# ON THE SECURITY AND PERFORMANCE OF MOBILE DEVICES IN TRANSPORT TICKETING

Submitted by
Assad Abdullahi Umar

Royal Holloway, University of London

*Thesis submitted to*
*The University of London*
*for the degree of*
*Doctor of Philosophy*
*2017.*

# Declaration of Authorship

I Assad Abdullahi Umar hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed: _____ Date: _____

# Statement of Sponsorship

The work presented in this thesis is the result of a PhD sponsored by Transport for London (TfL).

# Abstract

There have been significant developments in the field of transport ticketing. Major transport operators are transitioning from closed-loop, proprietary systems to open systems that utilise the global payment infrastructure for ticketing via smart cards and, increasingly, smartphones.

Modern smartphones support Near Field Communication (NFC) which can be used to emulate contactless smart card tickets. NFC transactions are quick, making them a viable technology for use in transport ticketing where speed is very critical. NFC transactions initially required a Secure Element (SE) for security reasons. However, commercial constraints and restrictive security practices relating to the SE have paved the way for Host Card Emulation (HCE). HCE facilitates NFC transactions without requiring an SE; this provides a simpler and more flexible ecosystem, at the expense of security.

This thesis investigates the impact of the aforementioned developments on the security and performance of mobile devices in ticketing. A comparative analysis of various security mechanisms that have been put forward as options to mitigate the security risks of HCE is carried out, and their suitability for ticketing is determined. A novel ticketing protocol based on Linkable Digital Signatures is proposed to solve the problem of blacklisting (the barring of invalid tickets/smartphones) in tokenised payments. A novel tokenisation framework based on Format Preserving Encryption (FPE) algorithms and Trusted Execution Environments (TEE) for secure token generation on the user's device has been proposed. All proposals were implemented on mobile devices to test the performance for efficiency.

The work conducted in this thesis shows that mobile devices, and particularly HCE, offer several benefits in ticketing, however, a new approach to

security is required. It also shows that despite the clear advantages of adopting open payments, careful considerations must be taken for it to be successful in ticketing.

# Acknowledgements

Firstly, I would like to thank God, the most beneficent and the most merciful, for giving me the strength, courage and ability to get through this journey.

I would like to thank my parents whose support, care, love and encouragement gave me the foundations necessary to embark on the PhD journey, and to successfully complete it. I love you!

I would like to thank my supervisor Professor Keith Mayes, Director of the Smart Card Centre (SCC) and Head of the School of Mathematics and Information Security at Royal Holloway,for his encouragement, support, patience, understanding and expert knowledge during the course of the PhD.

I would also like to thank Mrs. Sheila Cobourne for her support, guidance and for being a reliable source of motivation. Thanks to Professor Konstantinos Markantonakis for his encouragements and insights. I would also like to say a big thank you to Dr Raja Naeem Akram, Sarah Abu-Ghazalah and Iakovos Gurulian for all their help and criticisms that have helped in making this thesis a possibility. To the other members of the SCC: Mehari Msgna, Danushka Jayasinghe, Benoit Ducray, Hafizah Mansor and Lazaros Kyrillidis, thank you for your support, kindness and motivation. In all of you, I have made friends that I will cherish for life.

To my uncles, aunties, cousins and the entire extended family, thank you all for your encouragements, push and financial support. Thank you for your understanding when times were tough. One love!

A big thank you to Mr Vassilis Korkas, who painstakingly proofread the thesis, it would not have been possible without your help.

To the love of my life – Naima Ali, I cannot thank you enough for your encouragement, understanding, love and affection. Your contributions to this endeavour are dearly appreciated.

Last but not the least, I gratefully acknowledge the funding and support received towards my PhD from Transport for London.

# Contents

# List of Tables

# List of Figures

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Historically, public transport has gone through significant transformation with respect to ticketing and fare collection, from the very early days of cash and paper tickets, to transport ticketing systems with a degree of automation that accept *smart tickets*. A ticket is considered to be smart if it has some access-controlled memory where the entitlement to travel can be stored, read and/or modified electronically by an external terminal *(smart ticketing)*. Traditionally smart tickets took the form of a microchip embedded in a contactless smartcard, issued by the transport operator for use in proprietary systems. The state of the art today is the use of contactless bank cards and Near Field Communication (NFC)-enabled mobile devices. NFC is a short-range communication technology that is capable of emulating a smartcard on a mobile device, and facilitates communication between a device and an external contactless smartcard reader, which in this context is a terminal[1] at a train station or on a bus. What makes NFC even more suitable is its compatibility with existing ticketing infrastructure that already accepts contactless smartcards because they both conform to the ISO/IEC 14443 standards [5]. Crucially,

[1]For this point, the term terminal is used to collectively refer to a contactless reader, the software that manages the contactless transaction, and the turnstile that enforces physical access control according to the decision taken

transport ticketing applications are speed-critical, which makes NFC an attractive and suitable technology, because NFC transactions are very fast.

The integration of NFC into mobile devices has led to a paradigm shift in the way traditional payments and ticketing transactions are carried out. In comparison to smart-cards, NFC-enabled mobile devices present new possibilities, as well as constraints, for the transport ticketing ecosystem. Mobile devices provide a richer User Interface and User Experience (UI & UX respectively), in addition to higher processing power. They also help users by consolidating their payment instruments into a single device, rather than having multiple cards for similar purposes. Additionally, the transport operators make cost savings in terms of issuing and managing cards and card systems. However, the conventional security architecture of the mobile device presents its own constraints, which are more business than technical ones. These constraints, along with the promise shown by NFC technology, have led to radical changes in terms of the NFC security architecture, which presents new research frontiers that are yet to be explored.

From a research perspective, much work has been done in the area of transport ticketing security, with a major focus on user privacy. While privacy is undoubtedly an important aspect of ticketing, fundamental changes to the ticketing architecture in practice and the evolution of mobile technologies present research gaps, which the work in this thesis sets out to bridge. The beginning of the research presented here coincides with the onset of two new developments in relation to transport ticketing which have subsequently shaped its direction.

In recent developments, transport operators are moving away from closed-loop ticketing systems that only accept tickets issued by the transport operator, and thus can only be used for ticketing and related services, to open

ticketing systems. Open ticketing systems use the traditional payment infrastructure; they accept regular contactless bank cards and other mobile technology to access the transport service. This effectively shifts the business of payments away from the transport operator, and payments are handled by the payment industry in the same way as other payments in retail trading.

There has also been a radical change in the way transactions using NFC devices are carried out. NFC transactions using card emulation were expected to make use of a hardware Secure Element (SE) in the mobile device to house sensitive applications and related data. In a transaction using an SE card emulation, messages are routed to the SE directly, without being visible to the Operating System (OS). This concept achieves its purpose in terms of security because of the hardware-backed isolation and tamper-resistance provided by the SE. However, restrictive practices (motivated by commercial as well a security interests) prevented developers form having access to the SE; which frustrated service development. These factors led to a slow adoption of NFC in many sectors, including transport. To address this problem, a new way of performing card emulation that bypasses the SE emerged; this is referred to as Host Card Emulation (HCE). HCE lets an application on the host OS communicate directly with an external NFC terminal. This allows for a simpler and more flexible ecosystem at the expense of security. HCE offers less attack-resistance than SE card emulation since it relies on the software mechanisms of the device for security.

One of the known security mechanisms in mobile payment systems is to store temporary payment credentials in the smartphone, a technique known as *tokenisation*. However, this can be a problem for transport ticketing security processes, which rely on fixed and permanent payment credentials for blacklisting.

The work presented in this thesis considers different aspects of transport ticketing, and investigates the impact of the aforementioned developments on

ticketing security and performance.

## 1.2 Research Questions

1. How does the shift from SE-based to HCE-based NFC transactions impact the security and performance of transport ticketing systems?

2. What is the impact of tokenisation —a payment security mechanism— on open ticketing systems?

3. Can a new privacy-preserving protocol be proposed that will allow user blacklisting without requiring a unique identifier?

4. To what extent can advances in Trusted Execution Environments (TEE) for mobile devices be used to enhance the security of transport ticketing processes?

## 1.3 Contributions

The main contributions of the thesis are as follows:

1. The state of the art in ticketing and an analysis of current trends relating to ticketing are presented.

2. An assessment of the security and performance of NFC devices in transport ticketing is conducted: A variable performance behaviour of HCE applications is revealed, and its impact on security is considered.

3. Analysis of tokenisation, and how it calls into question blacklisting, which is an important fraud and security control process in transport ticketing systems.

4. A proposal for a novel, privacy-preserving transport ticketing scheme using linkable group digital signatures is given, solving the problem of blacklisting in tokenised transactions.

5. A proposal is given for a novel tokenisation framework that relies on a TEE and Format Preserving Encryption (FPE) algorithms, to generate on-device tokens.

## 1.4 Organisation of the Thesis

The remainder of the thesis is structured as follows:

**Chapter 2** sets the scene and tone of the thesis. The evolution of ticketing fare media is presented. A discussion of notable ticketing schemes around the world is then provided. Different ticketing concepts are described, providing an essential background to the work carried out in the thesis.

**Chapter 3** opens with a discussion of NFC devices and their applicability to ticketing. We look at the security aspects of NFC on mobile devices and their impact on ticketing. An analysis of various mechanisms proposed to mitigate the security risks introduced by Host Card Emulation (HCE) and the suitability of the mechanisms for ticketing is also presented. This chapter concludes with a summary of the trends in the payment industry, and what they mean for transport ticketing.

A comparison of the performance of similar implementations of an SE-based application, and a HCE-based Android application is carried out in **Chapter 4**. We identify a specific behaviour of applications in HCE, which calls into question the performance of mobile devices in transport ticketing. Regardless of the superiority of HCEs in terms of processing power as compared to an SE, we show how the behaviour affects security mechanisms and assumptions used in smartcards, and by extension SEs, which will not hold in mobile transactions.

**Chapter 5** focuses on the problem of blacklisting tokenised payments. It shows how tokenisation—which is a security mechanism proposed for HCE applications—contradicts blacklisting, which is an important function of transport ticketing systems. A novel solution to the problem using linkable group digital signatures is presented. Practical implementation and testing of the solution shows mobile devices have sufficient processing power to efficiently handle complex cryptographic operations, allowing for the realisation of a

privacy-preserving blacklisting solution.

In **Chapter 6**, the use of TEEs as a security mechanism for HCE applications is analysed. I highlight a possible problem with the closed nature of TEEs in the ecosystem. Different ownership models are considered with respect to their centricity, and a recommendation is made based on their suitability for adoption in the transport ticketing domain.

**Chapter 7** presents a novel tokenisation framework for the on-device generation of tokens. To achieve this, a FPE algorithm is implemented as a mobile application.

**Chapter 8** presents the major conclusions of the thesis in relation to the research questions. The future research directions that could be explored are also outlined,

## 1.5 List of Publications

1. Assad Umar, Keith Mayes, and Konstantinos Markantonakis. Performance Variation in Host-based Card Emulation Compared to a Hardware Security Element. *In First Conference on Mobile and Secure Services (Mobisecserv), pages 1–6.* IEEE. Gainesville, Florida, USA. 2015. (Chapter 4)

2. Assad Umar, Iakovos Gurulian, Keith Mayes, and Konstantinos Markantonakis. Tokenisation Blacklisting Using Linkable Group Signatures. *In the 12th International Conference on Security and Privacy in Communication Networks pages 182–198.* Guangzhou, People's Republic of China. Springer International Publishing, Cham, 2017. (Chapter 5)

3. Assad Umar, Raja Naeem Akram, Keith Mayes, and Konstantinos Markantonakis. Ecosystems of Trusted Execution Environment on Smartphones - a Potentially Bumpy Road. *In 2017 Third International Conference on Mobile and Secure Services (MobiSecServ), pages 1–8.* Miami, Florida, USA. Feb 2017. (Chapter 6)

4. Assad Umar and Keith Mayes. Trusted Execution Environment and Host Card Emulation. *In smartcards, Tokens, Security and Applications, pages 497–519.* Springer International Publishing, 2017. (Chapter 3 and Chapter 6)

# Chapter 2

# Background I: Ticketing Systems

## 2.1 Evolution of Ticketing Fare Media

The very early days of ticketing involved the use of cash in exchange for a paper ticket or a physical token that acts as the right to travel. However, the use of cash has many challenges that directly affects the efficiency of ticketing systems. Cash is very expensive and inefficient to handle for the transport operators. It also makes the boarding of transport vehicles time-consuming and generally impractical with the growing number of public transport users. The aforementioned problems with cash led to the introduction of the first automated ticketing systems.

Automated ticketing systems have the ability to validate tickets and apply logic as to whether the user has the right to travel. This decision is enforced using gates or turnstiles to allow or deny users access to the transport service. Earlier automated systems accepted cards and paper tickets with a Magnetic stripe (Magstripe) [6] that encodes the information. This significantly solves the problems introduced by the circulation of cash in the system and also makes boarding transport vehicles faster and more efficient. The low cost of production, and durability of these types of tickets makes them ideal for both single-journey and multiple-journey tickets. Nevertheless, magstripe tickets

have their own downsides: they have a very limited storage capacity of about 24 bytes, which is a constraint to the complexity and number of tickets it can hold. The level of security that can be achieved in magstripe is very low because of their inability to carry out any processing. Consequently magstripe tickets are prone to cloning. These issues with magstripe tickets, advances in smartcard technology, and reduced cost of silicon established contactless smartcards based on the ISO/IEC 14443 standard as an ideal option for ticketing.

Contactless smartcards have the ability to carry out processing, and have bigger storage capacities. This means they can support higher levels of security than the magstripe tickets. They are also more durable and are therefore suitable for long validity tickets (season tickets). From the transport operators perspective, the use of smartcards enhances travel data collection and mining, which can be used for a smarter resources allocation and a better service delivery [7]. As alluded to in Chapter 1, NFC devices are capable of emulating smartcards and communicating with an external terminal. This offers a natural progression from smartcards, but with an added advantage of a higher processing power, and a UI. More on NFC is covered in Chapter 3.

## 2.2 Description of Ticketing Systems

The specifics of each ticketing system will vary according to particular use-cases, assumptions, technological choices and business agreements in place. Nevertheless a general description of ticketing systems is given in this section. A generic architecture, the participants, and phases that make up a ticketing system are described. A classification of ticketing systems based on their payments model, as well as business logic is also presented.

### 2.2.1 General Architecture

The generic architecture of ticketing systems consists of various components at four levels:

**Level 0 - Fare Media**

The device used to deliver the ticket payload to the validation terminals. These include contactless smartcards and NFC-enabled mobile devices.

**Level 1 - Validation Terminals**

The term *terminal* in ticketing may be used to refer to different components depending on the implementation. For example, it can mean a mobile device used by ticket inspectors to scan tickets during a journey. In this context, validation terminal refers to a ticketing device capable of reading, writing or modifying a ticket. In gated stations, the terminal also includes a physical mechanism –typically a turnstile– to enforce access control according to the decision-taking after interacting with the ticket. The terminals are connected to the back-office systems for periodic reporting and status updates.

**Level 2 - Back Office Systems**

Back office systems are managed and maintained by the transport operator. They are used for the general management and administration of ticketing systems. Functions of the back office include: fare and ticket management, user account management, report management and handling of terminals updates.

In closed-loop ticketing systems (see Section 2.3.1) with a single transport operator, the back office also carries out financial clearing and settlement functions.

**Level 3 - Central Clearing and Settlement Systems**

Most real-life implementations will involve different operators controlling different parts of a geographic location. The central clearing and settlement offices are used to provide financial functions including: apportionment of revenue among the relevant transport operators, central reporting and updating services, and general ticket and user administration.

### 2.2.2 Participants

The participants in a ticketing system include:

1. Users: A user is the entity that makes use of the transport service. The user either purchases the ticket prior to travel, and verifies its possession at the point of travel, or the user is reliably associated with the usage of the service, for post billing and payment.

2. Transport Operator: The transport operator provides the transport service. It may consist of different sub-entities providing services such as ticket issuing and accounting services. It manages the ticketing infrastructure such as payment terminals and turnstiles and puts in place controls to ensure system security and functionality.

3. Payment Service Provider: With the advent of open ticketing systems, the payment ecosystem has become directly involved in ticketing. The term Payment Service Provider is used here to collectively refer to the payments ecosystem including banks and the payment networks (see Section 2.3.2).

The aforementioned participants represent the minimum participants according to most proposals, this is however not exclusive. For example the work

in [8] includes a Certifying Authority (CA) which provisions credentials to the user's device and performs accounting services. Chaumette et al. considered a ticketing architecture in [9] that requires a Trusted Service Manager (TSM) to personalise, load and manage tickets in the Subscriber Identity Module (SIM) of the user's device.

### 2.2.3 Phases

Ticketing is made up of different independent but related phases as follows:

1. Set-up Phase: Prior to travel, there is usually a step where the user registers to the system, pays for the service if the scheme is prepaid and is issued with the tickets. The set-up phase is also when the user downloads the ticketing application, or is been issued a smart-card in the case of smart-card based ticketing schemes. For cryptographic ticketing solutions, this is the stage where cryptographic handshake, such as key exchange, and other parameter settings take place. There are also instances where the transport operator may need to personalise the profile of some users, for example, to offer special discounts to senior citizens.

2. Purchase Phase: In this phase, users select and pay for their tickets. For prepaid travel, users top up their devices with the desired amount of money. More on the payment models on ticketing systems is presented in Section 2.3.

3. Provisioning Phase: This is the phase where the tickets are fulfilled[1] to the users' device, following a successful payment authorisation of tickets in the previous phase. The tickets are ready for use at the end of this phase. Some proposals [10, 11] merge the purchase and provisioning phases into a single phase.

[1]Ticket fulfilment refers to the delivery of tickets to the user or user's device.

4. Validation Phase: This is the phase where users travel with their purchased tickets or in the case of Pay-As-You-Go (PAYG), their topped-up devices. The ticketing system validates that the ticket presented to it is valid and has the right to perform the intended journey.

5. Inspection Phase: This phase is usually carried out at random to verify that user's have a valid right to travel, which could be a pre-purchased ticket or an evidence to show they properly tapped their device prior to travel. Inspection is carried out by a personnel either with a Revenue Inspection Device (RID), which interrogates the fare media or visually to check that a valid ticket is held by the user (either physically or within a fare media). Researchers have shown visual inspection of tickets held within mobile devices to be vulnerable [12], unless certain measures – such as a dynamic watermarking are deployed.

6. Accounting Phase: This phase happens at the back-office after journeys have been made. For account-based implementations such as the one proposed in Chapter 5, this is the phase where journey construction[2] is carried out to determine the fare to charge the user. For ticketing systems that include multiple transport operators, the apportionment of the revenue between them occurs at this stage [13].

7. Blacklisting Phase: This is a fraud and control measure, it only becomes necessary when there is a need to deter a particular user or user's device from travel. Blacklisting may occur due to a number of reasons including lost device, overspending, lack of funds, or in the case of system compromise.

[2]Journey Construction is the process of aggregating the entry and exit points of a user on a transport network, to determine the fare to be charged at the back-office.

## 2.3 Classification of Ticketing Systems

Ticketing systems can broadly be classified from two perspectives: the level at which 'business logic' is applied, and according to the 'payments model' they conform to. Business logic here refers to the encoding of real-world business rules which are used to make a decision on how a user's ticket should be treated and on how fares should be determined. The business logic of ticketing systems can be *card-based*[3] or *account-based*.

According to their payment model, ticketing systems can also be classified into *closed-loop Systems* and *open-loop Systems*. The payment model represents the type of infrastructure they rely on to clear and settle funds generated through the collection of fares. These categories of ticketing systems are further explained below:

### 2.3.1 Card-Based Ticketing Systems (CBT)

This category of ticketing systems apply the business logic prior to usage of the system, i.e. the system validates that the user has the right to travel at the front-end. Users are required to either purchase a ticket prior to travel, or top up funds in the case of Pay-As-You-Go (PAYG) travel. The pre-purchased ticket or top up value is stored on the device, and prior to travel, the terminal checks that the ticket is valid or that the existing value on the device is enough to pay for the fare.

CBT systems have drawbacks that is making them less and less favourable for most transport operators. Any change of business logic, such as fare prices and discounts, will have to be propagated to all the terminals. This is an expensive and often a daunting process. In addition, the loss of a user's fare media usually leads to the loss of the balance stored on the card. Account-Based ticketing, presented below, addresses some of these challenges.

[3]The term card-based has evolved to include mobile-based solutions, but the industry has retained the term.

### 2.3.2 Account-Based Ticketing Systems (ABT)

In the case of ABT systems, the logic is applied at the back-office level, and the terminal front-end only verifies the user is legitimate, and acknowledges usage of the system in a secure and reliable way. The evidence of usage captured is widely referred to as a *tap*, and typically includes information such as time and location. These taps are used at the back-office to determine the right fare to charge the user.

One of the main advantages of ABT is an easier and more efficient way of changing the business logic. Changes only need to be applied at the back-office where the accounts are held, rather than updating all the terminals in the field. This also enables the transport operator to give more tailored services to the users. Moving the logic to the back-office also opens up the possibility of using different 'tokens' aside from the transport-specific fare media, to reliably identify users since all that is needed is to reliably establish their use of the transport network.

### 2.3.3 Closed-Loop Ticketing

Closed-loop have for a long time been the *de facto* systems for ticketing. The term Closed-loop here refers to the fact that the fare media is only valid for use within a specific ticketing scheme, which is typically the scheme managed by the transport operator. This means that the fare media itself, tickets and the value they hold cannot be used to pay for any other services - not even for travel on a different transport operators scheme. While these systems offer advantages over the paper-based ticketing, their proprietary nature introduces some disadvantages;

#### 2.3.3.1 Disadvantages of Closed-Loop Ticketing Systems

1. High Costs of Infrastructure: There is a significant cost incurred by the transport operator for proprietary infrastructure both in terms of capital

and operational expenditure upfront, and for ongoing maintenance. The cost of issuing and managing the fare media is also significant, especially for smartcard based systems.

2. High Cost of Fare Collection: The cost of fare collection is can also be significant since the transport operator is especially performing its own financial services. It is widely believed that the cost of managing these type of system is between 10% - 20% of fares collected on the system [14].

3. Lack of Interoperability: Due to their proprietary nature, these systems are not compatible with each other. This also leads to vendor lock-in, and switching to other vendors is expensive and complex because each vendor has its own proprietary solution which must be implemented from scratch.

4. Inconvenience for the Users: Users spend time queuing to top up their media with value spendable on the system. The lack of interoperability across these systems means that as they travel across different locations, they will have to carry different ticketing media for the respective operators.

### 2.3.4 Open Ticketing Systems

The problems associated with closed-loop ticketing, and the high market penetration of contactless bank cards makes a strong case for open ticketing systems. According to the UK Cards Association (UKCA), there are about 108 million contactless bank cards currently in issue in the UK alone [15]. Open ticketing systems utilise the globally accepted and standardised payment infrastructure, and they use contactless payment devices. Contactless bank cards and NFC-enabled mobile devices are suitable candidates because transactions are quick and do not even require card-holder verification (see Section 3.5.2).

This takes away accounting services from the transport operator, and is outsourced to the financial institutions, thereby allowing it to concentrate on its core business of providing transport services.

Nevertheless, due to the requirements of transport ticketing (see Section 5.5.1), some new concepts had to be introduced in payments, to accommodate open ticketing systems. Of significance are the concepts of *delayed authorisation* and *first time travel risk*. In delayed authorisation, instead of requesting authorisation for every transaction as usual, the transport operator only acknowledges that the user's device has been seen at various points on the transport network. The evidence of the device's usage on the transport network is referred to as a 'tap', these taps are sent to the back-office. At the back-office, all taps by the same user are aggregated at the end of the day, and only then does the transport operator request authorisation of funds. The apparent risk here is that a dishonest user can travel with no funds in their account since authorisation is not done at the time of travel. We refer to this as the first time travel risk. Currently, this risk is negotiated and accepted between the transport operator and the bank issuers [16].

The advantages of open ticketing system are given below:

#### 2.3.4.1 Advantages of Open Ticketing Systems

1. Cost Savings for Transport Operators: The Transport operator reduces expenses in many instances. Firstly, there is no longer cost for issuing and managing its own fare media since payment media issued by the banks will be used. Costs are also saved due to reduction in customer support since the operators no longer manage the payment media, and can then focus on providing better ticketing-related support. By leveraging globally standardised and accepted infrastructure, the high costs of certifying proprietary infrastructure are eliminated.

2. Ease of Deployment: Open systems are easier and faster to deploy since

there will be no need to build one from scratch, leveraging on standards makes deployment more straight forward.

3. Forms Basis for Interoperability: Open systems make interoperability between different transport operators easier. It reduces the problem of interoperability from the lack of technological compatibility, to a case of business decision, which are always easier to get around.

4. More Established Security Mechanisms: Open systems explore the security benefits provided by the well-established payment infrastructure. The infrastructure also undergoes public scrutiny since they are not based on proprietary solutions. Previous attacks on Mifare classic family of smartcards [17, 18] show how the proprietary nature of systems can affect security negatively.

5. Increased User Convenience: Users enjoy the convenience of not carrying another payment medium just for ticketing. Also, users are not required to queue to top up or buy tickets, this saves time and significantly improves user experience. User convenience leads to increased ridership, this is evident from TfL's daily usage statistics since it launched open ticketing (see Fig. 2.1).

#### 2.3.4.2 Disadvantages of Open Ticketing Systems

1. Terminal Certifications: Terminals handling payment data are required to undergo EMV and Payment Card Industry (PCI) certifications [19–21] before they are deployed in the field. This is to ensure their ability to process data correctly and feed it into the payment network without integration problems. EMV certifications are known to be a daunting process for any organisation to embark on, especially for the first time [22].

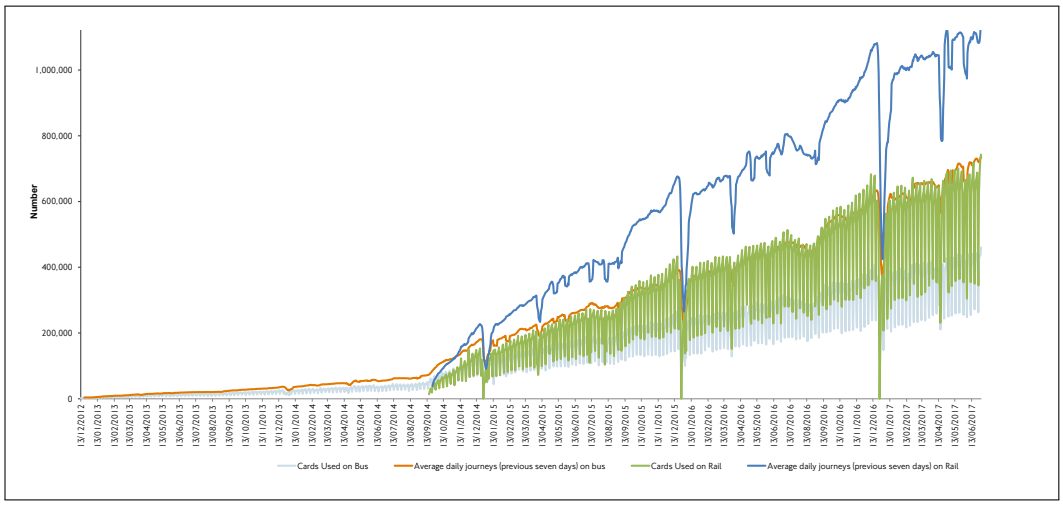


Figure 2.1: Use of Contactless Payment Cards on Bus and Rail [1]

2. Initial Capital Expenditure: As is the case with any investment, there is usually a high initial cost incurred. Open ticketing is not an exception in this aspect. The cost of upgrading the terminals and certifying them is significant. Table 2.1 shows examples of the cost of open ticketing implementations around the world. It is worth noting, however, that the costs vary, depending how mature the technological infrastructure is, before implementing open ticketing. If a closed-loop system is already in place, then savings can be made on the terminals and communication device for example.

## 2.4 Ticketing System Requirements

Requirements differ across different ticketing system models, for example, open ticketing systems typically have fewer requirements than close ticketing systems, and even less for ABT systems. This is because in open payments, ticketing processes rely on some of the provisions of the payment instrument itself, for example, the transaction between the device and the contactless reader are governed by the same standards (EMV) as that of regular payments, say in retail for example. Nevertheless, the requirements of ticketing systems can largely be divided into three categories; security, functional, and privacy requirements. Functional requirements are those requirements that do not necessarily affect the security of the system, but nonetheless affects the efficiency, performance and procedural aspects of the system. The requirements presented in the following sections do not represent the requirements any ticketing solution. Rather, a generic overview of the major ticketing requirements found in existing literature is provided.

### 2.4.1 Security Requirements

1. Integrity: Tickets or ticketing credentials should not be modified without being detected. All participants involved in the ticketing ecosystem should be able to verify that the tickets or related credential is accurate and complete since it was generated by its legitimate source.

   In open systems, the integrity checks provided by EMV can be relied upon. Specifically, the *data authentication* phase provides assurances that the data in the payment device hasn't been modified. See Section 3.5.1.

2. Authentication: No unauthorised user or unauthorised ticketing credential should have access to the transport system. Arguably this is the most important requirement from the transport operator's perspective

because it is directly linked to revenue generation and it could lead to potential avenues for Denial-of-Service (DoS) attacks.

3. Non-repudiation: Any participant involved in any ticketing transaction, cannot deny involvement at a later stage. This requirement applies to different aspects. For pre-paid tickets, the issuer should not be able to deny generating tickets, and users should not be able to maliciously deny using a ticket. This requirement is closely linked with non-overspending explained below.

   For account-based systems, this requirement becomes even more necessary since establishment of fares relies on the evidence that the payment device interacted with the ticketing system at various points.

4. Non-overspending: A user should only be able to use tickets as stipulated in the contract with the transport operator. Tickets should be used within their validity as agreed on, or paid for by the user. In this context validity may include attributes such as distance, time, and geographical boundaries or a combination of these attributes. This requirement is also referred to as *unforgeability* in some proposals [23, 24]. For PAYG ticketing, users should not be able to access the transport service without the minimal necessary value in the payment device. It is important to note that for closed-loop, card-based systems, overspending is checked for prior to travel (verification phase). And if detected, travel is denied.

   In account-based ticketing, however, overspending can only be detected at a later stage when the user's travels are aggregated since all users with a valid payment device are allowed at least their 'first travel' *(first time travel risk)* (See Section 5.5). In such cases, blacklisting is relied upon to detect further usage by the same user or payment device, depending on if the ticket is transferable.

5. Blacklistability: It should be possible to detect and deny dishonest users from further travel or interaction with the ticketing system. The blacklisting requirements is the transport operators first line of defence against compromised user devices or dishonest users. In this context, a dishonest user may be a user in possession of a stolen device, a compromised device or a case of over-spending by a user. In addition, in account-based ticketing, the importance of blacklisting becomes even more significant because there is a direct monetary value involved. Blacklisting is necessary to deter users from taking advantage of the 'first time travel risk'.

6. Exculpability: It should not be possible for a transport operator to falsely accuse the user of overspending. The exculpability requirement has its origins in group digital signatures [25] and it was informally defined by Bellare et al. [26] as: Neither a group manager or any of the group members can sign on behalf of any of the group members. Exculpability in the context of ticketing therefore provides assurances that the transport operator will not falsely accuse a user of over spending [27]. Exculpability also encompasses the ability of the user to prove his tickets have been validated prior to usage [28].

7. Anti-Pass-back: Pass-back in ticketing is a fraudulent act where a user enters a transport system with a valid ticket, and then immediately passes the device back to a second user who also taps and get validated by the system. From a review of some of the existing ticketing proposals [29–31] and discussions with major stakeholders in the industry, two general ways for detecting pass-back have been identified. Anti pass-back can be at the 'ticket-level' where the reader writes back data to the ticket –a time-stamp for example– so that pass-back can be identified. Anti pass-back can also be detected at the 'station-level', in which case the terminals keep a record of all tickets they have validated within a

specific amount of time. Station-level anti pass-back is especially useful for ticketing systems that cannot modify the ticket on the fly, such as in barcode ticketing.

### 2.4.2 Functional Requirements

1. Efficiency: In the context of ticketing, efficiency is the level of user throughput by the terminal during ticket validation. This is directly related to the transaction time of the validation protocol, including any processing delays and also taking into account delays on the physical layer. The requirement is for the transaction to be completed in 300 - 500 milliseconds [32, 33]. Maintaining a high-level of user throughput is important because it affects user satisfaction and congestion may lead to health and safety hazards. The payment industry also requires contactless payments to be less the 500 milliseconds [34–36].

2. Offline Verification: The transaction between a device and a terminal should be fully offline. It should be possible to validate if the user is allowed to travel including a blacklist check offline. This is because it is difficult to guarantee connectivity in some cases such as underground train stations, or mobile vehicles (Buses). Connecting to a back-end also introduces latency to the transaction speed, which may be too costly for the overall efficiency of the system. The offline verification requirement is also referred to as versatility by some authors [24].

### 2.4.3 Privacy Requirements

It is important for any ticketing system to protect the privacy of users. Examples of threat to privacy include:

1. Unauthorised User Identification: Either through direct personal identifiers, or indirect identifiers such as user journey data.

2. User Profiling: It should not be possible to track the journeys of users, or to identify which user is making what journey.

The threats to privacy give rise to the following requirements, also referred to as the *privacy protection goals* [37]:

1. Anonymity: Users should be able to buy and use their tickets without disclosing their identity. This requires the identity of users to be preserved throughout the ticketing phases. However, according to the payment model used, this might be a difficult requirement to achieve because users' could be identified at the point of ticket purchase [38].

2. Unlinkability: It should not be possible to link multiple journeys made by a user. The unlinkability requirement is closely related to anonymity, but it goes a step further. There is a possibility of tracing the journeys of users even when they are anonymous. This could allow an unauthorised entity to build profiles of users.

| | | |
|---|---|---|
| Chicago Transit Authority (CTA) | Initially $454 million awarded to Cubic (increased to $519 million due to add-ons) to transition the Ventra system to open ticketing (2011) | 12-year contract to provide terminals, vending machines, and communication devices. Also includes upgrades on about 1800 buses, and over 770 rail gates. |
| Washington Metropolitan Area Transit Authority (WMATA) | Accenture was awarded the contract worth $184 million to transition open ticketing (2014) | 5-year to include 1000 terminals for rail, 450 ticket vending machines, 1500 bus terminals and 160 terminals for parking. |
| Singapore's Land & Transport Authority (LTA) | Awarded a $1.9 million contract to Orange Business Services for open ticketing. | To include a back-office, management services, and payment gateway. |
| South-eastern Pennsylvania Transportation Authority (SEPTA) | Awarded a $129.5 million contract to ACS Transport Solutions for open ticketing. (2011) | SEPTA have the option of buying $83 million worth of additional services like fare collection and data management, after system installation. |
| Budapesti Közlekedési Központ (BKK) Centre for Budapest Transport | Contract awarded to Scheidt & Bachman for an open ticketing system. Worth 91 million pounds over 5 years. (2014) | To include: Back office, 2100 terminals on buses and 120 terminals for rail.<br>Also includes 700 portable hand-held terminals |
| Dallas Area Rapid Transit (DART) | Awarded a $30 million contract to VIX Technologies for an account-based, open ticketing system. (2015) | VIX to provide "Easy and Open", its in-house developed open ticketing solution. |
| Transport for New South Wales (TfNSW) | Awarded a $7.6 million contract to Cubic Corporation to trial an open ticketing solution. (2016) | Trials scheduled to begin in 2017 |

Table 2.1: Trends in Open Ticketing

## 2.5 Notable Ticketing System Implementations in the UK

There are numerous examples of ticketing system implementations around the world. In this section, a description of the two predominant ticketing solutions in the UK is given. A highlight of trends in ticketing from transport operators is given in Table 2.1.

### 2.5.1 Integrated Transport Smartcard Organisation (ITSO)

ITSO is a membership-based organisation made up of public authorities, transport operators, and ticketing equipment suppliers in the UK. The key objectives of ITSO is the development and maintenance of the ITSO specifications which aims to deliver integrated and interoperable smart ticketing across the UK. ITSO also conducts testing and certification of components to ensure compliance to the ITSO specification [39].

The specification itself is in parts, covering the different components of ITSO necessary to deliver a smart ticketing solution. The keys parts include:

**Customer Media (CM)**

Initially the specification focused on smartcards, but has since evolved to make provision for other smart devices [39].

**Point Of Sale Terminal (POST)**

This is essentially a smartcard terminal, used to communicate with CM [40]. The POSTs are also connected to the back-offices to ensure reporting of transaction data.

**Host Operator or Processing System (HOPS)**

The HOPS carries out the back-office processes such as message handling and accounting functions [41].

**Message Data Elements and Structures**

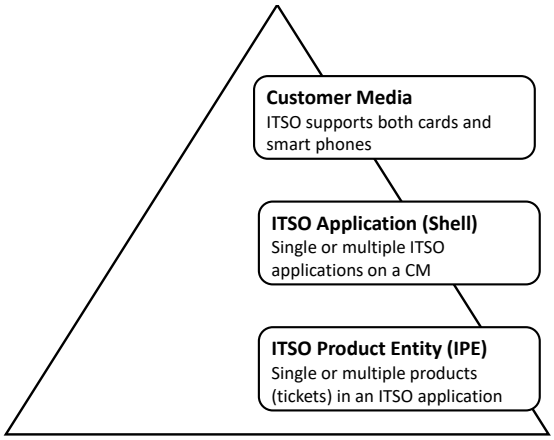


Figure 2.2: Layout of ITSO Ticketing Media

This is to ensure a common data structure and uniform data record definitions are used across all ITSO complaint implementations [42]. The security-related message flow between ITSO components is specified in [43].

**ITSO Secure Access Module (ISAM)**

ITSO mandates that both POSTS and HOPS have Secure Access Modules (SAM)[4] embedded, ISAM for the HOPS is referred to as a HSAM. The SAM is used to validate CMs and the data they hold, generate cryptographic keys, and store sensitive data to retrieve later.

The ITSO specification logically separates the fare media into three layers as shown in Fig. 2.2.

ITSO is a closed-loop based system and individual transport operators have to make provisions for their own payment functionality [44]. ITSO is a card-based solution with and supports pre-purchased and season tickets. It also supports PAYG ticketing. There are ongoing trials to enable ITSO on mobile devices through HCE.

[4]The SAM is typically in the form of an ID-000 punched smartcard, with a dedicated crypto-processor

### 2.5.2 Transport for London (TfL) Oyster Card

TfL, under the leadership of the Mayor of London, is the government body that is responsible for transport in London. This includes journeys on buses, trains, trams, ferries and bicycles. It also regulates taxis and other private car hire services in London. In 2003, TfL introduced a contactless card to be used for travel in London, known as the Oyster Card. There have been over 100 million oyster cards issued till date[5] [45]. The Oyster card can be used for rail, bus, tram and ferry travel in London. Until January 2010, Oyster cards were initially based on NXP's Mifare classic cards, but have since moved on to Mifare DESFire cards, due to security vulnerabilities discovered in the Mifare classic cards. The Oyster card supports season tickets and PAYG ticketing with daily price *capping*. Capping is a feature of PAYG whereby once the user's daily spend reaches a certain limit, further travel is free for the rest of the day.

The Oyster card system itself is based on a proprietary architecture and the contactless terminals were supplied by Cubic Transport Systems. However in 2012, TfL began transition to an ABT, open-loop ticketing and began accepting contactless bank cards on buses, followed by trains in 2014. Now over two million journeys are made daily, using contactless bank cards on the TfL network [1]. The system has since been extended to support mobile payments such as Apple Pay and Android Pay, and other implementations conforming to the EMV specification.

[5]This includes regular Oyster cards, Oyster photo-cards issued to those eligible for free or discounted travel, and the Visitor Oyster, tailored specifically for visitors.

## 2.6 Summary

This chapter presents a general background for ticketing systems. The evolution of the ticketing fare media, from the days of cash, to the current use of contactless technology has been highlighted. The general architecture, as well as the participants and phases of transport ticketing systems has been presented. This chapter also looks at the classification of ticketing systems, which lays a foundation on which significant parts of this thesis is built upon. Finally, a highlight of some of the ticketing implementations as well as the current trends from around the world is presented. The next chapter presents a background of technologies that are relevant to the work in this thesis.

# Chapter 3

# Background II: Technical Background

## 3.1 Near Field Communication

Near Field Communication (NFC) [46] is a short range contactless communication technology which enables the exchange of data between devices. It typically works within the range of less than 10 cm and at a Radio Frequency (RF) of 13.56 MHz. An NFC device may either be *passive* or *active*. An active NFC device (known as the initiator) is capable of producing its own Radio Frequency RF-field and directly transmitting data, while a passive NFC device (known as a target or a tag) relies on the initiator's RF-field for power and clock through load modulation, for data transmission. The communication between the two entities is half-duplex, meaning only one entity can send at a time. NFC is standardised by the NFC Forum [47] and is based on legacy standards such as ISO/IEC 14443 [5] and the European Computer Manufacturers Association (ECMA) [48] standards. NFC works in three different operating modes; *Reader/Writer Mode*, *Peer-to-Peer Mode*, and *Card Emulation Mode*.

### 3.1.1 NFC Modes of Operation

1. Peer-to-Peer Mode: This mode enables a two-way communication to be established between two NFC-enabled devices. Both devices serve

as an active entity and can initiate a communication and send data alternatively as 'peers'. The physical and data link layer of this mode are standardised in ISO/IEC 18092 [49]. This mode is usually used for quick exchange of small data, such as business cards and contacts.

2. Reader/Writer Mode:

   This mode enables an active NFC-enabled device to either read data from a tag or write data to it. This is typically used as information points in public places, for example smart posters for users to scan and access bus timetables on their devices.

3. Card Emulation Mode:

   The majority of the work presented in this thesis is based on this mode of operation. As the name suggests, an NFC-enabled device in this mode is capable of emulating a smartcard, and exchanging data in the form of Application Protocol Data Units (APDU)[1] with an external smartcard terminal. From the terminal's point of view, the mobile device appears and behaves like any other smartcard. There are two ways of performing card emulation: Card emulation using an SE, and Host Card Emulation. These are discussed in the coming sections.

[1] APDU is the unit of communication between a smartcard and a terminal.

## 3.2 Card Emulation Using A Security Element

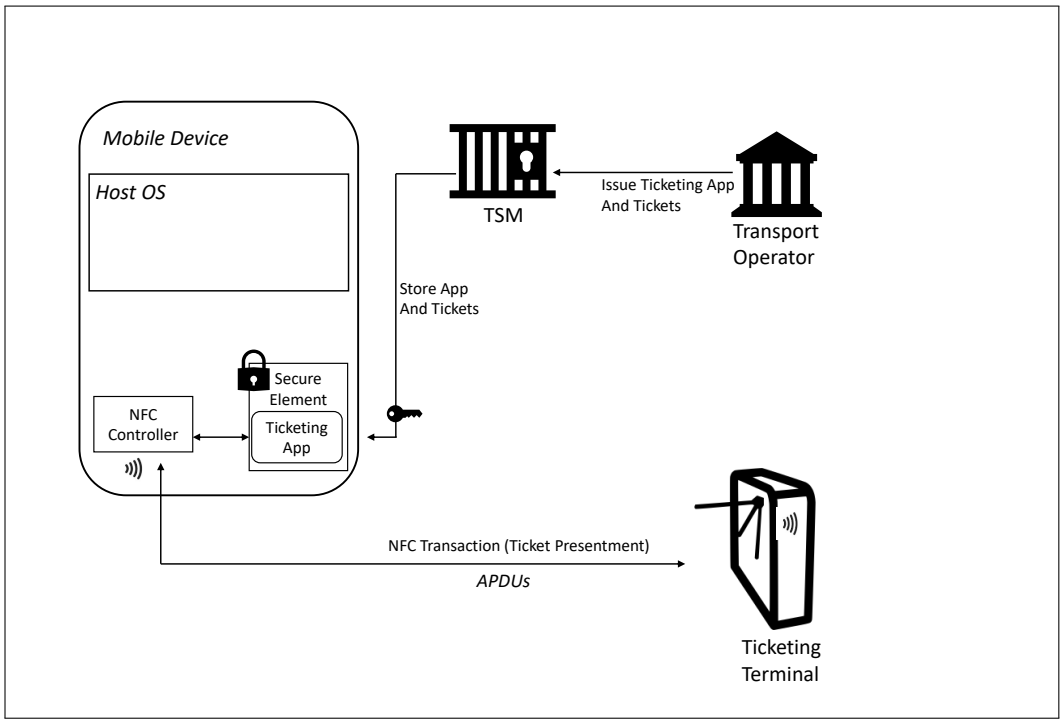


Figure 3.1: NFC SE Ecosystem

Near Field Communication (NFC) in card emulation mode, permits a mobile device to emulate a contactless smartcard and exchange APDUs with an external contactless terminal. From the card terminal's point of view, the mobile device appears and behaves like any other smartcard. Traditionally, NFC card emulation mode has relied on a tamper-resistant hardware SE to provide security protection. This is because the host OS cannot provide the level of application security required in certain use-cases, such as payments and ticketing. The SE provides a secure storage and execution environment for the applications emulating the smartcard. When an NFC emulation device is tapped on a terminal, command APDUs from the terminal are received by the NFC controller, via the NFC channel, and are routed to the SE as

shown in Fig. 3.1. However, the use of an SE introduces practical constraints for an application developer and/or a service provider. Access to the SE is tightly controlled by the Mobile Network Operators (MNO) and/or the Original Equipment Manufacturer (OEM), requiring complex business agreements before an application or related data is provisioned to the SE.

While card emulation has proven to be attractive for applications such as mobile payments and transport ticketing, the tight control on the SE means small companies and mobile application developers have no access to the services of the SE and indeed the ability to use the NFC card emulation functionality. This has hindered the use of NFC phones and prevented usage from reaching its full potential. There have been moves to resolve this by using a Trusted Service Manager (TSM), although this has not been universally adopted by businesses and standards. The TSM provides secure application provision and personalisation services, key management services as well as post-issuance lifecycle management.

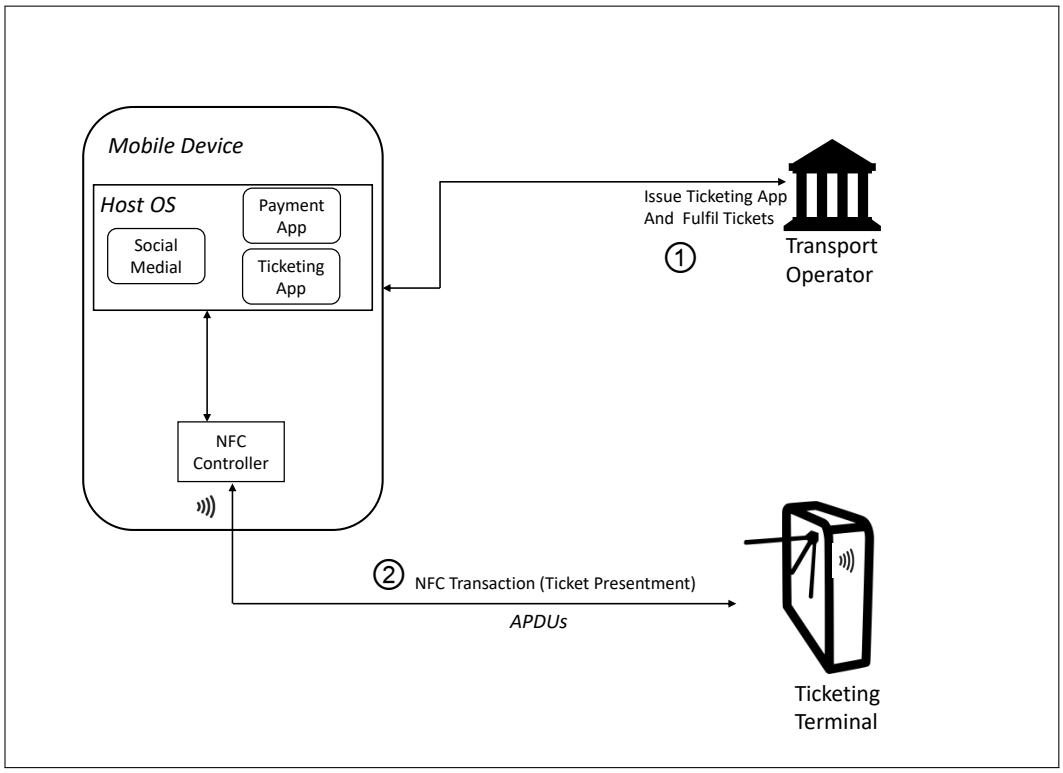


Figure 3.2: NFC HCE Ecosystem

## 3.3 Host Card Emulation

Host Card Emulation (HCE) [50] is a means by which an application running on the OS of the device, can emulate a smartcard, and communicate with a contactless terminal. This means that the NFC controller is effectively able to route APDUs to the application directly, bypassing the SE. This is depicted in Fig. 3.2. Therefore in HCE, security (attack resistance) is traded to facilitate NFC service development.

Research In Motion (RIM), on the Blackberry platform [51], were the first to incorporate this functionality in their phones. Subsequently Cyanogenmod integrated some patches [52] to the Android OS which permitted NFC enabled mobile phones to perform card emulation from the host. However, HCE attracted most attention when Google incorporated it within Android 4.4 (KitKat). The incorporation of HCE into Android has significant impact on the NFC Ecosystem considering Android phones account for about 88% of the global market share [53].

The key benefit of HCE is that anybody can implement an application that emulates a smartcard, which eliminates dependence on the SE. This opens up the NFC ecosystem to developers and businesses at no extra cost. HCE's impact on security cannot be ignored. The potential security issues are presented in the form of a threat model below.

### 3.3.1 HCE Application Development: Android Example

Android's HCE architecture is centred around Android services known as "HCE Services". Services have behaviours that are suitable for HCE applications for use in areas such as ticketing and payments. A service can run in the background and does not need user interference to operate. This means a user can simply tap the device against a terminal for the transaction to

go through, without necessarily launching the application. The implementation of an Android-based HCE application involves two stages; Application Selection and Data Exchange.

### 3.3.2 Application Selection

In a case where there is more than one HCE application on the device, application selection ensures that the correct application responds when the device is tapped against a terminal. All HCE applications running on the OS are identified using the Application Identifier (AID). The AID is registered at the beginning of development in the Android manifest file. The first command APDU received by the NFC controller is the *SELECT* command. The SELECT command contains the AID of the application that the external terminal wishes to communicate with. The NFC controller looks up the AID in its *routing table* and makes a decision on which application the AID is for. Subsequently, all further APDUs are sent to the corresponding application until a new SELECT command is received or the NFC link is broken [54].

### 3.3.3 Data Exchange

Card emulation using HCE requires the application to have a *service* component that handles NFC Transactions. All services *extend* the *HostApduService*. The HostApduService declares two abstract methods that as a minimum, are overridden and implemented. (Example code is shown in Listing Section 3.3.3.)

```
public class MyHostApduService extends HostApduService {
    @Override
public byte[] processCommandApdu(byte[] apdu, Bundle extras) {
        ...
    }
    @Override
    public void onDeactivated(int reason) {
        ...
    }
```

```
}
```

The *processCommandApdu()* method is called whenever the service receives a command APDU from an NFC terminal. Response bytes should be sent back almost immediately for a seamless user experience because the user will most likely be holding the device against the NFC terminal [50]. Otherwise, the response can be sent later using the *sendResponseApdu()* method. The *onDeactivated()* method is called when the NFC link is broken for any reason, or when a new SELECT command with a different AID is received. The *onBind()* method is used to return the communication channel to the service. The *notifyUnhandled()* is used to notify the OS when the service cannot complete the transaction.

### 3.3.4 NFC Applications Requirements

1. Security: Although specific requirements may vary from one case to the other, it is important to put strong security measures in place. The security of NFC applications is no different from the fundamentals of information security: Confidentiality, Integrity, and Availability (CIA). To use an NFC ticketing application as an example, the ticketing, payment, and cryptographic data must be safeguarded against unauthorised disclosure and modification both at rest and at run-time, and the transit system should be available for use at all times.

2. Performance: The performance of NFC-based transaction must be significantly fast in order to maintain high throughput and maintain overall user satisfaction; both the payment and the transit industries require that a transaction is completed within the range of 300-500 milliseconds [32, 55].

3. Flexibility: The use of NFC applications must not introduce rigidity to the ecosystem. For example, the case of card emulation using the SE

is rigid because provisioning an application into the SE requires permissions from the 'owner' of the SE, which in most cases is not easy. This means developers and researchers find it very difficult to make use of these services without signing contracts with various stakeholders. Therefore, an effective NFC solution must be flexible and work out-of-the-box.

4. Complexity: The complexity of the ecosystem should be as simple as possible. A complex ecosystem leads to increased implementation times and added cost. For example, the Trusted Service Manager (TSM) model used to provision applications to the SE increases complexity tremendously. A TSM is a trusted third party whose responsibility it is to facilitate provisioning, and to manage the entire life-cycle of the service. The inclusion of these extra parties makes it more complex and also more expensive because these services are not free.

5. Low Power Mode: In low power mode, the device's OS is shut down due to 'low' battery and therefore appears odd to the user; however, the Power Management Integrated Circuit (PMIC) is still on and can facilitate NFC transactions with help of power generated by the terminal in the field.[2]

6. Connectivity: Some NFC applications may require connectivity for every transaction, while some may only require connectivity from time to time, to update credentials. For example, some applications will require access to storage in the cloud for every transaction, while others may use tokenisation, and only require periodic connectivity to request new tokens.

[2]Low power mode should not be confused with "battery off mode" where even the PMIC has no power

7. Tamper-Resistance: NFC applications require protection against modifications and reverse-engineering by malicious entities. Software tamper-resistant techniques such as obfuscation and other techniques are used by developers to achieve this, although these techniques are usually costly in terms of performance and code size [56]. Other ways involve installing the NFC application and related data into a tamper-resistant physical silicon such as the SE.

8. Interoperability: In the context of this thesis, it represents the ability to host NFC applications on different devices from different vendors. This ensures the NFC ecosystem operates in a cohesive and efficient manner.

9. Standardised APIs and Ease of Development: The availability of standardised APIs goes a long way to easing the development process of NFC applications, thereby shortening the overall deployment time. Standardised APIs also ensure that developers adhere to software engineering global best practices, which is beneficial to the overall security fabric.

## 3.4 EMV Payment Technology

EMV represents Europay, MasterCard and Visa, the three companies[3] that initially constituted EMVCo, an organisation tasked with creating the technical specifications for chip-card payments (also known as Chip&PIN). Due to advances in mobile technology, the specifications have been extended to include other form factors, such as smartphones.

Prior to the introduction of chip-cards, magstripe cards were used for payments. Magstripe cards are vulnerable to data skimming and cloning [57], because they use static payment data. The chip-cards however are capable of performing cryptographic operations on dynamic data during transactions, thereby protecting against fraud.

1. More reliable methods of card-holder verification.

2. Robust risk management parameters.

3. Transaction integrity through the use of cryptograms.

4. Card authentication to protect against counterfeiting.

The aforementioned features are further elaborated in Section 3.5. EMV relies on application cryptograms generated using two-key triple DES algorithms to encrypt critical data elements used in the transaction. A cryptogram is the term used for a digital signature in EMV, and table Table 3.1 gives a summary of the different cryptograms and their functions.

[3]The membership of EMV has since grown to include other companies such as Discover, American Express and China UnionPay.

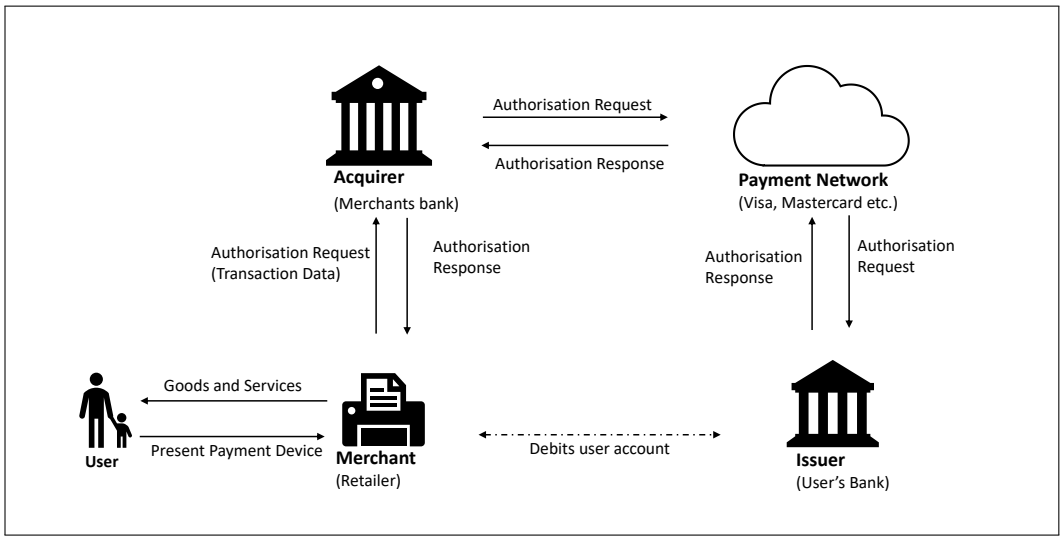


Figure 3.3: EMV Payment Architecture

Table 3.1: EMV Cryptograms

| Cryptogram | Function | Origin |
|---|---|---|
| ARQC | Request for transaction to go online for authorisation | Device |
| ARPC | Response to online authorisation request to device | Issuer |
| TC | Generated at the end of approved transactions | Device |
| AAC | Generated when a transaction is declined | Device |

### 3.4.1 EMV Architecture

The EMV follows the "four corner model" as shown in Fig. 3.3. It includes the *merchant* to whom the card is presented for a payment by the *card-holder*[4]. The merchant forwards the transaction data to his bank, which is referred to as the *Acquirer*. The Acquirer requests *authorisation* of payment from the *issuing bank* through a *payment network*. The issuing bank (or issuer) is the bank with which the card-holder has an account, and thus the bank that issued the card.

[4]The term card-holder is not exclusive to payment with cards, it also includes mobile devices.

### 3.4.2 EMV Primary Account Number

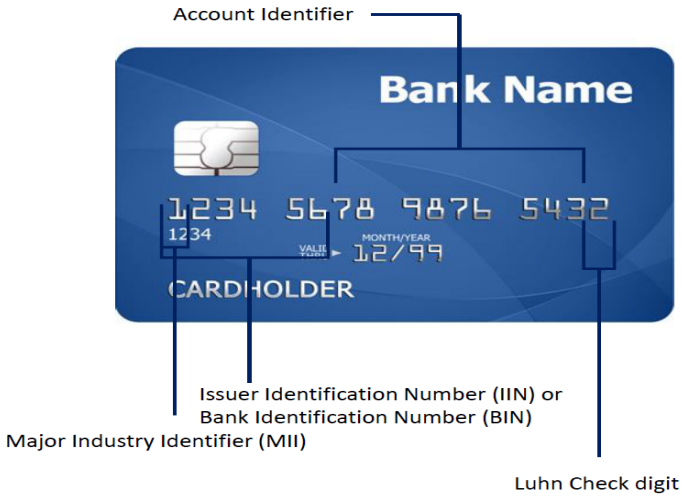


Figure 3.4: Anatomy of the PAN

Also referred to as card number, the PAN is a 16-19 digit number that serves as the payment device identifier of an EMV-compliant card or application and is uniquely issued as standardised in ISO/IEC 7812 [58]. From left to right, the first digit is the Major Industry Identifier (MII), with '4', '5' and '6' reserved for the payment industry. The first few digits including the MMI make up the Bank Identification Number (BIN) which identifies the type of card. The next digits excluding the last digit represent the account number, and the last digit is a checksum which is calculated using the 'Luhn check'. The luhn check is a simple checksum to verify the validity of identifiers including PANs, first described by Hans Peter Luhn of IBM in a patent [59] and standardised in [58]. The structure of the PAN is depicted in Fig. 3.4

## 3.5 EMV Key Concepts

A typical EMV transaction involves three key concepts; *Data Authentication*, *Card-Holder Verification*, and *Transaction Authorisation*. These are explained in detail below.

### 3.5.1 Data Authentication

Data authentication is a way of verifying if the payment device presented to the terminal is legitimate and not a counterfeit. Data authentication can either be online or offline. In online authentication, an internet connection is required to send transaction-specific data to the issuer for verification. There are three types of offline data authentication as explained below.

1. Static Data Authentication (SDA): The SDA verifies that the 'static' data on the card has not been manipulated since provisioned by the issuer. During a transaction, the terminal uses the issuer's public key to verify the cryptogram against the card data. SDA doesn't provide protection against card skimming because it uses static data and an attacker can skim the cryptogram and place it in a counterfeit card.

2. Dynamic Data Authentication (DDA): In DDA, a transaction-specific cryptogram is generated dynamically by the card using its own private key. During a transaction, the terminal generates a random number and sends it to the card. The card in turn generates a cryptogram which includes the random number. Successful verification of the cryptogram by the terminal provides assurances that the card data has not been manipulated. But even more importantly, it proves that the card is not counterfeit since it is infeasible to extract keys from the chip of a card.

3. Combined Data Authentication (CDA): CDA is a variation of DDA, and it uses a special cryptogram referred to as the Application Cryptogram

(AC). Similar to the case of DDA, the card must be capable of public key cryptography which is typically done using the RSA algorithm. But in CDA, the card generates the AC dynamically later during the protocol to prove that the card that performed the authentication initially is the same card authorizing the transaction.

### 3.5.2 Card-Holder Verification

Also referred to as Card-Holder Verification Method (CVM), this method is used to ascertain that the person presenting the payment device is indeed the legitimate card-holder. During the transaction, the terminal checks the CVM list which holds logic on which CVMs the payment device supports in order of priority. The CVMs supported by EMV in order of their security are; *online PIN*, *offline PIN*, *user-biometric*, *signature*, combination of two or *no CVM* at all. There are also options for the PIN to either be in plaintext or encrypted in both online and offline cases.

### 3.5.3 Transaction Authorisation

EMV Transactions can either be authorised online or offline. This is decided as a result of processes taken at this stage, as listed below:

- Terminal Risk Management: The goal here is to prevent fraud using mechanisms such as floor limits checking and random transaction selection [60]. Floor limits protects against overspending by making sure the transaction is below a limit set by the acquirer. Random transaction selection is when the terminal randomly makes an offline transaction to go online for authorisation.

- Terminal Action Analysis: The result of all the previous steps including risk management is analysed by the terminal, and as a result then informs the device if it prefers to finish the transaction offline or go online.

- Card Action Analysis: The device analyses the result of all previous steps, and makes a decision on whether the transaction should go online, be completed offline, or declined. If the decision is to go online, then a second card action analysis is carried out after the processing to authenticate the data received from the issuer. In any case the device's decision is communicated to the terminal by generating an Authorisation Request Cryptogram (ARQC), an Application Authentication Cryptogram (AAC), or a Transaction Certificate (TC) cryptogram for online, offline or decline transaction respectively (see Table 3.1). The decision of the card may differ from the decision taken by the terminal, but ultimately the card's decision is final.

- Transaction Completion: At the end of the processing in the previous stage, the device may be requested to verify an Authorisation Response Cryptogram (ARPC) from the issuer. The terminal may also carry out script processing on the device based on the issuer's commands. The user is alerted to remove the device and settlement is requested assuming the transaction was successful.

## 3.6 EMV Considerations for Ticketing

| Consideration | Retail Payments | Ticketing Payments |
|---|---|---|
| Card-Holder Verification (CVM) | A CVM method is required. e.g. PIN | No CVM is required when a user taps the payment device |
| Price to pay | Known at the time of payments. | Not known at the time of journey |
| Payment Terminal Field | Activated by the store attendant | Always active, ready to be tapped by users |

Table 3.2: Payments: Retail vs Ticketing

As mentioned in Section 2.3.4, open ticketing systems rely on the user's payment card or device for ticketing. However, from a functional point of view (see Section 5.5.1), ticketing payments differ from normal retail payments. Therefore the payment industry considered these factors, and adopted new rules to accommodate ticketing [61], Table 3.2 shows these considerations, in comparison to 'regular' retail payments.

## 3.7 EMV Payment Tokenisation

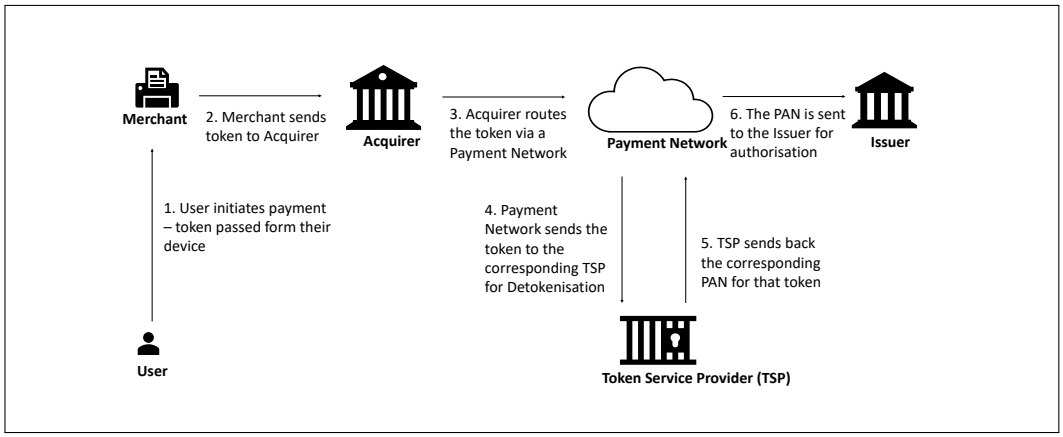


Figure 3.5: Transaction Flow of a Tokenised Payment

Tokenisation is a process of replacing sensitive data with a surrogate value to limit exposure. This is particularly important for protecting data such as payment data, health records or any Personally Identifiable Information (PII)[5]. The surrogate value is usually short-lived and therefore is of minimal importance to an attacker in case of a breach.

There have been various initiatives towards the standardisation of tokenisation. The American National Standards Institute (ANSI), under the X9F1 group have ongoing development of a standard that covers tokenisation algorithms for the financial industry [62].

The PCI Security Standards Council (SSC) also have guidelines for the evaluation of tokenisation-related products and services including both hardware and software components [63].

But the most relevant specification to the work presented in this thesis is the EMV payment tokenisation. In this context, tokenisation replaces the PAN with a short-lived, surrogate value referred to as a 'token' [64]. According to EMVco "Tokenisation may be undertaken to enhance transaction efficiency,

[5]A PII is any information that could individually identify a perso.

improve transaction security, increase service transparency, or to provide a method for third-party enablement".

EMVco introduces two new entities; the *token requestor* and the *Token Service Provider* (TSP) to the existing payment network. The TSP is authorised to generate, issue, and provision payment tokens to legitimate token requestors. The TSP is also responsible for maintaining the PAN-token mapping in the token vault, as well as *detokenisation*, i.e. the translation of tokens back to PANs for legitimate requests. Figure 3.5 shows the transaction flow in an EMV tokenised transaction.

The EMVco does not limit the role of the token requestor to any one party, perhaps to accommodate for different scenarios. The specification nevertheless specifies potential token requestors to include card issuers, digital wallet providers, OEMs when acting as payment enablers and payment gateways on behalf of merchants [64]. Prior to every token request, the specification requires an *Identification & Verification* (ID&V) step to ensure its legitimacy. (ID&V) methods include account verification[6], card-holder verification by the issuer etc.

[6]Account verification includes checks such as the so-called "$0 authorisation" to validate that the PAN is legitimate and is active at its issuer.

## 3.8 Summary

In this chapter, a general background is given for technologies that are relevant to the work presented in this thesis. NFC and its different modes of operation, particularly the two methods of card-emulation have been discussed. The security mechanisms that can be used to reduce the attack resistance of HCE applications will be discussed in Chapter 4. EMV as a payment technology has also been described. EMV is relevant to this work because the move from closed-loop to open systems increasingly puts the responsibility of ticketing payments in the hands of the payment industry — which uses EMV as the *de facto* standard — for these types of transactions. Finally, the concept of tokenisation from EMV's point of view has been discussed. Tokenisation is important because it is the preferred security mechanism for handling the risks of HCE in payments. More details on the impact of tokenisation in ticketing is presented in Chapter 5.

# Chapter 4

# Comparing SE and HCE Capabilities

## 4.1 Introduction

The emergence of HCE has raised a lot of interest and concerns from both research and industry communities. In many ways, HCE represents a sharp deviation from the traditional ways of performing contactless transactions with a smartphone, and by extension the SE. The transport ticketing industry relies heavily on smartcards and mobile technologies to facilitate and automate operations. In ticketing, HCE is a disruptive technology that offers many advantages for users and Transport Operators (TOs). However, HCE introduces some issues that call for scrutiny and careful consideration to ensure its success in ticketing. In this chapter, these issues are examined from three perspectives: implementational considerations, security considerations, and performance considerations. These factors affect the success of ticketing schemes at all levels of the transport ticketing ecosystem. Implementational considerations are issues that must be considered before a large-scale ticketing scheme based on HCE is rolled out. It is a common misconception to assume the transition from SE-based implementations to HCE-based ones is a simple like-for-like change in technology. From a technical point of view, this misconception is further exacerbated by the fact that, since HCE claims

to emulate smartcard-based services, then everything should work out of the box. While this is true to a large extent, there are some fundamental differences between the two approaches, some of which are specific to transport ticketing. In Section 4.3, these issues are looked at in more detail.

From a security point of view, there is no doubt of the magnitude of the inherent security risks of the HCE approach. To counter the risks, numerous security mechanisms have been proposed based on a generic assessment of HCE. In Section 4.4, these security mechanisms are analysed with transport ticketing in mind, to determine their feasibility in this area. Due to the security and functional requirements of transport ticketing, there may be distinct challenges that do not apply in other cases such as access control and mobile payments. We look at these issues and determine the feasibility of the security mechanisms in ticketing.

The performance of ticketing applications is paramount to the operation of transport ticketing schemes. This is because they affect the throughput of passengers at entry and exit points of the transport service, i.e. train stations or bus stops. There is a strict timing requirement with regards to ticket and/or user verification at a terminal. While the performance of such applications on smartcards and SEs is a hot research topic, their HCE counterparts have not undergone the same level of scrutiny. To that effect, in this chapter, we implement two similar applications, on an SE and also in HCE. We test these applications under laboratory conditions and present the results in Section 4.6. We also highlight an interesting behaviour in HCE applications that may call earlier assumptions into question and may jeopardise the security and functionality of these applications.

## 4.2 Related Work

There are many publications comparing the two approaches to card emulation on mobile devices. In [65], Pannifer et al. compared HCE to UICC-SE, discussing the ecosystems for both approaches and providing a comparison based on the provisioning infrastructure, usability, security and maturity of the technologies, as well as their business models. However, the comparison falls short of any practical analysis, so no performance results were provided. In [51], the author discussed the advantages and the security implications of HCE in comparison with an SE-based approach. However, there were no implementations and thus performance measurements were not given.

In [66], Roland and Langer gave an overview of the three NFC modes, and a relatively new concept of NFC communication referred to as the inverse reader-mode, which departs from the norm of using NFC devices either in card emulation or peer-to-peer mode to communicate with a terminal. First introduced by Saminger et al. [67], the inverse reader-mode switches the roles of the device and the terminal; therefore the NFC device will be in reader/writer mode, and the terminal in card emulation mode. The main advantage of this approach is that developers and Service Providers (SPs) can avoid the problem of accessing the SE for card emulation, and instead can provide their services in reader/writer mode, leaving that to the terminal. An appropriate analogy for this concept is switching the roles of client and server. Similarly, the work presented in [54, 68] analyses HCE from a security point of view, and suggests countermeasures to deal with the identified security risks. However,the work presented in [51,54,66–68] all fall short of practical implementations and performance analysis.

Although not particularly related to HCE, there has been a substantial amount of research on the feasibility of real-time applications on mobile devices [69–72]. Real-time applications are computer applications or systems

that require absolute response times. It is appropriate to consider transport ticketing applications as 'soft real-time' applications[1]. The authors of [69–72] agree that the Android OS is not appropriate for soft real-time applications out-of-the-box. However, Alejandro et al. in [69] looked at the isolation of CPU cores, which could be used to mitigate the magnitude of the variation in performance of soft real-time applications. This concept is very similar to the one used in this chapter. However, it is worth noting that the work in this chapter was published before the solution presented in [69].

[1]Hard real-time applications are considered to have failed if they fail to respond within the required response time e.g. Nuclear Systems. Soft real-time applications on the other hand are not considered to have failed even if they don't respond within the required time.

## 4.3 Implementational Considerations

In this section, a comparison of HCE and SE-based approaches is provided. The metrics used in the comparison represent the different Key Performance Indicators (KPIs) that determine the success of a mobile-based transport ticketing implementation.

### 4.3.1 Ecosystem Complexity

The ecosystems of the two approaches as shown in are different. In the SE-based approach, there is a Trusted Service Manager (TSM), which is a secure and trusted entity that acts as a link between the Service Provider (in this case the transport operator) and the SE itself. The TSM provides a secure technical infrastructure for service providers to provision and manage their applications in the SE remotely. In the HCE approach, however, the role of the TSM is not required. Application provisioning here is straight-forward and the life-cycle of the application and related credentials are solely the responsibility of the transport operator. Therefore in terms of complexity, the HCE-based approach offers a simpler ecosystem.

### 4.3.2 Provisioning

Provisioning in terms of the HCE application itself is relatively straight-forward. The most practical way is to let users download the application from the 'app store', as with other applications, since an interaction with the MNO or any other third party is not required. In addition, if tokenisation is used (see Section 3.7), then periodic tokens will have to be pushed to the device like a secure channel, Secure Socket Layer (SSL) for example. In addition other security measures such as device authentication and user verification prior to the provisioning of new tokens should not be neglected. This leads us to the next point on usability.

### 4.3.3 Usability

The usability of ticketing applications and procedures is paramount to the success of the ticketing scheme. It is important to note that unlike an SE, it is not possible to carry out a transaction with HCE applications when the battery of the host device is drained and cannot power up the OS. This may have a significant impact in ticketing, especially in rail ticketing where the tap-in tap-out concept is used, and the fare is typically charged at the end of the journey. The concern is the user's device may be out of battery to tap-out. Also, security measures taken may affect the usability of HCE applications. As mentioned in Section 4.3.2, the validity of tokens and the user verification method used will significantly affect usability. The ideal scenario would be for new tokens to be provisioned to the device with minimal interactions with the user.

### 4.3.4 Performance

The performance of applications in an SE-based approach is inherently slower than similar HCE-based implementations. This is because SEs by design are constrained devices, and have significantly lower processing power when compared to the host device. However, performance testing carried out in this chapter shows that HCE-based applications are prone to performance variation between transactions, making absolute performance benchmarking difficult. This can pose a challenge to ticketing, where timing is critical.

### 4.3.5 Cryptography

The cryptographic algorithms used for ticketing protocols in large-scale deployments must be chosen carefully. For example the Android keystore system only supports symmetric cryptography from API level 23 and above [73]. This means about 66% of [74] user devices on the Android platform do not support symmetric algorithms, which is very significant. Therefore cryptographic

compatibility must be considered when designing protocols for HCE-based applications.

### 4.3.6 Cost

There is usually an additional costs incurred from the use of an SE. SE-owners[2] require service providers to pay a determined 'rental fee' associated with housing their application in the SE. Service providers also pay for the cost of the TSM infrastructure. In HCE, there is no need for the Service Provider to pay for provisioning the application to the host device; however, risk mitigation mechanisms associated with HCE-based implementations might incur a cost. For example, if tokenisation is used there will be a cost determined by the Token Service Provider (TSP).

### 4.3.7 Standardisation

The level of standardisation usually reflects the maturity of the technology. The SE is backed by well-established standards and specification. The provisioning and management of applications on an SE is standardised by GlobalPlatform [75]. The framework for the Universal Integrated Circuit Card (UICC) as an SE is standardised by the Global System for Mobile Association (GSMA) and the European Telecommunications Standards Institute (ETSI). There are application-specific standards and specifications that also apply to SEs such as EMVco and MIFARE. HCE on the other hand is a less mature technology and therefore has less standardisation. Nevertheless, there are some application-specific specifications such as those involved with Mastercard and Visa.

[2]The SE-owner refers to the entity that holds access to the SE, which could be an MNO, OEM, or Issuers depending on the SE form factor.

## 4.4 Security Considerations

As discussed in Chapter 3, the security risks introduced by HCE are clear. To that effect, a number of mechanisms have been proposed — both academic and industry-led — to manage the security risks in HCE-based implementations [54,68,76–79]. In this section, an analysis of these security mechanisms is carried out. Specifically, we look at how the adoption of these security mechanisms may impact ticketing, and highlight the considerations that must be taken into account. A comparison of these mechanisms (including the Secure Element) is given in Table 4.1.

### 4.4.1 Cloud Storage

By design, cloud-based transactions will require an internet connection between the users' devices and the cloud server. In transport ticketing, an internet connection is difficult to guarantee in certain scenarios such as underground stations and moving vehicles. Therefore it is a strong requirement for transport ticketing solutions to grant or deny users' travel offline. In addition, the strict performance requirements of transport transactions may prove difficult to achieve due to network latency. Form a security perspective cloud storage shifts most of the security considerations from the mobile itself to the 'cloud'. Consequently, security considerations must be made around issues such as: effective access control measures, isolation of the data stored in the cloud, security of the data both in transit and at rest.

### 4.4.2 Tokenisation

Tokenisation replaces a long-term, security-sensitive piece of data with an item of data that is short-lived and referred to as a token. In EMV, the main component that is tokenised is the Primary Account Number (PAN). Tokenisation

has shown a lot of promise. However, there are a few considerations necessary for successful adoption in ticketing. Firstly, an internet connection is needed to provision the device with a fresh token. Currently, the EMV specification [64] allows for both single-use and multiple-use tokens, and does not specify the length of the validity of tokens. The validity of these tokens will depend on the perceived risk of exposure on the device. It is unreasonable to require an internet connection before every transaction because it is difficult to guarantee connectivity in certain cases, such as at underground train stations. Therefore the choice of the validity of tokens must take this into account. In addition, the use of PANs has evolved beyond being a mere account identifier to blacklist dishonest users. By using tokenised payments, blacklisting could be compromised since it is expected that the token will be short-lived. More on this is presented in Chapter 5.

In terms of security, tokenisation provides stronger security guarantees than storing long-term credentials locally, since they have a shorter life-time, and can even be restricted to use within specific use-cases, thereby reducing the overall appetite for an attacker to get hold of tokens. However an ideal situation would be to have an extra layer of security. Point-to-point encryption and TEEs are good candidates to provide an extra layer of security.

### 4.4.3 Trusted Execution Environment (TEE)

A TEE provides a trusted area on the mobile device for secure data storage and code execution. It also provides other security services such as remote attestation, which could be useful for ticketing. With high market penetration, TEEs provide acceptable levels of security and performance for application in transport ticketing.

TEEs are, however, tightly controlled by the equipment manufacturers, similarly to SEs. This tight control adds to the complexity of the ecosystem because TEEs on devices cannot be used 'out-of-the-box' and will require

permission in the form of contracts with the respective OEMs. This also leads to an added economic cost in TEE adoption. This issue is looked at more closely in Chapter 6.

### 4.4.4 White-Box Cryptography (WBC)

With ticketing being a time-critical service, white-box implementations are slower than their black-box counterparts [80, 81]. This could jeopardise the ability of ticketing applications to meet the performance requirement. More importantly, it has been shown that WBC implementations are vulnerable to several types of attacks. For example the white-box DES and AES implementations presented in [82] and [83] respectively have been proven to be susceptible to numerous attacks as presented in [84,85]. As improved variants, some proposals have adopted the use of external encoding to provide some sort of shield. However, even the proposals adopting external encoding have proven to be vulnerable to cryptanalysis as shown in [86] and in [87]. WBC shows much promise for application in transport ticketing. However, its performance, security issues, proprietary nature and lack of robust standardisation makes large-scale adoption a challenge.

Table 4.1: Comparison of Security Mechanisms

| | **SE** | **TEE** | **Cloud-HCE** | **White-Box Crypto** | **Tokenisation** |
|---|---|---|---|---|---|
| **Security** | High | Medium | Medium | Medium | Medium |
| **Performance** | High | High | Medium | Medium | Medium |
| **Flexibility** | Low | Low | Medium | Proprietary | High |
| **Connectivity** | No | No | Yes | No | Yes[a] |
| **Standardised API** | Yes | Yes | No | No | Yes |
| **Interoperability** | Yes | No | No | No | Yes |
| **Tamper Resistance** | High | Medium | Medium | Medium | Medium |
| **Set-up Cost** | Yes | Yes | Yes | Yes | Minimal[b] |

[a] Connectivity is required periodically, when new tokens are provisioned.
[b] Initial set-up cost is minimal. There is a running cost of provisioning new tokens to device.

## 4.5 Considerations of HCE Performance

This section provides a comparative analysis of the performance of HCE and SE-based applications. This is achieved by implementing similar applications (in terms of functionality), on an SE and in HCE. Their performance is analysed both in terms of computation and the transfer delays on the NFC communication link. The results raise some issues that could have an impact on the use of mobile devices in ticketing.

### 4.5.1 Android CPU Policy

Android devices typically have a multi-core processor design. This means that a single unit can consist of two or more independent CPUs referred to as cores. Android devices use both the ARM and x86 architectures. Android also uses the Symmetric Multiprocessing (SMP) design for managing the different CPUs [88]. Normally, all CPUs share the same CPU frequency policy, i.e. all CPUs are online at the same time and any process can run on any of them. However, Android devices switch between different frequency levels in response to variable CPU load; this is enforced and regulated by a driver known as the CPU governor [89]. There are several types of CPU governors with different characteristics. For this work, the ondemand governor which is the factory default for most Android phones, increases and decreases CPU frequency according to demand. We also used the userspace governor: it permits a user to choose which frequency state the CPU should run in.

### 4.5.2 Test Methodology

Table 4.2: Devices Used in Testing

| Device | Manufacturer | Operating System | RAM | ROM |
|---|---|---|---|---|
| **SE (Nokia)** | NXP | SmartCafe expert 3.1 | 4kb | 160kb |
| **Phone (Nexus 5)** | LG Electronics | Android Kit-Kat 4.4 | 2GB | |
| **Laptop** | SONY Vaio | Ubuntu 14.04.1 | 4GB | |
| **Card Reader** | SCM | | | |

In this section we explain how the testing was set up and conducted. To test performance with regard to variable delays, we developed two card-emulating applications. The first one was a Java card applet based on the Java card framework v2.2.1 [90]. We refer to this applet as the SE-app. We loaded the SE-app into the SE of a Nokia 6131 (one of the first NFC phones). Nokia, in [91] provides a tutorial on how to use the 'Nokia NFC Unlock Service MIDlet' to unlock the SE of the Nokia 6131. The second application was an Android application (HCE-app), running on Android platform version 4.4 with Android framework Application Programming Interface (API) level 19 as the target API. This application was deployed in a Nexus 5 mobile device. Table 4.2 provides a summary of the devices used to conduct the testing.

Both applications ran the same cryptographic protocol designed for the tests. The purpose of the protocol is simply to require the emulating processor to carry out some non-trivial and representative cryptographic processing. The *javacardx.crypto* package was used for the SE-app, and the *java.crypto* package for the HCE-app.

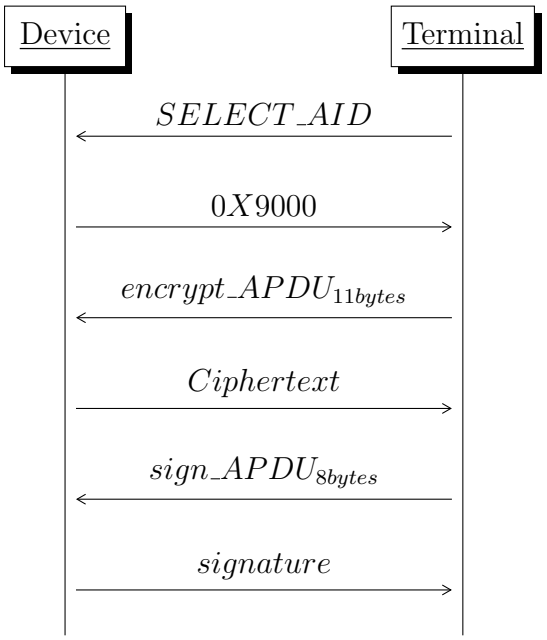


Figure 4.1: Protocol Used in Testing

A 1024-bit RSA algorithm keypair was used, and to simplify testing, we departed from cryptographic best practice not only by using this key size, but also by using the same pair for both encryption and signing. For the encryption we used the *ALG_RSA_PKCS1* field of the *cipher* class from the *javacardx.crypto* package. *ALG_RSA_PKCS*1 uses the RSA cipher for encryption and the Public-Key Cryptography Standards (PKCS1) (v1.5) for data padding. For the signature, we used *ALG_RSA_SHA_PKCS1* field of the *cipher* class from the *javacardx.crypto* package. *ALG_RSA_SHA_PKCS1* firstly computes a hash of the message resulting in a 20 byte SHA digest, which is then padded using the PKCS1 before signing using the RSA algorithm.

We recognise that SHA-1 is no longer recommended; however the goal was to test performance variation rather than establish a secure protocol. For performance testing, we first used the Java *timer* class in the terminal application to take measurements. To ensure accuracy of the values produced by the timer in our program, we then used the CLT Move-Contactless Spy tool from COMPRION [92] to double-check the measurements. This tool monitors

the flow of APDUs between the terminal and the phone, giving a byte-level view of messages and their corresponding delays.

## 4.6 Experimental Results

The first phase of any communication between a smartphone and a terminal is the initialisation and anti-collision phase; this [46] is where they establish identities and agree on the parameters of communication for all subsequent messages. The parameters include the number of bits they can handle at a time and also the agreed amount of time the terminal has to wait for a response from the phone before it times out, known as the Frame Waiting Time (FWT). The anti-collision phase exists to resolve the situation of multiple smartcards within range of a single terminal, which was not the case for our tests.

After a successful initialisation and anti-collision, the terminal application selects the applet by sending a select APDU using the AID; the application sends back the 0x9000 status word if the select APDU was successful. The terminal then sends a command APDU (ENCRYPT) with an 11-byte data to be encrypted by the card-emulating application. The application encrypts the message and sends back a response APDU containing the corresponding cipher text together with the 0x9000 status word (or error code on failure). Subsequently another command APDU (SIGN) is sent to the phone in order to get a signature on the 8-byte data contained in the APDU. The phone signs the data and sends the result back to the terminal. Timing is measured from the time the command is sent from the terminal to the time it receives the response back. For accuracy, we carried out 400 runs of the protocol and computed the average. All timings were recorded in milliseconds (ms).

### 4.6.1 SE-app Testing

Table 4.3 shows a statistical analysis of the results from the SE-app on the Nokia hardware SE. It shows the statistics for individual commands as well as the protocol in full over 400 runs.

Table 4.3: SE-app Testing Results in Milliseconds

| SE-app | SELECT | SIGN | ENCRYPT | FULL PROTOCOL |
|---|---|---|---|---|
| **AVERAGE** | 18 | 1679 | 285 | 1982 |
| **MODE** | 18 | 1691 | 283 | 2002 |
| **MEDIAN** | 18 | 1680 | 284 | 1984 |
| **MAX** | 116 | 1746 | 300 | 2105 |
| **MIN** | 17 | 1618 | 279 | 1917 |

From Table 4.3 we see that the SELECT command had an average execution time of 18.6ms. SELECT is a fairly light-weight command requiring no cryptographic processing and simply returning a status response. However, the first SELECT command of the 400 runs took 116ms. This is because of the initialisation and anti-collision phase, which is not present in the remaining runs, which had very consistent execution times with the minimum and maximum times differing by at most 3ms. It is important to note that in a real-life scenario, for every tap of the phone unto a terminal by the user, there will be an initialisation and anti-collision phase. Overall our test protocol had an average execution time of 1982ms, with about 85 percent of the time spent on digitally signing the message. The absolute duration would be too long for timing-critical applications (e.g. transport ticketing), although the measurements are very consistent, with some variation due to experimental tolerances.

### 4.6.2 HCE-app Testing

For the HCE-app, we did things slightly differently; we ran the test in two different scenarios which we will henceforth refer to as *Case 1* and *Case 2*. As suggested in [93] "While measuring CPU power, or holding CPU power constant in order to make other measurements, it may be best to hold the number of CPUs brought online constant, such as to have one CPU online and the rest offline (hotplugged out)". In our case, the Nexus 5 has a multi-core processor with four cores, so we hot-plugged three of them, leaving only

one online. This was to ensure that all processes ran on the same CPU, giving us better control of the testing platform.

#### 4.6.2.1 HCE-app Testing: Case 1

Here, the ondemand governor running at 960Mhz (default) was used. While running the application, we engaged in a deliberate simulation of the day-to-day things a mobile phone user could actually be doing, such as opening social media applications and answering phone calls. This was to mirror the day-to-day behaviour of a typical user's device when it is tapped on a terminal at a train station. From Table 4.4 below, the first SELECT command took longer than the others (as was the case with the SE-app) i.e. 60ms as compared to the average of 12ms. The full protocol ran in an average of 213ms; however, it is interesting to see that the range between the MAX and MIN values of the full protocol was 708ms, which is a significant variation. Similar variations were measured for SIGN and ENCRYPT values individually, which ran on average for 110ms and 91ms respectively but with a range of 475ms and 404ms respectively.

Table 4.4 shows a statistical analysis of the results from the HCE-app (Case 1). It shows the statistics for individual commands as well the protocol in full over 400 runs.

Table 4.4: HCE-app Testing Results in Milliseconds (Case 1)

| | **SELECT** | **SIGN** | **ENCRYPT** | **FULL PROTOCOL** |
|---|---|---|---|---|
| **AVERAGE** | 12 | 110 | 91 | 213 |
| **MODE** | 11 | 85 | 82 | 171 |
| **MEDIAN** | 12 | 95 | 81 | 188 |
| **MAX** | 60 | 544 | 465 | 853 |
| **MIN** | 10 | 69 | 61 | 145 |

#### 4.6.2.2 HCE-app Testing: Case 2

Table 4.5 shows a statistical analysis of the results from the HCE-app (Case 2). It shows the statistics for individual commands as well the protocol in full over 400 runs.

Table 4.5: HCE-app Testing Results in Milliseconds (Case 2)

| | **SELECT** | **SIGN** | **ENCRYPT** | **FULL PROTOCOL** |
|---|---|---|---|---|
| **AVERAGE** | 20 | 301 | 250 | 572 |
| **MODE** | 20 | 271 | 235 | 529 |
| **MEDIAN** | 20 | 294 | 236 | 550 |
| **MAX** | 99 | 883 | 809 | 1146 |
| **MIN** | 16 | 265 | 222 | 514 |

After noticing the high variance in the results produced in Case 1, we decided to clock the CPU at its lowest possible frequency state of 300Mhz. To achieve this we changed the CPU governor from the default ondemand governor to the userspace governor; with this, we were able to set exactly the CPU frequency. We also set the minimum and maximum frequency to both be 300Mhz as suggested in [93], giving us assurance that the CPU was fixed at 300Mhz.The protocol was run under these conditions and Table 4.5 shows the results. As in previous tests, the first SELECT command was slower, taking 99ms as compared to the overall average of 20ms. The full protocol took an average of 572ms, but similar to Case 1, there was a significant range of 632ms. The same applied to the SIGN and ENCRYPT values, with averages of 301ms and 250ms respectively, and ranges of 618ms and 587ms respectively.

## 4.7 Discussion

In this section, we put all three tests together to conduct analysis and make comparisons.


Figure 4.2: Graph Showing the Results of All the Tests

Figure 4.2 shows that the SE-app has a more clustered line graph (almost a constant), while the two HCE-app cases show marked variations in values. We can see how the CPU clock affects average execution time for the protocol; however, wide variations exist even when the CPU core is allowed to run at full speed. The absolute execution time of the SE is notably slower than the HCE-apps although this is perhaps an unfair comparison as the Android phone is still current whereas the Nokia phone (and its SE) was produced in 2006. Furthermore, the HCE-app would likely be significantly slower in practice if software security measures were required to reduce side-channel leakage and improve attack resistance. What is most interesting is the variation in the HCE-app response times, which may call into question HCE use in time-critical applications. For example, in transport ticketing applications there is a strict requirement that a gate transaction should be completed in less than 500ms [36, 94]. We are seeing variations that are greater than this, regardless

of the expected execution time.

This timing variation could also prevent the use of some security measures to detect fake cards or attacks in progress. For example, there has been prior work in detecting RFID relay attacks [95] using distance bounding protocols [96–98] as a way of detecting relay attacks. These protocols establish an upper bound on the distance of the proving party. This is done by taking into consideration the delay introduced into the channel from the time a challenge is sent to the time a response is received. This is only possible when there is a reasonable benchmark for an acceptable delay. In the case of HCE, the tolerance around the benchmark will be extremely large, making it very difficult to distinguish between a relay attack and variation due to normal phone operation.

## 4.8 Summary

In this chapter, HCE was analysed in the context of transport ticketing and some of the underlying issues regarding large-scale implementation were discussed.A range of security mechanisms put forward to mitigate security risks of HCE where analysed against factors that determine the success of mobile devices in ticketing. In addition, a comparative analysis of similar implementations of SE and HCE applications was conducted. Results show the HCE-app to be faster in terms of performance, however a significant variation in transactions times from one transaction to another introduces a new dynamic that requires careful consideration.

# Chapter 5

# Blacklisting Tokenised Payments

## 5.1 Introduction

Transport operators are transitioning from closed-loop ticketing systems to open ticketing systems that rely on a well-established payment infrastructure for fare payments. This allows users to make use of payment media such as debit/credit cards, and mobile payment applications already in their possession, to pay for transport fares.

Historically, smartcards have provided an acceptable level of security as required by the payment industry. The embedded chips provide a secure and tamper-resistant storage and execution environment for payment applications and related credentials. Of most relevance to this chapter is the Primary Account Number (PAN) [58], which provides a unique reference to the user's payment device - and by extension identifies the user on the payment network.

PANs have evolved from being just an account reference of the user. Merchants use PANs for unique identification of users for different purposes, such as customer loyalty and uniquely tailored services. More significantly, in open ticketing, PANs are used as unique identifying factors for users of the transport network. In the event of dishonest behaviour by a user, such as fare evasion, the PAN is also used to blacklist the user so no further travel is

allowed [99, 100].

### 5.1.1 Problem Statement

Different mechanisms have been proposed to manage the risks of HCE's reliance on software and make it acceptably secure for payments. These mechanisms are discussed in more detail in Section 4.4, but the focus in this chapter is tokenisation. The goal of tokenisation is to replace the PAN in the user's device with a surrogate value that has a shorter life-span than the original PAN. The rationale here is to limit the validity of the credential, so that it is of limited use in the case of a compromise by an attacker. In transport ticketing, however, tokenised payments call into question the ability to blacklist dishonest users on the transport network. This is because the PAN is replaced with short-lived tokens that are variable. This variability makes it difficult for transport operators to pin down a user through blacklisting since the token is periodically renewed, in contrast to a long-term PAN. This variability in 'identity' exposes transport operators to attacks similar to the Sybil attack [101]. In the United Kingdom alone, over 200 million was lost in revenue due to fare evasion and other ticketing-related fraud [102] in 2016. Therefore the tokenisation of PANs poses a significant threat to both academic proposals and real-life implementations that rely on the PAN to identify or blacklist users.

### 5.1.2 Proposed Solution

As a solution to the problem of blacklisting, a transport ticketing scheme based on Linkable Group Signatures (LGS) is proposed [103, 104]. In simple terms, an LGS lets a verifier link the signatures of a user on different messages, while keeping the identity of the signer anonymous. This 'linkability' property is relied upon by the transport operator to blacklist dishonest users, regardless of their short-lived tokens. We also exploit the anonymity inherently provided by the LGS, which is an important user privacy requirement in transport

ticketing systems. The details on how the LGS works is given in Section 5.4

## 5.2 Privacy and Accountability

The quest for authentication and accountability in a privacy-preserving way is not a new phenomenon, and has applicability in different use-cases. Many cryptographic schemes facilitate the authentication of users anonymously. This offers users a high level of privacy; however, some use-cases such as ticketing also require accountability (i.e. the service provider is able to punish dishonest users).

There are different approaches to balancing authentication, privacy, and accountability [105]. The use of so-called *anonymous credentials* [106, 107] has been proposed by many authors to authenticate users anonymously, and if required, a dishonest user can be de-anonymised, thus revoking the user's privacy. There are also proposals based on group digital signatures, such as in [108–111] that facilitate anonymous authentication. One unifying factor with the aforementioned proposals is that they all require a Trusted Third Party (TTP) to de-anonymise or open the real identity of a dishonest user. This TTP is typically referred to as the Opening Authority (OA), or the Revocation Authority (RA).

However, in some scenarios, exposing the real identity of the user may be too harsh, or insignificant. For example, a Transport Operator may want to blacklist an offending user so further travel is not allowed until the money due is settled, but may not necessarily want to identify the user. In scenarios like this, linking of the users' transactions is enough to deter further usage of the transport system while still maintaining the anonymity of the user.

In addition, there are strict functional requirements in ticketing, in which the transaction is expected to be completed offline and within a limited time-frame. It is very challenging to meet these requirements using any solutions that rely on a TTP for accountability since there must be a connection to the TTP before a decision can be taken. Therefore, transport operators need a

way to blacklist users locally, i.e. offline and without the need for a TTP. We term this type of blacklisting 'Local Blacklisting'.

## 5.3 Related Work

There are numerous transport ticketing schemes proposed that include a blacklisting functionality, and have approached blacklisting from different perspectives. Blacklisting in these proposals can generally be divided into two categories; *privacy preserving* and *non-privacy preserving* blacklisting schemes. The non-privacy preserving schemes are straight-forward, and typically include a blacklist database made up of fixed User IDs or user-device specific IDs (UIDs). For example, the work presented in [8] blacklists users based on their transport ID, which is a device-specific ID. As pointed out by the authors, their solution does not provide user privacy since it is possible to track the movement of users based on their fixed UIDs from back-end systems. Fixed UID blacklisting not only goes against the privacy requirements of the user, but the existence of cloning mechanisms has proven this method to be rudimentary and ineffective [112–114]. The focus in this chapter is on privacy-preserving blacklisting schemes.

Transport ticketing schemes rely on cryptographic methods to achieve the blacklisting of users in a privacy-preserving manner. For example the work in [115] presents a privacy-preserving e-ticketing system that relies on the homomorphic properties of the underlying encryption algorithm used. However the performance testing presented by the author shows that the blacklisting solution only meets the functional requirement (see Section 5.5.1) with less than 1000 entries in the blacklist. This shows that the solution is not scalable, and is inefficient for real-life implementations since a blacklist will be expected to contain well above 1000 entries. Moreover, the authors did not take the delay caused by the physical NFC channel into consideration.

Arfaoui etal. in [116] proposed a privacy-preserving NFC pass for transport systems with a user blacklisting mechanism [116]. In their scheme, a user's actions on the transport network cannot be linked to his/her identity. The

authors' solution relied on the group signature proposed by Brickell et al. as Direct Anonymous Attestation (DAA) [117] to keep a user anonymous in a group of registered users. User blacklisting relies on an opening authority, who is tasked with revealing the identity of a dishonest user. However, their solution is based on pre-purchased tickets by pre-registered users and therefore is a closed-loop ticketing system.

Similarly, D. Quercia and S. Hailes in [118] proposed a ticketing scheme which is based on the e-cash blind signature proposed by Chaum in [119]. The pre-issued e-tickets are globally spendable, and in the case of overspending, users' anonymity can be revoked. The authors however did not provide any practical implementation, and therefore the efficiency of their solution remains questionable.

To the best of our knowledge, the only academic proposal that conforms to EMV and thus can be considered as an open ticketing scheme was proposed by Ekberg et al. in [120]. The authors proposed a mass ticketing scheme using NFC mobile devices. They also relied on the EMV infrastructure specifically. The PAN was used as the user identity on the transport network. The PAN is securely tied to a location at a particular time to create a 'tap', which is further used to determine the fare to be paid. However, the most relevant part of their proposal is their approach to blacklisting. The authors rely on a Certificate Revocation List (CRL) to blacklist dishonest users. In simple terms, a CRL is a list containing digital certificates that have been revoked before their actual expiry date. It is assumed that the terminals at transport stations will be updated with an up-to-date version of a CRL periodically. However, the use of CRLs for large-scale implementations has its challenges as the distribution of CRLs to all terminals is relatively expensive [121]. Also, there are concerns about the unmanageable size of a CRL, especially in mass ticketing where huge numbers of transactions are carried out. It will be challenging to update CRLs in an efficient way [122]. In addition, the strict timing requirements of

ticketing systems may be difficult to adhere to due to the long look-up times of CRLs.

It is also important to highlight an industry solution to the problem of blacklisting discussed in this chapter. The release of a tokenisation specification for mobile payments by EMVco in 2014 [64] was the motivation for research into the problem. In 2016, EMVco subsequently released an update of the specification with the addition of the Payment Account Reference (PAR). According to EMVco, the objective of the PAR is to:

*"re-introduce a relationship that already exists in the payment ecosystem today for Primary Account Number (PAN) post EMVCo Payment Tokenisation. PAR may be used to link transactions initiated on Payment Tokens with transactions initiated on the underlying PAN to support the needs of a variety of payment processing and value added services that rely on PAN prior to the introduction of Payment Tokenisation."*

Furthermore, among the permissible uses of the PAR given by EMVco is using the PAR for performing risk analysis. This affirms the importance of the problem of blacklisting, which this chapter attempts to solve. It is also worth noting that the solution in this chapter was proposed and published prior to the release of the PAR specification. In addition, it could be argued that using a PAR for blacklisting in ticketing goes against the privacy requirement of the user since each PAR is unique to a user, and is fixed for the lifetime of the account. Therefore the solution presented in this chapter is still superior to the PAR in terms of privacy preservation.

## 5.4 Cryptographic Background

This section provides a background to the cryptographic building blocks used in the proposed scheme. The main cryptographic building block used is a special type of group digital signature with a 'linkability' property. We also provide a detailed explanation of the different phases involved, as well as the intractability assumptions the signature is based upon.

### 5.4.1 Linkable Group Signatures (LGS)

Group signatures as first proposed by Chaum et al. in [108] allow any member of a particular group to generate signatures anonymously. The verifier gets cryptographic assurances that a legitimate member of the group signed the message without revealing the signer's identity.

Group signatures with different properties have been proposed in the literature. In this study, the LGS first proposed in [104] (referred to as a list signature) and standardised by ISO/IEC in [103] is used. In its original construction, it supports the linking of signatures provided they were signed using the same linking tag. In [104] a time frame was used as the linking tag, allowing the linking of all signatures generated by a user within a given time frame. However, [103] shows the linking tag can also be any random value, as long as it is constant. This signature also supports both private key revocation and verifier blacklist revocation. In the next section, we give an overview of the processes involved in this signature. For a detailed outline of the process and mathematical proofs, please refer to [103, 104].

### 5.4.2 Intractability Solutions

#### 5.4.2.1 Strong RSA Assumption

First introduced in [123]; Let $p'$ and $q'$ be two distinct primes of equal length such that: $p = 2p' + 1$ and $q = 2q' + 1$ are also primes. The multiplicative

| Notation | Meaning |
|---|---|
| $\boldsymbol{T_k}$ | Token generated from the PAN |
| $\boldsymbol{t_{nt}}$ | Timestamp at the point of entry |
| $\boldsymbol{t_{xt}}$ | Timestamp at the point of exit |
| $\boldsymbol{R_n}$ | Random nonce |
| $\boldsymbol{stn_{id}}$ | Unique train station identity |
| $\boldsymbol{Sig_x}$ | linkable signature with key x |
| $\boldsymbol{bsn}$ | linking base |
| $\boldsymbol{T_4}$ | linking tag |
| $\boldsymbol{A, e, x}$ | signature key |
| $\boldsymbol{G_{pk}}$ | group public key = $(n, a, a_0, g, h, b)$ |
| $\boldsymbol{G_{mk}}$ | group membership issuing key = (p', q') |
| $\boldsymbol{CHALL}$ | $\{t_{nt}/t_{xt}\|\|R_n\|\|stn_{id}\}$ |
| $\boldsymbol{Tap}$ | $\{t_{nt}/t_{xt}\|\|R_n\|\|stn_{id}\|\|T_k\|\|T_4\}$ |

Table 5.1: Notations and Meanings

group of quadratic residues modulo $n$ denoted by $QR(n)$, is a cyclic group of order $p'q'$, where $n = pq$, and is referred to as the safe RSA modulus.

#### 5.4.2.2 Decision Diffie-Hellman Assumption(DDH)

Let $g$ be the generator of a finite cyclic group $G$. The DDH assumption [124] for group G states that it is hard to distinguish the DDH tuple: $(g^x, g^y, g^{xy})$ from random triples $(g^x, g^y, g^z)$, for a random $(x, y, z)$ modulo the order of group $G$.

In general, the DDH problem can also be constructed for arbitrary finite abelian groups. Therefore, if $G = QR(n)$, then $G$ has composite order. If the group composition of $G$ is known, then the DDH problem in $G$ is reduced to the DDH problem in the components of G.

### 5.4.3 Linkable Group Signature Processes

The phases involved in the usage of an LGS are described below, the notations used in [103] are maintained, and their meanings are given in Section 5.4.2.2.

1. **Key Generation Process** The key generation is made up of two parts:

set-up phase and the group membership issuing phase. In the set-up phase, the group manager creates the group public parameter, $G_{pk}$, and $G_{mk}$. The group membership issuing process is a protocol run between the group manager and a group member to create a unique signature key $(A, e, x)$, where $(x)$ is the private key and $(A, e)$ are the group membership certificate for each group member. We assume the presence of a secure channel between the group manager and the group member.

(a) Set-up Process: We assume the existence of two hash functions H: $\{0,1\}^* \rightarrow \{0,1\}^k$ and $H_\Gamma : \{0,1\} \rightarrow \{0,1\}^{2lp}$. The group manager chooses the group public parameters: $(l_p, k, l_x, l_e, l_E, l_X, \epsilon)$. The group manager also chooses random generators: $(a, a_0, g, h, b)$ of QR(n). $G_{pk}$ = $(n, a, a_0, g, h, b)$ and $G_{mk}$ = (p', q')

(b) Group Membership Issuing Process: At the end of this phase, the member knows a random $x \in [0, 2^{lx} - 1]$ and the group manager knows $a^x$ mod n and nothing more. Then the group manager chooses a random prime $e \in [2^{lE} - 2^{le} + 1]$ and computes A = $(a_0 C_2)^{d1}$ mod n where $C_2 = a^x$ mod n and $d_1 = 1/e$ mod n. The group manager sends $A$ and $e$ to the member. The member checks that $A^e = a_0 a^x$ mod n. The group member signature key is $(A, e, x)$ and $x$ is the private key.

2. **Signing Process:** The signature process takes as input: the *($G_{pk}$)*, the group member's signature key $(A, e, x)$, a *linking base (bsn)* and the message to be signed and outputs a linkable signature $Sig_x$.

**Algorithm 1** Signing

1: Compute f = $(H_\Gamma(bsn))^2$ (mod n)
2: Chooses random integers: $w_1, w_2, w_3 \in [0, 2^{2lp} - 1]$
3: Compute: $T_1 = Ab^{w1}$ (mod n),
$T_2 = g^{w1}h^{w2}$ (mod n),
$T_3 = g^e h^{w3}$ (mod n),
$T_4 = f^x$ (mod n).
4: Choose random integers:
$r_1 \in [0, 2^{\epsilon(le+k)} - 1]$,
$r_2 \in [0, 2^{\epsilon(lx+k)} - 1]$,
$r_3, r_4, r_5 \in [0, 2^{\epsilon(lp+k)} - 1]$
5: Choose random integers: $r_9, r_{10} \in [0, 2^{\epsilon(2lp+le+k)} - 1]$
6: Compute: $d_1 = T_1^{r1}/(a^{r2}b^{r9})$ (mod n)
$d_2 = T_2^{r1}/(g^{r9}h^{r10})$ (mod n)
$d_3 = g^{r3}h^{r4}$ (mod n)
$d_4 = g^{r1}h^{r5}$ (mod n)
$d_5 = f^{r2}$ (mod n)
7: Compute:
c = H$(a||a_0||g||h||T_1||T_2||T_3||T_4||d_1||...d_5||m)$
$s_1 = r_1 - c(e - 2^{lE}), s_2 = r_2 - c(x - 2^{lX})$,
$s_3 = r_3 - cw_1, s_4 = r_4 - cw_2$,
$s_5 = r_5 - cw_3, s_9 = r_9 - cew_1$,
$s_{10} = r_{10} - cew_2$
8: Set the signature as:
$Sig_x = (c, s_1, s_2, s_3, s_4, s_5, s_9, s_{10}, T_1, T_2, T_3, T_4)$

3. **Verification Process:** The verification process takes as input a message, *bsn*, a linkable signature $Sig_x$, and $G_{pk}$ corresponding to the group of the signer. It returns 1 if the signature is *VALID*, otherwise it returns 0.

**Algorithm 2** Verification

1: Compute:
f = $H_\Gamma(bsn))^2$ (mod n)
$t_1 = a_0^c T_1^{s1-cl'}/(a^{s2-cL}b^{s9})$ (mod n) where $l' = 2^{lE} and L = 2^{lX}$
$t_2 = T_2^{s_1-cl'}/(g^{s9}h^{s10})$ (mod n) where $l' = 2^{lE}$
$t_3 = T_2^c g^{s3} h^{s4}$ (mod n)
$t_4 = T_3^c g^{s1-cl'} h^{s5}$ (mod n) where $l' = 2^{lE}$
$t_5 = T_4^c f^{s2-cL}$ (mod n) where $L = 2^{lX}$
2: Compute:
c' = H($a||a_0||g||h||T_1||T_2||T_3||T_4||d_1||d_2||d_3||d_4||d_5||m$)
3: If
c'= c, $s_1 \in [-2^{le+k}, 2^{\epsilon(le+k)} - 1]$,
$s_2 \in [-2^{lx+k}, 2^{\epsilon(lx+k)} - 1]$,
$s_3 \in [-2^{2lp+k}, 2^{\epsilon(2lp+k)} - 1]$,
$s_4 \in [-2^{2lp+k}, 2^{\epsilon(2lp+k)} - 1]$,
$s_5 \in [-2^{2lp+k}, 2^{\epsilon(2lp+k)} - 1]$,
$s_9 \in [-2^{2lp+le+k}, 2^{\epsilon(2lp+le+k)} - 1]$,
$s_{10} \in [-2^{lp+le+k}, 2^{\epsilon(2lp+le+k)} - 1]$ return 1 (valid signature) else return 0 (invalid signature)

4. **Linking Process:** The linking process takes two valid linkable signatures and determines if they are linked, i.e. if they were signed by the same user.

**Algorithm 3** Linking

Takes two **valid** linkable signatures:
$(c, s_1, s_2, s_3, s_4, s_5, s_9, s_{10}, T_1, T_2, T_3, T_4)$ and
$(c', s'_1, s'_2, s'_3, s'_4, s'_5, s'_9, s_{10'}, T'_1, T'_2, T'_3, T'_4))$ If $T_4 = T'_4$ output 1 *i.e* they are linked, otherwise 0

5. **Revocation Process:** The original construction of the signature supports two types of revocation: *Private Key Revocation* and *Verifier Blacklist Revocation*. In this proposal, we focus on the verifier blacklist revocation. To blacklist a user, the verifier generates a blacklist using the corresponding $T_4$ of the user. Therefore the verifier can check if the user is on the blacklist by first verifying the signature as valid, and then carrying out the blacklist check as follows: for each $T_4$', check $T_4$' $\neq T_4$. If any of the checks fail, output 0 (revoked), otherwise, output 1 (valid).

## 5.5 Proposed Ticketing Scheme

In this section, we outline the general architecture of our proposed model. We define the entities involved, as well as the roles they play in Section 5.5.5. We also highlight the general assumptions made which are necessary for our proposed model in Section 5.5.4. The proposed ticketing scheme is an account based, open-ticketing scheme, that is based on the 'Aggregated Pay As You Go' model [16] proposed by the UK Cards Association (UKCA) which intended for EMV cards and NFC-enabled mobile devices[1]. As mentioned in Chapter 2, ticketing requirements always depend on the specifics of its implementation, and also open ticketing systems have fewer requirements than closed ticketing systems. Therefore the requirements presented in Sections 5.5.1 and 5.5.2 respectively, represent a subset of the ticketing requirements presented in Section 2.4. The scheme is divided into different but interconnected phases as presented in Section 5.5.6.

Figure 5.1 shows the sequence of messages exchanged in the proposed protocol.

### 5.5.1 Functional Requirements

1. Offline Transaction: transport ticketing systems are required to complete transactions offline, i.e. without requiring internet connection to instantly to carry out back-office validations.

2. Efficiency: Transport ticketing systems should be very efficient in terms of passenger throughput. Therefore they are required to produce transaction speeds of 300 – 500 milliseconds [32, 33] from the time the user taps the device to the time the terminal grants or rejects access.

[1]Prior to using the scheme, it is assumed that a user already has a payment application that conforms to the EMV specification already provisioned to the device.

### 5.5.2 Security Requirements

The security requirements of open tickets are less that the requirement of closed-loop tickets because the logic is in the back-office. Nevertheless, the security requirements of the protocol proposed in this chapter are presented below:

1. Integrity: It should be possible to verify whether a wrong ticketing credential is used. There should also be cryptographic evidence binding the user's transaction to a location at a particular time.

2. Anonymity: Although more of a privacy concern, the identity of the user of a transport system must not be revealed.

3. Exculpability: It should be impossible for any entities, including the group manager, to falsely accuse a user of making a transaction at an entry or exit point on the transport network.

4. Blacklisting: It should be possible to build a blacklist of dishonest users (or compromised devices), and be able to deny them further use of the transport network.

### 5.5.3 Adversary Model

The motivation for an adversary here is to abuse the 'first time travel risk' (see Section 2.3.4), by maliciously evading detection on the blacklist. The adversary could either be an attacker in possession of a stolen device, or a legitimate user trying to cheat the system. A determined attacker will try to avoid the blacklisting mechanism by producing a signature with a fake linking tag. According to the described Adversary model, we list the presumptive capabilities of the attacker below:

1. The attacker cannot break the linkable signature algorithm used in this chapter.

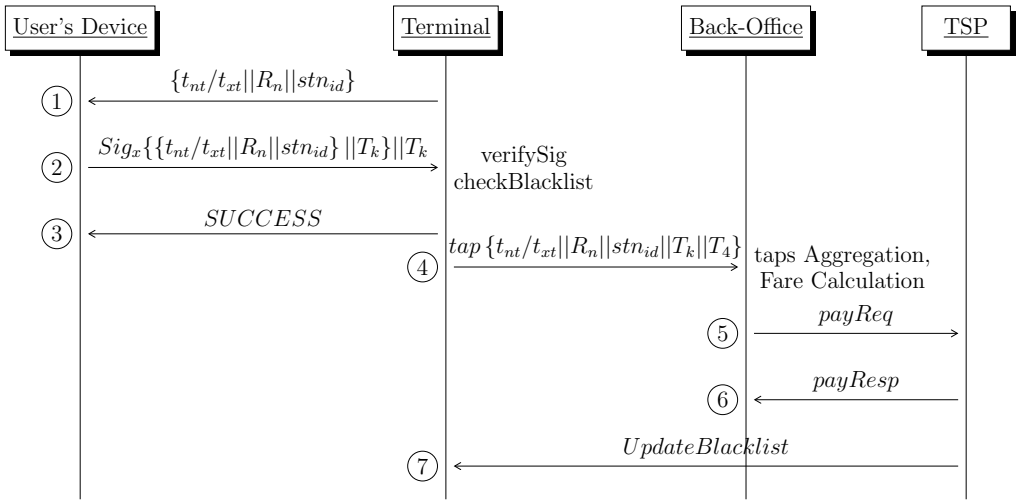


Figure 5.1: Protocol Diagram of the Proposed Solution

2. The attacker is active, and can generate fake tokens and linking tags.

3. The attacker has access to the payment device, as well as a legitimate signing key.

### 5.5.4 Assumptions

1. The transport application, and credentials including cryptographic material, shall be provisioned to the user's device using secure best practices such as GlobalPlatform.

2. The payment networks act as a TSP, and shall subject users to necessary identification and verification (ID&V) prior to issuing new tokens.

3. Each user is part of a group of users depending on their payment network. For example, all MasterCard users are part of the same group.

4. The validity of the tokens used is at least 24 hours. This is a configurable value, in reality, the validity of the token will be based on the perceived residual risk of exposure on the users' devices.

5. We assume the security features available via the platform/OS out of

the box, will be in place to store tokens, keys, and other cryptographic material.

6. The user is in possession of an NFC device with a payment application used for regular tokenised payments such as retail.

7. There is mutual trust between the TrO and the rest of the EMV ecosystem, and the terminals will be provided with the group public keys of the payment network.

8. Each train station has terminals at entry and exit points. Terminals are NFC readers equipped with turnstiles to grant or deny entry to users.

### 5.5.5 Entities

In the following subsections, the functions of the entities that make up the architecture of the proposed model are described.

#### 5.5.5.1 User

A user in this context will already have a bank account and possibly a bank card. The user also has a NFC-enabled mobile device as well as a payment application provisioned to the device.

#### 5.5.5.2 Terminal

The terminal owned and managed by a TrO, and is used to validate that users have valid tickets on their devices before travelling. The taps made on terminals at each station are collected and are periodically sent to the back-office for processing. The TrO also maintains a blacklist of dishonest users in cases where the user has insufficient funds or in the case of a compromise. The terminals are periodically updated with the most up-to-date blacklist, to ensure they don't allow blacklisted users travel.

#### 5.5.5.3 Back-Office

The back-office provides fare management and accounting services on behalf of the TrOs. The back-office collates and aggregated all the taps from the terminals, to establish the right fare to charge users. The back-office also the initiates the process of authorisation via the TSP to recover funds from users' account. The back-office also manages user-blacklisting, and periodically updates the terminals.

#### 5.5.5.4 Token Service Provider (TSP)

The TSP provides card management services including:

1. Digitising Physical card PANS into tokens.
2. Perform the necessary identification and verification (ID&V) of users prior to issuing new tokens.
3. Initial and subsequent provisioning of tokens to users' devices.
4. Translation of tokens back to PANs, to facilitate authorisation of funds from the users' accounts.

For the ticketing solution presented in this chapter, we assume the payment networks will act as the TSP. The rationale behind the decision to use the payment network as the TSP is that globally there are fewer payment networks than banks[2]. Therefore this means that the TO's terminal will have to keep a few group public keys for signature verification.

### 5.5.6 Phases

Our solution is divided into 4 phases: set-up, validation, accounting, and the blacklisting phase. The specifics of the accounting phase are beyond the

[2]The EMVco specification on tokenisation indicates that the payment networks can additionally act as the TSP, while still maintaining their primary roles in the EMV ecosystem.

scope of this chapter. It is however important to mention as it precedes the blacklisting phase.

#### 5.5.6.1 Setup Phase:

This phase is executed between a user and the payment network. A user initiates this phase by opting to use the payment application on his device for transport payments. They both engage in an ID&V process to verify the user's identity and bank account, and check if the user's has any outstanding transport fares. The process is terminated if any of the checks fail. Otherwise they go through the key generation process as explained in section 5.5.6. In the end, the user will have a unique signature key; $(A, e, x)$, a token $(T_k)$, and a TrO-specific $(bsn)$ securely stored on their device. We assume the TSPs group public keys to be well known and are provided to the TrOs well before hand.

#### 5.5.6.2 Validation Phase:

This phase is illustrated in Figure 5.1. We see that a user taps their device on a terminal at a train station, the terminal sends a challenge to the device as shown in **Message 1**. The challenge includes; timestamp $(t_X)$, random nonce $(R_n)$ and the station ID $(stn_{id})$. The device concatenates the token $(T_k)$ to the challenge, and signs as explained in the signing phase in section 5.5.6, using the $(bsn)$ of the TrO. The $(t_x)$ could either be $(t_{nt})$ or $(t_{xt})$ for entry and exit gates respectively. The device the concatenates $(T_k)$ to the signed message and sends to the terminal in **Message 2**. The terminal verifies the signature using $(G_{pk})$ as outlined in verification phase in section 5.5.6. If the signature is valid, the terminal checks to see if the user's $(T_4)$ is included in the blacklist. If it doesn't correspond with any $(T_4)$ on the blacklist, the user is allowed to travel otherwise the user is denied travel **(Message 3)**. Afterwards, the terminal records a $TAP$. A $TAP$ includes the challenge signed by the user's

device, the ($T_k$), and the amount to be charged which is determined in the *accounting phase* below. The blacklist check is only needed at the entry gates.

#### 5.5.6.3 Accounting Phase

This is a back-office process where taps of all users for the day are aggregated to determine the fares to be paid by the users. The office received all the taps made by users from the terminals as shown in **Message 4**. The back-office also initiates the process of payment authorisation by sending a payment request (payReq) in **Message 5**, which includes the ($T_k$) and the amount to be charged, to the TSP for authorisation. The payment network, acting as the TSP, translates the token back to a real PAN and authorisation is processed via the users' issuing bank as per normal EMV flow. The details of the payment authorisation is beyond the our scope.

#### 5.5.6.4 Blacklisting Phase

This phase only becomes necessary in cases where an authorisation fails due to insufficient funds in the user's account. The TSP sends a *transaction decline* message to the TO. The TrO then puts the user's $T_4$ in its blacklist database and updates the terminals at the stations with the latest blacklist entries. A user's device can also be put in the blacklist in the case of compromise or a lost device.

### 5.5.7 Proof of Concept

A Proof of Concept (PoC) was developed to test the feasibility of our proposal and also analyse it against the requirements mentioned in section Section 5.5.2. A HCE-based Android application was installed on an NFC device for digital signature implementation. We adapted an implementation of the digital signature in [125] which was part of an analysis of group signatures on mobile devices [126]. For the terminal, we had a Java application using the smartcard

I/O API running on a PC; this acted as the terminal at a train station.

| Device | Manufacturer | Operating System | RAM |
|---|---|---|---|
| Phone (Nexus 5) | LG Electronics | Android 5.1.1 (Lollipop) | 2GB |
| Laptop | Dell | Windows 10 | 8GB |
| Reader | ACS(ACR1281U) | N/A | N/A |

Table 5.2: Devices Used in Proof of Concept

### 5.5.8 Lessons Learned and Considerations

Support for extended Application Protocol Data Units (APDUs)[3] on NFC devices is still not as extensive as that for smartcards. Therefore most NFC devices can only send normal APDUs with a maximum length of 256 bytes. We realised this was a software-based restriction rather than the NFC controller's inability to handle bigger messages. Due to the size of the signature in our protocol, we modified the Android source to allow the device to send back the signature in one APDU, rather than in chunks.

Sending a substantial amount of data over the NFC channel may not always be efficient. Due to the size of the signature we used, we realised the time cost of compressing the message and decompressing at the terminal's side is trivial. We found the *BZip2* compression algorithm to be the most efficient.

Most of the parts of the signature can be precomputed; that is those parts that do not depend on a challenge from the terminal.

### 5.5.9 Performance Analysis

The total size of $Sig_x\{\{t_{nt}/t_{xt}||R_n||stn_{id}\}\,||T_k\}||T_k$ is 3552 bytes, with a 512-bit key, including 16 bytes for concatenating the token to the signature in plain text. This is compressed to 1617 bytes, providing 45.7% compression.

We took average timings of individual processes, as well as the total time

[3]Application Protocol Data Unit is the unit of communication between a device and a terminal. APDUs are specified in ISO/IEC 7816.

| | CHALL | Sign | Verify | Full Protocol |
|---|---|---|---|---|
| **Average** | 420.75 | 9.92 | 20.5 | 451.17 |
| **Min** | 405.55 | 7.65 | 17.32 | 430.45 |
| **Max** | 445.63 | 10.65 | 23.75 | 480.03 |

Table 5.3: Protocol Transaction Times in Milliseconds

it took the full protocol to run over 100 iterations. The mobile device takes an average of 9.92ms to sign the challenge received from the terminal and also compress the signature. The *Round Trip Time* (RTT), i.e. the time from when the terminal sends the challenge to when it receives the response, takes on average 420.75ms. We refer to this as $CHALL$. It is important to note that about 90% of the RTT is spent on the NFC communication link. The signature verification on the terminal side, including decompression of the received data, takes 20.5ms on average. The whole protocol takes on average 451.17ms.

For performance measurements when checking the blacklist, we relied on a comparative study of Database Management Systems (DBMS) in [127]. Each DBMS was populated with 1,000,000 records, and the timings for a *'select'* query for each was taken. The select query emulates the look-up of a user's ($T_4$) from a blacklist database. The SQL Server was the fastest and took 18ms, while the slowest was Oracle which took 23ms. These projections show that the delay introduced by searching a blacklist is trivial and therefore, our protocol still runs within the accepted transaction time range for transport usage.

### 5.5.10 Requirements Analysis

We analysed our proposal against both the security and functional requirements mentioned in section Section 5.5.2. Our model meets the *offline verification* requirement because the terminal is able to verify a signature, as well as run the blacklisting function offline, i.e. without connecting to a back-office

or relying on a third party. The protocol, as shown in Section 5.5.9, is within the acceptable transaction speed range, as stipulated by the *efficiency* requirement. It is worth noting that, currently, NFC devices in HCE only operate at the lowest NFC data rate of 106kbps[4]. We found out this limitation is also a software limitation and not the NFC controller's inability to operate at higher data rates. Therefore at higher rates, our solution is expected to be much faster.

In terms of security, for a signature verified to be valid, it is computationally hard for anyone other than the group manager to reveal the identity of the actual signer. In the random oracle model, the proof of knowledge that is part of the signature can be proven in statistically zero knowledge. Also trying to identify a particular signer with certificate $(A, e)$ requires the adversary to know if $\log_b T_1/A, log_g T_2$, and $log_g T_3/g^e$ are equal. This is assumed to be infeasible under the decisional Diffie-Hellman assumption. Therefore our protocol meets the *anonymity* requirement.

As shown in the key generation phase in Section 5.5.6, the group manager does not learn any new information about the user's private key $(x)$, and at the end of the phase, the group manager only learns $a^x$. Also, because (T1, T2 and T3) represent an unconditional binding commitments to ($A$ and $e$). This implies that if the factorisation of $n$ is feasible, the group signature is a proof of knowledge of the discrete logarithm of $A^/a0$ [109]. Therefore no entity, including the transport operator and the TSP – acting as the group manager, can sign a message on behalf of a user as computing a discrete logarithm is assumed to be infeasible. Therefore our protocol meets the *exculpability* requirement because a user cannot be framed for a false transaction.

In addition, *integrity* is achieved because it is not possible for anyone without access to the private key $(x)$ to generate a valid signature. Secondly, the TrO is able to verify that the signed message includes the correct challenge

[4]NFC supports data rate of 106, 212, 424, and 848kbps

it had sent, thereby cryptographically linking the user to that point on the transport network at that particular time, creating the tap.

User *blacklisting* is achieved because a legitimate user cannot avoid detection on the blacklist by forging a false linking base. The $(T_4)$ is linked with $(T_1)$ through the proof of knowledge and also the private key $x$. In addition, a legitimate user cannot repeatedly cheat the system by signing on a rogue token with a legitimate credential, because after the first payment request is declined, the TrO can blacklist the user with the corresponding $(T_4)$.

## 5.6 Summary

The work presented in this chapter looks at how tokenisation affects the unique identification of users in certain scenarios. In particular, how tokenisation calls into question user blacklisting in transport ticketing has been highlighted. It has been shown how LGS can be used to link two transactions regardless of the changing token. This concept is used to create a blacklist of dishonest users.

The feasibility of the solution has been tested by building a PoC which is analysed against the outlined functional requirements. The solution can also be used in use-cases outside of ticketing that rely on the static nature of PANs. For example, it can be used in retail to link different transactions of a user (with different tokens) for loyalty and promotional purposes.

# Chapter 6

# Ecosystems of Trusted Execution Environments

A TEE is a secure and trusted area of the main processor that provides memory protection, and secure execution for applications (or part of an application) on a mobile device. The TEE ensures isolation and integrity protection from any applications running in non-TEE environments of the same device. These security services make TEEs suitable candidates to increase the attack-resistance of HCE applications. HCE applications can rely on the TEE to isolate their cryptographic operations and keys from the rest of the 'untrusted' applications. This idea has been widely suggested in many publications [68, 128, 129]. However, access to the TEE is not open to users, service providers, or third-party application developers, due to both commercial and security reasons. The OEMs restrict the TEE and its functionality for their own internal use, or for the selected applications they approve [130, 131]. The TEE suffers from the same conundrum as the SE, and current implementational models cannot be fully utilised by HCE applications, as it is widely hypothesised.

In this chapter, an outline of TEE as a technology and the security services they provide is given. A brief description of two of the most prevalent TEE implementations on the market is given. As a solution to the issue of lack accessibility to TEEs, two possible ecosystem models for TEEs —Security as a Service and Consumer-Centric models— are proposed. These models,

alongside the existent model of TEEs (centralised model) are then compared according to a set criteria to identify the advantages as well as the shortcomings of each model.

## 6.1 Trusted Execution Environment

A Trusted Execution Environment (TEE) represents a set of software and hardware components relied upon to provide a secure storage and execution environment on devices. The TEE where applicable, forms the backbone of the device's Trusted Computing Base (TCB). The TCB of a device is a collection of all software, firmware, hardware components that are explicitly trusted to enforce security on a device. Typically, TEEs are logically separated from the Operating System(OS), applications, and other peripherals running in the normal execution environment of the device, thereby taking them out of the TCB . This provides the notion of "Trusted Execution Environment" and "Rich Execution Environment"(REE) for the trusted/secure and untrusted sides of the device respectively. There have been efforts to standardise TEEs by GlobalPlatform [132] through its "Device Committee", and have published a set of specifications. The specifications do not insist on any specific architecture, but specify the software [133] and hardware configurations [134] that make up a TEE. It also highlights various components that could be used to host a TEE, as well as resource sharing and memory management between the Rich OS and the Trusted OS. In the coming section, a description of the generic services provided by a TEE is given.

### 6.1.1 TEE Security Services

1. Isolated Execution: A TEE should have the ability to execute a piece of code in complete isolation from the other applications running on the device (including the main OS. This offers *confidentiality* and *integrity* of the code and related data at "run time". For example, a payment application on a mobile device should be able to generate digital signatures, without other applications observing the process or having access to the signing keys.

2. Secure Storage: A TEE should also provide assurances with regards to the integrity, confidentiality and in most cases the freshness of data at rest for the applications. For example, applications such as transit and payment make use of sensitive data such as cryptographic keys, passwords and primary account numbers (PANs) that must be protected against unauthorised access at all times.

3. Remote Attestation: Attestation, means "to vouch for something". *Remote attestation* is the ability to vouch for an entity or its characteristics to a third party. In the context of TEEs, it is used to give assurances to a service that a piece of software or OS is trustworthy. For example, in mobile payments, the issuer can get assurances through remote attestation on the integrity of the banking application running on the device. A remote attestation protocol normally begins with an integrity measurement of the TCB as a whole. This is done by taking a digest of the code, for example by using a hash function, and signing it with a key, stored preferably in a hardware RTS. A third party can verify the signature and compare it against a list of trusted OS hashes.

4. Secure Provisioning: Secure provisioning is the ability to send data to a device or a particular software component running on the device in a secure fashion. This gives service providers a way to provision applications and make relevant updates. For example, in payment applications, banks require a secure mechanism to deliver the application to the device of its user. There must be assurances on the integrity of the application as well as the confidentiality of other cryptographic data.

5. Trusted Path: A TEE should also provide a secure input/output mechanism with which users can interact with applications on the device. For example, payment applications often require user verification through PINs or biometrics. The TEE should provide a trusted path between the

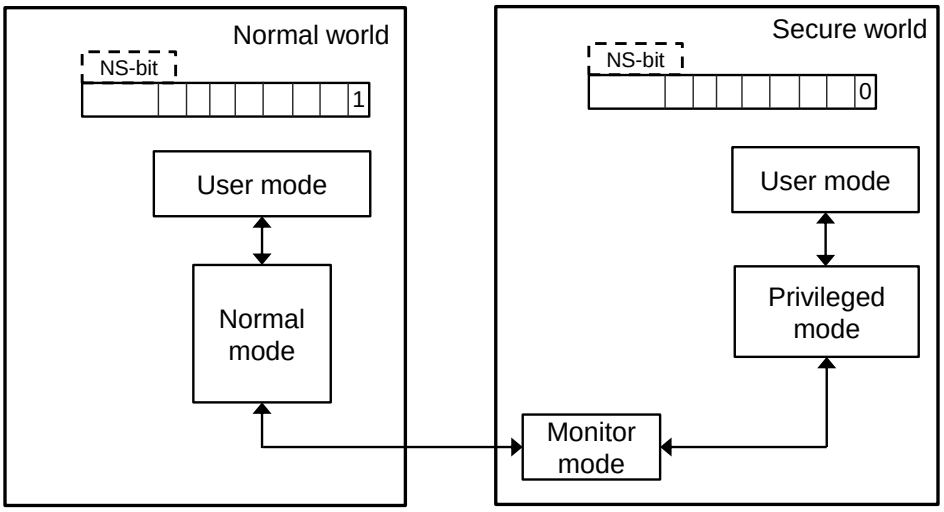


Figure 6.1: Architecture of TrustZone on a Mobile Device [2]

application and the keypad. This provides protection against malware on the device, which normally might be able to access sensitive data entered by user by logging keystrokes, or by more sophisticated methods such as sniffing data on the physical communication layer. Globalplatform has published specifications [135] for providing a trusted path through a TEE on mobile devices.

### 6.1.2 ARM's TrustZone

TrustZone is ARM's security technology solution provided by the more recent ARM processors. TrustZone achieves security by logically separating hardware and software resources into two distinct modes, referred to by ARM as the "secure mode" and "normal mode"(non secure). The resources in the secure mode cannot be accessed by the resources in the normal mode.

#### TrustZone Hardware Architecture

The secure/normal world separation is extended to the hardware components, and this is enforced by a hardware logic in the TrustZone-enabled AMBA3

AXI bus fabric [2]. The AMBA3 AXI bus provides an extra *control signal*, known as the *NS bit*, to read and write channels which are on the main system bus. If NS-bit = 0, then the processor is in the secure mode; otherwise, if NS-bit = 1 it indicates the processor is operating in the normal mode. This transition between the two modes is controlled by a mechanism referred to as the *monitor mode* as shown in Fig 6.1.

TrustZone also secures peripherals such as interrupt controllers, timers, and I/O components through the AMBA3 APB peripheral bus. This ensures that the system is monitored by a task that cannot be interrupted by malware. The AMBA3 APB peripheral bus is a low gate-count and low-bandwidth peripheral bus that is connected to the system bus using an AXI-to-APB bridge. This bridge controls access to the peripherals, ensuring only requests with the necessary security status reach the peripherals.

The processor switches between the two modes (context switching) in a *time-sliced* fashion and is controlled by the "monitor mode". For an application running in normal mode to make use of services offered by another application in the secure mode, for example to encrypt a document, a special instruction known as the "Secure Monitor Call (SMC)" is used. SMC immediately sets the NS-bit to '0' and then fully transfers control to the secure mode. On the other hand, switching back from secure mode to normal mode is less controlled, and the secure side can directly alter the "Current Processor Status Register". So, essentially, the monitor mode can be seen as providing *gate-keeping* services between the two modes.

### 6.1.3 Intel Software Guard Extensions (SGX)

Intel's SGX are a set of instructions that provide extensions to the Intel architecture processors. These extensions provide a TEE within the computer's untrusted environment. Intel SGX thus aims to provide confidentiality, integrity, and replay protection using *enclaves* [3].

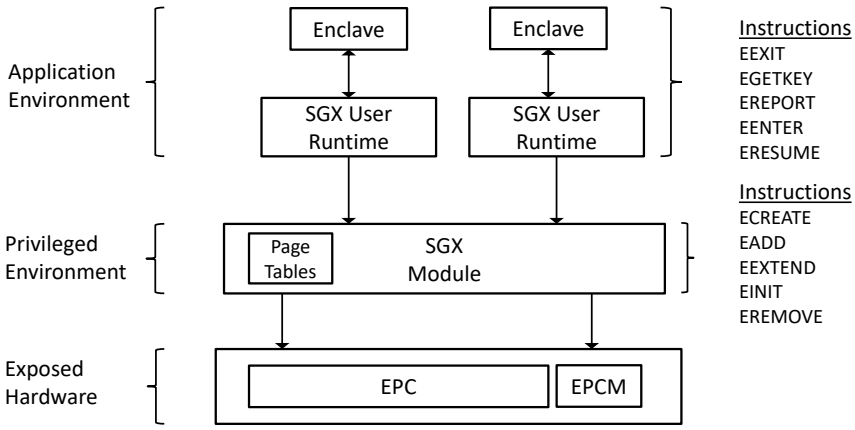


Figure 6.2: Hardware and Software Architecture of Intel SGX [3]

An enclave is part of the applications memory space which is hardware protected. An enclave has a reserved area of memory from which it runs, known as the enclave page cache (EPC). Access to the EPC is protected from processes outside of the enclave. The OS can manage the enclaves, but cannot tamper with the code and data within the enclave itself.

The management of an enclave is carried out through a set of instructions. The *ECREATE* instruction creates an enclave and also sets the base linear address as well as the physical address. After the creation of an enclave, the *EADD* instruction is used to add relevant code and data to the enclave. This adds 4KB of protected data. For integrity protection, the *EEXTEND* instruction is used to measure the contents of the enclave. This measures 256 bytes at a time; therefore, to measure the whole contents of the enclave, the instruction is called 16 times [3]. The enclave is then initialised using the *EINIT* instruction. This sets the INIT attribute to true, which means the code in the enclave is ready for execution.

## 6.2 Ecosystem Models for TEE

In this section, the different ecosystem models for TEEs are highlighted.

### 6.2.1 Centralised Model

This model represents the current state of TEE owenership models. In this model, the OEM has full control of the TEE service provisioning. At the core, this is the same ecosystem as the Issuer-Centric Ownership Model in smart-cards [136]. In the centralised model, only "trusted applications" are allowed to be hosted on the TEE. These trusted applications in reality represent a monopoly by the OEM. For example Samsung's mobile payment solution — SamsungPay— makes use of the TEE on Samsung devices, when it is not possible for any other service provider to do the same. From a security point of view, this model is ideal, however these restrictions limit the potential of mobile services.

### 6.2.2 Security as a Service Model

The concept of Anything As a Service (*aaS) is arbitrarily used typically in cloud computing to refer to something that is provided to end users as a service through the cloud [137]. For example Back-end as a Service (BaaS) and Data as a service (DaaS). Using the same concept in TEEs, SaaS is a model were the OEM still maintains ownership and manages the TEE, but provides security services to the applications of service providers on a contract basis. As an example, instead of hosting a ticketing application on the TEE, it's hosted in the normal environment of the device, while the TEE does its cryptographic operations such as digital signatures, and secure key storage.

### 6.2.3 Consumer-Centric Model

This model gives full control of the TEE services, and application provisioning to the user, therefore, it's the most flexible of the three models. This model is similar to the User-Centric Ownership Model (UCOM) for smartcards and a potential consumer-centric model proposed by GlobalPlatform [138]. This model is also similar to the current application provisioning mechanism deployed in the smartphone industry, where a user can install or delete any application as they desire. The user in this model has the privilege of enrolling, evoking or blocking access to any application that requests the TEE services. From the liability point of view, the applications should manage their own risk mitigation processes and users (or the OEM) might not be liable.

## 6.3 Comparison Between Models

In this section, we set comparison criteria on which we will later compare and contrast different ecosystem proposals for TEE.

### 6.3.1 Comparison Criteria

The rationale for constructing these criteria is to illustrate the positives as well as the shortcomings of individual ecosystem models discussed in previous sections. Economic considerations may decide the choice of model among the competing models, however, this does not mean that it is the best possible model. Nevertheless, the following criteria are based on the potential elements that might play a critical role in successfully bridging the transition of smartcard services from specialised hardware (smartcard) to smartphones.

1. Market Segmentation: Is the proposed model restrictive to a level that it might create pockets of market access? In market segmentation, certain applications might only be available on particular devices due to the business relationship between service providers and OEMs.

2. Scalability: The model enables a wide scale deployment of a number of applications, from heterogeneous service providers, therefore serving applications with varied and potentially changing requirements.

3. Flexibility: The model is flexible in a sense that a small service provider can also gain access to TEE services like a big corporation can.

4. Impartiality: The model does not discriminate any particular or set of service provider(s). Any service provider that abides by the security and privacy policies set by the model is allowed to access TEE services.

5. Consumer-Involvement: Does the consumer (users) have any involvement in either provisioning or evoking the TEE services to individual

service providers?

6. Open Provisioning: Any service provider can create an application requiring access to the TEE services and potentially use these services without requiring expansive and potentially long approval processes set by OEMs.

7. Closed Provisioning: The provisioning of applications is solely decide by a central authority, the OEM for example.

8. User Privacy: A set of applications on a smartphone might also signify potential privacy information about a user. Therefore, does the model reveal the set of applications using the TEE services to an external entity including OEMs, and/or other service providers?

9. Application Intellectual Property (IP) Protection: The model does not require the service provider to reveal the source code of their application to the entity that provisions the access to the TEE services.

### 6.3.2 Comparison and Discussion

Based on the defined criteria in the previous section, the comparison is shown in Table 6.1.

For the SaaS model, most of the partially met criteria is due to the nature of the model. It does provide flexibility to small service providers to gain access to secure services from TEE, without going through the OEMs' approval model. For the criteria impartiality we have marked all models except the consumer-centric one as partially meeting the criteria, for the reason that there is a centralised entity and guaranteeing its impartiality would be difficult. However, for the consumer-centric model, it was marked as meeting the criteria in full because users can give any application access to the security services provided by the TEE.

Table 6.1: Comparison of Different Ecosystems for TEE Deployment

| **Criteria** | **Centralised Model** | **Security as Service** | **Consumer Centric** |
|---|---|---|---|
| 1) Market Segmentation | ◑ | ◑ | ● |
| 2) Scalability | ◑ | ◑ | ● |
| 3) Flexibility | ○ | ● | ● |
| 4) Impartiality | ◑ | ◑ | ● |
| 5) Consumer-Involvement | ○ | ◑ | ● |
| 6) Open Provisioning | ○ | ◑ | ● |
| 7) Closed Provisioning | ● | ◑ | ○ |
| 8) User Privacy | ○ | ○ | ● |
| 9) Application IP Protection | ○ | ◑ | ● |

**Note:** ● if criteria is fully satisfied, ◑ if criteria is partially met and ○ if criteria is not satisfied.

Centralised models do not fare well on the criteria including open provisioning, consumer-involvement, user privacy and applications IP protection. The reason behind this is the process which an application has to go through before getting access to the TEE. It can be argued that App Stores managed by Apple and Google already do so. However, most of these applications might not have proprietary code like banking, transport-ticketing and mobile network operators (soft-SIMs) might have. Such application, along with other high security sensitive applications, might not accept it.

The consumer-centric model meets the highest number of criterion in comparison to all other models discussed in this chapter. It might be argued that such a model might introduce security issues, like a malicious application can also run in the TEE. This is possible, but TEE by no means provide an assurance that only non-malicious application code will execute in it. Such assumptions are based on prior vetting of an application by OEMs and/or TAP, by analysing the application source code. This, as discussed before, might not be preferable to a large set of application providers.

## 6.4 Making a Case for Security as a Service

Applications such as transit and payments that require higher levels of security that initially relied on hardware modules (such as smartcards and SEs), are increasingly transitioning to the mobile environment. To provide comparative levels of security, the TEE is a well-established technology to provide security. However, as discussed in this chapter, the centralised model does not currently provide the flexibility, for service providers to fully utilize the TEEs on their customers devices. This is partly due to commercial reasons, as well as the design architecture of TEEs. The current design architecture of TEEs cannot support multi-party party access to its services. This makes the idea of having a user-centric model of TEEs not feasible.

However, the SaaS model provides a bridge between the two extremes. The applications of service providers can utilise the security services of the TEE, without having complete control over the provisioning of applications or the general management of the TEE. From a commercial perspective, this also provides a middle ground for all the entities involved. The service providers and users have higher levels of security, while the OEMs still control their TEEs, and also enjoy the dividends of the security services they provide.

## 6.5 Summary

In this chapter, the issue of lack of accessibility to the security services provided by the TEE was discussed. To address the issue, two ecosystem models where considered. The user-centric model proved to be the most desirable, but the assumptions that would need to be in place for it to be successful are not realistic. The SaaS model, though not the most desirable, it seems to be the most feasible model to address the issue. In the next chapter, a novel on-device tokenisation framework using the TEE is proposed. This is an example of how the OEM can provide generic security services to service providers (such as transport operators), without giving unfettered access to the TEE.

# Chapter 7

# On-device Tokenisation

## 7.1 Introduction

The state-of-the-art in tokenisation is the periodic provisioning of tokens and related cryptographic data to the device. The EMVCo specification [64] does not dictate the validity of tokens, it allows both single and multiple use tokens. The validity of the tokens is typically determined by factors such as; the perceived level of risk on the payment device, the value of the transactions, payment use-case e.t.c. Consequently, the shorter the validity of tokens, the more frequent the process of token provisioning occurs.

In this chapter, the potential shortcomings of implementing tokenisation in its classic form are discussed. These issues show the need for innovative thinking in implementing tokenisation in certain scenarios. To make improvements on these shortcomings, and to build up on the work presented in Chapter 5, a novel tokenisation framework — On-device Tokenisation — is proposed. On-device tokenisation using Format Preserving Encryption (FPE), and relies on the TEE on the mobile devices to securely generate and store tokens. This removes the need to periodically provision new tokens to the mobile device. The tokenisation framework proposed also uses a cryptographically secure pseudo-random number generator to enable the Token Service Provider (TSP) and the mobile device to generate synchronised tokens on each side, instead of

de-tokenisation, as it is in the classic tokenisation model.

The framework envisages the OEMs playing the role of TSPs (see Section 3.7), and generating tokens through trusted applications on the TEE. This is a fair assumption since the OEMs have unfettered access to the TEEs on the devices they manufacture. It also makes sense to envision OEMS as TSPs considering the fact that Apple and Google jointly control %99.6 of the global smartphone market [139].

A PoC was implemented to have an idea of the performance of the FPE algorithms on mobile devices which to the best of our knowledge has not been published before. It is worth mentioning however that the PoC fell short of implementing the FPE algorithm in a TEE itself, and was implemented in the normal world of the mobile device. This will form part of the future research direction of the thesis.

## 7.2 Shortcomings of Tokenisation

This section discusses the shortcomings of tokenisation which the work in this chapter aims to solve:

**Additional Cost:** Although the cost of tokenisation is low compared to other security mechanisms (see Table 4.1), there is the cost generating and provisioning new token to the device [140]. In high-risk scenarios such as HCE, tokens are expected to have a shorter validity, hence a higher frequency of token generation and provisioning. This translates to additional costs in processing, which likely to be incurred by the merchants and indirectly transferred to the users [141].

**Connectivity:** Token provisioning requires the mobile device to have connectivity, therefore single use tokens cannot be used in scenarios that connectivity cannot be guaranteed. This requirement contradicts one of the fundamental functional requirements of mobile devices in ticketing, (see Section 5.5.1), because connectivity cannot be guaranteed in certain scenarios such as; underground train stations and moving vehicles.

**Potential Privacy Compromise:** The use of 'limited time' time tokens has the potential to compromise user-privacy in ticketing systems. For example if the validity of the token is one week, the transport operator can potentially track the journeys of a user until a new token is issued at the end of the week. While this is better than having a PAN which is static, there is still room for improvement with regards to privacy protection.

## 7.3 Related Work

Tokenised payments in offline environments were considered by Jayasinghe et al. in [142]. The author's considered payment scenarios where connectivity cannot be guaranteed. They proposed a tokenisation mobile payments protocol based. Their solution uses risk analysis techniques similar to velocity checking and floor-limit checks as specified by EMVCo (see page. 111–113 in [143]). The main drawback on this proposal is the efficiency of the implemented protocol. According to the authors, a transaction using their protocol is expected to take about 3.9 seconds on average. This makes it unsuitable for use in transport ticketing payments or even retail payments (see Section 5.5.1).

A framework for enhancing tokenisation using 'dynamic transaction tokens' was proposed by Jayasinghe et al. in [144]. The authors proposed the use of a fresh token per transaction (dynamic), which are generated in the SE of a smartphone. The fundamental issue with this proposal is that if the payment application has access to the SE, then tokenisation will not be required since the SE is secure enough to hold the PAN itself.

The concept of 'Updatable Tokenization' was introduced by Cachin et al. in [145]. They considered the problem of re-keying tokenisation algorithms while maintaining the *referential integrity*[1] of already tokenised data. The authors argue that current tokenisation systems require the whole data set to be re-tokenised with a new key any time a key update is carried out, otherwise referential integrity is lost. As a solution, the authors proposed a scheme where a new key and a tweak is used to generate a set of tokens for a particular host, by the tokenising entity. The host can use the same tweak to generate a similar set of tokens, which are used to roll forward previously generated tokens. Their solution is designed for the storage management of token in the back-office systems of TSPs. The solution presented in this chapter defers to

[1]Referential integrity here means it should be possible to previously generated tokens to the original PAN, regardless of the Key used.

this because the front-end generation and use of tokens is the focus, rather than how they are stored.

### 7.3.1 Encryption of PANs

The encryption of PANs to create tokens presents two issues that the framework presented considers; tweaking of the underlying algorithm, and handling Luhn checks.

Tweaking in cryptography is a process of including an additional input (tweak) to a block-cipher, alongside the usual key and plaintext. The use of tweaks in cryptography was formalised in 2002 by Liskov et al. in [146]. Tweaks are used to increase the otherwise limited number of possible values of a short plaintext. Recall that only the account identifier (Digits 7 to final number minus 1) part of the 16-19 digit PAN are encrypted (see Fig. 3.4). Note that in a database 100-million PANs, at least 100 PANs will be expected to have the same ciphertext digits (but the unencrypted part will be different) [147].

The tweak used for the tokenisation framework is a random number generated from a shared seed value $S$. For the random number generation, any deterministic random number generator can be used. For the purpose of the framework presented here, we assume the existence of a secure random number generator such as the ones recommended by the National Institute of Standards and Technology (NIST) in [148]. The deterministic random number generation is required to allow both the TSP and the mobile device to generate synchronised seeds$S$.

## 7.4 Format Preserving Encryption (FPE)

Some applications such as in payments processing, work based on industry defined data formats and will therefore require any use of cryptography to preserve the required format. For example encryption of the PAN '1314 5678 9743 1265' should result in a ciphertext of '2314 5463 8965 5482'. However, that is not the way traditional cryptographic algorithms work. For example, block-ciphers will pad the plaintext to the size of the underlying block if necessary and output the ciphertext in Hex or Base64 formatting. Take the DES-encryption of the PAN; '1314 5678 9743 1265' for example. The resulting ciphertext is; '8f8cbc3c86908031a1c229cbfb889c4d571a57b4372c8641'. This remained an open problems in cryptography until the initial solutions were proposed in 1997 by Smith and Brightwell in [149], and referred to it as "datatype-preserving encryption".

Nevertheless, the definitions of a cryptographically secure[2] FPE were first given by Black and Rogaway in [150]. They proposed three ways to achieve a solution: the prefix method, the cycle-walking method, and the feistel structure method. The FPE used for the work in this chapter is the FF1 algorithm, the"FF" indicates that is format-preserving and based on a feistel structure. FF1 was submitted by Spies. et al. in [147] as a draft NIST submission and was later accepted and published as part of a NIST specification in [4]. A description of how feistel structures work is given below.

### 7.4.1 Feistel Structure

The feistel structure (also known as a Feistel network) is symmetric mechanism used in the construction of block-ciphers. It is made-up of several iterations of a reversible transformation, these iterations are referred to as rounds. The reversible transformation consists of the following steps:

[2]with security provably related to the security of the underlying block cipher in use

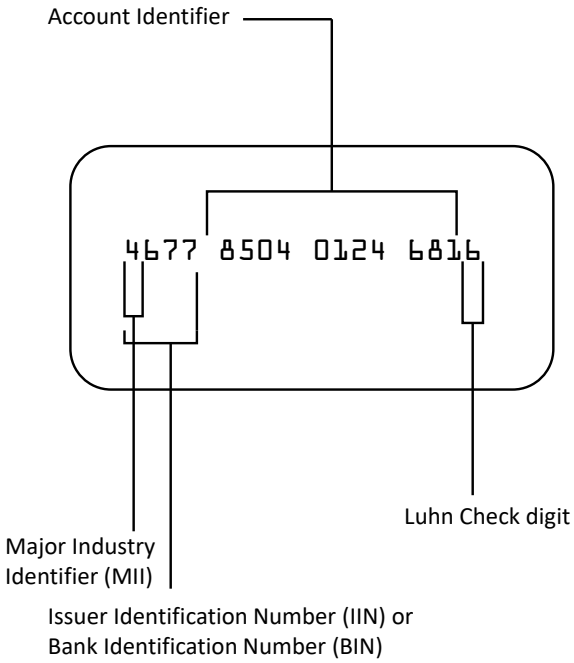


Figure 7.1: Layout of an On-Device Token

- Split the data into two parts $A$ and $B$
- Apply keyed function, $F_K$ to one part of the data, which is used to modify the other part
- In the next round $i$, the two parts of the data are swapped and the second step is repeated.

The encryption and decryption functions using the feistel structure are shown in Fig. 7.2. For simplicity, the diagram only shows the four rounds, but the FF1 requires ten rounds according to the specification.

#### 7.4.1.1 Encryption and Decryption

The length of the two plaintext strings $A$ & $B$ is denoted by $u$ & $v$ such that $u+v=n$. For encryption, the FF1 takes as input the plaintext $X$ as numeral strings, an optional tweak $t$ and an encryption $K$ and outputs ciphertext $Y$. For each Round, the function $F_K$ is applied to one of the strings $B_i$, alongside

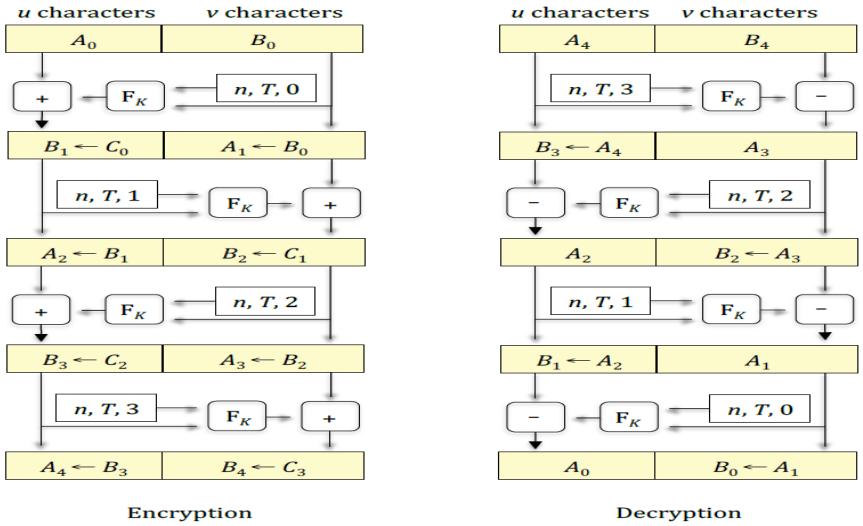


Figure 7.2: Encryption and Decryption Functions using a Feistel Structure. Source: [4]

additional variable; the length $n$, the round number $i$ and the tweak $t$. The output of this operation is used to modify the other half of the string $A_i$ through modular addition $+$. The resulting string is stored as a temporary variable $C_i$, which is used as the $B_{i+1}$ in the next round, and $B_i$ becomes $A_{i+1}$.

The decryption process is similar to that of encryption, with the following differences; the order of the round indices is reversed, modular subtraction is used instead of addition. Also, the input to the round function $F_k$ is changed to $A_{i+1}$, and the output is used to modify $Bi + 1$ to produce $A_i$.

## 7.5 Proposed Tokenisation Framework

### 7.5.1 Entities

1. **Mobile Device (device):** The mobile device, held by a user, is an NFC phone with a Trusted Execution Environment (TEE) where trusted applications and cryptographic operations are carried out.

2. **TSP-app:** The TSP-app is a trusted application issued and installed into the TEE within a user's device. The procedure for installing the TSP-app typically will involve an Identity and Verification (ID&V) step to assert the identity of the user, this is beyond our scope.

3. **Token Service Provider (TSP):** The TSP is the entity that manages the tokenisation. It is in charge of issuing the TSP-app to the users' devices, issuing of the initial token into the TSP-app, and also the detokenisation and verification of tokens during transaction processing to determine their validity. The TSP and its systems serves as the 'source of truth' for all tokens of users at any time.

4. **Merchant:** The merchant is a provider of goods and/or services with a point-of-sale to receive payments made via the users' devices. As part of the transaction, the merchant receives a token, which must be verified with the TSP before the transaction can be authorised. The details of the relevant transaction data that will usually be passed here is beyond out scope.

### 7.5.2 Phases

The On-device tokenisation framework is divided into five (5) distinct phases as presented below.

#### 7.5.2.1 Set-up Phase

This is a one-time process carried out to configure the user's device to use the on-device tokenisation framework. It is assumed that the TSP has installed the TSP-app within the TEE of users' devices, and a shared key $K_s$ has been established between them.

#### 7.5.2.2 Tokenisation Request Phase

Following a successful ID&V step, the user is prompted to enter the original PAN for the payment card to be tokenised into the TSP-app, alongside the result of the ID&V step is sent to the TSP-app through a trusted path. See for a description of trusted path. Note that at no point is the PAN stored on the device, not even in the TEE.

The TSP-app encrypts the information received with the shared key $K_s$, and sends it to the TSP. We assume the TSP and the TSP-app within the TEE of the users' devices have a secure interface between them such as one specified by GlobalPlatform in the *Device TEE Sockets API Specification* [151].

#### 7.5.2.3 Master Token Issuance

The TSP chooses a Master Token $M_t$ (which has the respective OEMs IIN as leading digits) for a each PAN at random and stores the relationship. $M_t$ is sent to the device, alongside the chosen shared seed $S$. Both $M_t$ and $S$ are sent to the device ready to be used for the subsequent generation of tokens on-device.

#### 7.5.2.4 Token Generation

The token is generated by the TSP-app prior to a transaction with a merchant. The IIN and the luhn check digit are stripped from the $M_t$, to be left with 11-digit $t_i$; which is the part that would have been the account identifier if it was a real PAN (see Fig. 7.1). The FPE encryption algorithm takes as input;

the 11-digit $t_i$, the tweak $t$ generated from the shared seed $S$, the shared key $K_s$ and outputs an 11-bit string $t_x$. The IIN is appended to $t_x$, the luhn check digit of the whole string is then calculated and appended to produce the on-device per transaction token $T_k$.

#### 7.5.2.5 Detokenisation

Detokenisation is simply the reverse of the token generation process. When a merchant receives a token from the user's device during a transaction, it uses the IIN to determine which TSP to route the transaction to. The TSP receives the token and then strips the IIN and the luhn check digit from $T_k$ to recover $t_x$. The TSP then inputs $t_x$ the pre-shared key $K_s$, the tweak $t$ which it can generate using the seed $S$ into the FPE algorithm to get $t_i$, which acts the account identifier of the user. Using $t_i$, the TSP can effectively route the transaction through the payment network for authorisation as usual.

### 7.5.3 Proof-of-Concept and Testing

The tokenisation framework was implemented as an Android application (TSP-app) on Samsung S5 smartphone, to test for performance. The Android implementation itself is based on an earlier Java implementation for the FPE algorithm published in [152]. The goal of the implementation was not to asses the security of FPE algorithms, but rather, to test their performance on a mobile device. The process of token generation happens in the background (within the TEE) and completely invisible to the user. The Luhn check functionality was also implemented as part of the TSP-app to ensure that the tokens generated conform to the EMV specifications. The token generation process on the device takes an average 19.6 milliseconds over 100 runs. The performance of the detokenisation is not important since that happens later at the TSPs back-office.

## 7.6 Summary

In this chapter, a novel on-device framework has been presented. The goal was to provide the preliminaries for a different approach to tokenised payments especially in ticketing. The idea of using one-time on-device tokens is suitable for ticketing because connectivity is not required for token provisioning. The performance testing shows that the processing-times for FPE algorithms is trivial, and therefore their efficiency is suitable for ticketing. In addition, the framework presented in this chapter can be used in other usecases apart from ticketing. For example, the Second Payments Directive (PSD2) regulation mandates banks to provide 3rd Party access to users' account if the users give their consent. The on-device tokenisation framework can potentially be used as a mechanism to prove the to prove user's consent to the their respective bank.

# Chapter 8

# Conclusion

This thesis set out to investigate the potential impact HCE may have on the overall security of mobile devices in ticketing. As transport operators transition from closed-loop to open systems, this thesis also explores this relatively new phenomenon, and determines potential problems that require different approach. The conclusions to the main contributions of the thesis, and the research questions were summarised.

A comparative testing of hardware SE-based card emulation and HCE-based card emulation was conducted. HCE-based applications proved to be the faster, what is interesting however is the significant variation in the HCE execution times. This behaviour could call into question its use in transport ticketing, where the performance is paramount. It also shows that security controls such as distance-bounding protocols used for combating relay attacks will not be feasible for HCE applications because they rely on knowledge of expected execution times. From investigations conducted during the course of this thesis, it is evident that the shift from SE-based to HCE-based NFC transactions has significant impacts on transport ticketing. HCE in comparison to the SE, provides a richer user interface, superior processing power, and a more flexible ecosystem for the transport operators. However, in terms of security, HCE undeniably introduces security risks to transport ticketing

data and processes. The security mechanisms to mitigate these risks and increase the attack resistance of HCE applications exists. Nevertheless, analysis conducted in this thesis show that not all of mechanisms are feasible for ticketing. The choice of which security mechanism will largely depend on specific use-cases, level of perceived risk, and cost.

In addition to the security mechanisms mentioned in Chapter 4, it is important to take other supplementary security measures. Periodic user-verification and context-aware security checks before important ticketing functions to mitigate the risk of exposure. For example, prior to the ticketing application or ticket provisioning, the device can be verified to ensure it is not rooted.

The current state and maturity of these security mechanisms make tokenisation the most feasible mechanism. This is partly because it can be used with devices out-of-the-box and does not require special agreements or set-up, as is the case with using WBC or TEEs. Tokenisation is also the preferred security mechanism for the payments industry. Therefore as ticketing systems increasingly transition to open ticketing systems, transport operators will have to design ticketing systems that conform to the process in the payment ecosystem.

Utilizing the well-established payment infrastructure has a lot of advantages for transport ticketing. However, the problem of user blacklisting in the case of tokenised payments shows an example of why payment mechanisms should not be implemented out-of-the-box for ticketing without rigorous considerations. The protocol presented in Chapter 5 provides a solution to the blacklistability problem of tokenised payments that is suitable for ticketing. And its implementation also shows that mobile devices are capable of efficiently handling complex cryptographic operations to provide stronger levels of security for ticketing. Despite the fact that the LGS used resulted in a payload of considerable size (2552 bytes, compressed to 1617 bytes), modifications to the Android OS prove that the NFC controller of the device is capable of

sending messages that are bigger than what the OS limits it to. This further proves the capabilities of mobile devices today in handling such algorithms.

The problem of lack of accessibility to TEEs on mobile devices was highlighted in Chapter 6. Potential ecosystem models that will provide more access to TEEs were also discussed. A comparison of these models show that the user-centric model to be the most favourable based on the set criteria. Nevertheless, by virtue of the security design of current TEE implementations, it is not realistic to envision TEEs in a user-centric ecosystem. In this light, the SaaS model provides a middle ground to maximise the full potential of the TEE, without the OEMs necessarily giving full access to it.

The novel tokenisation framework presented in Chapter 7 builds up on the SaaS model discussed in Chapter 6. A method of generating tokens in the TEE of the user's device using FPE algorithms was proposed. This eliminates the need to periodically provision new tokens to the device. And for open ticketing systems, it enhances user privacy because it is possible to use one-time tokens, thereby protecting the user's journey from tracking. Implementation on a mobile device shows that the framework is efficient for ticketing, and further asserts the fact that mobile devices are capable of providing better and innovative security solutions.

### 8.0.1 Future Work

During the course of the work presented in this thesis, they've been areas that required further investigations to be carried out. But due to limited resources, in terms of time and access, it was not possible. These areas will be further investigated in the future. These are summarised below:

1. Conducting a more comprehensive testing: In this thesis, all proposals presented in this thesis where tested on single devices, to build upon the initial findings, further work is planned to consider testing on additional platforms to have a more representative view. In addition, the work in

Chapter 5 and Chapter 6 were implemented in the 'normal world' of the mobile devices due to limitations on accessing the TEE. As future work, the plan is to provide similar implementations on the TEE, to test its efficiency.

2. More Efficient forms of blacklisting: The future plan is to investigate more efficient methods of achieving linkability while protecting the privacy of users. The blacklisting solution presented in Chapter 5 resulted in a payload that is rather big, and requires compression to fit into an APDU. While this works, a solution like this may not be suitable for a large-scale implementation. We also aim to consider this idea of linkability in other use-cases such as retail, where merchants can use the similar protocols to offer users bespoke services while still respecting their privacy.

# Bibliography


[1] Transport for London. Top-line Contactless Figures. Statistics, TfL, London, UK, July 2017.

[2] ARM Limited. ARM Security Technology Building a Secure System using TrustZone Technology, April 2009.

[3] Intel Corporation. Intel Software Guard Extensions Programming Reference, October 2014.

[4] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. *NIST Special Publication*, SP-800-38G, 2016.

[5] ISO/IEC. Identification cards – Contactless integrated circuit cards – Proximity cards . Technical Report 14443, International Organization for Standardization (ISO), 2008.

[6] ISO/IEC. Information Technology Identification Cards Financial Transaction Cards 7813. Standard, International Organization for Standardization, Geneva, CH, 2006.

[7] Marie-Pier Pelletier, Martin Trpanier, and Catherine Morency. Smart Card Data Use in Public Transit: A literature Review. *Transportation Research Part C: Emerging Technologies*, 19(4):557 – 568, 2011.

[8] Sandeep Tamrakar, Jan-Erik Ekberg, and N. Asokan. Identity Verification Schemes for Public Transport Ticketing with NFC Phones. In *Proceedings of the Sixth ACM Workshop on Scalable Trusted Computing*, STC '11, pages 37–48, New York, NY, USA, 2011. ACM.

[9] Serge Chaumette, Damien Dubernet, Jonathan Ouoba, Erkki Siira, and Tuomo Tuikka. *Architecture and Comparison of Two Different User-Centric NFC-Enabled Event Ticketing Approaches*, pages 165–177. Springer, Berlin, Heidelberg, 2011.

[10] Jinguang Han, Liqun Chen, Steve A. Schneider, and Helen Treharne. PPETS-FGP: Privacy-Preserving Electronic Ticket Scheme with Fine-Grained Pricing. *Computing Research Repository (CoRR)*, abs/1706.03016, 2017.

[11] U. B. Ceipidor, C. M. Medaglia, A. Marino, M. Morena, S. Sposato, A. Moroni, P. Di Rollo, and M. L. Morgia. Mobile ticketing with nfc management for transport companies; problems and solutions. In *2013 5th International Workshop on Near Field Communication (NFC)*, pages 1–6, Feb 2013.

[12] Rosario Giustolisi. Free rides in denmark: Lessons from improperly generated mobile transport tickets. In Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevičius, editors, *Secure IT Systems*, pages 159–174, Cham, 2017. Springer International Publishing.

[13] Dimitrios Tsamboulas and Constantinos Antoniou. Allocating Revenues To Public Transit Operators Under an Integrated Fare System. *Transportation Research Record: Journal of the Transportation Research Board*, (1986):29–37, 2006.

[14] Transport Committee. The Future of Ticketing. Technical report, London Assembly, 2011. Greater London Authority.

[15] The UK CARDS Association (UKCA). Contactless Statistics. `http://www.theukcardsassociation.org.uk/contactless_contactless_statistics/index.asp`, April 2017.

[16] Briony Krikorian-Slade, Adrian Burholt, and Nicola Moir. Contactless Transit Framework. Standard Version 2.0, UK Cards Association, London, UK, 2016.

[17] Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. *Dismantling MIFARE Classic*, pages 97–114. Springer, Berlin, Heidelberg, 2008.

[18] David Oswald and Christof Paar. *Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World*, pages 207–222. Springer, Berlin, Heidelberg, 2011.

[19] EMVCo. Mobile Product Level 1 Type Approval – Administrative Process. Specification Version 2.2, EMVCo, LLC, June 2017.

[20] EMVCo. Terminal Type Approval PCD Level 1 Administrative Process. Specification Version 2.6b, EMVCo, LLC, December 2016.

[21] Branden R. Williams and Anton Chuvakin. *PCI Compliance: Understand and Implement Effective PCI Data Security Standard Compliance*. Syngress Publishing, 4th edition, 2014.

[22] DNA. U.S. EMV Debit Implementation Guidelines for POS Acquirers. Report Version 1.0, Debit Network Alliance, August 2014.

[23] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. User Privacy in Transport Systems Based on RFID E-Tickets. In *Proceedings of the 1st International Workshop on Privacy in Location-Based Applications, Malaga, Spain, October 9*, 2008.

[24] Ghada Arfaoui, Jean-François Lalande, Jacques Traoré, Nicolas Desmoulins, Pascal Berthomé, and Saïd Gharout. A Practical Set-Membership Proof for Privacy-Preserving NFC Mobile Ticketing. *CoRR*, abs/1505.03048, 2015.

[25] Giuseppe Ateniese and Gene Tsudik. *Some Open Issues and New Directions in Group Signatures*, volume 1648 of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*. 1999.

[26] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. *Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions*, pages 614–629. Springer, Berlin, Heidelberg, 2003.

[27] Arnau Vives-Guasch, Magdalena Payeras-Capella, Macià Mut-Puigserver, and Jordi Castellà-Roca. *E-Ticketing Scheme for Mobile Devices with Exculpability*, pages 79–92. Springer, Berlin, Heidelberg, 2011.

[28] Arnau Vives-Guasch, Magdalena Payeras-Capell, Maci Mut Puigserver, Jordi Castell-Roca, and Josep Llus Ferrer-Gomila. A secure e-ticketing scheme for mobile devices with near field communication (nfc) that includes exculpability and reusability. *IEICE Transactions*, 95-D(1):78–93, 2012.

[29] Thyla Joy van der Merwe. Investigating the use of Elliptic Curve Cryptography in Transport Ticketing. Technical report, Royal Holloway, University of London, 2015.

[30] Makoto Eguchi, Susan Fredholm, Shan Liu, Paulina Ponce de Leon Barido, and Jacqueline Ye. Policy Issues in Implementing Smart Cards in Urban Public Transit Systems. 2007.

[31] Ghada Arfaoui. *Design of Privacy Preserving Cryptographic Protocols For Mobile Contactless Services*. PhD thesis, Université d'Orléans, 2015.

[32] Transit and Contactless Open Payments: An Emerging Approach for Fare Collection. White paper, Smart Card Alliance Transportation Council, November 2011.

[33] Visa. The Future of Ticketing: Paying for Public Transport Journeys Using Visa Cards in the 21st Century. Whitepaper, Visa Europe, January 2013.

[34] MasterCard. Contactless Performance Requirement. Online, MasterCard Incorporated, March 2014.

[35] EMV Contactless Specifications for Payment Systems: Book D - EMV Contactless Communication Protocol Specification. Specification Version 2.6, EMVCo, LLC, March 2016.

[36] Visa. Transactions Acceptance Device Guide (TADG). Specification Version 3.0, Visa Europe, May 2015.

[37] Gilles de Chantérac and Jean-Louis Graindorge. Focus Paper on Privacy in Transport IFM Applications. IFM Project, March 2009. Draft Deliverable 2.2.

[38] Arnau Vives-Guasch. *Contributions To The Security and Privacy of Electronic Ticketing Systems*. PhD thesis, Universitat Rovira i Virgili, 2013.

[39] ITSO Ltd. Interoperable Public Transport Ticketing using Contactless Smart Customer Media Part 2: Customer Media Data and Customer Media Architecture. Standard: ITSO TS 1000-2 Version 2.1.4, ITSO LTD, Milton Keynes, UK, February 2010.

[40] ITSO Ltd. Technical Specification Interoperable Public Transport Ticketing using contactless smart customer media Part 3: Terminals. Standard: ITSO TS 1000-3 Version 2.1.4, ITSO LTD, Milton Keynes, UK, February 2010.

[41] ITSO Ltd. Technical Specification Interoperable Public Transport Ticketing using Contactless Smart Customer Media Part 4: HOPS. Standard: ITSO TS 1000-4 Version 2.1.4, ITSO LTD, Milton Keynes, UK, February 2010.

[42] ITSO Ltd. Technical Specification Interoperable Public Transport Ticketing using contactless smart Customer Media Part 6: Message data. Standard: ITSO TS 1000-6 Version 2.1.4, ITSO LTD, Milton Keynes, UK, February 2010.

[43] ITSO Ltd. Technical Specification Interoperable Public Transport Ticketing using Contactless Smart Customer Media Part 7: ITSO Security Subsystem. Standard: ITSO TS 1000-7 Version 2.1.4, ITSO LTD, Milton Keynes, UK, February 2010.

[44] Mark Turner and Ruth Wilson. Smart and Integrated Ticketing in The UK: Piecing Together the Jigsaw. *Computer Law & Security Review*, 26(2):170–177, 2010.

[45] Sarah Thomas. Oyster Card Circulation, December 2016. FOI: Transport for London.

[46] W. Rankl and W. Effing. *Smart Card Handbook*. Wiley, 2010.

[47] NFC-Forum. NFC Forum Specification Architecture. Technical report, NFC-Forum, 2006.

[48] ECMA-340. Near Field Communication - Protocol and Interface (NFCIP-1). Technical report, European Computer Manufacturers Association, 2004.

[49] ISO/IEC. Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1). Standard, International Organization for Standardization, Geneva, CH, 2013.

[50] Android. Android Developer Guide – Host-based Card Emulation, 2013.

[51] Micheal Roland. Software Card Emulation in NFC-enabled Mobile Phones: Great Advantage or Security Nightmare. In *Fourth International Workshop on Security and Privacy in spontaneous Interaction and Mobile Phone Use*, Newcastle UK, June 2012.

[52] Doug Yeager. Added NFC Reader Support for Two new Tag Types: ISO PCD type A and ISO PCD type B, 2012. GitHub Repository.

[53] Linda Sui. Android Captures Record 88 Percent Share of Global Smartphone Shipments. Statistics, Strategy Analytics, Boston, USA, November 2016.

[54] Smart Card Alliance. Host Card Emulation (HCE) 101. Technical Report MNFCC-14002, SmartCardAlliance, August 2014.

[55] Martin Emms, Budi Arief, Leo Freitas, Joseph Hannon, and Aad van 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*, pages 716–726. ACM, 2014.

[56] Mariusz H. Jakubowski, Chit Wei (Nick) Saw, and Ramarathnam Venkatesan. *Tamper-Tolerant Software: Modeling and Implementation*, pages 125–139. Springer, 2009.

[57] H. Guo and B. Jin. Forensic Analysis of Skimming Devices for Credit Fraud Detection. In *2010 2nd IEEE International Conference on Information and Financial Engineering*, pages 542–546, Sept 2010.

[58] ISO/IEC. Identification cards – Identification of issuers – Part 1: Numbering system. ISO/IEC 7812-1. Standard, International Organization for Standardization, Geneva, CH, 2015.

[59] Hans Peter Luhn. Computer for Verifying Numbers, August 23 1960. US Patent 2,950,048.

[60] Christian Radu. *Implementing Electronic Card Payment Systems*. Artech House computer security series. Artech House, 2003.

[61] Mike Burden. EMV Considerations for Transit. Technical report, Smart Card Alliance, 2013.

[62] Sandra Lambert and Jeff Stapleton. Cybersecurity vs. tokenization. Technical report, RSA Conference Presentation, February 2017.

[63] PCI Security Standards Council. Tokenization Product Security Guidelines. Technical Report Version 1.0, PCI Security Standards Council, April 2015.

[64] EMVCo. EMV Payment Tokenisation Specification Technical Framework. Standard Version 1.0, March 2014.

[65] Steve Pannifer, Dick Clark, and Dave Birch. HCE and SIM Secure Element: Its Not Black and White. Technical report, Consult Hyperion, Guildford, Surrey, June 2014.

[66] Michael Roland and Josef Langer. Comparison of the Usability and Security of NFC's Different Operating Modes in Mobile Devices. *E&I Elektrotechnik und Informationstechnik*, 130(7):201–206, 2013.

[67] C. Saminger, S. Grnberger, and J. Langer. An nfc ticketing system with a new approach of an inverse reader mode. In *2013 5th International Workshop on Near Field Communication (NFC)*, pages 1–5, Feb 2013.

[68] Thom Janssen and Mark Zandstra. Whitepaper HCE security implications, analysing the security aspects of HCE. Technical report, Underwriters Laboratories, UL, January 2014.

[69] Ruiz, Alejandro Pérez and Rivas, Mario Aldea and Harbour, Michael González. CPU Isolation on the Android OS for Running Real-Time Applications. In *Proceedings of the 13th International Workshop on Java Technologies for Real-time and Embedded Systems*, JTRES '15, pages 6:1–6:7, New York, NY, USA, 2015. ACM.

[70] Cláudio Maia, Luis M. Nogueira, and Luis M. Pinho. Evaluating Android OS for Embedded Real-Time Systems. In *6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2010)*, pages 62+. Politécnico do Porto, July 2010.

[71] Igor Kalkov, Dominik Franke, John F. Schommer, and Stefan Kowalewski. A real-time extension to the android platform. In *Proceedings of the 10th International Workshop on Java Technologies for Real-time and Embedded Systems*, JTRES '12, pages 105–114, New York, NY, USA, 2012. ACM.

[72] L. Perneel, H. Fayyad-Kazan, and M. Timmerman. Can android be used for real-time purposes? In *2012 International Conference on Computer Systems and Industrial Informatics*, pages 1–6, Dec 2012.

[73] Android Website. Android Developer Guide – Android Keystore System. https://developer.android.com/training/articles/keystore.html.

[74] Android. Dashboards Platform Versions, July 2017. https://developer.android.com/about/dashboards/index.html.

[75] GlobalPlatform. Secure Element Remote Application Management. Standard 1.0.1, GlobalPlatform, Novemebr 2015.

[76] GSMA and Underwriters Laboratory. Mobile Payment Security. White paper, Groupe Speciale Mobile Association, October 2014.

[77] Mouhannad Alattar and Mohammed Achemlal. Host-Based Card Emulation: Development, Security, and Ecosystem Impact Analysis. *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS)*, pages 506–509, 2014.

[78] A. Armando, A. Merlo, and L. Verderame. Trusted Host-based Card Emulation. In *2015 International Conference on High Performance Computing Simulation (HPCS)*, pages 221–228, July 2015.

[79] M. Alattar and M. Achemlal. Host-Based Card Emulation: Development, Security, and Ecosystem Impact Analysis. In *High Performance Computing and Communications, 2014 IEEE 6th Intl Symposium on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conference on Embedded Software and System (HPCC,CSS,ICESS)*, pages 506–509, Aug 2014.

[80] Jong-Yeon Park, Jung-Nyeo Kim, Jae-Dock Lim, and Dong-Guk Han. A Whitebox Cryptography Application for Mobile Device Security against

Whitebox Attacks-How to Apply WBC on Mobile Device. In *IT Convergence and Security (ICITCS), 2014 International Conference on*, pages 1–5. IEEE, 2014.

[81] Joppe W. Bos, Charles Hubain, Wil Michiels, and Philippe Teuwen. *Differential Computation Analysis: Hiding Your White-Box Designs is Not Enough*, pages 215–236. Springer, Berlin, Heidelberg, 2016.

[82] Stanley Chow, Phil Eisen, Harold Johnson, and Paul C. van Oorschot. A White-Box DES Implementation for DRM Applications. In *Digital Rights Management: ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002. Revised Papers*, pages 1–15, Berlin, Heidelberg, 2003. Springer.

[83] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-Box Cryptography and an AES Implementation. In *Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*, SAC '02, pages 250–270, London, UK, 2003. Springer-Verlag.

[84] Matthias Jacob, Dan Boneh, and Edward Felten. *Attacking an Obfuscated Cipher by Injecting Faults*, pages 16–31. Springer, Berlin, Heidelberg, 2003.

[85] H. E. Link and W. D. Neumann. Clarifying Obfuscation: Improving the Security of White-Box DES. In *International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II*, volume 1, pages 679–684 Vol. 1, April 2005.

[86] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. *Cryptanalysis of a White Box AES Implementation*, pages 227–240. Springer, Berlin, Heidelberg, 2005.

[87] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. *Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings*, pages 264–277. Springer, Berlin, Heidelberg, 2007.

[88] Android Developer Guide. SMP Primer for Android. `https://developer.android.com/training/articles/smp.html`.

[89] S. Tarkoma, M. Siekkinen, E. Lagerspetz, and Y. Xiao. *Smartphone Energy Consumption: Modeling and Optimization*. Smartphone Energy Consumption: Modeling and Optimization. Cambridge University Press, 2014.

[90] Sun Microsystems. Java Card Platform Specification Version 2.2.1 . `http://java.sun.com/products/javacard/specs.html`.

[91] Nokia. Nokia NFC Unlock Service MIDlet. Technical report, Nokia, 2007.

[92] COMPRION. CLT Move - Contactless Spy Tool. `http://www.comprion.com/en/products/monitoring/clt_move/overview`.

[93] Android. Android Developer Guide – Power Profiles for Android.

[94] Mastercard. MasterCard Contactless Performance Requirement Application Note Number 7. Technical report, MasterCard International Incorporated, March 2014.

[95] G. Hancke, K. Mayes, and K. Markantonakis. Confidence in Smart Token Proximity: Relay Attacks Revisited. *Elsevier Computers and Security*, 28(7):615–627, October 2009.

[96] S. Brands and D.Chaum. Distance-bounding Protocols. In *Advances in Cryptology EUROCRYPT '93*, Norway, September 1993.

[97] Gerhard P. Hancke and Markus G. Kuhn. An rfid distance bounding protocol. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, SECURECOMM '05, pages 67–73, Washington, DC, USA, 2005. IEEE Computer Society.

[98] Pedro Peris-Lopez, Julio César Hernández Castro, Juan M. Estévez-Tapiador, and Jan C. A. van der Lubbe. Shedding Some Light on RFID Distance Bounding Protocols and Terrorist Attacks. *Computing Research Repository (CoRR)*, abs/0906.4618, 2009.

[99] Christian Radu. *Implementing Electronic Card Payment Systems*. Artech House computer security series. Artech House, 2003.

[100] Samsung Pay Will Transform the Mobile Wallet Experience. Standard, Samsung Electronics Co, Ltd, 2016.

[101] John R. Douceur. *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA*, chapter The Sybil Attack, pages 251–260. Springer, Berlin, Heidelberg, March 2002.

[102] Annual Fraud Indicator. Report, University of Portsmouth, Centre for Counter Fraud Studies, Portsmouth, England, 2016.

[103] Information technology – Security Techniques – Anonymous Digital Signatures. Standard ISO/IEC 20008-2, International Organization for Standardization, Geneva, CH, 2013.

[104] Sébastien Canard, Berry Schoenmakers, Martijn Stam, and Jacques Traoré. List Signature Schemes. volume 154, pages 189–201, Amsterdam, The Netherlands, feb 2006. Elsevier Science Publishers B. V.

[105] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing Accountability and Privacy Using e-Cash (Extended Abstract). In

*Proceedings of the 5th International Conference on Security and Cryptography for Networks*, SCN'06, pages 141–155, Berlin, Heidelberg, 2006. Springer-Verlag.

[106] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '01, pages 93–118, London, UK, 2001. Springer-Verlag.

[107] Jan Camenisch and Anna Lysyanskaya. *Signature Schemes and Anonymous Credentials from Bilinear Maps*, pages 56–72. Springer, Berlin, Heidelberg, 2004.

[108] David Chaum and Eugène Van Heyst. Group Signatures. In *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'91, pages 257–265, Berlin, Heidelberg, 1991. Springer-Verlag.

[109] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '00, pages 255–270, London, UK, 2000. Springer-Verlag.

[110] Dan Boneh, Xavier Boyen, and Hovav Shacham. *Short Group Signatures*, pages 41–55. Springer, Berlin, Heidelberg, 2004.

[111] Aggelos Kiayias and Moti Yung. Group Signatures with Efficient Concurrent Join. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, Proceedings*, volume

3494 of *Lecture Notes in Computer Science*, pages 198–214. Springer, 2005.

[112] Roel Verdult, Flavio D. Garcia, Peter van Rossum, Ravindra Kali, and Vinesh Kali. Security Analysis of RFID Tags, 2008. `https://www.semanticscholar.org/paper/Security-analysis-of-RFID-tags-Verdult-Garcia/1d135642dd3454318203793c4b1a60e21afe5091`.

[113] Nicolas Courtois. The Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes Anywhere, Anytime. *IACR Cryptology ePrint Archive*, page 137, 2009.

[114] Roel Verdult. Proof of Concept, Cloning the OV-chip Card. Technical report, Technical report, Radboud University Nijmegen, 2008. `http://www.cs.ru.nl/~flaviog/OV-Chip.pdf`.

[115] Ivan Gudymenko. A Privacy-Preserving E-Ticketing System for Public Transportation Supporting Fine-Granular Billing and Local Validation. In *Proceedings of the 7th International Conference on Security of Information and Networks*, SIN '14, pages 101:101–101:108, New York, NY, USA, 2014. ACM.

[116] GhadaArfaoui, Guillaume Dabosville, Sébastien Gambs, Patrick Lacharme, and Jean-François Lalande. A Privacy-Preserving NFC Mobile Pass for Transport Systems. volume 2, page e4, 2014.

[117] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 132–145, New York, NY, USA, 2004. ACM.

[118] Quercia Daniele and Stephen Hailes. MOTET: Mobile Transactions using Electronic Tickets. In *Proceedings of the 1st IEEE Conference on*

*Security and Privacy for Emerging Areas in Communication Networks*, pages 374–383, Athens, Greece, September 2005.

[119] David Chaum, Amos Fiat, and Moni Naor. *Untraceable Electronic Cash*, pages 319–327. Springer New York, New York, NY, 1990.

[120] Jan-Erik Ekberg and Sandeep Tamrakar. Mass Transit Ticketing with NFC Mobile Phones. In *Trusted Systems - Third International Conference, INTRUST 2011, Beijing, China, November 27-29, 2011, Revised Selected Papers*, pages 48–65, 2011.

[121] Craig Gentry. *Advances in Cryptology — EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings*, chapter Certificate-Based Encryption and the Certificate Revocation Problem, pages 272–293. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[122] Silvio Micali. Scalable certificate validation and simplified pki management. In *1st Annual PKI research workshop*, volume 15, 2002.

[123] Niko Bari and Birgit Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer, 1997.

[124] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[125] Klaus Potzmader. ISO20008-2.2 Group Signature Scheme Evaluation on Mobile Devices, 2013. GitHub Repository.

[126] Klaus Potzmader, Johannes Winter, Daniel Hein, Christian Hanser, Peter Teufl, and Liqun Chen. Group Signatures on Mobile Devices: Practical Experiences. pages 47 – 64, 2013.

[127] Youssef Bassil. A Comparative Study on the Performance of the Top DBMS Systems. volume abs/1205.2889. CoRR - Computing Research Repository, 2012.

[128] Susan Pandy and Marianne Crowe. Payment Strategies: Understanding the Role of Host Card Emulation in Mobile Wallets. Technical report, Federal Reserve Bank of Boston, May 2016.

[129] Borko Lepojevic, Aleksandar Radulovic, and Bojan Pavlovic. Implementing NFC Service Security – SE VS TEE VS HCE. *SymOrg - International Scientific Symposium - Faculty of Organizational Sciences, University of Belgrade*, June 2014.

[130] Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. The Untapped Potential of Trusted Execution Environments on Mobile Devices. *IEEE Security & Privacy*, 12(4):29–37, 2014.

[131] Thomas Nyman, Brian McGillion, and N. Asokan. *On Making Emerging Trusted Execution Environments Accessible to Developers*, pages 58–67. Springer, Cham, 2015.

[132] GlobalPlatform. GlobalPlatform Device Technology, TEE System Architecture v1.0, December 2011.

[133] GlobalPlatform. GlobalPlatform Device Technology, TEE Internal API Specification, December 2011.

[134] GlobalPlatform. GlobalPlatform Device Technology, TEE Client API Specification v1.0, July 2010.

[135] GlobalPlatform. GlobalPlatform Device Technology, Trusted User Interface API Specification v1.0, June 2013.

[136] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. A Paradigm Shift in Smart Card Ownership Model. In *ICCSA Workshops*, pages 191–200. IEEE CS, 2010.

[137] Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra, and B. Hu. Everything as a service (xaas) on the cloud: Origins, current and future trends. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 621–628, June 2015.

[138] A New Model: The Consumer-Centric Model and How it Applies to the Mobile Ecosystem. Whitepaper, GlobalPlatform, March 2012.

[139] James Vincent. 99.6 Percent of New Smartphones Run Android or iOS, February 2017. The Verge.

[140] Flora Liu. Analysis of Tokenization in Digital Payment, 2016. Cyber Security Fall Final Paper.

[141] Pat Carroll. Tokenization: 6 Reasons The Card Industry Should Be Wary. `https://www.darkreading.com/perimeter/tokenization-6-reasons-the-card-industry-should-be-wary-/a/d-id/1316376`, July 2014.

[142] D. Jayasinghe, K. Markantonakis, I. Gurulian, R. N. Akram, and K. Mayes. Extending EMV Tokenised Payments to Offline-Environments. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 443–450, Aug 2016.

[143] EMVCo. EMV Integrated Circuit Card Specifications for Payment Systems – Book 3 Application Specification . Standard Version 4.3, November 2011.

[144] Danushka Jayasinghe, Konstantinos Markantonakis, Raja Naeem Akram, and Keith Mayes. *Enhancing EMV Tokenisation with Dynamic*

*Transaction Tokens*, pages 107–122. Springer International Publishing, Cham, 2017.

[145] Christian Cachin, Jan Camenisch, Eduarda Freire-Stögbuchner, and Anja Lehmann. Updatable Tokenization: Formal Definitions and Provably Secure Constructions. In Aggelos Kiayias, editor, *Financial Cryptography and Data Security*, pages 59–75, Cham, 2017. Springer International Publishing.

[146] Moses Liskov, Ronald L. Rivest, and David Wagner. *Tweakable Block Ciphers*, pages 31–46. Springer, Berlin, Heidelberg, 2002.

[147] Mihir Bellare, Phillip Rogaway, and Terence Spies. The FFX Mode of Operation for Format-preserving Encryption. *NIST Submission*, 20, 2010.

[148] Elaine B. Barker and John M. Kelsey. SP 800-90A. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. Revision 1. Technical report, Gaithersburg, MD, United States, 2015.

[149] Michael Brightwell and H Smith. Using Datatype-preserving Encryption to Enhance Data Warehouse Security. In *20th National Information Systems Security Conference Proceedings (NISSC)*, pages 141–149, 1997.

[150] John Black and Phillip Rogaway. Ciphers with Arbitrary Finite Domains. In *Cryptographers Track at the RSA Conference*, pages 114–130. Springer, 2002.

[151] GlobalPlatform. GlobalPlatform Device Technology, TEE Sockets API Specification v1.0.1, January 2017.

[152] Minolan. Format Preserving Encryption: JavaFPE. `https://github.com/Minolan/JavaFPE`, 2016. GitHub Repository.