Study of Wireless smart cards and JAVA card Concepts

Credit Seminar Report

submitted in Partial fulfilment of the requirements
for the degree of
Master of Technology

by

1. Maj Chetan Dewan
2. Deepak Shinde
3. Ravikant Kr Nirala
4. Sudheer Kr

Under the guidance of

Prof. Bernard Menezes

Department of Information technology
INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

November 2006

Acknowledgement

I would like to thank Prof. Bernard Menezes for his invaluable support, encouragement and guidance. He is a constant source of inspiration and his guidance has made my endeavor an exciting experience.

Maj Chetan Dewan

Abstract

A contactless smartcard is a smartcard that can communicate with other devices without any physical connection, using Radio-Frequency Identifier (RFID) technology. Contactless smartcards are becoming increasingly popular, with applications like credit-cards, national-ID, passports, physical access. The security of such applications is clearly critical. A key feature of these systems is their very short range: typical systems are designed to operate at a range of 10cm. In this paper we look at the architecture as well as difference between RFID and smart cards. Over the last twenty years, the software in smart cards has radically changed. This has happened for several reasons, smart card software was initially rigid and monolithic and has now become more flexible with a clear separation between “operating system level” and “application level” parts. What is more, application-level resources are now much more accessible (nearly to end user level). Nevertheless, smart cards have evolved separately from an ever more distributed “outside world”. Next we look at JAVA for smart cards particularly the split VM model and how secure is a JAVA card. We also look at future trends and specification in the realm of Java cards.

TABLE OF CONTENTS

Abstract
Table of Contents

1. INTRODUCTION **1**
1.1. Motivation 1
1.2. Scope of the Seminar 1
1.3. Report Organization 1

2. STRUCTURE **2**
2.1. What is a Smart Card? 2
2.2. Classification of Smart Cards 2
2.3. Architecture 3
2.4 Contact vs Contactless 3
2.5 RFID Tags and Contactless Smart Card 3-5
2.6 *Operating Systems* 6

3. JAVA CARD **7**
3.1. Java on a Smart Card? 7
3.2 How Can Java Fit on a Card? 7
3.3 Smart card reader 8
3.4 PC/SC 8
3.5. OpenCard Framework 8
3.6 Software for communicating with the reader 9
3.7 Application Protocol Data Units (APDUs) 9
3.8 transporting APDUs and the classes' 9
3.9 Server Authentication Data Flow 10

4. JAVA CARD SECURITY **11**
4.1 How Secure Are Smart Cards? 11
4.2 The Terminal Problem 11
4.3 Physical Attacks on Smart Cards 12
4.4 Differential Power Analysis 12
4.5 How Card Java Lessens Security Risks 12
4.6 How Card Java Increases Security Risks 12
4.7 *Persistent memory and Garbage collection* 13
4.8 *Exception propagation* 13
4.9 *Inter-application attacks* 13
4.10 *Vendor introduced Native methods* 14
4.11 Object sharing 14

5. FUTURE JAVA CARD SPECIFICATIONS **15**
5.1. Future card 15
5.2. Next generation H/W Platform 16
5.3 No APDU 17
5.4 Java Card 3.x (Future card) 18

6.CONCLUSION FINDINGS AND FUTURE WORK 21

6.1 Conclusion 21
6.2 Findings 21
6.3 Future work 21

REFERENCES **22**

CHAPTER-1

INTRODUCTION

1.1. Motivation

Smart cards, credit-card sized devices with a single embedded chip CPU and RAM/ROM are viewed as “magic bullets" of computer security. They are being used for access control, electronic commerce, authentication, privacy protection, data storage etc. Because Smart cards are frequently used for security sensitive applications it becomes increasingly important to carry out analysis of the security risks particular to smart cards, and the unique threat environments that they face.

1.2 Scope of the Report

In this paper we first take a look at the basic structure of a smart card, its fundamental properties, its various varieties, classification basis, difference between RFID and a smart card and then we move on to Java card. We also look at security risks and the variety of attacks possible on a smart card. We also look at future version of smart cards and the various standards in the world.

1.3 Organization

The report is organised as follows:-

- Chapter two looks at Structure and Architecture in general
- Chapter three looks at Java card
- Chapter four looks into Java card security
- In chapter five, we look at future Java card
- Chapter six is conclusion crtics/findings and future work

CHAPTER-2
STRUCTURE

2.1 What is a Smart Card?

A smart card chip is actually a complete little computer with non volatile memory, storage, a card operating system (COS), and accompanying communication protocols. The most advanced smart cards on the market have the processing power once found in an IBM-XT (with less memory, of course). New technologies allow multiple applications on the same card and only one card can be issued to an end-entity for multiple applications. Smart cards hold these data within different files, and , this data is only visible to its program depending on the operating system of the card.

2.2 Classification of Smart Cards

Due to the communication with the reader and functionality of smart cards, they have varied classification based on components, Interface and OS used. The diagram below clearly shows smart card classification and sub classifications based on different parameters.

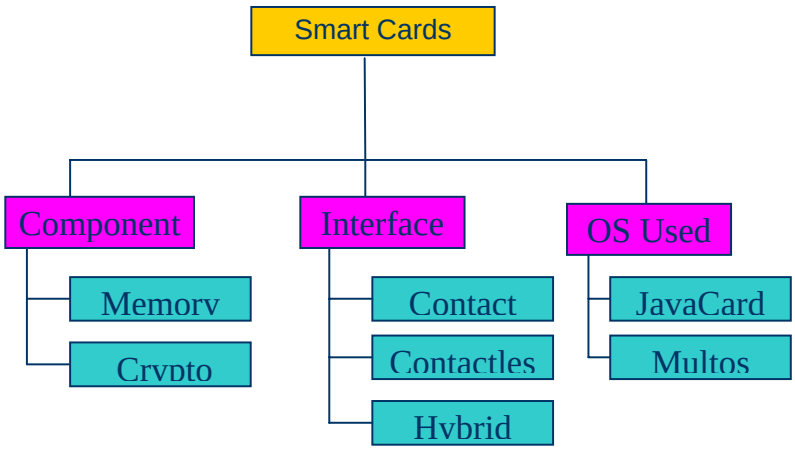


Fig 1. Smart card classification

2.3 Architecture

2.3.1 **Memory Cards.** These are the most common and the cheapest of cards available today costing anywhere between Rs 25-50/- only. They contain EEPROM ROM and some security logic. ROM holds card number, card holder name. EEPROM holds data that changes with time, usually application data. E.g. in pre-paid phone card, it holds talk time left. EEPROM can be locked with a PIN.

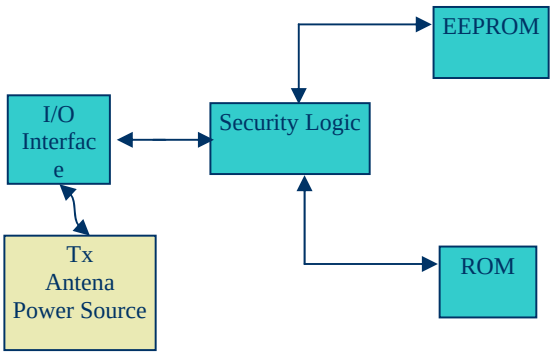


Fig 2. Memory card Architecture

These memory cards can have either a wired or wireless interface and mostly an external power source.

2.3.2 CPU cards. These contain a microprocessor or a microcontroller,ROM which is also called the Mask of the card. It holds the Operating System of the card, EEPROM which holds the application programs and their data and various keys used. A PROM holds the card unique serial number. It has RAM which is used as temporary storage space for variables. These cards have a Crypto Processor which is a 8-32 bit processor based on CISC architecture. Now a days Crypto processor are ASIC components based on some FPGA device. Crypto algorithms like DES, AES etc are commonly used. It has a wired/ wireless i/o Interface for data transfer to and from the card. They cost around Rs 100/- and can house multiple applications. They are very difficult to crack. They are used in Financial Cards, Electronic Purses, Access Control, Travel and Ticketing etc

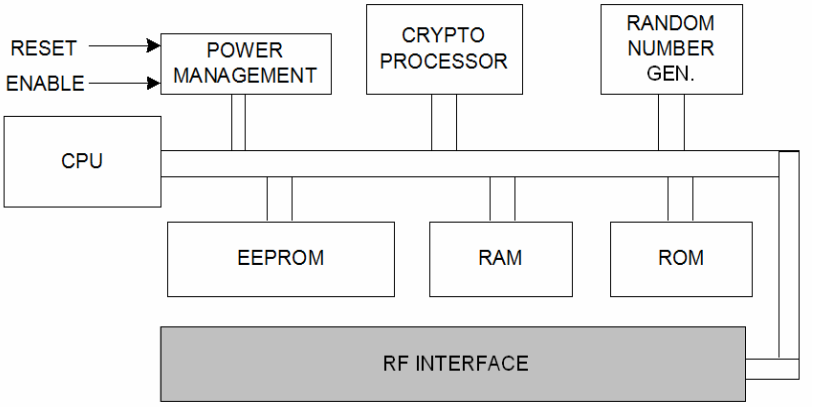


Fig 3. CPU Smart card Architecture

2.4 Contact vs Contactless

As smart cards have embedded microprocessors, they need energy to function and some mechanism to communicate, receiving and sending the data. Some smart cards have golden plates, contact pads, at one corner of the card. This type of smart cards are called *Contact* Smart Cards. The plates are used to supply the necessary energy and to communicate via direct electrical contact with the reader. When you insert the card into the reader, the contacts in the reader sit on the plates. According to ISO7816 standards the PIN connections are below:

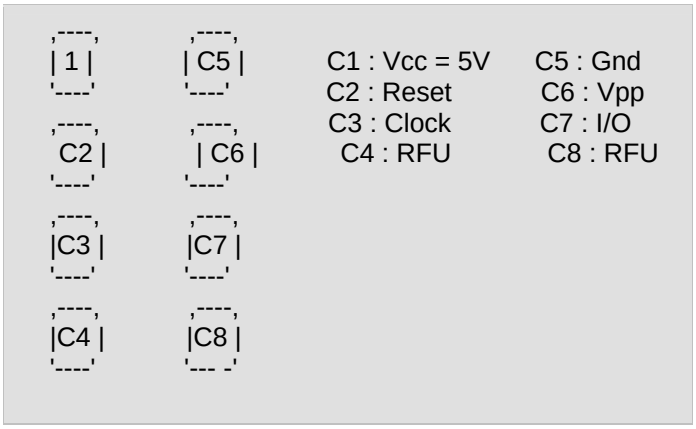
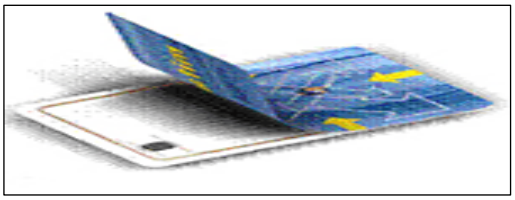


Fig 4. CPU Smart card Contact information

- I/O : Input or Output for serial data to the integrated circuit inside the card.
- Vpp : Programming voltage input (optional use by the card).
- Gnd : Ground (reference voltage).
- CLK : Clocking or timing signal (optional use by the card).
- RST : Either used itself (reset signal supplied from the interface device) or in combination with an internal reset control circuit (optional use by the card). If internal reset is implemented, the voltage supply on Vcc is mandatory.
- Vcc : Power supply input (optional use by the card).

The readers for contact smart cards are generally a separate device plugged into serial or USB port. There are keyboards, PCs or PDAs which have built-in readers like GSM cell phones. They also have embedded readers for GSM style mini smart cards.

Some smart cards do not have a contact pad on their surface.The connection between the reader and the card is done via radio frequency (RF). But they have small wire loop embedded inside the card. This wire loop is used as an inductor to supply the energy to the card and communicate with the reader. When you insert the card into the readers RF field, an induced current is created in the wire loop and used as an energy source. With the modulation of the RF field, the current in the inductor, the communication takes place.



5. Contactless Smart card

The readers of smart cards are usually connected to the computer via USB or serial port. As the contactless cards are not needed to be inserted into the usually they are only composed of a serial interface for the computer and an Fig antenna to connect to the card. The readers for contactless smart cards may or may not have a slot. The reason is some smart cards can be read upto 1.5 meters away from the reader but some needs to be positioned a few millimeters from the reader to be read accurately. There is one another type of smart card, combo card. A combo card has a contact pad for the transaction of large data, like PKI credentials, and a wire loop for mutual authentication.

2.5 RFID Tags and Contactless Smart Card

Many applications are now using radio frequency (RF) chip technology to automatically identify objects or people. These applications range from tracking animals and tagging goods for inventory control to enabling fast payment and securely identifying people. While these applications all use radio waves to communicate information, the RF chip technology used for each is quite different, addressing the application's unique storage, range and security requirements. As a general definition, radio frequency identification (RFID) tag technology is used in applications that identify or track objects and contactless smart card technology is used in applications that identify people or store financial or personal information. Applications most often have differing requirements in their use of RF technology, with RFID tag and contactless smart card technologies providing very different capabilities. The figure below gives a brief comparison between RFID tags and contactless smart cards on the basis of sensitivity of typical information used and strength of protection for information privacy and security.

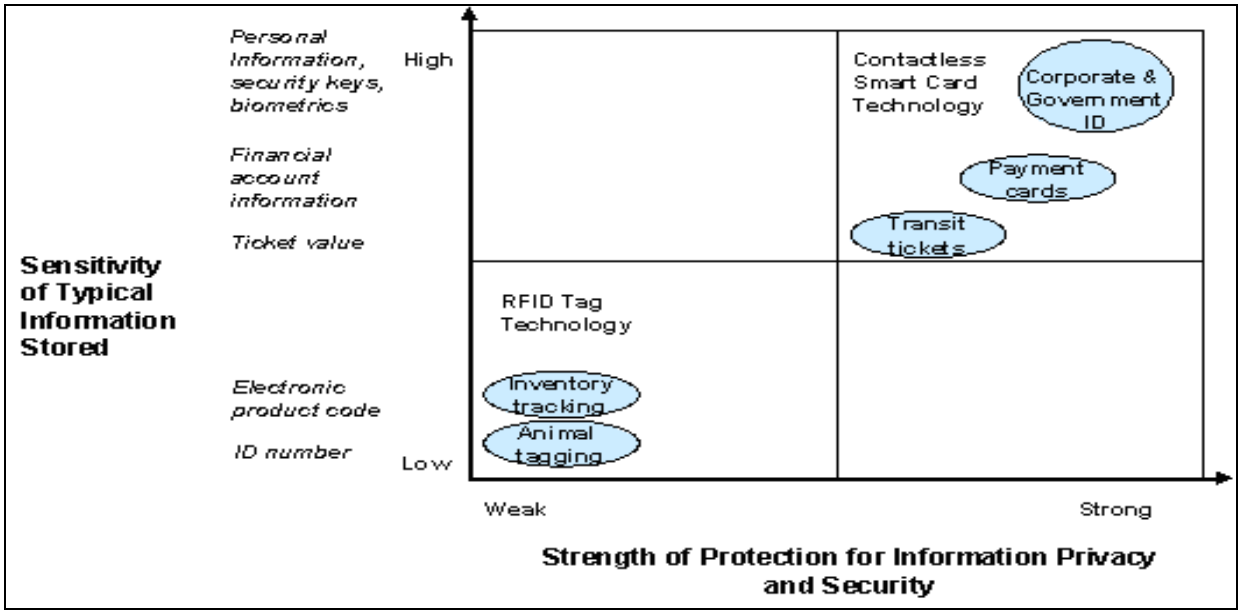


Fig 5. Contactless Smart card VS RFID comparison chart

RFID tags are simple, low-cost and disposable and are being used to identify animals, track goods logistically and replace printed bar codes at retailers. RFID tags include a chip that typically stores a static number (an ID) and an antenna that enables the chip to transmit the stored number to a reader. When the tag comes within range of the appropriate RF reader, the tag is powered by the reader's RF field and transmits its ID to the reader. There is little to no security on the RFID tag or during communication with the reader. Any reader using the appropriate RF signal can get the RFID tag to communicate its contents. Typical RFID tags can be easily read from distances of several inches (centimeters) to several yards (meters) to allow easy tracking of goods. RFID tags have common characteristics, including

- Low cost, high volume manufacturing to minimize investment required in implementation
- Minimal security, with tags able to be read by any compatible reader
- Disposable or one-time use
- Minimal data storage comparable to bar code, usually a fixed format written once when the tag is manufactured
- Read range optimized to increase speed and utility

Contactless smart card technology is used in applications that need to protect personal information or deliver secure transactions. Contact smart card technology provides similar capabilities but does not have the RF interface that allows contactless smart cards to be conveniently read at a short distance from the reading mechanism. There are an increasing number of contactless smart card technology implementations that capitalize on its ability to enable fast, convenient transactions and its availability in form factors other than plastic cards – for example inside of a watch, key fob or document. Current and emerging applications using contactless smart card technology include transit fare payment cards, government and corporate identification cards, documents such as electronic passports and visas, and contactless financial payment cards. The contactless device includes a smart card secure microcontroller, or equivalent intelligence, and internal memory and has the unique ability to securely manage, store and provide access to data on the card, perform complex functions (for example, encryption or other security functions) and interact intelligently via RF with a contactless reader. Applications that require the highest degree of information and communications security (for example, payment applications, government IDs, electronic passports) use contactless smart card technology based on an international standard that limits the ability to read the contactless device to approximately 4 inches (10 centimeters). Applications that need longer reading distances can use other forms of contactless technologies that can be read at longer distances.

Applications using contactless smart cards support many security features that ensure the integrity, confidentiality and privacy of information stored or transmitted, including the following:

- Mutual authentication. For applications requiring secure card access, the contactless smart card-based device can verify that the reader is authentic and can prove its own authenticity to the reader before starting a secure transaction.
- Strong information security. For applications requiring complete data protection, information stored on cards or documents using contactless smart card technology can be encrypted and communication between the contactless smart card-based device and the reader can be encrypted to prevent eavesdropping. Additional security technologies may also be used to ensure information integrity.
- Strong contactless device security. Like contact smart cards, contactless smart card technology is extremely difficult to duplicate or forge and has built-in tamper-resistance. Smart card chips include a variety of hardware and software capabilities that detect and react to tampering attempts and help counter possible attacks.
- Authenticated and authorized information access. The contactless smart card's ability to process information and react to its environment allows it to uniquely provide authenticated information access and protect the privacy of personal information. The contactless smart card can verify the authority of the information requestor and then allow access only to the information required. Access to stored information can also be further protected by a personal identification number (PIN) or biometric to protect privacy and counter unauthorized access.
- Strong support for information privacy. The use of smart card technology strengthens the ability of a system to protect individual privacy. Unlike other technologies, smart card-based devices can implement a personal firewall for an individual, releasing only the information required and only when it is required. The ability to support authenticated and authorized information access and the strong contactless device and data security make contactless smart cards excellent guardians of personal information and individual privacy.

2.6 Operating Systems

New trend in smart card operating systems is JavaCard Operating System. JavaCard OS was developed by Sun Microsystems and then promoted by the JavaCard Forum. Java Card OS is popular because it gives independence to the programmers over architecture. And Java OS based applications could be used on any vendor of smart card that support JavaCard OS.

Most of the smart cards today use their own OS for underlying communication and functions. But to give true support for the applications smart cards operating systems go beyond the simple functions supplied by ISO7816 standards. As a result porting your application, developed on one vendor, to another vendor of smart card becomes very hard work. Another advantage of JavaCard OS is, it allows the concept of post-issuance application loading. This allows you to upgrade the applications on smart card after delivering the card to the end-user. The importance is, when someone needs a smart card he/she is in need of a specific application to run. But later the demand can change and more applications could be necessary.

Another operating system for smart cards is MULTOS (Multi-application Operating System). As the name suggests MULTOS also supports multi-applications. But MULTOS was specifically designed for high-security needs. And in many countries MULTOS has achieved "ITSec E6 High" in many countries. Microsoft is also on the smart card highway with Smart Card for Windows. In a point of view the above Operating Systems are Card-Side API's to develop cardlets or small programs that run on the card. Also there is Reader-Side API's like OpenCard Framework and GlobalPlatform.

CHAPTER-3

JAVA CARD

3.1 Java on a Smart Card?

A Java card is a smart card that is able to execute Java byte code, similar to the way Java-enabled browsers can. But standard Java with all of its libraries (especially in the Java 2 guise) is far too big to fit on a smart card. A solution to this problem is to create a stripped-down flavour of Java. Card Java is just such a flavour. It's based on a subset of the Java API plus some special-purpose card commands.

Besides providing developers with a more familiar development environment, Card Java also allows smart cards to have multiple applications on them. For the most part, existing smart card products (especially in the financial arena) have only one application per card. This application is automatically invoked when power is provided to the card or the card is otherwise reset. The one-application-per-card paradigm doesn't scale well, to say the least. Who wants to carry 10 credit cards around? Card Java can solve this problem by allowing multiple applications, potentially written by different organizations, to exist on the same card. The idea of multiple applications running on the same VM by potential competitors raises a number of security issues, which we address later in the chapter.

3.2 How Can Java Fit on a Card?

Even a stripped-down version of Java and its accompanying VM requires a fair amount of computational power in order to work. To be capable of running Card Java, a smart card must have at least 16K of read-only memory, 8K of EEPROM, and 256 bytes of random access memory.

Given a Java Virtual Machine on a smart card, the number of possible new applications is mind-boggling. With an off-the-shelf (or off-the-Net) application development environment for Card Java, thousands of Java developers will be able to program smart cards. Gemplus and Schlumberger both distribute commercial Card Java environments. Of course, the memory and interface constraints of smart cards deeply affect programming style, testing concerns, and other aspects of program development.

Card Java has many features familiar to Java developers, especially those developers familiar with JDK 2.x.x. Card Java includes:

- Packages
- Dynamic object creation
- Virtual methods
- Interfaces
- Exceptions

Elements of Java that are not supported include:

- Dynamic class loading
- A security manager
- Threads
- Cloning
- Garbage collection
- Finalization
- sandboxing

In older Card Java paradigm, applets live on a card forever once they are installed during a process commonly called personalization. More specifically, although an applet's byte code may stay on the card forever once it is masked onto a card, an applet can be marked as unavailable and thus be permanently disabled. All applets that will be used on a card are installed at one time, before the card is issued to a consumer. These applets are selected and deselected by the Java Card Runtime Environment (JCRE). The JCRE is made up of the Virtual Machine and the core classes of the Java Card API. It keeps track of applets that are selected and currently active. The JCRE is in this sense the card executive, receiving messages (known as APDUs) on the input lines and carrying out the appropriate activities such as selecting an applet. Only one applet can run at a time in current Card Java implementations.

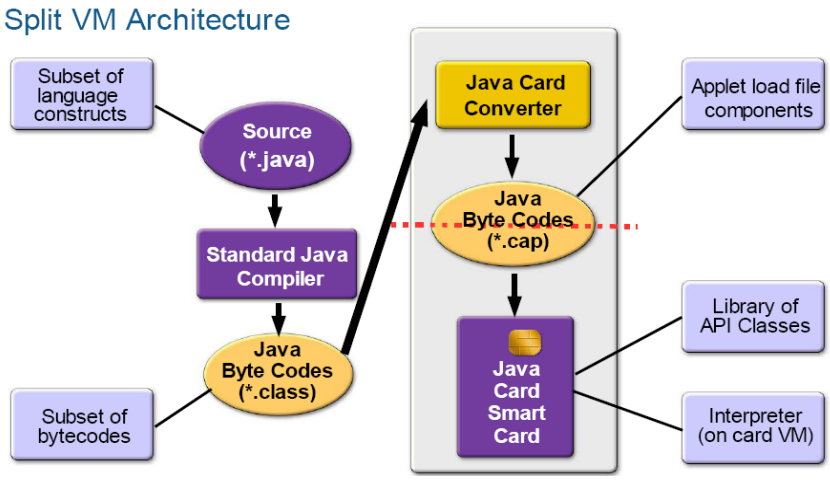


Fig 6. Split VM Architecture

Current Java cards are based on split VM models and allow applets to be loaded onto an existing card even after it has been issued to a consumer (much the same way that applet code is loaded into a browser's VM). The applets are created from class files and converted and then loaded onto smart cards. This introduces a number of security risks, including the risk that downloaded applet code will behave maliciously

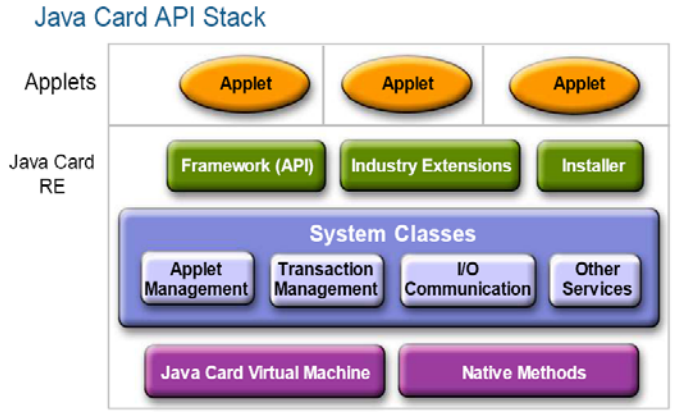


Fig 7. Java card API

3.3 Smart card reader

To communicate with a smart card or develop an application that is smart-card capable, we must have a reader. The reader provides a path for applications to send and receive commands from the card. There are many types of readers on the market, the most prevalent being the serial, PCCard, and keyboard models.

Each manufacturer provides a different protocol for speaking to a reader. Once you can communicate with the reader, there is one protocol for communicating with a smart card: Communication with a smart card is based on the APDU format.

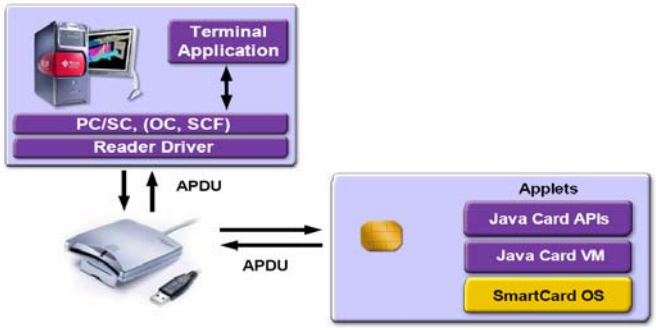


Fig 8. Java card client infrastructure

3.4 PC/SC

Microsoft and several other companies introduced PC/SC, a smart card application interface for communicating with smart cards from Win32-based platforms for personal computers. PC/SC does not currently support non-Win32-based systems and may never do so.

3.5 OpenCard Framework

OpenCard is an open standard that provides inter-operability of smart card applications across NCs, POS, desktops, laptops, set tops, and so on. OpenCard promises to provide 100% pure Java smart card applications. Smart card applications often are not pure because they communicate with an external device and/or use libraries on the client. (As a side note, 100% pure applications could exist without OpenCard, but without it, developers would be using home-grown interfaces to smart cards.) OpenCard also provides developers with an interface to PC/SC for use of existing devices on Win32 platforms.

3.6 Software for communicating with the reader

A number of object-oriented classes are needed for the smart card. These are:

- ISO command classes for communicating with 7816 protocol Classes for communicating with the reader
- Classes for converting data to a manufacturer-specific format
- An application for testing and using the cards for the purpose for which the application was designed
- Smart cards and smart card hardware

3.7 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. To provide this functionality, APDUs have a well-defined structure that is defined in a number of ISO documents belonging to the 7816 specification family.

APDUs consist of the following fields:

Command APDU Format

CLA	INS	P1	P2	Lc	Data	Le

Response APDU Format

Data	SW1	SW2

3.8 The following are some of the classes provided for transporting APDUs and the classes' functions:

Command -- encapsulate command APDU

Response -- encapsulate response APDU

ISOCardReader -- specify an interface. Each device has to implement this interface

ISOCommand -- construct an ISOCommand and execute the command through the ISOCardReader interface

3.9 Server Authentication Data Flow

A schematic showing the authentication data flow from the server is depicted below. The nine steps from challenge to validation of response are clearly shown in fig 9

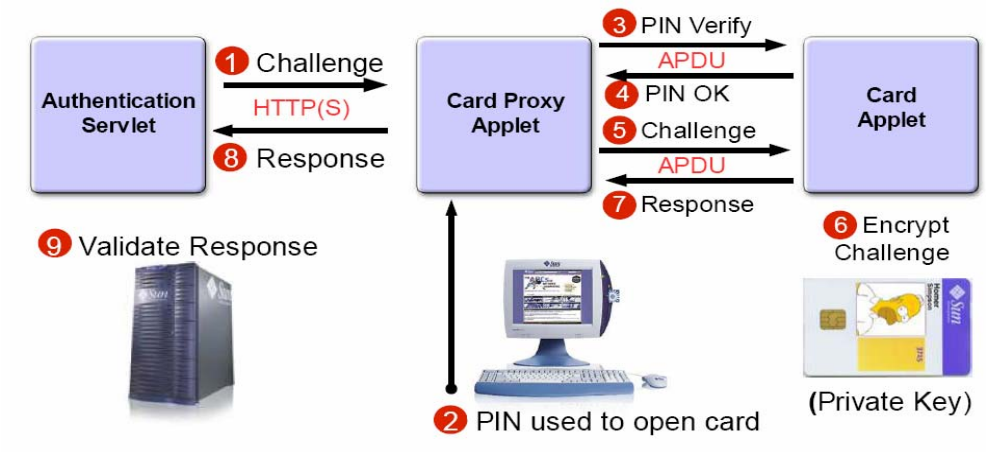


Fig 9. Authentication data flow

CHAPTER-4

JAVA CARD SECURITY

4.1 How Secure Are Smart Cards?

Before we dig into the security implications raised by putting a Java VM on a smart card, we need to address the issue of basic smart card platform security. Depending on how they're used, smart cards can sometimes be meant to keep secrets from the very people who carry them around and use them. Consider, for example, a smart card that stores monetary value in an internal register. If the card user can figure out a way to change the value of the register outside of traditional means, he or she might be able to mint money! Smart cards like this make tempting targets for bad guys.

4.2 The Terminal Problem

Smart cards need a way to interact with their users. Since there is no built-in display capability in most cards, the CAD must take on this responsibility. Any display used during critical transactions, such as transferring money, needs to have two properties: the display must be trustworthy, and it must be unspoofable. Making sure a terminal presents proper and trustworthy information to a user is known as the terminal problem.

The terminal problem is really a trust issue. How is a card user to be sure that the card is doing what it is supposed to be doing during a transaction? How can a card user check to see whether account balances (for example) have been properly debited or credited? The problem is that cards are very much black boxes. Many systems now on the drawing board include the use of personal computers as client-side CADs. Consumers will use a PC to interact with the smart card and to address the concerns raised by the terminal problem. The problem is that PCs are notoriously insecure, especially when they're used to exchange lots of documents and programs, as most consumers do.

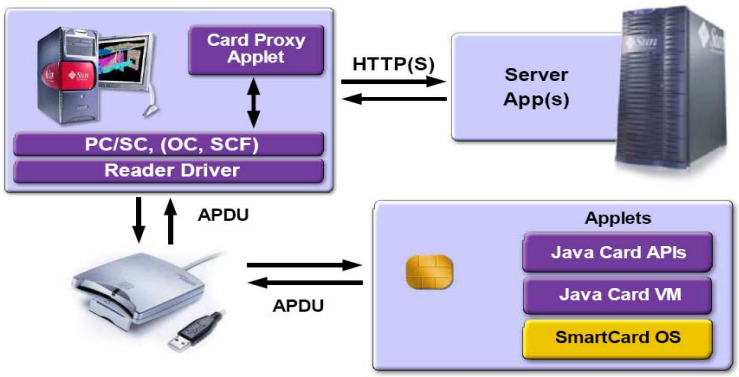


Fig 10. PC as a CAD

One direct consequence of PC untrustworthiness is a PC's impotence relative to the terminal problem. If your PC can't be trusted, how can you believe that what it is telling you on behalf of your smart card is correct? In fact, one excellent reason for using smart cards at all is that PCs can't be trusted. The reasoning goes that it is better to store secrets like PINs, sensitive personal data, and private keys on a smart card than on a PC. That way, if the PC is compromised, your secrets can't be so easily stolen.

However, this leaves us with the terminal problem. A scenario can make this more concrete. Imagine that someone has tampered with your Web browser either by hacking into your PC or by tampering with the Web browser executable before you downloaded it. Now clearly you can't trust the browser not to steal or rewrite data on the way from your smart card to you. Some things that might happen are:
The smart card requires a PIN before it can be used. Through a browser interface, you are queried for your PIN (which you faithfully enter). The corrupted browser sees the PIN go by and stores it for later illicit use.
The PC is used as a listening post in order to carry out capture/replay attacks against the smart card (these kinds of attack often work against cryptographic protocols unless the protocols are carefully designed to address this problem). The PC steals the private key off the smart card and is able to "legally" represent you by digital signature. What is needed is a trusted display. Some researchers have suggested that secure micro power displays be built into the smartcards.

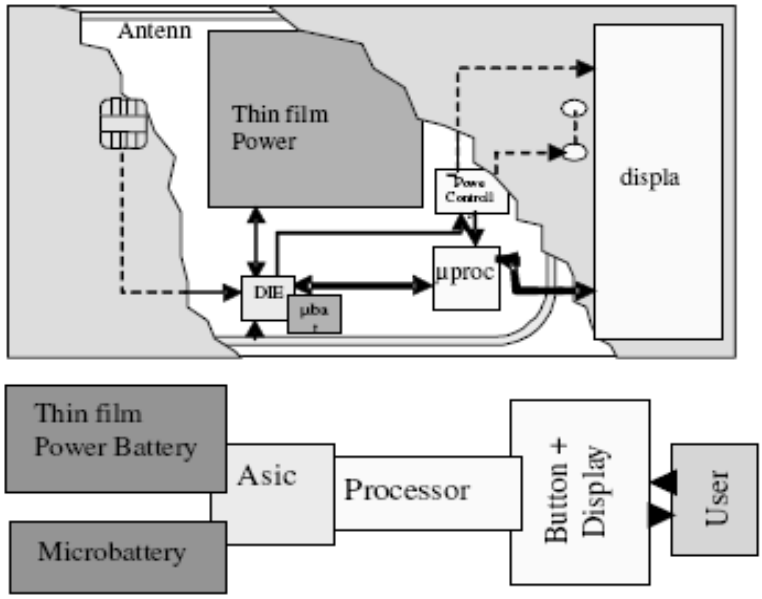


Fig 10. Display multimedia Java Card

4.3 Physical Attacks on Smart Cards

The most obvious and direct attack on a smart card is a physical attack on the card itself. In the case of a stored-value card, this sort of attack may even be carried out by the owner of a card. Physical attacks attempt to reverse engineer the card and determine the secret key

Boneh, DeMillo, and Lipton, three Bellcore researchers, published a paper called On the Importance of Checking Cryptographic Protocols for Faults in which they pointed out that an adversary who can introduce computational errors into a smart card can deduce the values of cryptographic keys hidden in the smart card How does the attacker introduce errors? There are plenty of ways. The attacker can subject the smart card to fluctuations in temperature, input voltage, or clock rate; point a radiation source at the card; or hit the card with a rubber mallet. Anything that is likely to cause voltages inside the card to fluctuate will do. In java cards it could be introduction of a faulty applet. This attack is a technique called Differential Fault Analysis, which works against a wide range of cryptographic algorithms. The upshot of all this is that unless a smart card cryptography mechanism is very carefully designed, any secret keys stored inside the card might be extracted by a determined attacker.

4.4 Differential Power Analysis

In 1998, Researchers at Cryptography Research, Inc., led by Paul Kocher, publicly announced a new set of attacks against smart cards called Differential Power Analysis (DPA). DPA can be carried out successfully against most smart cards currently in production.DPA is a complicated attack that relies on statistical inferences drawn on power consumption data measured during smart card computation. The equipment required to perform DPA is simple: a modified smart card reader and some off-the-shelf PCs. The algorithm itself is quite complex, but details have been widely published.

Chips inside a smart card use different amounts of power to perform different operations. By hooking a card up to an oscilloscope, a pattern of power consumption can be measured. Particular computations create particular patterns of spikes in power consumption. Careful analysis of the peaks in a power consumption pattern can lead to the discovery of information about secret keys used during cryptographic computations. Sometimes the analysis is straightforward enough that a single transaction provides sufficient data to steal a key. More often, thousands of transactions are required. The types of sensitive information that can leak include PINs and private cryptographic keys. Figure 11 is a conceptual diagram of DPA.

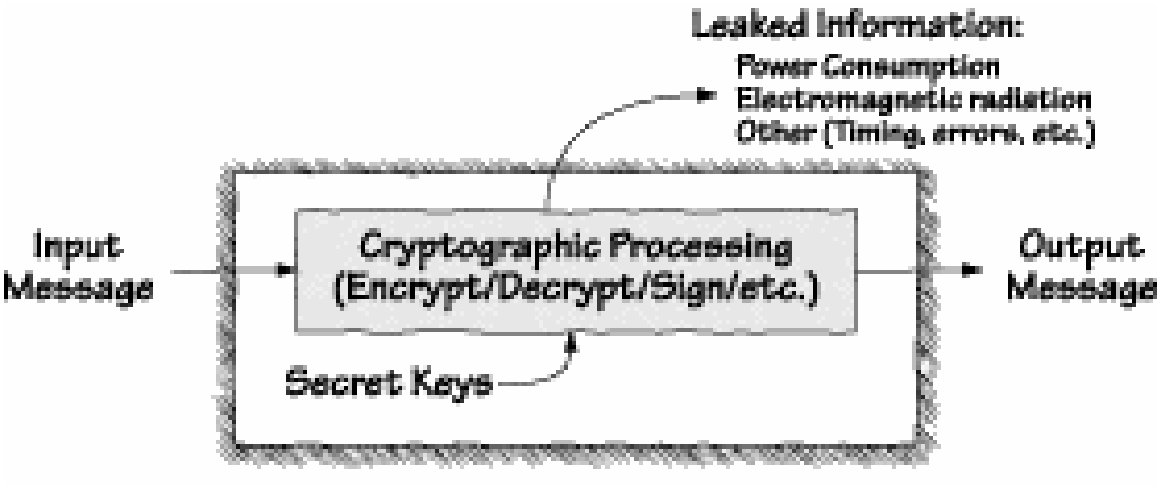


Figure 11. Differential Power Analysis.

Possible solutions include masking power consumption with digital noise or throwing random calculations into the mix. Another potential solution is randomizing the order of card computations so that in the end, the same computation is performed using different patterns of primitives. All of these potential technological solutions are ways to mask the giveaway patterns in the power consumption of the card.

4.5 How Card Java Lessens Security Risks

One of the most difficult problems in Java security is figuring out how to preserve type safety, while at the same time allowing dynamic class loading. If you can confuse the VM about the types of objects it is manipulating, you can break the security model. In its current form, Card Java takes care of this problem by removing dynamic class loading, making type safety easier to enforce. Class loading has always been problematic in Java and has introduced a number of serious security holes. Because it has no dynamic class loading, Card Java is less risky than regular Java from this perspective.

Another constraint imposed by Card Java, lack of threading, makes security analysis of applet code much easier than it is normally. Threading is difficult to implement properly and to use properly, plus threading takes a fair amount of overhead in the VM and significantly impacts the VM footprint. Although there can be multiple applets resident on the same smart card, Card Java systems allow only one applet to be selected at a time. (The multiple-resident-applications concept introduces risks of its own, which we address later.)

4.6 How Card Java Increases Security Risks

Lack of threads and the absence of dynamic class loading impact security in a positive way, but the opposite effect can be seen with other Card Java features. In other words, the removal of some features of Java (clearly intended to make possible the migration to Card Java) may introduce new security problems. Risks that are introduced involve:

- Lack of garbage collection
- Exception propagation problems
- Multiple applications and applet firewalling
- Object-sharing loopholes
- Access to native code

4.7 Persistent memory and Garbage collection

Garbage collection is a good example of a feature whose absence has a security impact. Without a system for freeing allocated memory, the problem of denial-of-service attacks is exacerbated. The Card Java 2.x specification does not implement garbage collection on the card. Memory leaks are a classic problem in languages such as C++ that do not support automatic garbage collection. This leads to memory becoming full when objects are inefficiently created and destroyed. Since Card Java does not support garbage collection, logic errors in applet code may over time exhaust free memory, rendering the card useless. This problem is especially acute on cards with limited memory resources.

4.8 Exception propagation

Exception propagation is an interesting issue for Card Java as well, since uncaught exceptions could lead to a card being muted (disabled for normal use). The potentially fatal effect of unhandled exceptions implies another significant exposure to unintended denial of service, once again resulting from subtle programming errors. As with memory exhaustion, the requirement to ensure that these kinds of errors do not occur raises the requirements for extensive testing and analysis of the Card Java applet code.

4.9 Inter-application attacks

Since Card Java allows multiple applications to be resident on the same smart card, there is a risk of interapplication attacks. This risk is especially relevant in situations where applets may be provided by competing vendors. Card Java defines **applet firewalls** meant to protect applets from one another.

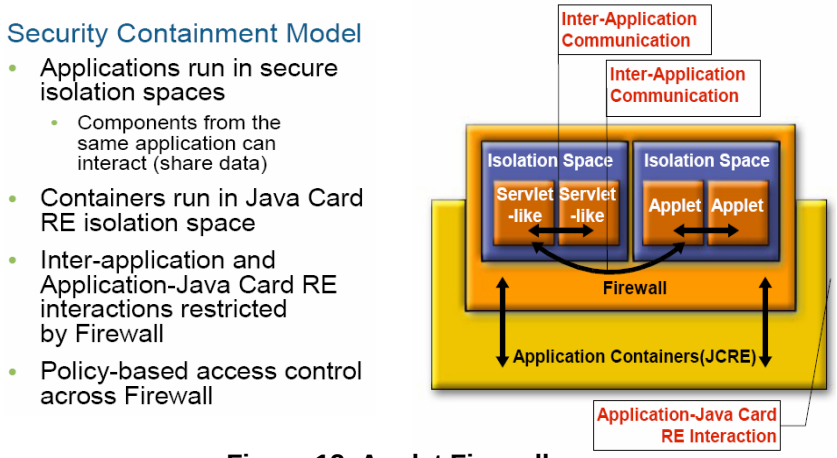


Figure 12. Applet Firewall

It appears that the main protections between applets are related to memory management by a security containment model as mentioned above

4.10 Object sharing

One feature that is in high demand in systems with multiple applications is object sharing. Card Java 2.0 includes an object sharing approach that includes a loophole. When using virtual methods sharing a virtual method of one with another applet is tantamount to allowing that applet complete control over the method. In short, granting an applet permission to access an object amounts to also granting that applet the ability to export indirect access to that object to any other applet. This has a clear weakening effect on any assurance about the protection of an object.

4.11 Vendor introduced Native methods

By far the biggest risk presented in the design of Card Java is a potential ability for a vendor to add and use Native methods on the platform. Obviously, this will compromise portability (applets that use Native methods will not be automatically portable to other cards), and it may expose the card to dangerous code that exists outside the applet firewalls. In fact, if Native methods are available, the concept of firewalls deteriorates. Native code is not restricted by the JCRE mediation of access, and misuse is possible.

The very real security concern is that an attack applet will make use of Native code to carry out operations that would otherwise be stopped by the JCRE. When Native code executes, all bets are off for the Java Virtual Machine and its protection mechanisms. Native code in applets completely breaks the idea behind Java security. Attack applets are likely to make use of Native method calls.

CHAPTER-5

FUTURE JAVA CARD SPECIFICATIONS

5.1 Future card

The risks covered earlier in chapter 4 were introduced into Java with its transformation to Card Java. Many of these risks will be taken care of in the future java card specifications listed below.

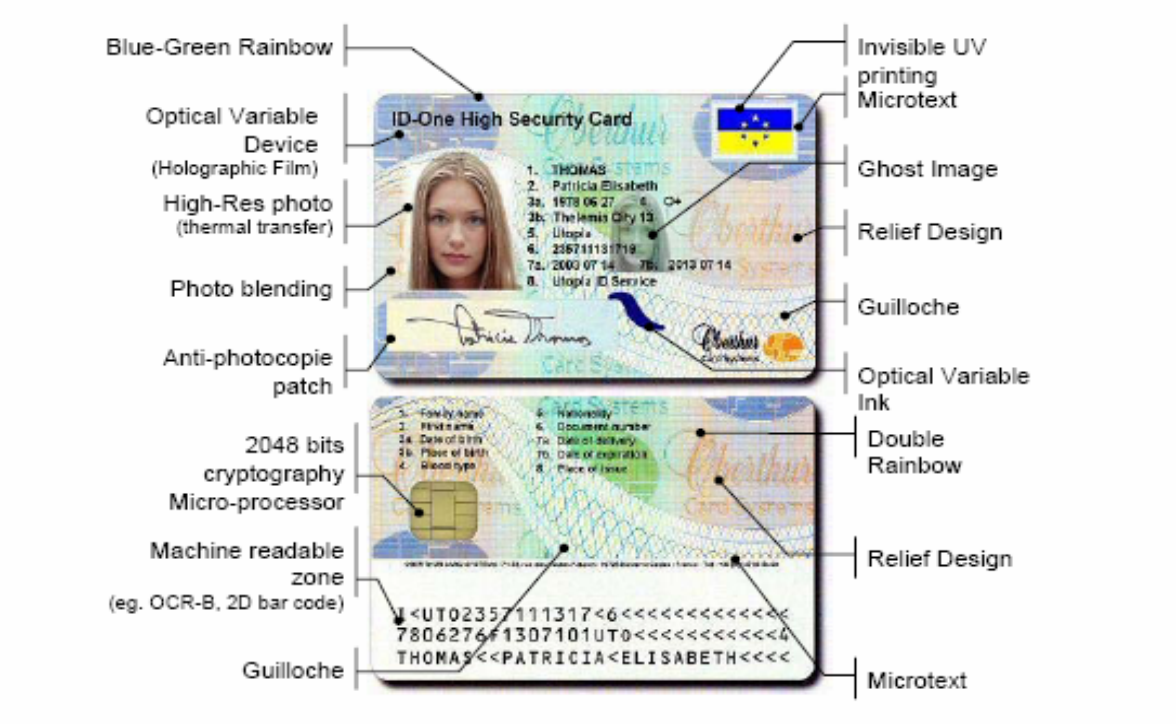


Figure 13. Future High security smart card

Even though it is widely believed through unconfirmed sources that a card with similar features to the one shown above in fig 13 was in use in US DOD as far back as 2001 no commercial version of such a card has as yet been seen the world over. This is therefore considered to be a futuristic idea in smart cards.

The card above uses a plethora of features as listed to authenticate itself.

- Hologram
- Biometrics photograph based
- Anti photocopy patch
- Guilloche
- 2048 bit crpto processor
- Relief design
- Optical variable ink
- Microtext
- Machine readable code
- Blue green rainbow
- UV writing

5.2 Next generation H/W Platform

Enhancements in hardware as shown have led to enhancements in java specifications in terms of VM, security, comm, programming model in future card as listed below.

Older h/w	New h/w
8/16 bit CPU	32 bit CPU
~ 2K RAM	16K RAM
48–64K ROM	>256K ROM
8–32K EEPROM	>128K EEPROM, or Several MB Flash
External Clock: 1–5 MHz	Internal Clock: 50 MHz
Serial I/O Interface 9.6–30K Baud Half Duplex	High Speed Interfaces 1.5 Mb/s–12 Mb/s Full Duplex

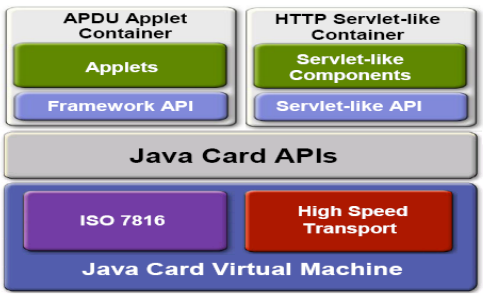
Fig 14. Next Generation Smart Card Platform
Traditional vs. High-end Smart Card Hardware (2007+)

- VM technology
  - 32 bit VM
  - .class file loading
  - Concurrent execution of apps/multithreading
  - On-card byte-code verification

- Security
  - Enhanced context-isolation
  - Policy-based security

- Network-oriented communication
  - ISO 7816 and TCP/IP communication support
  - Communication over USB, MMC
  - Concurrent contact/contact-less card access
  - Embedded web server
  - Service static and dynamic content through HTTP(s)
  - Client and server communication mode
  - Generic communication API

- Programming model
  - Fully backward compatible
  - Support additional Java language types: char, long
  - String support
  - Multi-dimensional arrays
  - Support for large data structures—Multimedia content
  - Application models
    - Classic APDU-based applet model
    - HTTP servlet-like model
  - Enhanced inter-application communication framework
  - Generic event framework
  - Evolutive cryptography framework



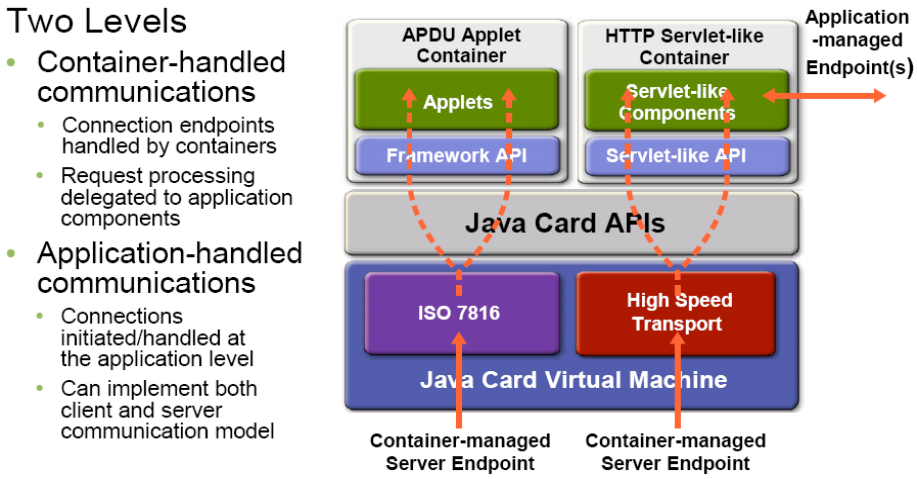


Fig 15. Next Generation Java Card Platform Proposed Software Stack

5.3 No APDU

The APDU has been done away with in next generation smart cards and comm. is based on either USB, MMC, wireless interface vide servelet applets using TCP/IP. APDU is only retained temporarily for backward compatibility reasons.

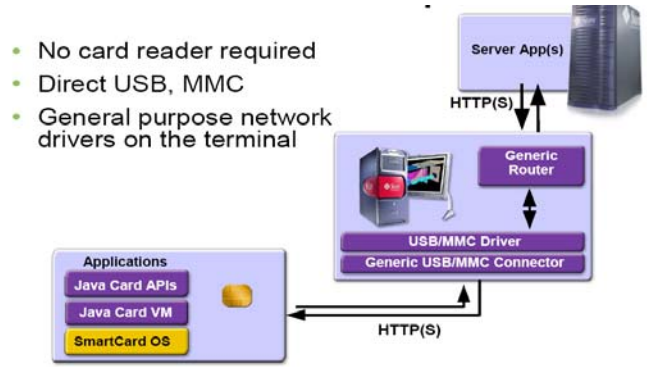


Fig 16. Next generation card

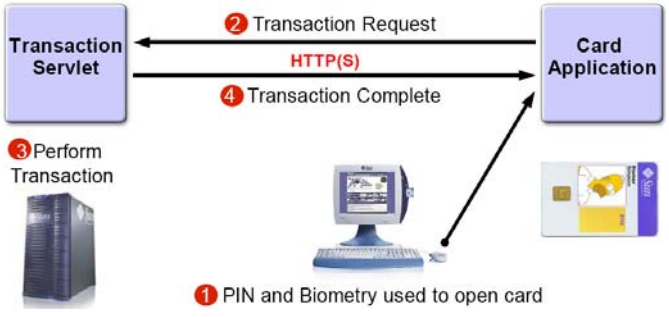
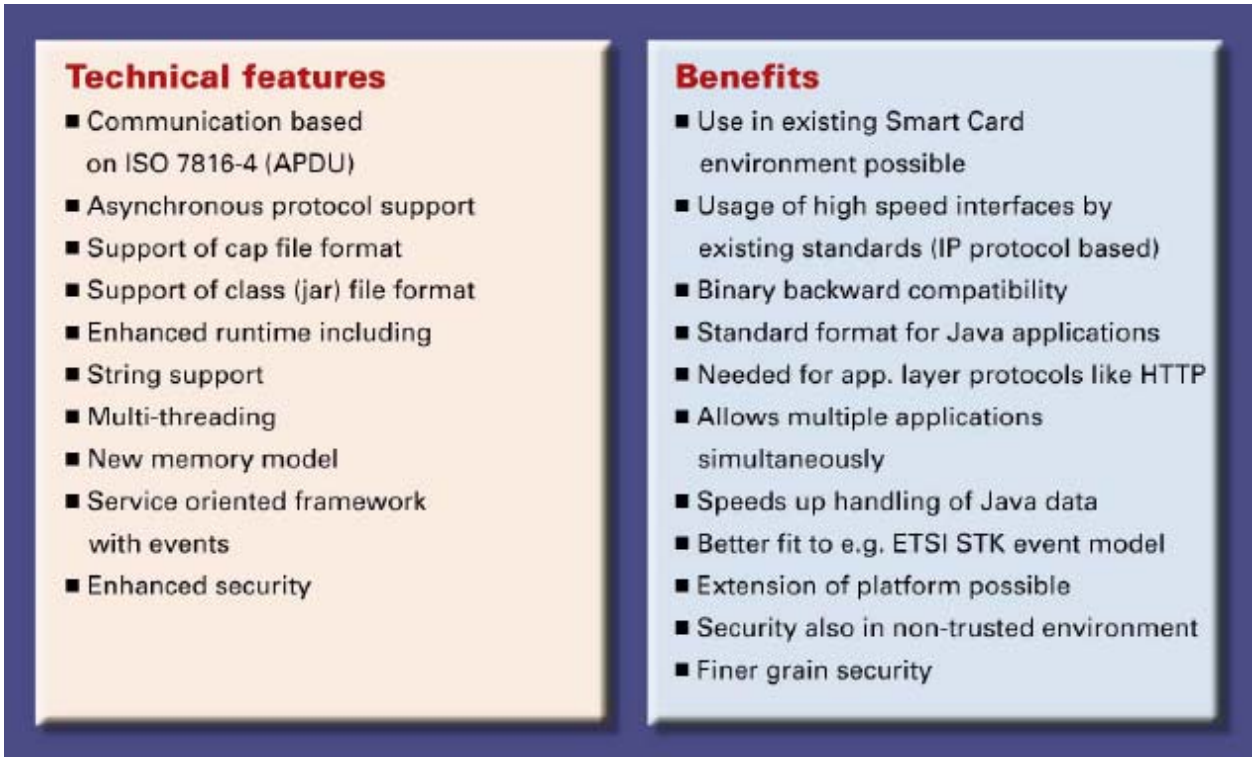


Fig 17. Server Authentication Data Flow

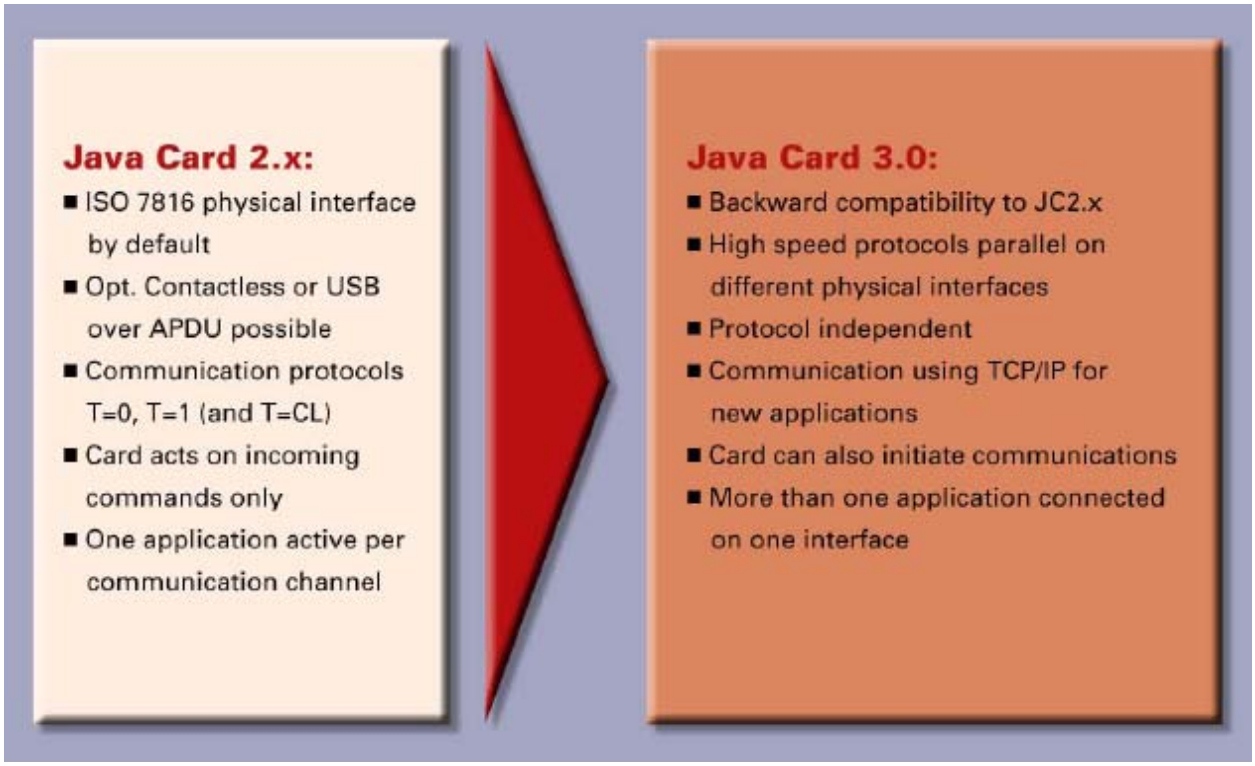
5.4 Java Card 3.x (Future card)

Most of the features have already been mentioned above in this paper. Some Specifications found on Java Card forum based on technical features, Comn details, runtime details, security and API framework and their comparison with Java Card 2 are listed below for reference.

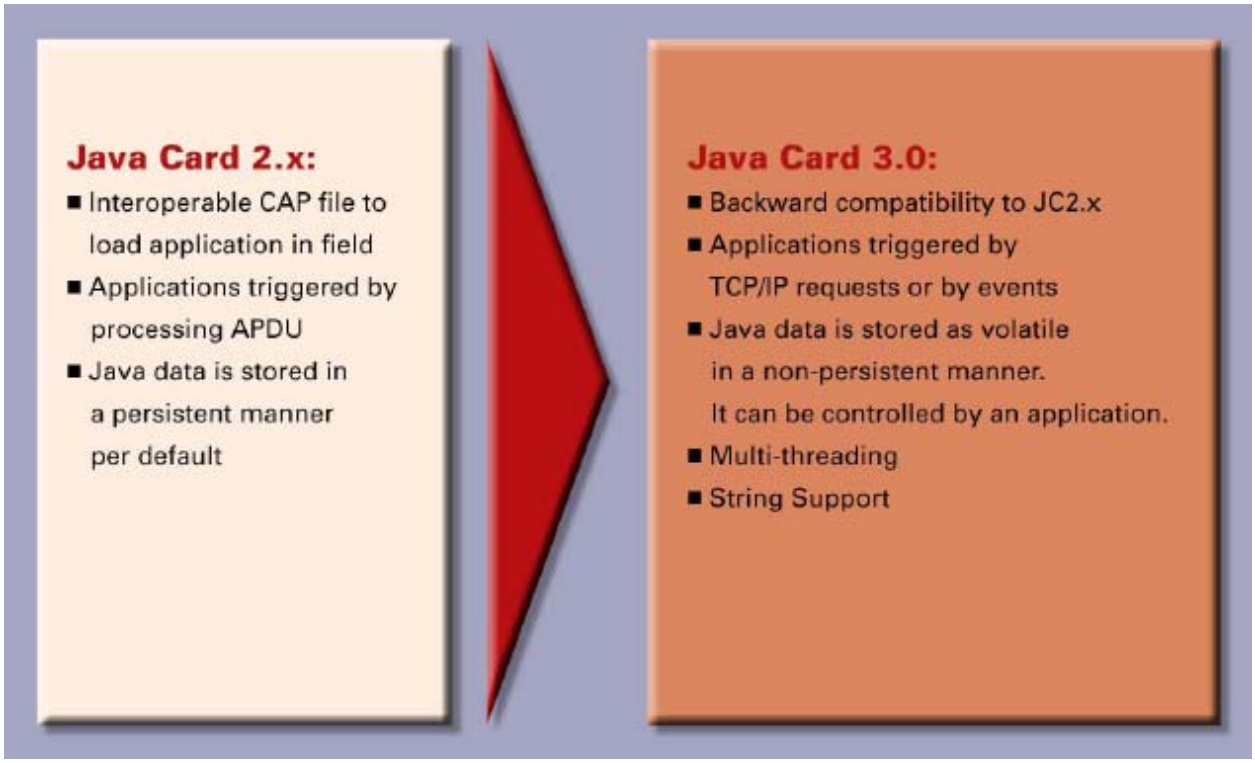
5.5 JAVA CARD 3.0 TECHNICAL OVERVIEW



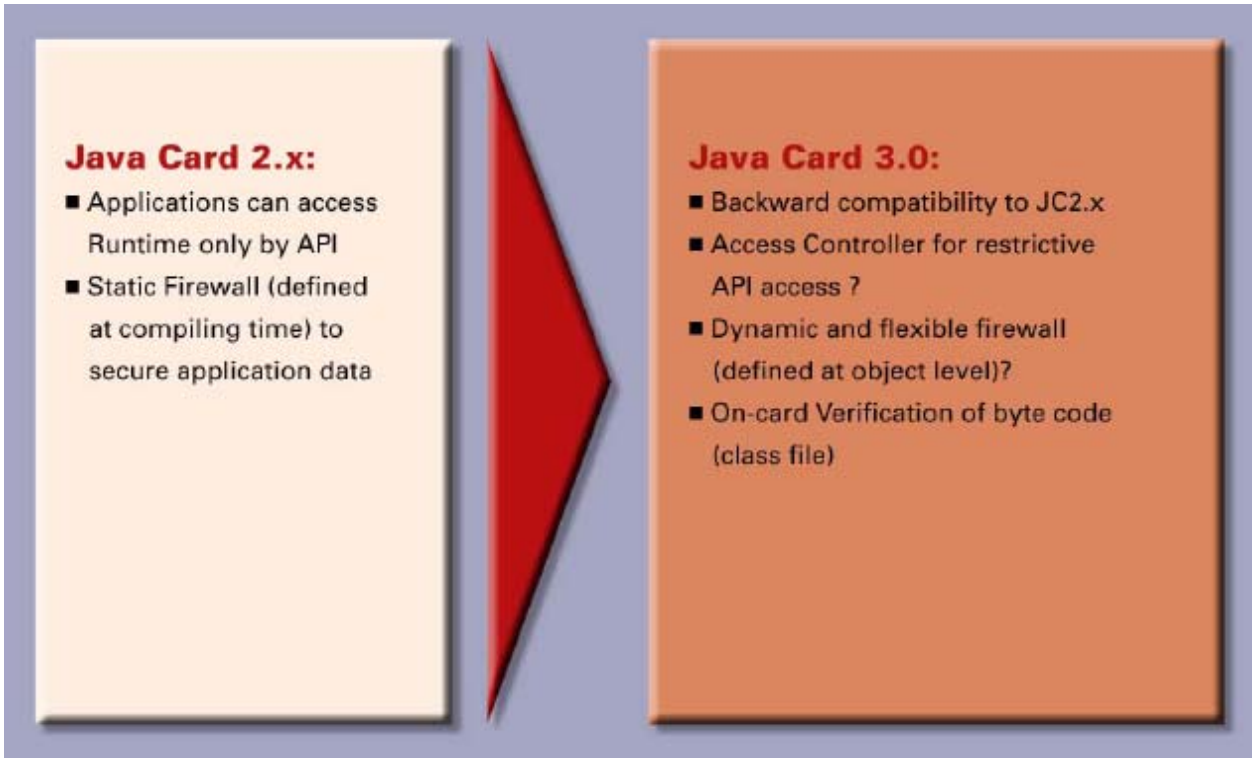
5.6 COMMUNICATION DETAILS



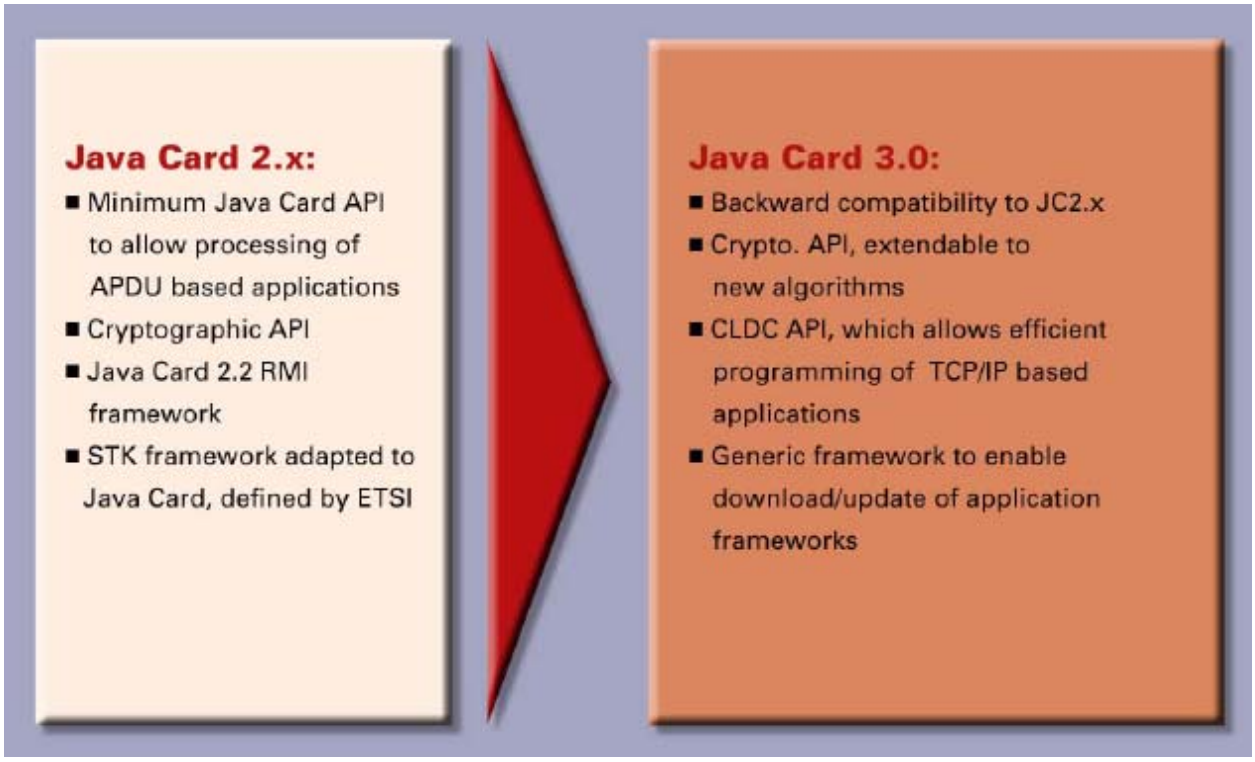
5.7 RUNTIME DETAILS



5.8 SECURITY DETAILS



5.9 APIS+FRAMEWORK DETAILS



CHAPTER-6

CONCLUSION FINDINGS AND FUTURE WORK

6.1 Conclusion

New functionality in the form of smart cards promises to help solve some tough real world problems and address important security concerns. But with new functionality comes new risks. The security dilemma remains: what degree of risk are you willing to take, and to what benefit? At the moment it is evident that Java is the way to go for smart cards. There is no 100% secured/perfect system available for smart cards. A system design and built for minimal attack risk can be treated as secure. Secure system are evaluated/classified in different levels using international standards such as TCSEC/DoD(Orange Book-USA), ITSEC (Europe) and CCITSE (ISO15408).

6.2 Findings

Study of concepts of smart cards with respect to Java Card was a very fine exposure. It gave us insights into the working of such security devices and how they address important security concerns. Java Cards are the next step in Smart Card technology. Basically, a Java Card is a smart card that can be programmed in a high level language instead of assembly language. This naturally makes producing application software much easier. The choice of Java has other advantages: portability, its popularity in the computing community and its design around applets is perfect for smart cards.

6.3 Future Work

Future work should be aimed at reducing the size of the JVM to avoid a split JVM model as being used today. This would mean rewriting Java specifications. These specifications should be as close as possible to mainstream Java such that applet programming becomes easy. A micro linux kind of Operating system can be explored on which JVM can be mounted to enhance security. Future Java Card standards have already been discussed in Chapter 5. Since no single system is 100% secure a future smart card like the one in chapter 5 using a multitude of authentication technologies can be deemed as system with minimal attack risk and can be termed as secure.

References

1. [bs-99] Breaking Up Is Hard To Do: Modeling Security Threats for Smart Cards
   Bruce Schneier Adam Shostack
   Counterpane Systems Netect, Inc.
   schneier@counterpane.com adam@netect.com
   October 19, 1999

2. [ch-8] Chapter 8 Book on 'Securing java'
   Author Wiley

3. [Jc-06] Java Card™ Technology and Tomorrow's Security Architectures
   Tanjore Ravishankar, Aseem Sharma
   Sun Microsystems, Inc.

4. [TS-7136] Gemplus Research Labs TS-7136
   Introducing Research Issues for Next Generation Java-based Smart Card Platforms
   Gilles Grimaud , RD2P Labs, University of Lille, France
   gilles.grimaud@lifl.fr
   Jean-Jacques Vandewalle Gemplus Software Research Labs, France
   jeanjac@research.gemplus.com

5. [Xa-05] Java bytecode verification: algorithms and formalizations Xavier Leroy
   INRIA Rocquencourt and Trusted Logic S.A., Xavier.Leroy@inria.fr
   Java Card™ Platform Security
   Technical White Paper, SUN Microsystems -05

6. [TFT-06] Java Card™ Platform Evolution: Future Directions
   Tanjore Ravishankar, Florian Tournier, Thierry Violleau
   Java Card Team, Sun Microsystems, Inc. 06

7. [TAS-05] Java Card™ Technology and Tomorrow's Security Architectures
   Tanjore Ravishankar, Aseem Sharma
   Sun Microsystems, Inc. Gemplus Research Labs -05

8. [CI-06] New White Card Schemes and Java Card™ Technology
   Eric Vétillard
   CTO Trusted Labs -06
   www.trusted-labs.com

9. [JCF-06] www.javacardforum.org