Development of A Fuel Card Application Using Basic Smart Card

Dr. Mohammed Najm Abdullah **Huda Fathil Alwan***

Received on: 24/ 5 / 2009
Accepted on: 30/ 6 /2010

Abstract

This work is devoted for smart card technology and focusing on software and security. The main differences between smart card software and Personal Computer software have been reviewed. The BasicCard is adopted to design a Fuel Card Application; Fuel Card Application deals with the two parts of smart card software namely, card side and host side. In the present technique programs for Issuing Company, Card User and Fuel Card have been developed. The threats those may attack a Fuel Card application were analyzed then solutions have been proposed to fulfill the security requirements; this is done by designing Fuel Card Application with security which has different cryptographic algorithms that use different keys in a single Fuel Card. Under ZeitControl BasicCard @Development Kit, two Fuel Card Application models, with security and without security, were developed and simulated using Professional BasicCard, memory and time of execution for the two models have been compared. It is concluded that applying security increases memory in not more than 25%, while the increase in the execution time is about 60% but this increment is insignificant relative to the inputting time.

تطوير تطبيق بطاقة الوقود بأستخدام البطاقة الذكية البسيطة

الخلاصه

ان هذا البحث مكرس لدراسه تقنية البطاقات الذكيه ، بالاخص برمجة البطاقه الذكيه و أمنيتها. فقد تم تشخيص الفروقات الرئيسيه بين برمجة البطاقات الذكيه وبرمجة الحاسوب الشخصي .أختيرتBasic Card platform كمنصة عمل هذا البحث حيث تم تصميم تطبيق بطاقة الوقود بأستخدام بطاقة Basic Card. تطبيق بطاقة الوقود يغطي برمجة البطاقه الذكيه من كل جوانب :جانب البطاقه وجانب المضيف الذي تدخل فيه البطاقه، تمت تغطية هذه الجوانب في هذا البحث من خلال تطوير ثلاث برامج هي: برنامج بطاقة الوقود ،برنامج الشركه المصدره و برنامج مستخدم البطاقه،هذه البرامج تتشارك جميعها لاتمام عمل تطبيق بطاقة الوقود. بعد تصميم تطبيق بطاقة الوقود تم تشخيص كل الاختراقات والتهديدات التي يمكن ان تهاجم هذا التطبيق واقتراح الحلول التي يمكن ان تغطي كل المتطلبات الامنيه لضمان سرية المعلومات: هذا يتطلب تطوبر بطاقة وقود اخر يحتوي على خوارزميات تشفير مختلفه تستخدم مفاتيح شفرات مختلفه تعود لنفس التطبيق. بأستخدام مجموعة برامج التطوير Basic control Zeit)

*Computer Engineering & Inf. Technology Department, University of Technology/Baghdad

https://doi.org/10.30684/etj.28.19.14
2412-0758/University of Technology-Iraq, Baghdad, Iraq

تم تطوير ومحاكاة نموذجين من تطبيق بطاقة الوقود لالول (Card@Develoment ki) بدون امن والثاني مع الامن. تمت محاكة كلا النموذجين باستخدام بطاقه من نوع Professional Basic Card مما يستنتج من هذا العمل اضافة الامن الى تطبيق بطاقة الوقود يزيد من الذاكره المستخدمه بنسبه لا تزيدعن 25% بينما الزياده في وقت الاعدام تكون بنسبه60% لكن هذه الزياده غير محسوسه مقارنة مع وقت الادخال

1. Introduction

The smart card proved to be an ideal medium. It made a high level of security (based on cryptography) available to everyone, since it could safely store secret keys and execute cryptographic algorithms. In addition, smart cards are so small and easy to handle that they can be carried and used everywhere by everybody in everyday life [1]. The development of the smart card, combined with the expansion of electronic data processing systems, has created completely new possibilities for devising a new solution. Smarts card are very useful in the area of personal security, they can be used to add authentication and secure access to information systems that require a high security [2].

1.1 Smart Card Standardizations

An important feature of smart cards is that their properties are strongly based on international standards. This is fundamentally important with regard to the usually compulsory need for interoperability. Unfortunately, these standards are often difficult to understand, and in some critical places they require outright interpretation. Sometimes only the members of the associated standardization group can explain the intention of certain sections [3].The venue of choice for establishing such standards is the International Standards Organization (ISO). In some fields of technical activity, the International Electro technical Commission (IEC) collaborates with the ISO in the development of standards [4].In this work the contact BasicCard was used so ISO 7816-4 was needed because it is used to Identify Integrated Circuit(s) Cards with Contacts.

1.2 Types of Cards

In general cards can be classified into four types according to time of appearance and the storage capacity [1]:

a. Embossed Card

Embossing is the oldest technique for adding machine-readable features to identification cards. The embossed characters on the card can be transferred to paper using simple, inexpensive devices, and they can also be easily read visually (by humans). The nature and location of the embossing are specified in the ISO 7811 standard.

Figure (1.1 in appendix A) shows the embossed card.

b. Magnetic -Stripe Card

The fundamental disadvantage of embossed cards is that their use creates a flood of paper exact shape and size of a credit card. Smart cards are particularly useful components of computer systems that need to address data security, personal privacy, and user mobility requirements. The amount of data processed by a smart card program is usually quite small and the computations performed are typically quite modest. Smart cards can be divided into two groups, which differ in both functionality and price they are: Memory cards and Microprocessor cards. Memory cards are much less expensive and much less functional than microprocessor cards. Memory card does not have the ability to process data (Memory card stores data while the Microprocessor can store and process data.) Microprocessor Card can have many functions such as storing data, make calculations, process data, manage files, and execute encryption algorithms. It makes possible sophisticated and portable data processing applications such as contact and contactless credit/debit cards, transit payment cards and GSM Subscriber Identity Modules [6].

c. Smart Card

The smart card is the youngest and cleverest member of cards. Its characteristic feature is an integrated circuit embedded in the card, which has components for transmitting, storing and processing data [1,2]. The data can be transmitted using either contact on the surface of the card or electromagnetic fields, without any contacts. The fundamental characteristics and functions of smart cards are specified in the ISO 7816.Smart cards offer several advantages compared with magnetic-stripe cards. The main important advantages are: First, the maximum storage capacity of a smart card is many times greater than that of a magnetic-stripe card. Second, of the most important advantages of smart cards is that their stored data can be protected against unauthorized access and manipulation [1]. Since the data can only be accessed via a serial interface that is controlled by an operating system and security logic, confidential data can be written to the card and stored in a manner that prevents them from ever being read from outside the card. Such confidential data can be processed only internally by the chip's processing unit. Smart cards can be divided into two groups, which differ in both functionality and price they are: *Memory cards* and *Microprocessor cards.* The main difference between Memory card and Microprocessor card is how to process data (Memory card stores data while the Microprocessor can store and

process data). Figure (1.3 in appendix A) shows the Memory card and figure (1.4 in appendix A) shows the Microprocessor card [5].

d. Optical Memory Card

For applications where the storage capacity of smart cards is insufficient, optical cards that can store several megabytes of data are available. However, with current technology these cards can be written only once and cannot be erased. The ISO/IEC 11693 and 11694 standards define the physical characteristics of optical memory cards and the linear data-recording technique used with such cards [1].Up to now, use of optical memory cards has been severely limited by the high cost of equipment for reading and writing this type of card. One application for optical memory cards is recording patient data in the medical sector, since their large storage capacity allows even X-ray images to be stored on a card.

2. Smart Card Technology

A smart card is a portable, tamper-resistant computer with a programmable data store. Its characteristic feature is an integrated circuit embedded in the card, which has components for transmitting, storing and processing data. It is the exact shape and size of a credit card. Smart cards are particularly useful components of computer systems that need to address data security, personal privacy, and user mobility requirements. The amount of data processed by a smart card program is usually quite small and the computations performed are typically quite modest.

Smart cards can be divided into two groups, which differ in both functionality and price they are: Memory cards and Microprocessor cards. Memory cards are much less expensive and much less functional than microprocessor cards. Memory card does not have the ability to process data (Memory card stores data while the Microprocessor can store and process data.) Microprocessor Card can have many functions such as storing data, make calculations, process data, manage files, and execute encryption algorithms. It makes possible sophisticated and portable data processing applications such as contact and contactless credit/debit cards, transit payment cards and GSM Subscriber Identity Modules [6].

2.1 Physical Properties of the Smart Card

Small card has a small gold chip about ½"in diameter on the front , the typical smart card dimensions of 85.6 mm by 54 mm have been in use for a very long time as shown in figure (2.1 in appendix A). It is designated ID-1, and its size is specified in the ISO 7810 standard. Most smart cards have eight contact fields on the front face these are:-Vcc Supply voltage (specified at 5 volts ±10 percent), RST Reset, CLK Clock frequency,

RFU Reserved for future use, GND Ground, Vpp External programming voltage (to supply voltage to EEPROMs for programming and erasing), I/O Serial input/output communications[1,4]. Most of the physical characteristics are actually purely mechanical in nature, such as the format of the card and its resistance to bending or twisting, electromagnetic susceptibility, electromagnetic discharges and (UV radiation, X-ray radiation, moisture and temperature) resistance. The computer on a smart card is a single integrated circuit chip that includes CPU, the memory system, and the input/output lines. Figure (2.2 in appendix A) shows the hardware elements of a smart card computer system. The CPU in a smart card chip is an 8-bit microcontroller, typically using the Motorola 6805 or Intel 8051. Hitachi's H8 smart card chip is a notable exception. Some smart card chips include coprocessors to accelerate specifically the computations done in strong encryption [7, 8].

2.2 Smart Card Hardware

There are, in fact, three kinds of memory on a smart card:-Read-only memory (ROM) is where the smart card operating system is stored. Code and data are placed in read-only memory when the card is manufactured and cannot be changed, Nonvolatile memory (NVM) is where the card's variable data is stored. NVM can be read and written by application programs and a relatively tiny amount of random access memory (RAM) the CPU uses RAM as its working memory. The input/output channel on a smart card is a unidirectional serial channel. Even if there are several logical channels available just one of them can be active at a time. Smart card operating systems support either character or block transfers, but usually this level of detail is hidden from the smart card programmer.

2.3 Smart Card Interface Devices

A smart cart does not include power source or clock signals, As a result the smart card must be inserted in the device that provides both power and clock signals. This device may be known as Interface Device (IFD), Card Acceptance Device (CAD), terminal or reader. Smart cards can be categorized to [9]:-

a. Contact smart card (it must be inserted into a smart card reader with a direct connection to a conductive contact plate on the surface of the card)
b. Contactless smart card (it requires only close proximity to a reader. Both the reader and the card have antennae, and the two communicate using radio frequencies (RF) over this contactless link)

c. Hybrid smart card (it has two chips, one with a contact interface and one with a contactless interface. The two chips are not interconnected)
d. Dual-interface smart (it has a single chip with both contact and contactless interfaces). It is also called 'combicards

2.4 Smart Card Operating System

In contrast to personal computer operating systems such as Unix, DOS, and Windows, smart card operating systems do not feature user interfaces or the ability to access external peripherals or storage media because they are optimized for completely different functionality. Security during program execution and protected access to data has the highest priority. Smart card operating systems have a very small amount of program code, which normally lies in the range of 3–250 kB. The primary tasks of a smart card operating are [5]:-

a. Transferring data to and from the smart card.
b. Controlling the execution of commands.
c. Managing files.
d. Managing and executing cryptographic algorithms.
e. Managing and executing program code.

2.5 Smart Card Communication

The communication protocol between the host and the smart card is based on a master slave relationship. The host sends commands to the card and listens for a reply. The card receives the command from the reader, executes it and responds to the card reader by sending a response. Through the answer-to-reset (ATR) mechanism, a basic communication pathway is established between a reader and a smart card. This communication pathway is a half-duplex physical channel. The input/output channel on a smart card is a unidirectional serial channel [10].

2.6 Smart Card Data Transmission Protocols

Data transfer (defined in the ISO/IEC 7816 specification) between the card reader and the card takes place on a single line, The standards allow for the specification of 15 or more distinct data transmission protocols in smart card system, but actually there are two primary variants of this transmission protocol is used with smart card systems: the T=0 protocol and the T=1 protocol [11]. The main differences between T=0 and T=1 smart card protocols are that: The T=1 protocol fits the OSI Reference Model fairly well as a "data link" (or link-level) protocol layer, but the T=0 protocol tends to mix elements from several different protocol layers (as defined by the OSI Reference Model). The reader and the card communicate through a very strict command-and-

response protocol. In the present project T=0 protocol has been used.

2.7 Messages in Smart Card System

A protocol message structure is defined in ISO/IEC 7816-4, through which the function calls, their associated parameters, and status response parameters are exchanged between the reader-side application and the card-side application code. This message structure is characterized by application protocol data units (APDUs). APDU commands are always sets of pairs. Each pair contains a Command-*APDU*, which specifies a command, and a Response-*APDU*, which sends back the execution result of the command. The structure of APDU command comprises a mandatory *header* and a *optional body,* each of which is further subdivided into several fields as can be seen in figure (2.3 in appendix A) [10, 11].The mandatory header includes CLA, INS, P1, and P2 fields, each command must have all header fields.

- CLA and INS define an application class and instruction group.
- The P1 and P2 fields are used to qualify specific instructions and are therefore given specific definitions by each [CLA,INS] instruction. The body of the APDU command includes the Lc, Data field, Le fields:-
- The Lc field specifies the number of bytes to be transferred to the card as part of an instruction; it contains the length of the data field.
- The data field comprises data which must be conveyed to the card in order to allow its APDU processor to execute the command specified in the APDU.
- The Le field specifies the number of bytes which will be returned to the reader by the card's APDU processor in the response APDU.The body of the APDU command has a variable size and form (components), therefore we called as optional, optional body is used to convey information to the card's APDU processor as part of a command. It can have four different forms [5,11]:-

a. No data is transferred to or from the card, so the APDU command includes only the header component.
b. No data is transferred to the card, but data is returned from the card so the body of the APDU command includes only a non null Le field.

c. Data is transferred to the card, but no data is returned from the card as a result of the command so the body of the APDU includes the Lc field and the data field.
d. Data is transferred to the card and data is returned from the card as a result of the command so the body of the APDU includes the Lc field, the data field, and the Le field..

The structure of APDU response which is sent by the card in reply to the command APDU consists of an optional body and a mandatory trailer as shown in figure (2.4 in appendix A).The body consists of the data field whose length is specified by the Le byte of the preceding command. If there is an error in the executing of the command the data field is set to zero and the error is specified in the in the two single byte status word SW1 and SW2 in the trailer.

3. Smart Card Software

There are fundamentally two types of smart card software:- *Host software* is software that runs on a computer connected to a smart card and *Card software* is the software that runs on the smart card itself. Software in smart cards is inherently harder than software in a desktop environment for several reasons [12]:-

a. They lack natural input and output. Lack of proper input and output means that other systems such as PC's are needed as essential parts of the development.
b. Their processor and memory capacities are limited. In order to make the best possible use of the available memory space, the program code must be adapted to the specific type of chip that is used.
c. The standards are few and not very well followed by the industry.
d. The development environments and languages for the cards have been archaic.
e. Smart cards are often used as trusted storage and data processing systems to store cryptographic private keys and other valuable information. This means that they are usually used as part of larger access control or authorization architecture.
f. More skills and tools are required to get the actual job done.

3.1 Traditional Smart Card Software

Until the advent of the Open Card Platform, the only way to get software onto a smart card was to have it written and loaded into a smart card by a smart card manufacturer. This was a long, tedious, and surprisingly error-prone process. It was also a very expensive process and precluded all but the largest organizations

from creating purpose-built smart cards. Consequently, the development of software for smart cards has tended in the direction of highly specialized software for each new application on a chip. In the mid-1990s, work began on developing specifications for terminal-independent integration of smart cards into PC programs. Internationally, two industrial standards have come to prevail: Personal Computer / Smart Card (PC/SC) for terminal independent and Open Card Framework (OCF) for independent smart card program [7, 9].

3.2 Open Card Platform

Generally the term 'open platform' refers to smart card operating systems that allow third parties to load applications onto smart cards independent of the card manufacturer. In general open platforms specifications are public and are created by a consortium of companies. The benefit of an open application environment is that it speeds up the development process of smart card applications. And this leads to a reduction of development costs. Furthermore, it increases the flexibility of the smart card application development process. Several open platforms for smart cards are available:

- Java Card
- Multos
- Basic Card
- Windows for Smart Cards and Linux

In this project Java Card has been tried first, many differences with Basic Card had been evaluated; because of these differences Basic Card platform has used in this project. These differences are listed below:-

a. The Java Card platform supports only a carefully chosen, customized subset of the features of the Java language. The ZC-Basic programming language is fully functional, it has some special features make it more suited for the SC environment, BasicCard programming is easier than JC)
b. Java compiler do not cover the memory constrains of the SC system (java complier is less effective than Basic Card compiler). Basic Card has a specialized compiler cover storage requirements of the whole program , So compilation consist of a single compilation unit – there is no linking stage.
c. Two compilation phases required in Java Card. Basic Card required one Compilation phase.
d. Java code compiled into a virtual machine language known as P-Code(P-code is a machine independent language). Basic Card uses the same technology But P-code in the basic card is

differing from P-Code in java.

e. Java Card Development kit runs under DOS.Basic Card Development Kit runs under Windows Basic Card simulation interface easier than Java Card.
f. Java Card Development kit Simulate Card only, while Basic Card Simulate Card & Terminal.

3.3 Basic Card

BasicCard is a processor card with a P-code interpreter written in Basic. It is as a complete miniature computer, it contains a CPU, (16-64) KB of ROM, (8-32) KB of EEPROM, and (256-2048) bytes of RAM. It was designed with four criteria:An inexpensive solution for smart card software developers (the development software is free of charge), Easy to program, provide a secure developing environment (RSA, AES, and SHA-1) and it is designed according to the ISO/IEC 7816 specifications. Basic Cards can be classified into two types:-

a. Single-application BasicCards can contain single application. There are three types of Single-application:-Compact, Enhanced and Professional BasicCard (Professional BasicCard used in this project).
b. MultiApplication BasicCard enables multiple Applications to be loaded into a single Card without compromising each other's security.

4. Security in Smart Card System

In general the smart card security can be classified into two categories: Smart Card Hardware Security and Smart Card Software Security. Smart Card manufactures are responsible for increasing the smart card hardware robustness by using high quality material in the manufacturing of smart card components in order to increase the resistance of the smart card. Smart Card Software Security deals with all cryptographic techniques compatible with smart card system, all these techniques are used to maintain data security and data integrity of the data stored in the smart card. Data security means simply providing data only to those who are authorized to receive it. Wherever data is stored and whenever data is moved, the smart card programmer must ensure that this requirement is satisfied. Data integrity requires that all parties to a smart card transaction agree on the state of the data in the transaction [3]. In general smart card can protect its data using one or more of the following methods:**1)**Encryption/Decryption via cryptography algorithms making read and write key dependent**2)**Authentication the card and the terminal communicate with the card. By generating a challenge using with a secret key under specific cryptography algorithm.**3)**Using the assets

method such as PIN, Retry counter and Transaction Authentication Code Key.

4.1 Encryption in Smart Card System

Encryption/Decryption are used in smart card system in simplified terms, in terminal side the APDU command is encrypted using a specific Key under a specific Encryption algorithm then it is sent to the card. In the card side, the card operating system must decrypt the APDU command using the same Key under the same Decryption algorithm if one of these(Keys or algorithms) is incorrect the card cannot decrypt the APDU command. Smart card programmers typically do not have to design new authentication or encryption algorithms. Rather, they use the facilities that are built into the smart card. These facilities have been field tested and come with a certain level of assurance of correctness. Designing new algorithms is not easy, and validating the correctness of a new algorithm is probably not a subtask that a smart card application developer wants to assume.

4.2 Authentication in Smart Card System

Before a smart card can provide access to its resources, it must determine with whom it is dealing. Similarly, before it is accepted by other entities, it must be able to prove who it is. Therefore, one of the first tasks a smart card performs when it is activated is to authenticate entities outside itself, primarily the person who inserted it into the terminal and the terminal into which it has been inserted, and to authenticate itself to some or all of these entities. Figure (4.1) illustrates the classification of authentication procedures in smart card system.

4.2.1 Unilateral Authentication

A unilateral authentication serves to assure one party of the trustworthiness of the other party to a communication. For it to be possible, both parties must have a shared secret, the knowledge of which is verified by the authentication procedure. The principle of unilateral authentication with a symmetric cryptographic algorithm is illustrated as follow:-1) The terminal generates a random number and sends it to the smart card (This is the challenge). 2) The smart card encrypts the random number it receives, using a key known to both the card and the terminal. 3) The card then returns the result of the encryption to the terminal (This is the response to the challenge). 4) The terminal uses the secret key to decrypt the encrypted random number it has received, and then compares the result with the random number it originally sent. If the two numbers match, the terminal knows that the smart card is authentic.

4.2.2 Challenge Response Mutual Authentication Procedure

The principle of mutual authentication is based on dual unilateral authentication. In principle, two successive unilateral authentications could also be used, one for each of the communicating parties, in order to achieve mutual authentication. The principle of a mutual authentication with a symmetric cryptographic algorithm is illustrated below:-

a. The terminal generates a random number and sends it to the smart card.
b. The smart card encrypts the random number it receives, using a key known to both the card and the terminal.
c. The card then returns the result of the encryption to the terminal.
d. The terminal uses the secret key to decrypt the encrypted random number it has received, and then compares the result with the random number it originally sent. If the two numbers are matched, the terminal knows that the smart card is authentic.

By executing the above steps the card authenticate itself to the terminal, and then the terminal starts to authenticate itself to the card:-

a. The terminal asks the card to generate a random number the card generates a random number and sends it to the terminal in the response.
b. The terminal encrypts the random number it receives, using a key known to both the card and the terminal.
c. The terminal sends the result of encryption as a command to the card.
d. The card decrypts the encrypted random number it has received using the secret key known to both the card and the terminal, and then compares the result with the random number it originally sent. If the two numbers are matched, then the smart card knows that the terminal is authentic.

5. Designing Fuel Card Application

Regardless of the type of software being written, the smart card programmer must be constantly aware of the two central concerns of smart card software: data security and data integrity. As with any software development, before sitting down and writing programs, the design phase will be gone first. From the design phase of Basic Card application the architecture of Basic Card application becomes clear and easy to understand.

Figure (5.1 in appendix A) shows the procedure of design Basic Card application. The main steps to design are:-

a. Specifying the user requirement.

b. Specifying the programs for BasicCard application.
c. Defining the APDU commands and response, those are required to achieve application functions.
d. Defining flowcharts for BasicCard program and Terminal programs.
e. Writing programs code.

Fuel Card applications with and without security have been designed under these steps. First the Fuel Card application without security has been designed then Fuel Card application with suggested security solutions has been designed it was called Fuel Card application with security.

5.1 User Requirements of a Fuel Card Application

Fuel card stores electronic information and makes use of this information to reduce traffic and robbery. This information can be classified into:-

- Car Identification Information used to identify the car which is the card relative to such as car model, car number and car type.
- User Identification Information, this identifies car user who is the same card user such as user name, master PIN and user PIN.
- Storage of electronic fuel, allows the user to debit from this fuel.
- This card also stores expiry date.

This information is written in the card by the Issuing Company (government) that issues the card. The Car Identification Information is written in the card for the first time when the card is initialized and cannot update it again, While the other Information can be updated again by Issuing Company. For every car there is only one card relative to this car and only one user uses both card and car. The card holder must visit the Issuing Company every year to update it and refill the card with the electronic fuel if the user is late the card will expire and cannot be used again. This feature enables Issuing Company to authorize the car holder.

A worker in the fuel station must type Car Identification Information in keypad connected to the CAD. The card will receive this information then compares it with Car Identification Information stored in the card; if input information is incorrect the card is locked. The user of the card must type his master PIN and user PIN in keypad connected to the CAD. The card will receive this information then compares it with master PIN and user PIN stored in the card if two input information is incorrect the card is locked.

There are two types of cars public and private, the card related to the public car cannot be restricted by

any restrictions except expiry date. While the card related to the private car is restricted by the maximum fuel dispensed quantity, the amount of the fuel transacted in one use and the number of card use. These restrictions are made in order to reduce the traffic and the fuel dispensary. According to user requirement the Fuel Card Application must have the following functions:-

a. For the first time running Car Identification Information must be initialized first then User Identification Information can be personalized and expiry date can be updated.
b. If the card has been initialized expire date must be checked first. Then the card can be re-personalized. If the card is expired it cannot be re-personalized or updated.
c. The card must be re-personalized then expiry date can be updated.
d. Car Identification Information must be verified then master PIN and user PIN must be verified.
e. The card cannot be used before the verification is made.
f. The card is locked after one unsuccessful attempt of car verification and the card is locked after five unsuccessful attempts of user PIN verification.
g. For the card related to the private car the maximum card use is 50 for every year.
h. For the card related to the private car the maximum fuel dispensed quantity is 500L for one year and no fuel transaction can exceed 60 L for one use.

5.2 Programs for the Fuel Card Application

According to the FCA functions three programs re needed these are:-

a. Two Terminal Programs

- The first program is responsible for issuing fuel card to the users and writing information in the card. This program is used by the issuing company or financial institution. It is named "Issuing Company" program.
- The second program is used to the information stored in the card and managing the use of the electronic fuel stored in the card. This program is used by the user of the card. It is named "Card User" program.

b.One Basic Card Program

- This program is stored in the smart card and contains a series of commands, which is called by the terminal side (Issuing Company/ Card User). It is named "Fuel Card" program. These programs must have the file structure compatible with the files structures in the ZC-Basic language The BasicCard® Development Kit as shown in figure (5.2 in appendix A).

5.3 The Application Protocol Data Unit Commands and Response Required to Achieve Fuel Card Application Functions

Basic Card program running in a smart card communicates with the terminal program at the CAD using application protocol data units (APDU).

The terminal program sends a C-APDU through the CAD. The Basic Card OS receives the command and passes it to the Basic Card program. The Basic Card program processes the command and returns a response APDU to the terminal program. The Basic Card developers must define sets of Command and Response pairs in accordance with the functions of the application. Using these definitions, the Basic Card compiler knows how to interpret each command and read the data. For each response the developer should define a set of status words to indicate the result of processing the paired- CAPDU, if the optional data field is included in the response APDU, the developer should define what to return. the structure of CAPDU-RAPDU must comply with the structure of ISO/IEC 7816-4 APDU. Basic Card technology has a set of the predefined Command and Response. Developers are free to define their own commands but they must prevent the coding of the predefined commands. Get Application ID command is a predefined command used to (to get the application ID string). Fourteen Command-Response Pairs have been developed in this project:

a. Check Expire Date C-R

This command is called by Issuing Company program. It receives the current date as input from (I.C.P). First it checks whether the card is initialized previously or not. If it not yet initialized the command generates and sends SW1SW2= Card not initialized then exits. Otherwise it continues to compare the current date with expire date stored in the card if the current date < expiry date then it sends SW1SW2= sw Card cannot Depersonalize in time or if the current year > the year

in expiry date then it sends SW1SW2=sw Card is expired. Otherwise the command generates and sends SW1SW2= 9000 to indicate to the terminal (Issuing Company program) that the command is executed successfully. The structure of this ommand and Response are specified in table (5.1 in appendix B).

b. Initialized the Card C-R

This command is called by Issuing Company program. It receives Car dentification Information, then it stores this information in EEPROM and sets initialization flag to true. The command generates and sends SW1SW2=9000 in response to indicate successfully processing. The structure of this command and Response are specified in table (5.1 in appendix B).

c. Personalize the Card C-R

This command is called by Issuing Company program. It stores User name, initial dispensed quantity, Max.fuel balance and Max. Transaction in EEPROM and sets Personalization flag to true, then it generates and sends SW1SW2=9000 in response to indicate successfully processing. The structure of this command and Response are specified in table (5.1 in appendix B).

d. Update MasterPIN C-R

This command is called by Issuing Company program. It stores Master PIN in EEPROM, then it generates and sends SW1SW2=9000 in response to indicate successfully processing. The structure of this command and Response are specified in table (5.1 in appendix B).

e. Update User PIN C-R

This command is called by Issuing Company program. It stores the User PIN in EEPROM, then it generates and sends SW1SW2=9000 in response to indicate successfully processing. The structure of this command and Response are specified in table (5.1 in appendix B).

f. Get Initialization data C-R

This command is called by Issuing Company program. It retrieves the initialization information from EEPROM, then it sends this information in response. The structure of this command and Response are specified in table (5.1 in appendix B).

g. Update Expire Date C-R

This command is called by Issuing Company program.

It controls the updating of Expire Date. The structure of this command and Response are specified in table (5.1 in appendix B).

h. Get Expire Date C-R

This command is called by the Card User program. It retrieves the Expire Dateinformation from EEPROM then it sends this information in response. The tructure of this command and Response are specified in table (5.1in appendix B).

i. Verify Car info C-R

This command is called by the Card User program. It receives the Car Information, it checked the entered Information with the Car Information stored in the EEPROM if they are matched it sets verify car flag, otherwise it exit. The structure of this command and Response are specified in table (5.1 in appendix B).

j. Verify Master PIN C-R

This command is called by the Card User program. It checked the input Master PIN with Master PIN stored in EEPROM if they are matched it sets Master PIN flag, otherwise it exit. The structure of this command and Response are specified in table (5.1 in appendix B).

k. Verify User PIN C-R

This command is called by the Card User program. It verified the entered user PIN if it is true it sets User PIN flag, otherwise it exit. The structure of this command and Response are specified in table (5.1).

l. Get Personalization Data C-R

This command is called by the Card User program. It retrieves the Username, Remaining Balance info from EEPROM. The structure of this command and Response are specified in table (5.1 in appendix B).

m.Get the Remaining Use C-R

This command is called by the Card User program. It retrieves the remaining use information from EEPROM then it sends this information in response. The structure of this command and Response are specified in table (5.1 in appendix B).

n. Use Card command C-R

This command is called by the Card User program. It manages the use of the electronic fuel stored in the card. It receives the entered amount code as input from Card User program. This command starts operation by verifying initialization

flag, personalization flag, verify car flag, verify Master PIN flag and verify User PIN flag, if any one of these conditions is false it generates and sends SW1SW2 relates to this state and exit the command. If all of these are true it begins to check the car type, if it is public then the equation is executed:(Balance = Balance + amount)and sends the SW1SW2= sw PUBLIC car and exits. Otherwise it compares the input a mount with MaxTranactionAmount If (input amount > MaxTranactionAmount) then it generates and sends SW1SW2= swInvalidTransaction and exits. Otherwise the command checks the following equation If (Balance + amount > MaxBalance) is true then the command generates and sends SW1SW2= swExeecedMaxBalance and exits. Else it checks whether (Card Use < Maximum Card Use) if it is not the command generates and sends SW1SW2= ExeecedMaxUSE and exits, otherwise the following equations are executed: Balance = Balance + amount, Card Use = Card Use +1. Then the command generates and sends SW1SW2=9000 and exits. The structure of this command and Response are specified in table (5.1 in appendix B).

5.4 Fuel Card Application Development Process

Development of Basic Card Application begins as with any other software application. The main development process steps of Basic Card Applications are:-

a. A developer writes one BasicCard program and one or more terminal programs.
b. These programs are compiled using the compiler of Ziet' Control software development package *(ZC-Basic Compiler ZCMBASIC.EXE)* The compiler ZCMBASIC compiles ZC-Basic source code into P-Code, an intermediate language that can be thought of as the machine code for a Virtual Machine.
c. The application programs are tested, run and debugged using the *ZCMDTERM Terminal Program Debugger* and the *ZCMDCARD BasicCard Program Debugger.*
d. Finally after the application is tested, it becomes ready to be downloaded to the card using the *Card Loader*

BCLOAD.EXE. The P-Code is downloaded to the card using the BCLOAD Card Loader program. Then the Virtual Machine in the BasicCard executes the P-Code instructions at run-time. Fuel Card application without security has been developed under these steps.

6. The Threats Attack the Fuel Card Application

The threats are based on the following definition of an attacker: An attacker is a person who is trying to access sensitive information [7]. A large number of threats may attack the Fuel Card Application (FCA) during life cycle:

a. **Threat Attack Card_Issuing**
If an unauthorized programmer has information about the application procedure and the application does not have a specific key. This programmer will be able to developing, issuing and buying illegal cards.

b. **Threat.Hacker_PHYS**
Physical attack through the FCA interfaces. An attacker may obtain knowledge of cryptographic keys via physical attacks such as probing.

c. **Threat Attack Date Security and Integrity**
An attacker may change security data in the storage of the FCA. This happens when the application does not have a specific protocol to communicate with authorized parties.

d. **Threat Attack KEY_COMPROMISE**
If the FCA has a key an attacker may try to compromise the secret cryptographic key of the FCA. An attacker may try to perform this attack during the usage phase of the FCA or during the key update process.

7. Five Proposed Solutions to Fulfill the Fuel Card with Security Requirements

Some solutions have been proposed to fill the FCA with a full security in order to protect the FCA from an attacker. Five combined solutions have been suggested, these are:

a. Making the entire security of the card based on the secrecy of the key, used in this card. All cryptographic keys which are created in the environment to be usedwithin the FCA have to be created and handled in a secure manner and have to have sufficient quality. This solution can be done by keeping these cryptographic keys invisible to the user. This solution protect the FCA

from the *Threat Attack KEY_COMPROMISE*

b. Keeping the application procedure in secret, so that an attacker does not know how the system works. Development and testing of the FCA is protected in a secure environment for its integrity and confidentiality and done by trustworthy person. This solution protects the FCA from the Threat Attack Card Issuing
c. The communication between the card and terminal is done under a secure channel. This solution protects the FCA from the Threat.Hacker_PHYS
d. The initialization and personalization process must take place in an environment providing physical security and performed by trustworthy personnel. This solution protects the FCA from the Threat Attack Date Security and Integrity.
e. Implementing different cryptographic algorithms that use different keys in a single card. This solution is done by :A) Applying DES cryptographic algorithm, this solution protects the FCA from the Threat Attack Card Issuing (because each card has its own key which differs from the other cards and the card cannot be initialized or personalized without this key. This key is stored in the Issuing Company and the card) and the Threat Attack Threat.Hacker_PHYS (because the command is encrypted before it has beensent). B) Applying challenge response mutual authentication with AES cryptographic. This solution protects the FCA from the Threat Attack Date Security and Integrity (a smart card cannot provide access to its data unless it authenticates with whom it communicates).

7.1 Adding and Implementing DES Encryption Algorithm in a Fuel Card Application

Adding DES encryption algorithm in a Fuel Card Application aims to protect the FAC from the Threat Attack Card_Issuing and Threat.Hacker_PHYS. Therefore the DES encryption was added to Issuing Company program and Fuel Card program only; there is no need to add it to the Card User program. Within Fuel Card Application, Single-Application BasicCard has been used. To implement the encryption mechanism for single-application BasicCard, the following steps are to be followed:

a. Use the KEYGEN program to generate a key file,

containing cryptographic keys.

b. Include the generated key file in both the Terminal program and the BasicCard program.
c. Include the file COMMANDS.DEF in the Terminal program and BasicCard program, to define the StartEncryption, and End Encryption commands.
d. In the Terminal program, turn automatic encryption on and off.

7.2 Adding Challenge Response Mutual Authentication to the Fuel Card Application

The main purpose of adding Challenge Response Mutual Authentication to the FAC is to protect the FAC from the *Threat Attack Date Security and Integrity*. Since the Issuing Company program is used to initiate and issue the card and this is done by the Issuing Company which produced the card to market, Mutual Authentication must be added to the Card User Program and Fuel Card program only. The Card User Program is used to read and alter the information stored in the card. Adding Chellenge_Response Mutual Authentication to the Card User program leads to make the reading and altering of the FCA information impossible, unlessboth card and terminal authenticate each other. To add Challenge _Response Mutual Authentication in the Fuel Card, the following steps must be done:-

a. Design the Challenge_Response Mutual Authentication procedure in a form of command and response between Card User program and the Fuel Card program.
b. Select a symmetric cryptographic algorithm to do the encryption and decryption of the mutual authentication, this algorithm must have implementation in the ZC BasicCard except the DES algorithm.

The AES algorithm have been selected because it provides a high security and the ZC BasicCard supports implementation for this algorithm. Generate a Key which is required in the cryptographic algorithm; include this key in the Card User program and the Fuel Card program. The AES algorithm needs a key with (128 bits) length, so we must generate a key with 16 (128 bits) bytes (128 bits) length. All cards of the FCA have the same key if this key becomes known Challenge_Response Mutual Authentication will lose its security feature, therefore we must do our best to keep this key secure. After adding security the Fuel Card Application programs must

have the file structure as shown in figure (7.1 in appendix A).

8. Simulation and Results

The BasicCard® Development Kit has been selected as the development environments for this project; because it is very popular, ease of programming also this kit is free of charge on the Internet. There are no special hardware requirements for developing programs within the BasicCard® Development Kit; the support software can simulate the BasicCard programs in any PC, the compilation and test software can be executed in any Windows® 98/NT/2000/XP system. The BasicCard® Development kit includes a card reader, several BasicCards, and the entire necessary software package called the ZeitControl MultiDebugger software package. With this support package, the software can be tested even a card reader is not present. The package contains a fully functional Multiple Debugger, which can run Terminal and BasicCard programs simultaneously. So the idea of a smart card application can be tested without paying a cent. Simulation in the ZeitControl MultiDebugger passes through three phases:

a. Preparation phase.
b. Compilation phase.
c. Debugging phase.

During the development of this project, two fuel card models have been developed the first one without security and the second with security. From the compilation phase the memory space that is required for each program in one model can be excluded then the comparison in memory space between the two models (FuelCard without security and FuelCard with security) can be made. Table (8.1 in appendix B) shows the memory space dispensed in the FuelCard without security model and table (8.2 in appendix B) shows the memory space dispensed in the Fuel Card with security model. After doing the compilation for the two models, the debugging process can be done. In the debugging process The ZCMDTERM and ZCMDCARD must be run at the same time to see the communication between terminal program and the BasicCard program, the command is sent from the terminal to the BasicCard program by using the Step to card Button, and then the command is processed in the BasicCard program, after processing the command the response is sent to the terminal program by using Step to terminal Button. The communication result is appeared in the console window; the process (Step to card Button and Step to terminal Button) must be repeated until the terminal program end. The debugging is applied to the two models (FuelCard without security and FuelCard with security).

8.1 Memory Comparison

From table (8.1 in appendix B) Fuel Card Program has been utilized (before adding the proposed security) 1284 bytes of EEPROM of the Professional BasicCard; while from table (8.2 in appendix B) Fuel Card Program has been utilized (after adding the proposed security) 1615 bytes of EEPROM of the Professional BasicCard. The increment in memory consuming is not considerable since the proposed security which has been added is based on implementing combined simple cryptographic algorithms not on implementing hard and large memory consuming cryptographic algorithm. Memory is the most important factor in the smart card system.

8.2 Time Evolution

In order to evaluate the increase in the execution time of the Fuel Card Application after adding the security , the Time Functions supplied by ZeitControl MultiDebugger software package have been used, the function of :Function TimeInterval in conjunction with subroutine: Sub GetDateTime, is to be programmed to calculate the time during the execution. The following preparations have been included to satisfy the time calculations:-

a. The file MISC.DEF must be included; It contains the predefined timing function.
b. All external inputs must be replaced by fixed inputs, this step is necessary so as not to interfere with time consumed during user input.
c. In the debugging process, the Step to Card and Step to Terminal buttons are not to be used, instead the Run button is used to provide auto run. The returned time is in milliseconds, the results of execution are listed in table (8.3 in appendix B). It would worth noting that the listed time is the minimum time among 10 runs, the time in each run may differ due to software and hardware interrupts.

9. Conclusions

Software design of smart card is inherently harder than software design in desktop environment. Smart card software, by its nature, is distributed. The heart of the application design process is to distribute system functionality among the multiple computers that deal with a smart card and decide what functions go on the smart card itself and what functions go on the terminal. The ZeitControl BasicCard @ Development Kit is successfully utilized to simulate the Fuel Card application models using Professional BasicCard (microprocessor single application contact card). The developed software in conjunction

with the Professional BasicCard would have a practical governmental implementation. Fuel Card Application threats can be treated by combining the two cryptographic procedures (encryption and authentication) already facilitated in BasicCard, therefore there is no need to design new and hard cryptographic procedure that requires high memory, execution time and skills. Results show the increment in the storage between the two models is only 25% and the amount of time increment for the two cases is less than 60% , this increment is about 50 msec which is inconsiderable time compared with the inputting time.

10. References

[1] S. Petri ,"An Introduction to smart Card", A Column in Secure Service Provider, 1997. http://www.sspsolutions.com/sloutuon/whitepaper/introduction_to_smartcard.

[2] G. Pittaway, "Smart Card Backgrounder", part of a series of white papers known as "The Smart Card Deployment Cookbook", Microsoft Enterprise Services, March 2001. http://www.microsoft.com/es/.

[3] B. Guthery and M. Jurgensen, Smart Card Developer's Kit, Macmillan Computer Publishing, December 2002, [4] C. Enrique Ortiz, "An introduction to Java Card Technology- Part 1 ", A column in Sun developer series, May 2003.http://developers.sun.com/techtopics/ mobility/javacard/articles/javacard1

[5] W. Rankl and W. Effing, Smart Card Handbook Third Edition, Giesecke & Devrient GmbH, Munich, *Germany*, 2003. [6] B. Millier, "BasicCards 101", white paper published by CIRCUIT CELLAR®, March 2004. www.circuitcellar.com.

[7] F. Eisl, "Smart Card Security Services for an Open Application Environment used in Mobile Phones", Master thesis, Lund University, Department of Information Technology, June 2004.

[8] T. Kiliçli, "Smart Card HOWTO ", white paper, CIRCUIT CELLAR®, 2001. tolga@deepnight.org [9] N. Shillington and T. Waker , The design of the smart Card Interface Device, Data Network Architectures Laboratory,1998.

[10] Z. Chen, "How to Write Java Card applet: A Developer's Guide", A column in java developer series, 1999.

[11] T. Guilfoyle, "The ZeitControl BasicCard Family", Copyright© ZeitControl card systems GmbH, March 2005.

[12] T. Elo, "Lessons learned on implementing ECDSA on a Java smart card", Department of Computer Science, Helsinki University of Technology, 2000. tommi.elo@hut.fi.

Appendix A

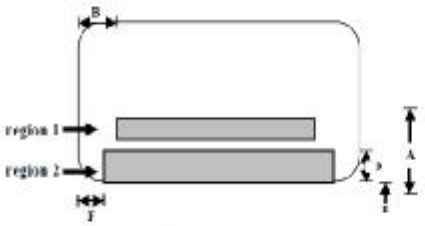


Figure (1.1): Embossed card

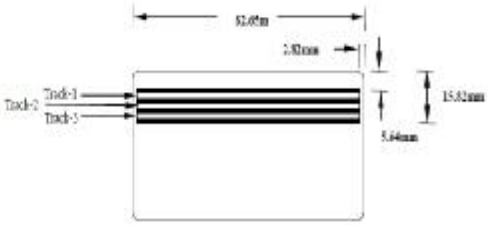


Figure (1.2) Magnetic stripe card

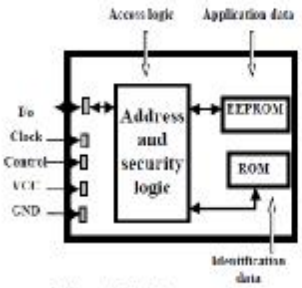


Figure (1.3): Memory card

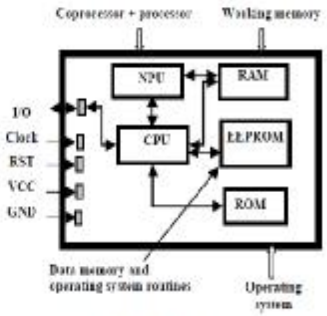


Figure (1.4): Microprocessor card

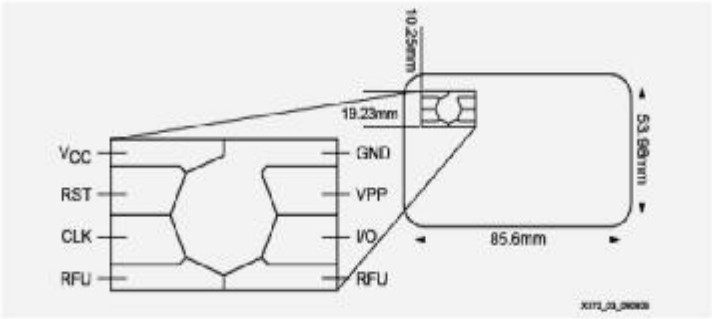


Figure (2.1): Smart card image and contact

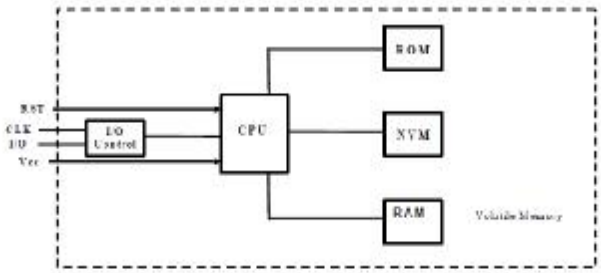


Figure (2.2): Elements of a smart card computer system [7]

Mandatory header				Optional body		
CLA	INS	P1	P2	Lc	Data field	Le

Figure (2.3): The format of the APDU command [10]

Optional body	Mandatory trailer	
Data field	**SW1**	**SW2**
• Data field (variable length): A sequence of bytes received in the data field of the response • SW1 (1 byte) and SW2 (1 byte): Status words denote the processing state in the card		

Figure (2.4): The structure of the APDU response [10]

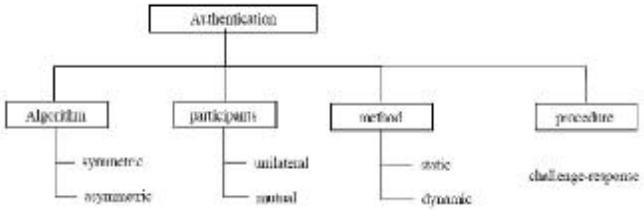


Figure (4.1): The classification of authentication procedures in smart card

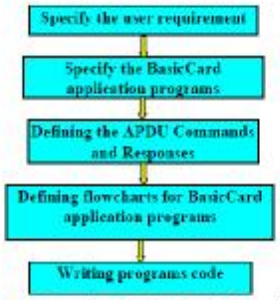
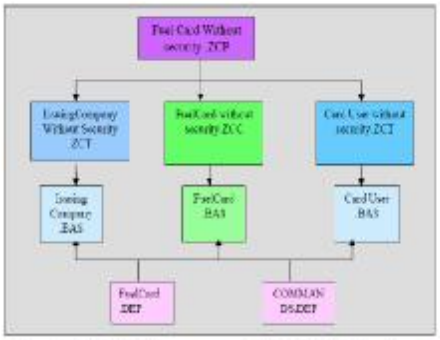
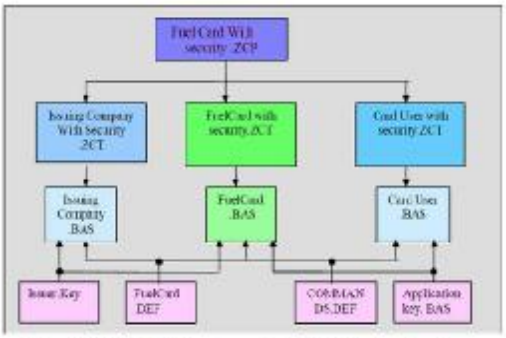


Figure (5.1): Procedure of design BasicCard application



Figure(5.2): The file structure of the Fuel Card Application without security



Figure(7.1): The file structure of the Fuel Card Application with security

AppandexB

Table(5.1):Shows the structure of Command-Response Pairs for the Fuel Card Application

CheckExpDate APDU command						
CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H0A	00	00	8	Current date(year and month)	N/A
Response APDU						
N/A			&H9000		Successful processing	
N/A			&H6B20		The Card can not Repersolalize in this time	
N/A			&H6B02		Card not initialized	
N/A			&H6B04		The card is expired	

Personalize the Card APDU command						
CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H0E	00	00	Length of input data	Username, initial dispensed quantity, maxbalance and maxtransaction	N/A
Response APDU						
N/A			&H9000		Successful processing	
N/A			&H6B02		Card not initialized	

Initialized the Card APDU command						
CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H0C	00	00	Length of input data	Car model, car number and car type	N/A
Response APDU						
N/A			&H9000		Successful processing	

Update Master PIN APDU command						
CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H10	00	00	4	Master PIN	N/A
Update Master PIN Response APDU						
N/A			&H9000		Successful processing	
N/A			&H6B02		Card not initialized	
N/A			&H6B08		Card not Personalized	

Update User PIN APDU command						
CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H10	00	00	6	User PIN	N/A
Update User PIN Response APDU						
N/A			&H9000		Successful processing	
N/A			&H6B02		Card not initialized	
N/A			&H6B08		Card not Personalized	

GetInitializzationdata APDU command						
CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H18	00	00	0	N/A	Length of output data
Response APDU						
Car Model, Car Number and Car Type			&H9000		Successful processing	
N/A			&H6B02		Card not initialized	

Update ExpDate APDU command						
CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H16	00	00	8	Expire Date (year and month)	N/A
Response APDU						
N/A		&H9000		Successful processing		
N/A		&H6B04		Thecardisexpired		
N/A		&H6B20		swTheCardcannotRepersolalizeinthistime		
N/A		&H6B22		Wrong input		

GetExpDate APDU command						
CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H16	00	00	0	N/A	8
Response APDU						
Expire Date (year and month)			&H9000		Successful processing	

VerifyCardInfo APDU command

CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H1A	00	00	Length of input data	Car number and Car type	N/A
Response APDU						
N/A	&H9000			Successful processing		
N/A	&H6B02			Card not initialized		
N/A	&H6B08			Card not Personalized		
N/A	&H6B0A			This card for anther Car		

VerifyMasterPIN APDU command

CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H1C	00	00	4	Master PIN	N/A
Response APDU						
Optional data	Status word	Meaning of status word				
N/A	&H9000	Successful processing				
N/A	&H6B08	Card not Personalized				
N/A	&H6B10	Car not verified				
N/A	&H6B0C	Invalid PIN				

Verify User PIN APDU command

CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H1E	00	00	6	User PIN	N/A
Response APDU						
N/A	&H9000	Successful processing				
N/A	&H6B08	Card not Personalized				
N/A	&H6B10	Car not verified				
N/A	&H6B12	PIN Required				
N/A	&H6B06	PIN Errors Exceeded				
N/A	&H6B0C	Invalid PIN				

GetPersonalizationeData APDU command

CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H20	00	00	0	N/A	Length of optional data in the response
Response APDU						
Optional data	Status word	Meaning of status word				
Username, Remaining Balance	&H9000	Successful processing				
N/A	&H6B02	Card not initialized				
N/A	&H6B08	Card not Personalized				
N/A	&H6B10	Car not verified				
N/A	&H6B12	PIN Required				

Get the Remaining Use APDU command

CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H24	00	00	0	N/A	6
Response APDU						
Optional data	Status word	Meaning of status word				
Remaining USE and Remaining balance	&H9000	Successful processing				

Use Card APDU command

CLA	INS	P1	P2	Lc	Data field	Le
&H80	&H22	00	00	4	Amount of transaction	N/A
Response APDU						
N/A	&H9000	Successful processing				
N/A	&H6B02	Card not initialized				
N/A	&H6B08	Card not Personalized				
N/A	&H6B10	Car not verified				
N/A	&H6B12	PIN Required				
N/A	&H6B14	PUBLIC car				
N/A	&H6B16	Invalid Transaction				
N/A	&H6B18	Exceed Max Balance				
N/A	&H6B1A	Exceed Max USE				