Abstract. Today’s urban transport systems are part of everyday life for millions of people in the world. A lot of electronic ticketing systems have been developed to facilitate ticket acquisition, and automatic ticket validation. One of the most used technologies to store acquired electronic tickets are the contactless smart cards. These have three major advantages: great portability, easy to use, and great security levels. Smartphones powered with Near Field Communication (NFC), have been proposed to substitute smart cards, although for commercial and security reasons, these don’t have the same level of acceptance as smart cards. In this work a solution based on Android’s Host-based Card Emulation (HCE) technology is presented. This technology allows an Android application to emulate a smart card, living all the security details to application. This solution relies on a remote server to acquire new transport contracts and improve security.

Keywords: NFC, HCE, smart cards, smartphones, urban transport, ticketing, security

1 Introduction

Ticketing systems have been around for a while, since the traditional paper tickets, which are still used, to the modern electronic tickets. In the urban transportation systems, a technology widely used to store acquired tickets and automatically verify them before a client starts his trip, are contactless smart cards. These cards are easy to use, and users can even validate their electronic ticket without taking the card out of their wallets, which increases user’s comfort. Smart cards also have embedded secure hardware that offers great levels of security for both users and transport operators.

With the appearance of Near Field Communication (NFC) enabled smartphones, emerged the idea of use the smartphone as a substitute of all the smart cards a user has in his wallet. To emulate a smart card, smartphones were required to have available some secure hardware, where the ticket’s data would be stored. This, commonly known as a Secure Element (SE), can be another smart card connected to the smartphone, or some embedded hardware in the smartphone main board.

The only secure hardware that all smartphone manufacturers include in the devices, is a Universal Integrated Circuit Card (UICC) slot, that will be used to insert a mobile operator smart card. This card is in most cases not available to third party developers. To solve this lack of available SE’s, the Operating System (OS) Android, version 4.4, introduced Host-based Card Emulation (HCE). This technology does not require the presence of a hardware SE to enable the emulation of a smart card in a NFC capable smartphone, instead all the smart card data will be handled by the device main Central Processor Unity (CPU), and could be store in the device’s file system, or in a remote server. This might be a problem in the security of electronic ticket’s data since, in a compromised device, it could be manipulated or stolen by an attacker.

The main goal of this work is the usage of an Android smartphone, without SE, to store transportation contracts, with a security level similar to smart cards. This work
will compare the security and performance of two main solutions. The first solution uses a remote server as a SE. The second solution uses a technique called tokenization that will reduce the value of a contract, and also reduce its temporal validity, so that a user can use this contract only one time, for limited amount of time. This tokenized contract, called token, will be downloaded to the device and used completely offline. As it will be later demonstrated, tokenization, has great performance and a good security level.

1.1 Outlining

After this short introduction some related work will be presented on section 2. Next on section 3 the implemented system will be presented, and on section 4 the obtained results will be presented. Finally at section 5 the work conclusions will be presented.

2 Related Work

In this section the related work will be presented. The main focus will be on contactless smart cards, and how smartphones with Near Field Communication (NFC) were used to substitute transport cards and credit cards in bank payments.

2.1 Contactless Smart Cards

Contactless smart cards are an easy way to store and validate electronic tickets, or a contract as it is called in public transportation systems. These smart cards can host several contracts, depending on the card being used, and the data model used to represent a contract in a card. In this work the cards analyzed will be the ones used in the city of Lisbon, which are two: Viva Viagem/Sete Colinas e Lisboa Viva. Both these cards are compliant with the ISO/IEC 14443-B[1].

2.1.1 Viva Viagem/Sete Colinas These cards are the most simple contactless smart cards, and also the most inexpensive, being generally made of paper. These smart cards also do not offer any kind of embedded security mechanism. All the security in this cards must be provided by the ticketing applications that write the contracts in the card. For those reasons these cards are only used to store contracts with low value, i.e. occasional trip tickets.

The card model used for this tickets is CTS512B, which has only 64 Bytes of memory[4]. Due to its small memory, using the Lisbon contract data model, it is only possible to store one contract in these cards.

2.1.2 Lisboa Viva The Lisboa Viva card, is a more robust and complex card. It is generally made of plastic and it is much more resistant than the previous cards. These cards have embedded a small processor, which allows the card to run a specific communication protocol. This protocol includes authentication mechanisms that guarantee the authenticity of all data written, which is a great improvement in security when compared to the previous card. This is possible because these cards have secret keys that only the card itself can access. All the cipher algorithms used by the protocol are also implemented in the card. These are known as Calypso cards.

These cards also have more memory and they could store at least four contracts simultaneously. Since these cards have higher security levels, any kind of contract can be stored on it, including the long term contracts.
2.2 Secure Application Module (SAM)

A SAM is a smart card used on validators and vending machines, in a ticketing system. This smart card, which is used with both the previous cards, has secret keys stored, and is able to perform some cipher operations. In the first case, the Viva Viagem/Sete Colinas card, is used to generate a Message Authentication Code (MAC) of the whole card data, when some data in the card is updated. The generated MAC is then written in the card along with the contract. This way when the card is presented to another validator or recharge machine, it can verify the authenticity and integrity of the card data. It is important to note that this is a requirement of the used data model.

The interaction between a SAM and a Lisboa Viva card is slightly different. The data model used in this card does not require the use of a MAC because the Calypso secure session mechanism required to perform changes in the card, assures that only authorized applications change contracts in that card.

2.3 Use Cases

After explaining the details of the contactless smart cards used in transportation system, it is important to explain what are the use cases. The use cases are equal for both the cards, and they are the follow:

1. **Acquisition** - The acquisition is the process in which a user acquires his card. For the Viva Viagem/Sete Colinas card the process is as simple as buying one in an automatic vending machine or a box office. For the Lisboa Viva card it is not that simple. The acquisition of this card requires the user to perform a registration as a client of the transport operator, and the card will be personalized with the client name and photograph. This card can only be used by the client himself, and cannot be borrowed.

2. **Loading** - Loading is the procedure where a client loads or reloads his card in the automatic vending machines or box offices.

3. **Validation** - Validation is the procedure where a client validates his right to use the transportation system. In some systems where there are gates to access the transports, only after a successful validation, the gate will open and permit the client access.

4. **Inspection** - Inspection is a procedure that can be executed after a client started his trip. At this point an inspector can ask the client for his card, and perform a secondary validation. This validation will check whether the client can or cannot be using that specific transport.

2.4 NFC with Secure Element (SE)

NFC is a technology that allows short range communications. This technology is specified by the NFC-Forum and it is based on the inductive coupling principle, which allows two inductive circuits to share power and data at a few centimeters distance. NFC specification defines three working modes:

1. **Reader/Writer mode** – In this mode a NFC device is capable of reading and writing in NFC-Forum specified tags.

2. **Peer-to-peer mode** – In this mode two NFC devices can share data with each other.

3. **Card Emulation mode** – In this mode a NFC device behaves like a contactless smart card to an NFC reader.

For this work it is relevant the card emulation mode. A smartphone in this mode is expected to have an available SE, to where the messages received by the NFC chip should be forwarded. A SE is a hardware module that is tamper proof, and could run code and store data. With this configuration the emulated card data is stored in the SE, and the security level is the same or close to the one offered by a Calypso card.
2.4.1 Available SE’s There are some types of SE that could be used in a smartphone:

1. **Universal Integrated Circuit Card (UICC)** – this is usually a smart card. In a smartphone this may be the mobile operator SIM card.
2. **Smart microSD** – this is a microSD card for data storage, which additionally has embedded a smart card.
3. **Embedded SE** – this is a circuit embedded by the manufacturer in the smartphone’s motherboard.
4. **Trusted Execution Environment (TEE)** – this is an execution environment existent in some devices that allows code to run in parallel with the device’s Operating System (OS). TEE isolates its hardware and software such that any application running in the OS cannot access any code or data from this environment, even when a device is compromised.

There are a few systems, like the ones presented in [15], [17] and [18], running with this configuration, yet the lack of uniformity in what SE’s every smartphone should have available, and the denial of access by the mobile operators to the SIM card to third party developers, led to a low acceptance of this solutions by the industry.

2.5 NFC with Host-based Card Emulation (HCE)

HCE was introduced by Google in the version 4.4 of Android OS. This was created to enable any smartphone running that Android version, to emulate a smart card, even if it doesn’t have a SE available. With this technology the messages received by the NFC chip are forwarded to the smartphone Central Processor Unity (CPU), where a designated application will handle the request. This is far less secure than a SE based solution, yet it is a lot easier to develop a system with this technology than with a SE. It also has the advantage that no secondary hardware has to be bought by the user.

Besides the advantages there is also one big disadvantage that has to be considered: if the device is compromised, something that is quite easy, the card data and code that is supposed to be secure, can be exposed to an attacker.

2.5.1 Online SE The easiest a most secure way to use HCE is to have an online SE. In this mode a SE is securely stored in a remote server to which the application must connect and deliver the received messages. Pascal Urien proposes a system with remote SE’s[19]. In this system the remote server has several grids of SE’s installed to be accessed by the remote smartphones.

This architecture has a major problem which is the need for the NFC device to have an internet connection. That can still be a major limitation in a ticketing solution.

2.5.2 Tokenization An alternative to the online SE is a technique called tokenization. This technique is basically the transformation of something, in this case the smart card data, to a equivalent representation with limited value and validity. Since this token has low value, and a very short validity, it can be stored in any smartphone and be used when the device is offline. BellID proposed a solution with HCE and the tokenization technique in [20]. This is a solution for bank payments with NFC smartphones, and here a token is a credit card data transformed so that it can only still be connected to the original credit card bank account, although it can only be used for a limited amount of time. An interesting feature of these tokens is that even if an attacker gains access to the token he will not be able to obtain the original credit card data.
3 Solution

In this section two solutions for the presented problem will be presented: one based in the online Secure Element (SE) presented on section 2.5.1, and another one based in the tokenization technique presented on section 2.5.2.

This section starts by presenting the use cases that must be supported in this solution. Next the system components and its respective roles in the whole architecture. Then it proceeds to the details of each solution, and finally the implementation of the components.

3.1 Use Cases

The use cases of the proposed solution are identical to the ones presented before in related work (section 2.3), in the context of the actual solutions based on contactless smart cards. Yet there are some modifications to the use cases. The new use cases are the following:

1. Acquisition - This use case is different because the user no longer gets a card. In the proposed solution a user will instead register himself as a client.

2. Loading - The loading use case is divided in two use cases:
   - Load Balance - In this use case a user loads to his account a balance, and pays the respective amount.
   - Load Product - In this use case the user uses the loaded balance to load a contract to his card.

3. Validation - To validate a contract a user must present his smartphone to the validator machine.

4. Inspection - In case of inspection, the user must present his smartphone to the inspector.

3.2 Ticketing System Components

A ticketing system is composed by some distinct modules with different responsibilities. The proposed solution components are based on current ticketing systems, and some efforts were made to use as many components as possible from the current solutions. The proposed solution components are depicted in figure 1, and they are the following:
Secure Application Module (SAM) Master – This component is responsible for generating and distributing all the system’s secrets keys used, generally done only once in the initialization of the system. This component will be used as it actually exists.

Validators – This component is the one that automates the process of verifying the user ticket before he begins his trip, and grants him access to the service. This component could require some adaptation to communicate with smartphones. In this work the validators already supported this feature.

Inspectors – An inspector is a component that verifies if a user has access to transportation service he’s using. Like the validators, the inspectors used already support smartphones.

Cloud Service – This component will combine multiple responsibilities as client registration, ticket sale, transactions collection, or fraud detection. This component interacts with all the remaining system components. Current systems already have a component.

Mobile Device – The mobile device is a component running in a smartphone, and will be used as a substitute of the user smart cards generally used. This component should be used to acquire new tickets and to access transport service.

In this work the identified components that will actually be implemented are the Cloud Service (CS) and Mobile Device (MD). The remaining components will be integrated as they already exist.

3.2.1 Components Interaction In figure 1 is illustrated how the components interact with each other, and how data flows between them. Since there is a lot to say in this subject, this section will only focus on the most relevant components for this work, which are considered to be MD, CS and validators.

The CS, a server, is the central component which will enable the MD, a smartphone, to be used as proximity card when presented to a validator. To make this possible the client must first use the MD to register himself as a user. After this step the user can load a balance to his account, that can be used later to load ticketing products. Once a user has some product loaded he can present his smartphone to a validator and use the transport service. The validator will perform some reads and writes in the card to perform the validation, and once it is completed it will create a record (or transaction) that will store information about the card used, the loaded contract, and also some other information about that specific service. This transactions are later sent by the validator to the CS that will use them to calculate revenues based on sales and validations, in a process called reconciliation. The CS will also use reconciliation to find possible cases of card misuse. In these cases CS can create a blacklist with these cards and send it to all the validators, that will deny the access of transport service to every listed card.

3.3 Implementation

After the explanation of how the components fit together, this section will describe some details of the implementation of the components.

3.3.1 Registration The registration is the process that allows a client to start using the system. In this process the user is required to insert some personal information as name and email address. The user will also be asked to add a password and a four digit numeric code (from now on called PIN code). The password will be used for the user to authenticate himself to the server. The PIN code will be used to derive a cipher key, and its application is explained in next section.
3.3.2 Loading  As explained in the solution use cases (section 3.1), loading is divided in two phases - balance load and contract load. In the first phase the user loads some amount of money to his account, and pays using an online payment service. After this step the user is able to load some contracts in his card, and the price of the contract loaded will be subtracted in user’s balance.

The user card is a concept created and prototyped called Virtual Card (VC), that will act as a SE. Two different techniques were used to enable MD to access VC: online VC and tokenization.

**Online VC** When a new user account is created, a new VC is created and associated to this user (VC structure is explained next in section 3.3.3). Then the user can load products to this card to be used later in validation. Note that with this technique the smartphone doesn’t store any card data locally, and it will need to be online to access the VC.

**Offline VC with tokenization** The online technique presented before is not useful when the MD isn’t connected to the CS. So an offline solution is proposed here. This solution is based on the creation of a token from a VC. As explained before, a token is another VC with some imposed limitations, specifically in value and temporal validity. So the user must input the time of the day that he will travel, and the token will only be valid after that time until one hour later. After this time the token will be expired and cannot be used. Also the token has low value, that means it has a contract loaded that can only be used in one validation.

The token, after created, is downloaded to the user’s smartphone and is stored locally. Before the token is sent to the smartphone it is ciphered with the key derived from the user PIN code. The token will be stored ciphered in the smartphone until the moment the user chose to start using the token. At this point the user needs to insert his pin code, that will decipher the token making it ready to be validated.

With these limitations the risk of having the VC stored in the smartphone file system decreases, since after the validity time expired the token cannot be used anymore. In cases that the token was never validated, the CS in the reconciliation phase will verify that a sold token wasn’t used and the user will be refunded. In case of a token being used more than once during its validity, the CS can blacklist that user.

3.3.3 Validation  The validation will occur when the smartphone is tapped to a the validator reader. At this point the validator will read the contract data, validate the contract, and write in the card some information about the validation, and if it is the case subtract the value of the trip. As it was explained before, the smartphone has two ways of access a VC, online or offline, depending the mode the user has chosen, the read and write messages will be handled by the CS or by the smartphone, respectively.

**Emulated Card** The validator used in this work already supported smartphones as proximity cards, and it requires the smartphone to emulate a CTS512B proximity card. To do this the VC, in the cloud or as token, must have a memory map of 64 bytes at least, and an interface that allows read and write operations with byte granularity.

The communication protocol used to communicate between the validator and the smartphone is specified in ISO/IEC 7816-4[21], in which are defined the commands for data interchange in proximity cards, or Application Data Unit Protocol (APDU). This is a requirement imposed by Android Host-based Card Emulation (HCE)\(^1\). The commands implemented are: Read Binary, Update Binary, and Select AID. The read

\(^1\)Android HCE - [https://developer.android.com/guide/topics/connectivity/nfc/hce.html](https://developer.android.com/guide/topics/connectivity/nfc/hce.html)
command contains an address and a length, and it allows to read a number of bytes
starting at the specified address with the given length. The update command allows
to write a given byte sequence starting at the specified address. The select command
is also required by Android, that will use this command to bind the message to an
application, and the application must recognize this command and simply reply with
the application identifier.

4 Results

This section will present how the solution was evaluated and what are the results.
The evaluation consisted in testing the solution’s performance against a traditional
smart card solution, and performing a threat analysis to the implemented solution.
The obtained results are presented next.

4.1 Performance Analysis

The tests to evaluate the performance of the systems were based on the validation
transaction time measured in a reference system with a CTS512B proximity card, and
the Virtual Card (VC), both online and tokenized. The tests were run in an application
developed specially for this purpose that uses a ticketing framework, called Ticketing
Kernel (TK), that will connect to a Near Field Communication (NFC) reader and
do all the validation work. The reader used was an ASK 518-U, and the smartphone
used was the LG Nexus 5 2013. For every card (CTS512B, VC online and offline), a
thousand validations were recorded.

![Online Virtual Card Statistics](image)

**Fig. 2.** Statistics obtained from the CTS512B and online VC.

4.1.1 Online VC To test the online VC two different network connections were
used: a WiFi network with local connection to the server, and a WiFi network with
connection to the internet via mobile 3G network. The measured download and upload
bit rates for each network were, respectively: 126.18 Mbps and 8.64 Mbps for local
network, and 2.64 Mbps and 1.24 Mbps for 3G network.
In the figure 2 are illustrated some statistics calculated from the transactions recorded with the card CTS512B and VC accessed from the two different networks. Here we can observe that the connection quality highly affects the performance of the system, since in a local network the transaction times are close to the reference times of a CTS512B. With this experiment we can conclude that the extra time consumed in the online model, is mostly associated with the communication between the smartphone and the Cloud Service (CS).

To understand if there was something that could be done in this solution to improve the transaction performance, the commands sent by the validator to the smartphone were logged. By observing the commands the following pattern was identified: one select AID command, followed by two reads, and two writes. All of these commands are always executed in the server, what makes a total of five messages exchanged. This could be improved if the select AID command was executed offline, since it does not need any of the card data, and the read commands were only executed once in the server reading all the card content, and cached in the smartphone memory, during the transaction, to avoid executing the second read in the server. The writes could also be improved by modifying the validator, since it can actually write all the differences in only one operation. With these modifications the communication time can be improved from up to 60%.

4.1.2 Offline VC Figure 3 shows the comparison of the measured transaction times between a CTS512B and a VC offline (or token). Here we can observe that the average transaction time of a CTS512B is approximately 50 ms higher than the VC in the smartphone. Since the number of messages exchanged is exactly the same, and the contracts stored in each card are equal, we can assume that the lower times obtained in the smartphone are due to fastest processing and possible higher NFC bit rates.

4.2 Threat Analysis

To evaluate the security of the proposed solutions a threat analysis was done. The threats presented next were identified using the STRIDE model, and the risk of each one calculated with the DREAD model.
<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Disclosure</td>
<td>A VC token after deciphered is vulnerable to copy if the device is compromised. If an attacker accesses this token he can use it to access the transportation services.</td>
<td>Medium</td>
</tr>
<tr>
<td>Information Disclosure</td>
<td>During online validation, all the VC data is read. This way an integral copy of all the card data is moved to the smartphone where it might be vulnerable.</td>
<td>High</td>
</tr>
<tr>
<td>Information Disclosure</td>
<td>If an attacker knows the read Application Data Unit Protocol (APDU)'s used by validators, he can use an NFC reader to clone the VC.</td>
<td>High</td>
</tr>
<tr>
<td>Denial of Service</td>
<td>If an attacker knows the write APDU's used by validators, he can use an NFC reader to invalidate the card.</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 1. Identified threats, and his risk to the system.

Table 1 shows the threats to which the solution is vulnerable, that could lead to information disclosure and denial of service. The presented threats regarding information disclosure can be handled after in the reconciliation phase, yet the denial of service is difficult to resolve but is also difficult to reproduce since it requires physical access to the smartphone. Next are presented some improvements that can be implemented to reduce the exposure to this attacks:

- Detect when Android allows elevation of privileges: if elevation of privileges is detected this means that the files stored in the file system can be accessed by any application, and is not secure to store a token locally.
- Emulate a new virtual card in the server that implement some secure protocol like Calypso: this way the attacker could no longer read the whole card data even if he knows which are the APDU’s used, neither write in the card without being successfully authenticated.

5 Conclusion

In this work, a solution was developed that enabled a regular Android Near Field Communication (NFC) smartphone to be used to access the transport services. There are already some solutions that do the same, yet all of them require that some secure hardware, called a Secure Element (SE), is present on the smartphone. The SE is then used to store the transport contracts and and directly communicates with the NFC chip, providing a higher security level.

In this work was created a prototyped Virtual Card (VC). The VC is the equivalent to the SE where the transport contracts are saved, and there are two ways for a smartphone to access the VC, online and offline. The online method has the VC stored in a server database, and the smartphone needs to be connected to the server during the validation. Alternatively it was created another method that allows the smartphone to access a VC offline. It is called tokenization and it defines that a VC must be created as a token with a limited temporal validity, and reduced value. So in this implementation the token can only be used during one hour, and the user can specify when the validity starts in the current day. Also the contract loaded in the VC can only be used once, so after the first validation it can not be used anymore.

To increase the security of the token downloaded to the smartphone, it is ciphered with a key derived from a PIN code. So before starting the validation, the user must insert his PIN code to decipher the token.

The obtained results show that this solution can be easily adopted by the public transport systems, since it does not require major changes in the available validators. At this time we believe that the solution that better fits the public transports...
requirements is the token solution, based on the great validation transaction time, and the satisfactory security level. The online mode still requires some improvements, specifically: a emulated card with a secure communication protocol, and the number of messages exchanged. With both this aspects improved, and the continuous evolution of wireless networks, we believe that in long term this method can become the primary solution used.

References

5. Calypso, Calypso Specification Secure Application Module SAM-C1, Calypso Networks Association, Revision 3, 1 October 2010.
7. INOV, SAM Study, D1.1 - SAM Study, INOV, 2014
9. Darian Škarica, Hrvoje Belani and Sanja Illeš, Implementation and Evaluation of Mobile Ticket Validation Systems for Value-Added Services, University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Telecommunications.
12. Simalliance, NFC Secure Element Stepping Stones, version 1.0, (July) (2013) 1-75
13. Cards, S.: TS 102 613 - V7.3.0 - Smart Cards; UICC - Contactless Front-end (CLF) Interface; Part 1: Physical and data link layer characteristics (Release 7). 0(Release 7) (2008) 1-57