NFC-mobile Payment System Based on POS Terminal Authentication

by

Bader Munif Aldughayfiq

Submitted in partial fulfilment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2014

Dedication

This work is dedicated to my father and mother for all the support they gave me, and to all the sacrifices they made to guide me to a successful life. Moreover, I would to dedicate this work to my wife, who support me and made my life full of happiness. I dedicate this work also, to my brothers and sisters for their advices and their support through my study. Lastly, I dedicate this work to my newborn daughter “Jouf”, who brought all the joy to our family, wish for you a long and happy life.

Table of Contents

List of Tables

Table of Figures

Abstract

Payment development has increased rapidly in recent years. A most recent development is contactless mobile payment systems that use NFC-enabled phones. Therefore, many researchers have proposed payment systems that use NFC-enabled phones in attempt to achieve availability, simplicity, security, and privacy in a transaction. Moreover, NFC can be subject to a number of attacks and more specifically a recent attack called relay attack. In this situation, an attacker will extend the range of communication using NFC devices and make an unauthorized payment with the victim's device. This thesis proposes a new mobile payment system using an NFC-enabled phone. Our proposed system is based on POS authentication by using the ability of NFC devices to read tags, where this tag will contain a random message generated by the POS. The proposed system also uses a new cryptography approach that offers a dynamic pre-shared symmetric key mechanism to protect banking information.

List of Abbreviations and Symbols Used

Abbreviations

NFC	Near Field Communication
RFID	Radio-frequency identification
SMS	Short Message Service
WAP	Wireless Application Protocol
POS	Point of Sale
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
RF	Radio Frequency
P2P	Peer-to-Peer
UICC	Universal Integrated Circuit Card
HCE	Host Card Emulation
SIM	Subscriber Identification Module
AID	Application Identifier
APDU	Application Protocol Data Unit
EMV	Europay, MasterCard and Visa
ID	Identification Number
XOR	Logical Exclusive OR operation
GPS	Global Positioning System
GSM	Global System for Mobile Communications
PRNG	Pseudorandom Number Generator

Symbols

Kc	Session Key
K_i	Encryption Key
R, Rand.	Random Number
StoreID	The Store Identification Number
Sign.	The Signature Message
ACT#	The Account Number
Enc.	Encryption Process
Dec.	Decryption Process
CMP	Comparison Operation
S	Signed Message
S_1	Hashed Signed Message
TS	Time Stamp
T, t	Time

Acknowledgments

I would like to acknowledge Al-Jouf University, who funded my studies at the Dalhousie University. Without their guidance and funding I was not able to get my master degree. Also, I would love to acknowledge and thank my supervisor Dr. Srinivas Sampalli for his guidance and effort to make this moment happen to me. Finally, I would to acknowledge and thanks my family and my friends, who supported and helped me during my master studies.

Chapter 1 Introduction

1.1 Overview

In recent years, the aim of payment's systems development was to provide security, simplicity, and availability of customer transactions. Therefore, a number of payment methods were developed through the years. Figure 1.1 shows the history of the evolution of retail payments in Canada [2]. These methods vary in terms of the service provided, how secure the transaction is, and the availability of a payment method. However, no complete security and availability were achieved and development kept growing rapidly to merge new technologies to develop a secure payment method. For example, nowadays we have a variety of payment methods that have been merged with technology such as online payment, prepaid cards, mobile payment, and contactless payment. In the next sections we take a closer look at the new payment methods.

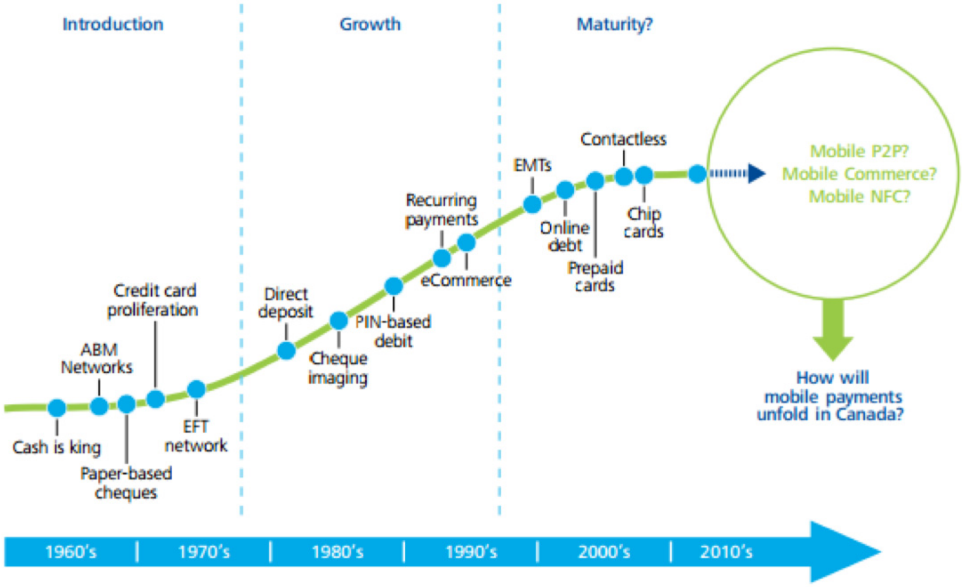


Figure 1.1 the history of the evolution of retail payments in Canada [2].

1.2 Contactless Payment

Contactless payment is a technology that transfers data stored in the internal memory chip to a reader using a radio frequency (RF) communication channel. In addition, contactless smart chip communication relies on the ISO 14443 standards that define two types of cards (i.e. type A, and type B) and a communication distance up to 10 cm. Contactless technology has many advantages such as simplicity, secure transaction, and convenience. The Contactless technology facilitates the use many applications such as access control and contactless payment which has been developed with contactless technology [1]. In 2005, contactless payment technology started to take place in payment systems development in North America. A year later, contactless payment technology became reality with the issuing of millions of contactless cards in the United States [1]. For example, American Express issued over two million cards with ExpressPay contactless payment technology and other banks issued more than five million cards with MasterCard PayPass contactless payment technology [1].

1.3 Mobile Payment

The continuously increasing growth of developments in mobile technologies opened new opportunities to maximize the benefits of using mobile on a daily basis. Nowadays, mobile phones have great capabilities for the development of new applications, which are engaged in our daily lives. Moreover, mobile phones provide a new style of life, where the availability of many applications such as Internet browsing, sending and receiving e-mails, and mobile payment is essential.

Thus, mobile payment introduced new methods of payment, which are convenient and available to the customer all of the time. Mobile payment is supported with many mobile technologies such as Short Message Service (SMS), Wireless Application Protocol (WAP), and wireless communication technologies [4].

1.3.1 Application of Mobile Payment System

In this section we will provide a summary of the most used mobile payment methods available as mobile applications. First, the Internet payment method uses the cell number to pay for products over the Internet and adds the amounts to the cell bill then receives a confirmation message through SMS. Second, payment for mobile commerce applications is used to pay for a service in a web-based application using WAP to access an online application, and SMS to receive a confirmation message of the payment [4]. This includes, for example, movie tickets applications such as MovieTickets.com [9], Tribute.ca [10], and cineplex.com [11]. Advantages of these methods are fast and secure transactions, and no additional components on the merchant side are required. A main disadvantage of these methods is that they only apply to fixed amount services [4]. Third, the Person-to-Person payment method is based on wireless communications technologies or WAP technology, and mainly is used to exchange money between two devices [4]. If a person owes you money (e.g. $200) that person can transfer that amount of money through a Person-to-Person payment method. An example of that service is interac.ca [12], where they use email to transfer money between two parties. Unfortunately, the down side of this method is that both parties who want to use it need to be registered with the same service provider [4].

1.3.2 NFC and Mobile Payment

A new technology in payment system development is called contactless mobile payment using Near Field Communication (NFC) technology. The Near Field Communication (NFC) technology is used to make a contactless communication with a Point of Sale (POS) terminal. NFC is developed over Radio Frequency Identification (RIFD) standards including the ISO 14443 standards. Nowadays, mobile industry leaders are offering a variety of NFC-enabled phones. This makes integrating mobile payments with contactless payments more logical in terms of simplicity and availability. The Smart Card Alliance Contactless Payment Council states in their 2007 proximity payments paper that "The convergence of payments and mobile communications is not just logical, it is inevitable" [5]. Many financial organizations (e.g. Visa, Master Card, American Express, etc.) and telecommunication companies (e.g. Verizon and T-Mobile) collaborated and launched this payment method using NFC technology [13].

1.4 Payments Security

Security and privacy concerns started to rise because of the security threats of NFC technology. A security advantage of NFC is the close range of communication, which makes it resistant to most attacks, but since NFC is built over RIFD that makes it more vulnerable to similar attacks launched on RIFD [14]. A more concerning threat related to mobile payments is the ghost and leech attack, where the attacker will relay the information between a tag and a reader. Specifically, the attacker will use two NFC enabled devices one called "ghost", to relay information from the legitimate POS to the second device called "leech" using a communication channel. Then, the leech will relay that information to the legitimate card [13].

1.5 Objective

The objective of this thesis is to present a proximity payment system with a novel security solution. The solution aims to authenticate the POS to the phone by scanning an NFC tag containing a hashed message generated by the POS. Also an encryption mechanism is used to secure the sensitive data (i.e. banking information) that had been transferred during the transaction. The goal of the proposed system is to enhance the privacy and security in current proximity payment systems.

1.6. Organization of Thesis

The rest of the thesis is organized as follows. Chapter 2 discusses background, related work, and proposed proximity payment systems from other publications. Chapter 3 presents the motivation of this thesis. Chapter 4 describes the framework of the proposed system and its resistance to the relay attack. Chapter 5 presents the implementation of the proposed system as well as analyzing the experimental results and security aspects of the proposal. Chapter 6 will conclude the thesis and discuss possible future work.

Chapter 2 Background and Related Work

2.1 Background

2.1.1 Contactless Payment System and Mobile Payment System

Contactless smart chip technology has been used for a variety of systems such as access control and contactless payment. The contactless smart chip technology runs over radio frequency communication (i.e. RIFD or NFC), where it uses a small memory chip to store sensitive data needed and has an antenna to communicate with a radio frequency interface [1]. Contactless payment is a popular application of contactless technology, which has been implemented for credit cards to provide secure, simple, and fast transactions.

Mobile phones are spreading widely and rapidly around the world, and they are becoming one of the most important aspects of our lives. Nowadays, mobile phones are used for a number of applications such as SMS, and WAP. Thus, mobile payment applications have become a logical step to combine both technologies (mobile payment and contactless payment) as developed by both the mobile industry and the finance industry [5]. Both industries have developed a number of applications using mobile phones' existing applications. Moreover, the proximity mobile payment using NFC is rapidly spreading, where a customer will use NFC-enabled phones to make a payment at the Point of Sale (POS), using a radio frequency interface [15].

2.1.2 Near Field Communication (NFC)

Near Field Communication (NFC) is a subset technology relies on Radio Frequency Identification (RFID) standards. RFID is a wireless technology used to identify objects. It consists of a tag, a reader, and a backend server. The tag is a microchip that contains a unique identification number. The reader can read the tag from few millimeters up to a hundred meters. After the reader obtains the tag's unique ID, the reader sends it to the backend server for further processing [45].

Near Field communication (NFC) is a short range wireless communication technology that operates on 13.56 MHz frequency with a data transfer speed up to 242 Kbit/s. NFC is based on the smart card standards ISO 14443, which makes it compatible to emulate the current contactless smart cards. Furthermore, NFC made the transferring of data between two devices simple and secure, when the communication range is less than 4cm [8][15][17]. The comparison to other wireless communication technologies is shown in Figure 2.1 [18].

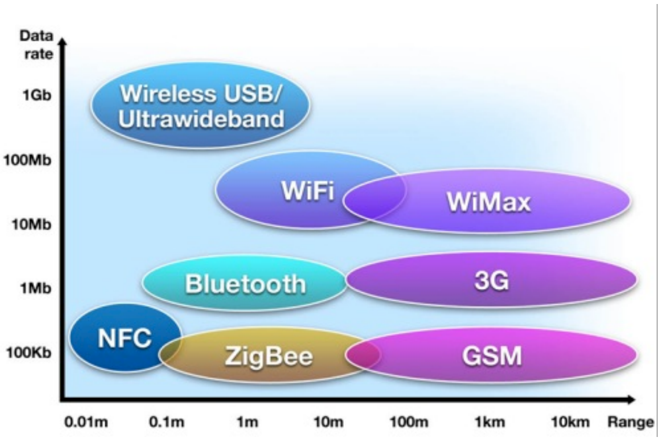


Figure 2.1 is showing comparison of the wireless technologies data rates and ranges [18].

In recent years, NFC technology is deployed in a number of smart phones such as Google Nexus, Samsung Galaxy, and BlackBerry Bold 9790. NFC-enabled phones have a built in NFC-chip and antenna, which enables the phone to transmit information to another NFC device. Moreover, the NFC-enabled phone is able to act as a contactless card, and functions also in a reader mode. Therefore, smart phones with NFC capabilities have a great potential to be used for several areas such as payments, healthcare applications, and ticketing in the transportation systems [19].

2.1.3 NFC- Enabled Phone Operating Modes

The NFC-enabled phone can operate in three modes. Each one of these modes will serve a different propose depending on the user's needs. In the next sections we will describe briefly the function of the three modes. Figure 2.2 shows the different NFC operating modes [17].

2.1.3.1 Reader Mode

In the reader mode the NFC-enabled smart phone is used to read and write data over NFC passive tags. The phone will send a power signal to the tag, and starts one-way communication from the active device to the passive device. An example of this mode, is an NFC tag attached to a poster, where a person can gather information about that poster from the tag using the reader mode [8][19].

2.1.3.2 Peer-to-Peer Mode

In the Peer-to-Peer (P2P) mode NFC technology is used to communicate with another NFC-enabled device through radio frequency. After this, a communication channel is established to transfer data between the two devices [8][19]. An example of this mode will be transferring pictures between two NFC-enabled phones through the established communication channel.

2.1.3.2 Card Emulation Mode

This mode will allow the phone to act as a contactless card based on ISO 14443 standards. If the mode is active, an external reader will not differentiate between a phone and a contactless card [8][19]. This mode will able the phone to be used in a number of systems such as access control and payment.

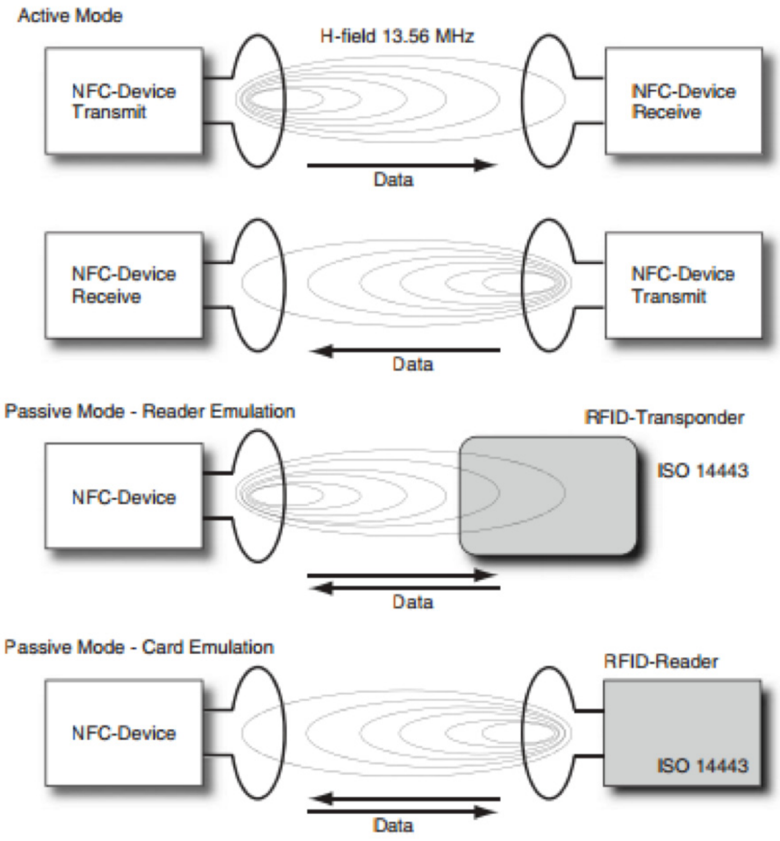


Figure 2.2 NFC operating modes [17].

2.1.4 NFC Architecture in Smart Phones

A part of the NFC architecture in the smart phone is a secure element, which will enable it to operate in card emulation mode. The secure element is used to store sensitive information securely. The secure element can be implemented as either a hardware module (i.e. embedded hardware, or Universal integrated circuit card UICC), or a software-based secure element. Beside the secure element, NFC architecture has two more parts the NFC controller and the host controller. The NFC controller is an analog digital converter that converts the signals transferred over a proximity connection to the secure element and the host controller. The host controller is used to enable authorized applications to exchange data with the secure element. Figure 2.3 illustrates the architecture of an integrated NFC within smart phones [8].

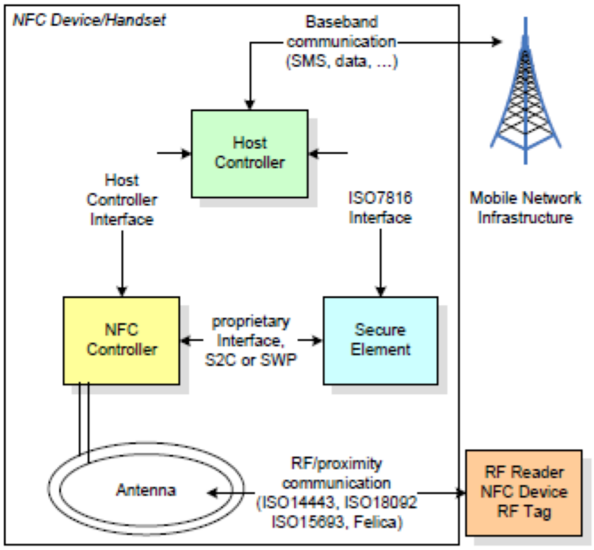

NFC Device/Handset
Baseband communication (SMS, data, ...)
Host Controller
Host Controller Interface
ISO7816 Interface
Mobile Network Infrastructure
NFC Controller
proprietary Interface, S2C or SWP
Secure Element
Antenna
RF/proximity communication (ISO14443, ISO18092 ISO15693, Felica)
RF Reader NFC Device RF Tag

Figure 2.3 NFC architecture in smart phones [8].

2.1.5 NFC-Enabled Phone Payment

With the increasing use of NFC-enabled mobile phones, financial organizations have started to provide proximity mobile payment service using NFC-enabled phones to consumers. A payment application (e.g. American Express ExpressPay, Discover Zip, MasterCard PayPass, and Visa payWave) is installed in an NFC-enabled phone and it is personalized with account information issued by the customer’s bank or finance organization. Simply, the customer uses the application by taping the phone to the POS terminal in order to send the information needed to complete the transaction. For example In the US, Google and a group of mobile carriers called Isis are launching payment applications using NFC-enabled smart phones [15].

Essentially, any payment system is formed by three factors: customer, a merchant, and a financial service provider. Moreover, a more advanced payment system will contain additional factors such as mobile operator and application provider. Figure 2.4 shows the number of stakeholders who may be involved in the payment system [5]. For example, Google is providing a payment application (i.e. Google Wallet) with MasterCard PayPass service into the payment system to enable the customer to use their phone to pay for goods. Mobile carriers like Isis group who are collaborating with American Express are using a payment application in the secure element to be used for payment [15][5].

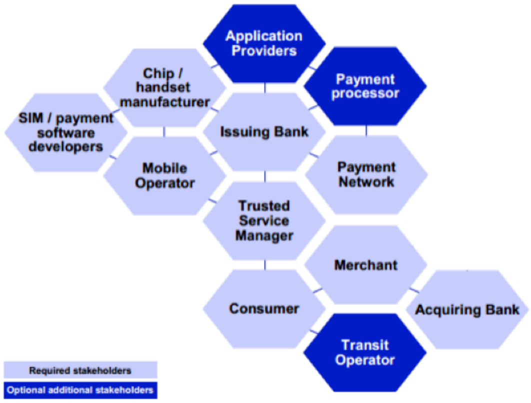


Figure 2.4 Mobile Payment Stakeholders [5].

Many of the mobile payment applications in NFC-enabled smartphones need either an embedded secure element to emulate the contactless card, or a SIM card from the mobile carrier that includes a secure element. Usually a transaction coming through NFC interface is handled by the secure element and no unauthorized applications are involved. Authorized applications can communicate with the secure element after a transaction to be notified about the transaction state [20].

However, Google Android introduced a new service called Host Card Emulation (HCE) to emulate a card without the need of the secure element. The new service will enable any application to emulate a card and manage the security of sensitive data [20]. Moreover, HCE service will enable the application to communicate directly with the external reader. For example in a case of payment, the data is transmitted through NFC controller to a running application over the host CPU. Figures 2.5 and 2.6 illustrate both methods for emulating a card in smart phones [20].

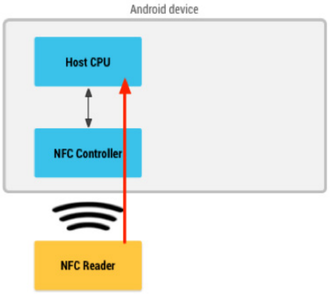


Figure 2.5 NFC card emulation without a secure element [20].

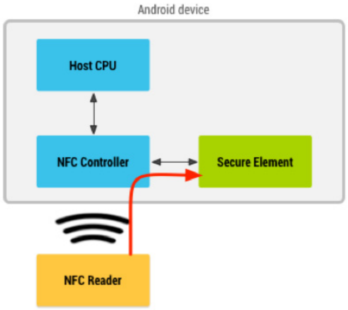


Figure 2.6 NFC card emulation with a secure element [20].

Since the HCE is provided as a service by the operating system, the service will be running in the background of the operating system. Therefore, the user only needs to tap the phone to the terminal in order to select the correct HCE service for completing a transaction. An HCE service will have a number of payment applications, each one defined by an Application ID (AID). A group of AIDs are defined into categories, for example all payment applications such as Visa, and MasterCard will be defined in the payment category. The NFC reader needs to select one of these AIDs most of these are reserved and known to the public, in order to communicate with an HCE service that handles the selected AID [20].

Moreover, security and privacy plays an important role in the payment industry. The next section will discuss the security threats to the NFC technology in general, and then related security concerns to NFC-enabled mobile payment system.

2.1.6 NFC Security Threats

Since NFC is used to make payments in mobile phones, security issues raise more concerns among consumers. In the payment system NFC technology is used to transfer sensitive data between a card and a reader with limited security standards provided by NFC specifications. This makes NFC a common target to a number of attacks. Moreover, due to the similarity between RFID and NFC, most of the attacks on RFID are applicable on the NFC [21]. In this section we will briefly overview the security threats shared that RFID and NFC face.

Due to the NFC being built with RFID standards, it inherits a number of security threats that targets RFID. The following is a summary of security threats related to both technologies:

- NFC-enabled mobile phones and NFC tags are vulnerable to an *unauthorized reading* attack. In this attack an attacker will target the information stored in a tag or an NFC device, by using an NFC device in the reader mode. For example this threat will become a high-risk when an attacker will attempt to target sensitive information like data used in payment, such as credit card information [21] [8].
- A *Falsification of content* attack usually will target information stored inside an NFC device or an NFC tag and forge that information. For example an attacker might attempt to tamper with the ID of an emulated card to disable it [21] [8].
- The risk of an *eavesdropping* attack is quite low, since the required communication distance to allow transferring data is less than 4 cm, which makes launching an eavesdropping attack on NFC device a low possibility. However, there are attacks that applicable such as replay attacks, where the attacker will replay the captured information from the card or device being simulated to the NFC reader. Another applicable attack is the relay attack, where an attacker will place rogue devices in a close range to the targeted NFC devices. An attacker will relay the data by using two NFC devices, which are connected to each other via a communication channel. An example of an environment to launch this attack will be in the payment systems, where the attacker will make a payment with the victim's card [21] [8].

2.1.7 Relay Attack

One of the recent worrisome attacks on RFID and NFC is the relay attack (i.e. Ghost and Leech attack). A relay attack will take advantage of NFC-enabled devices' features and extend the range of the wireless communication channel. An attacker will need three components to successfully launch the attack against a system that uses NFC or RFID technology. The components are as follows:

- A NFC- enabled device acting as reader (i.e. leech) Placed in close range with the victim card or phone that emulates a smart card,
- A NFC-enabled device to emulate a smart card (i.e. ghost) that is used to communicate with the legitimate reader, and
- A communication channel between these two NFC-enabled devices (e.g. Wi-Fi, and Bluetooth).

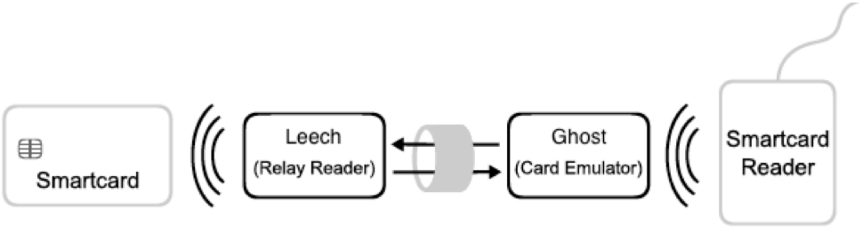


Figure 2.7 Relay attack setup with two NFC-devices [22].

An attacker will need to get the leech device into a range that enables the leech device to start a communication channel with the victim's card or phone. Meanwhile, the ghost device will be used as a card to communicate with a reader (i.e. Point of Sale (POS) terminal, and access control reader) First, the ghost device will start communicating with the reader, and then transferring any commands sent by the legitimate reader to the leech device. Next, the leech device sends the received commands to the victim's card or phone and waits for a response. The victim's card responds to these commands, where it thinks communication is established with the legitimate reader, with the required information sent to the leech device. Last, the ghost device forwards the received responses to the legitimate reader. Figure 2.7 shows the relay (ghost and leech attack) setup [22][23][24][25][26].

This scenario of the relay attack requires a physical contact with the victim's device or card. Therefore, a second version of the attack is software-based and was proposed [22] to target the victim's phone (i.e. emulating a card). In this scenario an attacker will develop a phone application to access the information stored in the secure element. This application will obtain authorized access through the application processor. This process is called an internal interface communication. Figure 2.8 illustrates the software-based relay attack and commands relaying in this scenario [22].

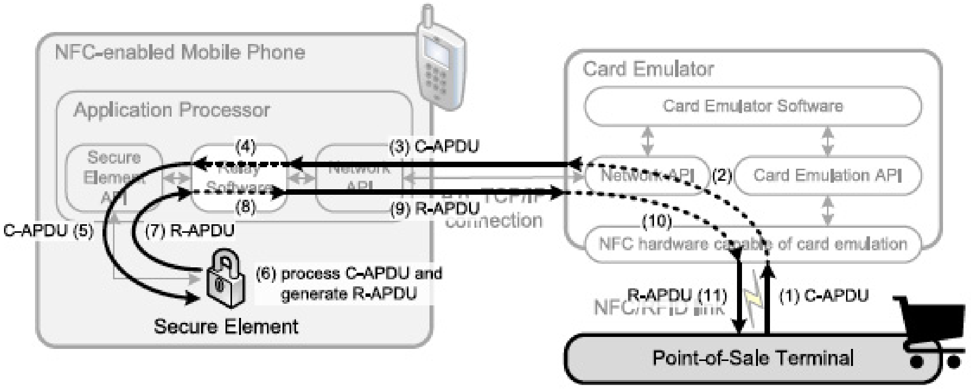


Figure 2.8 Software-based Relay Attack [22].

Furthermore, a practical implementation of the software-based relay attack was launched on the Google Wallet payment system [22], where the author used an NFC-enabled phone with Android 2.3.4 as an operating system, and the Google Wallet installed on it. The Google wallet application uses EMV (i.e. Europay, MasterCard, and Visa) payment standards. The author also developed a relay application (i.e. java-based Android application) to act as leech app. The application was installed on the NFC-enabled phone to communicate with the secure element requesting the information needed from a remote server that was connected to the card emulator. A TCP connection was established and managed by the relay app. The card was emulated using NFC-reader (ACS ACR 122U NFC reader) that is able to operate in the card emulation mode. This reader was connected to a desktop application developed and installed using a notebook computer. As a result, the author was able to launch this version of the attack with publicly available tools that can be acquired by an attacker at a cost of $400 to $500 [22].

In addition, many authors have implemented a practical relay attack on a smart card that relies on ISO 14443 standards such as [23] [25]. The risk of launching a relay attack on the currently existing proximity systems (e.g. NFC-enabled phone payment system, and control access system) is quite high. This is due to simplicity and affordability of lunching the attack, and that no application-level cryptography will prevent the relay attack [22].

As the focus of this thesis is preventing the relay attack, the next section will discuss the proposed payment systems that use the NFC enabled phone and their vulnerability to the relay attack. Moreover, counter measures will be analyzed to provide solutions to prevent the relay attack.

2.2 Related Work

The focus of this thesis is to propose an NFC-enabled mobile payment protocol, which is resistant to the relay attack. This section will discuss five proposed NFC-enabled phone payment systems, and analyze the proposed counter measures to prevent this type of attack.

2.2.1 Proposed NFC-Enabled Phone Payment Systems

The first system, called mFerio, was proposed by [27]. This system is a P2P mobile payment system using NFC technology. In this system a user will make a payment using two NFC-enabled devises, which are not connected to a backend server, based on the digital cash concept. The authors claim their system will provide more security and usability than the cash based payment system. The two claims are based on using NFC technology and the availability provided by the mobile payment system. In terms of security, the proposed system relies on two aspects, the physical security aspect and the user security aspect. First, the physical security aspect is defined as using an embedded secure storage with the phone (i.e. the secure element) to store data needed for a transaction, and authenticating the user. The authentication is required from users to authenticate themselves to the mobile phone before making a payment. The authors did not specify any authentication mechanisms to be used, but they gave examples such as graphical passwords, PIN codes, or biometric-based mechanisms. Second, the user security aspect relies on the user's awareness of any attack being launched, where the system is designed to enable the user to clearly detect any malicious acts and the user will be aware of the device that it is communicating with. Moreover, the system uses a Two-Touch payment protocol. The first touch is used to exchange identifying information to ensure both devices identify themselves to each other. Then, the second touch is used to finalize the transaction.

According to the authors, the proposed system provides a secure transaction using the secure element, a two-touch payment protocol, and an authentication mechanism. We can see this system does not prevent the relay attack since using the secure element will make a system vulnerable to the relay attack, as mentioned in section 2.1.3 [22]. Moreover, this system is not reliable because of the complexity that is being added through the number of steps needed to complete a transaction.

The second proposed protocol by Kadambi, Li, and Karp [28]. This protocol is based on the NFC-enabled mobile payment system that uses EMV payment standards. The protocol uses the secure element to issue a payment authorization token for a merchant and sends it over to the bank server to claim the payment. The aim of the protocol is to protect sensitive information from being publicly sent over the communication channel that can be tampered with. A public key infrastructure (PKI) is being deployed in the protocol, where at least a pair of keys needs to be stored in the secure element.

The authors assume their protocol provides end-to-end secure transaction with the use of payment authorization tokens to protect sensitive data over public networks. Moreover, they deployed a cryptography mechanism (i.e. PKI) in the secure element of the phone to issue these tokens. However, the authors did not provide a solution against relay attack, and the use of the secure element will make the protocol vulnerable to this type of attack. In addition, using application-level cryptography will not prevent the attack, because of the way that the attack is relaying the information [22].

The third proposed protocol was by W. Chen et al. [29]. This protocol uses the Global System for Mobile Communications (GSM) network authentication to secure a transaction. In this protocol both the user's phone and the POS terminal have to be registered to the same Mobile Operator (MO). The authors offer to use a triple authentication mechanism that uses a pre-shared key stored in the secure element (i.e. SIM card) in the phone with MO. The key is then used to generate a signed message called *S* using A3 algorithm and session key *Kc* to be used to authenticate the phone. The protocol has four steps to complete a transaction. A summary for these steps is as follows:

- The first step is the initial setup for both the phone and the POS terminal for those who wish to operate with this protocol. Both the phone and the POS terminal will receive a shared key with the MO.
- The second step is a visual price check from the user and the user agrees on the total. The total price presented by the POS terminal.
- The third step is the authentication. First, the user's phone will send the identification information to the MO through a POS to prove that the phone and the POS are running under the same MO. The MO will authenticate the phone and the POS by looking up the corresponding authentication information for both. Second, the MO will generate a random number *R* using the pre-shared key with the phone, and send it to the phone. Third, the phone needs to generate *R* with *Kc* using algorithm A8, and generate message *S* by using the pre-shared key with algorithm A3. Fourth, the phone will send back the generated *R* encrypted with *h(S)* (i.e. S_1) to verify itself to the MO. Fifth, the MO will generate S_1 in the same way and checks if the received message is a match. Finally, the MO will verify the POS by sending an encrypted message with the same pre-shared key with the POS. The message contains a session key used to secure the transaction between the phone and the POS.
- The last step is transaction execution where the phone and the POS will use the session key generated in the previous step. A request of the payment information

PI is sent from the POS to the phone. The *PI* contains a time stamp *TS* along with receipt number, total price, and a transaction counter to prevent the replay attack. Then the phone will encrypt that information and sends it back to the POS then to the MO. Next, the balance of the user's account is checked by the MO to determine a successful or failed transaction.

An improved version of this protocol presented by Saeed [30]. This optimized version offers a new security layer that requires a user's interaction. The author introduced a new step to the protocol that will be executed before the transaction execution step. That step is PIN code verification where the user needs to verify the PIN code with the secure element. Also a fresh new set of keys is being generated in this step to be used in the next step of the protocol. According to the author, this step will add random aspect to the original protocol where that will increase the level of security and resistance of the protocol toward attacks.

Despite both protocols providing a secure transaction and preventing the replay attack, both of them do not offer a protection against the relay attack. Moreover, the use of the secure element to store the pre-shared key in the phone might not be the optimal solution for the security of key storage [22]. According to W. Chen et al. and Saeed the protocol will require a level of trust from the customers to let the MO handle their sensitive information, which may raise the concern of information privacy. Also, this protocol will add more complexity to the concept of NFC-mobile phone payment (i.e. tap and pay) compared to other payment systems' simplicity.

The fifth protocol presented by Husni et al. [31]. This protocol is based on the capabilities of NFC-enabled phones to operate in different modes. A user will need to scan the tags of the products in order to buy them by using the reader mode in the NFC-enabled phone. Next, the user will provide the total amount to the POS to complete the transaction. Both the user's phone and the POS will generate a similar key using secret variables received from a third trusted party. The execution of a transaction in this protocol is divided into two types Micropayment (i.e. up to $10), and Macropayment (i.e. more than $10). The Micropayment is an offline payment mode where no third party is needed to provide the account number for a customer. On the other hand, the Macropayment will need a third party to complete a transaction in the form of providing the customer's account number. Both types will need session key agreement at the start of a transaction; these keys will be generated using the identification information for both the phone and the POS. keys are then sent to each other along with payment status, which is either accepted or rejected for verification.

According to the authors the use of a symmetric encryption mechanism will prevent a number of attacks. However as mentioned before, the application-level cryptography will not prevent the relay attack since it only used to send data from the POS to the phone without altering it [22].

2.2.2 Relay Attack Counter Measures

Since, the relay attack cannot be prevented using an application-level encryption, a number of counter measures have been proposed in the NFC-mobile payment system to overcome the relay attack. In this section the thesis will analyze the most highlighted solutions by a number of publications.

2.2.2.1 Location Detection Based Service

Finding the location information of a phone is an approach that many researchers have taken an interest in. The protocols introduced in this approach tend to use the resources of the phone to collect accurate location information of it, and then verify that phone by matching the location information with the POS. A location can be collected by a number of technologies such as the global positioning system (GPS), the use of GSM network, or by using sounds and light sensors for both the phone and the POS reader.

A security protocol was presented by Francis et al. [32] which was based on using divided graphical access zones for a number of services. Then, based on the location of the phone the MO will grant the user an access ticket to that zone service. For example a mall having a number of shops will use the same zone for their services, and an access to those services is given based on the location of the user's phone. If a user wants to make a payment inside those zones, the phone needs to authenticate its location with the mobile network. Then the MO will send a privileged access ticket for that absolute zone to the phone. That ticket includes phone ID, time stamp, random number, and privileged access to the zone. According to the authors using the random number will prevent the replay attack, and using the authentication mechanism will prevent the long-range relay attack (i.e. when the victim is out of the zone). Moreover, to overcome the short-range version of the attack they propose a chain of trust. In that, the mobile device will prove their location and proximity of the POS with other near devices. This chain needs the notify users to any malicious activities from other devices.

Thus, this protocol can prevent the long-range and short-range relay attack, still it needs the resources to provide the accurate location of the phone and the involvement of the users to make a secure transaction. This will add more complexity to the payment system, and raise concerns about the users' location privacy.

Another protocol proposed by Halevi et al. [33] uses ambient sensor data such as data collected using light and sound sensors. The aim of this protocol is to use the collected data to detect the location information of both the POS and the phone and verify the proximity of the phone to the POS. The collected ambient data from a phone need to match the data collected from a POS at the time of the transaction by the bank server. The authors specify a shared key between the bank and the phone to encrypt the location information, to prevent a rogue reader from tampering with the information. The authors tested this protocol by capturing the ambient data of different environments (e.g. restaurant, library, coffee shop, and supermarket), and they found that the results of most environments are not similar.

According to the authors, their approach manages to prevent the relay attack and they detected the inconsistency of collected data during the attack. Moreover, this approach does not violate the privacy of a user's location, since the used data does not illustrate a specific place. However, the attack is possible in a scenario where the victim is in a place similar to another place that the attacker wants to make a payment in. In this case the chance of both the victim's phone and the targeted POS generating a similar ambient data is very high. Moreover, an attacker trying to generate the same ambient data around a victim is possible.

2.2.2.2 Transaction Time Bounding

This counter measure is basically restricting the transaction time needed to successfully complete it. By this restriction the relay attack will be prevented because of the added delay in a transaction during the attack. However, this counter measure is not applicable to the ISO 14443 standards. In the specification of the ISO 14443 standards, the frame waiting time FWT can be set up to 4.95 seconds. In the attack scenario, an attacker could request a FWT of up to 4.9 seconds, which means the attacker will have plenty of time to perform the relay attack. Moreover, restricting the time of a transaction will eliminate the use of cloud-based payment applications using NFC-enabled phones, and that is due to the added time from communicating with the application cloud [22] [34] [35].

2.2.2.3 Other Counter Measures

A number of other counter measures were introduced to either prevent the attack or increase the attack time to exceed the time limit (i.e. 4.95 seconds). A number of publications suggested the use of a Faraday Cage (e.g. aluminum foil) to cover the smart card or the phone, so they will not be readable by a rogue reader. This suggestion might not be the optimal solution, where a user may forget to encase the phone or the card with the Faraday Cage [21][22] [36]. Moreover, adding a PIN code to the application and the card emulated by the phone is intended to verify the ownership of a phone and card; however an attacker might be able to listen in on the transaction from the victim's phone and then get the PIN code [22]. Last, disabling internal mode communication for the payment applets is a counter measure that will not fully prevent the attack. Most likely the software version of the attack will be prevented, but not the physical version of it [22].

2.3 Summary

In this chapter, we discussed the background of contactless payment and mobile payment along with development of mobile payment using NFC-enabled phones. We also took a closer look into the security threats on NFC-enabled mobile payments and NFC technology in general. Finally, we presented some of the proposed payment protocols and analyzed each protocol's advantages and disadvantages. We also presented the most recognized counter measures to prevent the relay attack and analyzed their reliability. In the next chapter, we will present the motivation and the objectives of the proposed protocol in this thesis.

Chapter 3 Research Focus

In this chapter we will discuss the motivation of this thesis, where we will explore the factors needed to develop a payment system. Then, we explain the objectives of this thesis to overcome and achieve the goals of the NFC-enabled phone payment systems.

3.1 Motivation

In the proximity payment system there are at least two entities (i.e. card, and reader). The two entities complete the payment system, where a transaction is done in a secure and a simple way. The NFC-enabled mobile payment system is the most recent development in payment systems. In this system a reader and an emulated card with the phone are exchanging data through a wireless communication channel (i.e. NFC wireless communication technology). Moreover, NFC is a sub technology of RFID and based on ISO 14443 standards, which means this payment system faces a number of security threats inherited from RFID and ISO 14443. One of the most explored security threats is a relay attack (i.e. ghost and leech attack). This attack is robust to the application-level cryptography solutions, and this is because of the attack is transferring (i.e. relaying) information between the two entities (i.e. legitimate POS, and legitimate phone) without altering the transferred information. Both entities have no knowledge that they are communicating with each other [22]. Thus, counter measures were proposed as solutions (i.e. discussed in section 2.2.1.2) to prevent this attack.

As a result of analyzing these counter measures we found that location privacy, reliability, and applicability are their main issues. Therefore, this thesis aims to propose a new payment system that uses NFC-enabled phones to prevent relay attacks and overcome the issues discussed above. The proposed system is not suitable for the RFID systems; because of the RFID cannot be a reader and a contactless card at the same time. In addition, contactless technology is not compatible with the proposed system because it cannot read a NFC-tag.

3.2 Objective

Motivated by the vulnerability of current payment system using NFC-enabled phones and the proposed NFC-mobile payment systems by other publications (discussed in section 2.2.1.1). The objective of this thesis is to propose a robust system against the related security threats and especially a relay attack. Moreover, the proposed system will overcome the limitations in the proposed counter measures (i.e. location privacy, reliability, and applicability). The proposed system will focus on:

- Detecting the location of a customer by ensuring that the customer is in a proximity to the POS in order to make a payment without the need for actual location information (i.e. collected by GPS, or GSM).
- Authenticating the POS reader to the phone to prevent a rogue reader from tampering with the communication.
- A dynamic symmetric key pre-shared for the first time only between the phone and the bank server.
- Only the phone and the bank server generate a hashed dynamic message used as signature to authenticate each other.
- A cloud-based payment application to securely store the information needed to complete a transaction.

3.3 Summary

This chapter presented the motivation of this thesis, which is the lack of security in preventing the relay attack in other proposed payment systems and current payment systems using NFC technology. We presented also, the objective of this thesis where a new NFC-mobile payment system is proposed. This system will overcome the security threats of other NFC-mobile payment systems and the limitations of the counter measures. The next chapter will present the framework of the proposed NFC-mobile payment system.

Chapter 4 Proposed Framework

In this chapter we will present the framework for the proposed NFC-enabled mobile payment system. The novelty of this system is authenticating the POS reader to the phone using a new location detection mechanism. Moreover, a new approach of encryption will be used to securely transfer sensitive data from the phone to the bank server through the POS reader. The data is encrypted using a dynamic symmetric key, which is shared at the application setup phase for one time only. Finally, a transaction is completed if the bank server successfully generates a hashed message using the shared key and matches it with the message generated in the same way at the phone side.

4.1 Assumptions

The following assumptions are needed for the proposed system:

- The communication between the phone and the application server is secure enough to transfer data between the phone and the application server.
- The application server has the resources needed to perform the encryption mechanism used.
- The sensitive data and the shared key are stored securely in the application server.
- The application server and the bank server communication channel is secure to transfer data between them, also to share the key for the first time.
- The POS reader and the bank server communication channel transfer data securely between them.

4.2 Proposed System Architecture

As mentioned previously the NFC-mobile payment system will consist of a phone to emulate the card, a POS reader, and a bank server. In the proposed system we introduce a new entity to the architecture of the NFC-mobile payment system, which is an NFC tag to be used in the location detection mechanism. A cloud-based application server is used to manage the computations of the encryption process and to store sensitive data securely. Figure 4.1 describes the architecture of the proposed system.

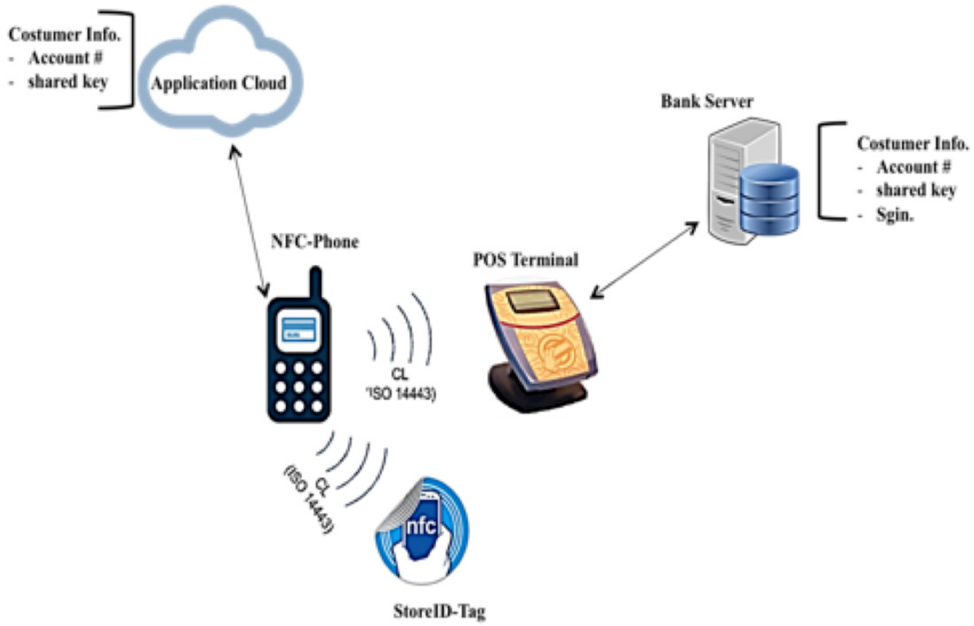


Figure 4.1 the architecture of the proposed system.

The bank server will store information for each customer's phone in its database; that information is represented as follow:

Act#: is the user account number that is given randomly by the bank server to the customers.

K_i: is the shared key, generated randomly for the first time at the bank server then shared with the application server.

Sign.: is a hashed message being generated by hashing the encrypted account number with K_i.

The application cloud will store the same information as the bank server. Moreover, the K_i is updated every transaction where both servers (i.e. the bank server, and the application server) will update the key at their sides without communicating with each other. The new cryptography approach provides this feature in the encryption or decryption process. The update process uses the encrypted message to generate the new key. In the same way, the sign. message is updated every transaction.

The POS reader will generate a random hashed message called *StoreID*, that will be stored in both the POS reader as well as the NFC tag.

4.3 Proposed System Phases

In order for the proposed system to function, three phases need to be followed. We assume the setup phase is already finished and the initial key was shared along with the account number provided by the bank server to the user.

4.3.1 Phase One

The first phase is executed by the store side. The workflow of this phase is as follows:

1- Generating the hashed *StoreID* at the POS reader.

2- Writing the hashed message into the NFC tag.

The POS reader needs to generate the hashed *StoreID* by using a randomly generated number and the reader ID. The random number is 32-bytes long, where it is generated from a set of 32 characters. Each character is randomly chosen from a pool that contains the lower case of letters from *a* to *f* and the numbers from *0* to *9*. This process generates the random number and it is unique for each store. The total number of possible random numbers is 32^{16}, which makes it sufficient to generate a unique number. Then, the *StoreID* will be generated using the logical exclusive-OR operation XOR with the random number and the Reader ID. The result of that is used in a hash function to generate the hashed message. The hash function is used to strengthen the message and makes it difficult to tamper with or generate the same message. Table 4.1 shows the generating process of the hashed *StoreID*. The generating of the hashed message is represented by the following equation:

$$StoreID = h\ (rand.\oplus ReaderID)$$

`Input:`	`String Pool = {0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f};`
	`ReaderID;`

`Output:`	`StoreID;`

`Process:`	`For = 1 to 32{`
	`Choose a random character form Pool, Store it in Rand};`
	`Generate message = Rand XOR ReaderID;`
	`StoreID = Hash Function (messgae);`

Table 4.1 shows the *StoreID* generating process.

In the second step, the POS reader is an NFC-enabled device that is able to write the generated *StoreID* in the NFC tag and store the same message inside the internal memory. The NFC tag can then be placed next to the POS reader. This phase is executed once every day by the POS reader, which will keep the randomness factor of the message fresh and make it difficult to tamper with. Figure 4.2 shows the steps of the First Phase.

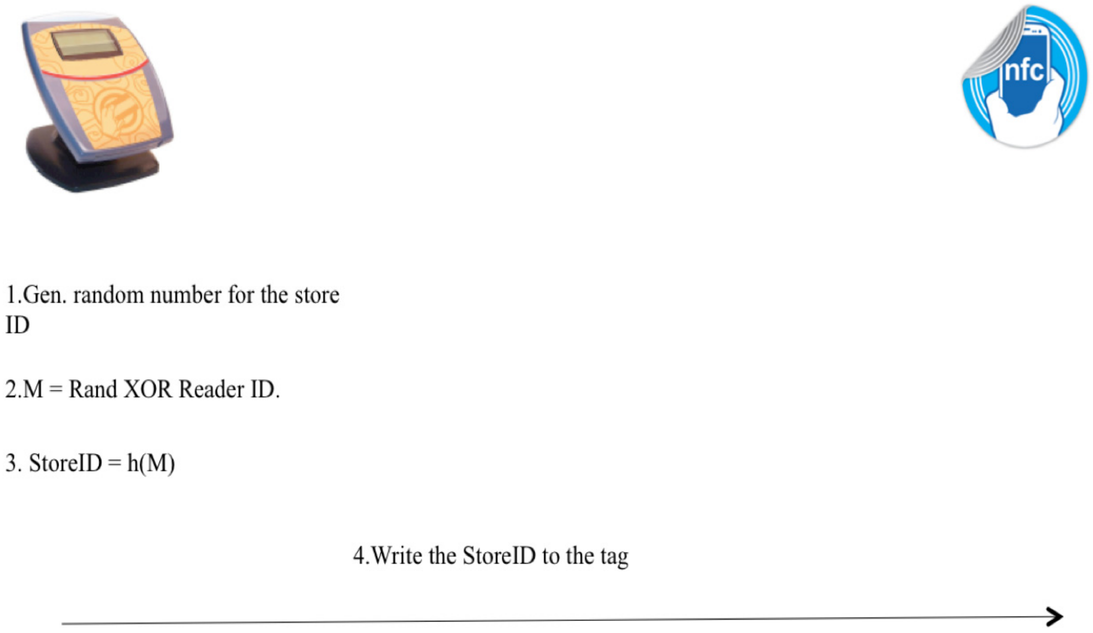

nfc
1.Gen. random number for the store ID
2.M = Rand XOR Reader ID.
3. StoreID = h(M)
4.Write the StoreID to the tag

Figure 4.2 Phase one process.

4.3.2 Phase Two

The second phase needs user interaction. As we mentioned previously the NFC-enabled phone can operate in three modes, and one of them is the reader mode where a phone can read the content of an NFC tag and process it. Therefore, this phase requires the user to scan the NFC tag (i.e. placed next to the POS reader) in order to collect the *StoreID*, and store it securely on the phone. The importance of this phase is to verify the proximity of the phone to the reader. Later in Phase Three this message will be used to match it with the same message stored in the POS reader memory. As simple as it seems, the function of this phase is crucial to the process of preventing the relay attack. Figure 4.3, shows the steps needed for Phase Two.

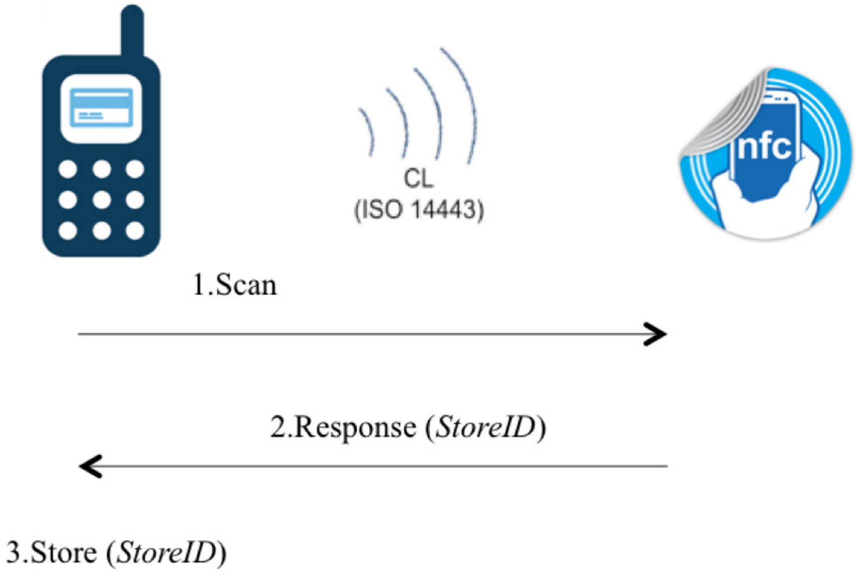


Figure 4.3 Phase Two process.

4.3.3 Phase Three

After the first two phases, a customer needs to tap the phone to the POS reader. This phase consists of three steps, which are as follows:

1- Authenticating the POS reader to the phone and verifying the proximity of the phone.

2- Encrypting the *ACT#* and a new K_i will be generated.

3- Authenticating the phone to the bank server and a new K_i will be generated.

In communication between the NFC-enabled phone and the reader, messages called APDU (i.e. Application Protocol Data Unit) commands will be exchanged. The APDU command is a byte array message that is used to transfer the information. At the start of communication between a phone and a reader in the NFC-mobile payment system a select command will be sent from the POS to the phone application. This APDU command called PPSE (Proximity Payment Service Environment) select command. This command is used to request the phone applet to send a list of the available payment applications. Then, an application is selected by its ID. That is the Application ID (AID), an AID consists of up to 16 bytes. A phone application might support a number of payment services by having a set of AIDs (e.g. MasterCard AID, Visa AID, and Gift Card AID). The AID is reserved for each payment card issuer and only used for that payment service. Figure 4.4 shows the overall process of this phase.

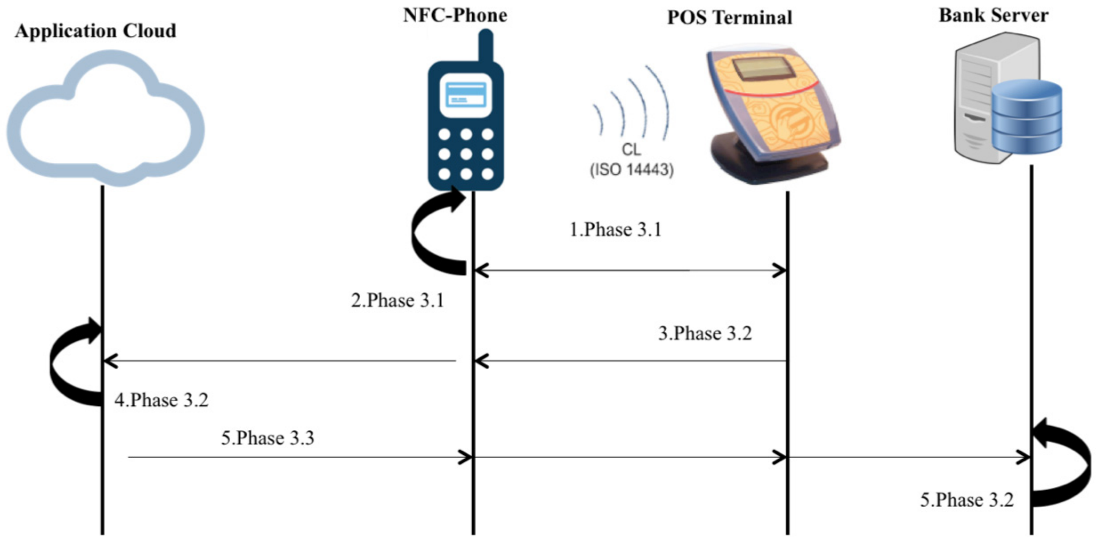


Figure 4.4 shows the overall process of Phase 3.

4.3.3.1 Phase 3.1

After selecting the correct AID, a second APDU will be sent by the POS reader to the phone containing the *StoreID* stored in the reader from the first phase. A phone will match both messages received from the POS reader and the scanned message from the NFC tag in phase two. At this point we have three scenarios, which are based on the result of a message matching process. Figure 4.5 shows the flowchart of the phase 3.1 process. The three scenarios are as follows:

- If the messages match, the phone will ensure the POS reader has generated both messages. Moreover, matching the messages will ensure that the phone is in close range to the POS reader because the phone needs to get a copy of the message from the NFC tag (i.e. located next to the POS reader).
- If the messages do not match, in this case a phone will respond with a fail of execution response to the POS reader. As a result, the phone will reject any further commands and deactivate the service (i.e. Host Card Emulation service). This scenario may occur if the stored value does not match with the received one; that might be another *StoreID* or an attacker tampered with the *StoreID* stored in the NFC-tag.
- A phone will reject further commands and deactivate the service, if the message does not exist at the phone side. Which means that the user did not scan the NFC tag, or the phone is not in proximity range of the POS reader. In the first case the cashier will ask the user to scan the tag first in order for the transaction to go through. Otherwise, it is the second case, where there is a suspicion of a relay attack being launched on the victim's device. Either way, the phone has a mechanism as explained to prevent the transaction from being completed.

In summary, this step is very important to resist the relay attack. The reader was able to authenticate itself to the phone by sending the generated *StoreID*. The phone also, was able to prevent the attack by matching the messages and act based on the result of the matching process.

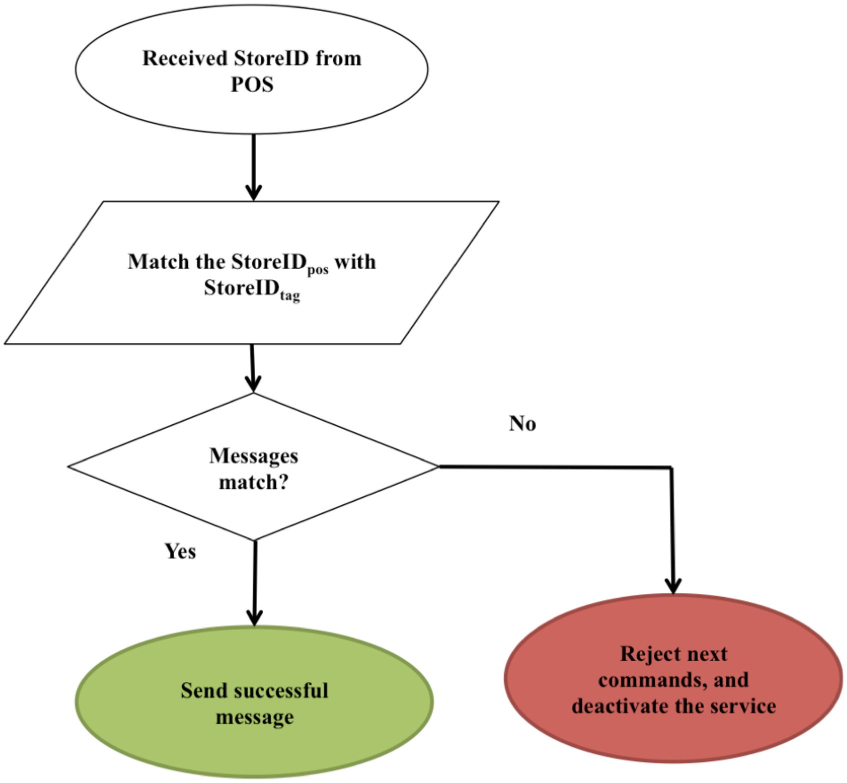

Received StoreID from POS
Match the StoreID_{pos} with StoreID_{tag}
Messages match?
No
Yes
Send successful message
Reject next commands, and deactivate the service

Figure 4.5 shows the flowchart of the phase 3.1 workflow.

4.3.3.2 Phase 3.2

If the message matching result is successful (in phase 3.1), a third APDU will be sent after receiving the successful matching response from the phone. The third APDU is the get processing command, where the POS reader will request the phone to provide *ACT#* in order to complete the transaction. The phone, after receiving the command will send the request to the application cloud server.

The cloud application server will encrypt the *ACT#* information and generate the new encryption key K_i during the encryption process, then store it to be used in the next transaction. Next, the application's cloud server will generate the *sign.* message by hashing the encrypted *ACT#* message. Finally, the application's cloud server will send the encrypted *ACT#* concatenated with *sign.* message to the POS reader through the phone's application.

4.3.3.3 Phase 3.3

Lastly, the POS reader will send the payment information along with the message received from the phone to the customer's bank server. The bank server, in its role, will authenticate the phone by matching *sign.* message with its database. In this phase the bank server first needs to authenticate the phone by matching the *sign.* message generated from the phone. The authentication has two scenarios, which are explained as follows:

- If the *sign.* messages match, the application server will retrieve the corresponding K_i to decrypt the encrypted message to get the *ACT#.* Then, the bank server will match the *ACT#* with corresponding *ACT#* in the database to verify the integrity of the encrypted message. The *ACT#* matching process also has two cases, are explained as follows:
    - If the *ACT#* matches, the bank server will approve the transaction and keep a record of the transaction in it database. Therefore, a conformation message is sent to the POS reader and then to the phone to complete the transaction.
    - If the *ACT#* does not match, the bank server will reject the transaction and send a transaction denied message to the POS reader then to the phone.
- Going back, if the *sign.* message from the phone does not match with the bank server database, the transaction will be rejected and a message will be sent to the

POS reader and the phone to inform them of the transaction rejection. A flowchart is presented in figure 4.6 to show the process of this phase.

In summary, the three phases illustrate the process of the proposed payment system's framework. We have showed how the system is resistant to the relay attack. We also provided a novel solution to prevent it by ensuring the proximity of a phone to a POS reader. In the next section we will discuss the new cryptography approach and the key generation part.

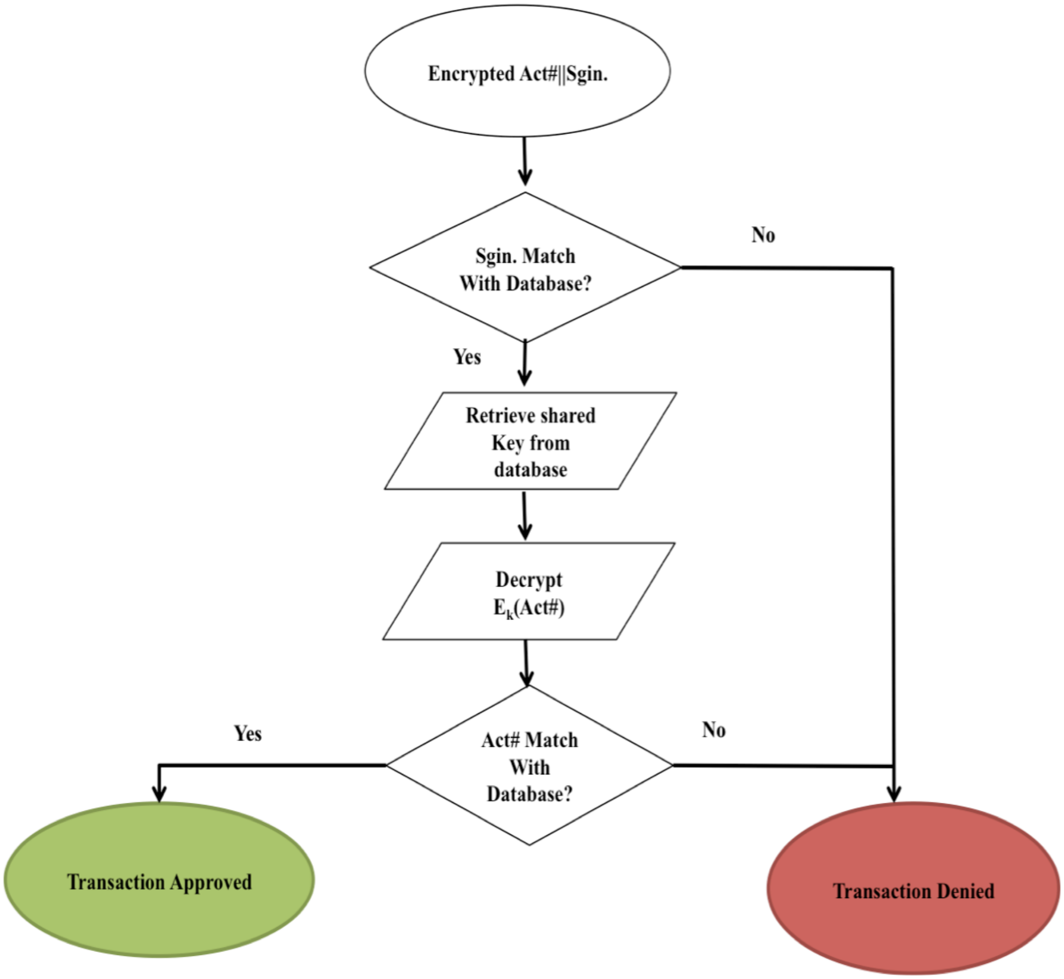


Figure 4.6 flowchart of phase 3.3 process.

4.3 The Encryption Mechanism & Key Generation

The new algorithm is a hybrid cryptographic approach that encompasses both stream cipher and block cipher features. The algorithm consists of two parts, the key generation that inherits the stream cipher features and two rounds of encryption or decryption based on block cipher cryptography. This algorithm uses XOR operation for encryption or decryption, and in addition to the XOR operation an inversion operation is used in the key generation part. Mainly, the algorithm divides the message as well as the key into a number of blocks, each one is 128-bit long and each block into chunks, and each chunk is 8-bit. By that, the algorithm encrypts or decrypts the message divided chunks with key chunks by using the operations mentioned above. The key generation part is processed during the encryption or decryption by driving half chunks of the key from the cipher text at the end of each round. Then the final key is to be used in the next communication. By concatenating both halves, the new key is derived at the end of both rounds of encryption or decryption [37].

This algorithm is suitable for systems such as RFID, and NFC (i.e. resource constrained systems), because the use of basic operations such as XOR do not require expensive computation. Moreover, this algorithm offers a pre-shared, dynamic, and symmetric key that is required by our proposed system to offer more security for the transferring of data between a phone and a bank server.

4.4 Summary

In this chapter, we presented the framework of the proposed payment system using NFC-enabled phones. Moreover, we presented the process of preventing relay attack by using a new solution that uses a hashed message generated by the reader in order to authenticate itself to the phone and write it that message inside an NFC tag. The phone on the reader mode (i.e. one of the three modes that NFC phones can operate on) will scan the NFC tag to verify its proximity to the reader. The NFC tag is updated by the reader every 24 hours and placed next to a POS reader. In addition, we used a new cryptography approach that uses the symmetric key concept. This approach will update the keys after each transaction. In the next chapter, we will discuss the implementation and the evaluation methodology of the proposed system.

Chapter 5 Implementation and Evaluation Methodology

In this chapter, we will discuss the implementation of the proposed payment system and demonstrate its feasibility. Moreover, we will demonstrate launched the relay attack that we launched on a number of payment systems, which are the contactless card payment system, the NFC-enabled phone payment system, and the proposed payment system. These payment systems are based on NFC technology. Next, we will evaluate the experimental results. Finally, we will present the evaluation methodology of the proposed system.

5.1 Implementation of the proposed system

In this section, we present the proof of concept implementation of the proposed system. First, we present the experimental tools that we used. After that, we present the experimental model of the proposed system and show the three phases needed to complete a transaction.

5.1.1 Experimental Tools

In this section we will present the tools were used in the implementation of the proposed system:

1. **NFC-enabled phone**

We used an NFC-enabled phone to emulate the card. The phone we used is LG Nexus 4 with Android 4.4.2 operating system, Snapdragon S4 Pro quad-core 1.5GHz processor, and internal memory storage of 16 GB [38]. We installed an Android application on the phone to emulate a smart card.

2. NFC reader

We used an NFC reader to act as POS. The NFC reader model is ACR122U that supports both ISO 14443, and ISO/IEC18092 NFC standards. This reader can be linked to a PC and work with developed applications that uses the NFC technology either as reader or card emulator [39]. We used this reader as a POS reader with a developed desktop application to be used on the proposed system.

3. Eclipse IDE

Eclipse is a java development tool used to develop java applications. The eclipse tool uses the plugins concept to provide more functionality and flexibility to develop an application [40]. One of the plugins is Android SDK Tools. This plugin is used to develop java applications that can be installed on phones run on Android operating system [41]. We used this tool to develop an Android application to emulate the card and scan NFC tags.

4. NetBeans IDE

NetBeans is a development tool for a number of programing languages such as Java, HTML, and C/C++ [41]. We used this tool because of its compatibility with Mac OSX laptops. With this tool, we developed a desktop application for the POS reader that is used in the proposed system.

5. Mac OSX laptop

We used a Mac laptop to develop the POS reader application. This laptop is running on OSX version 10.9.4 operating system, 2.7 GHz Intel Core i7 processor, and a RAM memory of 8 GB [42]. We used this laptop as a platform to develop and install the desktop application for the POS reader.

5.1.2 Experimental model

In this section we describe the proof of concept implementation process using an NFC-enabled phone with a developed Android application installed on the phone to emulate the card and read the NFC tags. We also used an NFC reader with a developed java application to emulate a POS terminal. The implementation demonstrates the execution of the three phases we described in the last chapter. Moreover, we will describe the launching of the relay attack on the previous mentioned systems.

5.1.2.1 Phone Application and Desktop Application Interfaces

First, at the user side, we installed an android application on the phone. Figure 5.1 shows the application installed on the user's phone. The interface has two buttons, one for accessing the tag-scanning screen and another for showing the card-emulating screen. Figure 5.2 shows the interface of the phone application. The user will press the scan button, and bring the phone in close range to the NFC tag. Then, the application will take the user to a new screen that shows information collected from the tag. Figure 5.3 shows the screen that displays the collected information from the NFC tag. The second button is used to access the card emulation screen where the user needs to tap the phone to POS reader to start the transaction (see figure 5.4).

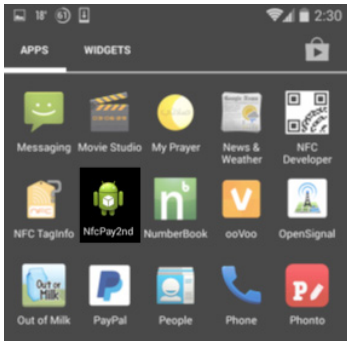
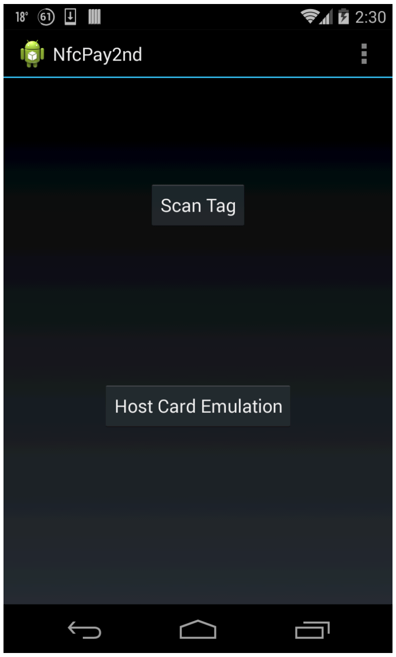


Figure 5.1 The Phone application installed in Nexus 4.

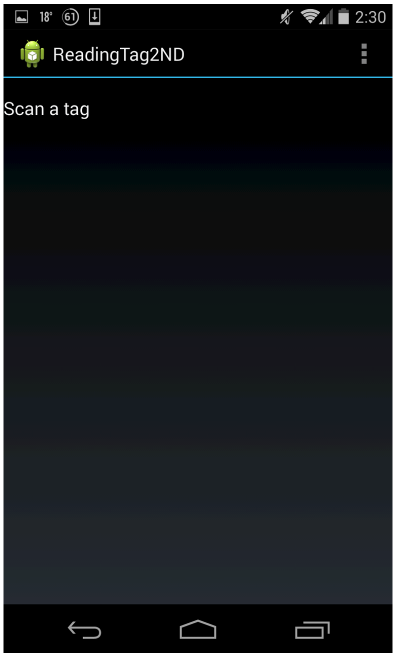


NfcPay2nd

Scan Tag

Host Card Emulation

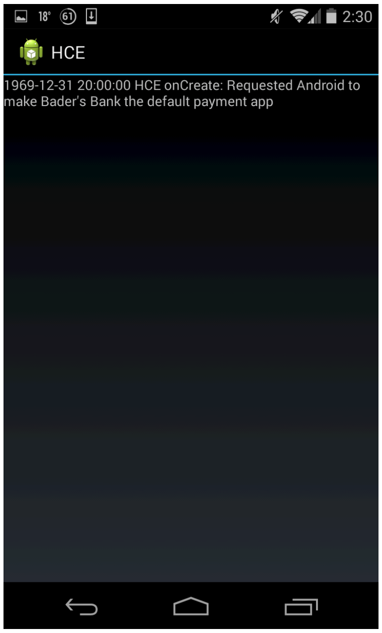
Figure 5.2 the phone application interface screen.



ReadingTag2ND

Scan a tag

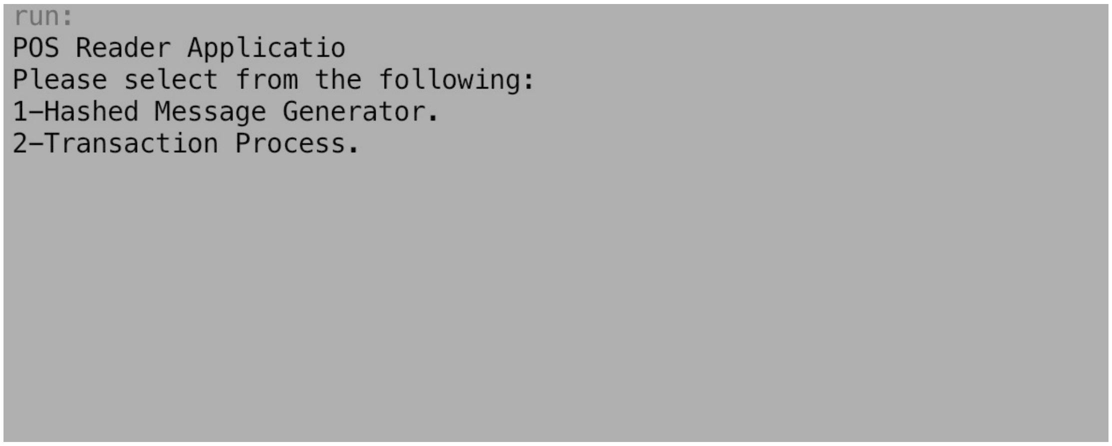
Figure 5.3 scanning-tag screen in the phone application.



1969-12-31 20:00:00 HCE onCreate: Requested Android to make Bader's Bank the default payment app

Figure 5.4 the card emulation screen in the phone application.

The application's interface of the POS reader consists of two parts. Figure 5.5 shows the POS interface in the desktop application. The first part is the *StoreID* generator, which storeowner needs to execute the option of generating *StoreID* once every 24 hours. The second option is the transaction process, which is selected when a customer wants to make a payment for a product or a service.

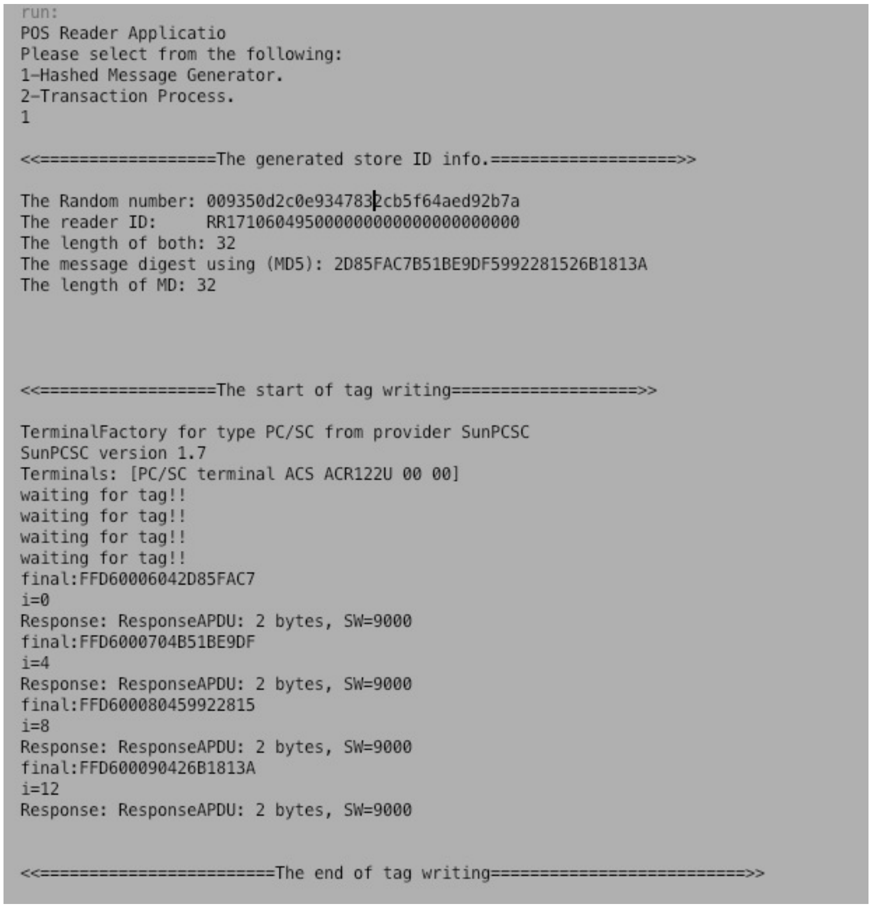


```
run:
POS Reader Applicatio
Please select from the following:
1-Hashed Message Generator.
2-Transaction Process.
```

Figure 5.5 the desktop application interface snapshot.

5.1.2.2 Phase 1 Implementation

As presented in Chapter Four, this phase is executed at the POS reader. This phase is used to generate a *StoreID* message and write it on the NFC tag. In the desktop application interface we need to select the first option, as shown in figure 5.5 (i.e. desktop application interface). After selecting the option, the first part of the screen is showing the *StoreID* generating process. Figure 5.6 shows the process of generating *StoreID.* On the second part of the screen, the application will ask to tap the NFC tag in order to write the *StoreID* on the tag through the POS reader. Figure 5.6 shows the process of writing on the tag. At this point we generated and wrote the *StoreID* in the NFC tag.

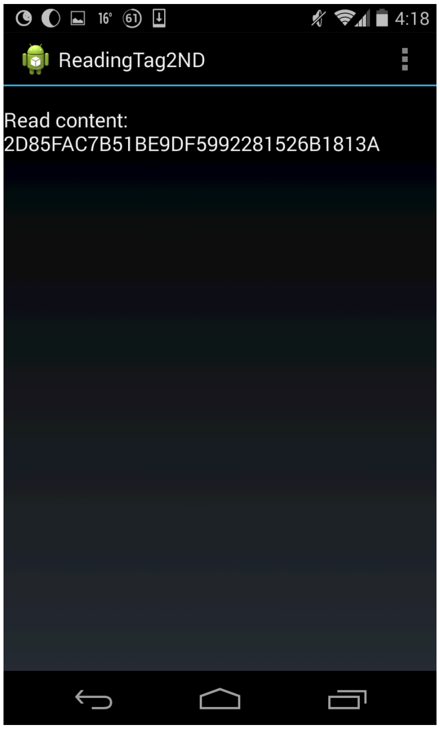


```
run:
POS Reader Applicatio
Please select from the following:
1-Hashed Message Generator.
2-Transaction Process.
1

<<==================The generated store ID info.===================>>

The Random number: 009350d2c0e9347832cb5f64aed92b7a
The reader ID:     RR171060495000000000000000000000
The length of both: 32
The message digest using (MD5): 2D85FAC7B51BE9DF5992281526B1813A
The length of MD: 32



<<==================The start of tag writing===================>>

TerminalFactory for type PC/SC from provider SunPCSC
SunPCSC version 1.7
Terminals: [PC/SC terminal ACS ACR122U 00 00]
waiting for tag!!
waiting for tag!!
waiting for tag!!
waiting for tag!!
final:FFD60006042D85FAC7
i=0
Response: ResponseAPDU: 2 bytes, SW=9000
final:FFD6000704B51BE9DF
i=4
Response: ResponseAPDU: 2 bytes, SW=9000
final:FFD600080459922815
i=8
Response: ResponseAPDU: 2 bytes, SW=9000
final:FFD600090426B1813A
i=12
Response: ResponseAPDU: 2 bytes, SW=9000


<<========================The end of tag writing=========================>>
```

Figure 5.6 Shows the StoreID generating process at desktop application.

5.1.2.3 Phase 2 Implementation

This phase is used to ask the customer to scan the NFC tag and collect the *StoreID* by operating the phone in the reader mode. The interface of the application, as mentioned previously, has two buttons. After pressing the scan tag button the scan screen will request the user to scan the NFC tag. By placing the NFC tag in close range (i.e. up to 4cm) to the phone, the application was able to collect the information needed (i.e. *StoreID*). Figure 5.7 shows the collected *StoreID* from the tag.



```
Read content:
2D85FAC7B51BE9DF5992281526B1813A
```

Figure 5.7 shows the collected StoreID from the tag at phone application.

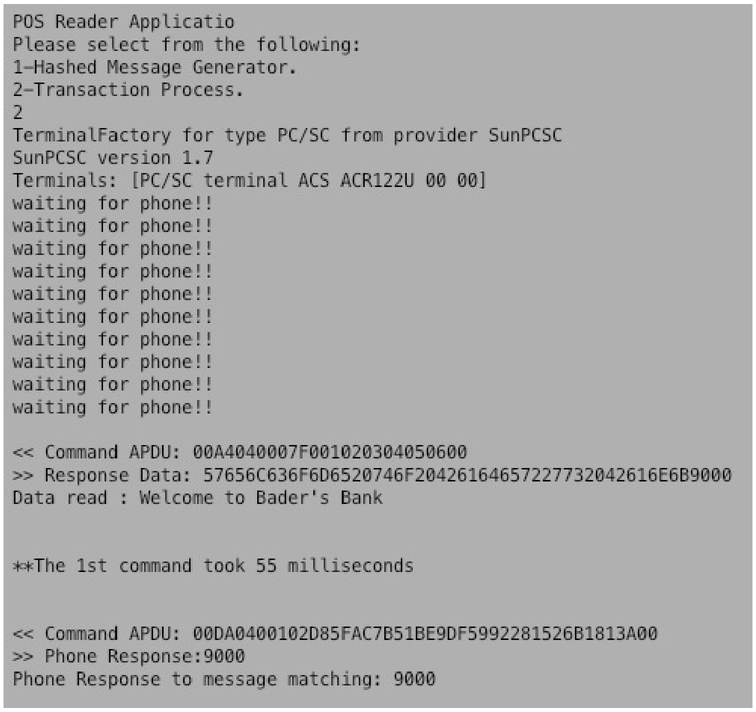
5.1.2.4 Phase 3 Implementation

As we illustrated in Chapter Four, this phase has three sub phases. Therefore, the implementation of this phase will be described in three sections. This phase require the user to tap the phone to the POS reader in order to start the transaction process. First, the POS reader will send the select APDU command to the phone, then the phone responds to that command with a list of available payment services. The list contains the AID's of the payment application supported by the phone's application. In this implementation we used a fake AID (i.e. *F0010203040506)* for our payment application.

Next, the POS reader sends a second command that contains the generated *StoreID* to the phone. Then, the phone authenticates the POS reader by matching the *StoreID* message with the collected *StoreID* from the NFC tag (in the previous phase). The phone responded with a successful execution message (i.e. 9000), which means the message matched.

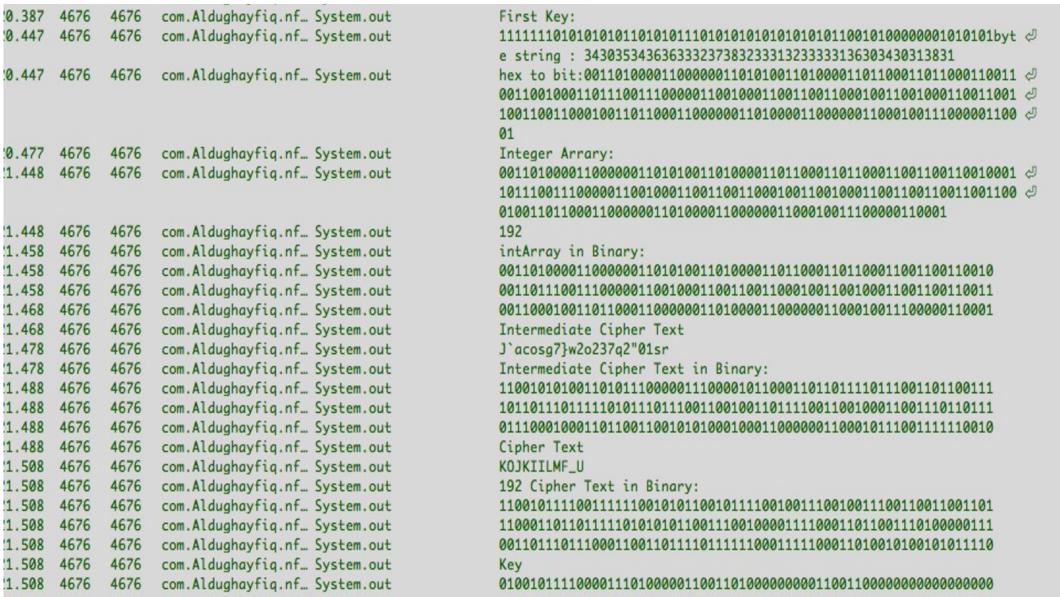
The POS sends a third command, which is the get processing command, to request the ACT# from the phone. Then, the phone sends this request to the application server. However, in our implementation this step is processed at the phone's application. Thus, The phone's application encrypts the *ACT#* and generate the new key as well as the *sign.* message. Then, it sends the encrypted message concatenated with *sign.* as a response to the POS reader. The POS sends that response to the bank server. In this step, we also used the bank server as a part of the desktop application and we used a Microsoft Office Excel sheet to act as the bank's database. We stored the needed information of the customer in the Excel sheet. The desktop application matches *sign.* message with the stored *sign.* messages in the Excel file to authenticate the phone. After, matching the *sign.* message, the desktop application by using the corresponding key K_i decrypts the encrypted message and match the *ACT#* with the database to ensure the integrity of the encrypted message. After the *ACT#* matches, the desktop application sends a message to the POS reader contains the approval of the transaction. Subsequently, an acknowledgment command is sent to the phone with the approval of the transaction.

Following are snapshots of the interface of the desktop application and the phone application showing the process of this phase (see figures 5.8,5.9,5.10,and 5.11):



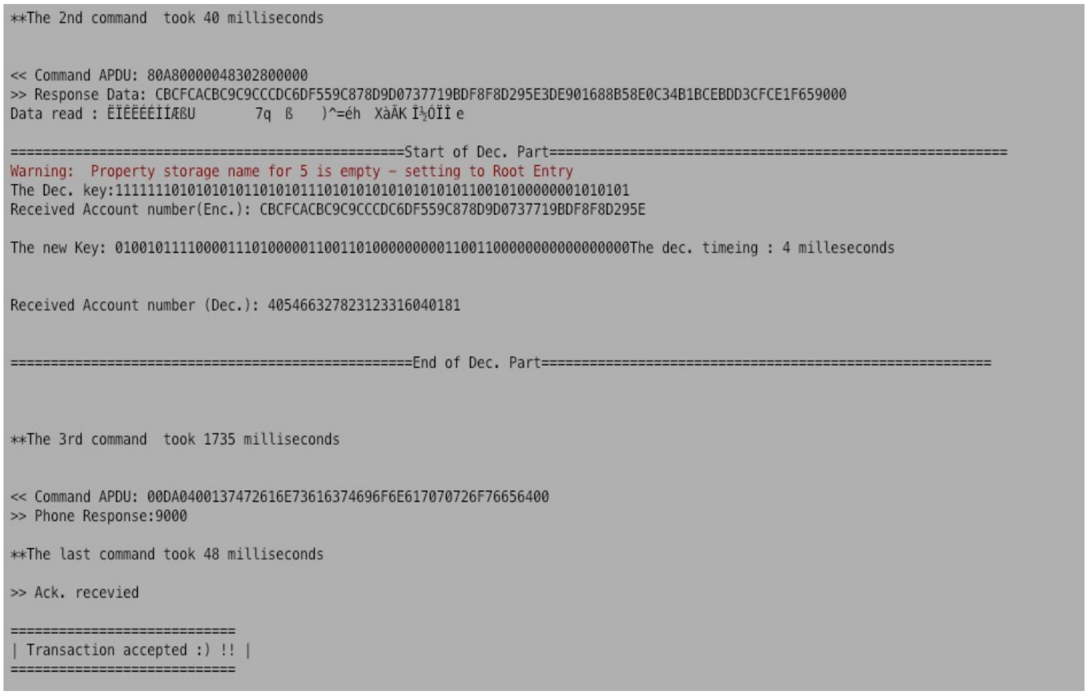
```
POS Reader Applicatio
Please select from the following:
1-Hashed Message Generator.
2-Transaction Process.
2
TerminalFactory for type PC/SC from provider SunPCSC
SunPCSC version 1.7
Terminals: [PC/SC terminal ACS ACR122U 00 00]
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!

<< Command APDU: 00A4040007F001020304050600
>> Response Data: 57656C636F6D6520746F204261646572277320426616E6B9000
Data read : Welcome to Bader's Bank


**The 1st command took 55 milliseconds


<< Command APDU: 00DA0400102D85FAC7B51BE9DF5992281526B1813A00
>> Phone Response:9000
Phone Response to message matching: 9000
```

Figure 5.8 shows the message matching response at desktop application (Phase3.1).



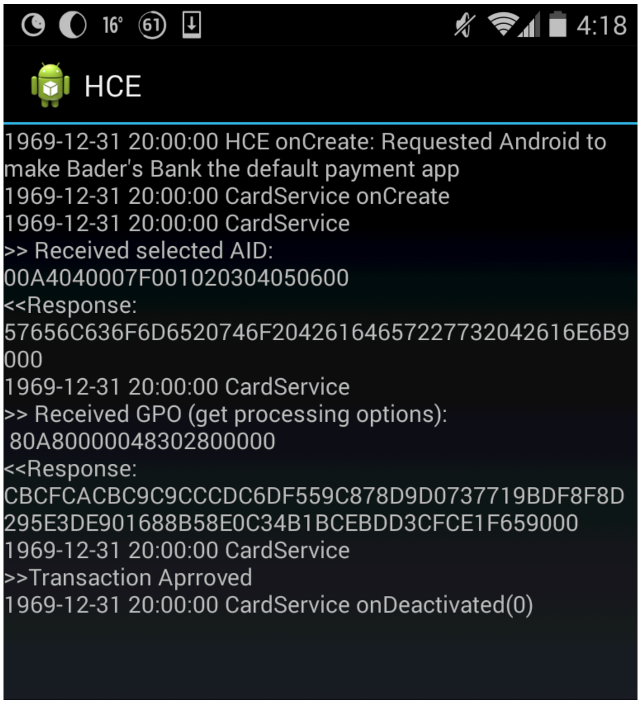
```
0.387  4676  4676  com.Aldughayfiq.nf… System.out      First Key:
0.447  4676  4676  com.Aldughayfiq.nf… System.out      1111111010101010110101011101010101010101010110010100000001010101byt ⏎
                                                       e string : 3430353436363332373832333132333331363034303133831
0.447  4676  4676  com.Aldughayfiq.nf… System.out      hex to bit:0011010000110000001101010011010000110110001101100011001 ⏎
                                                       0011001000110111001110000001100100011001100110001001100100011001100 ⏎
                                                       1001100110001001101100011000000110100001100000011000100111000001100 ⏎
                                                       01
0.477  4676  4676  com.Aldughayfiq.nf… System.out      Integer Arrary:
1.448  4676  4676  com.Aldughayfiq.nf… System.out      0011010000110000001101010011010000110110001101100011001100110010001 ⏎
                                                       1011100111000000110010001100110011000100110010001100110011001100 ⏎
                                                       0100110110001100000011010000110000001100010011100000110001
1.448  4676  4676  com.Aldughayfiq.nf… System.out      192
1.458  4676  4676  com.Aldughayfiq.nf… System.out      intArray in Binary:
1.458  4676  4676  com.Aldughayfiq.nf… System.out      001101000011000000110101001101000011011000110110001100110010
1.458  4676  4676  com.Aldughayfiq.nf… System.out      001101110011100000110010001100110011000100110010001100110011
1.468  4676  4676  com.Aldughayfiq.nf… System.out      001100010011011000110000001101000011000000110001001110000011000 1
1.468  4676  4676  com.Aldughayfiq.nf… System.out      Intermediate Cipher Text
1.478  4676  4676  com.Aldughayfiq.nf… System.out      J`acosg7}w2o237q2"01sr
1.478  4676  4676  com.Aldughayfiq.nf… System.out      Intermediate Cipher Text in Binary:
1.488  4676  4676  com.Aldughayfiq.nf… System.out      1100101010011010111000001110000101100011011011110111001101100111
1.488  4676  4676  com.Aldughayfiq.nf… System.out      1011011101111101011101110011001001101111001100100011001110110111
1.488  4676  4676  com.Aldughayfiq.nf… System.out      0111000100011011001100101010001000110000001100010111001111110010
1.488  4676  4676  com.Aldughayfiq.nf… System.out      Cipher Text
1.508  4676  4676  com.Aldughayfiq.nf… System.out      KOJKIILMF_U
1.508  4676  4676  com.Aldughayfiq.nf… System.out      192 Cipher Text in Binary:
1.508  4676  4676  com.Aldughayfiq.nf… System.out      110010111100111111001010110010111100100111001001110011001100110 1
1.508  4676  4676  com.Aldughayfiq.nf… System.out      1100011011011111010101011001110010000111100011011001110100000111
1.508  4676  4676  com.Aldughayfiq.nf… System.out      0011011101110001100110111101111110001111100011010010100101011110
1.508  4676  4676  com.Aldughayfiq.nf… System.out      Key
1.508  4676  4676  com.Aldughayfiq.nf… System.out      0100101111000011101000001100110100000000001100110000000000000000
```

Figure 5.9 shows the encryption part of Phase 3.2 at the phone application.



```
**The 2nd command  took 40 milliseconds


<< Command APDU: 80A80000048302800000
>> Response Data: CBCFCACBC9C9CCCDC6DF559C878D9D0737719BDF8F8D295E3DE901688B58E0C34B1BCEBDD3CFCE1F659000
Data read : ËÏÊÊÉÉÌÍÆßU       7q  ß    )^=éh  XàĂK Î½ÓÏÎ e

================================================Start of Dec. Part========================================================
Warning:  Property storage name for 5 is empty - setting to Root Entry
The Dec. key:1111111010101010110101011101010101010101011001010000000001010101
Received Account number(Enc.): CBCFCACBC9C9CCCDC6DF559C878D9D0737719BDF8F8D295E

The new Key: 0100101111000011101000001100110100000000001100110000000000000000The dec. timeing : 4 milleseconds


Received Account number (Dec.): 405466327823123316040181


=================================================End of Dec. Part======================================================



**The 3rd command  took 1735 milliseconds


<< Command APDU: 00DA0400137472616E73616374696F6E617070726F76656400
>> Phone Response:9000

**The last command took 48 milliseconds

>> Ack. recevied

============================
| Transaction accepted :) !! |
============================
```

Figure 5.10 shows the response of the phone to the second command as well as the bank server process (Phase 3.3).



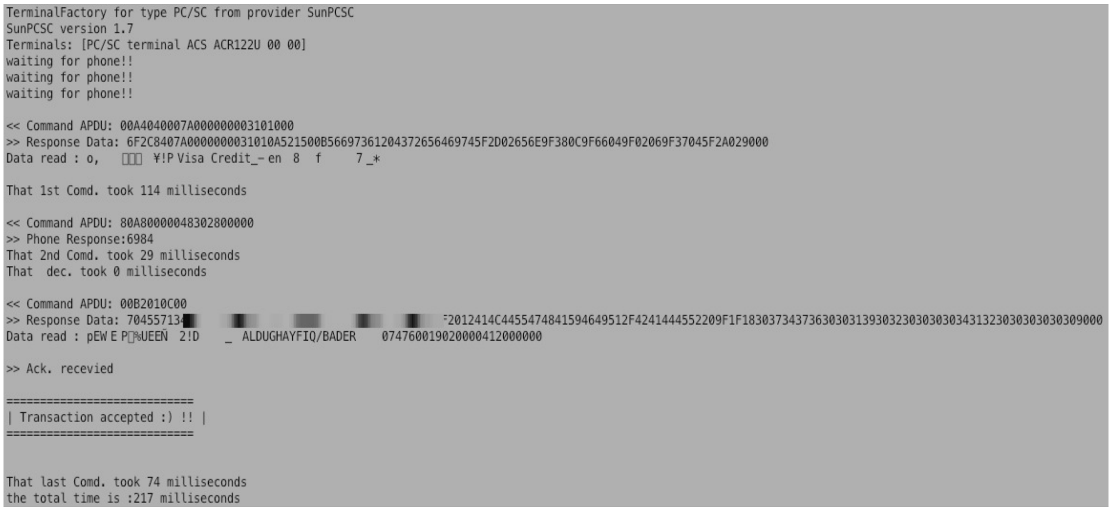
```
1969-12-31 20:00:00 HCE onCreate: Requested Android to
make Bader's Bank the default payment app
1969-12-31 20:00:00 CardService onCreate
1969-12-31 20:00:00 CardService
>> Received selected AID:
00A4040007F001020304050600
<<Response:
57656C636F6D6520746F20426164657227732042616E6B9
000
1969-12-31 20:00:00 CardService
>> Received GPO (get processing options):
 80A80000048302800000
<<Response:
CBCFCACBC9C9CCCDC6DF559C878D9D0737719BDF8F8D
295E3DE901688B58E0C34B1BCEBDD3CFCE1F659000
1969-12-31 20:00:00 CardService
>>Transaction Aprroved
1969-12-31 20:00:00 CardService onDeactivated(0)
```

Figure 5.11 shows the process of Phase 3.2 and Phase 3.3 at the phone application.

5.2 Implementation of Other Payment Systems

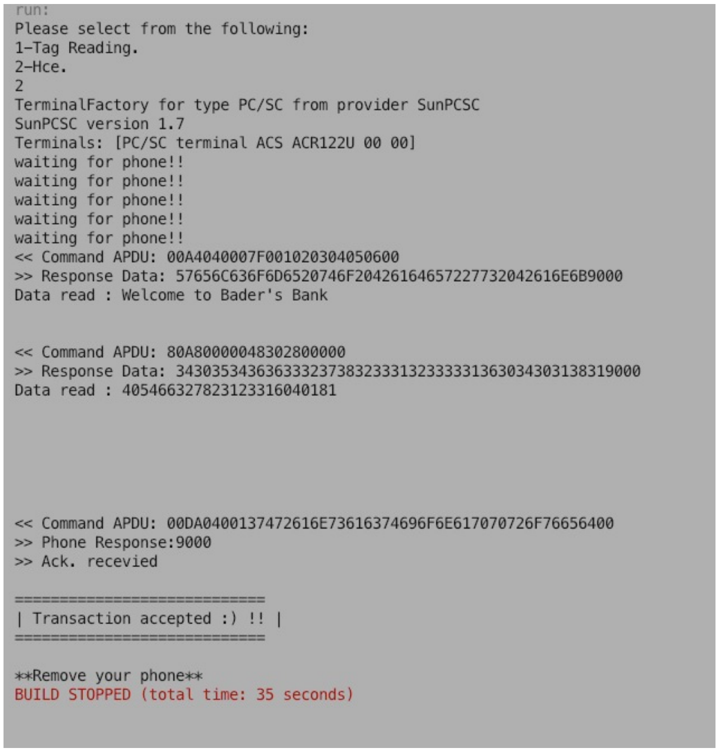
In this section we present an implementation of the contactless payment system and mobile payment system that uses the NFC technology. We developed a java application to enable the POS reader to communicate with the contactless card (i.e. Credit Card using the contactless technology). With the same application the POS reader communicates with an Android application (i.e. we developed) installed on the phone. The Android application emulates a credit card without any modification to function similarly to the currently used NFC-phone payment system.

In this implementation, we managed to collect all the banking information (i.e. the account number, the expiry date, the name of the card holder, and the issuer code) needed to make a payment. As a result, we found that the information was sent with no encryption. Similarly, the android application sent the information in plain text. Following are snapshots of the implementation (see figures 5.12, and 5.13).



```
TerminalFactory for type PC/SC from provider SunPCSC
SunPCSC version 1.7
Terminals: [PC/SC terminal ACS ACR122U 00 00]
waiting for phone!!
waiting for phone!!
waiting for phone!!

<< Command APDU: 00A4040007A000000003101000
>> Response Data: 6F2C8407A0000000031010A521500B566973612043726564697 45F2D02656E9F380C9F66049F02069F37045F2A029000
Data read : o,   ▯▯▯ ¥!P Visa Credit_- en  8  f     7_*

That 1st Comd. took 114 milliseconds

<< Command APDU: 80A80000048302800000
>> Phone Response:6984
That 2nd Comd. took 29 milliseconds
That  dec. took 0 milliseconds

<< Command APDU: 00B2010C00
>> Response Data: 70455713[redacted]F2012414C44554748415946 49512F4241444552209F1F18303734373630303139303230303030343132303030303030 9000
Data read : pEW E P▯%UEEÑ  2!D    _ ALDUGHAYFIQ/BADER    07476001902000041200000 0

>> Ack. recevied

============================
| Transaction accepted :) !! |
============================


That last Comd. took 74 milliseconds
the total time is :217 milliseconds
```

Figure 5.12 shows the implementation of contactless card payment system (we blurred the card number for security concerns).



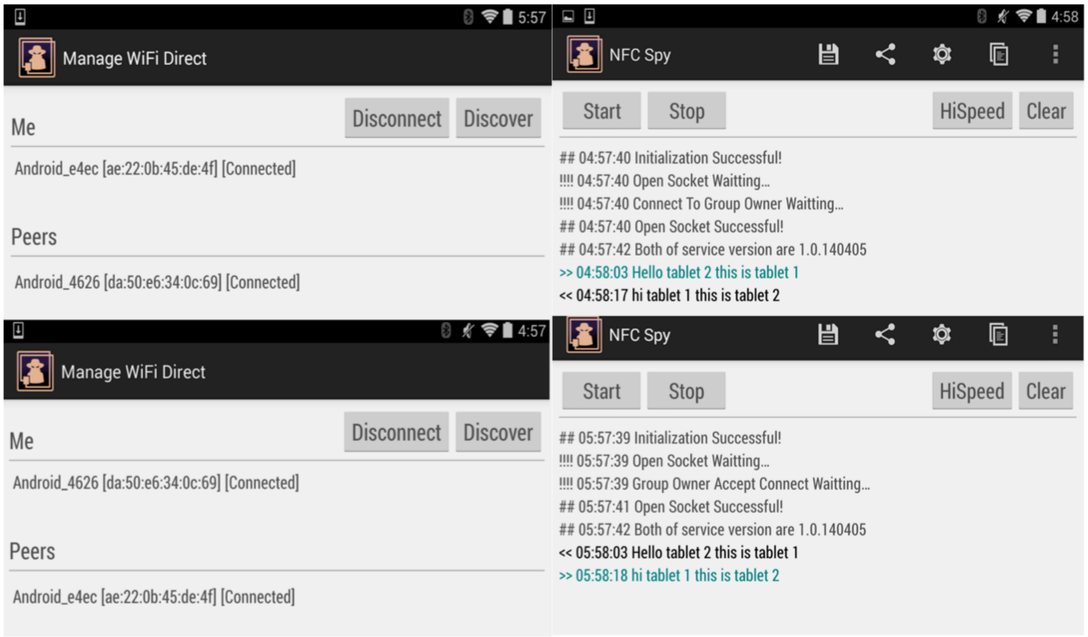
```
run:
Please select from the following:
1-Tag Reading.
2-Hce.
2
TerminalFactory for type PC/SC from provider SunPCSC
SunPCSC version 1.7
Terminals: [PC/SC terminal ACS ACR122U 00 00]
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
<< Command APDU: 00A4040007F001020304050600
>> Response Data: 57656C636F6D6520746F204261646572277320426616E6B9000
Data read : Welcome to Bader's Bank


<< Command APDU: 80A80000048302800000
>> Response Data: 3430353436363332373832333132333331363034303138319000
Data read : 4054663278231233160401 81
```

Figure 5.13 shows the implementation of NFC-enabled mobile payment system.

5.3 Relay attack Implementation

In this section we will launch the relay attack on a three payment systems, which all use the NFC technology. The three payment systems are the contactless card payment system, NFC-phone payment system, and the proposed system. Where we used the proof of concept implementation for all of them. The aim of launching the attack is to prove the resistance of the proposed system to the relay attack compared with the other currently used systems. To launch the attack we used two LG Nexus 7 tablets, which are enabled with NFC technology. The tablets use Android OS version 4.4.2, Snapdragon S4 Pro quad-core 1.5GHz processor, and internal memory storage of 16GB. The Android application we used is NFCSpy, which is an open source application developed by [43].

The application is used to launch the relay attack on the mentioned above systems. The application uses WLAN direct (WiFi-P2P), which enables the two tablets to communicate with each other. Figure 5.14 shows the connection setup and message exchanging between the two Nexus tablets in this implementation.



Me

Android_e4ec [ae:22:0b:45:de:4f] [Connected]

Peers

Android_4626 [da:50:e6:34:0c:69] [Connected]

```
## 04:57:40 Initialization Successful!
!!!! 04:57:40 Open Socket Waitting...
!!!! 04:57:40 Connect To Group Owner Waitting...
## 04:57:40 Open Socket Successful!
## 04:57:42 Both of service version are 1.0.140405
>> 04:58:03 Hello tablet 2 this is tablet 1
<< 04:58:17 hi tablet 1 this is tablet 2
```

Me

Android_4626 [da:50:e6:34:0c:69] [Connected]

Peers

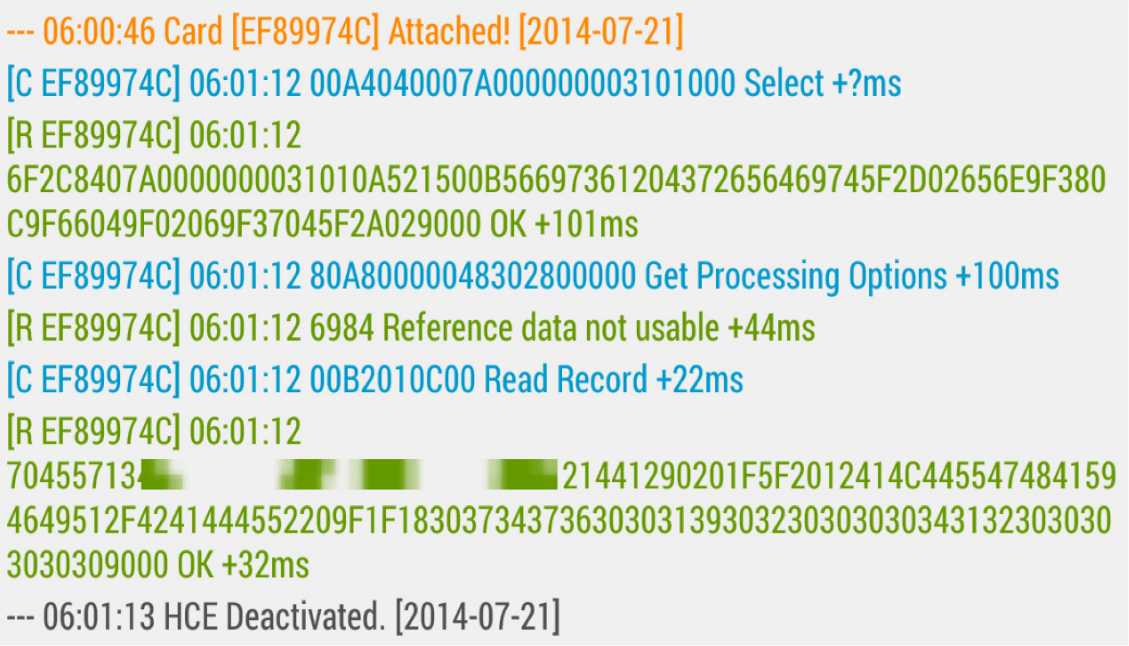
Android_e4ec [ae:22:0b:45:de:4f] [Connected]

```
## 05:57:39 Initialization Successful!
!!!! 05:57:39 Open Socket Waitting...
!!!! 05:57:39 Group Owner Accept Connect Waitting...
## 05:57:41 Open Socket Successful!
## 05:57:42 Both of service version are 1.0.140405
<< 05:58:03 Hello tablet 2 this is tablet 1
>> 05:58:18 hi tablet 1 this is tablet 2
```

Figure 5.14 shows the connection setup and message exchanging between the two tablets.

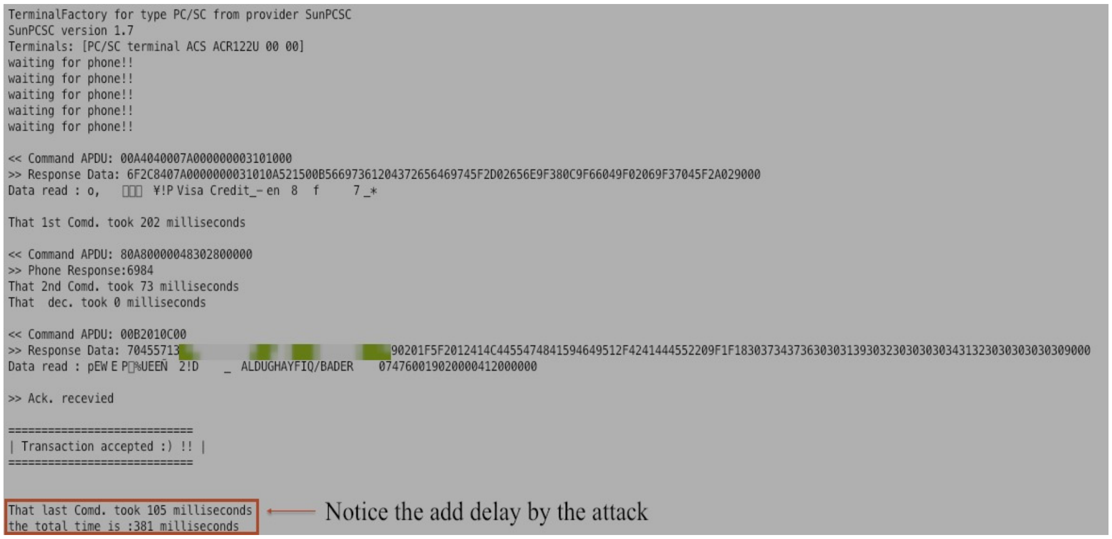
Snapshots are presented of the desktop application interface and the NFCSpy application in both tablets to demonstrate the launching of relay attack on the three systems.

1. Relay attack on contactless smart card payment system



```
--- 06:00:46 Card [EF89974C] Attached! [2014-07-21]
[C EF89974C] 06:01:12 00A4040007A000000003101000 Select +?ms
[R EF89974C] 06:01:12
6F2C8407A0000000031010A521500B56697361204372656469745F2D02656E9F380
C9F66049F02069F37045F2A029000 OK +101ms
[C EF89974C] 06:01:12 80A80000048302800000 Get Processing Options +100ms
[R EF89974C] 06:01:12 6984 Reference data not usable +44ms
[C EF89974C] 06:01:12 00B2010C00 Read Record +22ms
[R EF89974C] 06:01:12
70455713[redacted]21441290201F5F2012414C445547484159
4649512F4241444552209F1F183037343736303031393032303030303431323030 30
3030309000 OK +32ms
--- 06:01:13 HCE Deactivated. [2014-07-21]
```

Figure 5.15 shows a snapshot of the log screen in NFCSpy application (we blurred the card number for security concerns).

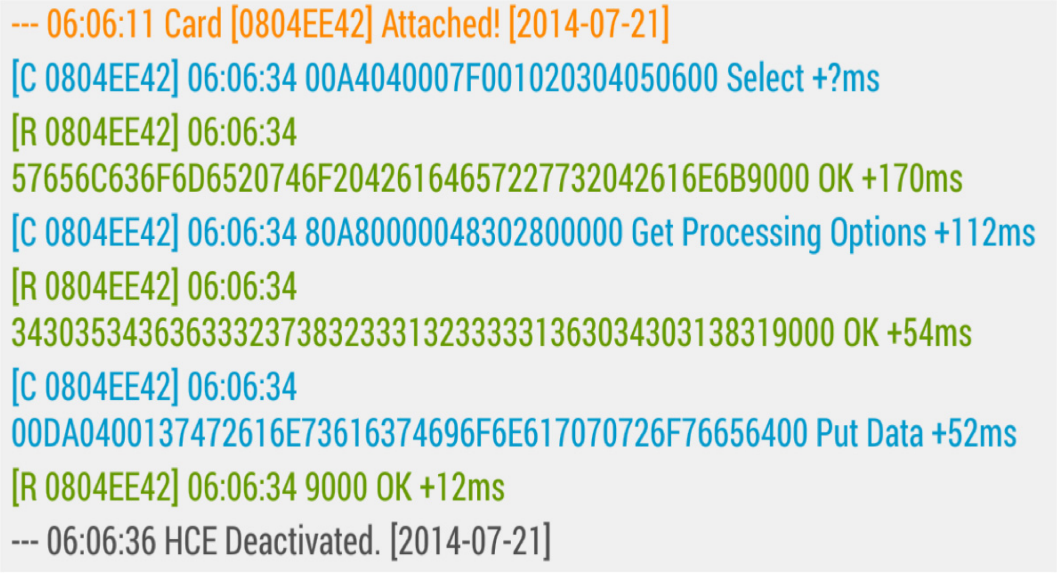


```
TerminalFactory for type PC/SC from provider SunPCSC
SunPCSC version 1.7
Terminals: [PC/SC terminal ACS ACR122U 00 00]
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!

<< Command APDU: 00A4040007A000000003101000
>> Response Data: 6F2C8407A0000000031010A521500B56697361204372656469745F2D02656E9F380C9F66049F02069F37045F2A029000
Data read : o,   ▯▯▯  ¥!P Visa Credit_- en  8   f     7 _*

That 1st Comd. took 202 milliseconds

<< Command APDU: 80A80000048302800000
>> Phone Response:6984
That 2nd Comd. took 73 milliseconds
That  dec. took 0 milliseconds

<< Command APDU: 00B2010C00
>> Response Data: 70455713[redacted]90201F5F2012414C4455474841594649512F4241444552209F1F183037343736303031393032303030303431323030303030309000
Data read : pEW E P▯%UEEÑ  2!D    _  ALDUGHAYFIQ/BADER    07476001902000041200000

>> Ack. recevied

=============================
| Transaction accepted :) !! |
=============================


That last Comd. took 105 milliseconds
the total time is :381 milliseconds
```

Notice the add delay by the attack

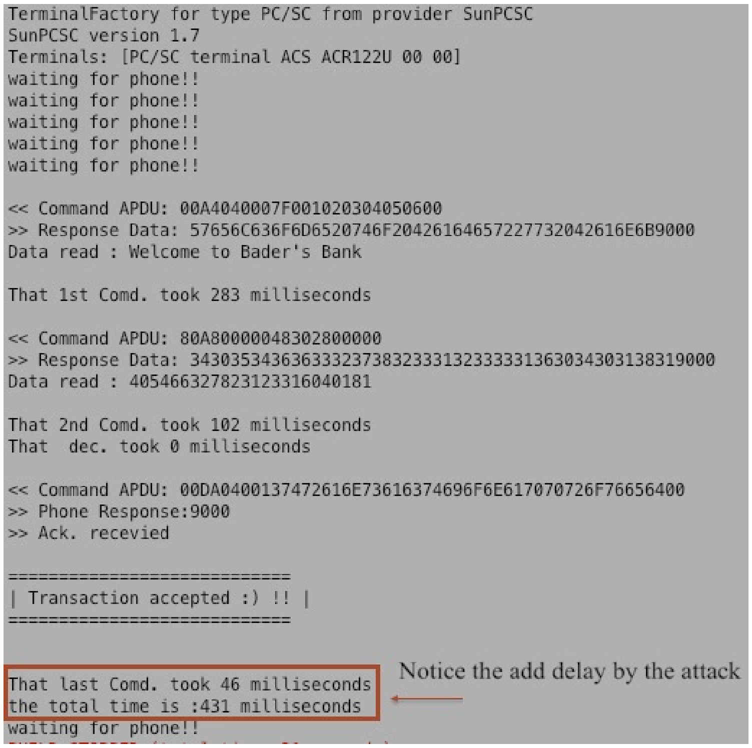
Figure 5.16 shows a snapshot of the POS reader application during the attack on contactless card payment (we blurred the card number for security concerns).

2. Relay attack on NFC-mobile application payment system



```
--- 06:06:11 Card [0804EE42] Attached! [2014-07-21]
[C 0804EE42] 06:06:34 00A4040007F001020304050600 Select +?ms
[R 0804EE42] 06:06:34
57656C636F6D6520746F2042616465722773204261 6E6B9000 OK +170ms
[C 0804EE42] 06:06:34 80A80000048302800000 Get Processing Options +112ms
[R 0804EE42] 06:06:34
3430353436363332373832333132333331363034303138319000 OK +54ms
[C 0804EE42] 06:06:34
00DA0400137472616E73616374696F6E617070726F76656400 Put Data +52ms
[R 0804EE42] 06:06:34 9000 OK +12ms
--- 06:06:36 HCE Deactivated. [2014-07-21]
```

Figure 5.17 a snapshot of the log screen in NFCSpy application, implementation the attack on NFC-enabled mobile payment.

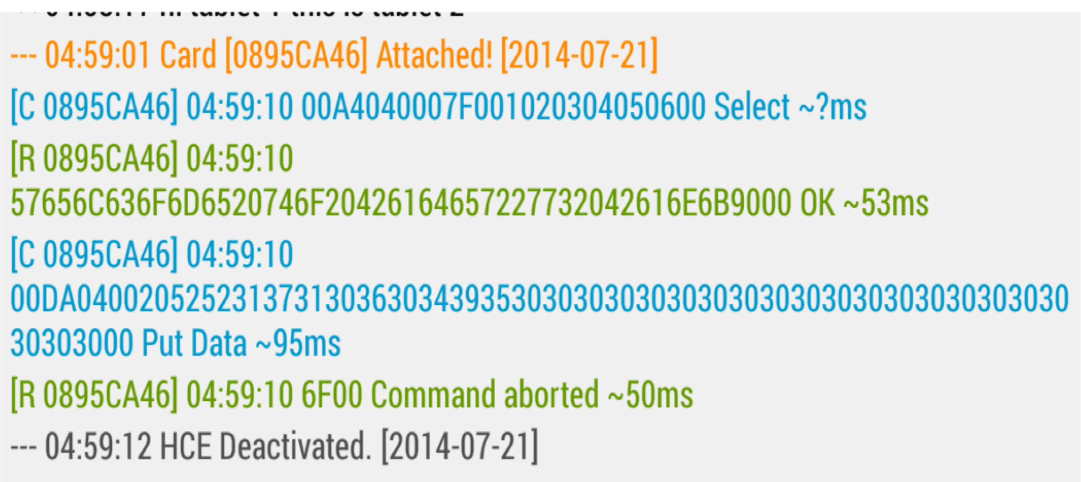


```
TerminalFactory for type PC/SC from provider SunPCSC
SunPCSC version 1.7
Terminals: [PC/SC terminal ACS ACR122U 00 00]
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!

<< Command APDU: 00A4040007F001020304050600
>> Response Data: 57656C636F6D6520746F20426164657227732042616E6B9000
Data read : Welcome to Bader's Bank

That 1st Comd. took 283 milliseconds

<< Command APDU: 80A80000048302800000
>> Response Data: 3430353436363332373832333132333331363034303138319000
Data read : 4054663278231233316040181

That 2nd Comd. took 102 milliseconds
That  dec. took 0 milliseconds

<< Command APDU: 00DA0400137472616E73616374696F6E617070726F76656400
>> Phone Response:9000
>> Ack. recevied

============================
| Transaction accepted :) !! |
============================

That last Comd. took 46 milliseconds
the total time is :431 milliseconds
waiting for phone!!
```

Notice the add delay by the attack

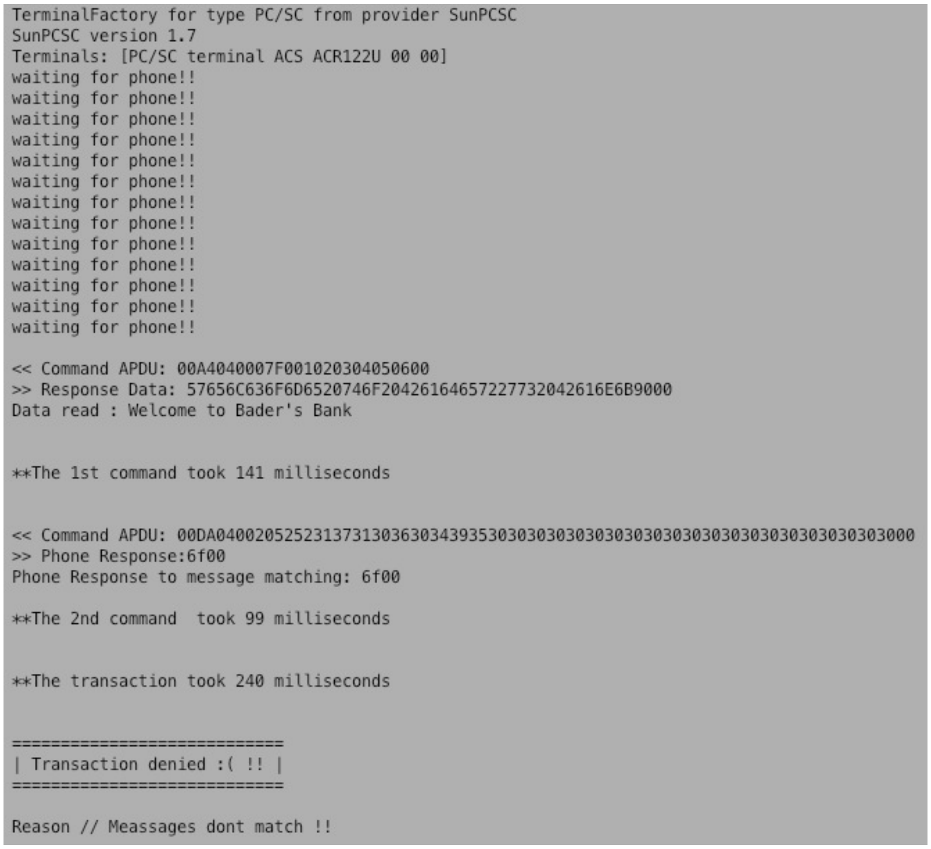
Figure 5.18 a snapshot of the POS reader application for NFC-mobile payment during the attack.

3. Relay attack on proposed payment system



```
--- 04:59:01 Card [0895CA46] Attached! [2014-07-21]
[C 0895CA46] 04:59:10 00A4040007F001020304050600 Select ~?ms
[R 0895CA46] 04:59:10
57656C636F6D6520746F204261646572277320426 16E6B9000 OK ~53ms
[C 0895CA46] 04:59:10
00DA040020525231373130363034393530303030303030303030303030303030303030
30303000 Put Data ~95ms
[R 0895CA46] 04:59:10 6F00 Command aborted ~50ms
--- 04:59:12 HCE Deactivated. [2014-07-21]
```

Figure 5.19 a snapshot of NFCSpy application during the attack on the proposed system.



```
TerminalFactory for type PC/SC from provider SunPCSC
SunPCSC version 1.7
Terminals: [PC/SC terminal ACS ACR122U 00 00]
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!
waiting for phone!!

<< Command APDU: 00A4040007F001020304050600
>> Response Data: 57656C636F6D6520746F20426164657227732042616E6B9000
Data read : Welcome to Bader's Bank


**The 1st command took 141 milliseconds


<< Command APDU: 00DA04002052523137313036303439353030303030303030303030303030303030303030303000
>> Phone Response:6f00
Phone Response to message matching: 6f00

**The 2nd command  took 99 milliseconds


**The transaction took 240 milliseconds


============================
| Transaction denied :( !! |
============================

Reason // Meassages dont match !!
```

Figure 5.20 a snapshot of POS application for proposed system during launching the attack.

5.4 Evaluation of Experimental Results

In this section we will evaluate the proposed system by calculating the computational cost, and the communication cost. The communication cost is calculated with and without launching the relay attack. In addition, we will analyze the encrypted message using an open source tool called Cryptool.

5.4.1 Computational cost

In this section we will discuss the computational cost for each entity of the proposed system. We analyzed the computational cost based on the use of the following operations:

- Exclusive Logical-OR (XOR) operation,
- Comparison operation (CMP),
- Pseudorandom Number Generator (PRNG),
- Hash function operation,
- And the operations added from the new encryption mechanism Inversion operation, and 3-bit comparison.

5.4.1.1 The Phone Application and The Application Server

After the phone received the command containing the *StoreID* from the POS, the phone needs to match both messages. The comparison operation will be executed once per a transaction. In addition, we added the computational cost of the communication between the phone and the application's server (*ComPhS*). As well as, the application's server will execute hash operations in addition to the computational cost of the encryption process. The computational cost of the communication between the phone and the application's server (*ComSPh*) is added to the total. The cost *ComPhS* and *ComSPh* are based on the assumption where both communications are secure. Table 2 and 3 show, respectively, the computational cost at the phone application and the application server.

Computational Operations	Number of times executed
CMP	1
Total	1+ComPhS

Table 5.1 shows the computational cost at the phone application.

Computational Operations	Number of times executed
Hash function	1
Enc. cost	((7*256)*2rounds)
Total	3+Enc.+ComSPh

Table 5.2 shows the computational cost at the application server.

5.4.1.2 POS reader

In the POS reader, the only operations calculated are related to generating the *StoreID*. Since we are assuming the communication between the POS reader and the bank server is secure. We added its computational cost variable depending on what cryptography algorithm is used we refer to it as *ComPB*. Table 4 shows the computational cost at the POS reader.

Computational Operations	Number of times executed
XOR	1
PRNG	1
Hash function	1
Total	3+ ComPB

Table 5.3 shows the computational cost at POS reader.

5.4.1.3 The Bank server

In the way of completion of a transaction, the bank server will execute a number of operations. These operations are related to the phone authentication and the integrity verification of the encrypted message. In addition, we added the decryption computational cost and the computational cost of the communication between the bank server and the POS server to the total cost. Table 5 shows the total computational cost at the bank server.

Computational Operations	Number of times executed
CMP	2*n
Dec. cost	((7*256)*2rounds)
Total	2*n+Dec.+ComBP

Table 5.4 the computational cost at the bank server.

5.4.1.4 The New Encryption Mechanism

Based on the operations we used in our implementation of the new cryptography approach, we calculated the computational cost of the encryption or decryption. Where the key generation process is included in the encryption or decryption process (see table 6).

Computational Operations	Number of times executed
XOR	4
Inversion	2
3-bit comparison	1
Total	(7*message bits) per round

Table 5.5 computational cost of the new cryptography approach implementation.

5.4.2 Communicational cost

In this section we calculated the total communication cost for all entities in the proposed system based on two cases. These cases are the normal execution of the system without any attacks and the execution with lunching a relay attack on the system.

First, we calculate the communication cost between the phone application and the POS reader, where they are exchanging the APDU commands. The communication cost is represented by the following equation:

$$\textbf{communication cost phone to reader } (\mathbf{t}_{Pr}) = \boldsymbol{tcomd_i} + \boldsymbol{tcomd_{i+1}} + \dots + \boldsymbol{tcomd_{i+2}} \textbf{ (Equation 1)}$$

Where:

$\boldsymbol{t_{comdi}}$: is the time for sending the command and receiving the response. It is calculated at the POS reader.

Second, The communication cost for the phone application to match the *StoreID* and for the application server to encrypt the *ACT#* is represented by the following equation:

$$\textbf{communication cost at the application server } (\boldsymbol{ts}) = \boldsymbol{t_{enc.}} + \boldsymbol{t_{cmp}} \textbf{ (Equation 2)}$$

Where:

$\boldsymbol{t_{cmp}}$: is the time to match the *StoreID* message,

and $\boldsymbol{t_{enc.}}$ is the time for the encryption process. The communication cost for the encryption process is represented by the following equation:

$$\boldsymbol{t_{enc.}} = \mathbf{4} * \boldsymbol{t_{XOR}} + \mathbf{2} * \boldsymbol{t_{inv.}} + \boldsymbol{t_{3-bit\ comp.}} \textbf{ per bit (Equation 3)}$$

Third, the communication cost at the bank server, which is represented by the following equation:

$$\textbf{communication cost at bank server } \boldsymbol{t_{Bs}} = \mathbf{2} * \boldsymbol{t}_{\boldsymbol{cmp}} + \boldsymbol{t_{dec.}} \textbf{ (Equation 4)}$$

Where:

$\boldsymbol{t_{cmp}}$: is the time to compare the *sign.* message and the *ACT#* with the bank server database,

$t_{dec.}$: is the time for the decryption process, and its cost represented by the following equation:

$$t_{dec.} = 4 * t_{XOR} + 2 * t_{inv.} + t_{3-bit\ comp.} \text{ per bit (Equation 5)}$$

Finally, the total communication cost for the proposed system in normal execution case, is represented by the following equation:

$$\text{The total communication cost } (T_c) = t_{pr} + t_{RBs} + t_{Bs} \text{ (Equation 6)}$$

Where:

t_{RBs}: is the communication cost between the POS reader , and the Bank server.

However, the total communication cost for the proposed system with launching the relay attack, is represented by the following equation:

$$\text{Total communication cost } (T_{cattk.}) = t_{comds} + t_{ad} \text{ (Equation 7)}$$

Where:

t_{comds}: is the communication cost of first two commands, because the phone will reject any further commands.

t_{ad}: is the communication cost of the delay add by the communication channel of the attack.

Moreover, we present an analysis of the communication cost calculated of the implementation experimental results. We implemented the three payment systems, the contactless smart card, the NFC-enabled phone payment system, and the proposed system.

First, we analyze the communication cost without launching the relay attack. Table 5.6 shows the time comparison between the payment systems in milliseconds. We noticed the time of the third command of the proposed system is more than the other payment systems and that is because of our implementation if the encryption mechanism, which is the worst-case implementation and can be optimized to reduce the delay time. However, the time for each command in the proposed system as well as the other systems is less than 5 seconds per command, which meets the time specifications of the standards. Figures 5.21 and 5.22 show the comparison Charts of the communication cost of the three payment systems.

Scenarios	**1st command**	**2nd command**	**3rd command****	**Total time**
Credit Card	88.00	0.00	72.00	187.00
NFC-enabled phone	215.00	0.00	55.00	279.00
Proposed system	56.00	31.00	1756.00	1843.00

*The 2nd command is not applicable for other scenarios, because it is about the *StoreID* message matching process.

Table 5.6 shows the time for each command of the three payment systems.

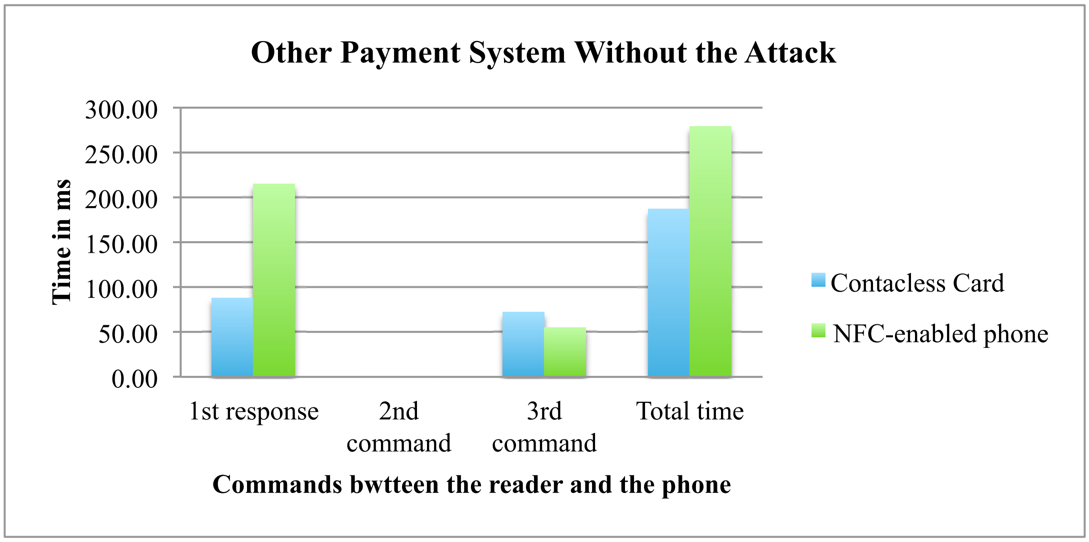


Figure 5.21 shows the communication cost for the other payment systems.

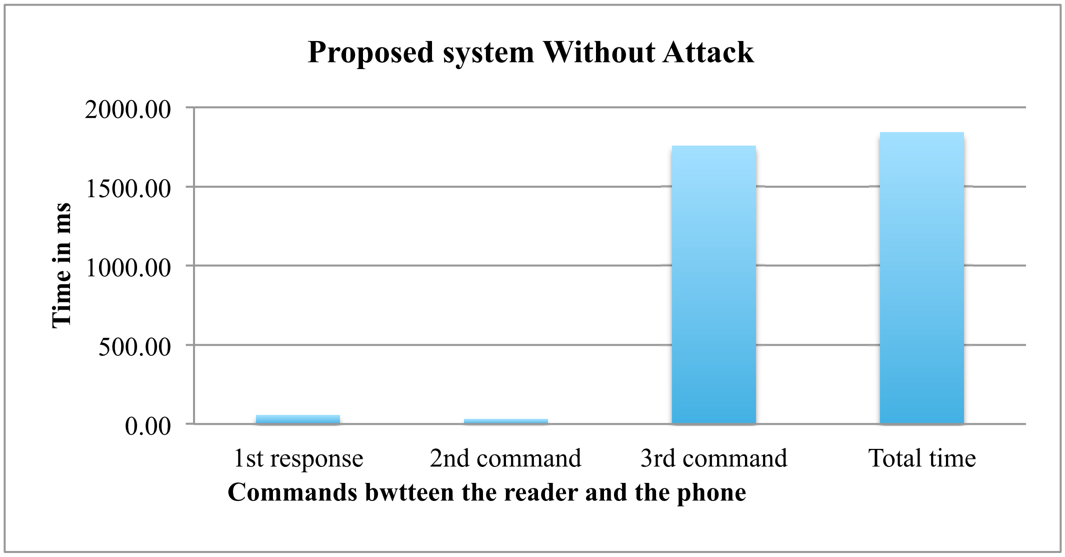


Figure 5.22 the communication cost for the proposed system.

Second, we analyze the communication cost with launching the relay attack on the three systems. Table 5.7 shows the comparison of the three systems during the attack. We can see that the proposed system has no value for the third command, because the *StoreID* message did not match in the phone, so further commands were rejected. On the other hand, we can see the other systems were vulnerable to the attack and the transaction went through. Also, we can notice that the added delay time from the attack is less than 5 seconds, which meets the time for a transaction that specified by EMV standards. Figures 5.23 and 5.24 show the communication cost of the three systems with launching the attack.

Scenarios	**1st command**	**2nd command**	**3rd command**	**Total time**
Credit Card	88.00	0.00	103.00	191.00
NFC-enabled phone	173.00	0.00	207.00	380.00
Proposed system	262.00	254.00	0.00	516.00

***The 3rd command value is zero in the last scenario, because the new approach will not advance to the other command if the messages don't match in the 2nd command. *The 2nd command is not applicable for other scenarios, because it is about the StoreID message matching process.**

Table 5.7 shows the comparison of the three systems during the attack.

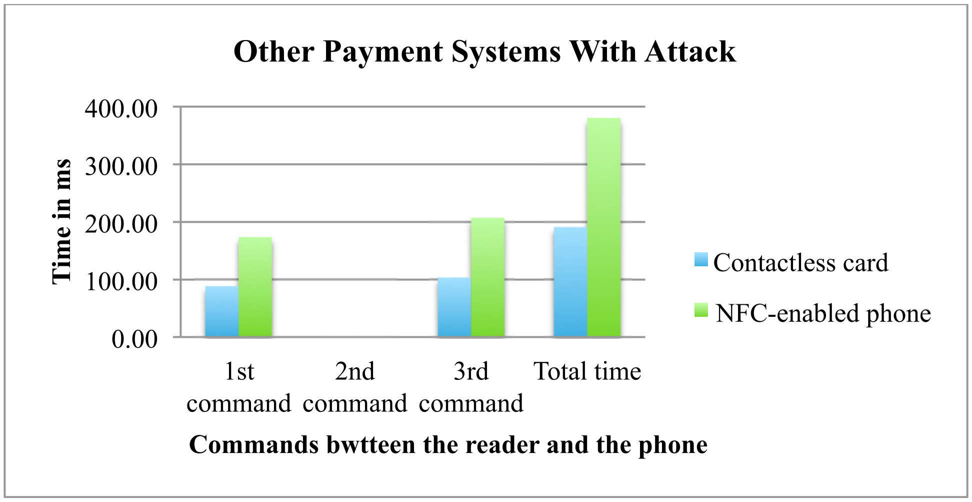


Figure 5.23 shows the communication cost comparison of other payment systems with attack.

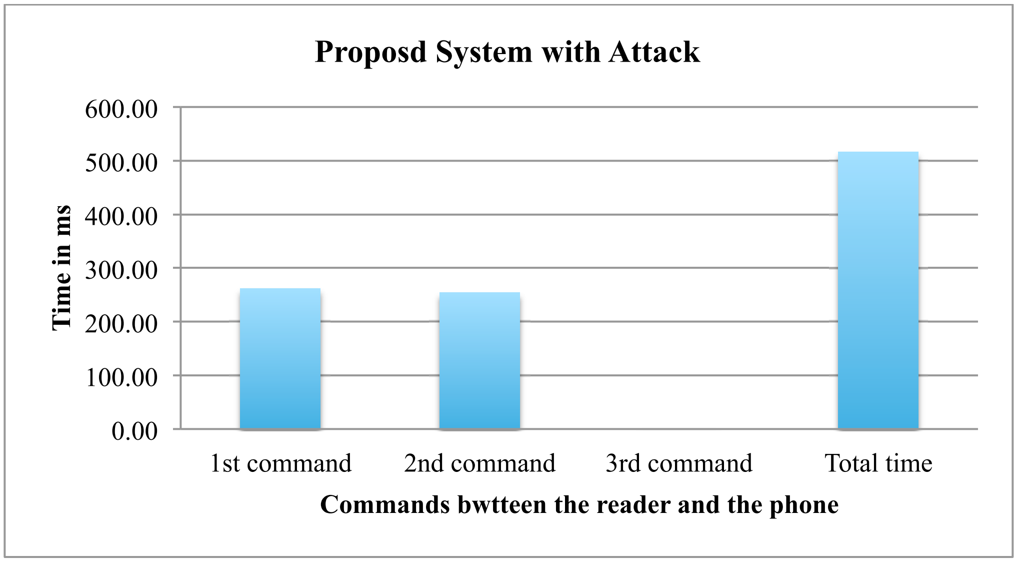


Figure 5.24 shows the communication cost of the proposed system with attack.

5.4.3 Cryptool analysis for the encryption mechanism

The Cryptool is an open source tool used for cryptanalysis [44]. We tested the encrypted message with the new cryptography approach [37] using this tool. As we mentioned in section 4.3, the key generation part in [37] uses a stream cipher process; however we also encrypted the original message (i.e. before encryption) with other well-known algorithms such as AES, DES, IDEA, MARS, and Two Fish. We present in figure 5.25 a graph shows of a comparison of the encrypted messages for the new approach and the other mentioned above algorithms. The comparison was based on executing a number of cryptanalysis tests such as the entropy test, periodicity test, frequency test, poker test, run test, and serial test. The entropy test, and the frequency test are statistical tests used to test the frequent a letter or a certain pattern of letters occurring in the message bits. The periodicity test will check if a letter or a number is cycling through the message. The poker test, the run test, and the serial test will tests the randomness in the encrypted message. Moreover, the original message consists of hexadecimal numbers, which makes the randomness tests results less effective tests. Therefore, we are only interested on statistical tests results to compare the new approach with other algorithms. Table 5.8 shows the comparison result of the mentioned above tests for all algorithms. Even though, we run the other tests on the message to get an overview of the results in general.

As a result of the comparison, we found the level of strength for the encrypted message using the new approach is higher than most of the other algorithms, which are included in this analysis.

Alpha = 0.05		Entropy	Periodicity	Frequency Test		Poker Test		Run Test				Serial Test	
								Normal		Long			
		6.61		3.84	Status	14.07	Status	9.48	Status	34	Status	5.99	Status
Sample - 1 (256 bits)	AES 128	3.96	No	0.015625	Pass	0.929412	Pass	1.095022	Pass	7	Pass	0.121163	Pass
	Triple_DES_ECB_128	3.41	No	0.1875	Pass	3	Pass	6.867241	Pass	5	Pass	0.121401	Pass
	Triple_DES_CBC_128	3.45	No	5.0625	Fail	13.35294	Pass	7.742203	Pass	8	Pass	5.145343	Pass
	IDEA	3.39	2	0.015152	Pass	6.909091	Pass	1.783558	Pass	7	Pass	0.985021	Pass
	MARS	3.32	No	0.0625	Pass	2.435294	Pass	2.948561	Pass	8	Pass	0.970409	Pass
	Two Fish	3.64	No	0.5625	Pass	6.952941	Pass	1.52706	Pass	11	Pass	0.876605	Pass
	New Approach	4.45	No	0.390625	Pass	5.258824	Pass	3.350504	Pass	7	Pass	1.142157	Pass

Table 5.8 the detailed comparison of encrypted message between the new approach and the other algorithms.

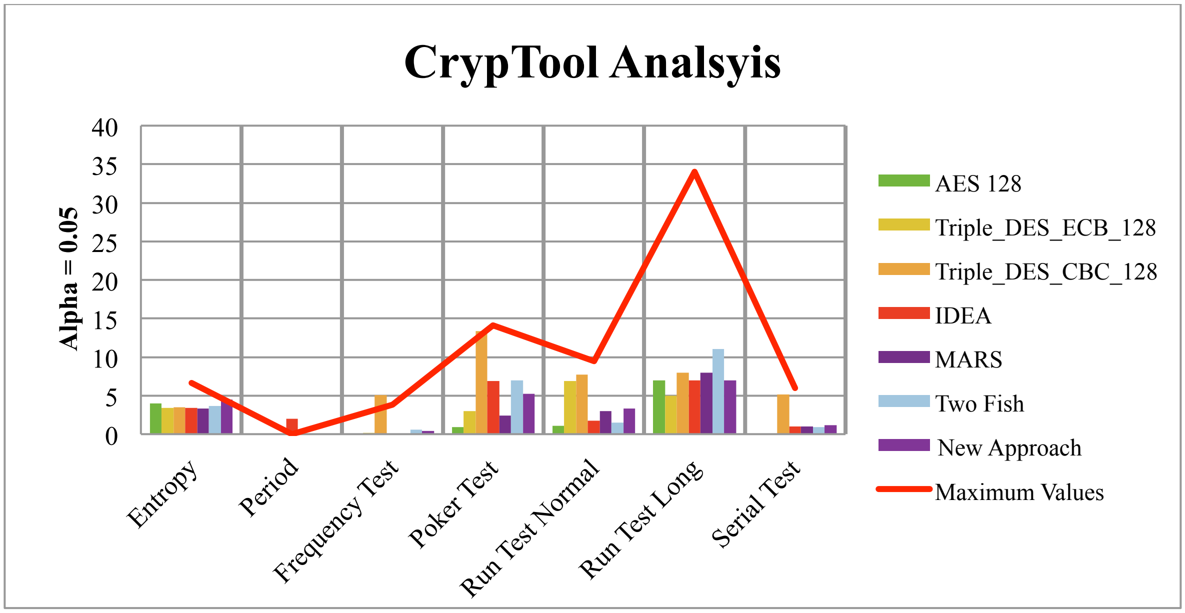


Figure 5.25 The comparison between the new approach and the other algorithms.

5.5 Summary

In this chapter we presented a proof of concept implementation of our proposed system. As well as, we presented an implementation of the currently used payment systems such as the contactless smart card, and the NFC-enabled phone payment. Moreover, we showed a practical implementation of the relay attack on the proposed system and the other systems mentioned above, in order to show the proposed system's ability to prevent the relay attack. We presented the result of our analysis of the three systems' behavior during the attack. In addition, we presented the evaluation methodology of our proposed system, where we analyzed the communication cost, and computational cost. Finally, we analyzed the new cryptography approach using a cryptanalysis tool called Cryptool. The next chapter will be the conclusion of this thesis.

Chapter 6 The Conclusion and Future Work

6.1 Conclusion

In this work, we proposed a new payment system that is based on NFC-enabled mobile payment. The system uses a phone application that emulates a contactless card to make payments. In the proposed system a phone will authenticate a POS reader using a new location detection mechanism that uses a hashed message (i.e. *StoreID*) generated by the POS reader. This message is stored in an NFC tag as well as in the internal memory of the POS reader to be used in the authentication process. Another feature, introduced by this mechanism is insuring the proximity of a phone to a POS reader in order to make a payment. This process proved that the system is resistant to relay attacks. Moreover, we offer the use of a new encryption mechanism to encrypt the banking information in order to protect the information from being sent over a public network in a plain text. This new approach uses a dynamic symmetric key for encryption. That key is updated after each transaction.

The proposed system meets the criteria mentioned above in Chapter 2 for payment systems, which are simplicity, reliability, and availability. It also resists the relay attack as we showed earlier in this thesis.

6.2 Future work

The proposed system can be optimized to offer a new mechanism that authenticates the phone at the POS reader, where this system only offers the authentication of a phone at the bank server. Moreover, the proposed system can be optimized to provide an offline mode to make a payment, where this system proposes an online mode only for making payments. Finally, possible future work is to develop a new protocol for RIFD systems that is resistant to the relay attack.

References

[1] Smart Card Alliance Contactless Payments Council White. (2006, 11). *The What, Who and Why of Contactless Payments.* Retrieved 07 2014, from Smart Card Alliance: http://www.smartcardalliance.org/resources/pdf/CP_What_Who_Why_Final.pdf

[2] *dialing in the future of mobile payments in canada.* (2012, 01). Retrieved 07 2014, from Deloitte: http://www.deloitte.com/assets/Dcom-Canada/Local%20Assets/Documents/FSI/ca_en_FSI_The_future_of_the_mobile_payments_211211.pdf

[3] Ondrus, J., & Pigneur, Y. (2007). An Assessment of NFC for Future Mobile Payment Systems. *Management of Mobile Business International Conference* (p. 43). Toronto, Ont.: IEEE.

[4] Valcourt, E., Robert, J.-M., & Beaulieu, F. (2005). Investigating mobile payment: supporting technologies, methods, and use. *Wireless And Mobile Computing, Networking And Communications. 4*. IEEE.

[5] Alliance, S. C. (2011, 09). *The Mobile Payments and NFC Landscape: A U.S. Perspective, A Smart Card Alliance Payments Council White Paper.* Retrieved from Smart Card Alliance : http://www.smartcardalliance.org/publications-the-mobile-payments-and-nfc-landscape-a-us-perspective/

[6] Commission, I. O. (n.d.). ISO/IEC 18092:2013: Information technology -- Telecommunications and information exchange between systems -- Near Field Communication -- Interface and Protocol (NFCIP-1).

[7] Commission, I. O. (2012). ISO/IEC 21481: Information technology -- Telecommunications and information exchange between systems -- Near Field Communication Interface and Protocol -2 (NFCIP-2).

[8] Madlmayr, G., Langer, J., Kantner, C., & Scharinger, J. (2008). NFC Devices: Security and Privacy. *Availability, Reliability and Security.* Barcelona: IEEE.

[9] *MovieTickets.com*. (n.d.). Retrieved from http://www.movietickets.com/

[10] *Tribute.ca*. (n.d.). Retrieved from http://www.tribute.ca/

[11] *Cineplex*. (n.d.). Retrieved from http://www.cineplex.com.

[12] *Interac.* (n.d.). Retrieved from https://www.interac.ca

[13] Halevi , T., Ma , D., Saxena, N., & Xiang , T. (2012). Secure Proximity Detection for NFC Devices Based on Ambient Sensor Data. *7459*, 379-396.

[14] Juels, A. (2006). RFID security and privacy: a research survey. *Selected Areas in Communications , 24* (2), 381 - 394.

[15] Alliance, S. C. (2011, 09). *The Mobile Payments and NFC Landscape: A U.S. Perspective, A Smart Card Alliance Payments Council White Paper.* Retrieved from Smart Card Alliance : http://www.smartcardalliance.org/publications-the-mobile-payments-and-nfc-landscape-a-us-perspective/

[16] DeLisle, J.-J. (2014, 04). *NFC Prepares For Wide Adoption*. Retrieved from microwaves and rf: http://mwrf.com/active-components/nfc-prepares-wide-adoption

[17] Finkenzeller, K. (2010). *FUNDAMENTALS AND APPLICATIONS IN CONTACTLESS SMART CARDS, RADIO FREQUENCY IDENTIFICATION AND NEAR-FIELD COMMUNICATION.* John Wiley and Sons.

[18] wifi, r. o. (2012, 02). *Graphic: Comparing Wireless Technology Range and Data Rates*. Retrieved 07 2014, from revoluti on wifi: http://www.revolutionwifi.net/2012/02/graphic-comparing-wireless-technology.html

[19] Pankaj, A., & Sharad, B. (2012). Near Field Communication. *10*, pp. 67-74.

[20] Android, G. (n.d.). *Host-based Card Emulation*. Retrieved 07 2014, from http://developer.android.com/guide/topics/connectivity/nfc/hce.html#ImplementingService

[21] Hoepman, J.-H., & Siljee, J. (2007). *Beyond RFID: the NFC Security Landscape.* TNO information and communication technology.

[22] Roland, M. (2012). *Applying recent secure element relay attack scenarios to the real world: Google Wallet Relay Attack.* University of Applied Sciences

Upper Austria, NFC Research Lab Hagenberg . University of Applied Sciences Upper Austria.

[23] Hancke, G. (2005). *A Practical Relay Attack on ISO 14443 Proximity Cards*. University of Cambridge.

[24] Kfir, Z., & Wool, A. (2005). Picking Virtual Pockets using Relay Attacks on Contactless Smartcard. *In Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05). 58*, p. 47. Washington, DC: IEEE Computer Society.

[25] Francis, L., Hancke, G., Mayes, K., & Markantonakis, K. (2012). Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. *ISO Press*

[26] Hancke, G., Mayes, K., & Markantonakis, K. (2009). Confidence in smart token proximity: Relay attacks revisited. *Computers & Security , 28* (7), 615-627.

[27] Balan, R., Ramasubbu, N., Prakobphol, K., Christin, N., & Hong, J. mFerio: the design and evaluation of a peer-to-peer mobile payment system. *MobiSys 2009* (pp. 291-304). New York: ACM.

[28] Kadambi, K., Li, J., & Karp, A. (2009). Near-field communication-based secure mobile payment service. *In Proceedings of the 11th international Conference on Electronic Commerce* (pp. 142–151). ACM.

[29] Chen, W., Hancke, G., Mayes, K., Lien, Y., & Chiu, J.-H. (2010). NFC Mobile Transactions and Authentication Based on GSM Network. *Near Field Communication (NFC), 2010 Second International Workshop on* (pp. 83 - 89). Monaco: IEEE.

[30] Saeed, M. (2013). Improvements to NFC Mobile Transaction and Authentication Protocol. (p. 35). IACR Cryptology ePrint Archive.

[31] E., H., Kuspriyanto, K., Basjaruddin, N., Purboyo, T., Purwantoro, S., & Ubaya, H. Efficient tag-to-tag near field communication (NFC) protocol for secure mobile payment. *Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2011 2nd International Conference* (pp. 97-101). IEEE.

[32] Francis, L., Mayes, K., Hancke, G., & Markantonakis, K. (2010, November). A location based security framework for authenticating mobile phones. In Proceedings of the 2nd International Workshop on Middleware for Pervasive Mobile and Embedded Computing (p. 5). ACM.

[33] Halevi, T., Ma, D., Saxena, N., & Xiang, T. (2012). Secure proximity detection for NFC devices based on ambient sensor data. In Computer Security–ESORICS 2012 (pp. 379-396). Springer Berlin Heidelberg.

[34] Francis, L., Hancke, G., Mayes, K., & Markantonakis, K. (2010). Practical NFC peer-to-peer relay attack using mobile phones. In Radio Frequency Identification: Security and Privacy Issues (pp. 35-49). Springer Berlin Heidelberg.

[35] Issovits, W., & Hutter, M. (2011, September). Weaknesses of the ISO/IEC 14443 protocol regarding relay attacks. In RFID-Technologies and Applications (RFID-TA), 2011 IEEE International Conference on (pp. 335-342). IEEE.

[36] van Dullink, W., & Westein, P. (2013). Remote relay attack on RFID access control systems using NFC enabled devices.

[37] Narayanaswamy, J. HIDE: Hybrid Symmetric Key Algorithm for Integrity Check, Dynamic Key Generation and Encryption.(Unpublished) International Workshop on Trustworthy Embedded Devices 2014. ACM.

[38] Google. (n.d.). *Nexus 4* . Retrieved 07 2014, from Google Play: https://play.google.com/store/devices/details/Nexus_4_16GB?id=nexus_4_16gb&hl=en

[39] Ltd., A. C. (n.d.). *ACR122U USB NFC Reader*. Retrieved 07 2014, from Advanced Card Systems Ltd.: http://www.acs.com.hk/en/products/3/acr122u-usb-nfc-reader/

[40] Eclipse. (n.d.). *eclipse*. Retrieved 07 2014, from https://www.eclipse.org/org/

[41] Google. (n.d.). *Google Android*. Retrieved 07 2014, from http://developer.android.com/sdk/index.html

[42] Apple. (n.d.). *MacBookPro*. Retrieved 07 2014, from https://www.apple.com

[43] NFCSpy. (n.d.). *NFCSpy*. Retrieved 07 2014, from Google Play: https://play.google.com/store/apps/details?id=com.sinpo.nfcspy&hl=en

[44] Cryptool. (n.d.). Retrieved 07 2014, from http://www.cryptool.org/en/

[45] Sun,H.---M.andTing,W.---C.“AGen2---BasedRFIDAuthenticationProtocolforSecurityandPrivacy” .IEEETransactionsonMobileComputing.vol.8,no.8,pp.1052,10622009