. The Issuer has control over which methods are used but may not prioritize certain methods. Furthermore, if CDCVM is used then the wallet developer has full control over the method used. This might bring certain risk if the wallet developer decides to downgrade the CDCVM method for convenience purposes, to satisfy the cardholder thereby reducing the security of the application. Such risk is more for wallet app such as Android Pay rather than Issuer's proprietary ones.

The second section of the analysis dealt with tokenization. The shift from static a static PAN and key towards tokenization limits the effects of a stolen PAN but given the mobile device is a connected device it is prone to many more network related attacks then what a simple physical payment card could be subjected too. Most importantly is the fact that the mobile device could be compromised through malware, or similar malicious software. Such attacks can be replicated on a very large scale especially through the use of social media and other social sharing networks. Hence, it is of high importance that the tokenized values are safely stored in a manner where a compromised phone would not leak these values, even if these values have a limited use. Tokenization limits the risk for the cardholder, considering that only one payment is made with a stolen token, but for the Issuer the risk would still be high if potentially a large number of tokens and keys are stolen.

When using Tokenization, the wallet application needs a connection to a Token Requestor service to be able to replenish tokens when these are depleted or expired. One key risk identified is the fact that the limited use keys are being used as a means of authentication of the mobile device to the Token Requestor. This gives rise to a discussion on how 'Limited Use' are these keys since they can be used to obtain further keys. Thus it is important to store these keys, safe, in a way that ensures the keys are not exposed even if the phone is compromised, rooted or hooked.

Unlike iOS (Apple), Android OS is developed by Google but used by many vendors of mobile devices. The vendors decide the type of hardware used, including the processor. This means that a wallet application can be installed on different devices with varying levels of security hardware (i.e. with or without a TEE). Furthermore, since the OS itself is freely available, several tools can be used to root the device, even by non-technical users rendering the device less safe and more prone for malware installations. The EMVCo provides a token assurance level value but the criteria on which this value is calculated does not factor the



hardware/software properties of the mobile device. For better risk analysis, the hardware and software properties such as whether the mobile is rooted, availability of the TEE, etc., should be sent to the TR and become part of the formula used to calculate the token Assurance Level. This way, high value transaction on non-reputable or rooted hardware will not be allowed limiting the risk for the Issuer.

Another point raised during the analysis of tokenization is the fact that tokens are not device bound. This means if a token is stolen from a device it is free to be used on another device. The EMVCo, allows tying a token to a particular method such as a token that can only be used for a contactless payment and cannot be used for online payments but the token is not tied to the device it was enrolled to. This risk is particularly high in MSD payments where the dynamic CVV does not include any data provided by the terminal, not even the amount of the transaction. This means a stolen token can simply be used on another mobile device to make a payment for any amount as long as it is under the limit. A method whereby a unique device ID becomes part of the signature, has been proposed as a countermeasure to this risk.

The final point raised in the analysis of tokenization is the requirement of a synchronization mechanism. With current methods, the TR provides tokens to the mobile device but the TR does not have a way of identifying if the tokens provided have been used. Hence a mobile device can request a replenishment even if the current token pool has not yet been depleted. To ensure that a device only contains a limited amount of tokens, a synchronization mechanism is required whereby the TR can identify, through the TSP, how many tokens have been used from a particular device. Again this is why the previous point, token-device mapping, is an important requirement. This synchronization mechanism will create a feedback loop, between the TR, mobile device and the TSP.

In the last section of the analyses, the implications in storing cryptographic keys and running cryptographic functions on a mobile device was analyzed. The main outcome of this section was the requirement of a TEE in the mobile device to store the keys and operate cryptographic functions in the TEE. Both LUKs and the RSA key used during Offline Data Authentication should be stored in the TEE. It is the author's opinion that although LUKs are 'limited in use' it is not enough to warrant storing the LUK's in the HCE wallet application's memory due to the impacts of a wide breach and the fact that LUKs provide access to download further keys.

Overall it is the author's opinion, that, to achieve a secure HCE based payment system, four important requirements have to be met:

- i. A TEE is required on the mobile device. The TEE provides a safe storage for keys in a way where a fraudster, through a compromised phone, can use the key but cannot gain access to the key. This is required to store the LUKs and the RSA Key used during Offline Data Authentication. If a TEE is not available then techniques like white box cryptography can be used to store the keys in application memory.
- ii. If a TEE becomes a requirement then it is the author's opinion that the TR should also be stored in the TEE. This is based on the following facts:
  - a. A key, typically the LUK, is required to authenticate to the TR. This is a 'master' key to obtain further LUKs. The risk value of such a key is similar to the tUDK stored within the TR in the cloud. Thus the TR can be situated in the TEE as the LUK used to access the TR carries the same level of risk as the tUDK.
  - b. A secure connection is required between the TR and the HCE wallet. If the TR is in the TEE then no data is passed over the internet as it would be a direct connection from the HCE Wallet App to the TEE. This reduces the risk of attacks on data in transit between the TR in the cloud and the HCE Wallet App.

- c. No need to store a set of LUKs on the mobile device. The TR in the TEE would be developed to generate LUKs on the fly during a transaction and subsequently generate the application cryptogram and send it to the HCE Wallet Application. This way the LUKs will never leave the TR, within the TEE.
  
- iii. A strong way of uniquely identifying a mobile device is required. This will be used to:
  - a. Authenticate the device during cloud services communication.
  - b. Map tokens to that particular device thereby limiting the chances of that token to be used elsewhere or on other devices.
  - c. Since the Issuer uses the mobile phone number to ID&V the cardholder during enrolment, the mobile phone number should become part of the Mobile's unique Identity.
  
- iv. A closed loop supply-demand synchronization is required to keep track of issued tokens (tPAN and LUKs). Token requestor's should only generate and issue new LUKs to a Wallet if they have confirmation that previously issued LUKs have expired.

Furthermore, for HCE to be successful it depends on the level of trust between all the stakeholders involved during the enrolment, provisions and the lifecycle management of a token. This is potentially achieved through ongoing due diligence, agreed terms and conditions and ensuring that security mechanisms, as proposed in this section and by industry good practices, are in place.

## 6 General Conclusion

### 6.1 Summary

The aim of this project, overall, was to review how HCE, the supporting technologies and processes are implemented to make up a secure electronic mobile payment method.

The evolution of HCE has only just started and by time it will go through a series of refinements. The technology will stabilize by time and, as more deployments are carried out, its weaknesses are identified and mitigated. During the course of this project, a number of standards and specifications related to contactless payments and HCE have been published and/or updated by industry expert bodies. HCE is also empowering various bodies to work together to deliver and strengthen adequate security, interoperability and integrity of the payments ecosystems.

The aim of this project was achieved through a set of objectives described below:

#### **i. Review relevant existing literature on NFC technology and HCE payment implementations**

HCE relies on different technologies and as such the first objective of this project was to review existing literature related to these technologies. At the physical layer, HCE relies on NFC and hence the first part of the analysis was focused on attacks on NFC. It was immediately clear that controls at higher levels are required to countermeasure the weaknesses at the physical layer. The next step was to review the attacks at the application layer including attacks on the POS and the payment infrastructure and subsequently attacks on the cryptography and key management used in contactless payments.

The analysis was then focused on known weaknesses in the hardware and software that make up an HCE payment wallet. These included the kernel, the operating system and secure memory areas within a mobile device. HCE also relies on different authentication methods such as biometric methods hence known attacks on these methods were outlined too. To conclude the section a review on the tokenization infrastructure and its weaknesses outlined to date was provided.

Within the same section, a review of two possible HCE implementations was provided along with a list of the advantages and disadvantages of each implementation. Overall a thorough review of the weaknesses associated with HCE has been provided.

#### **ii. Model the HCE payment transaction process flow as well as the tokenisation process using formal methods**

A finite-state machine model of an HCE payment application was developed. The model was based on the specifications and requirements provided by VISA. The model was implemented using Stateflow which is a tool provided by Matlab within Simulink. The model defines a finite set of states and transitions between different states during a payment and during network communication with several online services.

The design of the model resembles a typical wallet application with different processes running in parallel. The model was split into three processes; the Payment Transaction responsible for the NFC communication with the POS during a payment, the Account Management process responsible for storing and updating credentials and the Token Requestor responsible for generating and replenishing tokens and LUKs on the mobile device.

In this project, the model served as a tool to identify weaknesses and possible countermeasures at a later stage in this study. Specific conditions can be simulated by altering the input data to simulate conditions that can occur in a payment transaction or during communication with online services such as during token replenishment.

**iii. Analyze the security aspects and weaknesses within the elements that make up an HCE payment scheme and EMV payment tokenization infrastructure**

To tackle this objective the payment processes were split into different sub-processes. For example, the cardholder authentication process was isolated and analysed. Each sub-processes was then subjected to different conditions within the allowances of the standards and specifications. The finite state model was used to simulate how the payment wallet behaves in certain conditions and as a verification tool to verify the existence of a weakness.

The first part of the analysis was focused on cardholder verification. The issues identified were mostly related to the implementation of CDCVM where the mobile device is used for cardholder verification. The second section of the analysis was focused on tokenization. Different issues were identified starting from the enrolment process, the replenishment process up to when the token is used during a transaction. Further analysis was done on the efficacy of the risk analysis done by the TSP and portability of a token.

The third section in the analysis was aimed at credentials storage and their use during a payment and during communication with online services. The analysis highlighted the importance of safe storage even when tokenization is used.

The last part of the analysis was focused on generic issues that have to be considered. The aim here was to identify and define issues that come part in parcel with the use of smartphones, software and new methods of authentication (biometric).

**iv. Conduct a high level risk assessment on the identified threats and how certain controls can be applied to mitigate certain risks**

The work on this objective builds on the outcome of the third objective. For each weakness identified the risk was outlined by providing a scenario and conditions required for the risk to occur. For each risk identified a countermeasure, where possible, was provided. It is important to note that some of the countermeasures provided rely on the availability of certain techniques and technologies that are currently not yet developed or available. These have been mentioned as possible areas of further research in Section 6.2.

I believe the project has achieved the objectives set and the foreground generated in this project serves as a reference for future work on the subject. I believe the main findings of this work can be used, as a reference to develop tighter controls to further improve the security in HCE based payments. I would also have preferred to also extend the scope of this project to include security analysis on other parties' involvements such as Merchants, Acquirers, Payment network providers and Issuers. This may be another consideration that I may decide to explore in the future when time permits.

I have found this project challenging, interesting and I have enjoyed it throughout. The project covered several phases: research, model development and security analysis to name a few. The modules that I have took during my MSc, particularly "An Introduction to Cryptography", "Security Testing Theory and Practice", "Computer Security" and "Security Management" were an invaluable source of information and guidance while undertaking this project.

This project strengthened my knowledge about electronic payment schemes and the security techniques used to protect them. Furthermore, through this research it was interesting to observe how information security is a necessity in shaping current and future technology.

## 6.2 Areas of Further Research

During the security analysis in Section 5, several weaknesses have been identified. A number of countermeasures for such weaknesses have been provided but a number of these rely on the availability of certain techniques that are currently not yet implemented or available. The following are the areas where further research would be required:

### **Mobile Device Unique Identification**

One of the requirements for a number of countermeasures listed in Section 5 is a way to uniquely identify a mobile device. Current methods such as using the mobile's IMEI have been proven weak since such identification can be spoofed. To provide further security, some form of identification which is not, relatively easy, to change from within a rooted device and during transit is required. Currently, an attacker, through malware running at root level, on a rooted mobile device, can change most of the unique IDs on the mobile device.

A possible technique could involve some form of globally unique ID which is hard coded in the TEE. Therefore, this can be integrated as part of the cryptographic functions without the need for the ID to actually leave the TEE in plaintext. It is the author's opinion that further research on this subject is required to enable better authentication and mapping of credentials to a device to further limit the portability of tokens, and credentials used during a transaction.

### **API provided by card scheme for TEE functions**

Currently, HCE Wallet Application developers are provided with requirements and specifications, but none of them are detailed enough to specify how certain functions and processes are to be implemented on the mobile device. It is also difficult to verify and/or certify such 'correct implementations' especially if the code for the application is not available or provided.

It is the author's opinion that an industry body, such as VISA and other card payment schemes, should provide an API (Library) to HCE developers. This way the card scheme has assurance that keys are stored safely and cryptographic functions are implemented correctly within the TEE. This API can utilize the TEE directly or else utilize the OS's API, such as Android's Trusty OS. The API could have functions to store and/or replenish tokens, generate the Application Cryptogram and generate the signature in an offline transaction.

### **Specifications for evaluating biometric ID&V methods**

Different biometric ID&V methods have been mentioned and for the majority of them a number of weaknesses and attacks have been identified by academics and researchers. Still, at the time of writing, no specification has been proposed by the industry on acceptance of an ID&V method. While this opens up new opportunities of innovation in the field, it is the author's opinion that some form of baseline acceptance metrics should be developed. Any method used for HCE should pass these baseline tests before being approved for use in HCE.

## 7 Bibliography

- [1] ISACA, "Is Mobile the Winner in Payment Security?," ISACA, 2016.
- [2] Intel Research, "An Introduction to RFID Technology," *IEEE Pervasive Computing*, no. 1536-1268/06, 2006.
- [3] International Organization for Standardization, "ISO/IEC 14443-3:2016," June 2016. [Online]. Available: [http://www.iso.org/iso/home/store/catalogue\\_ics/catalogue\\_detail\\_ics.htm?csnumber=70171](http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=70171). [Accessed September 2016].
- [4] Smart Payment Association, "An Overview of Contactless Payment Benefits and Worldwide Deployments," 2016.
- [5] The UK Cards Association Limited, "Contactless Statistics," [Online]. Available: [http://www.theukcardsassociation.org.uk/contactless\\_contactless\\_statistics/](http://www.theukcardsassociation.org.uk/contactless_contactless_statistics/). [Accessed September 2016].
- [6] I. Gurulian, C. Shepherd, K. Markantonakis, R. N. Akram and K. Mayes, "When Theory and Reality Collide: Demystifying the Effectiveness of Ambient Sensing for NFC-based Proximity Detection by Applying Relay Attack Data," Information Security Group, Smart Card Centre, Royal Holloway, University of London, United Kingdom, 2016.
- [7] Barclaycard, "Our story so far...," December 2014. [Online]. Available: <https://timeline.barclaycard/>. [Accessed September 2016].
- [8] Microsoft, "Nokia's NFC phone history," 11 April 2012. [Online]. Available: <https://blogs.windows.com/devices/2012/04/11/nokias-nfc-phone-history/>. [Accessed September 2016].
- [9] NFC World, "Google Wallet ends support for physical secure elements," March 2014. [Online]. Available: <http://www.nfcworld.com/2014/03/17/328326/google-wallet-ends-support-physical-secure-elements/>. [Accessed September 2016].
- [10] BMO Bank of Montreal, CIBC, National Bank of Canada, RBC Royal Bank, Scotiabank and TD Bank Group, "Payments Security White Paper," Canadian Bankers Association, 2015.
- [11] Visa, "Visa to Enable Secure, Cloud-Based Mobile Payments," 19 February 2014. [Online]. Available: <http://investor.visa.com/news/news-details/2014/Visa-to-Enable-Secure-Cloud-Based-Mobile-Payments/default.aspx>. [Accessed October 2016].
- [12] MasterCard, "MasterCard to Use Host Card Emulation (HCE) for NFC-Based Mobile Payments," 19 February 2014. [Online]. Available: <http://newsroom.mastercard.com/press-releases/mastercard-to-use-host-card-emulation-hce-for-nfc-based-mobile-payments/>. [Accessed October 2016].
- [13] Microsoft, "Microsoft Wallet with tap to pay is now available for Windows Insiders," 21 June 2016. [Online]. Available: <https://blogs.windows.com/windowsexperience/2016/06/21/microsoft-wallet-with-tap-to-pay-is-now-available-for-windows-insiders/#t4eSj4oE3XGqRSiC.97>. [Accessed September 2016].
- [14] Secure Technology Alliance, "Smart Card Alliance," [Online]. Available: <http://www.smartcardalliance.org/>. [Accessed September 2016].
- [15] EMVCo, LLC, "EMV Contactless," [Online]. Available: <https://www.emvco.com/specifications.aspx?id=21>. [Accessed September 2016].
- [16] International Organization for Standardization, "ISO/IEC 14443-1:2016," March 2016. [Online]. Available: <https://www.iso.org/standard/70170.html>. [Accessed September 2016].
- [17] International Organization for Standardization, "ISO/IEC 14443-2:2016," July 2016. [Online]. Available: <https://www.iso.org/standard/66288.html>. [Accessed September 2016].
- [18] International Organization for Standardization, "ISO/IEC 14443-4:2016," June 2016. [Online]. Available: <https://www.iso.org/standard/70172.html>. [Accessed September 2016].
- [19] M. Bolhuis, "Using an NFC-equipped mobile phone as a token in physical access control," Master's thesis, University of Twente, 2014.
- [20] NXP, "MIFARE® ICs," [Online]. Available: [http://www.nxp.com/products/identification-and-security/mifare-ics:MC\\_53422](http://www.nxp.com/products/identification-and-security/mifare-ics:MC_53422). [Accessed September 2016].

- [21] International Organization for Standardization, "ISO/IEC 18092:2013," March 2013. [Online]. Available: [http://www.iso.org/iso/home/store/catalogue\\_tc/catalogue\\_detail.htm?csnumber=56692](http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56692). [Accessed December 2016].
- [22] D. Tushie, "An Introduction to NFC Standards," *ICMA Card Manufacturing*, October 2012.
- [23] International Organization for Standardization, "ISO/IEC 7816-4:2013," April 2013. [Online]. Available: <https://www.iso.org/standard/54550.html>. [Accessed December 2016].
- [24] Accredited Standard Committee X9, Inc., "History of X9," [Online]. Available: <http://x9.org/about/history-of-x9/>. [Accessed October 2016].
- [25] EMVCo, "The EMV Payment Tokenisation Specification – Technical Framework," March 2014. [Online]. Available: <https://www.emvco.com/specifications.aspx?id=263>. [Accessed December 2016].
- [26] PCI Security Standards Council, "Tokenization Product Security Guidelines - Irreversible and Reversible Tokens," PCI Security Standards Council, 2015.
- [27] National Institute of Standards and Technology, "About NIST," 25 August 2016. [Online]. Available: <https://www.nist.gov/>. [Accessed October 2016].
- [28] Smart Card Alliance, "Technologies for Payment Fraud Prevention: EMV, Encryption and Tokenization," SMART CARD ALLIANCE PAYMENTS COUNCIL, 2014.
- [29] Visa, "Visa Europe Payment Token Service Android Pay Member Implementation Guide for Issuers," Visa, 2016.
- [30] EMVCo, "mPOS FAQs," October 2016. [Online]. Available: <http://www.emvco.com/faq.aspx?id=287#12>. [Accessed October 2016].
- [31] FIDO Alliance, "About The FIDO Alliance," [Online]. Available: <https://fidoalliance.org/about/overview/>. [Accessed September 2016].
- [32] FIDO Alliance, "EMVCo and the FIDO Alliance Collaborate on Mobile Payment Authentication," July 2016. [Online]. Available: <https://fidoalliance.org/fido-emvco-mou/>. [Accessed October 2016].
- [33] FIDO Alliance, "FIDO 1.0 Specifications are Published and Final Preparing for Broad Industry Adoption of Strong Authentication in 2015," 9 December 2014. [Online]. Available: <https://fidoalliance.org/fido-1-0-specifications-published-and-final/>. [Accessed October 2016].
- [34] NFC World, "Fido Alliance adds authentication support for NFC and BLE," [Online]. Available: <https://www.nfcworld.com/2015/07/01/336359/fido-alliance-adds-authentication-support-for-nfc-and-ble/>. [Accessed October 2016].
- [35] FIDO Alliance, "Members: Bringing together an ecosystem," 2017. [Online]. Available: <https://fidoalliance.org/participate/members/>. [Accessed February 2017].
- [36] EyeVerify Inc., "EyeVerify Achieves FIDO Alliance Certification for its Eyeprint ID™ Authentication Technology," 5 January 2016. [Online]. Available: <http://www.eyeverify.com/eyeprint-id-fido-certified>. [Accessed February 2017].
- [37] FIDO Alliance, "InfoSecurity: FIDO, EMVCo Prep for Pay-by-Selfie Era," 25 October 2016. [Online]. Available: <https://fidoalliance.org/infosecurity-fido-emvco-prep-for-pay-by-selfie-era/>. [Accessed February 2017].
- [38] Mastercard, "Mastercard makes fingerprint and 'selfie' payment technology a reality," 4 October 2016. [Online]. Available: <http://newsroom.mastercard.com/eu/press-releases/mastercard-makes-fingerprint-and-selfie-payment-technology-a-reality/>. [Accessed February 2017].
- [39] CNBC LLC., "HSBC customers can open new bank accounts using a selfie," 5 September 2016. [Online]. Available: <http://www.cnbc.com/2016/09/05/hsbc-customers-can-open-new-bank-accounts-using-a-selfie.html>. [Accessed February 2017].
- [40] T. J. Neal and D. L. Woodard, "Surveying Biometric Authentication for Mobile Device Security," *Journal of Pattern Recognition Research 1 (2016) 74-110*, p. 37, 2016.
- [41] PCI Security Standards Council, "Payment Card Industry Council Advances point-to-point encryption standard," June 2015. [Online]. Available: [https://www.pcisecuritystandards.org/pdfs/15\\_06\\_30%20PCI%20P2PE\\_v2\\_Press-Release.pdf](https://www.pcisecuritystandards.org/pdfs/15_06_30%20PCI%20P2PE_v2_Press-Release.pdf). [Accessed September 2016].

- [42] A. Greenberg, "Hacker Lexicon: What Is End-to-End Encryption?," 25 November 2014. [Online]. Available: <https://www.wired.com/2014/11/hacker-lexicon-end-to-end-encryption/>. [Accessed February 2017].
- [43] EMVCo, LLC., "Software-based Mobile Payment Security Requirements," EMVCo, LLC., December 2016.
- [44] GSM Arena, "Phone finder results," September 2016. [Online]. Available: <http://www.gsmarena.com/results.php3?chkNFC=selected>. [Accessed September 2016].
- [45] NFC World, "More than 100m people will make an NFC mobile payment in 2016," September 2016. [Online]. Available: <http://www.nfcworld.com/2016/03/10/343180/more-than-100m-people-will-make-an-nfc-mobile-payment-in-2016/>. [Accessed September 2016].
- [46] ISO, "ISO/IEC 18000-3:2010," November 2010. [Online]. Available: [http://www.iso.org/iso/catalogue\\_detail.htm?csnumber=53424](http://www.iso.org/iso/catalogue_detail.htm?csnumber=53424). [Accessed February 2017].
- [47] NFC Forum, "NFC Forum Technical Specifications," 2017. [Online]. Available: [http://members.nfc-forum.org/specs/spec\\_list/](http://members.nfc-forum.org/specs/spec_list/). [Accessed February 2017].
- [48] Smart Card Alliance, "Host Card Emulation (HCE)," 18 June 2015. [Online]. Available: [http://www.smartcardalliance.org/downloads/HCE\\_Webinar\\_FINAL\\_061815.pdf](http://www.smartcardalliance.org/downloads/HCE_Webinar_FINAL_061815.pdf). [Accessed October 2016].
- [49] International Organization for Standardization, "ISO/IEC 7816-5:2004," December 2004. [Online]. Available: <https://www.iso.org/standard/34259.html>. [Accessed October 2016].
- [50] I. Kirschenbaum and A. Wool, "How to Build a Low-Cost, Extended-Range RFID Skimmer," in *In Usenix Security*, 2006.
- [51] Mail Online, "The electronic pickpocket: 'Scammer steals hundreds of pounds by touching victims' pockets with sales device – and transferring cash from their contactless payment credit cards'," 17 February 2016. [Online]. Available: <http://www.dailymail.co.uk/news/article-3451307/The-electronic-pickpocket-Scammer-steals-hundreds-pounds-touching-victims-pockets-sales-device-transferring-cash-contactless-payment-credit-cards.html>. [Accessed September 2016].
- [52] International Business Times, "Virtual pickpockets steal money from contactless bank cards by bumping into victims claims Londoner," 22 October 2015. [Online]. Available: <http://www.ibtimes.co.uk/virtual-pickpockets-steal-money-contactless-bank-cards-by-bumping-into-victims-claims-londoner-1525211>. [Accessed September 2016].
- [53] T. S. Heydt-Benjamin, D. V. Bailey, K. Fu, A. Juels and T. O'Hare, "Vulnerabilities in First-Generation RFID-enabled," *USENIX-SS'06 Proceedings of the 15th conference on USENIX Security Symposium*, vol. 15, September 2006.
- [54] A. Alfaraj, "RFID Insecurity for Entity Authentication," Diss. Master's thesis University College London, Computer Science, 2006.
- [55] Google, "Host-based Card Emulation," [Online]. Available: <https://developer.android.com/guide/topics/connectivity/nfc/hce.html#ScreenOffBehavior>. [Accessed 12 September 2016].
- [56] Microsoft, "How to write an NFC Host Card Emulation (HCE) app with Windows 10 for Mobile," 1 May 2015. [Online]. Available: <https://blogs.msdn.microsoft.com/nfc/2015/05/01/how-to-write-an-nfc-host-card-emulation-hce-app-with-windows-10-for-mobile/>. [Accessed September 2016].
- [57] NFC World, "SpotMe app lets Android users spend Apple Pay cash," 3 March 2015. [Online]. Available: <http://www.nfcworld.com/2015/03/03/334455/spotme-app-lets-android-users-spend-apple-pay-cash/>. [Accessed September 2016].
- [58] MasterCard, "MasterCard Contactless Performance Requirement," 20 March 2014. [Online]. Available: [https://www.paypass.com/Card\\_TA/PayPassPerformMeasureAppNoteNumb7.pdf](https://www.paypass.com/Card_TA/PayPassPerformMeasureAppNoteNumb7.pdf). [Accessed October 2016].
- [59] G. Hancke, "Eavesdropping Attacks on High-Frequency RFID Tokens," *Journal of Computer Security - 2010 Workshop on RFID Security (RFIDSec'10 Asia)*, vol. 19, no. 2, June April 2011.
- [60] The Smart Card Alliance Mobile & NFC Council, "Host Card Emulation (HCE) 101," Smart Card Alliance, August 2014.



- [61] T. P. Diakos, J. A. Briffa, T. W. C. Brown and S. Wesemeyer, "Eavesdropping near-field contactless payments: a quantitative analysis," *IET Journal of Engineering*, p. 7, 12 September 2013.
- [62] T. W. C. Brown, T. Diakos and J. A. Briffa, "Evaluating the Eavesdropping Range of Varying Magnetic Field Strengths in NFC Standards," in *Antennas and Propagation (EuCAP), 2013 7th European Conference on. IEEE*, 2013.
- [63] F. Pfeiffer, K. Finkenzeller and E. Biebl, "Theoretical Limits of ISO/IEC 14443 type A RFID Eavesdropping Attacks," in *Smart SysTech 2012; European Conference on Smart Objects, Systems and Technologies*, 2012.
- [64] G. Hancke, "A Practical Relay Attack on ISO 14443 Proximity Cards," *Technical report, University of Cambridge Computer Laboratory*, vol. 59, pp. 382-385, 2005.
- [65] L. Francis, G. Hancke, K. Mayes and K. Markantonakis, "Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones," in *The 2012 Workshop on RFID and IoT Security*, 2012.
- [66] M. Roland, J. Langer and J. Scharinger, "Applying relay attacks to Google Wallet," in *2013 5th International Workshop on Near Field Communication (NFC)*, Zurich, Switzerland, February 2013.
- [67] W. Issovits and M. Hutter, "Weaknesses of the ISO/IEC 14443 Protocol Regarding Relay Attacks," in *R2011 IEEE International Conference on RFID-Technologies and Applications*, November 2011.
- [68] J. v. d. Breekel, "Relaying EMV Contactless Transactions using Off-The-Shelf Android Devices," in *BlackHat Asia*, March 2015.
- [69] J. Vila and R. J. Rodríguez, "Practical Experiences on NFC Relay Attacks with Android," in *Radio Frequency Identification. Security and Privacy Issues*, Springer International Publishing, 2015, pp. 87-103.
- [70] EMVCo, LLC, "EMV Contactless Specifications for Payment Systems," EMVCo, LLC, August 2007.
- [71] STMicroelectronics, "TN1216 Technical Note - ST25 NFC guide," 2016 October. [Online]. Available: [http://www.st.com/content/ccc/resource/technical/document/technical\\_note/f9/a8/5a/0f/61/bf/42/29/DM00190233.pdf/files/DM00190233.pdf/jcr:content/translations/en.DM00190233.pdf](http://www.st.com/content/ccc/resource/technical/document/technical_note/f9/a8/5a/0f/61/bf/42/29/DM00190233.pdf/files/DM00190233.pdf/jcr:content/translations/en.DM00190233.pdf). [Accessed October 2016].
- [72] Y. Oren, D. Schirman and A. Wool, "RFID Jamming and Attacks on Israeli e-Voting," in *Smart SysTech 2012; European Conference on Smart Objects, Systems and Technologies*, 2012.
- [73] J. Gummeson, B. Priyantha, D. Ganesan, D. Thrasher and P. Zhang, "EnGarde: Protecting the Mobile Phone from Malicious NFC Interactions," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, 2013.
- [74] Armourcard, "What is Armourcard?," 2016. [Online]. Available: <https://www.armourcard.com/what-is-armourcard>. [Accessed September 2016].
- [75] T. R. Harris and R. L. Teece, "Inhibiting unauthorised contactless reading of a contactless readable object". Australia Patent 2013354900, 5 December 2013.
- [76] E. Haselsteiner and K. Breitfuß, "Security in Near Field Communication (NFC)," in *In Workshop on RFID Security*, Austria, 2012.
- [77] M. Roland and J. Langer, "Cloning Credit Cards: A combined pre-play and downgrade attack on EMV Contactless," in *WOOT'13 Proceedings of the 7th USENIX conference on Offensive Technologies*, Washington, D.C., 2013.
- [78] M. Emms and A. v. Moorsel, "Practical Attack on Contactless Payment Cards," *HCI2011 Workshop-Heath, Wealth and Identity Theft*, 2011.
- [79] M. Bond, O. Choudary, S. J. Murdoch, S. Skorobogatov and R. Anderson, "Chip and Skim: cloning EMV cards with the pre-play attack," in *2014 IEEE Symposium on Security and Privacy*, 2014.
- [80] EMVCo, "EMVCo Type Approval Terminal Level 2 Test Cases - 4.3a," November 2011. [Online]. Available: <https://www.emvco.com/approvals.aspx?id=59>. [Accessed October 2016].
- [81] M. Emms, B. Arief, L. Freitas, J. Hannon and A. v. Moorsel, "Harvesting High Value Foreign Currency Transactions from EMV Contactless Credit Cards Without the PIN," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

- [82] ProSecurityZone, "Tamper detection for POS terminals," 6 August 2008. [Online]. Available: [http://www.prosecurityzone.com/News\\_Detail\\_Tamper\\_detection\\_for\\_pos\\_terminals\\_4728.asp#axzz4ZLVmdp1x](http://www.prosecurityzone.com/News_Detail_Tamper_detection_for_pos_terminals_4728.asp#axzz4ZLVmdp1x). [Accessed February 2017].
- [83] The Telegraph, "Chip and pin scam 'has netted millions from British shoppers'," 10 October 2008. [Online]. Available: <http://www.telegraph.co.uk/news/uknews/law-and-order/3173346/Chip-and-pin-scam-has-netted-millions-from-British-shoppers.html>. [Accessed February 2017].
- [84] Visa, "Targeted Hospitality Sector Vulnerabilities," 1 December 2009. [Online]. Available: [http://www.visa.com.sg/merchants/include/targeted\\_hosp\\_vulnerabilities.pdf](http://www.visa.com.sg/merchants/include/targeted_hosp_vulnerabilities.pdf). [Accessed October 2016].
- [85] Visa, "Preventing Memory-Parsing Malware Attacks on Grocery Merchants," 11 April 2013. [Online]. Available: <https://usa.visa.com/dam/VCOM/download/merchants/alert-prevent-grocer-malware-attacks-04112013.pdf>. [Accessed October 2016].
- [86] KrebsSecurity, "Data Breach At Oracle's MICROS Point-of-Sale Division," 8 August 2016. [Online]. Available: <https://krebsonsecurity.com/category/data-breaches/>. [Accessed December 2016].
- [87] CNN, "Breach at third party payment processor affects 22 million Visa cards and 14 million MasterCard.," 27 July 2005. [Online]. Available: [http://money.cnn.com/2005/06/17/news/master\\_card/](http://money.cnn.com/2005/06/17/news/master_card/). [Accessed October 2016].
- [88] Bloomberg L.P., "Heartland Payment Systems: Lessons Learned from a Data Breach," 7 July 2009. [Online]. Available: <https://www.bloomberg.com/news/articles/2009-07-06/lessons-from-the-data-breach-at-heartlandbusinessweek-business-news-stock-market-and-financial-advice>. [Accessed October 2016].
- [89] Cryptomathic A/S, "EMV Key Management - Explained," Cryptomathic A/S, 2013.
- [90] DELL Technologies, "What Key Size Should Be Used?," October 2016. [Online]. Available: <http://www.passmarksecurity.com/emc-plus/rsa-labs/standards-initiatives/key-size.htm>. [Accessed October 2016].
- [91] EMVCo, LLC, "EMV - Issuer and Application Security Guidelines," EMVCo, LLC, 2014.
- [92] Cisco, "Crypto key generate rsa," 30 April 2015. [Online]. Available: [http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/security/a1/sec-a1-xe-3se-3850-cr-book/sec-a1-xe-3se-3850-cr-book\\_chapter\\_0110.pdf](http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/security/a1/sec-a1-xe-3se-3850-cr-book/sec-a1-xe-3se-3850-cr-book_chapter_0110.pdf). [Accessed October 2016].
- [93] J.-S. Coron, D. Naccache and J. P. Stern, On the Security of RSA Padding, Springer-Verlag, 1999, pp. 1-18.
- [94] Smart Payment Association, "Strengthening Card Authentication: a migration to DDA," Smart Payment Association, 2015.
- [95] J. P. Degabriele, A. Lehmann, K. G. Paterson, N. P. Smart and M. Strefler, "On the Joint Security of Encryption and Signature in EMV," in *Cryptographers' Track at the RSA Conference*, 2012.
- [96] D. Bleichenbacher, Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS 1, Springer, 1998, CRYPTO '98, volume 1462 of LNCS, pages 549-570.
- [97] MasterCard, "PayPass M-TIP Test Case User Guide," MasterCard, 2014.
- [98] MITRE Corporation, "CVE Details," 10 October 2016. [Online]. Available: [http://www.cvedetails.com/product/19997/Google-Android.html?vendor\\_id=1224](http://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224). [Accessed 10 October 2016].
- [99] X. Hei, X. Du and S. Lin, "Two Vulnerabilities in Android OS Kernel," in *2013 IEEE International Conference on Communications (ICC)*, 2013.
- [100] M. Xu, C. Song, Y. Ji, M.-W. Shih, K. Lu, C. Zheng, R. Duan, Y. Jang, B. Lee, C. Qian, S. Lee and T. Kim, "Toward Engineering a Secure Android Ecosystem: A Survey of Existing Techniques," *ACM Computing Surveys* 49,2, p. 47, August 2016.
- [101] Zimperium, "Experts Found a Unicorn in the Heart of Android," 27 July 2015. [Online]. Available: <https://blog.zimperium.com/experts-found-a-unicorn-in-the-heart-of-android>. [Accessed October 2016].
- [102] MITRE Corporation, "Common Vulnerabilities and Exposures List," [Online]. Available: <https://cve.mitre.org/cve/cve.html>. [Accessed February 2017].

- [103] ISO, "ISO/IEC 9899:2011," December 2011. [Online]. Available: [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=57853](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57853). [Accessed February 2017].
- [104] GlobalPlatform, "About GlobalPlatform," 2017. [Online]. Available: <http://www.globalplatform.org/>. [Accessed February 2017].
- [105] GlobalPlatform, "GlobalPlatform made simple guide: Trusted Execution Environment (TEE) Guide," 2017. [Online]. Available: [http://www.globalplatform.org/mediaguidetee.asp#\\_Toc419214142](http://www.globalplatform.org/mediaguidetee.asp#_Toc419214142). [Accessed February 2017].
- [106] ARM Ltd., "ARM Trust Zone," October 2016. [Online]. Available: <https://www.arm.com/products/security-on-arm/trustzone>. [Accessed October 2016].
- [107] ARM Ltd., "The mobile ecosystem runs on ARM," October 2016. [Online]. Available: <https://www.arm.com/markets/mobile>. [Accessed October 2016].
- [108] V. Costan and S. Devadas, "Intel SGX Explained," Cryptology ePrint Archive, 2016.
- [109] Intel, "Intel® Software Guard Extensions Remote Attestation End-to-End Example," 8 July 2016. [Online]. Available: <https://software.intel.com/en-us/articles/intel-software-guard-extensions-remote-attestation-end-to-end-example>. [Accessed February 2017].
- [110] Google, "Trusty TEE," October 2016. [Online]. Available: <https://source.android.com/security/trusty>. [Accessed October 2016].
- [111] Federal Reserve Bank of Boston, Giesecke & Devrient, "Understanding the Role of Host Card Emulation in Mobile Wallets," Federal Reserve Bank of Boston, 2016.
- [112] Google, "Security Tips," October 2016. [Online]. Available: <https://developer.android.com/training/articles/security-tips.html>. [Accessed October 2016].
- [113] MWR Labs, "Sandbox bypass through Google Admin WebView," 13 August 2015. [Online]. Available: <https://labs.mwrinfosecurity.com/assets/AdvisoriesFiles/Sandbox-Advisory1.pdf>. [Accessed October 2016].
- [114] Google, "Android Keystore System," October 2016. [Online]. Available: <https://developer.android.com/training/articles/keystore.html>. [Accessed October 2016].
- [115] Google, "Hardware-backed Keystore," October 2016. [Online]. Available: <https://source.android.com/security/keystore/index.html>. [Accessed October 2016].
- [116] T. Cooijmans, J. d. Ruiter and E. Poll, "Analysis of Secure Key Storage Solutions on Android," in *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, 2014.
- [117] R. Hay and A. Dayan, "Android KeyStore Stack Buffer Overflow," 2014.
- [118] Apple Inc., "Consumer Device Cardholder Verification Method," 11 October 2016. [Online]. Available: <https://support.apple.com/en-mt/HT202527>. [Accessed October 2016].
- [119] Google, "Android 6.0 APIs," October 2016. [Online]. Available: <https://developer.android.com/about/versions/marshmallow/android-6.0.html>. [Accessed October 2016].
- [120] NFC World, "EMVCo and Fido to deliver biometric authentication standards for mobile payments," 24 October 2016. [Online]. Available: <http://www.nfcworld.com/2016/10/24/347983/emvco-fido-deliver-biometric-authentication-standards-mobile-payments/>. [Accessed October 2016].
- [121] MasterCard, "MasterCard Best Practices for Mobile POS Acceptance," November 2013. [Online]. Available: [http://www.mastercard.com/us/company/en/docs/MasterCard\\_Mobile\\_Point\\_Of\\_Sale\\_Best\\_Practices.pdf](http://www.mastercard.com/us/company/en/docs/MasterCard_Mobile_Point_Of_Sale_Best_Practices.pdf). [Accessed October 2016].
- [122] L. Simon and R. Anderson, "PIN Skimmer: Inferring PINs Through The Camera and Microphone," in *In Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices*, pp. 67-78. ACM, 2013., 2013.
- [123] R. Spreitzer, "PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices," in *SPSM '14 Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, Scottsdale, Arizona, USA, 2014.

- [124] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze and J. M. Smith, "Smudge Attacks on Smartphone Touch Screens," in *WOOT'10 Proceedings of the 4th USENIX conference on Offensive technologies*, Washington, DC, 2010.
- [125] Neurotechnology, "VeriFinger SDK," October 2016. [Online]. Available: <http://www.neurotechnology.com/verifinger.html>. [Accessed October 2016].
- [126] The Guardian, "Hacker fakes German minister's fingerprints using photos of her hands," 30 December 2014. [Online]. Available: <https://www.theguardian.com/technology/2014/dec/30/hacker-fakes-german-ministers-fingerprints-using-photos-of-her-hands>. [Accessed October 2016].
- [127] Y. Zhang, Z. Chen, H. Xue and T. Wei, "Fingerprints On Mobile Devices: Abusing and Leaking," in *Black Hat Conference*, 2015.
- [128] J. Galbally, J. Fierrez and J. Ortega-Garcia, "Vulnerabilities in Biometric Systems: Attacks and Recent Advances in Liveness Detection," 2014.
- [129] S. Mendoza, "Samsung Pay: Tokenized Numbers, Flaws and Issues," 4 August 2016. [Online]. Available: <https://www.blackhat.com/docs/us-16/materials/us-16-Mendoza-Samsung-Pay-Tokenized-Numbers-Flaws-And-Issues.pdf>. [Accessed October 2016].
- [130] Visa, "Visa Cloud-Based Payments Contactless Specification," Visa, 2016.
- [131] T. MalcomVetter, "Breaking Credit Card Tokenization – Part 2," 5 January 2016. [Online]. Available: <https://www.optiv.com/resources/blog?page=1&searchQuery=credit-card-tokenization&year=>. [Accessed October 2016].
- [132] GSMA; Consult Hyperion, "The New Mobile Payment Landscape," GSMA, 2015.
- [133] CO-OP Financial Services, "Android Pay FAQ," October 2016. [Online]. Available: [https://www.co-opfs.org/media/248639/androidpay\\_faq.pdf](https://www.co-opfs.org/media/248639/androidpay_faq.pdf). [Accessed October 2016].
- [134] Google, "Android Pay," October 2016. [Online]. Available: [https://www.android.com/intl/en\\_uk/pay/#supported-banks](https://www.android.com/intl/en_uk/pay/#supported-banks). [Accessed October 2016].
- [135] Gemalto Mobile Financial Services, "Token Service Provider for Mobile Payments," 6 April 2016. [Online]. Available: [http://www.smartcardalliance.org/secure/events/20160404/PACIFICA-8-10\\_WED\\_315\\_LURIE\\_SCA-Naomi-Lurie.pdf](http://www.smartcardalliance.org/secure/events/20160404/PACIFICA-8-10_WED_315_LURIE_SCA-Naomi-Lurie.pdf). [Accessed October 2016].
- [136] Barclays, "Barclays launches its own contactless payments service for Android phones," 12 May 2016. [Online]. Available: [http://www.newsroom.barclays.com/r/3352/barclays\\_launches\\_its\\_own\\_contactless\\_payments\\_service\\_for](http://www.newsroom.barclays.com/r/3352/barclays_launches_its_own_contactless_payments_service_for). [Accessed December 2016].
- [137] NFC Times, "Barclays: No Plans to support Android Pay or Samsung Pay; Sees Own HCE Apps as only android payments option needed," 20 May 2016. [Online]. Available: <http://nfcetimes.com/news/barclays-no-plans-support-android-pay-or-samsung-pay-sees-own-hce-app-only-android-payments-opt>. [Accessed December 2016].
- [138] M. Clark, "Bell ID gets HCE certification from global payment schemes," 30 September 2015. [Online]. Available: <https://www.nfcworld.com/2015/09/30/338268/bell-id-gets-hce-certification-from-global-payment-schemes/>. [Accessed December 2016].
- [139] ANZ Bank New Zealand Limited, "ANZ Mobile Pay," 2015. [Online]. Available: <http://www.anz.com/personal/ways-bank/mobile-banking/mobile-pay/>. [Accessed October 2016].
- [140] J. C. Mitchell, V. Shmatikov and U. Stern, "Finite-State Analysis of SSL 3.0," in *SSYM'98 Proceedings of the 7th conference on USENIX Security Symposium*, San Antonio, Texas, 1998.
- [141] W. Marrero, E. Clarke and S. Jha, "Model Checking for Security Protocols," Carnegie Mellon University Pittsburgh PA Dept of Computer Science, 1997.
- [142] P. Zimmer, "Fizzim – the Free FSM Design Tool," [Online]. Available: <http://www.fizzim.com/>. [Accessed December 2016].
- [143] William H. Sanders and the Board of Trustees of the University of Illinois, "Möbius Overview," January 2017. [Online]. Available: <https://www.mobius.illinois.edu/>. [Accessed January 2017].
- [144] Altova GmbH, "Altova XML Tools," January 2017. [Online]. Available: [https://www.altova.com/xml\\_tools.html](https://www.altova.com/xml_tools.html). [Accessed January 2017].

- [145] Quantum Leaps, LLC. , “Quantum Leaps, LLC.,” January 2017. [Online]. Available: <http://www.state-machine.com/qm/>. [Accessed January 2017].
- [146] MathWorks, “Simulink,” January 2017. [Online]. Available: <https://uk.mathworks.com/products/simulink.html>. [Accessed January 2017].
- [147] MathWorks, “Simulink Verification and Validation,” January 2017. [Online]. Available: <https://uk.mathworks.com/products/simverification.html>. [Accessed January 2017].
- [148] J. v. d. Breckel, D. A. Ortiz-Yepes, E. Poll and J. d. Ruiters, “EMV in a nutshell,” 2016.
- [149] Visa, “Visa Mobile Contactless Payment Specification (VMCPS),” Visa, 2015.
- [150] VISA, “Transaction Acceptance Device Guide (TADG),” VISA, 2016.
- [151] EMVCo, LLC, Book C-3 Kernel Specification, EMVCo, LLC, February 2016.
- [152] VISA, “Visa Token Service - Introduction for Issuers,” VISA, November 2016.
- [153] Visa, “Visa Developer Centre,” 2016. [Online]. Available: [https://developer.visa.com/products/vts/reference#vts\\_provision\\_token](https://developer.visa.com/products/vts/reference#vts_provision_token). [Accessed December 2016].
- [154] SimplyTapp, “Apple Pay and Android payment eco-systems,” September 2014. [Online]. Available: <http://blog.simplytapp.com/2014/09/apple-pay-and-android-payment-eco.html>. [Accessed January 2017].
- [155] SAMSUNG, “Token Handling by Samsung Pay,” [Online]. Available: <http://developer.samsung.com/tech-insights/pay/token-handling-by-samsung-pay>. [Accessed December 2016].
- [156] Federal Reserve Bank of Atlanta, “PIN authentication versus signature authentication,” 23 January 2012. [Online]. Available: <http://takeonpayments.frbatlanta.org/2012/01/pin-authentication-vs-signature-authentication.html>. [Accessed January 2017].
- [157] Visa, “Visa Token Service Product Overview,” Visa Europe, 2016.
- [158] Android, “Fingerprint HAL,” [Online]. Available: <https://source.android.com/security/authentication/fingerprint-hal.html>. [Accessed February 2017].
- [159] European Union Agency for Network and Information Security (ENISA), “Security of Mobile Payments and Digital Wallets,” ENISA, 2016.
- [160] Creditcall Ltd, “EMV Visa Contactless Transaction Flow,” 2016. [Online]. Available: <https://www.level2kernel.com/emv-visa-contactless-kernel.html>. [Accessed February 2017].
- [161] Google, “Portable radio frequency emitting identifier,” 17 May 1983. [Online]. Available: <https://www.google.com/patents/US4384288>. [Accessed September 2016].
- [162] A. D. White and M. A. Borrett, “Apparatus for bidirectional data and unidirectional power transmission between Master and Slave units using inductive coupling”. European Patent Office Patent WO1997023060, 26 June 1997.
- [163] Mobey forum, “The Host Card Emulation in Payments: Options for Financial Institutions,” Mobey Forum, 2014.
- [164] Nokia, “NFC technology takes its next step with the Nokia 6216 classic,” April 2009. [Online]. Available: <http://company.nokia.com/en/news/press-releases/2009/04/23/nfc-technology-takes-its-next-step-with-the-nokia-6216-classic>. [Accessed September 2016].
- [165] BBC, “Contactless card limit rises to £30 as card use surges,” September 2015. [Online]. Available: <http://www.bbc.com/news/technology-30911111>. [Accessed September 2016].
- [166] The Guardian, “Contactless payments and London travel: your questions answered,” September 2014. [Online]. Available: <https://www.theguardian.com/money/2014/sep/16/contactless-payments-london-travel-questions-answered>. [Accessed September 2016].
- [167] UL Transaction Security, “HCE Security Implications, analyzing the security aspects of HCE,” UL Transaction Security, 2014.
- [168] The Guardian, “Apple Announces Apple Pay,” July 2015. [Online]. Available: <https://www.theguardian.com/technology/2015/jul/14/apple-pay-launches-uk-how-to-use>. [Accessed September 2016].

- [169] M. Brignall, May 2016. [Online]. Available: <http://www.theguardian.com/money/2016/may/12/barclays-contactless-payment-app-android-phones>. [Accessed September 2016].
- [170] R. Boden, "EMVCo to certify mobile devices for NFC and HCE payments," 18 August 2016. [Online]. Available: <http://www.nfcworld.com/2016/08/18/346763/emvco-to-certify-mobile-devices-for-nfc-and-hce-payments/>. [Accessed October 2016].
- [171] I. R. Diego Ortiz-Yepes, "A critical review of the EMV payment tokenisation specification," *Computer Fraud & Security*, October 2014.

## 8 Appendix

### 8.1 Appendix 1 – History of Contactless Payments Timeline

The industry has evolved from contactless EMV card payments, to mobile NFC payments using secure elements to HCE payments.

Month	Year	Event	Details
September	1898	Early development of contactless technology	Early development of contactless technology by Nikola Tesla. He invented a tele-automation system using remote control technology to control a miniature boat at Madison Square Garden in New York City without touching the devices directly [161].
March	1997	Speedpass key fob	Mobil Oil Corp gas stations offered contactless payment devices known as Speedpass. The device is a key fob which is a cryptographically enabled capable of responding to a challenge response protocol, whereby customers in United States could pay for fuel at participating Exxon and Mobil stations [7].
June	1997	NFC first patent	NFC patent awarded to David Andrew White and Adrian Marc Borrett (Patent WO9723060) [162].
February	2003	NFC first prototype	Nordea Bank AB, a financial services group and Luottokunta, specialising in card payment services created a prototype device that was capable of making payments using NFC technology [8].
March	2004	NFC Forum founded in 2004	NFC Forum standards body was founded in 2004 with the aim to advance the use of NFC technology and to create standards in the mobile industry for interoperability from different manufacturers and service providers. Participating members included Nokia, Philips and Sony established the Near Field Communication Forum [8].
January	2007	Nokia 6131 handset - NFC enabled	The Nokia 6131 was the first mobile phone to incorporate NFC. In London from late 2007 heading into 2008, 500 Londoners were given this device to be used for their travelling substituting the Oyster Card. Such phone was also used to purchase items as if the customers had a debit card [8].
August	2007	Tap and Go	Barclaycard introduced the first contactless cards in the UK in 2007 [7].
March	2008	Eat restaurant chain	EAT was the first retailer in the UK to accept contactless payments. Customers were able to pay for transactions of £10 from EAT retailer by tapping MasterCard PayPass or Visa payWave cards against the POS [7].
N/A	2009	Host-NFC chip standardization	NFC Forum decided in 2009 to standardise the Host-NFC chip interface so that device manufacturers could switch suppliers more easily. Before that, those interfaces were proprietary by each manufacturer and therefore different [163]. NFC Forum chose at that time to create its own protocol, NFC Controller Interface (NCI), rather than rely on the Host Controller Interface (HCI) as used in a Single Wire Protocol (SWP) by the European Telecommunications Standards Institute (ETSI) [163].

Month	Year	Event	Details
April	2009	Nokia's first SIM-based NFC device	The first SIM-based NFC device issued by Nokia - 6212 Classic [164].
January	2011	First Contactless mobile phone payments	In Europe, Barclaycard teamed up with Orange to launch Europe's first contactless mobile phone payments. The system went live in the second quarter of 2011. A limit up to a value of £15 was imposed on purchases when using this type of payment [7].
September	2011	Google Wallet	Contactless payments using mobile NFC devices started with Google Wallet in 2011. Google demonstrated the Google Wallet application at a press conference in May 26, 2011. This system was initially implemented using an SE-based model. Customers could make payments by simply tapping the phone on any PayPass-enabled terminal at checkout. The app was released in the United States on September 19, 2011. Google Wallet supports payment, loyalty, money transfer, offers, and online order tracking. [10].
June	2012	Contactless payment limit in UK raised to £20	Contactless payment limit in UK increased to £20 [165]. No pin or signature to authorise payment is required within this limit band.
August	2012	Simply Tapp using Secure Element	SimplyTapp and Bankinter S.A became the very early pioneers in software secure element (HCE) to introduce this technology to pay with your phone. SimplyTapp had the first trials in August 2012 whilst Bankinter S.A in April 2013 [163].  At this stage, adoption still was slow due to limited merchant acceptance and mobile NFC device availability.
December	2012	London Underground introduces contactless payments	Contactless payments introduced on London transport such as tube or busses [166].
October	2013	Google released First HCE architecture in Android 4.4 KitKat.	Google released HCE architecture in Android 4.4 KitKat for secure NFC-based transactions [167].
February	2014	Visa payWave and MasterCard	Visa and MasterCard announced that they will be supporting HCE for secure NFC payments transactions [11] [12].
April	2014	Apple Pay	Apple Pay was launched by Apple Inc. in April 2014 in the United States and in October in the United Kingdom. The implementation is based using traditional device-based Secure Element. It does not use HCE technology [168].  The launch of Apple Pay in October 2014 gave mobile NFC reignited interest in contactless payment.
April	2014	Google stopped supporting the Secure element Model	In April 14, 2014, Google stopped supporting the SE-based version and started supporting host card emulation (HCE) [9].
March	2015	Android Pay	Google announced Android Pay, a platform supporting HCE, tokenized card numbers, and NFC [10].



Month	Year	Event	Details
March	2015	Samsung Pay	Samsung Pay was introduced in the United States and Korea [10]. Customers in United States could make contactless payments using the same traditional mag stripe POS devices without NFC [10] though the use of Magnetic Secure Transmission (MST).
September	2015	Contactless payment limit in UK raised to £30	Contactless payment limit in UK rose to £30 [165].
May	2016	Barclays to launch contactless payment app for Android phones	First private Android contactless mobile payment platform launched in the UK by Barclays. Barclays will have their own app service. This will allow customers to use their mobile device to “wave and pay” in environments such as shops, restaurants and across the London transport network. Such service will be in direct competition with Google’s Android Pay service [169].
June	2016	Microsoft Wallet	Introduction of Microsoft Wallet in U.S.A operable with Lumia 950, 950 XL and 650 on Windows 10 operating system [13].
August	2016	EMVCo to certify mobile devices for NFC and HCE payments	EMVCo has introduced a formal industry testing (i.e. EMVCo’s Level 1 specification) and certification process for contactless mobile payment device both secure element (SE) and host card emulation (HCE) based services. The specification defines the physical characteristics, radio frequency interface and transmission protocol between credit and debit cards and a payment terminal [170].