# Smart Cards and their Operating Systems

Heng Guo
HUT, Telecommunications Software and Multimedia Laboratory

Hguo@cc.hut.fi

**ABSTRACT**

*This paper presents smart cards and their operating systems. First smart cards categories, evolution, their hardware, their software and general operating systems are discussed. Then main open operating systems that have bigger market exposure such as java card, MULTOS, windows card are described in successive chapters.*

## 1 INTRODUCTION

The smart card is one of the latest addition to the world of information technology. The smart card has a microprocessor or memory chip embedded in it. The chip stores electronic data and programs that are protected by advanced security features. When coupled with a reader, the smart card has the processing power to serve many different applications. Smart cards provide data portability, security and convenience.

Smart cards currently are used in telephone, transportation, banking, and healthcare transactions, and soon to be used in Internet applications. Smart cards are already being used extensively in Japan and Europe and are gaining popularity in the U.S. In fact, the development of the smart card industry is very fast.

The references listed in the end are important for this paper to the same extend. For further details, please refer to the corresponding reference.

## 2 SMART CARD OVERVIEW

A smart card is a credit card size plastic card with an embedded microchip. They are highly secure and contain significantly more memory than a magnetic stripe card. There are two basic types of smart cards: contact and contact-less.

Contact cards have a one-centimeter diameter gold plated pad that has 8 contacts on it. These contacts are in turn wired to a microchip underneath the pad (Figure 1). The microchip can be a memory only chip or a microprocessor chip containing memory and a CPU.

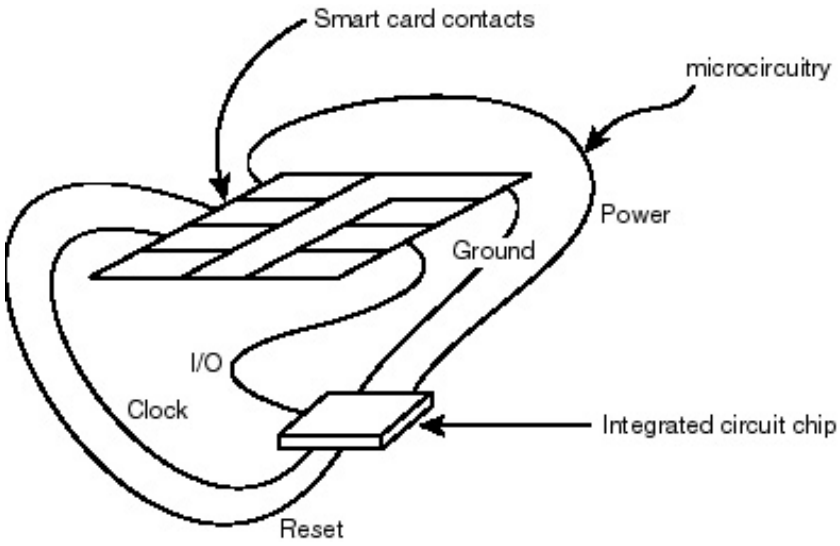


Figure 1: Smart Card Contacts

Memory cards are used mostly as telephone cards whereas microprocessor cards can be used for multiple applications on the same card. Although both cards can have stored value and stored data areas, the microprocessor card can in addition process the data since it contains a CPU, RAM and an operating system in read only memory (ROM).

Contact-less cards have an embedded microprocessor chip but also contain a miniature radio transceiver and antenna. They only operate within close proximity to the reader. Instead of inserting the card you simply pass the card close to the reader. Contact-less cards tend to be more costly than contact cards and are best suited for transportation and building access applications

Magnetic stripe cards have no processing capability and are limited to less than 100 bytes of memory. They are significantly less secure than smart cards but they do cost less. However, magnetic stripe card readers are often 10 times more expensive than smart card readers and are less reliable.

Hybrid cards can be any combination of contact, contact-less and magnetic stripe cards. Since 1982 the French banks have used the combination of chip and magnetic stripe cards as a bank credit card, allowing the banks to migrate to smart chip cards (Scott Guthery & Tim Jurgensen, 1998). They get the advantage of a more secure card while allowing a reasonable time to upgrade locations from magnetic stripe. There are even some hybrid cards that contain a microchip, magnetic stripe, bar code, optical code, picture and signature panel all in one card.

## 3 SMART CARD HARDWARE

The computer on a smart card is a single integrated circuit chip that includes the central processing unit (CPU), the memory system, and the input/output lines (Figure 2).

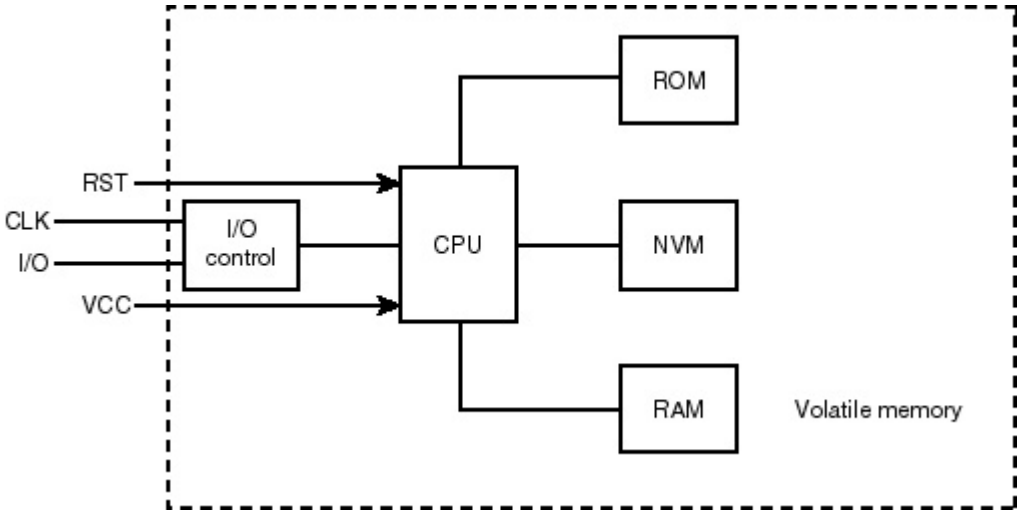


Figure 2: Elements of a smart card computer system

### 3.1 Memory System

Smart cards have a memory architecture that will be unfamiliar to most mainstream programmers. In fact, there are three kinds of memory on a smart card: read-only memory (ROM), nonvolatile memory (NVM), and a relatively tiny amount of random access memory (RAM). See Figure 2.

Read-only memory is where the smart card operating system is stored. General-purpose Here, one finds various utility routines such as doing communication and for maintaining an on-card file system along with encryption routines and special-purpose arithmetic routines. Code and data are placed in read-only memory when the card is manufactured and cannot be changed; this information is hardwired into the card.

NVM is where the variable data such as account numbers, number of loyalty points, or amount of e-cash is stored. NVM can be read and written by application programs, but it cannot be used like RAM. Although it can be written, the purpose and the performance of the action is totally different. NVM gets its name from the fact that it retains its contents when power is removed from the card.

There is some RAM on a smart card, but not very much. This is the most precious resource on the smart card from the card software developer's point of view. Even when using a high-level language on the smart card, the programmer is acutely aware of the need to economize on the use of temporary variables. Furthermore, the RAM is not only used by the programmer's application, but also by all the utility routines, so a programmer has to be aware not only of how much RAM he is using, but also how much is needed by the routines he calls.

### 3.2 Central Processing Unit

For earlier 8-bit microcontroller, the central processing unit in a smart card chip is typically using the Motorola 6805 or Intel 8051 instruction set. These instruction sets

have the usual complement of memory and register manipulations, addressing modes, and input/output operations. CPUs execute machine instructions at the rate of about 400,000 instructions per second (400 KIP), although speeds of up to 1 million instructions per second (1 MIP) are becoming available on the latest chips. The demand for stronger encryption in smart cards has outstripped the ability of software for these modest computers to generate results in a reasonable amount of time. Typically 1 to 3 seconds is all that a transaction involving a smart card should take; however, a 1024-bit key RSA encryption can take 10-20 seconds on a typical smart card processor. As a result, some smart card chips include coprocessors to accelerate specifically the computations done in strong encryption. (Scott Guthery & Tim Jurgensen, 1998)

After more than 20 years development, smart cards are evolving quickly. For example, memory sizes are increasing and processor architectures are moving from 8-bit to 16-bit and 32-bit configurations.

### 3.3 Smart Card Input/Output

The input/output channel on a smart card is a unidirectional serial channel. The smart card hardware can handle data at up to 115,200 bps, but smart card readers typically communicate with the card at speeds far below this.

The communication protocol between the host and the smart card is based on a master (host) and slave (smart card) relationship. The host sends commands to the card and listens for a reply. The smart card never sends data to the host except in response to a command from the host.

## 4 SMART CARD SOFTWARE

There are fundamentally two types of smart card software. One is host software, which is software that runs on a computer connected to a smart card. Host software is also referred to as reader-side software. The other is card software, which is software that runs on the smart card itself. As a counterpart of reader-side software, card software is also referred to as card-side software.

Most smart card software is host software. It is written for personal computers and workstation servers, accesses existing smart cards and incorporates these cards into larger systems. Host software will typically include end-user application software, system-level software that supports the attachment of smart card readers to the host platform, and system-level software that supports the use of the specific smart cards needed to support the end-user application. In addition, host software includes application and utility software necessary to support the administration of the smart card infrastructure.

Host software is usually written in one of the high-level programming languages found on personal computers and workstations C, C++, Java, BASIC, COBOL, Pascal, or FORTRAN and linked with commercially available libraries and device drivers to access smart card readers and smart cards inserted into them. In contrast, card software is usually written in a safe computing language such as Java, machine-level language such as Forth, or assembly language.

Host application software sometimes substitutes the smart card for an alternative implementation of the same functionality (for example, when an encryption key or a medical record is kept on a smart card rather than on a hard disk file on the local computer or in a central database on a server). Host system software manipulates the unique computing and data storage capabilities of the smart card by sending commands and data to it and by retrieving data and results from it

Card software is usually classified as operating system, utility, and application software, much as is the case with host software.

Card application software is typically used to customize an existing smart card for a particular application and amounts to moving some functionality from host application software onto the card itself. This may be done in the interest of efficiency in order to speed up the interaction between the host and the card or security in order to protect a proprietary part of the system. Card system software is written in a low-level machine language for a particular smart card chip and is used to extend or replace basic functions on the smart card.

Host application and card application are fundamentally different in their orientation and outlook. Card software focuses on the contents of a particular card. It provides computational services for applications in accessing these contents, and protects these contents from many applications, which might try to access them incorrectly. Host software, on the other hand, might make use of many different cards. It is typically aware of many cardholders and possibly many card issuers as well as many different kinds of cards.

Card software implements the data and process security properties and policies of a particular smart card. For example, a program running on the card might not provide an account number stored on the card unless presented with a correct personal identification number (PIN). Or a program running on the card might compute a digital signature using a private key stored on the smart card, but it would under no condition release the private key itself. Software running on a smart card provides secure, authorized access to the data stored on the smart card. It is only aware of the contents of a particular smart card and entities such as people, computers, terminals, game consoles, set-top boxes, etc. trying to get at these contents. Host software connects the smart cards and the users carrying them to larger systems. For example, software running in an automatic teller machine (ATM) uses the smart cards inserted by the bank's customers to identify the customer and to connect the customers with their bank accounts. Host software is aware of many smart cards and tailors its response based on the particular smart card presented.

Unlike most computer software, which relies on supporting services from its surrounding context, smart card software begins with the assumption that the context in which it finds itself is hostile and is not to be trusted. Until presented with convincing evidence to the contrary, smart cards don't trust the hosts they are inserted into and smart card hosts don't trust cards that are inserted into them. A smart card program only trusts itself. Everything outside the program has to prove itself trustworthy before the program will interact with it.

So it is useful to occasionally further categorize smart card software into application software or system software. (Scott Guthery & Tim Jurgensen, 1998) Application software uses the computational and data storage capabilities of a smart card as if they were those of any other computer and is relatively unaware of the data integrity and data security properties of the smart card. System software, on the other hand, explicitly uses and may contribute to and enhance the data integrity and data security properties of the smart card.

## 5 SMART CARD STANDARD

The basic contact smart card standard is the ISO 7816 series, part 1-10 while contactless cards will be governed by the ISO 14443 standard. These standards are derived from the identification card standards and detail the physical, electrical, mechanical, and application programming interface. Below is a list of the contact card standards(Smart Card Industry Association, 2000).

1. IS 7816 - 1 (1987) : Physical characteristics
    - Amendment 1 (1998) : Revised edition March 1998
2. IS 7816 - 2 (1988) : Dimension and location of contacts
    - Revised edition March 1998
3. IS 7816 - 3 (1989) : Electronic signals and transmission protocol
    - Amendment 1 (1992) : Protocol T=1
    - Amendment 2 (1994) : Revision of Protocol Type Selection
    - Amendment 3 (1998) : Introduction of 3 Volts ICCs
4. IS 7816 - 4 (1995) : Inter-industry commands and responses
    - Amendment 1 : (1998) : Revision Secure Messaging
5. IS 7816 - 5 (1994) : Registration system for application identifiers
    - Amendment 1 (1996) : Registration of identifiers
6. IS 7816 - 6 (1995) : Data elements for interchange
    - Amendment 1 (DIS) : Registration of IC Manufacturers
7. IS 7816 - 7 (1998) : Smart Card Query Language commands
8. DIS 7816 - 8 : Inter-industry Security Commands
9. CD 7816 - 9 : Inter-industry Enhanced Commands
10. ISO 7816 -10 (1999) : Synchronous cards

## 6 SMART CARD OPERATING SYSTEM

The smart card's Chip Operating System (frequently referred to simply as COS; and sometimes referred to as the Mask) is a sequence of instructions, permanently embedded in the ROM of the smart card. Like the familiar PC DOS or Windows Operating System, COS instructions are not dependent on any particular application, but are frequently used by most applications.

Chip Operating Systems are divided into two families :

- The general purpose COS which features a generic command set in which the various sequences cover most applications, and
- The dedicated COS with commands designed for specific applications and which can even contain the application itself. An example of a dedicated COS would be a card designed to specifically support an electronic purse application.

The baseline functions of the COS which are common across all smart card products include:

- Management of interchanges between the card and the outside world, primarily in terms of the interchange protocol.
- Management of the files and data held in memory.
- Access control to information and functions (for example, select file, read, write, and update data).
- Management of card security and the cryptographic algorithm procedures.
- Maintaining reliability, particularly in terms of data consistency, sequence interrupts, and recovering from an error.
- Management of various phases of the card's life cycle (that is, microchip fabrication, personalization, active life, and end of life).

The basic relationship between a smart card terminal and the smart card itself is of master and slave. The terminal sends a command to the smart card, the smart card executes the command, returns the result if any to the terminal, and waits for another command.

In addition to describing the physical characteristics of a smart card and the detailed formats and syntaxes of these commands and the results they return, smart card standards such as ISO 7816 and CEN 726 also describe a wide range of commands that smart cards can implement. Most smart card manufacturers offer smart cards with operating systems that implement some or all of these standard commands, together with manufacturer-specific extensions and additions.

Earlier in COS evolution, the issuer has to commit to a specific application developer, operating system and chip for each service the issuer wished to provide to its customer base. This leaves almost no flexibility to change any of these components without

having to invest funds into a new software and/or hardware implementation. As a result early smart cards were costly and inflexible.

But today we can clearly see a development towards open operating systems that support multiple applications. For on-card application development of programs that run inside the secure environment of the smart card chip, we highly recommend operating systems that have bigger market exposure such as JavaCard OS, MultOS and lately Windows for smart cards (JCI Smart Card System Consulting, 2001). They will be described distinctly in next chapters.

**Smart Card File Systems**

Most smart card operating systems support a modest file system based on the ISO 7816 smart card standard. Because a smart card has no peripherals, a smart card file is really just a contiguous block of smart card memory. A smart card file system is a singly rooted directory-based hierarchical file system in which files can have long alphanumeric names, short numeric names, and relative names.

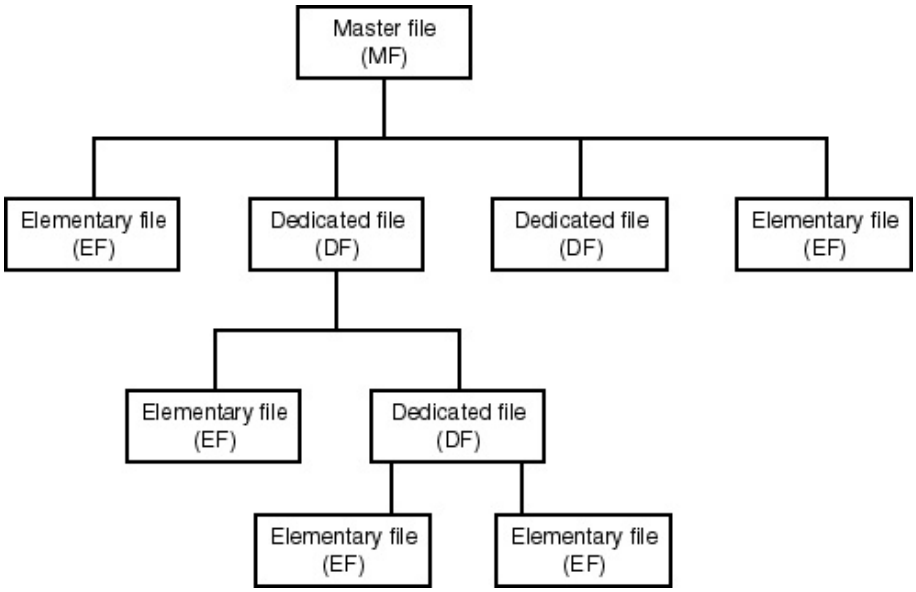


Figure 3: File system architecture of smart card

Smart card operating systems support the usual set of file operations such as create, delete, read, write, and update on all files. In addition, operations are supported on particular kinds of files. Linear files, for example, consist of a series of fixed-size records that can be accessed by record number or read sequentially using read next and read previous operations. Furthermore, some smart card operating systems support limited forms of seek on linear files. Cyclic files are linear files that cycle back to the first record when the next record after the last record is read or written. Purse files are an example of an application-specific file type supported by some smart card operating systems. Purse files are cyclic files, each of whose records contains the log of an electronic purse transaction. Finally, transparent files are single undifferentiated blocks of smart card memory that the application program can structure any way it pleases. (Scott Guthery & Tim Jurgensen, 1998)

Associated with each file on a smart card is an access control list. This list records what operations, if any, each card identity is authorized to perform on the file. For example, identity A may be able to read a particular file but not update it, whereas identity B may be able to read, write, and even alter the access control list on the file.

**Application Protocol Data Units (APDUs)**

The basic unit of exchange with a smart card is the APDU packet. The command message sent from the application layer, and the response message returned by the card to the application layer, are called an Application Protocol Data Units (APDU). Communication with the card and the reader is performed with APDUs. An APDU can be considered a data packet that contains a complete instruction or a complete response from a card.

ISO 7816-4 defines two types of APDUs: Command APDUs, which are sent from the off-card application to the smart card, and Response APDUs, which are sent back from the smart card to reply to commands.

APDUs consist of the following fields:

**Command APDU Format**

| CLA | INS | P1 | P2 | Lc | Data | Le |
|---|---|---|---|---|---|---|

Each Command APDU contains:

- A class type (CLA). It identifies the class of the instruction, for example if the instruction is ISO conformant or proprietary, or if it is using secure messaging.
- An instruction byte (INS). It determines the specific command.
- Two parameter bytes P1 and P2. These are used to pass command specific parameters to the command.
- A length byte Lc ("length command"). It specifies the length of the optional data sent to the card with this APDU.
- Optional data. It can be used to send the actual data to the card for processing.
- A length byte Le ("length expected"). It specifies the expected length of the data returned in the subsequent response APDU. If Le is 0x00, the host side expects the card to send all data available in response to this command.

**Response APDU Format**

| Data | SW1 | SW2 |
|---|---|---|

Each Response APDU contains:

- Optional data.

- Two status word bytes SW1 and SW2. They contain the status information as defined in ISO 7816-4.

The application software makes use of a protocol, which based on APDUs to exchange control and information between the reader and the card. These APDUs are exchanged by making use of the T=0 and T=1 link-layer protocols. A software component on the card interprets these APDUs and performs the specified operation; this architecture is illustrated in Figure 4.

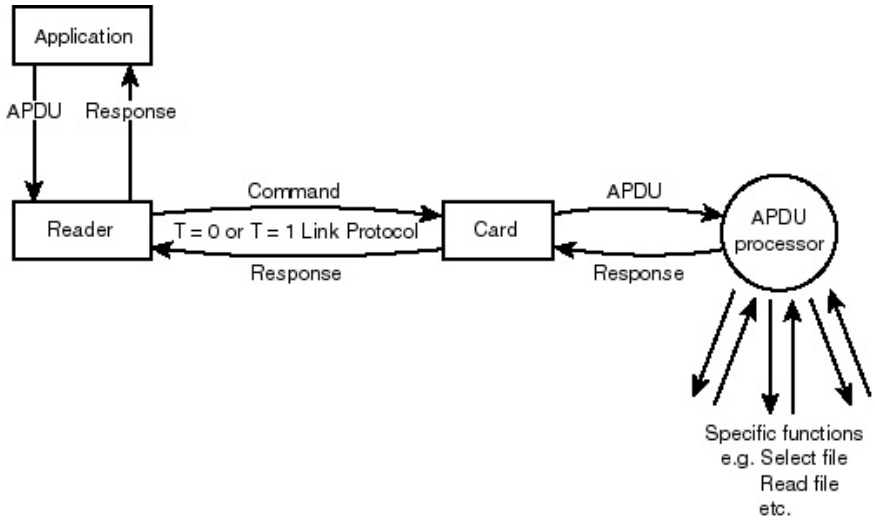


Figure 4: Application Communication Architecture

## 7 JAVA CARD

Java Card was introduced by Schlumberger and submitted as a standard by JavaSoft recently. Schlumberger has the only Java card on the market currently, and the company is the first JavaCard licensee. A smart card with the potential to set the overall smart card standard, JavaCard is comprised of standard classes and APIs that let Java applets run directly on a standard ISO 7816 compliant card. JavaCards enable secure and chip-independent execution of different applications.

The Java Card OS Architecture is as following,

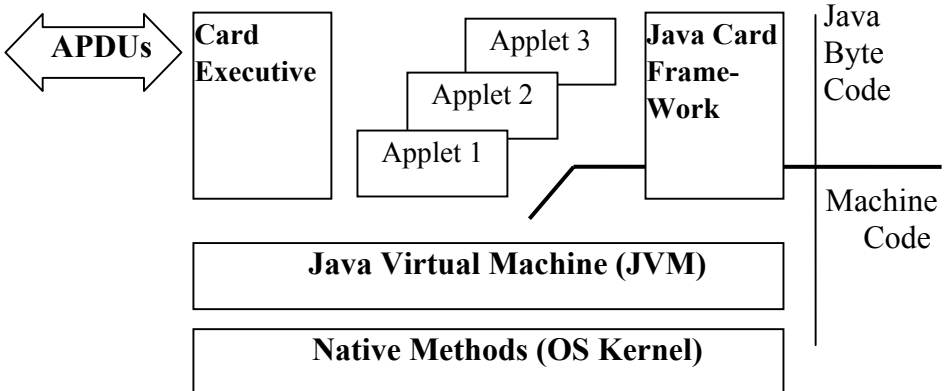

Figure 5: Java Card OS Architecture

Java Card Operating System allow the applications on a smart card to be written in Java. This brings the platform independence of Java to on-card software development, which used to be very proprietary for each card operating system manufacturer. In addtion, it provides a good basis for multi-application cards, which support more than one application at a time. The on-card executables are referred as Card applets and consist of a Java Card Specific byte code, which is interpreted by the Java Card Runtime Environment. This runtime environment controls the execution and makes sure that different applets do not interfere.

The process for developing a Card Applet is as following,

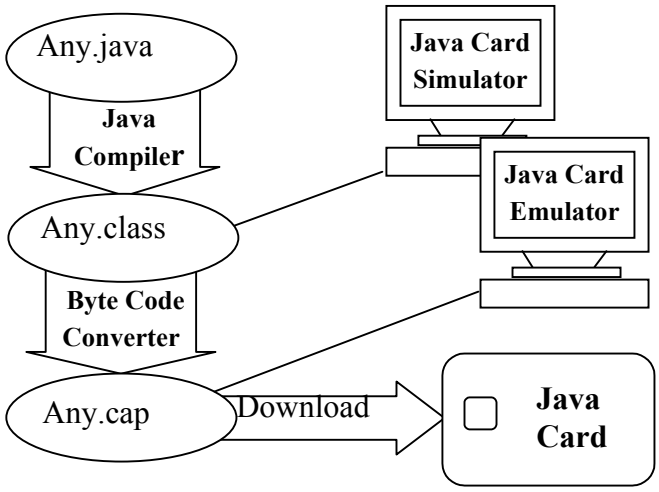


Figure 6: The Process for Developing a Card Applet

The source files can be compiled into regular Java byte code with a standard Java compiler. The Java Card Framework is included in compiling. The “class” files could be tested in the Java Card simulation environment. The byte code converter verifies the “class” file and optimizes them for the limited resources of a smart card. They are statically linked and converted into “cap” files. The “cap” files could be tested in the Java Card emulator environment.

## 8 MULTOS

MULTOS ('MULT-OS'), originally developed by Mondex International, is an open high-security multi-application operating system for smart cards, enabling a number of different applications to be held securely on the card at the same time. For application development, Mondex International has developed a smart card-optimized language: MEL (MULTOS Enabling Language), and the MULTOS-API specification. The

MULTOS specification is openly licensed and controlled by leading international organizations, collectively known as the MAOSCO Consortium, that are spearheading the development and roll-out of smart card technology across all business sectors and markets around the world. MULTOS is openly licensed by MAOSCO Ltd. (MAOSCO Consortium, 2000)

The most important feature of MULOS is the language independency.

**Assembler Programming Language**
MULTOS is the only platform that has an easy to use assembler language. MULTOS smart card applications were originally developed in MEL (MULTOS Executable Language), which contains typical assembler instructions plus “smart card friendly functions” called primitives.

**C Programming Language**
MULTOS is the only platform that currently has a C compiler. This is the most common embedded language at the moment. The SwiftC development tool from SwiftCard is an ANSI compliant compiler that allows you to very quickly port an existing application to the MULTOS operating system. Proof of concept of most ports can be completed within two weeks.

Scott Guthery of Mobile-Mind recently ported a DOS FAT file system from GNU source code in two days having never used Swiftcard before. Of the Swiftcard C compiler Scott said "The compiler is quite rigorous and caught lots of ambiguities. The generated MEL code is very efficient. In size I found it to be only about 25% larger than 8051 native code generated by the best 8051 C compiler on the market."

**Java Programming Language**
Both JavaCard and MULTOS can support the Java language. In both cases a Java compiler translates the source to Java class. For JavaCard the classes are converted to JavaCard byte code. For MULTOS, the SwiftJ compiler from SwiftCard translates the Java classes (or Basic, or Modula2) to MEL code.

**Visual Basic Programming Language**
Smartcard for Windows has chosen this as their application development language of choice. To make MULTOS accessible to the VB community SwiftCard Technology are currently working on a Visual Basic to MEL translator.

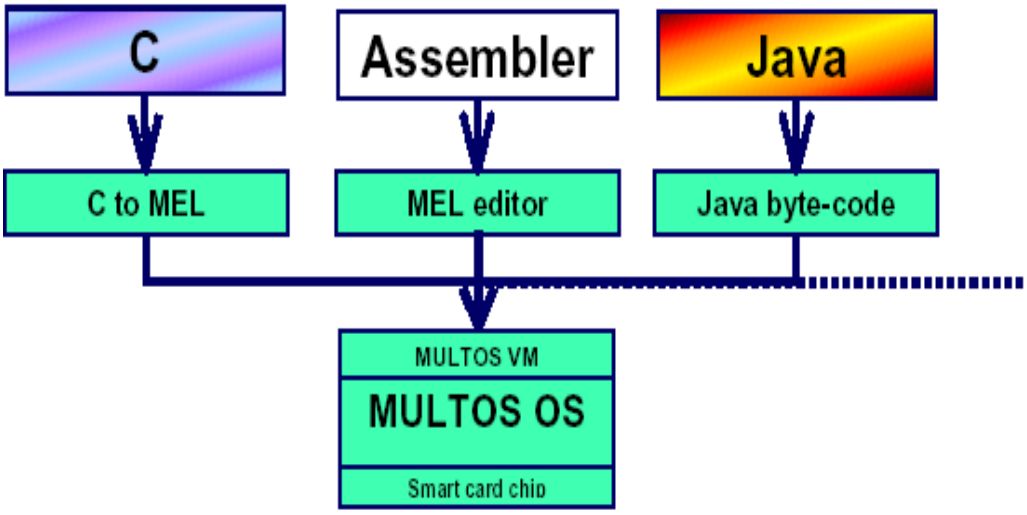


Figure 7: MULTOS Multi-application architecture

According to MULTOS materials, MULTOS is far more than an operating system. MULTOS is a complete scheme for managing smart card applications. The scheme defines a secure, efficient and cost-effective process for application management in a new world where a single card may house many applications from different sources.

MULTOS is designed and independently evaluated to a high level of security to ensure that issuers, application developers and other MULTOS service providers can build their business proposition without having to undertake expensive and lengthy evaluations of the underlying technology.

## 9 WINDOWS CARD

In 1999, Microsoft has entered the smart card Business and presented their Windows for Smart Cards operating system. As the newest member of the Windows operating system family, Windows for Smart Cards extends the benefits of the Windows environment to the smart card segment.

The Microsoft Windows for Smart Cards is an 8-bit, multiapplication operating system for smart cards with 8K of ROM. (Microsoft, 2000) It is designed to be a low-cost, easy-to-program platform that runs Visual Basic applications, and is designed to meet the criteria mentioned earlier: Extending the PC environment into smart card use.

The Windows for Smart Cards Architecture is as following,

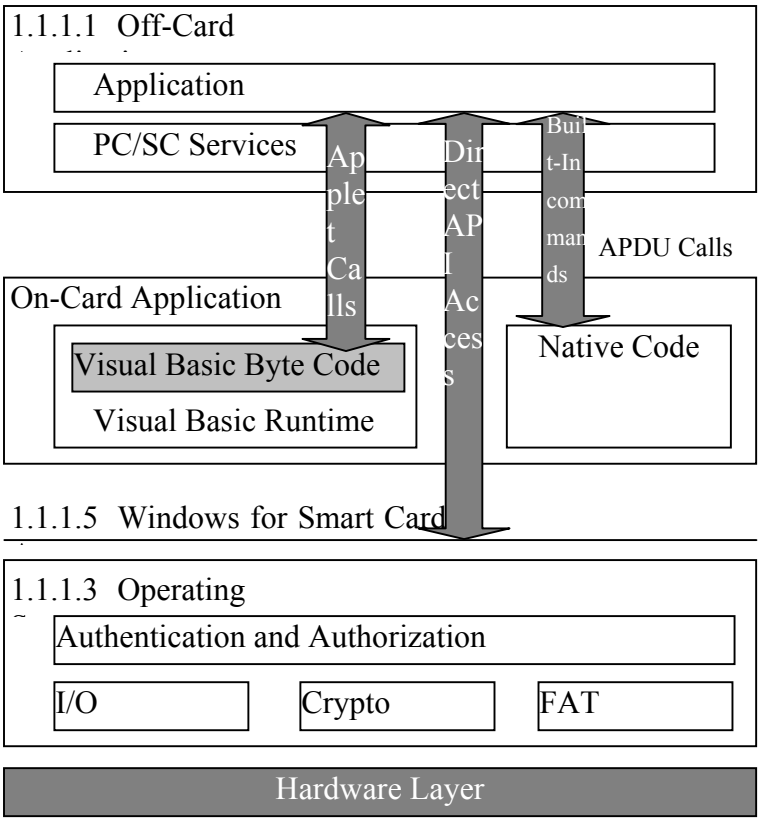


Figure 8: Windows for Smart Card Architecture

Quite similar to Java Card, the development of applications running on a smart card is leveraged by providing a high-level programming language. Instead of Java, Microsoft chose to use byte code generated from Visual Basic to be executed in a runtime environment on the card. The on-card applet communicates with a corresponding off-card application using common APDUs. The operating system exposes an API for working with the smart card's contents. That API is designed language-neutral and can be accessed either by Visual Basic or by native applets. It provides the following categories of functions: (Hansmann, Merk, Nicklous, Stober, 2001)

1. The File Interface includes 18 functions, for working with files.
2. The Authentication and Authorization Interface takes advantage of the Known Principals and ACL mechanisms, described earlier. There are functions for authenticating a principal, as well as for changing access rights.
3. The Cryptography Interface is similar to the PKCS#11 standard and exposes cryptographic algorithms.
4. The Utility Interface.

## 10 CONCLUSION

The important thing about smart cards is that they are everyday objects that people can carry in their pockets. They have the capacity to retain and protect critical information stored in electronic form.

The smartness of smart cards comes from the integrated circuit embedded in the plastic card. The same electronic function could be performed by embedding similar circuits in other applications, such as key rings, watches, glasses, rings or earrings. Smart keys are already being used for pay-TV subscriptions.

Smart cards are a relatively new technology that already affects the everyday lives of millions of people. This is just the beginning and will ultimately influence the way that we shop, see the doctor, use the telephone and enjoy leisure.

**References**


JCI Smart Card System Consulting, 2001. Smart Card Operating System.

Sun Microsystems Inc, 2001. Java Card Technology. <http://java.sun.com/products/javacard/>

Java Card Forum. <http://www.javacardforum.org>

MAOSCO Consortium, 2000. MULTOS: The Multi-Application Operating System for Smart Cards. <http://www.multos.com/>

Microsoft, 2000. Windows for Smart Card. <http://www.microsoft.com/windowsce/smartcard/>

Hansmann, U., Merk, L., Nicklous, M.S., Stober, T., 2001, Pervasive Computing Handbook, 409p.

R. Merckling, A. Anderson, March 1994. RFC 57.0. Smart Card Introduction.

Scott Guthery, Tim Jurgensen, 1998. Smart Card Developer's Kit. Macmillan Computer Publishing.

Smart Card Industry Association, 2000. Knowledge Base.

Rinaldo Di Giorgio, 1997. Smart cards: A primer. Develop on the Java platform of the future. *Java World*. December 1997.