MULTOS & JAVACARD

White Paper

JAN KREMER
CONSULTING SERVICES

TABLE OF CONTENTS

1. INTRODUCTION

This white paper provides an overview of Smartcard Operating Systems MULTOS and JavaCard

For an overview of the Smartcard based National Identity Cards see my white papers "National Identity Cards White Paper"

1.1. Document Outline

Chapter 1 provides an introduction and outline of this document.

Chapter 2 provides an introduction to Smartcards and the MULTOS and JAVACARD smartcard operating systems

Chapter 3 provides history and product details of both Operating Systems

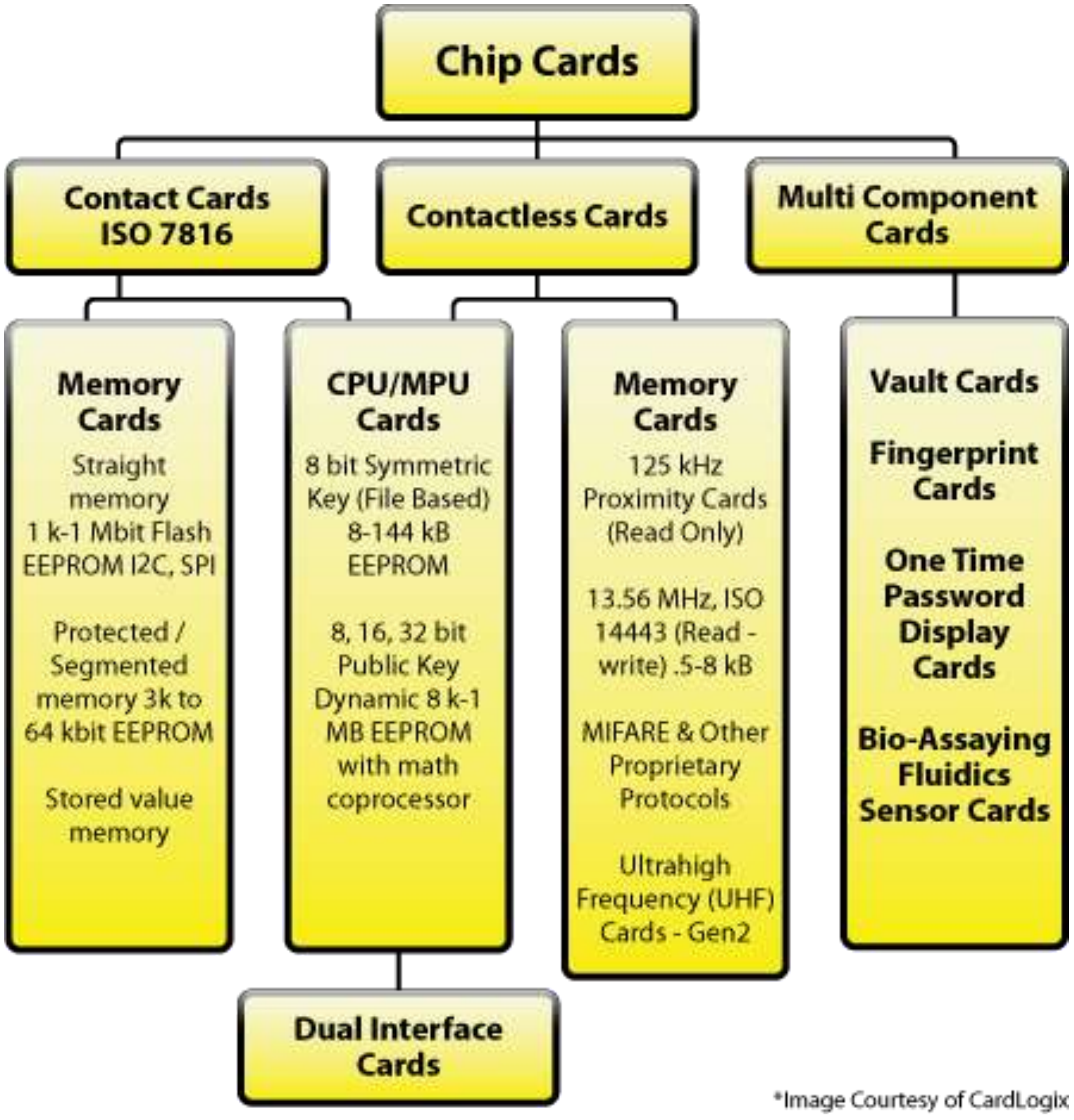
Chapter 4 provides an overview of the security aspects

2. Introduction to MULTOS and JAVACARD

2.1. Introduction

Before we start discussing the smartcard operating systems most commonly used, MULTOS and JavaCard, let's first explain some of the basic smartcard information.

Smart cards are defined according to 1). How the card data is read and written 2). The type of chip implanted within the card and its capabilities. There is a wide range of options to choose from when designing your system.

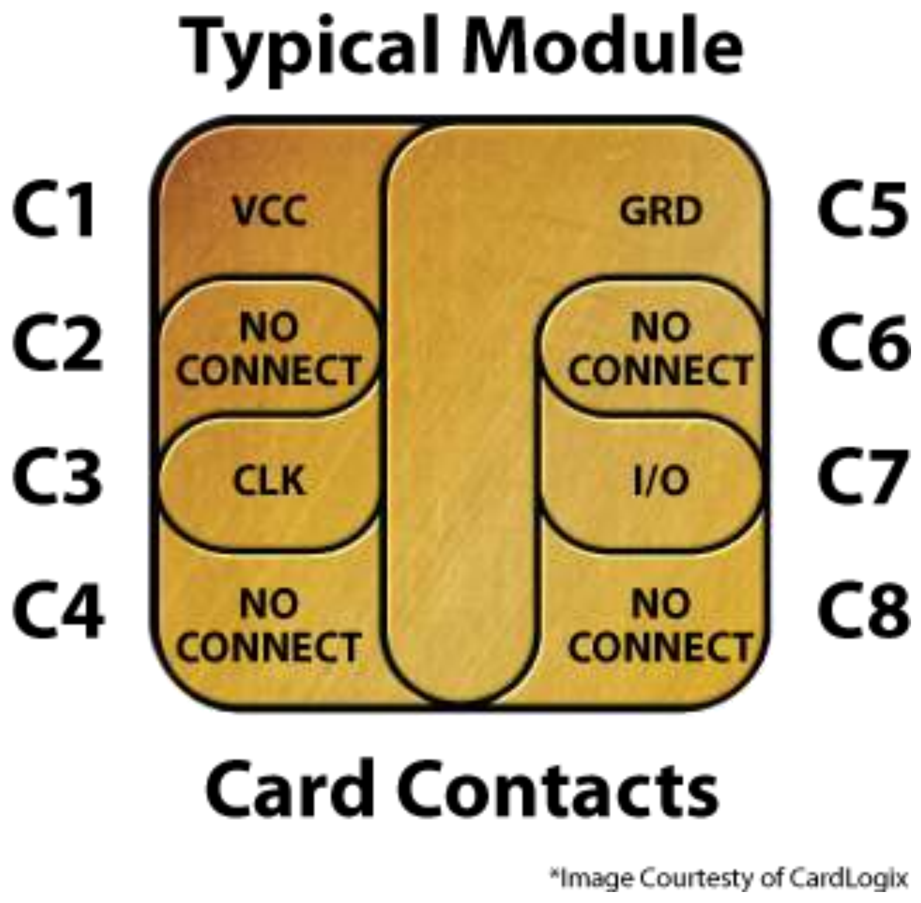

Chip Cards
Contact Cards ISO 7816
Contactless Cards
Multi Component Cards
Memory Cards
Straight memory
1 k-1 Mbit Flash EEPROM I2C, SPI
Protected / Segmented memory 3k to 64 kbit EEPROM
Stored value memory
CPU/MPU Cards
8 bit Symmetric Key (File Based) 8-144 kB EEPROM
8, 16, 32 bit Public Key Dynamic 8 k-1 MB EEPROM with math coprocessor
Memory Cards
125 kHz Proximity Cards (Read Only)
13.56 MHz, ISO 14443 (Read - write) .5-8 kB
MIFARE & Other Proprietary Protocols
Ultrahigh Frequency (UHF) Cards - Gen2
Vault Cards
Fingerprint Cards
One Time Password Display Cards
Bio-Assaying Fluidics Sensor Cards
Dual Interface Cards
*Image Courtesy of CardLogix

2.1.1 Card Construction

Mostly all chip cards are built from layers of differing materials, or substrates, that when brought together properly gives the card a specific life and functionality. The typical card today is made from PVC, Polyester or Polycarbonate. The card layers are printed first and then laminated in a large press. The next step in construction is the blanking or die cutting. This is followed by embedding a chip and then adding data to the card. In all, there may be up to 30 steps in constructing a card. The total components, including software and plastics, may be as many as 12 separate items; all this in a unified package that appears to the user as a simple device.

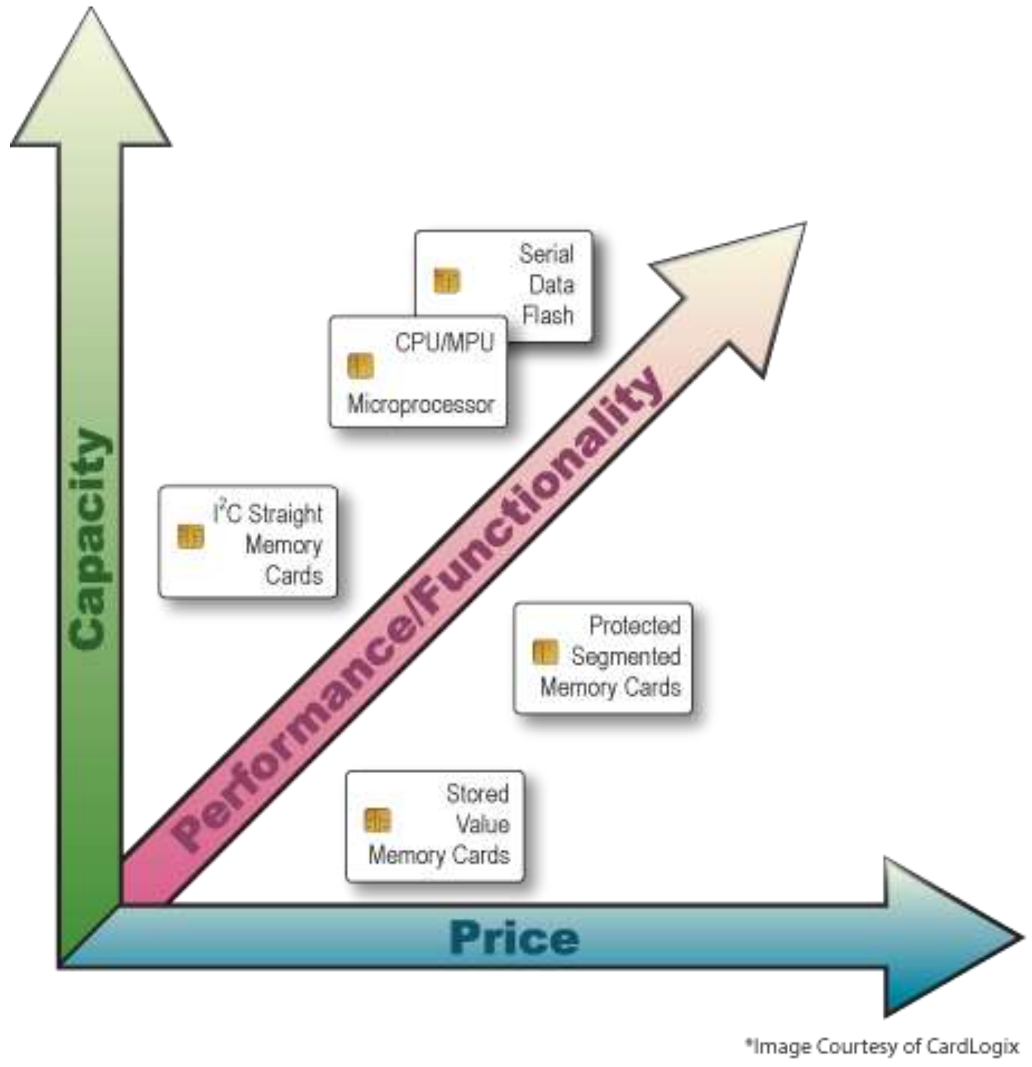
2.1.2 Contact Cards

These are the most common type of smart card. Electrical contacts located on the outside of the card connect to a card reader when the card is inserted. This connector is bonded to the encapsulated chip in the card.



Increased levels of processing power, flexibility and memory will add cost. Single function cards are usually the most cost-effective solution. Choose the right type of smart card for your application by determining your required level of security and evaluating cost versus functionality in relation to the cost of the other hardware elements found in a typical workflow. All of these variables should be weighted against the expected lifecycle of the card. On average the cards typically comprise only 10 to 15 percent of the total system cost with the infrastructure, issuance, software, readers, training and advertising making up the other 85 percent. The following chart demonstrates some general rules of thumb:

2.1.3 Card Function Trade-Offs


Capacity
Price
Performance/Functionality
Serial Data Flash
CPU/MPU Microprocessor
I²C Straight Memory Cards
Protected Segmented Memory Cards
Stored Value Memory Cards
*Image Courtesy of CardLogix

2.1.4 Memory Cards

Memory cards cannot manage files and have no processing power for data management. All memory cards communicate to readers through synchronous protocols. In all memory cards you read and write to a fixed address on the card. There are three primary types of memory cards: *Straight*, *Protected*, and *Stored Value*. Before designing in these cards into a proposed system the issuer should check to see if the readers and/or terminals support the communication protocols of the chip. Most contactless cards are variants on the protected memory/segmented memory card idiom.

2.1.5 Straight Memory Cards

These cards just store data and have no data processing capabilities. Often made with I2C or serial flash semiconductors, these cards were traditionally the lowest cost per bit for user memory. This has now changed with the larger quantities of processors being built for the GSM market. This has dramatically cut into the advantage of these types of devices. They should be regarded as floppy disks of varying sizes without the lock mechanism. These cards cannot identify themselves to the reader, so your host system has to know what type of card is being inserted into a reader. These cards are easily duplicated and cannot be tracked by on-card identifiers.

2.1.6 Protected / Segmented Memory Cards

These cards have built-in logic to control the access to the memory of the card. Sometimes referred to as Intelligent Memory cards, these devices can be set to write-protect some or the entire memory array. Some of these cards can be configured to restrict access to both reading and writing. This is usually done through a password or system key. Segmented memory cards can be divided into logical sections for planned multi-functionality. These cards are not easily duplicated but can possibly be impersonated by hackers. They typically can be tracked by an on-card identifier.

2.1.7 Stored Value Memory Cards

These cards are designed for the specific purpose of storing value or tokens. The cards are either disposable or rechargeable. Most cards of this type incorporate permanent security measures at the point of manufacture. These measures can include password keys and logic that are hard-coded into the chip by the manufacturer. The memory arrays on these devices are set-up as decrements or counters. There is little or no memory left for any other function. For simple applications such as a telephone card, the chip has 60 or 12 memory cells, one for each telephone unit. A memory cell is cleared each time a telephone unit is used. Once all the memory units are used, the card becomes useless and is thrown away. This process can be reversed in the case of rechargeable cards.

2.1.8 CPU/MPU Microprocessor Multifunction Cards

These cards have on-card dynamic data processing capabilities. Multifunction smart cards allocate card memory into independent sections or files assigned to a specific function or application. Within the card is a microprocessor or microcontroller chip that manages this memory allocation and file access. This type of chip is similar to those found inside all personal computers and when implanted in a smart card, manages data in organized file structures, via a card operating system (COS). Unlike other operating systems, this software controls access to the on-card user memory.

This capability permits different and multiple functions and/or different applications to reside on the card, allowing businesses to issue and maintain a diversity of 'products' through the card. One example of this is a debit card that also enables building access on a college campus. Multifunction cards benefit issuers by enabling them to market their products and services via state-of-the-art transaction and encryption technology.

Specifically, the technology enables secure identification of users and permits information updates without replacement of the installed base of cards, simplifying program changes and reducing costs. For the card user, multifunction means greater convenience and security, and ultimately, consolidation of multiple cards down to a select few that serve many purposes.

There are many configurations of chips in this category, including chips that support cryptographic Public Key Infrastructure (PKI) functions with on-board math co-processors or JavaCard with virtual machine hardware blocks. As a rule of thumb - the more functions, the higher the cost.

2.1.9 Contactless Cards

These are smart cards that employ a radio frequency (RFID) between card and reader without physical insertion of the card. Instead, the card is passed along the exterior of the reader and read.

Types include proximity cards which are implemented as a read-only technology for building access. These cards function with a very limited memory and communicate at 125 MHz. Another type of limited card is the Gen 2 UHF Card that operates at 860 MHz to 960 MHz.

True read and write contactless cards were first used in transportation applications for quick decrementing and reloading of fare values where their lower security was not an issue. They communicate at 13.56 MHz and conform to the ISO 14443 standard.

These cards are often protected memory types. They are also gaining popularity in retail stored value since they can speed up transactions without lowering transaction processing revenues (i.e. Visa and MasterCard), unlike traditional smart cards.

Variations of the ISO14443 specification include A, B, and C, which specify chips from either specific or various manufacturers. A=NXP-(Philips) B=Everybody else and C=Sony only chips.

Contactless card drawbacks include the limits of cryptographic functions and user memory, versus microprocessor cards and the limited distance between card and reader required for operation.

2.1.10 Multi-mode Communication Cards

These cards have multiple methods of communications, including ISO7816, ISO14443 and UHF gen 2. How the card is made determines if it is a Hybrid or dual interface card. The term can also include cards that have a magnetic-stripe and or bar-code as well.

2.1.10.1 Hybrid Cards

Hybrid cards have multiple chips in the same card. These are typically attached to each interface separately, such as a MIFARE chip and antenna with a contact 7816 chip in the same card.

2.1.10.2 Dual Interface Card

These cards have one chip controlling the communication interfaces. The chip may be attached to the embedded antenna through a hard connection, inductive method or with a flexible bump mechanism.

2.1.11 Multi-component Cards

These types of cards are for a specific market solution. For example, there are cards where the fingerprint sensor is built on the card. Or one company has built a card that generates a one-time password and displays the data for use with an online banking application. Vault cards have rewriteable magnetic stripes. Each of these technologies is specific to a particular vendor and is typically patented.

2.1.12 Smart Card Form Factors

The expected shape for cards is often referred to as CR80. Banking and ID cards are governed by the ISO 7810 specification. But this shape is not the only form factor that cards are deployed in. Specialty shaped cutouts of cards with modules and/or antennas are being used around the world. The most common shapes are SIM. SD and MicroSD cards can now be deployed with the strength of smart card chips. USB flash drive tokens are also available that leverage the same technology of a card in a different form factor.

2.1.13 Integrated Circuits and Card Operating Systems

The two primary types of smart card operating systems are (1) *fixed file structure* and (2) *dynamic application system*. As with all smartcard types, the selection of a card operating system depends on the application that the card is intended for. The other defining difference lies in the encryption capabilities of the operating system and the chip. The types of encryption are *Symmetric Key* and *Asymmetric Key (Public Key).*

The chip selection for these functions is vast and supported by many semiconductor manufacturers. What separates a smart card chip from other microcontrollers is often referred to as trusted silicon.

The device itself is designed to securely store data withstanding outside electrical tampering or hacking. These additional security features include a long list of mechanisms such as no test points, special protection metal masks and irregular layouts of the silicon gate structures. The trusted silicon semiconductor vendor list below is current for 2010:

- Atmel
- EM Systems
- Infineon
- Microchip
- NXP (Philips)
- Renesas Electronics
- Samsung
- Sharp
- Sony
- ST Microelectronics

Many of the features that users have come to expect, such as specific encryption algorithms have been incorporated into the hardware and software libraries of the chip architectures. This can often result in a card manufacturer not future-proofing their design by having their card operating systems only ported to a specific device.

Care should be taken in choosing the card vendor that can support your project over time as card operating system-only vendors come in and out of the market. The tools and middleware that support card operating systems are as important as the chip itself. The tools to implement your project should be easy to use and give you the power to deploy your project rapidly.

2.1.14 Fixed File Structure Card Operating System

This type treats the card as a secure computing and storage device. Files and permissions are set in advance by the issuer. These specific parameters are ideal and economical for a fixed type of card structure and functions that will not change in the near future.

Many secure stored value and healthcare applications are utilizing this type of card. An example of this kind of card is a low-cost employee multi-function badge or credential. Contrary to some biased articles, these style cards can be used very effectively with a stored biometric component and reader. Globally, these types of microprocessor cards are the most common.

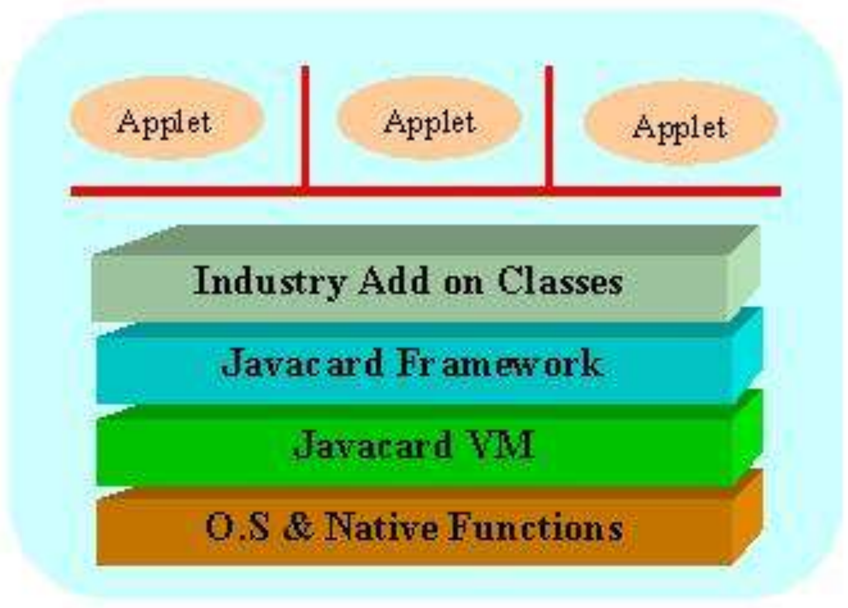
2.1.15 Dynamic Application Card Operating System

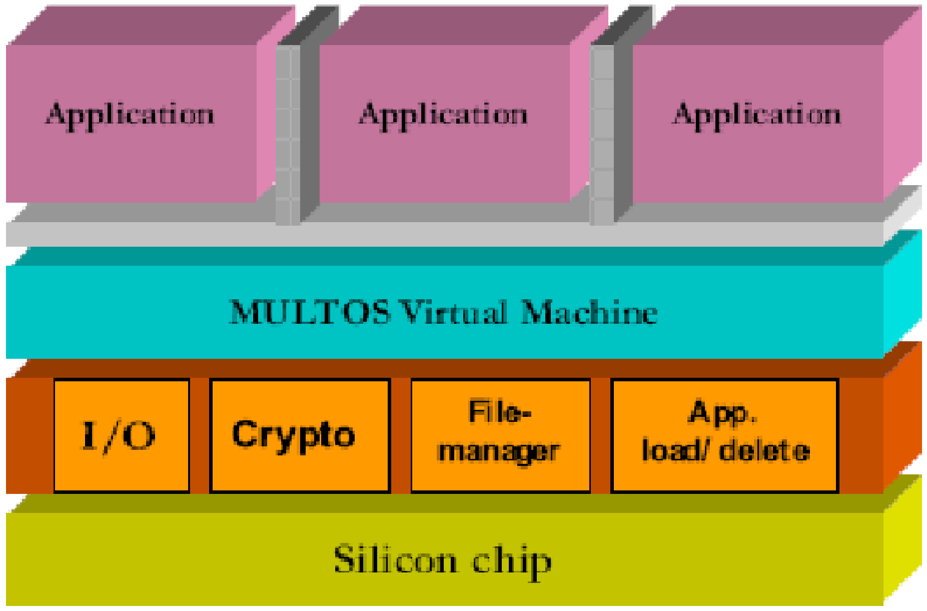
This type of operating system, which includes the JavaCard and proprietary MULTOS card varieties, enables developers to build, test, and deploy different on card applications securely. Because the card operating systems and applications are more separate, updates can be made. An example card is a SIM card for mobile GSM where updates and security are downloaded to the phone and dynamically changed. This type of card deployment assumes that the applications in the field will change in a very short time frame, thus necessitating the need for dynamic expansion of the card as a computing platform. The costs to change applications in the field are high, due to the ecosystem requirements of security for key exchange with each credential. This is a variable that should be scrutinized carefully in the card system design phase.

Currently, MULTOS and JavaCard are the two main OS standards for a smartcard. The emergence of these OSs in the late 1990's resulted in JavaCard receiving backing from Visa while MULTOS was endorsed by MasterCard. Back in 2004, the OSCIE standard only considered these two operating systems as options for a smart card design.

JavaCard versus MULTOS

JavaCard was a natural progression from the growing use of the Java programming language in the 1990s while MULTOS was developed as an operating system for an electronic purse system and so the emphasis was always focused on high security to ensure fraudulent financial transactions could not occur.

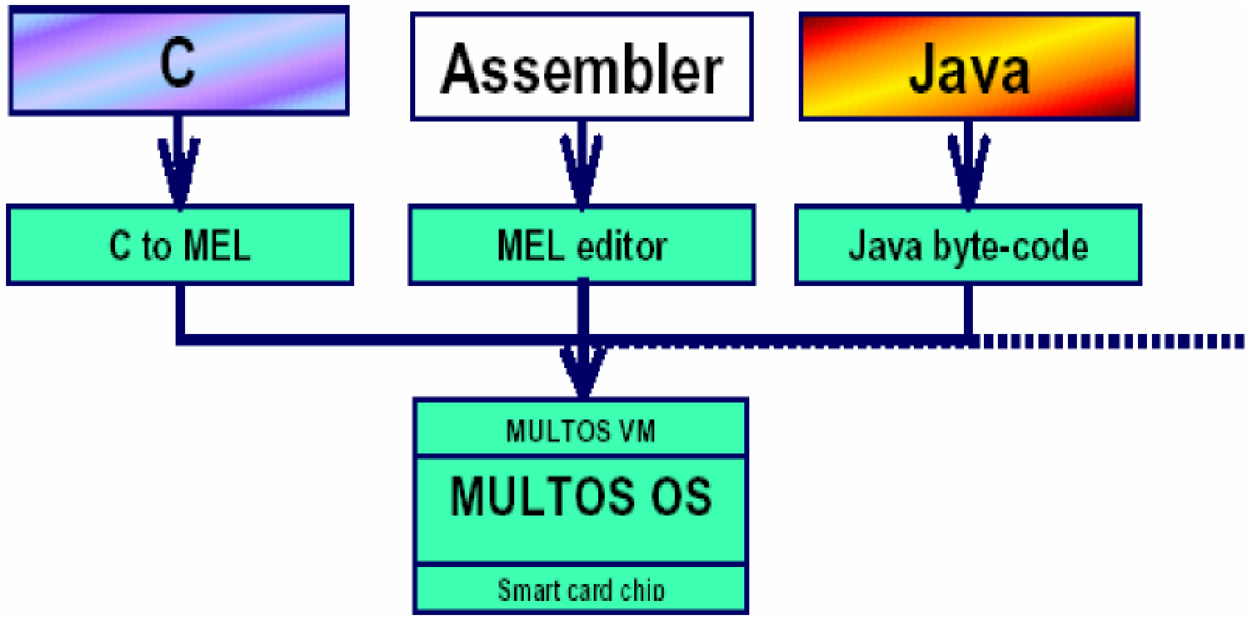




One of the security features of a smartcard mentioned in the table below relates to applet/application loading and deleting. This is one of the key differences between the JavaCard and MULTOS card.

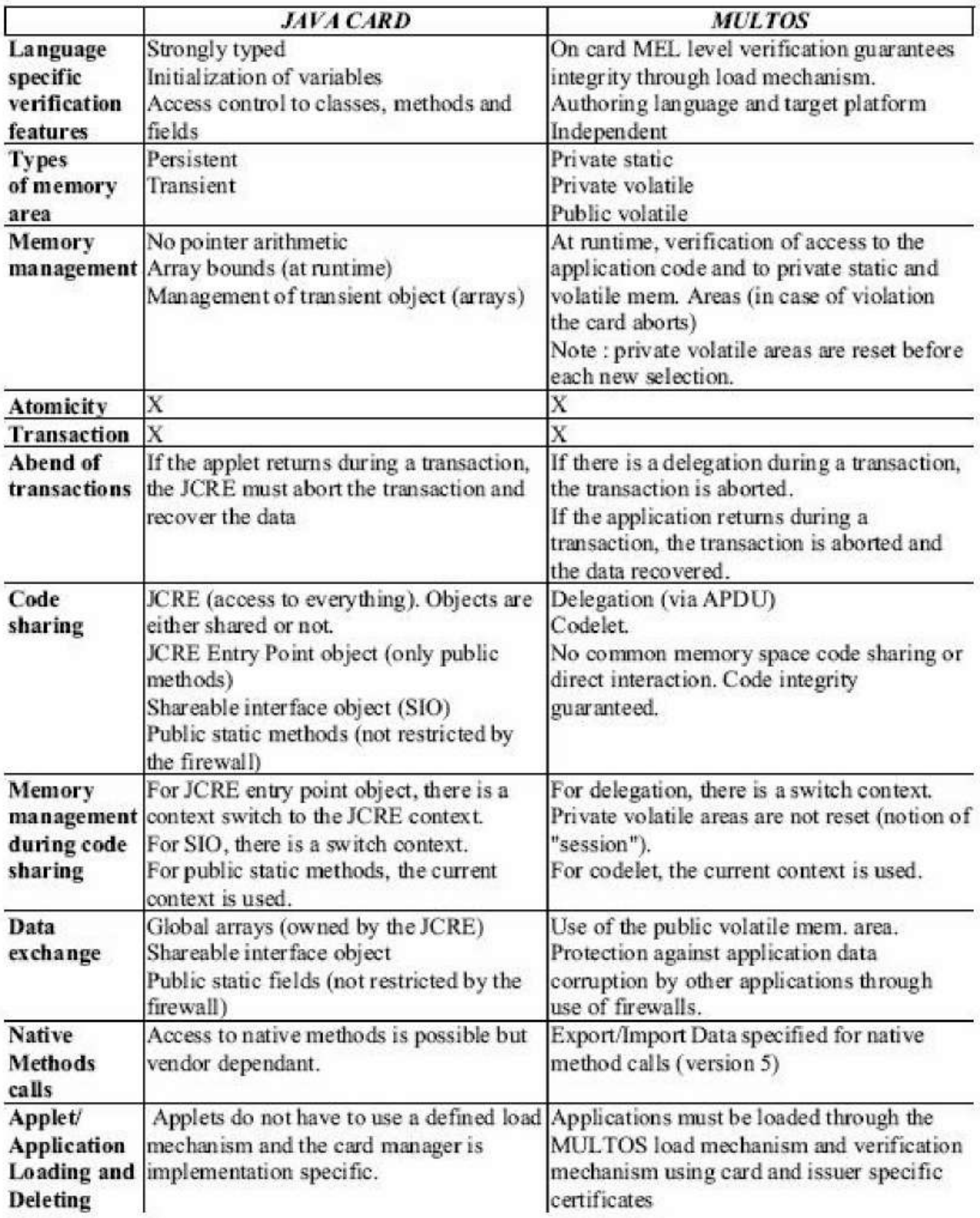
The JavaCard has no requirement for applets to use a defined load mechanism. Compare this to the MULTOS card where applications must be loaded through its specific load mechanism.

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



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.

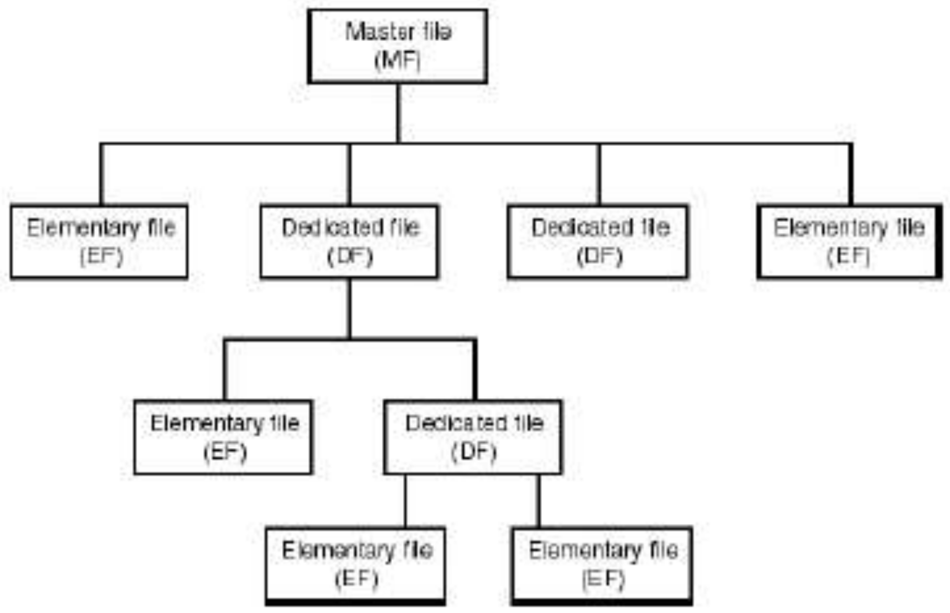
MULTOS is designed and independently evaluated to a high level of security to ensure that issuers, application developers and other MULTOS service providers can build their business proposition without having to undertake expensive and lengthy evaluations of the underlying technology.



	JAVA CARD	*MULTOS*
Language specific verification features	Strongly typed Initialization of variables Access control to classes, methods and fields	On card MEL level verification guarantees integrity through load mechanism. Authoring language and target platform Independent
Types of memory area	Persistent Transient	Private static Private volatile Public volatile
Memory management	No pointer arithmetic Array bounds (at runtime) Management of transient object (arrays)	At runtime, verification of access to the application code and to private static and volatile mem. Areas (in case of violation the card aborts) Note : private volatile areas are reset before each new selection.
Atomicity	X	X
Transaction	X	X
Abend of transactions	If the applet returns during a transaction, the JCRE must abort the transaction and recover the data	If there is a delegation during a transaction, the transaction is aborted. If the application returns during a transaction, the transaction is aborted and the data recovered.
Code sharing	JCRE (access to everything). Objects are either shared or not. JCRE Entry Point object (only public methods) Shareable interface object (SIO) Public static methods (not restricted by the firewall)	Delegation (via APDU) Codelet. No common memory space code sharing or direct interaction. Code integrity guaranteed.
Memory management during code sharing	For JCRE entry point object, there is a context switch to the JCRE context. For SIO, there is a switch context. For public static methods, the current context is used.	For delegation, there is a switch context. Private volatile areas are not reset (notion of "session"). For codelet, the current context is used.
Data exchange	Global arrays (owned by the JCRE) Shareable interface object Public static fields (not restricted by the firewall)	Use of the public volatile mem. area. Protection against application data corruption by other applications through use of firewalls.
Native Methods calls	Access to native methods is possible but vendor dependant.	Export/Import Data specified for native method calls (version 5)
Applet/ Application Loading and Deleting	Applets do not have to use a defined load mechanism and the card manager is implementation specific.	Applications must be loaded through the MULTOS load mechanism and verification mechanism using card and issuer specific certificates

2.1.16 Smartcard 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.



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.

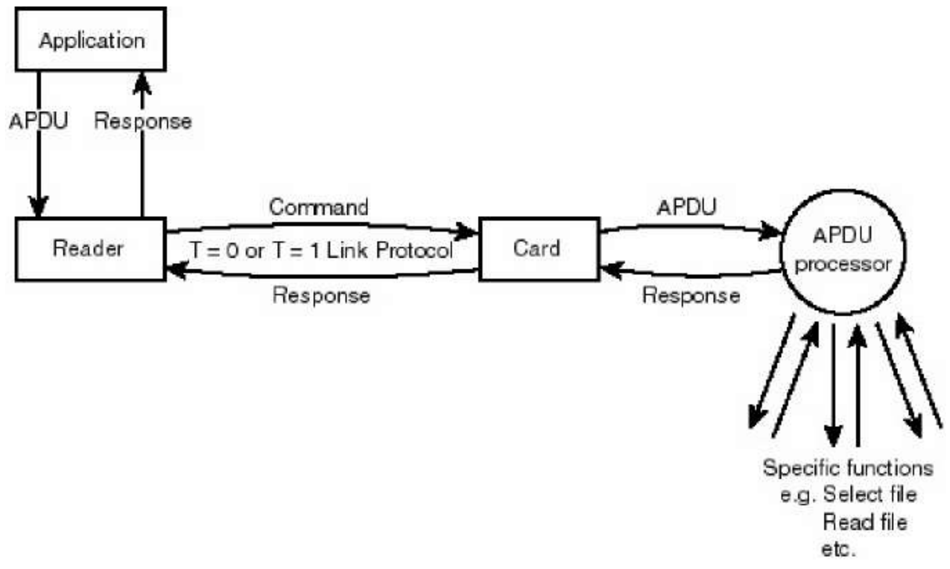
2.1.17 Application Protocol Data Units (APDUs)

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 78 16-4 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.

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 below

.



3. MULTOS and JAVACARD Details

3.1. MULTOS Details

3.1.1 MULTOS History

Since its inception in the early nineties MULTOS has gained a reputation as the most robust, flexible and secure smart card platform in the market. The multi-application operating system enables a smart card to carry a variety of applications, from chip and pin payment to secure ID and ePassport. Previously, application developers had to write a separate version of the application for each type of smart card. MULTOS changed all that. Whereas earlier systems did not allow new applications to be installed, MULTOS enabled several applications to reside at once, regardless of microchip used.

Today, millions of MULTOS smart cards have been issued across a range of industries including contact less payment, Internet authentication, biometrics and healthcare. More than 75 issuers are now committed to issuing cards on the MULTOS platform, as well as 70 companies supplying MULTOS-related products and services. It’s fair to say MULTOS is in demand. Earlier this month, French giants Gemalto acquired MULTOS, the latest in a long line of companies battling to gain control of a seriously lucrative asset.

The smart card platform has a multitude of benefits. For starters, it’s the only operating system for smart cards to have been certified with the prestigious ITSEC Level E6 security rating, the highest available. MULTOS also boasts true interoperability, allowing multiple vendors to supply components and the ability to load and delete multiple applications. MULTOS is also ably supported by several of the industry’s leading organizations known as the MULTOS consortium, a group of international blue chip companies whose objective is to promote MULTOS as the smart card industry standard across all market sectors.

Natwest Group and Platform Seven

In 1993 the Natwest Development Team (NWDT), a department led by Smartcard News founder David Everett, invented MULTOS. It was originally developed to support the Mondex International e-purse application. The NWDT undertook the ground breaking technical design, development and implementation of the Mondex electronic purse and other Mondex devices. The team played a pivotal role in one of the most security demanding projects ever when they developed a smart card operating system, brought to the market as MULTOS.

Then, Natwest Group decided to exploit its extensive knowledge of security in the smart card industry with the creation of Platform Seven, a UK-based centre offering secure component products and a wide range of services for the e-commerce market. Members of the new unit were drawn from the former Natwest Development Team.

Mondex International

In 1996, MasterCard purchased a 51% stake in Mondex, promising to promote MULTOS and support the technology's development. Although no financial details were disclosed, the gross value of Mondex was estimated at around $100 million. The merger proved a great success and three years later the MULTOS chip became the first commercial product ever to achieve a security rating of ITSEC level e6. In the same year, Mondex International set its sights on conquering the Internet, with intentions to become one of the first online e-cash services.

Having become a major global firm in the e-cash business Mondex agreed a deal with ActivCard to develop secure ID and access information for the MULTOS smart card environment. In return, ActivCard's access tools technology would transfer over to MULTOS enabling Mondex e-cash cards to arrive with e-commerce facilities already built in.

MasterCard

By 2001, MasterCard International assumed full ownership of Mondex, snatching direct control of all the companies operations, most notably MULTOS. Under the agreement, Mondex continued to provide service to the MULTOS consortium, MasterCard's preferred operating system for multi-application smart cards. However, despite it's previous success MULTOS experienced fresh difficulties under the leadership of MasterCard. Once a contender to become the standard operating system for smart cards MULTOS now found itself in fierce competition with Java Card, a US based operating system threatening to dominate the market.

The MULTOS consortium announced plans to develop a low-cost version of the operating system aimed at banks gearing up for the inevitable conversion to smart cards known as 'Step-one'. By providing a cheaper stepping-stone to fully-fledged MULTOS, the consortium hoped the project would convince card issuers to embrace the technology without forcing them to bear the cost and complexity of a full rollout all at once.

MULTOS backers characterized the move as a logical strategy that would allow financial institutions to make a more gradual transition to MULTOS based chip cards. Conversely, critics suggested it was a futile effort to keep MULTOS relevant in a market increasingly saturated by Java Card systems. It’s easy to see why this assumption was made. In 2002, the industry shipped 5-million microprocessor cards carrying MULTOS property compared to a staggering 170 million cards with Java Card properties.

One reason for this was Java card software governing the mobile phone SIM market. Since MULTOS was not a player in such a market it was crucial to consolidate its position in banking if it were to survive in the long term

StepNexus

In 2006, a deal to move control of the MULTOS smart card operating system from MasterCard to a new consortium of companies was completed. The holding company revealed its new corporate identity, StepNexus Inc. The name was derived from Secure Trusted Environment Provisioning, with Nexus providing the connection between the technology and the consumer.

In a partnership with Hitachi, Keycorp and MasterCard StepNexus acquired full jurisdiction over MULTOS technology. John Wood overlooked the consortium. Today he is the CEO of MAOSCO Ltd, a legal company set up to promote and develop the MULTOS specifications as an open industry standard.

Aiming to reverse the fortunes of MULTOS StepNexus devised several initial solutions with the main focus being on security and flexibility of the product. These were:

- To make MULTOS the world’s standard secure multi-application smart card operating system. This was implemented using a similar project to ‘Step-one’ although this time StepNexus particularly targeted the SDA payment markets.
- To reach new markets and sustain current ones

The solution aimed to extend to other environments where users were ‘seeking to benefit from enhanced trust and security’

StepNexus announced the acquisition of GlobalPlatform on MULTOS technology from Sentry, a privately owned Sydney-based company. The application enabled any MULTOS chip to support GlobalPlatform services and meant that the flagship platform MULTOS could provide compatibility with existing Global Platform systems.

Keycorp

In mid 2008 Australian-based Keycorp bought out the UK subsidiaries of StepNexus and subsequently acquired MULTOS as well as intellectual property rights and registered patents. As part of the deal Keycorp relinquished its existing shareholding in StepNexus and was forced to pay the holding company $1million. After the completion of the deal Keycorp owned the majority of shares, although former StepNexus partners Hitachi and Mondex International retained 20% of the shares in the new venture.

Gemalto

Keycorp's reign was short-lived. By September Gemalto confirmed it had attained all Keycorp smart card business assets including trademarks, IP portfolio and of course, MULTOS, and all for a cool £22 million. As well as taking over the operating system Gemalto have also bagged the Key Management Authority (KMA) that manages MULTOS cards worldwide. Around 40 technical experts will join the company; most are based in the UK and Australia. Gemalto hope the deal will contribute over $15 million extra revenue to the secure transactions and government program segments of Gemalto on an annual basis.

3.1.2 MULTOS OS

MULTOS is a multi-application smart card operating system, that enables a smart card to carry a variety of applications, from chip & pin application for payment to on-card biometric matching for secure ID and ePassport. MULTOS is an open standard whose development is overseen by the MULTOS Consortium - a body compromised of companies which have an interest in the development of the OS and includes smart card and silicon manufacturers, payment card schemes, chip data preparation, card management and personalization system providers, and smart card solution providers.

One of the key differences of MULTOS with respect to other types of smart card OS, is that it implements *Secure Trusted Environment Provisioning* or *STEP*, which is a patented mechanism by which the manufacture, issuance and dynamic updates of MULTOS smartcards in the field is entirely under the issuer's control. This control is enforced through the use of a **Key Management Authority** (KMA). The KMA provides card issuers with cryptographic information required to bind the card to the issuer, initialize the card for use, and generate permission certificates for the loading and deleting of applications under the control of the issuer.

Millions of MULTOS smart cards are being issued by banks and governments all around the world, for projects ranging from contactless payment, internet authentication and loyalty, to national identity with digital signature, ePassport with biometrics, healthcare and military base and network access control

A MULTOS implementation provides an operating system upon which resides a virtual machine. The virtual machine provides:

- ❖ Application run-time environment.
- ❖ Memory management.
- ❖ Application loading and deleting.

3.1.2.1 Run-time Environment

The run-time environment operates within the application space. This consists of code space and data space. The code is assembled and is interpreted every time it is executed. The virtual machine performs code validity and memory access checks. The data space is divided into static and dynamic portions.

The key component of dynamic memory is the last in, first out (LIFO) stack as this makes using the various functions much easier. A MULTOS chip is a stack machine, which makes use of this dynamic memory to pass parameters and perform calculations. In addition, the Input/output buffer resides in another dynamic memory segment.

3.1.2.2 Memory Management

Each application resides with a rigorously enforced application memory space, which consists of the application code and data segments. This means that an application has full access rights to its own code and data, but can not directly access that of another application. If an application attempts to access an area outside its space, it results in an abnormal end to processing.

3.1.2.3 Application Loading and Deleting

A MULTOS card permits the loading and deleting of applications at any point in the card's active life cycle. A load can take place once the application and its corresponding certificate are transmitted to the chip. A delete is permitted if a certificate that corresponds to a loaded application is transmitted to the chip.

3.1.2.4 More Information

MULTOS smart card technology delivers high security, interoperable platforms for any application. MULTOS consists of two unique technologies that deliver the secure architecture - the on-card virtual machine that securely executes applications and the MULTOS security scheme, an implementation of STEP technology, that secures the smart card, application code and application data.

3.1.2.5 Secure Multi-applications

MULTOS applications are developed in high-level languages such as 'C' or Java (or in low-level assembly language) and compiled into MEL bytecodes that are executed by the virtual machine.

When an application executes, the virtual machine checks each and every bytecode instruction to ensure it is valid and properly formed. All memory areas accessed by the instructions are also checked that they are within the memory area of that application. Any invalid instructions or attempted memory accesses are rejected by the virtual machine and all smart card application execution will stop.

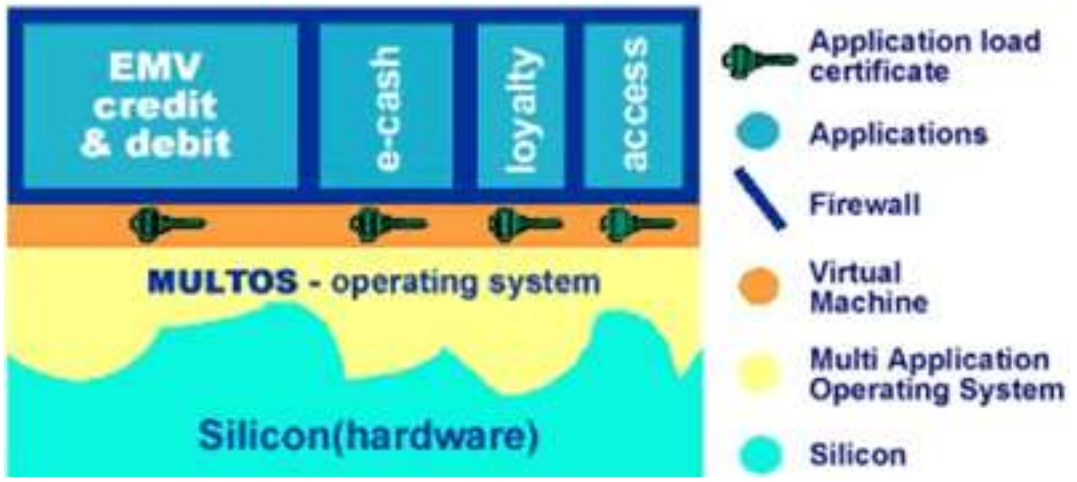
The execution-time checking ensures the complete safety of application execution and data - it is not possible for an application to access the data of another application on the smart card. As application data sharing is not permitted, application providers can be assured that their data is safe from other applications that may reside alongside theirs in the smart card.

The MEL bytecode instruction set is limited to data manipulation and simple arithmetic operations; however MULTOS operating systems provide a wide range of additional built-in functions, termed Primitives that provide more complex operations such as cryptography or operating system data access. The same memory access checking applies to memory areas manipulated by the primitives; ensuring applications cannot even unintentionally access memory outside their permitted space.

3.1.2.6 Interoperability Defined and Delivered

All MULTOS OS Implementations include the standard Virtual Machine and a standard set of Primitive functions. There are a number of optional Primitive functions, usually related to specific hardware features that may be present, such as a contactless interface.

This ensures that applications are 100% compatible between different MULTOS and MULTOS step/one products from different vendors. All products undergo rigorous Type Approval to ensure compliance with specification and security of implementation.



3.2. JAVACARD Details

In June 2005, one billion Java Card cards had been sold; more than any single smart card operating system has achieved before. This result is due partly to socio-economic circumstances, serendipity, and, as we will argue, in a large part to the technical excellence of Java Card.

In the early nineties, conceived as a third party smart card OS under the code name MASS and later marketed as ‘Macsime,’ before adopting its present name, the Java Card concept was endowed with a number of features. Some of these have been *realized* in the product and have contributed to the commercial success. Other features, present in the implemented prototype of Macsime, have yet to be introduced in Java Card products, and, as we will argue, are entirely within the Java Card spirit:

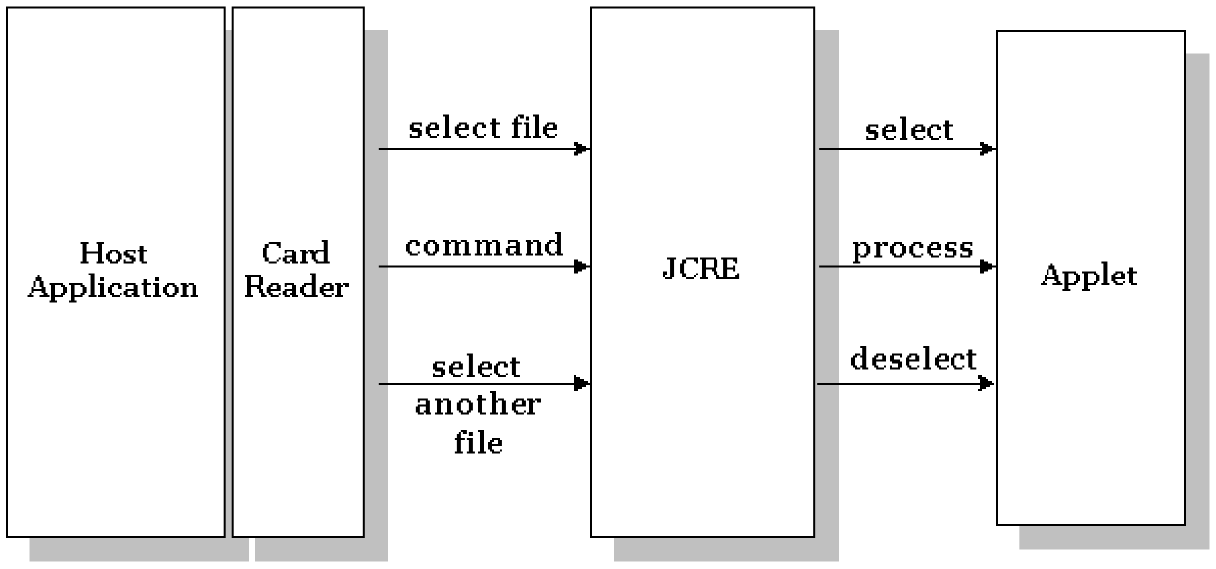
To deliver the core of a card operating system that is easy to use, secure and amenable to formal analysis. A third set of *complementary* features has been proposed, which may be useful for specific applications. We discuss each of these three categories of features.

Java Card products have *realized* a number of innovative features, including the ability to program smart card applications in a high level language (a subset of the Java language), the ability to upgrade a Java Card smart card with new applications after the card has been issued, and strong security mechanisms such as fine-grain access control and strict applet separation. These mechanisms rely on the fact that Java has built-in concepts of security, and that it is object oriented; a Java Card programmer thus uses object oriented design methods to develop the applet code and possibly also the terminal code, supported by programming tools, such as emulators and debuggers.

Java Card technology was *intended* to be used in what is now called a model-based approach. This means that the starting point for any development is a conceptual model of the system under development, in this case which the model consists contains concepts such as persistence, transactions, authentication, authorization and integrity and the establishment of trust in a business transaction. The foundations for the model based analysis of Java Card applications have already been laid in our earlier work on the transaction model, the three-phase interaction protocol and the card/terminal co-design development model.

Depending on the area of deployment, the Java Card specifications could be enhanced with further, *complementary* features, such as the provision of an IP address or the addition of web-server functionality, interfacing to peripherals such as keyboards and displays, or the integration of Biometrics on the card.

The object-oriented core is well understood and fully embraced by the industry. The main elements of the conceptual model have been proposed earlier, but the present paper emphasizes the integration of these elements into a coherent model. Secondly, we offer a comparison of smart card operating systems and the features they provide showing how the developments of each system contribute to Java Card features. Thirdly, we present a time line showing when Java Card technology and its conceptual model emerged, and who were responsible.



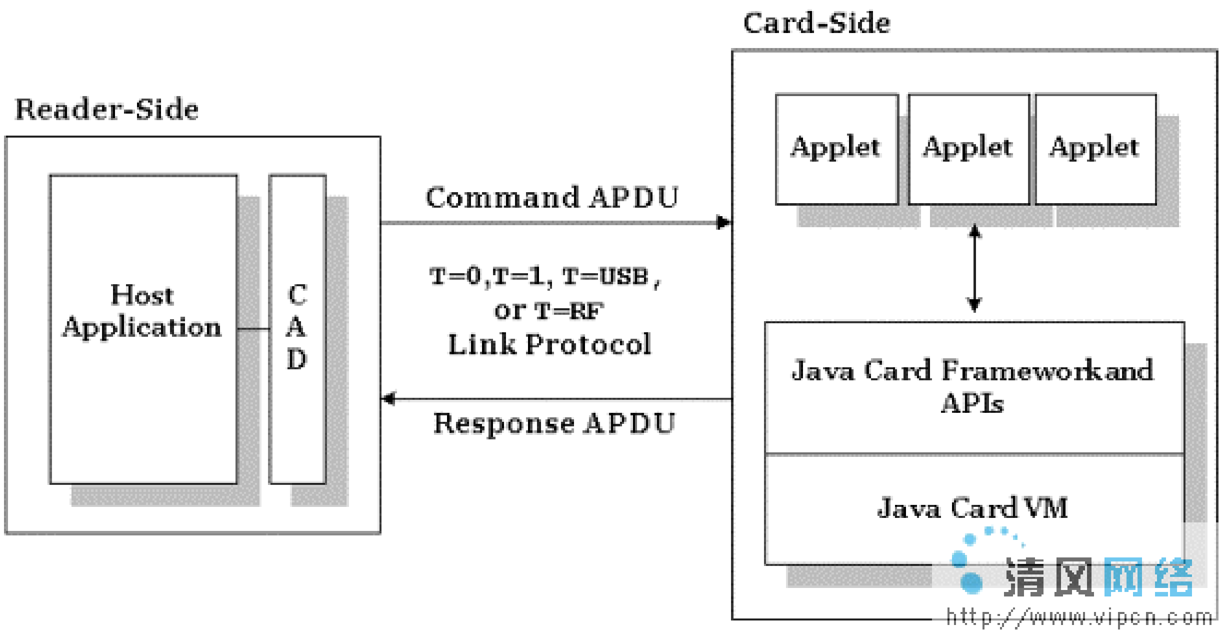
Java Card applets are small ‘programs’, that can communicate to each other and the terminal, that are developed in a high level language, and then either pre loaded, or, more interestingly, loaded into a smart card once it has been fielded. Other smart card operating systems allow development in high level languages (e.g. Multos, Basic card), post issuance update (e.g. Carte Blanche), and secure communication between applets (e.g. Firewall patent). The innovation of the Java Card technology is that it supports a combination of these features all exploiting the fact that it is object oriented.

Loading an applet means loading a class (which once instantiated becomes an object in its own right). The security of applet loading is intimately connected with access control to the class and the resulting object. This idea can be traced directly back to Java itself, which offers secure loading of classes. The case for post issuance update of card functionality has been demonstrated with the Blank Card concept of Peltier

The Java Card specification itself does not cover life cycle management. With the hindsight of its success it is clear that a practical business case for dynamic cards involves an active card issuer that guards access to the card. The Visa Open Platform, later Open Platform, was introduced by Visa international to implement this commercially critical role of the card issuer.

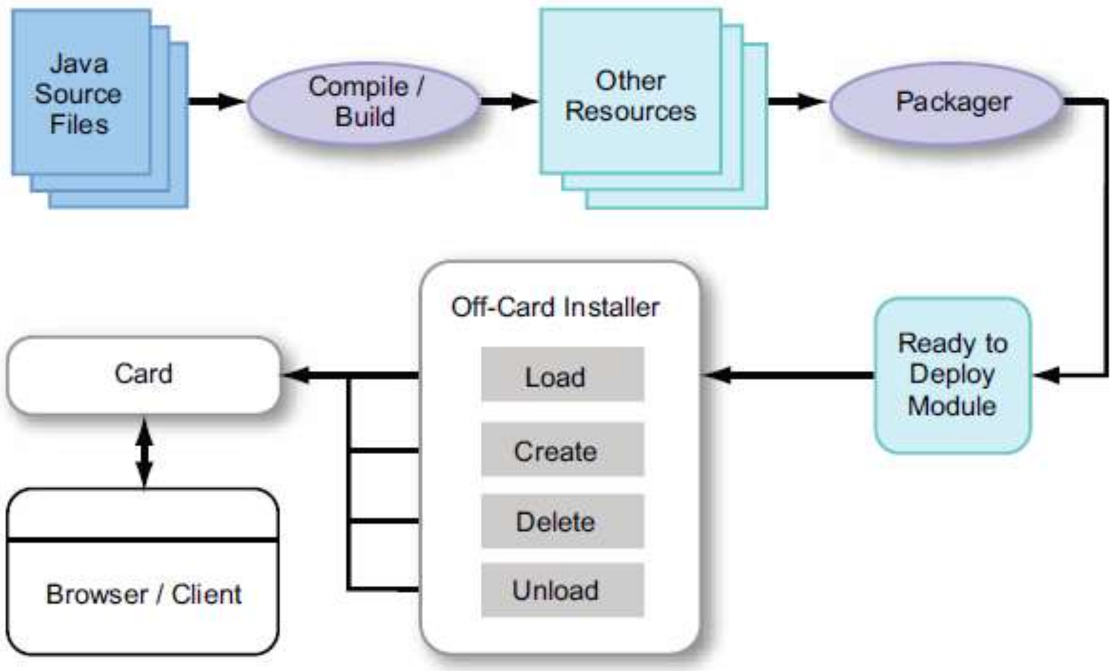
Applets communicate by sharing objects, again linking the access control directly to the (shared) object. Not all objects can be shared by all applets, that is, the Java heap is partitioned in to a number of virtual heaps. This form of access control has become known as the Java Card firewall, which is actually a bit of a misnomer. A more appropriate description would be an ‘execution context’, which is something that main stream operating systems have offered for at least 50 years.

The application of this idea into the smart card world originates from Banerjee’s work on virtual cards in OSCAR, Ugon’s work on SPOM



As part of applet loading, Java Card run-time environment performs byte code verification. This is conventionally done in two stages, where the verifier runs off-card, and once the applet has been verified it is digitally signed and installed on the card. Lightweight byte code verification for Java Card management performs on-card verification. In both cases the implicit assumption is that once verified, the applet cannot be changed.

This assumption may be too strong, especially if native code can be downloaded and executed in the card. Recently an attack on the Java runtime has been shown by inducing memory errors, while, as published only applicable to applications that can control a large amount of RAM, which is not available on smart cards it demonstrates that additional defensive mechanism may be needed, for instance computing and verifying checksum over code before executing. Typically a smart card memory is well protected by error correction and tamper detection circuitry, but additional hardware may be needed to support such run-time code verification mechanism. We believe that this is a clear example where the Java Card platform's Java parentage alone is not sufficient for security.



4. Security

4.1. MULTOS

As more and more trust is placed in the ability of smart cards to prove who we are, whether for making payments in a shop, clearing immigration or signing a contract over the internet, then the more attractive it is for fraudsters, organized crime gangs, people traffickers and terrorists to replicate a genuine identity to gain access to funds, benefits, locations or identities that they are not entitled to.

That means that if we are to vest our trust in the ability of smart cards to prove who we are, either face to face, or over remote channels such as the internet, then it is essential that the personal information and unique data, such as biometrics or unique encryption and signing keys that prove who we are, remain protected.

MULTOS has achieved a reputation for being the most secure smart card operating system in the world. But what does that actually mean? Aren't all smart cards "secure" these days? Don't they all protect our identity?

The answer is that security is a relative thing. EMV cards are more secure than magnetic stripe cards, because it takes more know-how and it costs more to copy a chip card than a swipe card.

But when a security measure like chip cards becomes ubiquitous, then the weakest link becomes the chip or the software in the chip and its ability to protect your personal data from a hacker. Techniques exist for penetrating the physical silicon chip hardware of some chips – with electron microscopes and lasers – and with enough knowledge to deduce the values of encryption keys that protect the data stored in the chip. The latest chips from the silicon manufacturers will incorporate defenses against the latest known forms of attack developed by evaluation laboratories. Hopefully the laboratories and the silicon manufacturers always stay ahead of the hackers. In this respect, yes, the latest microprocessor smart card chips represent the most secure form of storage for digital information or executable code.

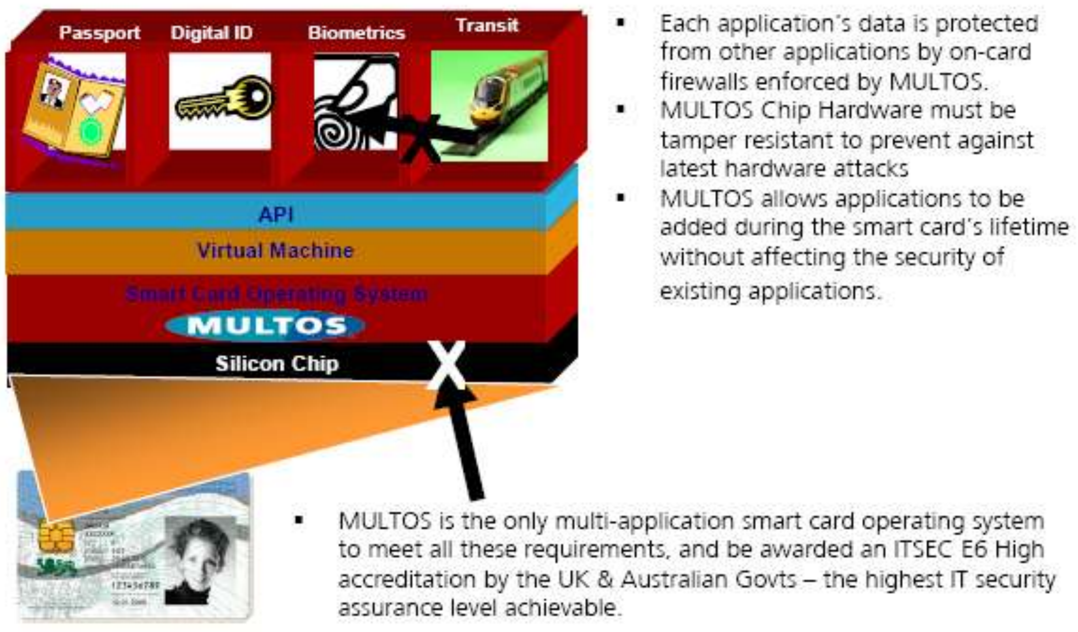
Similarly, the software of the smart card, which contains the operating system and the actual applications that run on the chip – such as your Passport, Payment or Healthcare applications – also needs to contain measures to protect the personal data and unique keys that identify you.

After all, a smart card is like a mini computer. It has the memory and processing capacity of a home computer of just less than two decades ago. So called "open standard" multi-application smart card platforms like MULTOS and JavaCard allow applications to be installed and executed on the smart card, even after the chip card has been "issued" to the recipient. So like PCs, there needs to be a mechanism in place to protect applications on the card that contain sensitive data from Trojans or denial of service attacks. It should be the job of the smart card operating system to make sure that all the applications are protected from unauthorized external and internal attempts to read or change sensitive data.

This is where MULTOS earns its reputation for being the most secure smart card platform in the world. When a smart card OS developer writes a MULTOS operating system on a silicon chip of their choice, they have to ensure that the chip and the operating system comply with the strictest of security assurance targets. Specifically, the MULTOS security target demands that:

1. Applications can only to be loaded onto a card or removed from the card with the permission of the Card Issuer – this means that the card issuer can ensure that applications can only be loaded by trusted third parties. This objective is met by means of cryptographic load certificates that the card issuer uses to authorize the loading of an application.
2. Applications are to be segregated from other applications - an application may not read from or write to another application's code or data. This means that once the card is issued with for instance an Identity application, the Government or Corporation that issued the card does not need to worry that a subsequently installed application masquerading as one from a trusted third party could gain access to the memory areas of the identity application. one application cannot access the memory of another application through the use of “on-card” firewalls which police the execution of each application during runtime. If one application illegally tries to access the memory space of another application, then the MULTOS OS will immediately detect this violation and end the execution of the offending application. This differs from JavaCard, whereby the “fire walling” is achieved when the byte codes of each applet are “verified” off-card to check that there is no unauthorized object sharing between applets. This has the consequence that if one wishes to load a new applet post issuance, then the newcombination of applets needs to be “verified” by someone. This means the new combination undergoing a new security evaluation (an expensive and time consuming process). The MULTOS on-card firewalls on the other hand mean that you CAN add a new application to MULTOS, without needing to re-evaluate the whole combination.
3. Loading or Removing an application must have no effect on the code and data of existing applications – when a new application is loaded, it is allocated a fixed area of memory in which it can store its static and dynamic data and executable code. A MULTOS application cannot grab more memory after it has been loaded. If, as with JavaCard, it could, then there would be a risk of a denial of service attack on the other applications on the card, whose own memory requirements could be restricted by the denial of service applet.
4. The application load process must be able to guarantee the authenticity, integrity and confidentiality of the application code and data. This means that an issuer can send new applications, such as a healthcare application, to the identity card, after it has been issued, and the new application and the data in it are completely private to the application provider (the health authority) and are encrypted in transport to the card. MULTOS uses an “asymmetric” key mechanism to allow third parties to

encrypt their application with the target card’s public key and download it on the card without needing to send it over a secure network or via a secure session. The MULTOS card then decrypts the application code and data with its unique private key. This is a patented feature of the MULTOS OS, so it is the only smart card operating system that offers the ability to protect code and data in this way.



- Each application’s data is protected from other applications by on-card firewalls enforced by MULTOS.
- MULTOS Chip Hardware must be tamper resistant to prevent against latest hardware attacks
- MULTOS allows applications to be added during the smart card’s lifetime without affecting the security of existing applications.
- MULTOS is the only multi-application smart card operating system to meet all these requirements, and be awarded an ITSEC E6 High accreditation by the UK & Australian Govts – the highest IT security assurance level achievable.

4.2. JAVACARD

The Java Card technology enables smart cards and other devices with limited memory to run Java-based applications. Java Card technology brings a whole set of advantages to smart cards by offering a secure execution environment, platform-independence, the ability to store and update multiple applications, and compatibility with existing smart-card standards.

It is also important to note that the Java Card technology [JavaCard] was developed specifically to enhance the security of smart cards.

The Java Card platform provides a number of security features that can be enforced at every level at the inception of application development. They are characterized as follows:

- The Java Card technology supports a subset of the Java programming language and JVM specifications, inheriting the security features built into the supported subset.
- The Java Card platform stores the objects and data in memory. During a power loss or unexpected failure, the platform makes sure that the objects and data are stored to its previous state before such failures.
- The Java Card applets are verified to ensure their integrity, because they can be downloaded over an unsecured network. A trusted third party can also cryptographically sign Java Card applets. The digital signature of an applet can be verified during on-card installation to further ensure the safety of the code.
- In the Java Card runtime, the notion of a sandbox is implemented via the applet firewall mechanism. The firewall essentially assigns an object space, called a context, to each applet on the card. Data access within a context is allowed, but the access to an applet in a different context is prohibited by the firewall. Applets residing in different contexts can share objects using secure object-sharing mechanisms by implementing a shareable interface.
- The Java Card applets are not allowed to execute native methods except for the card's vendor-issued applets. This means that applets installed after issuance of the card are restricted from running native methods.
- The Java Card technology embraces techniques using compressed archive files with cryptographic signatures to provide tamperproof distribution and installation procedures for Java class files and Java Card applets

Java applets are subject to Java security restrictions; however, the security model of Java Card systems differs from standard Java in many ways.

The Security Manager class is not supported on Java Card. Language security policies are implemented by the virtual machine.

Java applets create objects that store and manipulate data. An object is owned by the applet that creates it. Even though an applet may have the reference to an object, it cannot invoke the object's methods, unless it owns the object or the object is explicitly shared. An applet can share any of its objects with a particular applet or with all applets. An applet is an independent entity within a Java Card. Its selection, execution, and functionality are not affected by other applets residing on the same card.

How things work together inside a Java Card

Inside a Java Card, JCRE (Java Card Runtime Environment) refers to the Java Card virtual machine and the classes in the Java Card Framework. Each applet within a Java Card is associated with unique AID assigned by JCRE.

After an applet is correctly loaded into the card's persistent memory and linked with the Java Card Framework and other libraries on the card, JCRE calls the applet's install method as the last step in the applet installation process. A public static method, install, must be implemented by an applet class to create an instance of the applet and register it with JCRE.

Because memory is limited, it's good programming practice, at this point, to create and initialize the objects the applet will need during its lifetime

An applet on the card remains inactive until it is explicitly selected. The terminal sends a "SELECT APDU" command to JCRE. JCRE suspends the currently selected applet and invokes the applet's deselect method to perform any necessary cleanup. JCRE then marks the applet whose AID is specified in the "SELECT APDU" command as the currently selected applet and calls the newly selected applet's select method.

The select method prepares the applet to accept APDU commands. JCRE dispatches the subsequent APDU commands to the currently selected applet until it receives the next "SELECT APDU" command.

Java Card Applet Security

The Java Card provides a multi-application smart card environment that allows multiple applets to coexist on a Java Card and also provides the flexibility to download applets after manufacture or issuance. The Java Card provides a virtual applet firewall protection mechanism that isolates an applet package to its designated firewall partition (referred to as context).

The context mechanism disallows object access from another applet located in a different context. To support cooperative applications running on a single card, it provides a secure object sharing mechanism. The sharing mechanism under specific conditions enables one context to access objects belonging to another context by performing a context switch. When an object is accessed, the JCRE enforces access control, and if the contexts do not match, the access is denied and results in a *SecurityException*