7th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 27 – 29, 2004

SMART CARD TECHNOLOGY BASED ON JAVA

Philip Keersebilck
KAHO St Lieven
Associated partner of K.U.Leuven
Department of Electronic Engineering
Gebr. Desmetstr. 1
B9000 Gent, Belgium
Philip.Keersebilck@kahosl.be
www.kahosl.be

Abstract. *This paper introduces smart card technology. It deals with the characteristics and the benefits of the Java Card Technology, and also discusses important topics such as security and limitations of this technology.*
Keywords*: Smart Cards, Java Card Techology, security*

Background to (Java) Smart Cards

A smartcart looks like a credit card, with a chip embedded in the card's material. It can be easy recognized by their striking gold terminals. Figure 1 [1] shows such a smartcard.

Fig 1

Smartcards aren't new. They were introduced in Europe two decades ago in the form of *memory cards* for phone systems. This is the first type of chipcards. Memory cards store data locally, but can not perform computations on data.
A special category of a memory card is an optical memory card. It looks like a card with a piece of a CD glued on top. Once written with data, those data cannot be changed or removed.
Later on, the "intelligent" smartcard was developed, often called *microprocessor card*. This is the second type of chipcards. This card contains a memorychip and a microprocessor and is able to perform calculations on locally stored data. These smartcards can carry all necessary functions and information on the card, so they do not require access to remote databases at the time of the transaction.
In both cases, chipcards are safer than the traditional magnetics stripcards, whereon the information is on the outside of the card and thus can be copied easily.
In another way, you can buy smartcards with or without electrical contacts. In the first case, this "contact smartcard" can be physically attached to existing *card accepting devices* like e.g. a keyboard [2] or to standalone devices attached via a USB or a serial port.
In second case, we talk about "contactless smartcards". These cards communicate by means of a radio frequency signal.
Traditional development of a smart card application has been a lengthy process. Card manufactures developed their own proprietary solutions and programming languages for the smart card environment [3].
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 for all manufactures.
Sun saw this lack of standardisation among smart cards, and designed the *Java Card*

Technology for this purpose. All interested parties in the field have been able to take part in the *standardisation process*.
The Java Card Technology enables Java technology to run on smart cards. Sun provides for a common Java environment (the JCRE, or Java Card Runtime Environment) for these cards.
The Java Card Application Environment is licensed to smart card manufacturers, representing more than 90 percent of the worldwide smart card manufacturing capacity.

Architecture of a (Java) Smart Card

The smartcard is mostly a "one chip" cart, which means that memory and CPU are integrated in one chip.
Although Java Card Technology is independent of the supporting hardware, a collection of industry standards (ISO 7816) [4] has been made for smart cards. It defines the physical characteristics [5] of such cards.
This standard also defines other aspects such as transmission protocol, electronic signals, etc…
Figure 2 shows the surface contacts of a typical smart chip [2]. The VCC input powers the chip (typically 3 Volts). The I/O-contact transfers data between the smart chip and the connected host (via the card reader device). C4, C6 en C8 are reserved for future use.

Generally, there are typically three types of memory on the smard card system:

- ROM (basically used for storing fixed information, such as the O.S.)

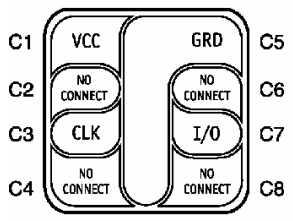

Fig 2

- EEPROM (used for data, which can be modified during operation of the card – this memory can – for security reasons - only be accessed by the way of the CPU)
- RAM (for temporary datastorage)

Example: a typically Java Card Device has an 8- or 16-bit CPU, running at 3.7MHz, with 1kB of RAM, 16 kB of EEPROM [1].
In the ROM, the Card Operating System is stored. Actually, a number of Card Operating Systems exist. Unfortunately, there is no standard operating system for smartcards just like Windows/Linux/etc. for PC-environment. This fact delays the development of new applications! Nevertheless, efforts are made to develop standards.
Rivals for Java Card Technology are *Multos* and *WindowsCard.* Multos is developed by Mondex. In contradiction to JavaCard, applications are programmed in C, so the performance is, in comparison with Java-applets, somewhat higher. Microsoft has developped his "own" standard, called WindowsCard.
But standard Java with all of its libraries, is far too big to fit on a smart card. Therefore, a "stripped-down" version of Java, named "JavaCard" is used for it.
JavaCard can load and reuse Javaprogram's for different applications. Because Java is used, it is clear that the concept is totally object oriented. The most important issues like encapsulation and information hiding are integrated.
Java Card includes many features, familiar to Java developpers, such as packages, dynamic object creation, virtual methods, interfaces and exceptions. But some elements of Java are **not** included such as dynamic class loading, threads, cloning, garbare collection and finalization. [7]
Figure 3 shows a *layered* concept [6] of the JavaCard.

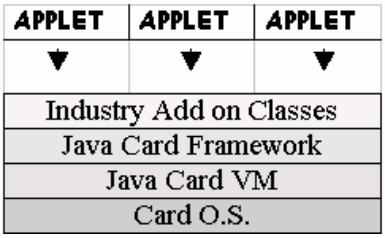

Fig 3

The Java interpreter translates the Java statements, present in the operating sytem, to machinecode, destinated for the Java virtual machine. But, instead of implementing a

complete Java virtual machine (is not possible as result of the limited memory space), a set of *API* classes (sometimes called "Java Card Framework") is defined "above" the Java interpreter.

The JVM (Java Virtual Machine), implemented in ROM, controls access to all smart card resources such as memory and I/O. It executes the Java bytecode on the smartcard, providing so the functions accessible from outside such as signature, log-in, applications. [7]

Like in browserapplications, Java Card applications are called *applets*. Multiple applets can reside on a cart. Therefore, a "bankcard" for example, with Java technology, can be used as phonecard and as an identity card, …

JavaCard is compatible with the EMV-standard (Europay, MasterCard and Visa). This standard describes the financial actions and transmission-protocols for financial transactions [4].

Smartcard manufacturers such as Schlumberger, Gemplus, Siemens, Visa, IBM, Bull,… work with the Java technology.

Java SmartCard implementation

Sun offers a *Java Card Development Kit* which is a suit of tools [10 for designing Java Card technology based implementations and

developing applets based on the Java Card API Specification [9]. A developer's guide can be found on different websites. [11]

The first two steps are the same as when developing traditional Java applets. But, once you have created the class files, the process changes: before these classes can be loaded on a Java Card device, they must be converted to the standard Converted Applet (CAP) file format, and then (optionally) verified. The cap file generation process does stringent type checking on the class file for value consistency (a much more thorough check than the Java compiles performs).

Once verified, the CAP file is ready to be installed on the Java Card Device (JCD).

You can test Java Card Applets without a smart card or card reader, just by using the JCD kit.

Lifetime of a Java Card

The lifetime of a Java Card's *hardware* starts when the native O.S., Java Card VM, API classes libraries and optionally, applets, are burned into ROM (*masking*). Before such a card reaches the user, it needs to be loaded with general data such as e.g. manufacturer's name (*initialising*) and personal data such as personal key, pin number, name, … (*personalization).* At this point, the card is ready to use and to be sold. Now you can insert your card into a reader and start downloading applets.

The physical lifetime ends when the card is expired or blocket due to a fatal error.

The lifetime of a Java Card's *software* starts when the applet is installed and registered. They stay alive when power is switched off.

Benefits of Java Card Technology

Java Card Techology produces a number of benefits [12]:

- *Interoperable*: applets will run on any Java Card Technology–based smart card (independent from the hardware)
- *Multi-application* capable: multiple applications can coexist on a single smart card
- *Post-issue-downloading*: new applets can be installed after a card has been issued
- *Secure*: Java Card technology provide a *relative* good secure execution (see discussion in point 7)
- *Open environment:* developers have access to Java developent tools
- *Compatible with international standards* (see point 2)

Applications of Java Card Technology

Thanks to the incorporated security functions, a Java smartcard can be used for many purposes.

The practical applications can be broadly classified into 3 main categories:

- Data carrier: card is used as a convenient, portable and secure means of information
- Identification: the card provides a secure

means of identifying the holder to allow access to e.g. a personal computer
- Financial operations

Some *examples*:

- Financial transactions:
  - Cardholder can be used to store value (credit/debit) and, because it can store information of multiple applications, it can be used to access all the accounts of the customer.
  - Cardholders can dial the bank and download some money onto the card.
- Medical information: a Java Card can store important medical information, destinated for the pharmacy, medicin, … (Germany, France, Japan)
- Public transport:
  - With a Java Card, the traveller could book in one time parking place, train- or airplainticket, hotelreservation, …
  - Automatisation of electronic roadpricing (Singapore)
- Government: the electronic passport, in combination with biometrical recognition (f.e. fingerprint). [12] In Belgium, the electronic passport, with Java Technology, will be introduced in 2004. In the Netherlands, a number of universities has already introduced the "Studentchipcard" [13]

Security of JavaCards

Java, an object oriented language, is, as known, a secure language [16]. Java applets are subject to Java security restrictions.

The security model of Java Card systems differs from standard Java in many ways! For example: the Security Manager class is *not* supported on Java Card; garbage collection is not present.

Otherwise, enhanced language security policies are implemented by the virtual machine, such as:

- Applet firewall: this prevents that an individual applet could access the contents or behaviour of objects by other applets
- Security and cryptographic classes: they support symmetric and asymetric encryption tools, pinmanagement, random datageneration, cardholder verification,…

It's pity that three issues oppress the security-level of the Java Smart Card: there exists a terminal problem, physical attacks threaten datasecurity [17] and post-issue downloading has risks!

First of all, there is the *terminal problem*. Since there is no built-in display in most cards, the CAD (Card Acceptance Device) must take on responsibility. Any display, used during transactions (such as transferring money), needs to be trustworthy. And that is now the problem: how sure can the card user be that the card is doing what it is supposed to be doing during transaction? How can a card user check to see for example whether account balances have been properly debited or not? So, what the user really need, is a *trusted display*.

The second problem concerns *physical attacks* on smart cards. Smart cards are tamper resistant, but not tamper proof [14]. An attacker can introduce errors by plenty of ways such as: pointing a radiation source at the card, hitting a card, applying unusual voltages, washing with chemical mixtures... Of course , some of these methods need expensive laboratory conditions and a team of specialists, but the security can not be taken for granted.

Finally, we know that new applets can be loaded on existing cards even after it has been issued to a consumer. This introduces a number of security risks, including the possibility that the downloaded appletcode will behave maliciously [15]. This significant risk can be mitigated by a solid software risk management.

By this point, it should be clear that the use of Java Cards, especially when used in e-commerce systems, has important security implications.

Limitations of JavaCards

JavaCards have some limitations. First of all, Java is an interpreted language and, this results in a relative slower execution of algoritms.

Moreover, smart cards have limited memory resources and computing power. The worst limitation is the amount of RAM. This limits the designing of applications. Memory usage can be improved by using advanced data-

compression techniques.

Evolutions

As shown in table 1, intelligent cards are, in comparison with the widely used magnetic stripe cards, expensive.

Table 1: Properties smart cards

	Max. data capacity	Processing power	Cost of card (€)
Magnetic Stripe Cards	140 bytes	None	0.2 – 0.75
Integrated Circuit Memory Cards	1 kB	None	1 – 2.5
Integrated Circuit Processor Cards	8 kB	8-bit … 16-bit cpu	7 - 15
Optical Memory Cards	4.9 MB	None	7 - 12

The expectation is that, with growing use of these cards in the future, the price will be significantly reduced.
With the announced introduction of 32-bit microprocessor, instead of the actual 8- or 16-bit microprocessors, smart cards will accept more advanced applications. Advanced encryption techniques will improve security.

Conclusion

Smart cards with the Java Card API represent a relatively new set of technologies. They open a wide range of applications for the growing e-commerce market. All in all, Java Card as a platform independent tool, seems a good choice for smart card applications. Especially multi-application smart cards just like the Java Smart Card have a great future: who likes to wear a great number of cards in his wallet?
Otherwise, the high cost is a problematic issue at the moment. Finally, applying this new technology introduces some security risks.

References

[1] *"Java Card Technology"*, http://java.sun.com/products/javacard/
[2] "*Smart-Card Devices and Applications*", http://www.dell.com/us/en/gen/topics/vectors_2001-smartcard.htm, January 2001
[3] *"P. java based smart cards.", Paavilanen P.*
[4]"*Understanding Java Card 2.O*" in http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html
[5] *The TB Smartcard Product Family* http://www.bull.gr/bull/www.cp8.bull.net/prod/tbfam.htm
[6] *"An introduction to Java Card Technology part 1"*, C.E. Ortiz http://wireless.java.sun.com/javacard/articles/javacard1
[7] *From hype to reality", Baentsch, Buehler*. http://www.computer.org/concurrency/pd1999/pdf/p4036.pdf
[8] *"Java Card Language Subset and Virtual Machine Specification", Sun Microsystems"*, http://www.javasoft.com/products/javacard/index.html
[9] *"An introduction to Java Card Technology – part 2"*, C.E Ortiz, http://wireless.java.sun.com/javacard/articles/javacard2
[10] *"Securing iMash"*, G. Yeung, A. Kaplan, P. Brisk http://www.cs.ucla.edu/~gavin/pub/cs239smartcard.pdf
[11] *"How to write a Java Card applets: a developer's guide"*,http://www.javaworld.com/javaworld/jw-07-1999/jw-07-javacard.html
[12] *"Moving towards biometrics electronic purse with Java Technology Card"*, Heng Su Miang. http://www.javacard.org/paper/c_paper.htm
[13] *"Stichting Studentenchipkaart: functionaliteit"* http://nieuws.surfnet.nl/nieuws/beleid/jg99-00/04.html
[14] *"Tamper Resistance of smartcard"* in The Second USENIX Workshop on Electronic Commerce Proceedings. ISBN 1-880446-83-9
[15] *"Java security – Hostile applets, holes and antidotes"*, Gary McGraw. ISBN 0-7881-9196-9
[16] *"JavaCard Platform Security"*, Sun. http://java.sun.com/products/javacard
[17] *"Securing Java"*, Gary Mc Graw & Ed Felten. ISBN
[18] *"Java Card Technology for Smart Cards"*, Zhiqun Chen. ISBN 0201703297