P- ISSN 1991-8941 E-ISSN 2706-6703 *Journal of University of Anbar for Pure Science (JUAPS)* Open Access
2008,(2), (1) :12-23

INVESTIGATION OF JAVA SMART CARD TECHNOLOGY FOR MULTI-TASK APPLICATIONS

***Sufyan T. Faraj, **Nidhal E. Berbat, **Sinan G. Abid Ali**

*College of Computer, University of Anbar, Iraq
**Computer Eng. Dept., College of Engineering-University of Baghdad, Iraq

ARTICLE INFO

Received: 01 / 04 /2008
Accepted: 24 / 04 /2008
Available online: 30/04/2008
DOI: 10.37652/juaps.2008.15423

Keywords:
APDU,
Java card applet,
Java smart card,
Package,
Smart cards.

ABSTRACT

The objective of this work is to build a number of different secure applets for Java smart card were each applet is designed for a specific task. Three packages were designed; the first package is the "Secure Wallet" which represents the electronic money storage card for financial services such as banks. The second package is "Card Connection". This package was designed to be used in prepaid communication applications such as telephone, Internet, etc. The third package is "Health Care", which represents the medical file for the card carrier. It is used in hospital, clinic and medical establishments. These applets were simulated by development kit for the Java card platform. Each applet was compiled, converted, verified, and installed successfully using the development kit tools. During the installation step, a script file was produced which contains Application Protocol Data Unit (APDU) commands. Each APDU command was processed and the result of processing was saved to a log file that represents both the command and the response APDU. Both the inputs and the results were in hexadecimal.

INTRODUCTION

A smart card is a card that is embedded with either a microprocessor and a memory chip or only a memory chip. Traditionally, the smart card has been used only for electronic identification and storage of personal data. However, the smart card of today contains a fully operational processor, and the increase in the memory capacity of the smart card makes it possible for the processor to run complex applications. The microprocessor card can add, delete, and manipulate information on the card, while a memory-chip card can only undertake a pre-defined operation.

To date, card manufacturers have developed their own proprietary solutions and programming languages for the smart card environment. The wide range of possibilities in smart card applications has given rise to the need to develop a commonly accepted solution, which can be used for developing applications that suit the smart cards of all manufacturers.

The Java Card has been designed for this purpose, and all interested parties in the field have been able to take part in the standardization process [1]. Using Java technology in smart cards introduces several benefits over conventional smart cards, these benefits are [2]:

- Interoperable - Applets developed with Java Card technology will run on any Java Card

* Corresponding author at: College of Computer, University of Anbar, Iraq, Iraq.E-mail address: sufyantaih@yahoo.com

technology-based smart card, independently of the card vendor and underlying hardware.

- Secure - Java Card technology relies on the inherent security of the Java programming language to provide a secure execution environment.
- Multi-Application Capability - Java Card technology enables multiple applications to co-exist securely on a single smart card.
- Dynamic - New applications can be installed securely after a card has been issued, providing card issuers with the ability to dynamically respond to their customer's changing needs.
- Compatible with Existing Standards - The Java Card Application Programming Interface (API) is compatible with international standards for smart cards such as International Standards Organization7816 (ISO7816), or Europay-MasterCard-Visa (EMV). It is referenced by major industry-specific standards.

The Java Card specifications enable Java card technology-based applets to run on smart cards. The Java Card specifications consist of three parts: The Java Card API, the Java Card virtual machine (JCVM), and the Java Card runtime environment (JCRE). The Java Card API specification defines the core framework and extension Java packages and classes for smart-card applications. The Java Card virtual machine specification defines a subset of the Java programming language and Virtual Machine (VM) for smart cards. The Java card runtime environment specification defines the runtime behavior for Java-based smart cards [3].

2. OPERATION AND ELEMENTS OF JAVA SMART CARD

The major components inside Java smart card are microprocessor and memories. The basic architecture of Java smart card consists of Applets, Java Card API, Java Card Virtual Machine, and Operating System (OS), which all included inside memories [4]. Fig. 1 shows the basic architecture of Java smart card. Java card consists of:

- Applet: It is defined as Java application and many of them can be fit inside one single Java smart card with each applet being identified by Application Identifier (AID).
- Java Card API: The API class library supports the Card bytecode in applets that allows inter-application communication if needed as well as provides loading services for the applets. It also takes care of ISO 7816 and provides classes to assist the compatibility of ISO 7816 in applets.
- Java Card Virtual Machine: The Java Card Virtual Machine is used to convert the data into suitable subset format which takes up less space in memory and also optimize the performance when executing in Java Card VM.
- Operating System: It deals with basic cryptography, I/O, memory access and application load services.
- Microprocessor: Calculations inside Java smart card will be done by microprocessor.
- Memory: Three types of memories used in the Java smart card are shown in Table 1.

Table 1: Memories type in Java smart card

TYPE	USAGE
ROM	Storing OS
EEPROM	Storing applets, Java Card API and Java Card VM
RAM	Use as a buffer for storing transmission data

Since the smart card is usually the only component of the system that the user holds in his or

her hand, it is enormously important with regard to the recognition and acceptance of the entire system. The smart card is normally one component of a complex system, since these complex systems (which are usually networked) are quite often hidden behind the card terminal, and it is these systems that make the services possible in the first place [5], Fig. 2 shows the elements of Java card applications.

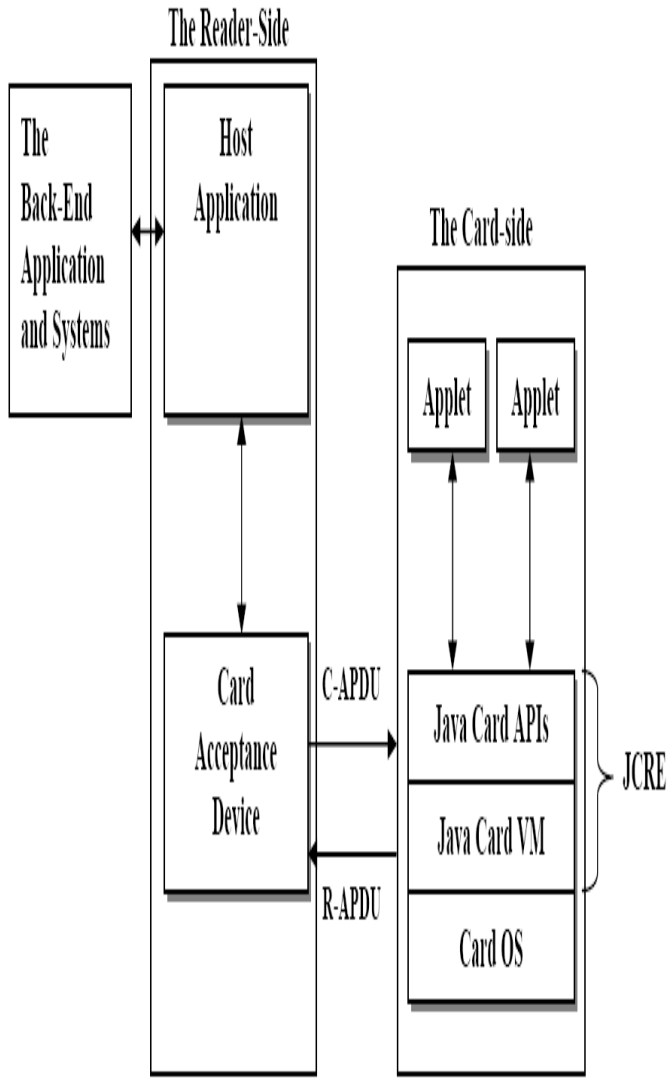

Fig. 2: Elements of Java card applications

The Back-end applications provide services to the in-card Java applets. The host application handles communication among the user, the Java Card applet, and the provider's back-end application. The Card Acceptance Device (CAD) is the interface device that sits between the host application and the Java Card device. Like computers, Java smart card uses for transmission the idea of data packaging to transmit data and this data package is the APDU. The interface device forwards APDU commands from the host application to the card, and forwards responses from the card to the host application. Also Java smart card introduces its own delivery control, the JCRE plays a very important role in controlling the data delivery [3] [4]. Fig. 3 shows the C-APDU and the R-APDU. The command and response APDU consists of the following:

- *CLA*: Class byte. It is to identify the application.
- *INS*: Instruction byte. To indicate the instruction code.

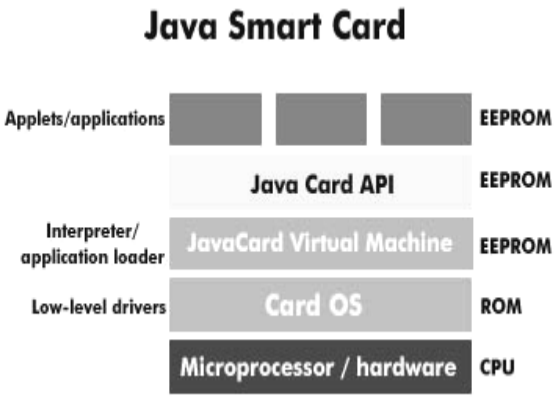

Fig. 1: The basic architecture of Java smart card

- *P1-P2*: Parameter bytes. To provide further qualification of the APDU.
- *Lc*: It indicates the number of bytes in the data field.
- *Data field*: The slot where data is actually allocated in the package.
- *Le*: It is the maximum number of bytes expected in the data field in the next response APDU.
- *SW*: status word. It indicates the status of the applet. Reader can notify the

occurrence of exception via the status words.

Command APDU						
Header				**Body**		
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**

Response APDU		
Body	**Status word**	
Data field	**SW1**	**SW2**

Fig. 3: The command and response APDU formats

JAVA CARD APPLETS

A Java Card applet is a Java program that adheres to a set of conventions that allow it to run within the Java Card runtime environment, without these applets the smart card is only a useless plastic card. The reason the name applet was chosen for Java Card applications is that Java Card applets can be loaded into the Java Card runtime environment after the card has been manufactured. That is, unlike applications in many embedded systems, applets do not need to be burned into the Read Only Memory (ROM) during manufacture. Rather, they can be dynamically downloaded onto the card at a later time [6]. The major four steps that comprise the applet design phase can be described as follows:

a) Specifying the functions of the applet:

The functions of the applet will be defined in this step and according to the design requirements. Also the characteristics and the condition of applet operation is defined in this step.

b) Assigning AIDs to both the applet and the package containing the applet class:

Packages and programs in the Java platform are uniquely identified using Unicode strings and a naming scheme. In the Java Card platform, however, each applet instance is uniquely identified and selected by an AID. Also, each Java package is assigned an AID. When loaded on a card, a package is then linked with other packages on the card via their AIDs [6]. ISO 7816 specifies AIDs to be used for unique identification of card applications. An AID is an array of bytes that can be interpreted as two distinct pieces. The first piece is a 5-byte value known as a resource identifier (RID). The second piece is a variable-length value known as a proprietary identifier extension (PIX). A PIX can be from 0 to 11 bytes in length. Thus an AID can range from 5 to 16 bytes in total length [7]. In the Java Card platform, an applet AID must not have the same value as the AID of any package or the AID of any other applet. However, the package AID and the AID(s) of applet(s) defined in the package must share the same RID. The package AID and the default applet AID for each applet defined in the package are specified in the Converted Applet (CAP) file. They are supplied to the converter when the CAP file is generated [6].

c) Designing the class structure of the applet programs:

A Java Card applet class must extend from the `javacard.framework.Applet` class. This class is the superclass for all applets residing on a Java Card. It defines the common methods an applet must support in order to interact with the JCRE during its lifetime. Fig. 4 shows the Java card applet methods. An applet must implement the static method `install()` to create an applet instance and register the instance with the JCRE by invoking the `register()` method. To notify the applet that a host application has selected it, the JCRE calls its `select()` method. Depending on the design of `select` method in the applet, if the selection is done, the JCRE passes incoming APDU commands to the applet for processing by invoking its `process()`

method, The `process()` method in class `javacard.framework.Applet` is an abstract method therefore a subclass of the `Applet` class must override this method to implement an applet's functions. Applet deselection occurs when the host application tells the JCRE to select another applet. The JCRE notifies the active applet that it has been deselected by calling its `deselect()` method, which returns the applet to the inactive, unselected state [3].

d) Defining the interface between the applet and the terminal application:

A Java Card applet should support a set of APDU commands, comprising a `SELECT` APDU command and one or more process APDU commands. The `SELECT` command instructs the JCRE to select the applet on the card. The set of process commands defines the commands the applet supports. These are defined in accordance with the functions of the applet. Structurally, the `SELECT` command and process commands are pairs of command and response APDUs. For each command APDU, the applet should first analyze the value of each field in the command. If the optional data fields are included, the applet should also determine their format and the structure. Consequently, the applet knows how to interpret each command and read the data. It then can execute the task specified by the command [3]. For each response APDU, the applet should define a set of status words to indicate the result of processing the paired-command APDU. During normal processing, the applet returns the success status word (`0x9000`, as specified in ISO 7816). If an error occurs, the applet must return a status word other than `0x9000` to denote its internal state. If the optional data field is included in the response APDU, the applet should define what to return.

SECURE WALLET PACKAGE

The first package “Secure Wallet” has two applets, the first applet is “Secure Wallet applet” which represents the electronic money storage card for financial services such as banks. The second applet is “Counter applet” which calculates the number of times of using the “Secure Wallet applet”.

The Functions of the Applets

The Secure Wallet applet stores electronic money and supports credit, debit, and check-balance functions. To prevent unauthorized use of the card, it contains a security algorithm. This algorithm requires the user to enter a two Personal Identification Numbers (PIN) (Master and User PIN), a string of eight digits at most. The security algorithm causes the card to lock after a specific number of unsuccessful attempts to enter the PIN. The Applet has the following conditions:

1. The first time running the applet, Master and User PIN must be updated.
2. The Master PIN must be verified before update the User PIN
3. The User PIN must be verified before any credit or debit transaction can be executed.
4. Master PIN is locked after one unsuccessful attempt of entering the PIN, while User PIN is locked after five unsuccessful attempt of entering the PIN.
5. The card's maximum balance for this applet is 32,000 Iraqi Dinar, and that no credit or debit transaction can exceed 100 Iraqi Dinar.

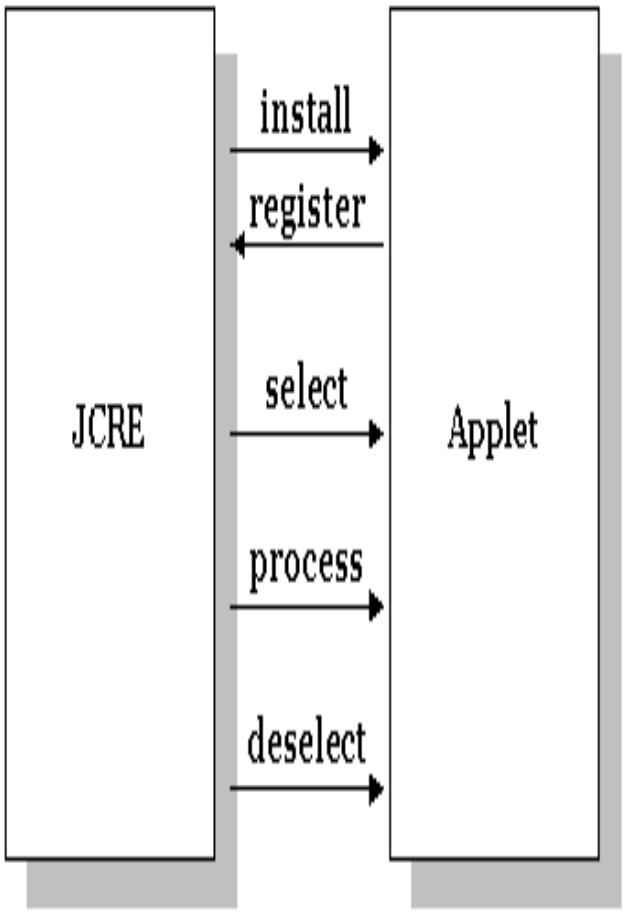

Fig. 4: Interaction between the JCRE and the applet by using the Java Card applet methods

Designing the Class Structure of the Applet Programs

To select the Secure Wallet applet, the select method has been designed to check if User PIN is blocked (because the User PIN by limit has been reached). If the User PIN is blocked the select method return false, otherwise it return true. The process method of Secure Wallet applet has the following methods:

1. *Credit*: this method credits an amount to the current balance, the amount to be credited to the current balance is in the data field of C-APDU. The User PIN must be verified before credit method can be called.
2. *Debit*: this method debits an amount from the current balance, the amount to be debited from the current balance is in the data field of C-APDU. The User PIN must be verified before debit method can be called.
3. *Get Balance*: this method returns the values of current balance, the value of current balance is returned in the data field of the get balance R-APDU.
4. *VerifyPin*: this method validates the Master PIN (or User PIN) depending on the value of P2. The Master PIN (or User PIN) to be validated is in the data field of C-APDU.
5. *Update*: this method updates both the Master PIN and the User PIN depending on the Tag, Length, and Value (TLV) code. The value of Master PIN to be updated is in the data field of C-APDU. To update the User PIN, the Master PIN must be verified first. The value of User PIN to be updated is in the data field of C-APDU.

In the design, the TLV codes are used as a switch to specify the state of the command APDU. The applet supports 2 TLV codes as follow:

CODES	
0xA1	UPDATE MASTER PIN
0xA2	UPDATE USER PIN

The process method of Counter applet has the following methods:

1. *Check*: this method checks the number of times for using this applet, the number of usage is returned in the data field of check R-APDU.
2. *Use*: this method calculates the number of times for using this applet.

CARD CONNECTION PACKAGE

This package contains two applets; the first applet is the "Card Connection applet" and the second is the "Account Calculator applet". These applets were designed to take advantage of logical channel, logical channel concept allows the concurrent execution of multiple applications on the card. In other words, the two applets have multi-session functionality and they will be able to inter-operate with each other from different channels. Indeed, they have the ability to be selected multiple times in different channels. This packaged was designed to be used in prepaid communication applications such as telephone, Internet, etc.

The Functions of the Applets

The Card connection applet is used to set or reset connection and if the connection is set the card send a command to debit a specific amount from the balance in Account calculation applet, also the Card connection determine the duration of card used (in seconds), in fact the duration is cumulative sum. The Account calculation applet is representing the monetary value (for example Card price) that will be debit from. The operation of the two Applets has the following condition:

1. The first time running the applet, the primary applet (Card Connection) must be selected on logical channel. And the other applet (AccountCalc.) must be selected on another logical channel.
2. The Card connection can't determine the duration of card use unless the connection is SET.
3. The Account calculator can't return the remaining amount unless the connection is SET.
4. If the balance in account calculator is equal to zero then a special status code will appear in the R-APDU.

Defining the Class Structure and Method **Functions of the Applets**

The select method of Card Connection applet is designed without any condition, it always return true if a match is found between AID of C-APDU and AID stored in smart card memory. To select another applet on another logical channel, a special select method for logical channel will be used, therefore it must be checked first if the applet already selected, if it already selected this means it has been selected on another logical channel and will return true, thus the condition will not be satisfied for the select method of logical channel and the method return false. Otherwise if it is not already selected it will return false and the condition will be satisfied and the method returns true. The process method of Card Connection applet has the following methods:

1. *Set connection*: this method is responsible to set the connection or not. The method check if the connection free or in use depending on the situation of the connection, the method then check the card balance value by branching to the debit method of Account Calculation applet. Depending on the answer of debit method, the method may set the connection or not.
2. *Card use*: this method responsible for calculating the duration of card using, it can't return the duration unless the connection is set. The method check also the card balance value, if the balance is zero then a notification message will be appear.
3. *Reset connection*: this method responsible for reset the connection.

The process method of Account Calculation applet has the following methods:

1. *Get balance*: this method returned the value of the remaining balance, the value of the remaining balance is returned in the data filed of the Get

balance R-APDU. The method can't return the remaining balance unless the connection is set.

2. *Debit*: this method compares the available balance with the debit value, this method can't be called by a C-APDU, it's called by Set connection method and Card use method of Card Connection applet, it returns true or false depending on the comparison.

When dealing with logical channel, there is specific APDU commands can contain encoded logical channel information. These are the commands whose CLA byte contains the bytes 0x0X, 0x8X, 0x9X, and 0xAX [8]. The X nibble is responsible for logical channels; only the two least significant bits of the nibble are used for channel encoding, which ranges from 0 to 3. When an APDU command is received, the card processes it and determines whether the command has logical channel information encoding. If logical channel information is encoded, then the card sends the APDU command to the respective channel.

HEALTH CARE PACKAGE

This package contains one applet; the "Health Care applet. The applet represents the medical record for the person who carries this card and it is used in hospital, clinic and medical establishments. Each time a person visit a medical office such that doctor's office, a clinic, or a hospital, the administrative stuff updates this record to keep the record current. This can particularly valuable in emergencies, when the carrier of the card might not be physically capable of communicating with medical personnel.

The Functions of the Applet

The applet is designed to store general information for the card such that card identification and expire date, also is store the medical information of the card carrier such as blood type, RH factor and the last two drug has been used by this person. Also it stores general information for administrative staff (Doctors or nurses) such that CAD identification (ID), Date and signature for security purpose. This applet also allows us to read the older record and give us complete information about the date, CAD ID, and another useful data. The operation of the Applet has the following condition:

1. For the first time running the applet, blood type and RH factor must be updated.
2. Only two drugs can be recorded in this applet each time.
3. Only the card ID and expire date can be modified.
4. Only 10 records can be reviewed.

Defining the Class Structure and Method Functions of the Applet

The Select method of Health Care applet is designed without any condition, it always return true if a match is found between AID of C-APDU and AID stored in smart card memory. The process method of this applet has the following methods:

1. *Initialize Update and Complete Update*: These two methods allow the administrative staff to show and update the card's and patient's information, the initialize update method allows the doctor (or nurse) to see health card ID, expire date and update number. The complete update method allows the doctor (or nurse) to update the expired date, health card ID, blood type and RH factor.

   In the design, the complete update method can't be called before the initialize update method.

2. *Initialize check and Complete check*: These two methods allow the administrative staff to enter and see the card's information and patient's information, the initialize check method allow the doctor (or nurse) to input the name of new drug and CAD ID, and it will show to administrative

staff the health card ID, expire date, old drug name and check number. The complete check method allow the doctor (or nurse) to input the signature of the patient and the date, and it will show the administrative staff the name of new drug that have been added.

In the design, the complete check method can't be called before initialize check method.

3.*Read record*: the read record method allow the administrative staff to read the previous records of the patient and made an idea about the drug taken by the patient and the date of token, his blood type and his RH factor.

SPECIAL CODES

1. Blood type

A = A type	B= B type	AB = AB type	0 = O type

2. RH factor type

00 = negative	01= positive

3. List of Drugs Example

A1 = penicillin	61 =Bactrim	03 =Depakene	c4 = Alprazolam
B2 = Zantac	a8 = Capoten	da = Estraderm	E5 = Inter-stop
C3 = Paracetoil	09 =Doxepin	D4 = Flu-out	08 = Femogex
F6 = Alarmine	f7 =Eramycin		

The file identification (FID) codes are supported in the applet, the applet support 2 FID codes as follow:

CODES	
0x9102	PARAMETER FID
0x9103	HEALTH_LOG_FID

Some special codes in the program are used as follow:

4. *Select file*: this method allows the administrative staff to review any record by selecting it.

In the design, records can be viewed by calling read record method after the select file method.

5. *Check the value*: this method is to add the new drug name to the record, this method can't be called by sending a C-APDU, and it called by another method (initialize check).

In the design, the TLV codes are used. The applet supports 3 TLV codes as follow:

CODES	
0xC5	EXPDATE_UPDATE
0xC6	HCARD_ID_UPDATE
0xC7	BLOOD_RH_TYPE_UPDATE

IMPLEMENTATION

The above mentioned applets were simulated by the development kit for Java card platform. This development kit is a tool for designing Java card technology-based implementations applets using the Application Programming Interface for Java card platform.

The development kit also provides an environment to test Java card applets. The steps of implementing Java card applets are as follow (see Fig. 5):

1. Compiling the Java source file using the J2SDK compiler.
2. Converiting the class files into a Converted Applet (CAP) file using the `converter` tool.

3. Optionally, verifying the CAP for validity. This step includes using the `verifycap` script to verify the validity of the CAP file and using `verifyexp` to verify the export files.
4. Installing the CAP file. By using the `scriptgen` tool to convert the CAP file into an (installation) APDU script file. Then using the `apdutool` to send the script file (installation APDU commands and CAP file) to the C-JCRE, or a JCRE on the Java Card device. The JCRE stores the CAP file in the card's memory.

The applets were compiled, converted, and verified successfully during the conversion step. In the installation step, the off-card installer tool produces a script file that contains commands APDUs that identify the beginning and end of the CAP file, its components, and component data.

The script file is used as input to APDUTool utility. The APDUtool reads a script file containing APDUs and send them to JCRE on-card installer. Each APDU is processed and returned to APDUtool, which write both the command and response APDU to a log file.

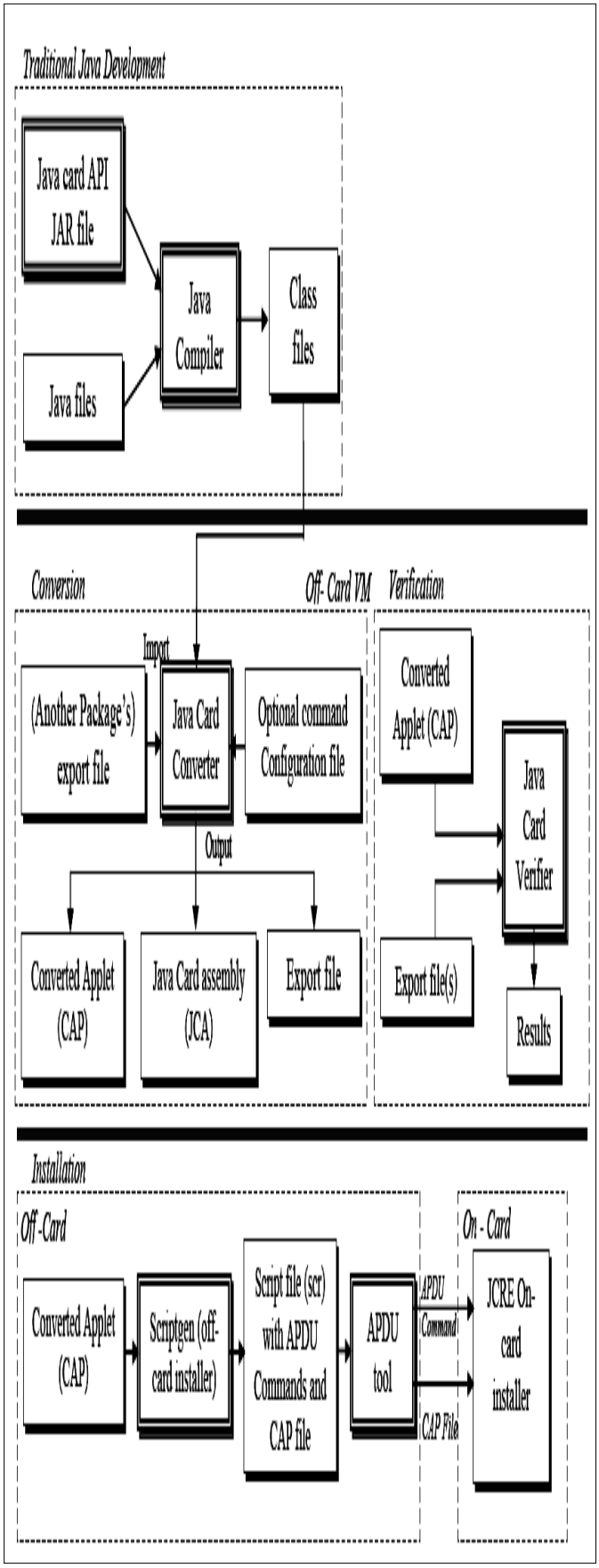

Fig. 5: Java Card applets development steps

CONCLUSIONS

The main goal of this work was to investigate Java smart card technology by designing secure applets each with a specific task for the Java smart card. All were written in Java language subset for the smart card. Smart cards with Java card technology are the most portable and secure ways for carrying digital personal information and computational calculations, a very powerful and needed technology. One of the most important issues is applets security. Applet's security can be obtained by different aspect through applet design phase.

In the secure wallet applet, the security was achieved by using both master PIN and user PIN. Master PIN controls the update of user PIN and thus no one can change the user PIN even if he\she has knowledge about user PIN. User PIN controls the credit and debit operations.

The security of health care package was achieved by defining a signature that is used by authorized persons to modify or enter new data related to the applet or to the card carrier.

This prevents from tampering or counterfeiting the original applet's data. As a future work, one may consider the design of Java card applets that use "Cryptography Extension" to support additional security for the applets.

REFERENCES

[1] Ruuskanen, J. P., "Java Card", Technical Report, University of Helsinki, 2000.

[2] Sun Microsystems, "Java Card Technology Datasheet", Java Card Technology, Sun Microsystems Inc., 2001.

[3] Ortiz, C. E., "An Introduction to Java Card Technology - Part 1", Sun Developer Series, Sun Microsystems Inc., May 2003.http:// developers. sun.com/techtopics/mobility/javacard/articles

[4] Chan, Y.L. and Chan, H.Y., "Java Smart Cards", Surprise 98 Report, 1998. http://www. iis.ee.ic. ac.uk/~frank/ surp98/ report/ ylc3/ report2.html.

[5] Rankl, W. and Effing, W., Smart Card Handbook, 3rd Ed., John Wiley & Sons Ltd., 2003.

[6] Chen, Z., "Java Card Technology for Smart Cards: Architecture and Programmer's Guide", Java Series, Addison-Wesley, September 2000.

[7] Sun Microsystems, "Virtual Machine Specification for Java Card Platform -Version 2.2.1", Sun Microsystems Inc., October 2003.

[8] Sun Microsystems, "Application Programming Notes for Java Card Platform -Version 2.2.1", Sun Microsystems Inc., October 2003.

P- ISSN 1991-8941 E-ISSN 2706-6703 *Journal of University of Anbar for Pure Science (JUAPS)* Open Access
2008,(2), (1) :12-23

استثمار قابليات تكنولوجيا بطاقات جافا الذكية في بناء تطبيقات متعددة المهام

د.سفيان تايه فرج نضال عزت بربات سنان غسان عبد علي

Email: sufyantaih@yahoo.com

الخلاصة:

أن هدف هذا المشروع هو بناء عدد من التطبيقات الآمنة المختلفة لبطاقة جافا الذكية حيث أن كل تطبيق تم تصميمه لوظيفة محددة. هناك ثلاث رزم تم تصميمها، الرزمة الأولى هي " المحفظة الآمنة" والتي تمثل بطاقة خزن المال الإلكتروني للبنوك. أما الرزمة الثانية فهي " بطاقة الاتصال " وهذه الرزمة صممت لتستعمل في تطبيقات الاتصالات المدفوعة مسبقا مثل اتصالات الهواتف والانترنت. الرزمة الثالثة هي " العناية الصحية والتي تمثل الملف الصحي لحامل البطاقة وتستعمل في المستشفى، العيادة، والمؤسسات الصحية.هذه التطبيقات تمت محاكاتها بواسطة عُدّة التطوير لمنصة بطاقة جافا. وكل تطبيق تم تجميعه، تحويله، التحقق من صحته، وتنصيبه بنجاح باستخدام أدوات عدة التطوير. خلال عملية التنصيب تم إنشاء ملف مستند و الذي يحتوي على أوامر APDU، كل أمر من APDU تمت معالجته وتم خزن نتيجة المعالجة بملف والذي يحتوي على كل من أمر – APDU واستجابة– APDU. وقد كان كلاً من الإدخال والنتائج ممثلين بالنظام السداسي عشر.