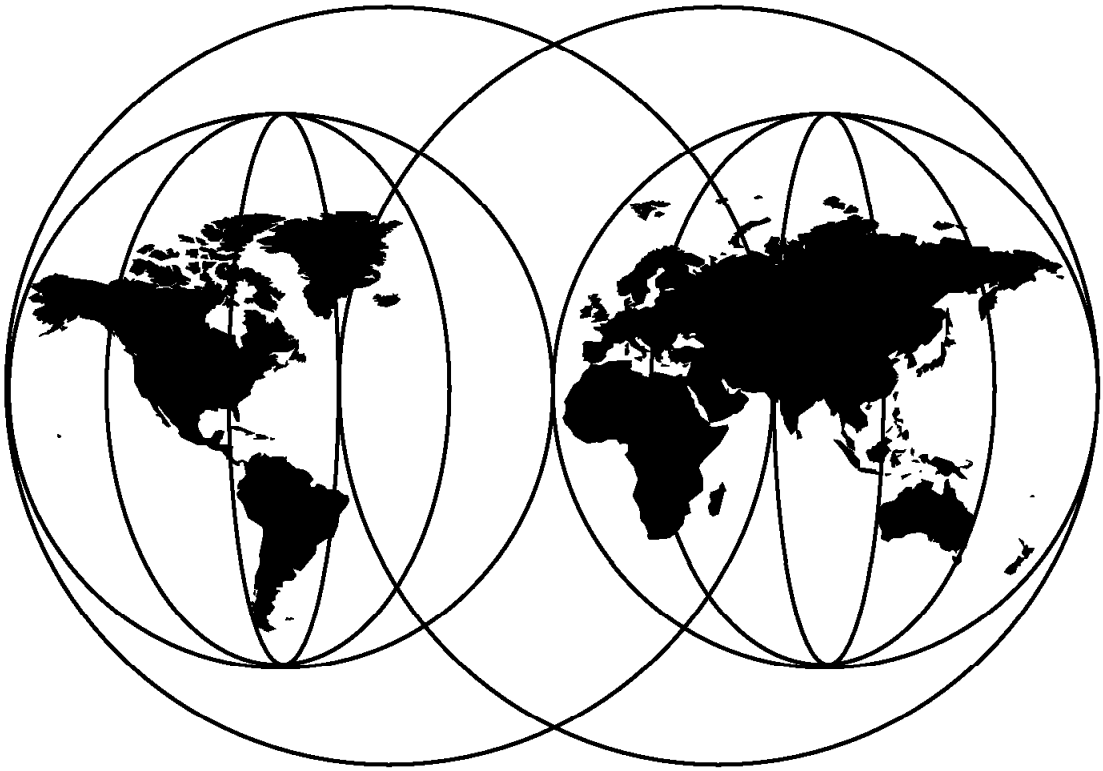
Smart Cards: A Case Study

Jorge Ferrari Robert Mackinnon
Susan Poh Lakshman Yatawara



International Technical Support Organization

http://www.redbooks.ibm.com

This book was printed at 240 dpi (dots per inch). The final production redbook with the RED cover will be printed at 1200 dpi and will provide superior graphics resolution. Please see "How to Get ITSO Redbooks" at the back of this book for ordering instructions.

SG24-5239-00

International Technical Support Organization

SG24-5239-00

Smart Cards: A Case Study

October 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix G, “Special Notices” on page 201.

First Edition (October 1998)

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

This redbook describes the components and the usage of smart cards and will help you to design, plan and implement projects that involve smart cards. These cards have the size and shape of credit cards, but contain a microprocessor and memory. They interchange data with the outside world through contacts or radio frequency. These cards provide a great deal of security for the data stored inside them.

The security and mobility inherent in these cards have made them grow in popularity in the recent past for a number of applications, mainly in the financial area.

In this book we describe the basic characteristics of a smart card, including standards, security features, programming and management of the cards, as well as the card readers. We go through the process of implementing an employee card solution in an imaginary company.

Anyone who provides consulting and service activities or implements smart card solutions will appreciate this book. We do not expect the reader to have previous knowledge of smart cards.

About This Book

There are many excellent books on smart cards available. The reason for writing this book is that we wanted to present a smart card case study, describing the customer′s requirements and the solution proposed. The books on the market in general do not address the solution of a specific case.

We specifically chose a rather simple case: a company that wants to implement an employee smart card. This simple case is far less complicated than the applications mentioned in 1.10, “Some Examples of Smart Card Applications and Projects” on page 7. In those examples, the technical problems are relatively small compared to the logistics of getting legal agreements between different enterprises that participate in the project, fees, ownerships, etc.

The beauty of our case is that we do not have to deal with that logistic nightmare. Our ″company″ has control of almost everything: users (the employees), applications, network, buildings where the readers will be installed, etc.

We present some simple and not-so-simple requirements that the company has sent to us. We describe how to satisfy some of these requirements and also explain why some of them cannot be implemented.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the Systems Management and Networking ITSO Raleigh Center.

Jorge Ferrari is a Network Security Specialist in the Systems Management and Networking ITSO Raleigh Center. Before working in the security area, he was a specialist in network design and capacity planning. He holds a degree in

Electronic Engineering from the Universidad de Buenos Aires, Buenos Aires, Argentina.

Robert MacKinnon is a Systems Engineer for the eBusiness Solutions group in IBM Norway. He has two years of experience in the area of online payment systems such as Secured Electronic Transactions (SET) and IBM Micropayments. He has worked at IBM for 19 years and, in that time, he has accumulated expertise in VM/370, UNIX systems administration, Internet security and firewalls, Internet payment systems and now smart cards.

Susan Poh is a Program Manager in Global Smart Card Solution in United States. She has 20 years of experience in system integration field. She has been involved with many smart card projects such as America Airline Advantage Card, American Express Corporate Smart Card, MasterCard Travel Card, and IC One KidsCard in North America. Her areas of expertise include consulting, solution design, project management and implementation.

Lakshman Yatawara is a Smart Card Solutions Specialist in IBM United Kingdom. Over the last four years he has gained extensive experience of the practical implementations of smart card solutions for clients around the world. He has worked at IBM for 15 years. His areas of expertise include PC application programming and the design, usability and fast prototyping of smart card applications.

We thank Siegfried Langer, IBM Germany for his advice during this project.

We would like to thank all the people from the IBM Boeblingen Lab for their invaluable contributions to this project, and especially to:

Udo Bussmann
Rainer Dammers
Berthold Gerber
Klaus Peter Gungl
Horst Henn
Andreas Nagel
Martin Scott Nicklous
Klaus Rindtorff
Albert Schaal
Achim Schneider
Frank Seliger

Thanks also to the following people from the IBM Smart Card Solution Center in the UK:

John Osmond
Alasdair Turner
Peter Holdsworth

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 213 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com/`
  For IBM Intranet users `http://w3.itso.ibm.com/`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Chapter 1. Introduction

The smart card that we describe in this book is a plastic card equal in size and shape to a credit card, that contains an integrated microprocessor and memory.[1] These two components allow the storing and processing of information in the card.

The smart card interchanges data with the outside world in two ways:

1. Through gold plated contacts. They are called contact cards.
2. Through radio frequency, using an antenna embedded in the card. This type of card is called contactless.

In order to interchange information, the card has to either be inserted in a reader (in the case of contact cards) or placed in the proximity of a contactless reader.

There are no batteries in a smart card; the power is provided from outside by the reader, be it contact or contactless. The CPU clocking also comes from the reader.

The memory is Electrical Erasable Programmable Read Only Memory (EEPROM), which will retain the contents even if there is no voltage applied.

Some cards have only memory; an application can increase or decrease counters in the card. These cards are referred to as memory cards.

Only those cards with chips that contain certain microprocessor logic are called smart cards. We will be referring to them throughout this book.

The two main characteristics of a smart card are its security and mobility. Since its mobility characteristic is obvious, we will concentrate on its security features. Almost all the applications that use a smart card are based on the fact that it is very difficult to forge the card or to access protected data on the card. If for some reason or other the security of the smart card could be compromised, it would be almost impossible to justify its use.

Smart cards are used in a wide variety of applications:

- Automatic fare collection in buses, trains and airline travel
- Financial transactions
- Electronic purse
- Biometric identification
- Access control
- Phones
- Warehouse and inventory control
- Loyalty programs

Government, financial services, transportation, telecommunications, healthcare, education, retail, and many other industries are planning to or already using

[1] There are many smart cards that do not have the size and shape of a credit card; for example, the mobile phones in Europe and Asia (GSM) have a smart card inserted in them. They are signicantly smaller in size than a credit card.

smart cards as a means of providing better security and improved services to its customers and users.

1.1 Smart Card History

Smart cards can be traced back to 1968 when using plastic cards as the carrier of microchips was first developed by the German inventors Jürgen Dethloff and Helmut Grötrupp. Two years later, in 1970, Kunitaka Arimura developed a similar application. The first formal reality of a smart card came with Roland Moreno′s smart card patents in France in 1974. With his patents, the semiconductor industries were able to manufacture and supply the required integrated circuits at a reasonable price.

The first field trial was successfully carried out by the French PTT (Postal and Telecommunications Services) with telephone cards in 1984. Germany conducted telephone card trials three years later.

Using smart cards in the financial industry as bank cards progressed much slower due to the complexity and the existing infrastructure of the banking systems. Not until the developments in recent years of modern cryptography technology, which enabled smart cards to have a high degree of security, did banking associations begin to take the smart card seriously. Other industries, such as health, education, retail, telecommunications, and transportation all have begun to employ smart cards as part of a total solution. As we move into the 21th century, the smart card will have a more pervasive role in electronic business.

1.2 Smart Card Operating System

The core of a smart card is its operating system. This is the code that handles the file systems, the security, the I/O, the handling of the different applications, etc. It is similar to the operating systems of PCs, except that it is limited to a few thousand bytes.

There are several companies that develop and market operating systems; IBM has been a pioneer in the area, since 1984. In 1990 IBM introduced the MultiFunction Card (MFC) operating system. Since then, many new versions of the MFC operating system have been developed. A unique version of the MFC was done for Zentraler Kreditausschuess (ZKA), the central committee of the Germany bank group, for payment systems standards. This development provided the platform for the GeldKarte, the smart card application with the most number of cards in the world (see F.3.1, “GeldKarte” on page 197).

1.2.1 Smart Card Operating Systems on the Market

Besides IBM′s MFC, there are many smart card offerings currently available. Smart card vendors have their own versions of smart card operating systems. The following is a list of operating systems offered by various smart card vendors. This is not a complete list.

- Bull: SmarTB, CC, Odyssey I (JavaCard), etc.
- DeLaRue: DS, DX, DXPLUS, CC, Mondex Card, JavaCard, etc.
- Gemplus: PCOS, MPCOS, GemVersion, GemXpresso(JavaCard), etc.
- Giesecke & Devrient: Starcos S, Starcos PK, Starcos X, etc.
- ODS: ODS-COS, etc.

- ORGA: ICC, etc.
- Schlumberger: ME2000, PayFlex, Multiflex, Cryptoflex, Cyberflex(JavaCard), etc.
- Siemens: Card OS

Sometimes smart card vendors license smart card operating system from other manufacturers, and extend and modify commands and applications for different purposes. For example, both Gemplus and Schlumberger license IBM′s MFC.

1.3 Smart Card Types

Since there is no official definition of the term ″smart card″, many different cards are being called smart cards as long as they have some kind of intelligent circuitry on the cards, such as a microprocessor.

We will cover the different types in detail in Chapter 4, “Card Selection Process” on page 41.

1.4 Smart Card Standards

There are many standards, specifications and recommendations for smart cards. Some of them come from recognized international bodies such as ISO. Some come from industry organizations such as financial institutions; some come from companies that want their products set the norms; some are de facto standards. We can categorize the standards in the following groups based on the standard organizations:

- The International Organization for Standardization (ISO) standards:
  - ISO 7810: plastic ID cards, dimensions
  - ISO 7811 Parts 1-6: ID cards
  - ISO 7816 Parts 1-8: contact integrated circuit (IC) cards
  - ISO 10536 Parts 1-4: close coupling cards
  - ISO 14443 Parts 1-4: remote coupling cards
- The country and/or industry standards. Some of them are not smart card standards, but are used by applications running in smart cards.
  - CCITT X.509: Directory for certificates
  - EN726 Parts 1-7: for telecommunications IC cards and terminals
  - ANSI (US Standard body) X9 series: for digital signature, secure hash, RSA, and data encryption algorithms
  - US Government standards:
    - FIPS-46: Data Encryption Standards
    - FIPS-81: DES Modes of Operation
    - FIPS-180-1: Secure Hash Standards (SHA-1)
    - FIPS-186: Digital Signature Standards (DSS)
  - GSM 11.11-11.12: European digital cellular telecommunications system
  - Europay, MasterCard, and Visa (EMV) Parts 1-3: IC card specification for payment systems
  - International Airline Transportation Association (IATA) Resolution 791: using smart card for electronic ticketing
  - PC/SC: specification for connection of smart card readers to PCs running Windows operating systems
  - G7: International health organization for health card

- OpenCard Framework: architecture specification framework for terminals and cards
- JavaCard: Specification for JAVA virtual machine
- Common Data Security Architecture: architecture for security plug-ins

This is an incomplete list. We cover this subject in Chapter 3, “Standards, Specifications and Recommendations” on page 27.

1.5 Smart Card Security

There are many reasons to use a smart card, but as we said earlier one of the main reasons is the built-in security features of a smart card. The microprocessor of the smart card has encryption keys and encryption algorithms built-in for performing ciphering/deciphering of data inside the card. The operating system file structure prevents the secret keys from being read from outside the smart card.

If multiple applications reside on the same smart card, they are protected from each other by a firewall between them.

In addition to the smart card itself, a smart card solution needs to address total system security, which includes readers, terminals, network, and back-end processing systems. In other words, to ensure overall security it is necessary to do a total system design, which covers all the subjects that can be attacked by criminals.

During the design process we must evaluate the risks involved versus the rewards of establishing strong security features, the expenses of implementing the solution against the exposure of suffering a security breach. This design process is probably best performed by a consultant who has no personal interest in a given product, than by vendors who may have high skills regarding a product but lack the overall system security knowledge.

We cover this subject in Chapter 2, “Smart Card Security” on page 11.

Because the smart cards can perform strong encryption, they are affected by export/import regulations. We cover this subject in Chapter 14, “Export and Import” on page 163

1.6 Smart Card Terminals/Readers

There are many different kinds of smart card readers, also known as card acceptance devices or IFDs (interface device). The term ″reader″ can be misleading since these devices can also write to the cards.

Some kind of intelligence is usually present in the card terminal, since the reader and the card must authenticate each other.

There are many ways of classifying the smart card readers:

- Type of card: contact or contactless
- Functionality: from simple to complex in functions
- Stand alone vs online
- Final product or an OEM to be integrated in another machine, such as ATM, kiosks, vending machines, etc.

- PC attachment: PCMCIA, RS232, etc.
- Price: below US$ 100 for some RS232 readers to several thousand dollars for cash loading machines

We cover the card terminals in Chapter 5, “Card Reader Selection Process” on page 55.

1.7 Smart Card Development Toolkits

Coding applications for a smart card requires a very rare set of skills. The programmer must know not only the programming language and the platform where the code is developed, but also the smart card operating system, the smart card reader, the communication protocol between the smart card and the reader, the smart card file structure, etc. This situation has been recognized by the industry and most of the smart card vendors provide development tools for their customers.

IBM′s Smart Card ToolKit thus far is one of the most complete and powerful tools for smart card application development. Using IBM′s ToolKit, an application developer does not need to know the smart card operating system internals and details. We cover this subject in Chapter 10, “Smart Card Programming” on page 119.

1.8 Smart Card Solution

An end-to-end smart card solution not only consists of many hardware and software components, but also requires network connectivity, consulting, application development, and system integration services from a technical perspective. On the business side, you also need a feasibility analysis and business case to justify re-engineering using a smart card.

As shown in Figure 1 on page 6, the hardware components are cards, terminals with or without workstations, mainframe host computers, servers, etc. The connectivity may be direct, over the phone line, on private network, or public network. Of course, some smart card terminals or Card Acceptance Devices (CAD) such as vending machines and copy machines may be just offline without online connectivity.

At the back end of a solution system there are hosts and servers for databases and transaction processes.

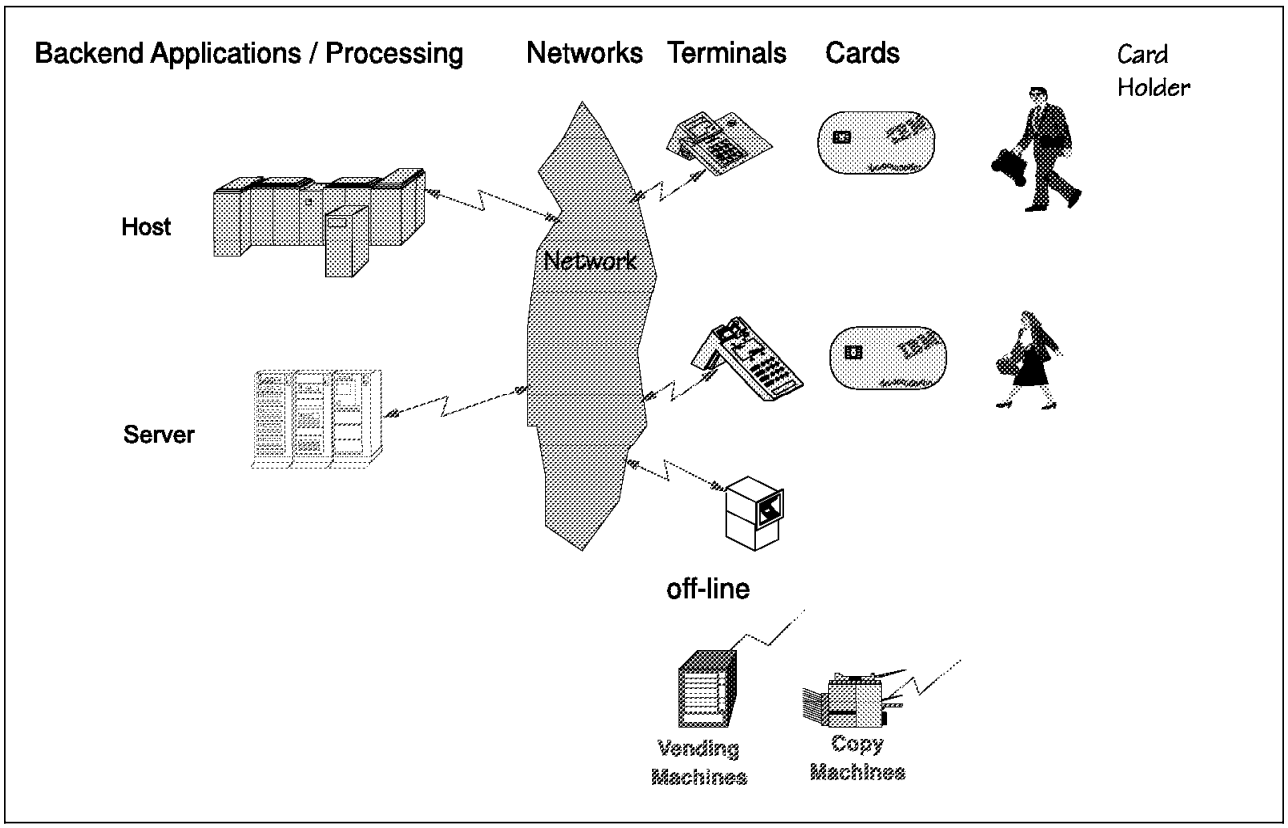


Figure 1. Smart Card Solution Components

We address this issue by presenting a case study and its solution starting in Chapter 6, “Case Study” on page 69.

1.9 Smart Card Card Management

The smart card has a life cycle, from its manufacture to its destruction, including the personalization, the distribution to the user, the replacement, etc. The managing of the card life cycle is called Card Management System (CMS).

CMS must address the following stages and participants in the life of a smart card:

- Chip manufacture
- Module manufacture
- Plastic card manufacture
- Card issuance service bureau
- Software and system integration company for providing smart card initialization and personalization software
- Card issuer
- Card application provider
- Card renewal/modification/revocation

Some companies can provide multiple functions in the process of smart card production and issuance. But in practice, companies usually team up in alliances because each technology (such as chip, mask, modules, plastic, and personalization) requires special skills and tools. Investment in each area can be very extensive and expensive.

To issue a smart card to a cardholder, there are usually two steps: initialization and personalization. During the initialization step, the file structure and some keys are loaded. During the personalization step, individual data unique to the cardholder, such as name, address, account, etc., plus some more encryption keys are loaded into the chip. Part of this information is also printed on the plastic and encoded into the magnetic stripe, if the card has one.

We cover CMS in Chapter 9, “Smart Card Production” on page 111 and Chapter 12, “Card Management” on page 145.

1.10 Some Examples of Smart Card Applications and Projects

There are many successful smart card applications and projects around the world. Based on the number of cards in use nowadays, there are more smart cards issued in Europe than any other continent at the present time. Pre-paid phone cards (memory cards) still are the vast majority of cards on the market.

Some examples of applications and projects are listed here to provide an idea of where the smart cards are used:

- Banking:
  - Geldkarte -- Germany
  - Visa Cash -- several pilots (New York city, 1996 Atlanta Olympics)
  - Mondex -- Pilot in many countries
  - Proton -- Belgium and 11 other countries, US American Express pilot
- Campus Card:
  - Dutch University Student Card -- Holland
  - Florida State University Student Card -- USA
  - Michigan University Student Card -- USA
  - IBM Germany Employee Card -- Germany
- Travel:
  - American Express Corporation Card pilot (Hilton, American Airline, Continental Airline) -- USA
  - Lufthansa, Germany
- Health:
  - Sesam Vitale -- France
  - Diabcard
  - RAMQ
  - Women and Infants-- USA, Western Governors Association
- Government:
  - Marc card -- USA
  - Citizen Card -- Finland
- Telecommunication:
  - Chipper -- Holland
  - Telecom, Malaysia
  - Telstra, Australia
- Retail (loyalty)
  - IC One (USA)
  - Shell (UK)

– Boots (UK)

Most of these projects are multi-application and multi-industry. For example, Chipper is used for telephone, home banking, payment, etc.

1.11 Future of the Smart Cards

The industry analysts and the specialized press forecast a brilliant future for the smart cards. Billions of cards are predicted to be circulating in a few years. We stay away from these forecasts in this book, but agree that the use of smart cards will grow significantly.

But the industry analysts and the specialized media also agree that the smart cards have yet to penetrate one of the most interesting markets in the world: the USA.

There are several reasons for this:

- No large centralized government applications (such as health care) which require the citizens to use smart cards
- No smart card infrastructure: many stores in Europe have some kind of card reader
- USA customers and merchants are more ready to accept the cost of using credit cards than its European counterparts
- Cellular phones in USA do not follow the GSM standard (Groupe Spéciale Mobile, see 3.3, “GSM” on page 30), which uses a smart card inside the phone. In Europe the phones use the GSM standard. These phones use smart cards with a special operating system and its size is even smaller than credit card size.

Another reason given for this situation is that in USA communications, over either direct or switched lines, are less expensive than in the rest of the world. What does this have to do with smart card usage? Many cards applications require an online transaction, for example a credit card may require to check the validity of the card. The beauty of the smart card is that allows the merchant to avoid this online checking, allowing an offline authentication and thus saving a phone call; France has credit and debit cards that use smart cards for this purpose.

An example of an offline application is the electronic purse. The user loads the smart card purse with money and goes shopping with it. When the user hands the smart card to the merchant, there is no need to require the approval from a credit organization, as the ″money″ is already in the card.

The merchant avoids the cost of a phone call and also gets the money right away.

Why are communications less expensive in the USA than in most of other countries? The usual answer is that outside the USA communications are a monopoly of the government-owned PTT. But this situation is changing; many PTTs are going private.

It is necessary the discovery of an ′killer′ application in the USA in order to increase the penetration of smart cards in this market. Of course, we do not know which this ′killer′ application(s) may be. As we said before, the most

important features of a smart card, from our point of view, are security and mobility. This ′killer′ application should exploit these features.

There is a situation in the IT business that may help the usage of smart cards: the fear of hackers attacking sensitive information from the Internet. Another is to protect assets, denying access to premises that contain valuable information.

In this book we emphasize these types of applications: digital signature, biometrics, access control, etc. These application will not generate the zillions of cards predicted in some forecasts, but they will require the help of consultants and integrators, to whom this book is addressed.

In summary, for the future we think that:

- In North America the use of smart cards for authentication will probably be one of the main drivers in the increase of usage of cards
- In the emerging countries where telecommunications still has to catch up with the USA′s standards and prices, the use of smart cards for offline transactions will be the main factor in increasing smart card usage.
- In Europe the goverment usage of smart cards for health and other federal prgrams will be a decisive factor in expanding the smart card market.

Chapter 2. Smart Card Security

Smart cards are used primarily in applications requiring high security, such as facilities access or in applications handling sensitive information such as financial applications. Thus, a criminal could benefit financially by trying to break the security controls that are designed into a smart card. In this chapter, we will look at the security controls that are in place in a smart card, both in a human-readable and machine-readable form, to prevent such attacks. We conclude by discussing security principles that are enforced by the smart card,/seeluding cryptographic techniques that are used for privacy and authenticity.

2.1 Magnetic Stripe Card Security

Before we describe the security features a smart cards, we will briefly talk about magnetic stripe cards. It will give us a better perspective of what we gain in terms of security when we move from magnetic stripe cards to smart cards.

The data in the magnetic stripe is usually coded using two or three tracks. The standard covering this area is ISO 7811. The technique for writing to the tracks is known as F/2F.

We can read from time to time in the news that some criminal has counterfeited some of these cards. The reason is that it is not that difficult and/or expensive to have the equipment to encode magnetic stripes. For this reason you can make as many copies as you want from a mag stripe.

As these cards are so widely used by financial institutions, a new way of encoding the stripe was developed; it uses a magnetic material called high-coercivity (HiCo), as opposed to the low-coercivity (LoCo) material used before. The HiCo material requires stronger magnetic fields to encode in it. Any card reader can read any one of these materials, since the encoding technique (F/2F) is the same.

The security resides in the fact that not many encoding machines in the market can handle the HiCo material, and are definitely more expensive than those to encode LoCo material. The manufacturers of these encoders will certainly want to know why someone may be interested in purchasing such a device.

Another good reason for using the HiCo material is that it is better suited to avoid local disturbances on the stripe due to magnetic fields and heat.

There are many visual security features to prevent reproduction of these cards. We cover them in 2.2.1, “Human Readable Security Features of Smart Cards” on page 12, as they are also used for smart cards.

Even if it very difficult to counterfeit a magnetic stripe card, it is far from impossible. Let′s look at now the security features of smart cards.

2.2 Smart Card Security Features

Some components that play a role in smart card security:

- Human-readable security features
- Security features of the smart card chip
- Security features of the operating system
- Security features of the network
- Security features of the application

The fifth component will not be discussed in this book because the features selected are very dependent on the application program itself. The total system security is strengthened when all these components work in combination with each other. There should not be a ″weak link″ in the chain of security.

2.2.1 Human Readable Security Features of Smart Cards

There is often a need to include human-readable security identifiers on smart cards; these features try to prevent smart card falsification. These features, of course, do not protect the data in the card. They prevent the misuse of the card as a badge identificator. You may find some of these features in your credit card.

These features are more important for the magnetic stripe cards than for the smart cards, since it is easier to alter the content of a magnetic stripe than the memory of a smart card. However, since many smart cards also have a magnetic stripe in the back, these features are also valuable for these cards.

Here are a few of these features:

Photo lamination The smart card is personalized when issued to the cardholder with a passport-sized photograph of the cardholder. The photo would be laminated to the card. The security is embodied in the procedure followed before the photo is laminated. The cardholder must present the photo in person to a certified representative so that the cardholder′s identity can be confirmed before the smart card is issued.

Signature strip This is a very familiar feature on credit cards. A signature strip is bonded onto the card at manufacture. The cardholder must sign the strip using indelible ink when the smart card is issued. The signature cannot be modified nor the strip replaced without being obvious to the naked eye. Some credit card companies have electronic images of their customers′ signatures on record and these signature images are indelibly printed onto the card when a new card is issued.

Holograms This is another common feature found on credit cards. The holograms are bonded to the card at manufacture and cannot be separated from the card without destroying the substrate. The security feature of the hologram is based on:

- The difficulty in reproducing the hologram
- The limited number of firms who can manufacture the hologram itself

Holograms are expensive.

Microprinting This feature is ultra-fine printing that appears as a line to the naked eye but is visible under magnification. The printing itself is difficult to reproduce.

Embossing This is another familiar feature of credit cards. The card number is pressed into the card, sometimes over the hologram for additional security, so that the numbers are raised above the surface of the card. An impression of the number can be transferred to paper using a machine.

Unfortunately, as fate would have it, the position of the embossing marks called ″domain 1″ in the ISO 7811 specification can interfere with the smart card chip position defined in ISO 7816 specification (see 3.1, “ISO 7816” on page 27 for more information about ISO 7816). The domain 1 position is very familiar to credit card users because this is the location where the credit card number is normally embossed. There is no overlap between the domain 1 position and the smart card chip position but the stamping process can sometimes fracture electrical connections to the chip if care is not taken. Qualified smart card manufacturers have reliable embossing processes in place and there is no risk with embossing if such a manufacturer is chosen.

Security patterns This expensive process is otherwise known as ″guilloche″. It is the printing of very fine, interwoven lines onto the card substrate. This is a typical security feature on paper currency. The security feature is similar to holograms; they are difficult to reproduce and only a limited number of companies can perform this procedure. Printing of credit card company logos is a variation on this security mechanism.

An additional requirement for credit card company logos is that the smart card laminating and printing process can only be performed by a limited number of registered and licensed firms. These firms must conform to very strict rules regarding the manufacture and shipping of the card substrates and the processes are certified by the credit card company regularly.

Lasergravure Using a laser, it is possible to burn images into the card substrate. The burning is indelible and very personal because it can be done when the smart card is issued to the cardholder. Examples of items that can be lasergravured are cardholder name, cardholder photograph and card number. This technology is limited to black and white text or images.

2.2.2 Security Features of the Smart Card Chip

It is necessary during production for the smart card chip to test the microcircuit. After the chip has been tested, the chip must be irreversibly converted to a mode where it is impossible to access the internal chip circuit, for example directly accessing memory from the outside. One of the last processes in the chip fabrication is to apply an electric current to the selected chip to blow a fusible link on the chip. Sometimes, in concert with blowing this fusible link, some manufacturers modify a location in EEPROM that logically changes the chip. The card operating system detects the blown link and/or reads this memory location to determine the current mode. Once changed at the factory, it is impossible to change the operating mode back to ″service mode″.

Other on-chip security features include burying componentry deep inside the chip in inaccessible locations. For example, the ROM is buried in the lower layers of silicon to prevent reverse engineering of the card operating system. The internal address and data buses that connect the major components on the chip together are scrambled. That is to say, the individual conductors are interchanged to make it more difficult for the attacker to deduce their function. Communication between on-chip components are encrypted on some smart card chips.

To prevent electrical signals emitted by the memory cells from being monitored externally, the area of the chip around the EEPROM is coated in a metal shield. Removing this shield will destroy the chip and it will no longer function. The chip is also coated with a passivation layer to stop ultraviolet light from erasing the memory on the chip.

Smart cards have circuits to detect external tampering with the chip. There are circuits to detect too high or too low supply voltage, too high or too low external clock frequency or sometimes too low an operating temperature.

2.2.3 Security Features of the Card Operating System

One of the enormous strengths of smart cards is the card operating system. All memory accesses must flow through the CPU so the design of the card operating system is critical for implementing security at a logical level. The logical organization of the dedicated files in EEPROM memory forms a security barrier. For example, when a dedicated file (DF) is selected, the card operating system prevents access to data in other DFs. See Chapter 8, “Card Layout Design” on page 87 for details.

Access to smart card files can be protected with a Personal Identification Number (PIN) or with cryptographic keys (see 8.1.4, “File System Security” on page 92 for details about the various commands available to a smart card developer for protecting file contents with a cryptographic key). If a PIN is entered at the card terminal incorrectly, then after a number of attempts, the smart card is inactivated. The number of attempts is determined by the smart card issuer. Depending on the design of the smart card, it is possible for the card issuer to reset the smart card if it is deactivated. Different PINs can be used to protect information in different directories.

2.2.4 Security Features of the Network

The system design should take into account the accessibility of data in transit and protect it accordingly or design the transport protocol such that tampering will not affect the overall system security.

One easy access point in the network is at the smart card contact pads. An attacker can insert the smart card into a hostile smart card reader whose purpose is to exercise the smart card and intercept the data stream flowing between the smart card and the smart card reader. Encrypting all sensitive data that will exit the smart card through the contact pads is one way to thwart this attack.

If the card terminal can be physically secured by building it into a wall for example, then equipping the card terminal with a motorized smart card reader with shutter will enhance the network security. The motorized smart card reader will draw the smart card into the machine when the card is inserted and seal the

smart card in the smart card reader while it is in use. People who have used an automated teller machine are familiar with this security method. The card remains inaccessible until the transaction is complete. The card could be retained if the system determines it is being misused.

Modern card terminals are just a part of a larger, more complex network of communications links between computers. These communications links must be physically protected from tampering if data integrity is to be maintained. The smart card reader and any communications links can be physically protected by placing them in a secured environment where personnel or monitoring equipment continuously observe the use of the smart card reader and prevent tampering.

2.3 Security Principles

There are several reasons one requires security in a smart card system. The principles being enforced are:

- Privacy
- Non-repudiation
- Authentication
- Integrity
- Verification

Smart cards use different encryption algorithms to implement these principles. In the following sections, we will describe the mechanisms use in smart cards to enforce these principles.

2.3.1 Privacy

Privacy is the act of ensuring the nondisclosure of information between two parties from casual third-party intrusion. There are two cryptographic techniques used to assure privacy — symmetrical cryptography and asymmetrical cryptography — and each cryptographic technique has different application areas in smart cards.

2.3.2 Symmetrical Cryptography

Symmetrical cryptography uses a single key to encrypt plain text into enciphered text and decrypt enciphered text back into plain text. Symmetrical cryptography is termed symmetrical because the same key is used to encrypt and decrypt the message. Figure 2 on page 16 shows the process. The most popular symmetrical algorithm is Data Encryption Standard (DES) because it is fast, reasonably secure and simple to implement in hardware. DES is a block-encryption algorithm developed by IBM and standardized in 1977 by the American National Standards Institute (ANSI) as Federal Information Processing Standard (FIPS) 46-2, otherwise known as the Data Encryption Algorithm (DEA). The reader is encouraged to go to the ANSI Web site at http://www.ansi.org/ for more information about ANSI and the FIPS standards.

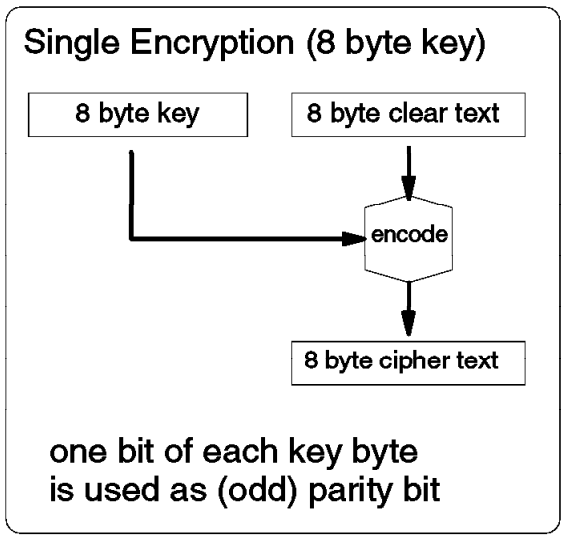


Figure 2. Symmetric Encryption using Single DES

DES can use different key lengths. The longer the key, the more difficult is to break it. A 40 bits key can be broken with a few CPU hours, while a 56 bits key would take considerably longer. These brute force attacks are benefiting from the dramatic increase of the processing power of the CPUs, so this ″considerable longer time″ that we mentioned is a relative term.

The key length makes an encryption weak or strong. There are some legal issues with the use of strong encryption; see Chapter 14, “Export and Import” on page 163 for details.

Data is encrypted in blocks of 8 bytes and results in cipher text of the same size. This is a ″noiseless″ algorithm because the enciphered text size and encryption time is constant and independent of size of the supplied plain text. A noiseless encryption algorithm is more difficult to attack because it is not possible to deduce the encryption mechanics based upon the time differences between encrypting large quantities of plain text input and small quantities.

DES can be implemented on smart card software, as it is a relatively fast algorithm. The time it takes to encrypt one 8-byte block of plain text is in the order of 10 milliseconds. As a comparison, the same text could be encrypted in about 60 nanoseconds with specialized DES hardware. The majority of data on a smart card is stored in EEPROM in clear-text form. However, certain confidential data is stored on the smart card in encrypted form. DES is used to generate that encrypted data.

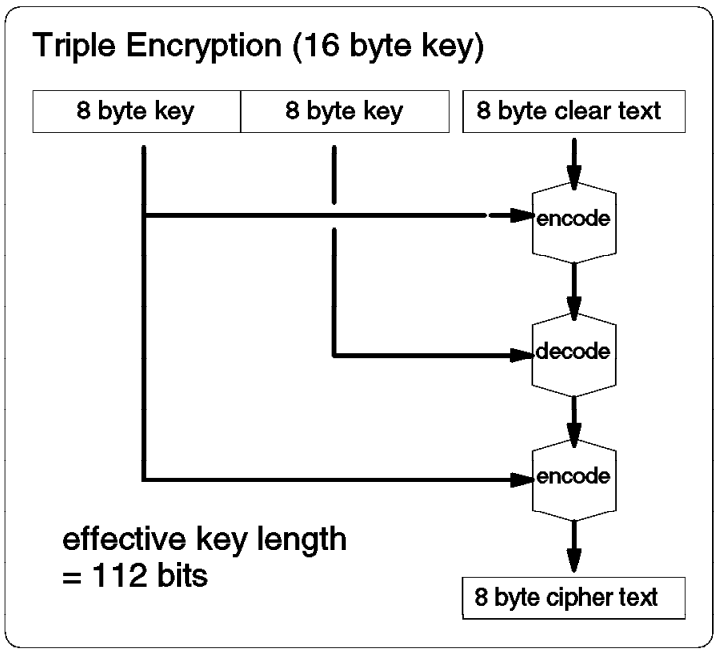


Figure 3. Symmetric Encryption using Triple DES

In cases requiring greater security, for example in the transmission of encryption keys, Triple-DES is used. Figure 3 shows the process. Plain text data in blocks of 8 bytes is enciphered three times using a 16-byte key (which, after accounting for the 16 parity bits, results in an effective key length of 112 bits).

Needless to say, this is strong encryption.

At the present time, nearly all smart cards perform DES encryption in software. All IBM′s MFC cards have DES cryptographic algorithms implemented in software in the card operating system.

The disadvantage of symmetrical encryption is that both partners need to know the key. The transfer of the key from one partner to the other can compromise the security that otherwise the encryption provides. Writing a DES key at card personalization time is the typical method of safely transfering keys to cardholders. If this is not possible, asymmetrical cryptography must be used, which is described below.

2.3.3 Asymmetrical Cryptography

In 1976, the idea of splitting the encryption/decryption key instead of sharing a common key was first proposed in an article by W. Diffie and M.E. Hellman entitled ″New Directions in Cryptography.″ This idea has since become known as asymmetrical cryptography. Asymmetrical cryptography uses two keys: one to encrypt the plain text and another to decrypt the enciphered text. The keys are mathematically related. Only messages encrypted with one key can be decrypted with the other key. The best-known asymmetrical cryptographic algorithm is RSA (named after its inventors Rivest, Shamir, and Adleman). The algorithm is defined in the PKCS-1 specification published by RSA Ltd. (see http://ftp.rsa.com/pub/pkcs/doc/).

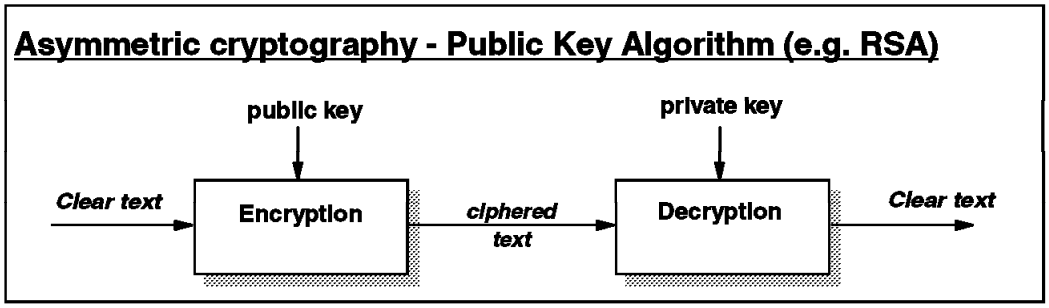


Figure 4. Asymmetric Encryption

Let's give an example of Bob sending an encrypted message to Alice using RSA. Figure 4 shows the encrypting process. The two RSA keys are called the private key and public key. Alice would distribute her RSA public key and keep her RSA private key private. To start the process, Bob would first obtain Alice's public key. Bob would encrypt the plain text using Alice's public key and send the enciphered text to Alice. Alice would use her private key to decrypt the enciphered text. Because of the relationship between the public and private keys, Bob can be assured that only Alice can decrypt the message. It is not necessary for Bob or Alice to share any secret between themselves and thus there is no risk of inadvertently disclosing such a secret to a third person.

Asymmetrical cryptography is used in smart cards but rarely to perform general data encryption. Symmetrical cryptography such as DES is used for that purpose. Instead, asymmetrical cryptography is used in smart cards for authentication purposes such as digital signatures. The private key of the key pair of a digital certificate is normally stored on the smart card for example. This is because it can be safely protected by the card operating system and not disclosed outside the smart card.

As in the case of symmetrical encryption, the keys can have different lengths. The three most common values are 512, 768 and 1024 bits. The last two values are considered strong encryption.

The disadvantage of the RSA algorithm is that it is slow, much slower than the DES algorithm. Due to the limited speed of the smart card CPU, it is not practical to implement the key generation algorithm in software. There are special smart cards that have crypto-processors for that purpose. The MFCs 4.0 and 4.21 are examples of such cards.

The symmetrical and asymmetrical encryptions usually complement each other. The asymmetrical encryption is often used to send the DES key safely from one partner to the other. Once both partners know the DES key, the data is transmitted symmetrically encrypted, which significantly improves the performance.

2.4 Integrity

Electronic communications links are prone to errors and data tampering. Cryptographic techniques are used to ensure that data content does not change while it is being transmitted from the originator to the recipient. This is called data integrity.

2.4.1 Message Authentication Code

A message authentication code (MAC) is an 8-byte value generated for a message that is unique to that message because a one-way cryptographic algorithm is used to generate the value. A one-way cryptographic algorithm is special because it cannot be reversed (that is, the original plain text cannot be recovered from the cipher text) and the enciphered text is guaranteed always unique. The MAC used in smart cards is calculated with DES using a key which is shared by both the smart card and the smart card reader. The MAC is appended to the end of the plain text message before being sent. When the message is received, the recipient calculates a MAC value from the contents of the message and compares the result to the MAC that accompanied the message. Because changing even one character in the message changes the MAC in an unpredictable way, the recipient can be sure that the message was not changed after the MAC was generated. A MAC is a guarantee of integrity, a guarantee that the message has not been tampered. All messages that are exchanged between the smart card and the smart card reader must be protected with a MAC for example.

2.5 Non-Repudiation

Cryptography can also provide authentication of the engaged parties and ensure non-repudiation of the transaction. Non-repudiation is proof of the integrity and origin of data exchanged in the transaction. It is forgery prevention.

2.5.1 Digital Signature

When Bob sent an encrypted message to Alice in the example on page 18, he used Alice′s public key to encrypt the message and Alice would use her private key to decrypt the message. One of the properties of asymmetrical cryptography would allow Alice to check that Bob actually originated the message. This property forms the basis for digital signatures.

A digital signature results from the process of encrypting a message authentication value with the originator′s private RSA key. The main property of a signature is that only one person can generate one (that is to say a digital signature is unique for each person) but anyone can check the digital signature.

In general, it would take too much time to sign a full clear-text message so instead only the message authentication value of the message is signed. Instead of using the MAC algorithm described above to create the message authentication value, the message is passed through a one-way cryptographic process called a hashing algorithm. One popular hashing algorithm used on smart cards is the Secure Hash Algorithm (SHA-1) defined in FIPS PUB 180-1 (refer to the Web page at http://www.ansi.org/ for more information about ANSI and the FIPS standards). Hashing the plain text message with SHA-1 and then encrypting the hash with a person′s RSA private key creates the digital signature. The signature can be verified with the originator′s RSA public key.

Figure 5 on page 20 shows a simplified signature signing and verification process.

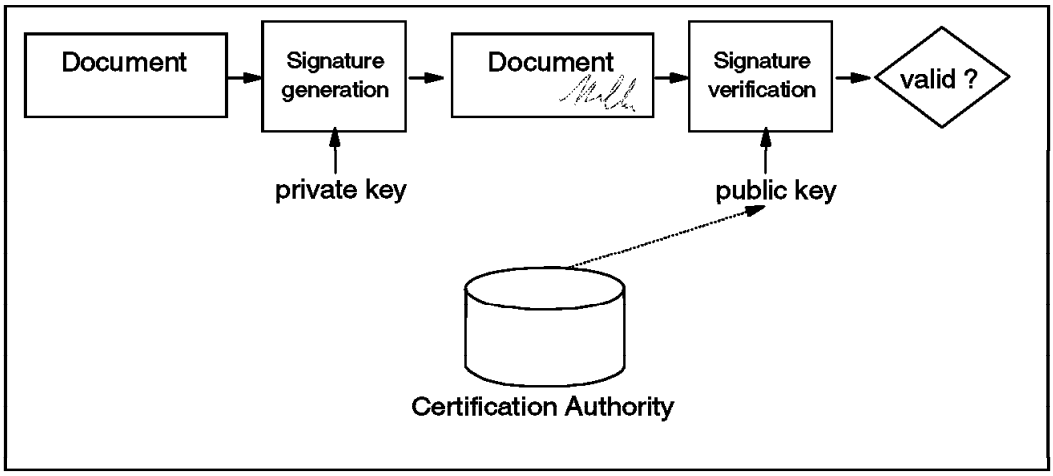


Figure 5. Digital Signature Signing and Verification

Let's take an example. Let's assume Bob wishes to sign a message and send it to Alice. First, he runs the plain text through the SHA-1 hashing process and produces a fixed length hash of the message. Then, Bob encrypts the hash using his private RSA key to create the unique signature for that particular message. Bob then sends the plain text message and the digital signature to Alice. Alice can use Bob's public key to decrypt the digital signature and recover the hash, then generate a hash for the received plain text message and finally compare the two hashes. Again, because of the mathematical relationship between the keys, if the hash decrypts and verifies properly, Alice can be assured that *only* Bob could have created the message in the first place and neither the message nor the hash changed during transmission. Digital signatures also serve as proof of the origin of a message.

RSA asymmetrical encryption is not the only mechanism for creating digital signatures. The Digital Signature Algorithm (DSA) is a specification for a process that creates signatures from message content. The advantage of DSA over RSA is that DSA is designed to use technology that is freely exportable worldwide. DSA signatures are as unambiguous as signatures created with RSA. However, DSA does not do general-purpose encryption and decryption as does RSA. More information about DSA can be obtained from the Web at http://www.nist.gov/.

At the present time, IBM's MFC 4.0 and 4.21 smart cards have an RSA asymmetrical cryptographic co-processor on the chip.

2.6 Authentication

Before two parties conduct business, each wants to be sure that the other party is authenticated.

For example, before Bob accepts a message with Alice's digital signature, he wants to be sure that the public key belongs to Alice and not to someone masquerading as Alice. This is what certificates do.

2.6.1 Certificates

Certificates are guarantees by the authority issuing the certificate that the holder of the certificate is who he/she claims to be. In short, a certificate is just a digitally signed message containing information about the certificate holder plus a copy of the holder′s public key. Anyone receiving the certificate has assurance that the key contained in the certificate is authentic because it is signed by the issuing authority, which is a trusted entity.

On the World Wide Web, Secure Sockets Layer (SSL) uses certificates for example. The Web browser will obtain the certificate of the Web server and use the public key within the certificate to encipher the initial encryption key. Similarly, when the smart card sends data to a card terminal′s Security Access Module, the smart card will construct an encrypted channel, like SSL does, using the card terminal′s public key.

The information fields in a certificate are extendible; however some fields are mandatory. A certificate issuer, known as a certificate authority, can add as many fields as it likes so it becomes a flexible way to associate personal information with a particular individual and maintain the integrity of the information. For example, a certificate can hold the person′s name and address, the person′s employment start date, their employee number, their security access level, and so on.

Before a certificate is issued to a person, depending on the grade of certificate, the issuing authority will require legally binding proof as to the identity of the person. Application for a low grade certificate would only require a passport or driver′s license as proof of identity, for example. A higher grade certificate would possibly need a document certified by an attorney. The higher the grade of certificate issued, the greater the assurance the recipient has that the person truly is who he/she claims to be.

Certificate chains are only based on trust. The receiver must trust that the authority has verified the authenticity of the certificate holder and has not tampered with the information before signing the certificate. If the recipient does not trust the issuing authority, then the certificate will have no value.

2.7 Verification

It is to the benefit of both a smart card owner and a smart card issuer that the identity of the cardholder is confirmed before the card is used. Before both parties transact business, they must be assured of the identity of the other party. When we meet in person, we use visual and verbal clues to help us to recognize the other party. With electronic communication, we use encryption technology to unambiguously verify that the other person is who they may claim to be.

2.7.1 PIN Codes

A Personal Identification Code (PIN) is usually a four or five-digit number that accompanies a smart card and must be memorized by the cardholder. The PIN is stored securely within the smart card in a way that can never be read from the external world. Data and functions on the smart card can be protected in a way that access from the external world is allowed only after the correct PIN code is presented. The IBM MFC can store up to two PIN codes per application but normally only one PIN code is required. This simplifies the user interface. It is possible to program the second PIN code as an administrator PIN code. For

example, this would be used to unblock the card in case the user either forgot the first PIN code or entered the code incorrectly too many times in sequence.

The PIN can be assigned and stored in the card during personalization. The application program can supply the PIN code in two different ways. If the user has an intelligent smart card reader attached (for example a smart card reader with a keyboard and display), then the application can ask the smart card reader to display the password prompt and accept the user′s input. If the user has a simple smart card reader attached, then the PIN code can be entered from within the application program itself and send to the smart card reader.

With the profusion of smart card applications, people are required to remember more and more PIN numbers. This puts a strain on the user′s memory. After all, who can remember 15 or 20 different PIN codes? Sometimes, people jot down the PIN number on the card itself as a means to jog their memory. This is dangerous as it nullifies the advantage of having the PIN in the first place. That is why recent emphasis on security measures have revolved around biometric measurement techniques as a means of identifying a person.

2.7.2 Biometrics

Biometrics is the science and technology of measuring human biological features to unambiguously identify an individual within a group of people. One of the driving forces behind the development of biometric identification technology is the reluctance within the user community to memorize passwords and PIN numbers for identification. Also, a PIN number does not uniquely identify an individual because PIN numbers can be shared among different people (sometimes inadvertently when people write their PIN numbers on the card itself and then lose the card). Biometrics identify the actual person and not the person′s knowledge of a shared secret.

Some of the biological features that are both unique to an individual and that can be measured are:

- Signature
- Fingerprint
- Voiceprint
- Hand geometry
- Eye retina
- Facial recognition

Signatures and fingerprints are two techniques that have been known for hundreds of years. Both techniques are in popular use, the latter most often associated with police identification. Using machines to automate the analysis is relatively new.

Fingerprint analysis is based on mathematical relationships in the direction of cutpoints through the minutia, the lines in your finger. The set of minutia vectors (that is, the number and direction of cutpoints) for an individual are unique for each and every individual. A precompiled minutia vector database can be stored on the smart card because the data takes up very little space, approximately 300-800 bytes. The fingerprint scanned at the biometric station can be mathematically compared to the reference and a statistically good match will be accepted as that individual.

Hand geometry is a biometric technique that uses features of the size and shape of the person′s hand to uniquely single out a person from a group. The

recognition speed is relatively fast. Also, the size of the reference pattern is small, normally 10-30 bytes, so it can easily be stored onto a smart card. The main limitation of hand geometry recognition is that the group size must be small. This technique is usually employed for facilities access where the total number of people to be allowed access is relatively small, a few dozen at the most.

The pattern of blood vessels on the back of the eye retina is also, like a fingerprint, unique in each person. A low-power laser can scan the retina and record the pattern. Like a fingerprint analysis system, the pattern of the retina can be compared to a statically stored pattern on the card and a match will authenticate the individual.

Today, smart cards contain reference data that will be used by the biometric station. The biometric station will compare the dynamic data obtained from the biological feature to the reference data stored in the smart card. The comparison is done on the biometric station and not within the smart card itself. Thus, the security of the system is dependent on the security of the biometric station itself. For example, if a fingerprint sensor is built into a computer station keyboard, the security of the system is no better than the preventative measures in place to prevent tampering with the keyboard and its connecting wires.

Table 1. Comparison of Biometric Key Factors

	Acceptance	**Cost of Enrollment**	**Rejects**	**Substitutions**	**File Size (bytes)**	**Relative Device Cost**
PIN	50%	Low	1%	0.1%	1-8	very cheap
Static Signature	20-90%	Low	5%	1%	1000-2000	cheap
Static/Ext Signature	20-90%	Low	5%	0.1%	1000-2000	cheap
Dynamic Signature	20-70%	Medium	1-20%	0.01%	40-1000	medium
Fingerprint	0-100%	Medium	1-10%	0.1%	300-800	medium to expensive
Hand Pattern	0-90%	Medium	5%	1%	10-30	medium to expensive
Voice Pattern	100%	Low	10%	1%	100-1000	cheap
Retinal Pattern	0-10%	High	1%	0.01%	80-1000	very expensive

The IBM Smart Card Development Group assembled Table 1 to compare several factors of the various traditional and biometric identification methods. Let's clarify the table columns:

Acceptance Acceptability in the user community for using this identification method. The percentage figure in the table is a probability that a user of the system would accept that technology and use it. Of course, the acceptance rate is only approximate because social factors, country of origin, age and so on will affect these numbers. Generally, a higher number is better.

Cost of Enrollment The "price of admission" or the initial investment that a company must make for this technology.

Rejects The probability of false refusal. This figure is a probability that the device will reject an attempt by an authorized user to use the system. A lower number is better.

Substitutions The probability of false acceptance. This figure is a probability that an unauthorized user will be accepted by the device instead of being rejected. A lower number is better.

File Size The approximate number of bytes that a reference file would occupy on the smart card. Obviously, a lower number is better, at least in terms of smart card memory consumption.

Device Cost The relative unit price in large quantities.

It is probable that biometric techniques will be used in concert with PIN code entry. There are legal reasons for this. Entering a PIN code at a card terminal is more than a means to identify an individual; it is a consent by the person entering the PIN to the operator of the smart card application that he/she agrees to the terms and conditions of the transaction. It is inconceivable that scanning a person′s retina from some distance as that person passes by a sensor is considered a form of consent by that person, in a legal sense.

Biometrics standards are being developed by the BioAPI Consortium (http://www.bioapi.org), but this effort is just starting as this entity was established in April of 1998.

2.7.3 Mutual Authentication

The smart card application and smart card are able to verify the identity of each other automatically when the smart card is inserted into the smart card reader. This automatic check is performed without explicit input from the user but involves the same concepts discussed above with PIN codes. A smart card application designer would want to do this so that the smart card can verify that it has not been plugged into a hostile smart card reader for the purpose of being attacked (this is called internal authentication). Conversely, a smart card reader′s secure application module (SAM) or application program on the host can verify that a smart card that it has detected is an authentic smart card (this is called external authentication).

This is an example of the process of mutual authentication. Before Bob sends any messages to Alice, Bob sends a random number to Alice. Alice encrypts the random number using the DES key that Alice has previously stored and returns the enciphered text back to Bob. Bob decrypts the enciphered text. If the random number Bob decrypts is the same as the number that was sent to Alice, then Bob knows that Alice shares the same DES key.

If both the smart card application and smart card authenticate each other in one process, that process is called ″mutual symmetric authentication″. In some cases, such as when the smart card lacks the ability to do RSA encryption or the smart card reader is offline, only the smart card reader will authenticate the smart card based on signed data preprogrammed into the smart card at personalization. This process of one-way authentication is called ″static asymmetrical authentication″.

2.8 Summary

The battle between the good guys and the bad guys in security is an on-going battle. As soon as the hackers find a security hole, the good guys will provide a fix to prevent its use. Security is constant vigilance.

At the time of writing this document, a new technique to find keys inside a smart card was announced. The technique is called Differential Power Analysis (DPA); according to the researchers that discovered this method, an attacker can gain knowledge of cryptographic keys by analyzing the pattern of power spikes caused by a smart card in use. A smart card will draw varying amount of power from the smart card reader when executing different sections of code. The presence or absence of these patterns over a number of computing cycles can give an attacker knowledge about the cryptographic keys (see ″Cryptography Research Q&A on Differential Power Analysis″, http://www.cryptography.com/dpa/qa/index.html).

The good news is that this technique was discovered by a company, Cryptographic Research, that works in trying to prevent attacks. This company published its finding almost a year after it was discovered, working in the meantime with smart card manufacturers to prevent it.

This attack can be prevented by adequate hardware design and implementation, combined with advanced programming techniques for the security-related features.

The use of a smart card depends almost exclusively on its security. Few of the applications implemented nowadays could continue if for some reason the security of the cards is in jeopardy. And at the same time, a would-be hacker can take all the time in the world in trying to break in. The hacker can take the smart card to his home and work on it as long as he wants, with almost no risk of being discovered. Compare this situation with the common network hacker, trying for example to get into a private network behind a firewall. This last one knows that if he is detected it can be traced back; he must work under pressure and as fast as he can.

But there are several factors in favor of the smart card. The traditional hacker must have in-depth knowledge of the TCP/IP protocol and, in most cases, of the UNIX operating system. In many cases hackers get this education in universities with plenty of equipment to try new tricks.

Compare this situation to the smart card hacker; he will have to work hard to get the skills he or she needs, and the equipment to try is not easily available.

Other factors in favor of the smart card are why the hacker does what he does. There are usually two incentives: the glamour of appearing in the news, and/or a possible financial reward. With respect to glamour, it surely is easier to get the attention of the media by breaking a military network or a government security agency than a smart card.

On the financial benefits, the hacker will not get rich by using a purse that stores a relatively small amount of money. It can get financial benefits by finding cryptographic keys, but probably this would be more related to industrial espionage.

In summary, the smart cards should continue to be as secure as they are nowadays.

Chapter 3. Standards, Specifications and Recommendations

In this chapter, we describe several of the major industry standards, specifications and recommendations for smart cards.

Smart cards themselves usually are a small part of a much more complex system. There are usually complex networks of card terminals connecting to other backend host computers that process the information from transactions occurring at the card terminals. Companies investing in this infrastructure have a vested interest in standardizing the system components to guarantee the longevity of the system. Without standards, different manufacturers′ components would not interoperate. The smart card systems would not be generally accepted because users would be forced to carry around many different, non-interoperable smart cards. This is an untenable situation for both users and manufacturers.

To describe all the standards and specifications related to smart cards would be enough to fill this book. It is important to understand the major industry initiatives for standardization and where they apply so that one can make informed decisions when selecting vendors′ smart card products and services.

In this chapters we will cover standards or recommendations issued by international standard bodies, like ISO and CEN, others issued by companies that are in the same industry segment, like EMV from VISA, MasterCard and Europay, and also others issued by individual companies that were later on accepted by the security and smart cards community, like PKCS from RSA or CDSA from Intel.

There are also encryption standards, which we have described in Chapter 2, “Smart Card Security” on page 11.

3.1 ISO 7816

The International Organization for Standardization (ISO) 7816 standard is a set of eight documents describing all physical aspects of integrated circuit cards (or ICCs as smart cards are known in the financial industry). ISO 7816 is follow-on work from the ISO 7810, 7811, 7812 and 7813 specifications which laid down the basis for ID-1 ″credit card sized″ identification cards. The ID-1 card size have long been used in the financial industry for credit cards, bank cards, health cards, and so on. See Figure 7 on page 31.

Smart card standardization began in 1983 and the first of the ISO 7816 specifications came out in 1987. The eighth part came out as late as 1998 and work on new parts is still going on.

The set of documents entitled ″ISO 7816 Identification Cards - Integrated Circuit(s) Cards with Contacts″ consist of:

- Part 1: Physical characteristics

  This specification defines characteristics of IC cards such as their dimensions, mechanical strength, response to UV light and x-rays, etc. Figure 6 on page 28 shows the dimensions of an ID-1 smart card. See also Figure 7 on page 31.

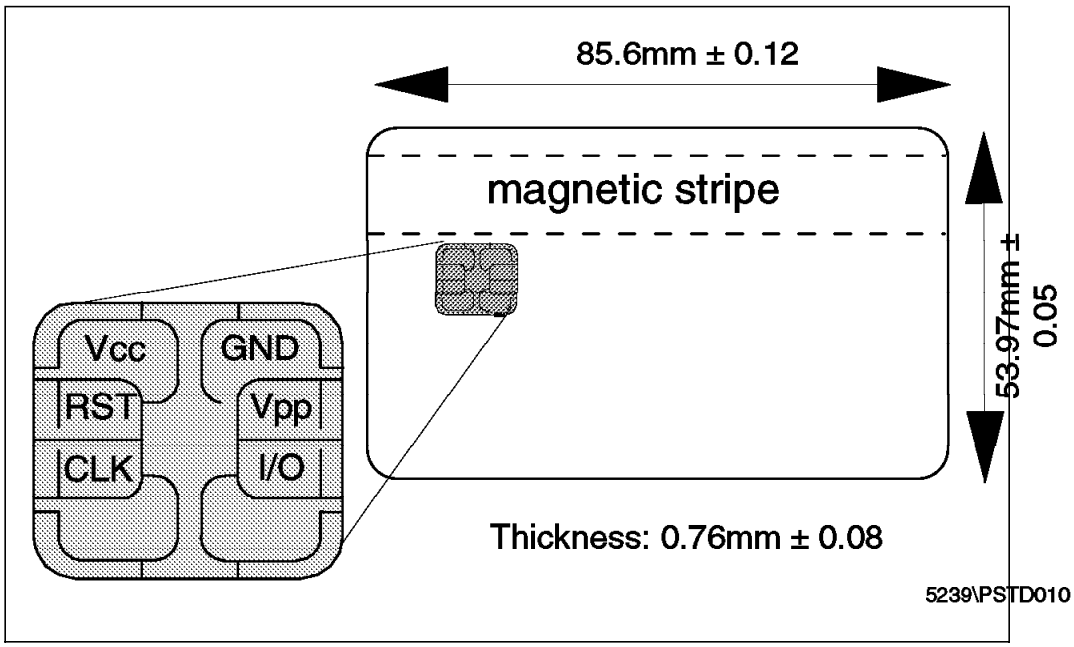


Figure 6. ISO 7816-2 Dimensions and Contact Location

- Part 2: Dimensions and location of the contacts

  This specification defines dimensions and location of the card contacts, the location of embossing, magnetic stripe and arrangement of the chip. Figure 6 shows a typical contact pad arrangement on a smart card.

  It is noteworthy to mention that ISO 7816-2 changed the location of the contact pad to its present position from the position the majority of cards had occupied up to that point in time. French-made cards up until 1987 had complied with the French national standard, AFNOR, which located the contact pad closer to the top edge of the card. This pad location interfered less with embossing on the card.

- Part 3: Electronic signals and transmission protocols

  Part 3 is very important. It describes the basic electronic characteristics of the chip and power requirements. Figure 6 shows an expanded view of the contact pad of an ID-1 smart card. The pads' functions are labeled:

  Vcc This is the voltage supplied by the card reader to the card chip. It is usually +3V to +5V DC.

  Gnd This is the power return or ground lead.

  RST This is the pin that receives the physical reset signal from the smart card reader. The smart card will respond immediately to this signal. See the description of Answer-To-Reset below.

  Vpp This used to be a programming voltage pin for programming the EEPROM on some smart cards. Because modern smart cards have internal voltage control circuits, this function is becoming less used.

  CLK This is the 3.5MHz to 5MHz external clock signal from the smart card reader that operates the smart card CPU.

  I/O This is the serial input and output pin. Data flows between the smart card and the smart card reader over this pin. Because smart cards use only one wire to communicate with the smart card reader, information can only flow in one direction at a time.

The last two pads are not used and are reserved for future use.

Part 3 also defines the response the card must give to a reset signal, known in smart card terminology as the Answer-To-Reset or ATR (see 8.3.1, “Answer to Reset (ATR)” on page 96 for more information). The primary use of the ATR is to tell the smart card reader the type of card that has just been inserted into the smart card reader. It goes on to describe the basics of data transfer at the physical level between the smart card and the smart card reader. Finally, there is a description of the smart card data transfer protocols (please refer to 4.7.1, “Card Communication Protocol (T=0, T=1)” on page 49 for further information about these protocols).

When manufacturers claim to be ISO compliant, they mean, in fact, that they comply fully with parts 1, 2 and 3. Parts 4 and onwards have many optional requirements such that one manufacturer′s card may not work in an application in which another manufacturer′s card works. This is possibly because the first smart card does not implement a part of the ISO standard that the second smart card implements. It is necessary to review carefully and question often exactly what a manufacturer means by saying ISO compliant.

- Part 4: Inter-industry commands for interchange

  This document is a thorough description of the instruction set for smart cards. Smart cards communicate information with a smart card reader via command blocks called ″Application Protocol Data Units″ or APDUs. This document describes the basic set of APDUs that smart cards must understand. This document goes on to describe data organization on the smart card including file types, file structure, security architecture, Application Protocol Data Units (APDUs), Transport Protocol Data Units (TPDUs), secure messaging, return codes and logical channels. More information about APDUs with examples can be found in 10.1, “The Card Instructions” on page 119.

- Part 5: Numbering system and registration procedure for application identifiers

  This defines the numbering system for applications within smart cards. The unique number given to an application on a smart card is called the Application IDentifier or AID. The AID helps to organize and collect files on the smart card that relate to a specific application (see 8.3.2, “Application Registration” on page 98 for more information about AIDs). In most cases, the applications within a smart card refers to the data stored within the smart card rather than code that is to be executed on the smart card CPU. The exact AID data structure is defined in Part 5 of the ISO 7816 specification. All smart card applications must be registered with ISO before smart cards containing those applications are issued. The AID registration procedure is also explained in Part 5 of the ISO 7816 specification.

- Part 6: Inter-industry data elements

  This document discusses the definition of data elements and relevant Type-Length-Value (TLV) structures for inter-industry use. More information about TLVs can be found in 8.1.3, “Data Structures” on page 91.

- Part 7: Enhanced inter-industry commands

  This working document covers additional commands for smart cards to make them comply with ISO 7816 part 4. This includes commands for SQL access to smart cards, mutual authentication and encrypting.

- Part 8: Inter-industry security architecture

This is a working document that covers the definition of a detailed security architecture for integrated circuit cards.

The majority of smart cards on the market comply with ISO 7816 parts 1, 2 and 3, a specific example being the ZKA ″Geldkarte″ (see F.3.1, “GeldKarte” on page 197 for a discussion about the Geldkarte). More information about ISO 7816 itself can be found by contacting the International Organization for Standardization on the Web at http://www.iso.ch/.

3.2 CEN726

The European Telecommunications Standards Institute (ETSI) determines and produces standards for the telecommunications industry. Like the ISO 7816 specification, the CEN726 standard defines the command set (APDUs), the access conditions, the file structures and the card terminal requirements for ICCs used in the telecommunications field. The reason that ETSI defined the same specifications as did ISO is because the ISO standards were not ready at the time that the telecommunications industry needed it. ETSI drafted the first specification as EN 726. Maintenance of the original ETSI specification was taken over by the Comité Européen de Normalisation (CEN) and the EN 726 standard renamed to CEN726.

The seven documents comprising the CEN726 specification are:

- Part 1: System overview
- Part 2: Security framework
- Part 3: Application independent card requirements
- Part 4: Application independent card-related terminal requirements
- Part 5: Payment methods
- Part 6: Telecommunications features
- Part 7: Security module

CEN726 shares the data structures, instruction set and return codes of the ISO 7816-4 standard.

The Dutch student card made by Chipper is an example of an CEN726 compliant smart card. More information about CEN726 can be found at http://www.etsi.fr/.

3.3 GSM

Although detailed coverage of the worldwide digital mobile telephone network Groupe Spéciale Mobile (GSM) standard is outside the scope of this book, we have included a summary here for completeness.

This standard was originally published by the Comité Européen de Normalisation (CEN) as GSM 11.11 but has since been renamed. The GSM standard is written in two parts. The first part, CEN prETS 300509, describes the general functional characteristics of the GSM network. The second part, CEN prETS 300608, covers the interface and logical file structure of the smart card Subscriber Identity Module (SIM). CEN prETS 300608 defines the physical ID-000 module size (see Figure 7 on page 31), the electrical operating parameters, the SIM logical file structure and the GSM instruction set.

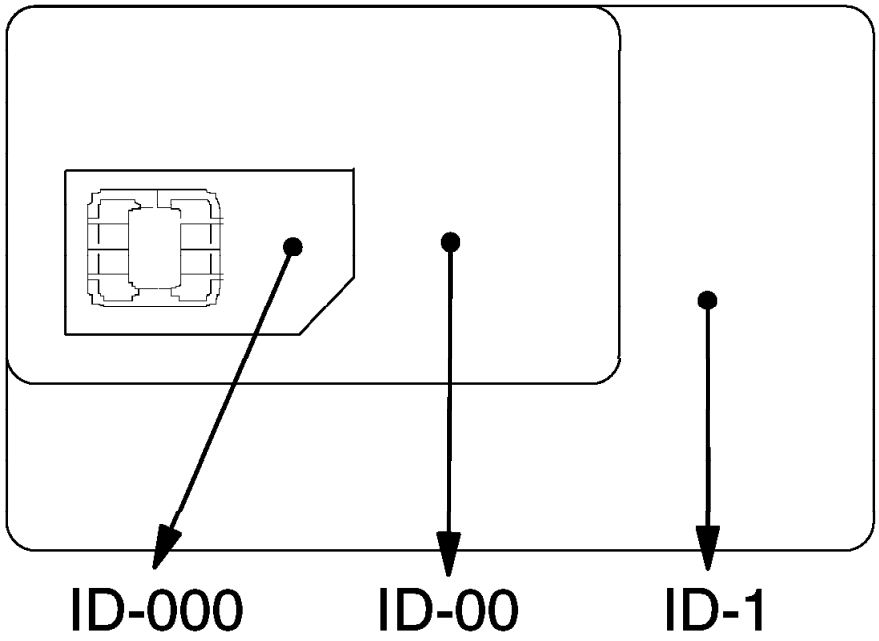


Figure 7. Derived Smart Card Sizes

The ID-000 format arose because the degree of miniaturization in the mobile telephone industry had progressed to a point where the ID-1 format was too large to fit in the actual phone. The ID-000 format was intended for minimal handling where the module would be inserted and left alone.

The dimension and format for the ID-000 size is derived from the ID-1 card size. Figure 7 shows the size relationship between ID-000 SIMs and ID-1 smart cards. This allows card manufacturers to standardize the production lines for one size card, imbed the modules and, in a separate production step, modify the cards from ID-1 to ID-000 by punching out the ID-000 outline in the plastic.

More information about GSM can be found by contacting ETSI.

3.4 EMV

In 1996, Europay International S.A., MasterCard International Incorporated, and VISA International Service Association jointly published a specification called ″IC Card Specifications for Payment Systems″. This set of three documents have become known as EMV after the first letters of the three parties who cooperated on the specification. EMV is three documents that cover design aspects for smart cards, smart card terminals and smart card debit/credit applications as they relate to use in credit card transactions using smart cards for storage of security information. The EMV specification refers to smart cards as Integrated Circuit Cards or ICCs.

The first EMV document is the Card Specification. It has many similarities to the first two ISO 7816 standards. EMV and ISO 7816-1/-2 define the same characteristics for ICCs so that ICCs which conform to one specification will be compatible with the other specification.

The next EMV document is the Terminal Specification. The terminal specification details mandatory and optional requirements for card terminals used for credit card transactions including automated teller machines (ATMs), branch terminals,

cardholder-activated terminals, electronic cash registers, personal computers, and point of sale (POS) terminals. In addition, it covers the requirements for:

- General physical characteristics of the card terminal
- Software architecture including software and data management
- Security
- Cardholder interface
- Acquirer interface

The last EMV document, the application specification, defines procedures necessary to effect a payment system transaction in an international interchange environment. These procedures include:

- Mapping of data elements to files
- Transaction flow
- Exception processing
- Coding of specific data objects described in the card specification

The MPCOS smart card made by Gemplus is EMV compliant and is certified by VISA International. However, since there are no procedures or tools in place to verify conformance to the EMV specification, any claims to EMV compliance are debatable. Usually any claim of EMV compliance means that the manufacturer has implemented one or more parts of the specification, most probably the application selection mechanism because this is the most interesting aspect of EMV.

It is interesting to note that, at the outset, there was a precondition that card terminals would not contain secure application modules (SAM) for verifying the ICCs to the card terminal. The reasoning was based on the enormous cost of managing the worldwide distribution and update of symmetrical keys to partially offline or wholly offline card terminals. Thus, the EMV specification requires the use of unidirectional static authentication with card-specific keys to verify the smart card to the card terminal (see 2.7.3, “Mutual Authentication” on page 24 for details about static authentication). This is a rather weak security mechanism but it is low risk because no debiting is performed on-card during a transaction. Rather, a ″voucher″ for the transaction is submitted to the card terminal and the only way to convert the ″voucher″ into an actual cash transaction is for the merchant to submit the ″voucher″ to the credit card company for clearing. Only licensed merchants can do this.

In our case study, you will see that GTT has a requirement for EMV compliance. This requirement exists because GTT needs the application selection mechanism and not because they need a credit/debit payment mechanism.

More information about EMV can be found at http://www.mastercard.com/emv/.

3.5 PC/SC

The PC/SC Workgroup was established in May 1996 to address the limitations in smart card technology then available for the PC platform. Its intention was to draft specifications for smart cards and card terminals for use on PCs. Microsoft Corp. owns and maintains the PC/SC specification. The workgroup identified three areas to standardize:

- The interfacing of card terminals (known in the specification as InterFace Device or IFD) to the PC

- The high-level application programming interface to access smart card functionality
- The mechanisms to allow multiple applications to effectively share the resources of a single ICC and IFD

The final specifications resulting from the workgroup's activities were announced at the April 1998 CardTech/SecurTech conference. The PC/SC Workgroup specifications are divided into eight parts. A brief summary of each part is provided below:

- Part 1 - Provides an overview of the system architecture and components
- Part 2 - Details compliant ICC-IFD characteristics and interoperability requirements.
- Part 3 - Describes the interface to compliant IFD devices and required functionality for those devices
- Part 4 - Discusses design considerations for IFD devices
- Part 5 - Describes the interfaces and functionality supported by the ICC Resource Manager
- Part 6 - Describes the ICC Service Provider model
- Part 7 - Describes design considerations for application developers and how to make use of the other components
- Part 8 - Describes functionality for ICCs with cryptographic support

Microsoft publishes a complete list of PC/SC-compliant smart card readers on its Web site at http://www.microsoft.com/smartcard/. More information about PC/SC can be found at http://www.smartcardsys.com/.

3.5.1 PC/SC Migration Interface

The PC/SC Migration interface allows an application developer to write to the PC/SC interface specification and use the equivalent services in a non-Windows environment. The smart card reader manufacturer needs to supply a hardware driver that works with the PC/SC Migration interface in the specific platform. Figure 8 on page 34 shows a comparison between PC/SC and PC/SC Migration. The PC/SC Migration Resource Manager implements only a subset of the functions of the API of PC/SC standards.

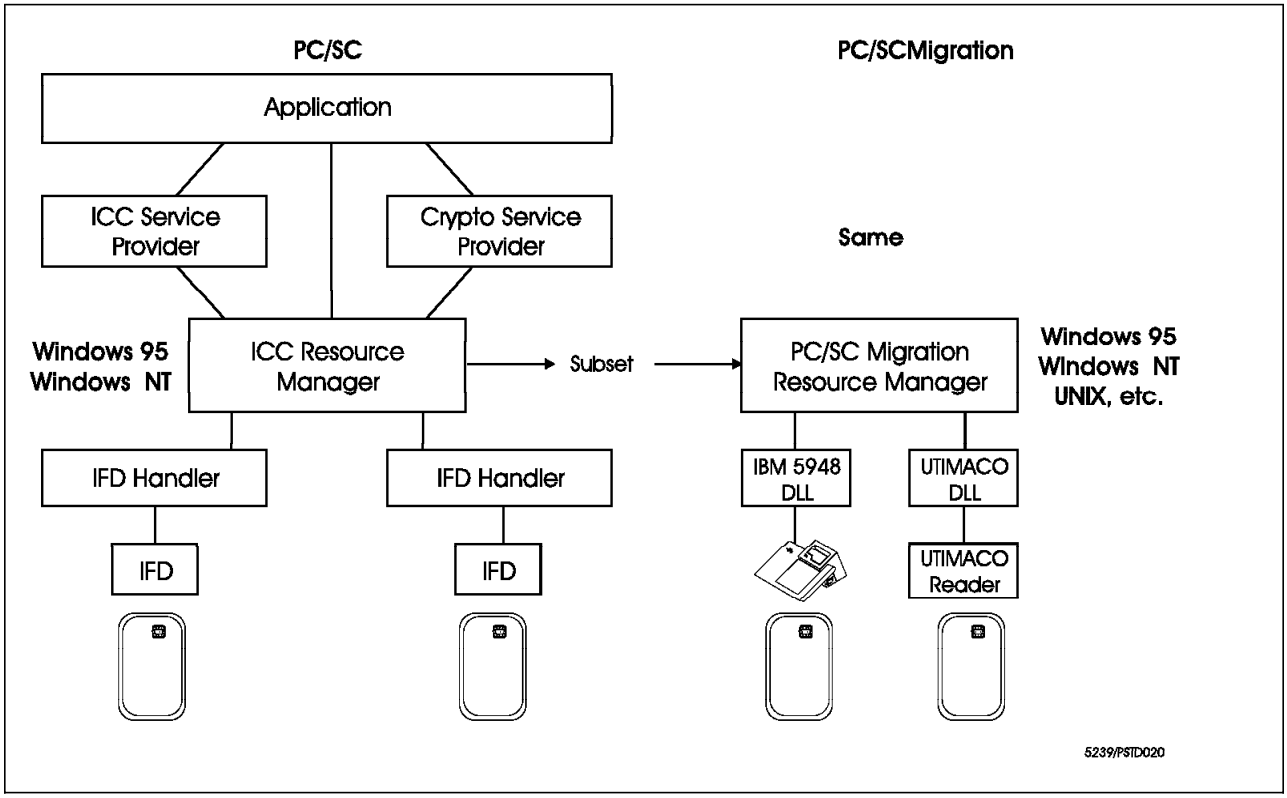


Figure 8. PC/SC vs PC/SC Migration Interfaces

Some examples of applications using the PC/SC Migration interface are:

- IBM Digital Signature Solution
- OpenCard Framework
- IBM Smart Card ToolKit

3.6 OpenCard Framework

Twelve industry leaders (Bull Personal Transaction Systems, Dallas Semiconductor Corporation, First Access, Gemplus, IBM, NCI, Netscape Communications Corp., Schlumberger, SCM Microsystems, Sun Microsystems, Inc., UbiQ Inc., and Visa International) joined their efforts to release a new technology specification, called OpenCard Framework, that helps software developers create smart card applications that can be used across a variety of business and consumer devices such as PCs, network computers, automatic teller machines, point-of-sale terminals, set-top boxes and emerging handheld devices and established the OpenCard consortium.

The OpenCard Framework resulted from work done by an ad hoc industry group committed to simplifying the use of smart card technology across different manufacturers′ systems. Version 1.0 of the OpenCard Framework Reference implementation was released in April 1998.

The Open Card Consortium has established a Management Board, named a Secretary and established several work groups, selected from within the consortium members, to develop further OpenCard Framework specifications. These groups are:

- Core & Infrastructure Workgroup
- Card Services Workgroup
- Card Terminal Workgroup
- Industry Specific Applications
- Testing & Compliance Workgroup

The OpenCard Framework provides a common interface for both the smart card reader and the application on the card. Basing the architecture on Java technology has resulted in enhanced portability and interoperability, which are key to widespread adoption. The Version 1.0 reference implementation also enables interaction with existing Personal Computer/Smart Card (PC/SC) 1.0 supported reader devices.

The OpenCard Framework can be divided into two main components:

- The CardTerminal component
- The CardService component

These hide the Terminal/Card specific details, and let you develop applications independently from any terminals or reader and any smart card.

The CardTerminal component contains classes and interfaces that allow you to access card terminals and their slots. Using these classes you can, for example, find out whether a smart card is inserted in a card terminal.

The OpenCard Framework offers significant advantages to application and service developers as well as card and terminal providers.

Application and service developers benefit from the OpenCard Framework as follows:

- Vendor independence -- Developers can choose cards and terminals from different suppliers.
- Asset protection -- Extensibility of the architecture enables developers to participate in future developments of smart card technology at a low cost by migrating at the level of the API.
- Improved time-to-market -- Developers profit from shorter development cycles by programming against a high-level API.
- Lower development cost -- Developers save the extra cost of porting their applications to different platforms and benefit from lower skill requirements to accomplish a given task.

Card and terminal providers benefit from OpenCard Framework in these ways:

- Increased market -- Providers gain access to new market segments reaching many more customers.
- Improved competition -- Providers can compete in terms of functionality and are less vulnerable to the predominance of a single vendor.
- Less development effort -- Providers inherit functionality provided by the framework, which reduces their development efforts.

Information on the consortium and copies of the OpenCard Framework 1.0 reference implementation and source code are available on the World Wide Web at http://www.opencard.org.

More details on this subject can be found in Appendix D, “Overview of OpenCard Framework” on page 185.

3.7 IATA Resolution 791

The International Air Transport Association is the control body for airlines worldwide. In 1996, the Smart Card Subcommittee drafted a resolution, IATA Resolution 791 ″Specification for Airline Industry Integrated Circuit Cards″. Resolution 791 describes the minimum standards for ticketless travel. This resolution covers smart card and smart card reader interoperability, smart card file organization and common data for flight information, itinerary, and immigration data. This resolution is a recommendation to the airline industry as to how they should go about designing a ticketless reservation system that would be compatible with other smart card activities including ICC credit cards based on EMV.

American Airlines offer its frequent flyer and business club members the option of ticketless check-in conforming to IATA Resolution 791. More information about IATA Resolution 791 can be found at http://www.iata.org/smartcard/index.htm.

3.8 SEIS

″Secured Electronic Information in Society″, or SEIS, is the name of a nonprofit organization in Sweden, which drafted specifications for the secure creation, distribution and use of electronic identity cards (EID cards). According to SEIS, ″an EID-card is a cryptographic token, with a publicly accepted picture type identity card surface, also containing safely stored private keys and certificates which at least certify the ID attributes of the card holder corresponding to the picture and data printed on the surface″.

SEIS depends on public key cryptography and digital signature technologies (see Chapter 2, “Smart Card Security” on page 11 for detailed information about these topics). Products developed according to SEIS specifications will allow:

- The origin and content of information to be verified by means of digital signatures
- Information to be protected against unauthorized or uncontrolled access

SEIS has also drafted a specification of the codes of practice that companies should follow for the distribution and use of EID cards. This is one of the first standards bodies to address the important area of card management, certificate management and directory services. This is an important consideration because the potential for misuse of EID cards is great. At the time of writing, there are no known agencies conforming to this recommendation. More information about SEIS can be found at http://www.seis.se/.

3.9 Cryptoki

The Public-Key Cryptography Standards (PKCS) are a set of standards for public-key cryptography written, owned and maintained by RSA Laboratories. As of this writing, there are 13 specifications in the PKCS family. The eleventh specification, PKCS #11, defines a technology-independent programming interface, called Cryptoki. Cryptoki stands for ″Cryptographic Token Interface″ Cryptoki is an application programming interface that hides the details of

hardware cryptographic devices and presents a model of the cryptographic device, called a cryptographic token (or simply token), to the application. A smart card can be a token. Applications access tokens through ″slots″. More than one application thread can simultaneously access one or more tokens through their respective slots. Thus Cryptoki also implements a multiprogramming environment for token access.

IBM′s MFC 4.0 and 4.21 smart cards contains a crypto-coprocessor on the chip, giving the application programmer access to a specific-use hardware processor.

More information about the PKCS specifications can be found at http://www.rsa.com/rsalabs/pubs/PKCS/

3.10 CDSA

Similar to PKCS #11, the Common Data Security Architecture (CDSA) is a specification for a cryptographic API written by Intel Corp. in cooperation with the OpenGroup.

CDSA supports access by an application program to generic cryptographic interfaces for generation and verification of digital signatures, public key delivery and validation, certificate management and bulk encryption services. It also optionally supports cryptographic key recovery. CDSA allows third-party manufacturers to implement their CDSA interfaces through the use of program plug-ins.

The smart card plug-in for CDSA allows application programs to access cryptographic key and certificate functions from removable smart card-based tokens. This is similar to Cryptoki; however, Cryptoki does not support removable tokens. Also, Cryptoki does not provide as complete an implementation of certificate and key management as does CSDA. More information about CDSA can be found at http://www.opengroup.org/security/cdsa/.

3.11 Co-operative and Competitive Standards

There are many specifications, standards and recommendations covering all aspects of smart card design and development. This chapter has touched on several. However, there are many others which we have not mentioned. The reader is encouraged to consult other books and Web resources that discuss smart card standards, such as:

- ″The Smartcard Handbook″, W. Rankl. and W. Effing, John Wiley 1997. ISBN 0-471-96720
- Semper′s links to Cryptography and Standards: http://www.semper.org/sirene/outsideworld/standard.html
- Smart Card Industry Association (SCIA) links to Standards: http://www.scia.org/resources/industry.htm

It seems confusing at first but it is possible to organize the specifications to show the smart card interface levels to which they apply. Table 2 on page 38 attempts to show, in a broad, general way, the major standards and specifications for smart cards and the scope of their coverage. Some standards address only one aspect of smart cards, others address several. Many smart

card standards and specifications make reference to and use aspects of other standards. So, even though they may appear to redefine parts of other specifications, in fact the spirit of those specifications are attempting to address a completely different aspect of smart card standardization.

Table 2. Scope of Smart Card Specifications and Standards

		ISO 7816	CEN726	PC/SC	EMV	IATA 791	Cryptoki	GSM	SEIS	OpenCard Framework
CARD	Physical Card	√						√		
	Contact Pads	√						√		
	Card Reader		√	√	√			√		√
INTERFACE	Logical File Organization	√	√					√		
	Command Set	√	√	√	√			√		
API	Application Programming Interface			√			√			√
APPLICATION	Digital Signature	√					√			
	Ticketless Travel					√				
	Digital Telephony		√					√		
	Debit/Credit		√		√					
CARD MGMT	Card Issuing and Card Management								√	

3.12 Standards Overlaps

Some of the standards described in this chapter address the same layers, and may seem incompatible. Two such standards are PC/SC and OpenCard Framework.

OCF and PC/SC can and will coexist on Wintel platforms. The present implementation of OCF can utilize PC/SC on the Wintel platform.

The question of whether one should base a development on PC/SC or OCF will largely depend on the foreseen operating environment for the smart card-aware application. PC/SC primarily addresses transparency with regard to card terminals and to a limited extent to card operating systems. OCF in addition provides transparency for the application programmer also with regard to the card issuer and the particular card application management scheme.

You should consider using PC/SC if you are targeting non-Java applications for Windows and you only need to support a particular card issuer.

Since OCF allows the use of PC/SC readers and OCF card terminals through Java in a Windows environment, it is the obvious choice for all Java-based applications.

You should use OCF when you have to support, besides PCs, many systems that use smart cards today and do not exclusively run Windows NT/95 presently and will not, for various reasons, e.g., resource requirements, likely do so in the future (for example, a POS terminal, set-top box, or a smart phone, or just not having enough resources). A subgroup of the OpenCard consortium is working on a scaled down version of OCF to address these environments. Smart card solutions developed for those systems currently only have two choices:

- They can either be tailored in an inflexible way to a given reader and card
- Or they can be based on the Java platform and make use of OpenCard Framework[2]

3.13 Smart Card Organizations

Two organization that gather most of the smart card key players are the Smart Card Forum (http://www.smartcardforum.org) and the Global Chipcard Alliance (http://www.chipcard.org).

Several companies that market smart cards solutions belong to both organizations; IBM is one of them. While there is overlap within the two membership bases, there is a different objective on what services these organizations provide to their members. The Smart Card Forum is a consortium comprised mainly of users who focus on applications and business issues with the support of manufacturers and technology suppliers. The Global Chipcard Alliance is composed by smart card solution providers to forge business alliances and interoperability agreements.

Both organizations deal with standards and interoperability. They announced in April 1998 that they will work together on the issue of interoperability.

[2] Today the use of Java in such terminals is an exception.

Chapter 4. Card Selection Process

Whether it's establishing a secure conversation on a mobile phone, decrypting TV signals inside satellite receiver sets or replacing coins when using a pay phone, the smart card has now proliferated into a diverse range of applications in our everyday life. To meet this demand, there are a wide range of smart cards available from card vendors, employing a vast array of chips supplied by different manufacturers. Naturally, this makes the task of selecting a card rather baffling.

To make the card selection more informed, we have identified distinguishing features of different smart cards and explained the pros and cons of each feature. Later in this book, we use our case study as an example and show the rationale behind choosing the GTT Employee Card.

4.1 Card Selection Considerations

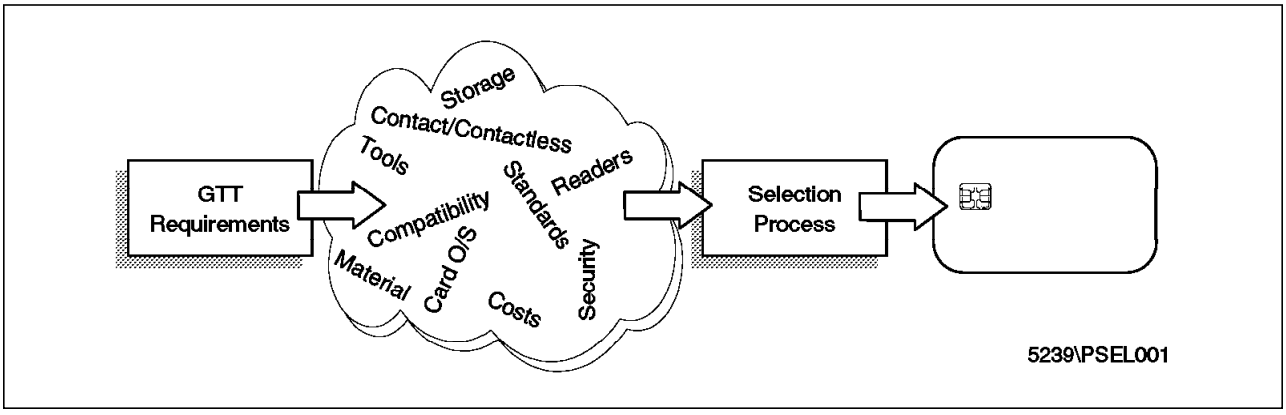


Figure 9. Card Selection Considerations

What influences the decision to select a particular card may just depend on the established reputation of the card supplier or the number of successful projects using a particular smart card in the public arena.

A better approach to the card selection process is perhaps to begin by first looking at the application, for example an airline card or a hotel card and then how or where the card will be used, for example in rapid transport or kiosks. To help with deciding upon a suitable card, we have described the different smart cards available and characteristics that influence the card selection using the following categories:

- Card type
- Interface method
- Storage capacity
- Card operating system functions
- Standards compliance
- Compatibility issues
- Reader interoperability

- Security features
- Chip manufacturers
- Card reliability and life expectancy
- Card material
- Quantity and cost

4.2 Card Type

Smart cards are available in two basic types, the distinguishing feature being whether the card has a microprocessor (CPU) or not. Cards without a CPU are called memory cards and those with are called intelligent cards, chip cards or microprocessor cards. The general usage of the term ″smart card″ is usually for cards with a CPU.

Each type of card, with its own particular characteristics relating to cost, operational simplicity or functional superiority has application in particular segment of the market.

Although in our case study, we will have to use a card with a CPU due to the requirements of some of the applications (for example, Digital Signature) we will describe both types of cards for completeness.

4.2.1 Memory Cards

&Mc.s are used for single function application, are inexpensive, and typically used in phone card type prepayment applications. Access to data is managed by a security module in the chip which guards against the data being erased or written to. In payment cards, reducing the card′s value is done by the chip and is irreversible. After use the card is discarded, although presently there is a more graceful destination to the international collectors market. The simple technology enables these cards to be manufactured very cheaply and costs below US$1 per card in large quantities.

This makes it still more expensive than magnetic stripe cards, which cost much less than US$1, but are cheaper than microprocessor cards. The simple technology also makes it possible for the reader devices to be manufactured cheaply. At first it may seem an obvious candidate for moving on to smart card technology. However, extra investment may be required to overcome incompatibilities in terminal infrastructure, and differences in programming APIs. A later migration to microprocessor card technology may also prove costly in the long-term due to the nonstandard proprietary nature of memory cards. The International Organization for Standardization has concentrated its efforts on microprocessor cards and there is very little effort to standardize memory cards within the industry.

The card′s memory for storing data can range from a few hundred bytes up to 8 KB.

A more intelligent version of these cards is available, that is capable of providing more security by authenticating itself, (for example, correctly responding to a random number challenge request from the terminal). This type of card is also capable of PIN verification, but is very limited in their flexibility.

Application Areas: Telecommunications, pre-payment, health insurance number, vending machines, car parking, public transport, frequent flyer cards, simple loyalty schemes.

Examples of Implementation: European phone cards, German health insurance card.

4.2.2 Microprocessor Cards

Microprocessor cards are able to provide read/write function and enhanced security with a CPU. They are more expensive than memory cards costing around US$5-15. With microprocessor cards you can write and update the data, once the card's access conditions are met.

The way a microprocessor card's internal architecture is designed bears a striking resemblance to PCs. The familiar building blocks of a PC are present in this type of smart cards: CPU, ROM, RAM, I/O port and in this case an EEPROM rather than disk for storage.

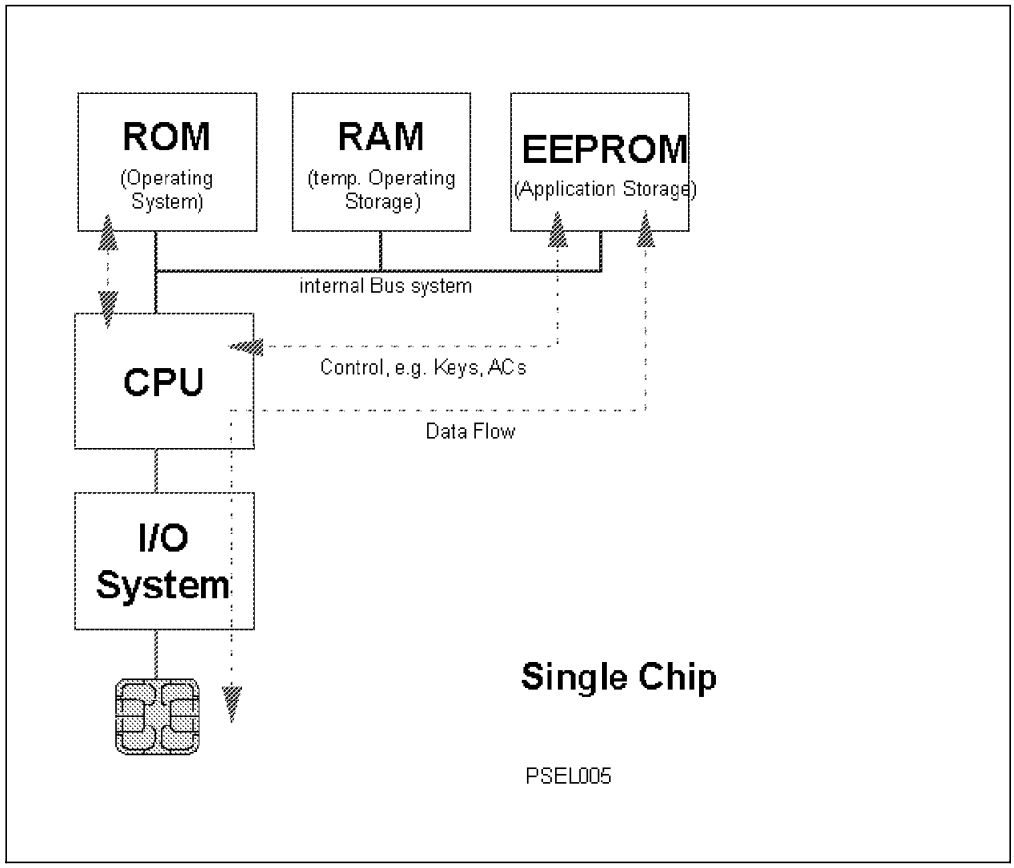


Figure 10. Smart Card Internal Architecture

CPU (Central Processing Unit) The CPU is usually an 8-bit microprocessor with a 16-bit address bus. This makes it possible for this type of processor cards to address up to a maximum of 64 KB.

RAM (Read Only Memory) The RAM is volatile memory that requires power to maintain the data. It provides working storage for the CPU. Usually the size of the RAM is about 256 bytes. The reason for this small size is that RAM memory takes up more space per byte than the EEPROM or ROM memory and is deliberately kept small to meet the

specification for smart card chips, which are limited in total size to 25mm•.

ROM (Read Only memory) The ROM contains the smart card′s operating system and is loaded during chip production. The software loaded is called a ROM-mask. The ROM size can vary from a few KB to around 32 KB depending on the operating system function. For example the IBM MFC 4.1 card has a ROM size of 16 KB.

When speaking of the operating systems the question that is likely to be asked is whether the card has provision for an operating system upgrade post-issuance. Since the operating system is in the ROM, there is no possibility for upgrading the ROM with a new version after the card is issued. (However, the card can execute code written in the EEPROM, which in the broad sense of the word could be used as an operating system upgrade, although its very rarely done in practice. Writing application code requires intimate knowledge of the CPU, which is closely guarded and requires specialist knowledge and skill.)

EEPROM (Electrically erasable programmable read only memory) This is nonvolatile memory and is used to hold all data and programs much like a PC hard disk. The operating system provides file protection by restricting access to the EEPROM. EEPROM sizes can vary in size, and typically this size is selected based on application needs. The most popular size at the time of writing this book is 8 KB, but a particular vendor may offer a selection of sizes to choose from. For example, the IBM MFC 4.1 card is available in 2 KB, 4 KB, 8 KB and 16 KB memory sizes. As technology and requirements evolve, the memory size is predicted to double every few years. At the time of writing this book the maximum size of 32 KB is available for the EEPROM.

Note: A common mistake which occurs when comparing EEPROM sizes of different cards is the way some vendors quote the EEPROM memory in *bits* rather than bytes. This of course translates the popular 8 KB EEPROM to an impressive 64 Kb size.

I/O Port The I/O port is used to transfer the data in a serial fashion, bit by bit. The default speed is 9600 bits per second; some cards support higher speeds.

Co-Processor The more advanced cards are provided with a co-processor which perform the exponential and modular operations on integers when handling encryption procedures, for example with digital signatures.

Usually the card uses an ISO 7816-defined file structure to store and protect the data. This file structure and protection mechanism enables different applications to store data in a single card with ″fire-walling″ between the applications. (Details of this are in Chapter 8, “Card Layout Design” on page 87).

Microprocessor cards are extremely versatile and have found a wide portfolio of applications.

Application Areas: access control, loyalty programs, electronic cash, airline ticketing, credit card, secure messaging, ID card.

Examples of Implementation: Dutch Student Card, Hilton Hotels Check-in, Digital Signature Card, German Geldkarte, Diabcard.

4.3 Interface Method

Both memory and microprocessor cards are available as contact or contactless cards. As the name implies, the main difference is in the method of communicating with the reader device. Contact smart cards require insertion into a reader device. Contactless cards require close proximity to a receiver device, but there is no insertion.

4.3.1 Contact Cards

The card must be inserted into a reader and must make a physical contact with the reader contacts in order to receive power and a clock signal for the chip to operate. The power to the card is switched on or off under the control of the host application or the reader. The reader may lock the card in place to avoid accidental removal before a transaction is completed. There is no restriction on the amount of data transferred or a time limit apart from a usability point of view.

The chip contacts have to adhere to the standard specified by ISO 7816-2, which is described in 3.1, “ISO 7816” on page 27.

4.3.2 Contactless Cards

Although the reliability of smart cards is much higher than magnetic stripe cards, careless handling and very frequent use can take its toll on the gold contact surface, leading to eventual failure of the chip contacts. Contactless cards manage to avoid this pitfall, as they do not require insertion into a card reader and can work up to several centimeters away from the reading device. This lends itself naturally to applications such as mass transit and access control.

The chip derives the power to work from capacitive coupling (close coupling) or inductive coupling (remote coupling). Inductive coupling works on the principle of a transformer where one coil induces current in another coil. The frequency of transmission is usually in the range of a few MHz. The chip is able to transmit a data signal by changing its resistance, which is picked up by the card reader and is interpreted as a data signal.

Writing to a card requires much more power (up to 10 times more) and hence the range is reduced to about 10 cm maximum for inductive coupling cards and about a few millimeters for capacitive coupling cards. Because there is a very short amount of time for the transaction to complete (up to 200ms if the user walks by the reader), the data transmission is limited to a few hundred bytes. These cards are therefore suited for single applications that require a faster transaction speed than contact cards can offer, like mass transit or access control.

The different methods developed to optimize the contactless solution has, however, resulted in incompatibility between different systems. The application of standards are in process (ISO 10536 for close coupling cards and ISO 14443 for remote coupling) and with no established standard the card is likely to be closely tied to the manufacturer′s own reader. The contactless smart cards operate at 4.9 MHz, 6.6 MHz or 13.5 MHz frequencies.

The above contactless cards are called passive since they have to derive the power externally. There are contactless cards that derive the power from an embedded battery and these are active contactless cards used, for example, at motorway toll gates, supporting greater distances between the card and the

reader. This type of contactless cards tend to cost more than contactless cards without built-in battery.

Contactless cards with their specialized readers tend to cost more than contactless cards.

4.3.3 Hybrid Cards

There is some overlap where different card types are implemented in one card leading to the so-called hybrid cards. A smart card can also have a magnetic stripe and this can provide a convenient path for migration.

But the term hybrid card generally refers to cards that have both a contact and a contactless interface. The contact interface is used by the microprocessor chip module and the contactless interface is used by the memory chip module. There is no physical connection between the two chips and therefore no shared memory is available.

4.3.4 Combi Cards/Dual Interface Cards

Combi cards also have a contact and contactless surface but the two interfaces are connected and have access to one shared data area via a microprocessor or logic module. The contact surface is always controlled by a microprocessor. The shared data area may be controlled by a microprocessor or a logic module. An example where the shared memory is controlled by a logic module is the ″Mifare Plus″ combi bard. An example where it is controlled by a microprocessor is the ″Mifare II″. Both are produced by Philips/Mikron.

Dual interface cards have one processor chip that can be operated through both I/O ports, the contact and the contactless. The smart card software controls the port usage by application; for example, loading the e-purse via contact side only, spending of small amounts through any port.

The disadvantage of a contactless/combi card over a contact card is the increased cost required to implement the antenna into the plastic card, and more expensive readers that need to have radio frequency (RF) transmitter/receivers. Combi card performance is likely to be slower than a contactless card because the RF unit has to get the data via the CPU. Another disadvantage with this type of card is that communication can be interrupted by removing the card from the RF field, or it can be possibly traced or disturbed.

4.3.5 Optical Cards

For applications where a very large amount of storage capacity is required, optical memory cards are available, which can store for instance X-ray images of a patient. These cards usually have a microprocessor chip embedded and use the smart card security to protect the optical data from unauthorized access.

The optical card provides some megabytes of write-once/read-many (WORM) storage. Data can be read by appropriate devices and is not protected, unless it is encrypted.

4.4 Storage Capacity (EEPROM)

The most commonly quoted memory size for today's smart card is 8 KB of cardholder and application-specific data space (see page 44). This may seem more than adequate to begin with, but there is a parallel with selecting a hard disk for a PC. The data fills up to whatever size of disk is chosen. Cards with 1 KB, 2 KB or 4 KB available at a lower unit cost can be chosen if this meets the storage requirements.

The quoted memory size (for example, 8 KB) can give a misleading picture of the usable storage because of other overheads.

For example:

- Use of EEPROM memory by some operating system extensions will reduce the storage area for data.
- Defining the file structure carries a small overhead, which means the complexity of the file structure has to be balanced with the requirements for data storage.
- Data files imposed by standards (for example an application directory file) will require some space.
- Key storage for each application provider will multiply the storage required for keys.
- Digital certificates, photo images, fingerprint are likely to take up several thousand bytes, which can significantly reduce the storage available.
- A purse application may have separate keys for loading, debiting and other functions that demand extra space.
- Long keys used for strong encryption will demand more storage.

In our case study, GTT has many applications and storage could be a limitation on the number of applications we decide to implement. In Chapter 8, "Card Layout Design" on page 87 we have discussed GTT's application requirements and calculated the memory that the smart card needs to satisfy.

4.5 Card Operating System Functions

A basic decision that will have to be made in our case study is which card operating system to use. The card operating system governs the level of function and standard compliance delivered with the card. Unlike a PC, however, the card operating system is loaded into ROM during manufacture and upgrading to a subsequent release is not possible. In some cases, card suppliers may be requested to create a special version of the operating system in order to provide unique card function. An example of such a modification is the IBM ZKA card supplied to German banks.

The key points to bear in mind is whether a manufacturer's particular version of a card operating system can deliver the function and permit interoperability.

For example:

- Digital signatures would require a card that is capable of handling public and private key encryption (RSA).

Handling the cryptography requires a co-processor and the card operating system must be able to handle it.

- An electronic purse debit/credit application would require conforming to a payment standard such as EMV.

  One element of EMV compliance, for instance, is that the card operating system would need to be able to access the files using the application ID (See 8.3.2, “Application Registration” on page 98).

Note: The operating system compliance with a standard might actually mean part compliance rather than full compliance during the evolving stages of the standard as well as the card operating system.

4.5.1 Interpretative Card Operating Systems

A departure from the traditional implementation of smart card with its rigid data structures to a flexible object-oriented structure is an interpretative card operating system. In our case study, GTT will need to make the decision whether to select an interpretative card operating system or use a traditional smart card. Examples of interpretative card operating systems are JavaCard and Multos.

The JavaCard and its implications are described in Appendix B, “JavaCard Overview” on page 179. Multos is discussed in 199.

4.6 Standards

Our GTT case study requirements include ensuring cross border inter-operability, multi-vendor sourcing and useful operation of the card throughout its life. In order to comply with these requirements, reviewing the card′s compliance with industry standards, specifications and recommendations is a mandatory step in the card selection process. These standards cover a broad range of specifications, from the physical locations of the chip contacts to specific application requirements.

The standards are described in Chapter 3, “Standards, Specifications and Recommendations” on page 27.

4.7 Compatibility Issues

The smart card technology is advancing rapidly and this has led to some differences in the card operating environment. Some points to bear in mind to avoid incompatibilities are:

- Card Communication Protocol (T=0, T=1)
- Voltage Supply (5V, 3V)

In GTT′s case, which is a new and closed operation with no legacy smart card infrastructure, compatibility is not a major concern. However, compatibility issues will need to be taken into consideration in any future enhancements to the system.

4.7.1 Card Communication Protocol (T=0, T=1)

When the card is first powered on, the card responds with an ATR indicating the type of communication protocol it's prepared to use. (See 8.3.1, "Answer to Reset (ATR)" on page 96 for more information). The standards have made provision for 15 possible data transmission protocols, each designated as a 'T =' followed by a number between 1 to 15. Currently two protocols T=0 and T=1 are most commonly used.

The T=0 is an implementation first used in the French pay phone system which was later standardized. Its adoption by other phone companies (example GSM) has given it a boost and is by far the most common protocol in use today. It was designed for early systems and used a simple byte-by-byte transmission technique that had minimum memory requirements. The T=1 protocol uses a block (a sequence of bytes) during its transmission and is more suited for secure messaging and modern sophisticated interfacing devices. The T=14 is used in Germany payphones and the remaining protocols are not yet defined or are in the process of being prepared. The EMV standard addresses both T=0 and T=1 protocols.

Some readers are designed to handle only one type of protocol, whereas others are designed to automatically switch protocols and handle both types of cards.

When selecting a card it is usual to adopt the T=1 protocol unless there is a requirement to use T=0 cards to maintain compatibility with the terminal infrastructure.

4.7.2 Voltage Supply (5V, 3V)

One topic of frequent debate is the question of the operating voltage of the card. This is quite important since a chip without the correct operating voltage would not function at all or at worst come up with spurious errors. The early standards established the operating voltage of cards at 5 volts since that was the established supply voltage in use for integrated circuits. As chip technology involving silicon has advanced and the devices using smart cards, such as mobile phones, demand less power consumption from the circuitry, chips operating at 3V have become the accepted norm. An added benefit is that less heat is dissipated, allowing the chip to operate at a higher frequency and hence much faster.

Cards from manufacturers are providing chips that can tolerate a range of supply voltages between 2.7V to 5V. Be aware that a card operating at 5V may be compliant with the standards but may not actually work properly with a palm top, for example. Another problem situation is using a 3V chip card with a 5V reader which results in the chip being damaged. Many of today's smart card chips tolerate 3 and 5 volts operation.

4.8 Security Features

The card's security features can be divided into:

- Security measures inside the chip module

  To prevent the chip from being tampered with, the construction employs "passive" and "active" security methods. For example passive measures include removing all access to the address and data bus and placing the ROM in a lower layer in the chip module. Active security measures might

include detecting a low/high clock rate, detecting a change in the supply voltage, and sensors to detect radiation and temperature.

- Application defined

  The application file security features are determined during the card layout design (See Chapter 8, “Card Layout Design” on page 87).

- External human-readable security features

  There may be implementation specific requirements that dictate certain identifiers to establish human-readable security characteristics:

  – Photo lamination
  – Signature strip
  – Holograms
  – Microprinting
  – Lasergravure
  – Embossing
  – Security patterns

  The above techniques are described in 2.2.1, “Human Readable Security Features of Smart Cards” on page 12.

4.9 Manufacturers

The information below is for reference purposes, and lists some of the manufacturers and suppliers in smart cards.

4.9.1 Chip Manufacturers

- Atmel
- Hitachi
- Mikron
- Motorola
- NEC
- Oki
- Philips
- ST Microelectronics (formerly SGS-Thomson)
- Siemens

4.9.2 Card Manufacturers

- Bull CP8
- De La Rue Card Systems
- Gemplus
- G&D (Giesecke & Devrient GmbH)
- IBM (through business partners)
- Oberthur
- ODS
- Orga
- Schlumberger
- Toshiba

4.9.3 Card Operating System Suppliers

- Bull CP8
- De La Rue Card Systems
- Gemplus
- G&D
- IBM
- Maosco (Multos)
- Oberthur
- Orga
- Schlumberger
- Siemens
- Toshiba

4.10 Card Reliability and Life Expectancy

Many of today′s magnetic stripe cards have a replacement interval of around 2-3 years, due to the wear and tear of the stripe. Smart Cards have a better life expectancy but inevitably the number of insertions can take its toll on the gold-plated chip card contacts, leading to scratches and dirt in the chip contact area. The scratches can lead to corrosion and oxidization, which will gradually cause the chip to fail over time. The chip manufacturers generally specify a figure of 10,000 insertion cycles, but optimal combinations of card and reader are able to handle 1,000,000 insertions.

The selection of the reader therefore plays a very important part on the life span of the smart card. The two types of mechanisms, landing and sliding chip contact heads are described in 5.2, “Smart Card Reader Features” on page 61.

The other factor that affects life expectancy is within the chip itself: the number of EEPROM Write/Erase cycles. Most chips manufactured today are guaranteed to last at least between 10,000 to 100,000 cycles, but on average the EEPROM may survive up to 2-4 times much longer. The EEPROM data′s life span is dependent on the technology used, but manufacturers usually guarantee its data retention for a 10 year limit.

Within these limitations, the factors that affect the card life cycle will depend on the type of plastic used for the card and the environmental conditions it′s subjected to. If the card has a magnetic stripe on the back, the cards will usually need to follow the magnetic card replacement cycle rather than chip-only smart card cycles.

4.10.1 Card Usage

When considering card selection, consideration must be given to its place of use. As discussed earlier, a contactless card can be used where speed is of the essence and the transaction is very small.

If the chip is in contact with a reader in the power-on state for a great deal of time, then as in any electronic equipment the chip heats up and it is important to make sure that it is well within the operating limits. As an example, a PCMCIA network card might generate enough heat to warm up both the adjacent PCMCIA card reader and the inserted smart card chip.

4.11 Card Material

There are several types of plastic used for smart cards. Each plastic has particular characteristics that needs to be taken into account when selecting a smart card.

The most widely available material for smart cards is poly vinyl chloride (PVC), which is used in the credit card industry. The following table summarizes the different card types in use today.

Table 3. Comparison of Card Material

Material	**PC**	**PVC**	**ABS**
Name	Polycarbonate	Poly vinyl chloride	Acrylonitrile butadiene styrol
Average Life Expectancy	10+ years	2-5 years dependent on usage	Normal aging up to 4 years. Maintains its quality.
Environmental Hazards	None	Chlorine compound is present. Controlled disposal recommended.	No risks are known, but the compound Benzene is a carcinogen.
Temperature range	-40 to +120°C	-5 to + 65°C	-25C to +85°C
Usage	ID cards	Credit cards	Mobile phones
Mechanical Properties	Susceptible to scratches. but, high resistance to mechanical chemical and thermal stress	Poor resistance to temperature	Limited acceptance of color
Personalization Considerations	Laser engraving, but not suited for thermal transfer printing	Suitable for thermal transfer printing; laser engraving is possible but quality is dependent upon material specification	Limited thermal transfer and embossing; laser, possible with special foil; surface greyish (not white)
Cost	High	Most economical	2 x PVC

For a cost-effective solution, PVC is normally the popular choice for smart cards.

4.12 Card Quantity and Cost

The unit cost of a smart card depends on several factors. These include:

- Chip function
- Storage capacity
- Card quantity
- Supplier terms and conditions
- Chip development cost over time (chips with a newly introduced card operating system are likely to cost more)
- Card surface printing
- Type of plastic used

Normally card prices are quoted for the plain white plastic. This excludes other cards costs such as personalization and graphic design. The cost of the smart card is usually negligible in comparison with the investment required to upgrade and modify the infrastructure. For example, introducing smart cards in a retail payment solution requires bringing together many changes to banking and retail services, such as payment processing, ATM services and retailers′ systems.

4.12.1 Reducing Card Cost

Smart cards need not necessarily mean an expensive solution. Some options to reduce card cost are:

- Cost sharing

  The provision of multiple applications would enable each organization having an application in the card to share the cost.

- Selling advertising space

  Another possibility is to sell advertising space on the card for affiliated service providers who wish to have their name associated with the successful launch of the new ″smart″ technology.

- Volume purchase

  The process of manufacturing smart cards with its initializing and personalizing steps involve a significant amount of setup work on the part of the card supplier. Increasing the volume, usually in the order of several thousand or more, can significantly reduce the cost per card.

- Combining card types

  Combining magnetic stripe, chip and contactless technology is possible at very little incremental cost (US$1-2 extra). This provides a cost-effective migration path for future enhancements.

- Delivery time

  Specifying the delivery time can also have a bearing on the cost of cards. A shorter delivery time can mean increased cost. For example, if an order for, say, 80,000 cards is given at the outset of the project, the supplier carries a heavy risk in terms of not being able to procure the components in time and may increase the price. However, if a small number (say a few thousand) is ordered for a pilot phase, the supplier can be confident of procuring the necessary components in time for the subsequent order, with lower risk and hence can reduce the cost of smart cards.

- Chip price fluctuations

  Since semiconductors costs can drop dramatically during a smart card project rollout time frame, flexible contracts with chip suppliers and optimized inventory can significantly reduce the overall price of the card, mainly if the project implies a large number of cards over a long term period.

Chapter 5. Card Reader Selection Process

The basic function of a smart card reader device is to provide power and a timing signal (clock) to the smart card and establish a communication link between the card and the application. Smart card readers are referred to by many names: Card Terminal, Smart Card Accepting Device, Interface Device (IFD), Chip-card Accepting Device (CAD), Chip-Card Reader (CCR) or Smart Card Adapter.

There is an ever-increasing variety of card readers from specialized manufacturers and vendors, but they can be broadly grouped by key application areas and functional capability. The reader selection is mostly governed by factors such as how the user interacts with the device and where the reader is used (Card Acceptance Points).

In Chapter 7, “Analysis of the Case Study Solution” on page 75 we have used these criteria to select the type of readers and features required for the case study.

5.1 Smart Card Reader Types and Use

Smart card readers range from very low-cost simple contact-type devices to sophisticated, programmable, high-security devices with cryptographic keys. With smart cards finding applications in almost every segment of the market, the readers have to follow closely behind and address the different needs of each segment. The readers generally fall into these categories:

- Low-cost readers
- Balance readers
- PC attached/integrated readers
- Stand-alone general-purpose readers
- Electronic purse readers
- EFTPOS readers
- Building blocks
- Hybrid readers
- Contactless readers

To illustrate the different types of readers available, we will show a small selection taken from vendor brochures and the Internet. These readers may be available as separate components or only as part of a comprehensive smart card solution from the vendor.

5.1.1 Low-Cost Readers

Figure 11. GCR410 Reader from Gemplus

These cost about US$ 100 or less and have a simple serial cable to connect to the PC. The function is basic and usually the readers do not have a keyboard (to enter a PIN for example) or a display. The reader is dependent on the PC application program for handling all communications with the card.

Typical Use: Home banking, home Internet shopping.

5.1.2 Balance Readers

Figure 12. Pocket Balance Reader

Very low-cost readers (priced in the region of US$ 10 or less), called balance readers come pre-programmed to read the amount from an electronic purse or phone card. Additional functions may include displaying the last transactions

made for ″credit″, ″debit″ and ″load″. They are usually sold as a key ring attachment.

Note: IBM does not manufacture these readers.

Typical Use: Electronic Purse

5.1.3 PC Attached/Integrated Readers

Figure 13. Readers from Gemplus and Verifone (Mobile, PCMCIA, PDA)

These readers are compact and designed to be attached to a portable or desktop PC. There are several possible options for connecting to the PC. They draw power from a built-in battery, serial port or an external power adapter.

For laptops and palmtops, readers are available that fit into the PCMCIA slot. For desktop PCs, keyboards can be bought separately with a built-in card slot. Other options include readers that fit inside the floppy bay or can even be inserted into a normal 3.5″ diskette drive.

Typical Use: Access control, home use card personalization, testing.

5.1.4 Stand-Alone General-Purpose Readers

Figure 14. Gemplus GCR500

These card terminals operate offline and have the applications and security modules loaded into the reader′s programmable memory. They may have dial-in access to a host system for updating a list of hot-listed cards or downloading software upgrades.

Typical Use: Access control, health.

5.1.5 Electronic Purse Readers

Figure 15. Electronic Purse from Verifone

Small and portable, the electronic purse is designed to support various forms of cashless payment, including card-to-card transfer. It can function as an offline terminal, automatically transferring the value from a customer′s smart card to the retailer′s smart card stored within the terminal′s Security Application Module (SAM). For example, diners at a restaurant can pay their bill without giving up possession of their cards. Other functions usually include a balance display, lock/unlock card facility and displaying the last 5-10 transactions.

Typical Use: Retail, restaurants, airports, taxis, public transport, newspaper stalls.

5.1.5.1 Cash Loading Devices

Figure 16. Cash Loading Device

This device is used to load the electronic purse of a smart card. It can accept bills and coins. The display is used to show the current balance and the amount loaded. It can also deplete the purse and give the cash to the user.

Typical Use: Banks, cafeterias of big corporations, etc.

5.1.6 EFTPOS (Electronic Fund Transfer and Point of Sale) Readers

These are specialized terminals that enable secure transfers of electronic funds to enable the payment of goods or services from a smart card. The terminals may be equipped to authorize payment via host dial-up or wireless communication using GSM SMS network.

Typical Use: Retail.

5.1.7 Building Blocks

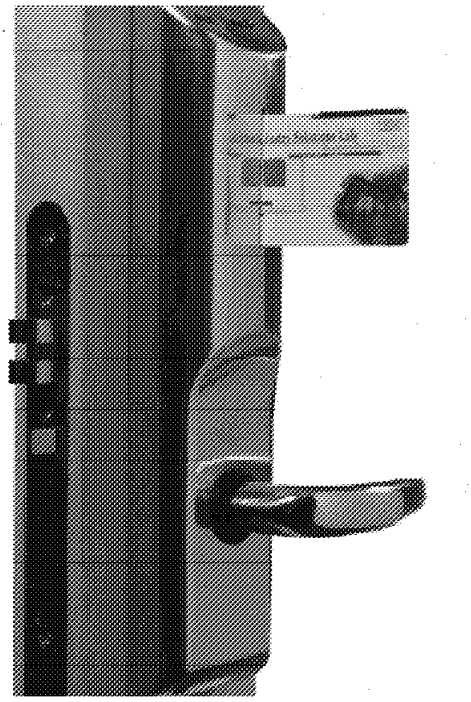


Figure 17. Door Lock Reader

These readers contain the bare-bone mechanical and electronic components for integration into various products marketed as part of a medium-to-large implementation.

Typical Use: ATMs, kiosks, vending machines, door locks, pay phones.

5.1.8 Hybrid Readers

In addition to reading the chip, these readers have the capability to read magnetic stripe cards or optical cards. The Kyoto reader from OMRON, which is used in kiosks, is an example of this.

Typical Use: ATMs, kiosks, health.

5.1.9 Contactless Readers

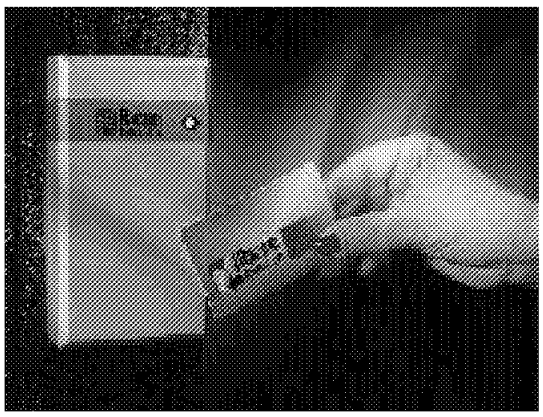


Figure 18. Contactless Reader from RACOM

Contactless readers are used mainly for applications requiring high throughput such as fare payments in public transport. The reader allows the card to be operated from a distance of 1mm up to several centimeters. Due to the closely integrated technology development with a particular contactless card and lack of standards, these readers may only function with a manufacturer's particular card.

Typical Use: airline boarding, automated fare collection, building access.

5.2 Smart Card Reader Features

We can now take a closer look at some of the features mentioned in the previous section. All of the features may not appear in any one card terminal and we have made up a generic reader for our purpose.

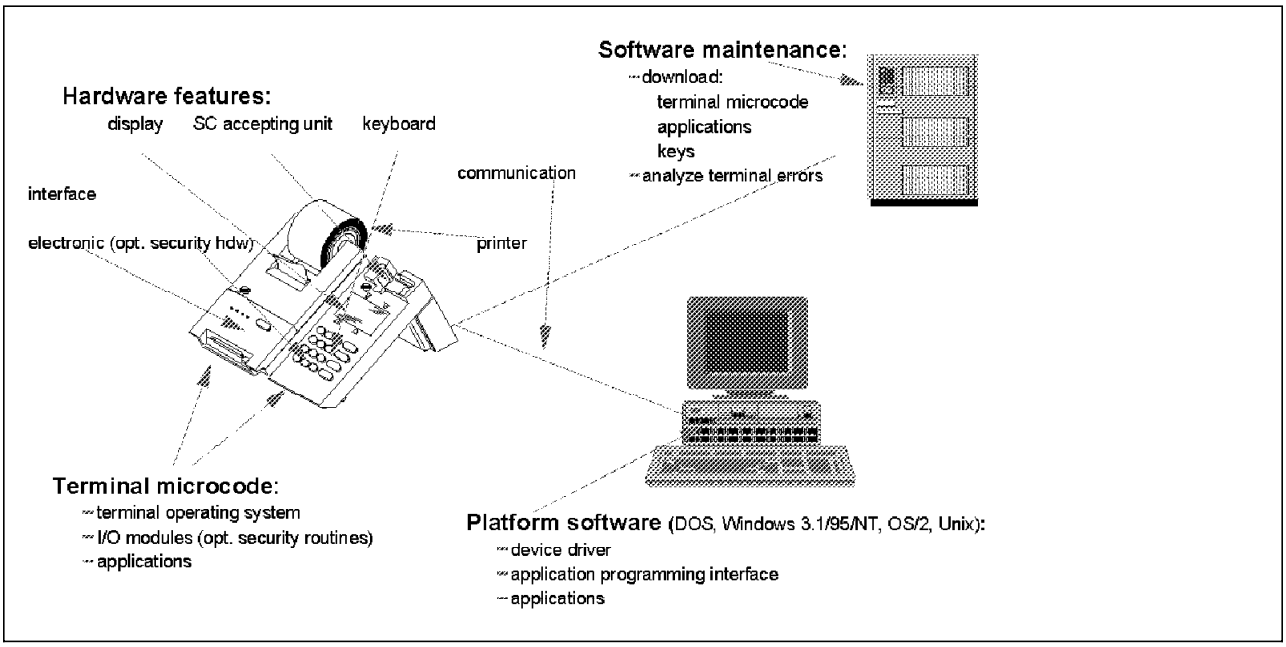


Figure 19. Smart Card Reader Features

5.2.1 Hardware Features

5.2.1.1 User Interface

- **Smart Card Accepting Unit**

  When a card is inserted into a reader, power must be applied to the chip contacts and a timing signal (clock) established for the card to be operational. Although most smart cards today conform to the location of the contact area according to the ISO standard, the cards from the earlier French trials conform to the French national standard (AFNOR) and have a slightly different position. As a result, some card readers have two contact modules in the slot.

  Contact Methods

  The accepting unit can have one of two methods to establish contact with the chip:

Sliding Contact The cheaper readers have simple sliding contacts that tend to leave scratch marks on the gold contact area. This affects the contact area after some use and the card may need several insertions and removals in order to get a good connection.

Landing Contact The landing method makes a more gentler contact by means of pins that land on the surface when the card is inserted. When the card is removed, the pins release the pressure so that there is very little wear and tear on the chip contacts.

These landing contact readers are more expensive than the simple devices that use the push-pull method. More advanced high-performance readers used in ATMs for example have motorized electro-mechanical units with card locking, ejection and retaining features.

- **Display**

  Smaller readers may have an LCD panel capable of displaying characters on a few lines. Larger displays may be equipped to display graphics.

- **Keyboard (PIN pad)**

  The reader may have a very simple numeric keypad or a secure keypad to enter the PIN. A PIN pad is usually sealed together with a security module to encrypt the key strokes and verify the PIN directly with the smart card. This avoids a breach in security by not exposing the PIN to the outside world. The security module is also able to detect any attacks. Additional programmable function keys may be provided depending on the reader.

5.2.1.2 Communication

- **Host Attachment**

  For PC-connected readers, communication is provided using the standard serial port operating at 9600 baud. Higher and lower speeds may be possible depending on the reader configuration settings and technical capability.

- **Printer Attachment**

  Stand-alone readers (for example a reader used at a health clinic) may have a printer port with the capability to print forms with details read from the card already filled in.

- **Host Dial-up Facility**

  Card terminals used in the payment environment, for example where electronic cash is used, may use a modem to dial-up the host and transfer the electronic funds to the retailer′s account by means of secure protocols.

5.2.1.3 Compliance

- **Certification by Banks and Other Organizations**

  An example of this is a smart card hotel application that uses a kiosk in the lobby. When a new component such as the smart card reader is installed, approval from a Fire Regulation Authority may be a requirement to certify that the kiosk meets the standards for fire safety. Where a smart card is used for payment, the reader devices would require certification by the bank concerned.

- **PC/SC Compatibility**

If the reader is interfaced to the PC platform, the reader may have PC/SC compliance. The reader manufacturer has to supply the appropriate device drivers to make this feature available.

5.2.1.4 Security

- **Card Retaining Feature**

  In a similar method used with magnetic stripe cards, the readers (for example those used in ATM machines that are motorized) have a feature to retain a smart card.

- **Physical Security Methods**

  Very secure card terminals would have a shutter to lock the card in during a transaction or clip any wires being used to read signals from the chip contacts while in use.

5.2.2 Terminal Microcode

- **Terminal Operating System**

  All the terminals have a small operating system inside written in the terminal microcode to handle different tasks such as polling, processing commands and interrupts, and controlling the keyboard, communication and display.

- **I/O Modules**

  Servicing the terminal′s display, electro-mechanical attachments, serial and modem communication is done by dedicated I/O modules under the control of the operating system.

- **Secure Application Module (SAM)**

  The terminals must also be able to handle the many different retailers′ and card issuers′ cryptographic keys, as well as to encrypt and log the communications with the outside. This is accomplished by a SAM module securely located inside the card terminal. The secure keys required for transactions may be stored in the SAM module or the keys may be stored in a smart card chip in the ID-000 card format (see Figure 7 on page 31) and is slid into contact with the SAM module. A terminal can have more than one SAM module (up to 6 in some readers), in order to service different card-issuing authorities. In some instances several of these authorities may jointly issue a smart card, which is then located inside the SAM.

- **Applications**

  The terminal can have applications loaded, for example to take the payment from a smart card and log it for later sending to a bank account. The terminal application code is usually downloaded using a secure host connection.

5.2.3 Software Maintenance

- **Facility for Software Upgrade/Download**

  Software maintenance/upgrade for readers may be provided using a dial-up host link.

- **Terminal error log**

  The terminal (for example, a kiosk) may have a log of transactions and any errors, which is uploaded to the host for analysis and diagnostics.

5.2.4 Platform Software

- **Device Driver**

  The card readers are usually provided with the drivers in the form of DLLs (dynamic link libraries). These work on a specific workstation platform.

- **Application programming Interface**

  As each reader requires its own drivers, the problem of handling these drivers is delegated to a software layer and the application uses a standard set of APIs as defined by PC/SC (for Windows 95/NT) or OCF (for Java, NC, UNIX and Windows 95/NT).

5.2.5 Other Features

- **Capacity to Hot List Cards**

  The host dial-up link can provide a downloading facility for updating a card hot list at the reader. These are primarily used at ATMs.

- **Capability to handle other card types**

  Readers can handle both magnetic stripe cards and smart cards or even cater for both the French (AFNOR) standard and ISO standard chip contact positioning.

- **Power Supply**

  PC attached smart card readers draw their power from various sources: a built-in battery, the RS-232 serial cable or even from the keyboard/mouse connection. In the case of battery-powered readers, the battery capacity and replacement would need to be considered as part of the overall maintenance plan. Sometimes a reader may work well with a desktop PC, but fail to work with some cards on a laptop. The reason for this is likely to be the current consumption of the card which the laptop is unable to meet. Smart cards with a coprocessor may also make additional demands on the current supply and this will affect the choice of reader.

5.3 Card Reader Suppliers

In selecting any product, the established reputation of vendors is one of the first considerations. Today there are several hundred smart card reader suppliers and readers. The Internet provides a valuable source of information on card reader suppliers and their product portfolios. Details of low-cost readers can also be found in specialist electronic store catalogs.

The following is a list of some of the major manufacturers and some perhaps less familiar names in the card terminal reader supplier market.

Table 4 (Page 1 of 2). Card Terminal Manufacturers

ALPS www.simpletechnology.com	American Magnetic www.amagnetic.com
Ascom www.ascom.com	Bull www.cp8.bull.net
Cherry www.cherrycorp.com	Dassault AT www.dassaultat.com
De La Rue www.delarue.com	Dione

Table 4 (Page 2 of 2). Card Terminal Manufacturers	
G&D www.gdm.de	Gemplus www.gemplus.com
Fischer International www.fisc.com	Hewlett-Packard www.hp.com
Ingenico www.ingenico.fr	Intellect www.intellect.com.au
Innovatron www.innovatron.com	Litronic Inc. www.litronic.com
Omron Electronic www.industry.net/c/mn/0107	Orga www.orga.com
Racom Systems, Inc. www.racom.com	Schlumberger www.slb.com
SCM Microsystems www.scmmicro.com	Siemens Nixdorf www.sni.com
Toshiba www.toshiba.co.jp	Tritheim www.tritheim.com
Utimaco www.utimaco.com	Verifone www.verifone.com

5.4 Card Reader Standards

With worldwide interoperability very much in focus, not surprisingly many organizations have an influence on the card terminal market, including:

- Standards Organizations (ISO, ETSI)
- Credit card organizations (EMV, AMEX)
- National / international payment systems (for example, MONDEX, ZKA, Carte Bancaire)
- Governments (Healthcare, Identification)
- Industry partnerships (PC/SC Workgroup, AMEX/IBM/American Airlines)
- Customers

5.4.1 Reader Standards, Compatibility Issues

With the many standards in place to guide the smart card technology, the aim is to enable any card to work with any reader. But in practice factors such as an existing infrastructure, technology improvements or influence of another standard may give rise to compatibility issues and these should be considered when selecting card reader devices.

Some of these considerations are described below:

T=0, T=1 Cards

Card terminals require the same protocols to communicate as the card, namely T=0 or T=1, described in Chapter 4, “Card Selection Process” on page 41. To ensure operation with both types of cards, the card terminal should be capable of automatically switching its communication protocol after receiving the cards first power-on response.

Voltage Supply

With the trend in the electronic component industry to reduce the operating voltage of microprocessors, smart card will also be

affected, and the card's operating voltage should be matched with the specification of the card terminal.

PC/SC Compliance

This is an evolving PC Windows software API specification to make life easier for both the application developer and the vendor. It is backed by major names in the smart card industry. The PC/SC compliance logo reader test is described at the end of this section. When selecting a reader, the question pertaining to the reader's PC/SC compliance is bound to be raised. The answer naturally depends on whether the target environment is Windows 95/NT platform or not. However, readers which have no PC/SC drivers can still be used and is a decision that has to be made. The Open Card Framework Architecture and the IBM Smart Card ToolKit allow for both PC/SC and non-PC/SC compliant readers.

5.5 Card Terminal Security

Terminals have to employ a means of protecting its external housing and internal security modules from attack. The level of sophistication depends very much on the area of smart card application, and the reader specification should be assessed against the customer requirements and potential risk.

5.6 Card Acceptance Points

A good starting point to determine the reader functional requirements is to look at it from the user interface point of view, that is, where the reader is going to be placed. The list below shows some of the wide range of card acceptance points that come up in smart card projects.

- Self-Service
  - cash
    ATM
  - non-cash
    information kiosks
    ticketing
    statement printer
    loyalty schemes
- Point-of-Sale
  - check-out
  - restaurants
  - home delivery service
  - vending machines
  - taxi, public transport
  - parking
- Identification
  - police
  - immigration, customs
  - driver license
- Access Control
  - building
  - computer
    terminal
    software
    data
- Attendance recording
- Home Use
  - home banking
  - electronic commerce
  - software copy protection
- Personal use
  - person to person transactions
  - balance reader
- Card system support
  - card personalization
  - maintenance
    (add/delete applications)
  - diagnostics
- Specialized applications
  - road toll

5.7 Smart Card Reader Evaluation

Although the PC/SC Workgroup standard provide the blueprint for achieving reader and PC compatibility, a testing procedure is required to pass this compliance test. It is possible to enlist the services of companies (such as IBM) to conduct a comprehensive evaluation, test and quality-assurance process to ensure trouble-free operation in the target project.

5.7.1 Microsoft Windows-compatible logo

The Microsoft Windows-compatible logo is a program developed by Microsoft in conjunction with smart card reader vendors to ensure Windows and card reader hardware compliance. Several leading smart card manufacturers have provided test suites and cards to carry out this test. For example, the IBM MFC 4.1 card is one of the cards used in carrying out this compliance test.

Details of this program are available from Microsoft:

- www.microsoft.com/smartcard/
- www.microsoft.com/hwtest/
- www.microsoft.com/hwtest/hcl/

Chapter 6. Case Study

In this chapter, we will present an imaginary customer who has issued a request for proposal (RFP) for a project involving smart card technology. First, we will profile the customer so that the reader can understand the customer's business, the size of the company and so on. Then, we will describe highlights of the RFP. This will help us later, in Chapter 7, "Analysis of the Case Study Solution" on page 75, to answer the RFP. When we answer the RFP, we will detail, step by step, the reasons for selecting a particular smart card component or supporting software as well as reasons for not selecting another component in its place.

6.1 A Profile of GTT

Our imaginary company is called "GTT (Global Talk 'n' Tell) International Corp.".

GTT is a medium size communications hardware and software manufacturer. They produce consumer communications equipment such as GSM telephones for Europe and the Far East, and cellular telephones for use in North America. They have recently branched out into the pervasive computing market. GTT can offer products like personal digital assistants for the consumer market and they supply components and subassemblies to other manufacturers of Network Computing (NC) workstations and consumer home electronics.

There are about 10,000 employees world wide who work for GTT. They have their headquarters and development laboratory in San Jose, USA. Their Far East operations are run out of the regional sales office in Kuala Lumpur, Malaysia and the European operations are headquartered in Stuttgart, Germany. All manufacturing and distribution is from two factories in Malaysia. The number of employees is roughly 3K in USA, 2K in Europe and 5K in Malaysia.

GTT is a modern, high technology company. All managers and most exempt employees have personal workstations based on the Windows platform and on Intel computers; however some locations in Europe are converting over to NC machines. The laboratories are primarily equipped with UNIX workstations. GTT Corporation headquarters communicates to the regional offices and laboratory over the Internet. E-mail is used for correspondence between the headquarters, development lab and the factories, and also with customers and vendors. The network is TCP/IP based.

GTT is aware of the importance of security. Today, all GTT employees wear an identification badge that has the GTT logo, a photograph of the employee, the employee's name, their identification number and the division in which they work. The photo id badge has a magnetic stripe on the back for facilities access through electronically monitored doors. These magnetic stripe badges also allow the employee to purchase subsidized food at the company cafeteria. All GTT employees are required to wear their badges while on GTT premises.

6.1.1 Security

GTT has recently raised the bar in the area of security. A rumor has been circulating among industry observers that hackers have broken into GTT′s network. GTT vehemently denied the rumor, with such an emphasis that confirmed to the industry observers and the specialized press that the incident indeed did happen.

GTT has beefed up its network security: more restrictive rules in the firewalls, creation of the demilitarized zone with a second firewall in each location, virtual private networks (VPN) between firewalls, implementing one-time passwords technology to log on to the intranet from the Internet integrated into the firewall, etc. In summary: network security is a big issue at GTT and expenses related to this subject are quickly approved.

Still GTT is not satisfied with the actual level of overall security. GTT has done some research on how to improve security and in many cases it has come across that smart cards are used for one reason or another in security projects. For this reason, GTT has decided to implement a solution that includes the use of smart cards. GTT will replace the existing employee magnetic stripe card with a smart card.

GTT has some experience with smart cards. They are used in the GSM phones. As explained in Chapter 3, “Standards, Specifications and Recommendations” on page 27, GSM is a standard used for a very specific purpose: communications.

One of the applications GTT would like to implement is the verification of the employee using some kind of biometric identification. In some cases, the mere fact that the employee is carrying the card is not enough security. The cardholder should prove that he/she is the owner of the card.

There are several human body characteristics used for biometrical identification: fingerprint, eye retina, hand configuration, etc.

GTT has learned that in Peru the government is identifying millions of citizens using fingerprints. GTT came to the conclusion that if this method is good for millions of people, it must be good enough for a few hundred people (just a fraction of GTT′s employees will need to use this form of identification, mainly for accessing secured areas).

In addition to the difference in number of people, the application in Peru is much more complex since the system has to find the scanned fingerprint in a database that contains millions of others, while in GTT′s case the system has to compare the scanned fingerprint against the digitized version stored in the smart card.

The digitized version of a fingerprint does not take more than a few hundred bytes, and that makes it feasible to store in the limited EEPROM of a smart card. When the employee needs to enter a secured area, he/she will insert the smart card in the reader, place his/her finger in the scanning area of the card terminal, and an application will compare both digitized fingerprints. If they match, the system will then look if that card is allowed to enter the secured area.

GTT also wants to impress some of its visitors; for that purpose, a system will be installed to quickly produce a smart card containing the fingerprint of the visitor.

6.1.2 Digital Signature

While investigating the smart cards, GTT came across another application that would benefit their business: digital signature.

Due to the fast pace of GTT′s business, a delay in signing a contract with a supplier or a vendor can be costly. Several contracts have been lost for this reason.

The idea of writing a contract using a word processor such as Lotus WordPro or Microsoft Word, digitally sign it, send it over the Internet (probably encrypted) to the other party, who will also digitally sign it and return it to GTT also over the Internet, is really attractive to GTT′s executives.

With more countries and USA states declaring the digital signature as legally binding as a traditional hand-written signature, the use of this process has potential to grow rapidly.

GTT knows that the most important aspect of this process is to preserve the secrecy of the private key. GTT believes that a perfect place to store this information could be a smart card. The digital certificate could also be stored in the smart card, making it very convenient: the executive signing the document can carry the smart card with him and sign the document almost anywhere.

But GTT has its feet firmly on the ground; it knows that many aspects of this process has to be worked out with lawyers and partners before it can be implemented. GTT will start using digital signature in e-mail, first between GTT′s employees and later extending it to business partners who are able to support it.

GTT will use the employee smart card to store the private key and the certificate.

GTT is undecided on whether to implement its own Certificate Authority to handle the certificates and public keys, or to contract with an external CA. GTT is asking for advice in the RFP.

GTT is using Netscape Communicator for Web browsing, e-mail and collaboration. The proposed solution should satisfy this software base and adhere to applicable standards, for example PKCS#11.

6.1.3 Network Access

There are three areas in which GTT would like to use the smart cards:

1. Network computers (NCs) do not have a local hard disk; therefore, all programs and data have to be loaded from the server in the network. GTT would like to use the smart cards in order to identify the employee trying to download software from the server, using the certificate provided for digital signature.

   To set up the NC for an individual user, a user-specific smart card could be inserted into a smart card reader integrated in the NC. After the user has enabled the smart card by entering the secret password, the NC would read from the card the network address of the user′s server and would start the communication to identify the user by generating a signature for a server-generated random message using the secret private key. After successful verification by the server, the user gets access to his/her application profiles, ideally in the same form as they were at last logoff. When using further services in the net the user will identify himself/herself

with the smart card. The network costs can be billed with the help of the identification furnished via the chip card.

2. GTT would like to ensure that only the right employee access certain workstations. One way to ensure this is for the PC to only accept input from the keyboard or mouse when the right smart card is inserted in the smart card reader attached to the PC.
3. Single sign on: GTT would like to use smart cards in a single signon system. The employee would use the smart card to authenticate himself/herself to the single signon system; once this is done, he/she could access any other system for which he/she has a user ID.

6.2 Summary of Requirements

GTT wishes to begin a pilot where they will introduce a smart card to replace the photo ID badge. To this end, the smart card should immediately duplicate the functionality of the present system:

- Picture identification
- Facilities access
- Cafeteria services

In addition to these features, GTT wishes to have additional functionality not available in the present system:

- Digital signature
- Electronic purse for cafeteria and vending machines
- Biometric access to restricted areas
- Network access
- Health alerts (blood type, allergies, doctor′s name and phone, medication, etc)
- Travel applications such as hotel and airline check-in and car hire

Also, with the anticipated smart card infrastructure being put in place, GTT is looking for additional application areas that haven′t already been considered yet.

6.3 Requirements

The following lists GTT′s mandatory and optional requirements as they appeared in the RFP.

6.3.1 Mandatory Requirements

- Support for existing applications

  GTT needs to support the existing human-readable security features like employee number and photo identification. For migration purposes, the proposed smart card should also have a magnetic stripe and the proposed smart card readers should also accept the old magnetic stripe cards.

- Support for secured facility access

  The proposed smart card should contain the printed name and serial number of the employee, their picture, and a GTT logo that will vary depending on the employee′s division, location and security access granted to that employee.

- Electronic purse for the company cafeteria

GTT employees use their magnetic stripe cards to pay for their meals in the company's cafeterias. GTT would like to use a purse in the employee's smart card to do this payment. GTT cafeterias use different currencies depending on their locations; the proposed solution should consider this situation.

- Support for digital signatures

  Employees using NC-based or PC-based computers will use Netscape Communicator for e-mail. GTT will adopt the S/MIME standard and will want to use smart cards to store the private key and certificate to be used to sign outgoing e-mail. At the beginning of the project, only employees handling sensitive data will be provided with smart card readers for their PCs. As the project rolls out company wide, all employees will get a PC-attached smart card reader.

  In a first stage, the digital signature will be used only for e-mail, but later it will be used to authenticate any type of document.

- Storage of employee health information on the smart card

  GTT would like to store employee's personal health data in the smart card. As this is a sensitive subject, it will be optional for the employee whether to store this data or not. The data stored in the smart card would help in finding the best care for the employee in case of an emergency should he/she be unable to respond.

  The doctor's office will have a card terminal for reading the health alert information. Employees will be able to update their own information on the smart card. Those employees without a smart card reader will be able to update their smart cards at the Human Resources office.

- Support for time and attendance in the smart card backend system

  Direct employees in the manufacturing plants use a magnetic stripe application for time and attendance today. Once the smart card system is in place, GTT would like the plant workers to start using the smart cards for this purpose.

- Support for biometric identification

  For highly secured areas, most notably the research laboratory, the employee will have to prove that he/she is the owner of the smart card. This can be accomplished by using a Personal Identification Number (PIN) but GTT wants to use the employee's fingerprint for this identification.

  Special smart card readers will be located at highly restricted areas to scan the fingerprint of the employee and compare it to the digitized fingerprint stored in the smart card.

The proposed solution must meet applicable ISO standards and smart card specifications.

GTT has specified that the smart cards must be ISO 7816 compliant and that they should support the EMV specification. The reason for EMV support is primarily for the multiple application selection mechanism but they realize that having compliance will give them a way to implement interoperative stored value applications.

For smart card readers that will be attached to PCs, they must meet PC/SC interface standards. GTT plans to move most of their employees from the

Windows/Intel platform to network stations. GTT wants to use the proposed smart card in these network stations.

6.3.2 Optional Requirements

The RFP also listed optional requirements, which address GTT's future needs. These optional requirements are:

- Network access
  - Support for Network Computers

    As explained earlier (see 6.1.3, "Network Access" on page 71), GTT plans to use the smart card to control the access to servers from network computers. GTT is asking for advice on how to proceed in this area.
  - PC will only accept inputs when the smart card is inserted in the smart card reader
  - Single signon
- Travel

  GTT has learned that smart cards are used in the travel industry for:
  - Loyalty
  - Hotel preferences
  - Car rental preferences
  - Frequent flyer programs data

  As some employees travel quite a bit, GTT wants to explore the possibility of using the smart card in this area.

Chapter 7. Analysis of the Case Study Solution

In this chapter, we will follow through with the analysis of the RFP for our case study that was presented in Chapter 6, “Case Study” on page 69. We detail the reasons for selecting a particular smart card component or supporting software and reasons for not selecting another component in its place. This analysis is very detailed, much more so than one would do for a typical response to an RFP. This detail is for the benefit of the reader and is not meant to imply that this level of response is necessary on all smart card RFPs.

The purpose of this chapter is to help the reader identify similar customer situations and be in a position to intelligently select smart card components to fulfill the customer’s requirements.

7.1 Card Selection

We will first focus on what type of card we will select. There are a couple of aspects in GTT requirements and applications that greatly facilitates the card selection:

1. Digital signature: in order to implement this application, the card will need to have a crypto processor. This requirement greatly reduces the number of cards that we can choose from. This requirement also implies that we will need a contact card. The contacts will also be needed by most of the other applications.
2. Facility access: GTT requires a quick pass through the gates for its manufacturing plants. The card will have to be contactless also. See 7.4, “Facilities Access” on page 77 for details.
3. In order to comply with ISO 7816-1, the card size will have to match the size of the common credit cards. The actual GTTs badges are slightly larger than the credit cards. We will have to keep this fact in mind when selecting readers during the migration and pilot phases.
4. Card material: since the employee photos must be thermally transferred to the card, Table 3 on page 52 dictates that the card material must be poly vinyl chloride (PVC). The polycarbonate and ABS materials cannot accept thermal transfer printing; therefore they cannot be used in this application. ABS introduces limitations in printing quality.
5. EEPROM size: we will have to add up the memory requirements of each application to come up with the minimum EEPROM size required. This subject is discussed in detail in 8.7.1, “Sample Memory Requirements” on page 107. From this we come to the conclusion that we will need a minimum of 8 KB.
6. Operating system: the operating system must support the cryptographic processor and multiple applications. As this is going to be a closed system, with GTT in total control of it, GTT does not want to pay extra fees to use an operating system that would allow the card to contain purses that will be accepted by merchants all over the world.

   Although the press and industry analysts agree that interpretative smart cards, like JavaCard and Multos, are the cards of the future, GTT recognizes that this is an emerging standard and they cannot wait until the dust settles and does not plan to play a pioneering role. GTT will not use a JavaCard or

a Multos card;[3] its requirements can be addressed with existing, conventional card operating systems.

GTT also wants an extensively proven operating system, with many implementations already running.

7. GTT has explicitly requested that the card should be EMV compliant. Although GTT does not plan to implement any EMV payment scheme, GTT wants to use the EMV application selection structure.
8. GTT does not have any specific preference on chip manufacturer.
9. The card vendor should have an established presence in all the countries where GTT is located.
10. The operating system must have a powerful tool kit to help in the application development.

There are several cards on the market that comply with these requirements. We selected the IBM MFC 4.21 card.

This card has an 8-KB EEPROM, crypto coprocessor, 512 bytes of RAM, using the chip CF68 from ST Microelectronics, formerly SGS Thomson. The contactless chip has 1024 bytes, and it is not connected to the contact EEPROM.

This card was selected based on the size of the EEPROM (we will see later that this is enough to implement GTT's applications), the asymmetric encryption capabilities provided by the crypto coprocessor required for digital signature and the proven functionality of the MFC operating system.

This card is ISO 7816 compliant, but so are almost all the other cards on the market.

Associated with this card is the powerful IBM Smart Card ToolKit, although this tool can also be used with other smart cards. We will use this tool later in the book to show the complete GTT card layout, plus the use of agents to access the card.

Nevertheless, you will see that most of what we say later on in the book applies to many cards in the market.

7.2 Reader Selection

The factors needed to be taken into account when selecting a reader have been described in Chapter 5, "Card Reader Selection Process" on page 55. We will list here a couple of factors that we think are common to all readers selected for the different applications:

- The reader vendor will have to be established in all the countries where GTT will install the readers.
- Some of the readers will have to be PC/SC complaint. Although most of the applications will be developed specifically for GTT and could be developed for some specific reader, the digital signature application will be acquired and it requires PC/SC Migration compliant readers.

[3] For more information on JavaCard see Appendix B, "JavaCard Overview" on page 179; for more information on Multos see 199 or browse http://www.multos.com.

GTT will generalize this requirement and apply it to all the readers.

Several type of readers will have to be selected; for the facility access, for digital signature, for the biometric verification, for the contactless application, etc.

We will describe the readers selected as we describe the different applications. See Table 4 on page 64 for a list of some of card terminal vendors.

7.3 Existing Applications

The new smart card will have a magnetic stripe in the back. The content of the stripe will be the same as it is in the existing cards.

There will be a transition period in which the employees will have the new card, but the readers will still be the existing ones.

The details on how to personalize the magnetic stripe in the new cards, as well as the print data and employee's picture is covered in Chapter 11, "Project Management" on page 139 and Chapter 12, "Card Management" on page 145.

7.4 Facilities Access

GTT will use the smart cards for building access. The most practical smart card readers to use to detect the ID cards are contactless. A smart card must be fully inserted into a contact smart card reader to work properly unlike a magnetic stripe card which is passed over the magnetic reading heads. It is difficult for people to insert a smart card into a thin slot quickly so this delay would cause large queues of people to form outside the main building entrances. This is especially relevant at the plant locations because people work shift hours and not flexi-hours. So there are periods in the morning and evening when large groups of people enter and leave the building creating a congestion problem. A contactless smart card reader is more tolerant of positioning errors because the detection zone is large. With these contactless smart card readers installed, employees could pass near the readers (within one meter) and the smart card reader would detect the card and read the card serial number.

A contactless reader will also reduce the wear of the smart card contacts that would otherwise occur if the card is inserted in a contact reader. This should not be a deciding consideration in most cases since contact smart card vendors guarantee the life of the card for a large number of insertions.

A small selection of vendors that manufacture contactless smart card readers are:

- Racom
- CYTEC Casses & Computer Gmbh
- DATASEC Electronic Gmbh
- Deister Electronic
- UniVision
- Gemplus

Since the GTT's old magnetic stripe card did not conform to the physical sizes specified in ISO 7816-1, a slight modification will have to be done to the old magnetic stripe readers. The old cards are a few millimeters wider that the new ones. In order to make the new cards not to wiggle in the reader, In order to

accommodate the new cards with reduced widths and for the magnetic stripe alignment, a small metal piece will be added to the card slot of the reader.

Once this piece is added, the old badges will not fit any longer. For employees who for one reason or another do not have the new card, the old card will have to be chopped in length in order to insert.

Although there are advantages of using a contactless reader, as mentioned above, for cost reasons GTT may decide to use a contact smart card reader in some locations. A contact reader costs less than a contactless reader.

The smart card readers for facilities access must be connected online in order to check the card number against a black list, to deny entrance in certain cases.

In summary, there will be different phases of the implementation of facility access:

- When the new cards are distributed, the employees will use the old readers to access the facilities. The old readers will have to be modified slightly as described above to help the employees to correctly insert the new cards. For one reason or another, some employees will not get the new cards, so they will continue to use the old ones. These old cards will not fit anymore in the old readers; security officers will be on hand to cut the old badges so that they will fit in the modified readers.
- The installation of the contactless readers will start; it could be before or after the distribution of the new cards.
- Some contact readers, like the one shown in Figure 17 on page 60, will also be installed.
- Some of the old magnetic stripe readers will remain installed, to cut costs, depending on maintenance, location, usage, etc.

The pace of installing the new smart card readers will probably be slow since this particular use of the smart cards is not adding any new function to what already is in place.

The access control using biometrics is covered in 7.9, “Biometric Access to Restricted Areas” on page 84.

7.5 Cafeteria Services and Vending Machines

GTT will have smart card readers in all cafeteria locations when this project is rolled out company-wide. The smart card reader should be able to accept a contact ID badge and debit a pre-paid ″wallet″ on the smart card. The smart card reader could be a stand-alone device but most probably it will be a very simple smart card reader attached to a PC because this is the most cost-effective solution. The PC and reader are both inexpensive and development of the debit application on the PC can be done quickly and cheaply. A dedicated smart card reader and stand-alone application program running on the reader would be much more expensive to develop and maintain.

There must be an easily accessed facility for an employee to load the electronic wallet with money.

The terminal will be located in the cafeteria. The employee will insert local currency bills in the slot. See Figure 16 on page 59 for an example of such a terminal.

In locations outside the USA the application will convert the local currency amount into dollars; in the USA the purse will be increased by the amount of dollars inserted. It will use an exchange rate that will be entered into the application periodically. This period will be established by GTT; the employees will be informed when the exchange rate is changed. This period of time will depend of how the exchange rate changes; in periods of economic turmoil, the period will be smaller than in periods of economic bonanza.

Vendors that manufacture this kind of machines are, among many others, Girovend (http://www.girovend.co.uk) and GeWeTe. See Figure 16 on page 59 for an example.

These smart card readers in the cafeterias will also be used to check the balance remaining in the purse.

The prices of the food will be in local currency; when the employee pays for the meal using the card at the cashier′s terminal, the amount will be converted in dollars using the same exchange rate and deducted from the purse.

This system presents some small problems. One of them is that even if the employee does not use the card to pay in the cafeteria or vending machine, the balance in the purse in local currency may change due to a change in the exchange rate. To avoid wrong interpretations, the cafeteria card terminal, when requested to show the balance, it will do it in local currency and also in dollars. In this way, the employee will be able to see that if he/she does not use the purse, the balance in dollars remains the same.

Besides, in times of economic turmoil, when the people are more sensitive to money matters, the value of the purse in local currency will tend to go up, since the value of the dollar in local currency will probably be greater.

Another problem could be if the country has restrictions on buying and selling foreign currency. Someone may argue that an employee in a country outside the USA is able to buy dollars without following the official procedures, since the employee could load the purse in his country, travel to the USA and use the purse there to buy food in dollars.

Finally, an accountant may find holes in this implementation.

But we should put this into perspective:

- This is a close purse; it can only be used in GTT′s cafeterias and vending machines
- The amounts paid are relatively small
- GTT will set a rather low limit on the maximum amount that can be loaded in the purse
- The use of the purse is optional; the employee can still pay in cash
- The food price is subsidized by GTT

GTT will have to educate the employees on this subject, as well as make them aware that losing the card is the same as losing cash. No attempt will be made to reconstruct how much money was left in the purse of the lost card.

This type of purse is actually being implemented in some cases in Europe, in preparation for the conversion to a common currency: the Euro. The value stored in the purse is in Euros, not in the local currency inserted in the machine.

Other situations must also be covered with the purse application:

- For example, for employees who need to change the card (new name, new logo, etc), a system must be in place to transfer the purse from the old card to the new.

  There are several possible solutions:

  – A card terminal like the one in Figure 16 on page 59 can read the content of the purse, set the purse to zero and dispense the equivalent cash.

  – If the solution above is too expensive because of the price of the machine, a smart card reader with two slots to transfer the purse is necessary. Both cards, the old and the new, will be inserted in each slot and the transfer will occur. This system is used by some merchants in Europe to transfer the money from the customer purse to the merchant's smart card.

    An incomplete list of vendors that produce two slots machines is:

    - Cherry Mikroschalter Gmbh
    - Gemplus
    - Keycorp Ltd.
    - Pacific Tech Co. Ltd.
    - Verifone

  – An application could be developed to decrease the purse in one card and increase in the other. This solution has the disadvantage that it could be misused.

- If an employee leaves the company, the remaining purse balance should be returned to the employee. The owner of the purse-loading machine (GTT or the cafeteria vendor) is the one that has the money to be returned.

Another option that GTT could have chosen, but did not, was to define several purses on the smart card, one for each country. Each purse could hold a local currency value depending on the location in which the employee found themselves. The backend systems would not need to perform currency conversions because all cafeteria purchases would be in the local currency.

There were several reasons GTT chose not to implement several purses:

- With a multiple purse setup, the employee may end up with unused currency remaining in a purse that cannot be used or converted. It might be possible to add a feature to transfer and convert value from one purse to another within the application program. This adds complexity to the application program, however.
- A single purse scheme is more flexible should GTT expand into other geographic locations. It would not be necessary to define a new purse in that case.

7.5.1 Security

We would like to make a note about the security of the purse in the case study. We have to distinguish between a simple purse such as the one used in the case study and the purse schemes described in Appendix F, “Electronic Purse Schemes” on page 195.

In the case study, the purse is very simple. It has minimum security features for the sake of simplicity and will only be used in a closed system within GTT. The card issuer and application provider are one and the same: GTT. GTT has total control over the application and the flow of money within the scheme. The users of the purse scheme are from a small, select group of people (for example GTT employees). As you can see in Table 8 on page 107, we have reserved room for only one key; in public electronic purse scheme can be as many as fifteen.

In the second case, where purses are implemented by banks and other financial institutions, the smart cards and smart card readers are distributed countrywide or worldwide. The card issuer may not necessarily be the application provider. The card issuer cannot select the customers to whom he issues cards.

In summary, the purse we describe for GTT will not satisfy the security requirements of a publicly available purse of a financial institution.

7.6 Digital Signature

GTT must decide whether it wishes to operate its own certificate authority (CA), the machine that will issue, track and revoke digital certificates. If GTT decides to have its own CA, then it must obtain and install CA software like GTE′s CyberTrust, Nortel′s Entrust or IBM′s Vault Registry. In this case, GTT would have to install a user support infrastructure within the company to answer user questions and solve technical problems with the CA. They would also have to come up with policies to issue, maintain and revoke certificates plus the tools and infrastructure to enforce those policies. There are also issues about the physical protection of the server to be resolved: building, access control, backups, etc.

Due to the limited number of employees that will be doing digital signature, GTT will be better off contracting CA services from a company that operates a public CA such as Equifax or Verisign. In some cases, this decision simplifies the process of establishing digital certificates within a company because the policies and tools for managing the certificates are already in place.

As was mentioned in 2.6.1, “Certificates” on page 21, a digital certificate can have many extendible fields. GTT can add as many as it likes so it becomes a flexible way to associate personal information with a particular individual and maintain its integrity. GTT will have to negotiate a contract with the selected CA to get a reduced price for a given number of certificates. In the present situation there is no need to add any additional fields to the certificate.

The smart card MFC 4.21 is capable of doing asymmetrical encryption on the card (see 2.3.3, “Asymmetrical Cryptography” on page 17 to find out more about asymmetrical encryption). It is equipped with a crypto processor.

The software proposed is the IBM Digital Signature Solution. It runs under Windows 95 or Windows NT, supports the PKCS#11 standard and for e-mail uses

Netscape Communicator. See *IBM Digital Signature Solution: Concepts and Implementation*, SG24-5283 (available at a later date) for more information.

In Chapter 12, “Card Management” on page 145 we cover how the keys and certificates should be stored in the cards.

The data stored in the card is basically the keys and the certificate. A total of 2874 bytes have been set aside to store this data. See 8.7.1, “Sample Memory Requirements” on page 107 for details.

The employees using digital signature will have to have a smart card reader attached to their PCs running a Windows operating system. They need to be PC/SC complaint or PC/SC Migration compliant.

Two type of readers in this class are:

1. A reader that will fit in a PCMCIA slot for use by the employees who have been assigned laptops
2. A reader that hooks up to the serial port (RS-232) for desktop PCs

These are contact readers and do not require landing contacts. Care will have to be taken with the readers′ electrical power consumption. It has been known that some COM ports are not able to provide enough millamperes to the serial reader, mainly in laptops. To overcome this situation, some of these readers take the power from the keyboard plug. The keyboard attaches to a special plug in the reader′s cable.

Fortunately for GTT, the selected laptop for all employees is the IBM ThinkPad, what will mean fewer tests.

On the desktop side, GTT has many different brands, which will make the tests more difficult. But in these systems the COM port power is usually not an issue.

In the first stage, only a fraction of the employees will be given these readers. As GTT will use the IBM Digital Signature Solution, this offering is sold as a bundle with a choice of different readers from Utimaco: PCMCIA and serial port.

An incomplete list of vendors that manufacture PCMCIA and RS-232 readers follows:

- De La Rue Card Systems
- Gemplus
- Schlumberger Electronic Transactions
- SCM Microsystems
- Toshiba
- Tritheim
- Utimaco

See Table 4 on page 64 for details.

Another interesting possibility is to use keyboards that have an integrated smart card reader. An incomplete list of vendors that produce this product is:

- Cherry Mikroschalter Gmbh
- Alps
- Keycorp Ltd.
- Keytronic

7.7 Network Access

There were three areas to cover:

1. Network computers

   GTT will be migrating some PCs to Network Computers (NC). Digital certificates can be employed in this application for two purposes. The first purpose is for checking the integrity of the operating system that gets downloaded to the NC. The second purpose is to verify the sign-on of a user who attempts to use the NC. Once the identity of the user is verified, the user's application profile will be automatically set up.

   This approach is in an early stage. No attempt will be made to implement this solution in the first stages of the smart card implementation.

   An area within the digital signature space will be used for this purpose. See 8.7.1, "Sample Memory Requirements" on page 107 for details. If a digital certificate is needed for authentication to the NC server, the one stored for digital signature could be used.

2. Workstation access

   Utimaco sells a product that does what GTT wants to do. Only when the smart card is inserted in the smart card reader will the PC be functional. The data for this application is stored as an object within the area reserved for digital signature and access using PKCS#11. We did not reserve space for this object in the card layout, but it could be added later.

3. Single sign-on

   This area will be addressed in the future, as we cannot offer an integral solution right now.

7.8 Health Alerts

This alert system will be voluntary. The employees are not obliged to participate in the program if they feel it impinges on their rights to privacy. For example, some people may have an objection to letting people know that they have a communicable disease.

The following data can be stored on the smart card for employee health alerts:

- Blood type
- Special health requests
- Allergies
- Name of the family doctor
- Name of the health insurance company
- Hospitalization plan number or health insurance number
- Employee will or will not donate organs in case of death

140 bytes have been reserved for this data. See 8.7.1, "Sample Memory Requirements" on page 107 for details.

It was decided to make the information read always because there might be a situation where the employee is unconscience or otherwise unable to communicate to medical personnel critical health problems. The medical personnel will have access to readers on GTT premises in an emergency.

An application will be available to download from a server that will allow an employee with a smart card reader attached to his/her PC to read and update the health data. For employees without a smart card reader or a PC, stations will be available for updating and reading health information. See Chapter 12, “Card Management” on page 145 for more details.

The employee will be informed that if he loses the card, another employee would be able to read the health information.

A more sophisticated method was studied and later discarded. It was proposed to store the health data as an object created using the IBM Digital Signature Solution in the private area of the card. This area is password protected, but can be accessed by a security officer. In this particular case the security officer would be the physician or the nurse. This method has the advantage that only the employee and the emergency unit people have access to the data. The disadvantage is that only a fraction of GTT employees will have access to the digital signature software.

An even more sophisticated method was also studied and discarded. In this method all the cards would have the same data as the cards used for digital signature. In the emergency room there would be a smart card reader with two slots. The physician would insert his smart card in one slot and the patient′s card would be inserted in the other slot. Both cards would authenticate to each other and after a successful exchange, the patient′s card would provide the health data to the application to display.

7.9 Biometric Access to Restricted Areas

The objective and implementation of this application have been explained in Chapter 6, “Case Study” on page 69.

A couple of vendors that manufacture fingerprint smart card readers are Omron Electronic and Cherry.

Two areas of 402 bytes have been reserved in the card EEPROM to store two fingerprints: one for the right and one for the left thumb. See 8.7.1, “Sample Memory Requirements” on page 107 for details. GTT will have to establish some policy in the case of missing thumbs.

The smart card readers will perform the same functions as the access control terminals, once the fingerprint check is successful. This means that the card terminals must be online to check the card against the black list.

7.10 Time and Attendance

The smart card does not add any new functions to this application, at least during the migration period.

The existing magnetic stripe terminals can continue to be used.

In a second stage, the contactless smart card readers could be used for this application. By this time, it will be possible to store in the card the time the worker has been accumulating, divided into normal hours and overtime. The worker would be able to check both counters with a pocket balance reader, like the one in Figure 12 on page 56.

Some additional information related to time and attendance could also be stored in the card, such as hours worked on Sundays, or holidays, or Saturday afternoons, etc. The worker would not have to wait until the paycheck is issued to detect possible errors.

No extra memory was reserved for this application. The necessary data for time and attendance is already stored in EF_Employee file. See 8.7.1, “Sample Memory Requirements” on page 107 for details.

7.11 Travel Applications

This requirement will not be addressed in this case study. The reason is the complexity. The travel application requires that GTT negotiate with outside firms to participate in the scheme.

If they successfully find a travel firm to participate, the individual transactions systems that currently exist would have to be examined to see how compatible they are. If they prove to be compatible, a plan will have to be drawn up to detail the required changes, the testing procedure and the rollout plan. All this is a major project in itself.

We did not reserve any memory for this application.

Chapter 8. Card Layout Design

So far we have concentrated on the external aspects of the smart card. In this chapter, we will look at how the chip is organized internally and what's required to transfer it from the manufactured state into a functional smart card.

We will describe the card's file system and security architecture and use the case study as an example of how to fit the applications we have chosen into the smart card. We show how this layout can be easily created using the IBM Smart Card ToolKit to prepare a working smart card.

8.1 Purpose of a Card Layout

When a smart card is manufactured, it's in an empty state with the exception perhaps of providing some information about itself (protocol, operating system version etc.) to the card terminal. Before the card can be used with an application, it must first go through a setup process called *initialization*. This is similar to formatting a PC hard disk and creating files and directories common to all the machines.

The file layout, which is a representation of the smart card applications that we require, is generated by giving a sequence of instructions to the smart card processor chip. To enable thousands of cards to be prepared this way, the instructions are stored in a file called an *initialization table* (or in an *initialization script*).

During initialization, the card's EEPROM memory is erased and files and directories are created according to the layout that's specified. In addition, some data common to all the cards being issued may also be loaded into the EEPROM memory locations during the initialization.

Depending on the operating system/card supplier, the layout generating procedure is accomplished using different methods.

One method uses a simple tool to manually create the card structure, file by file and save the commands in a script file. A card simulation tool may be provided to enable the layout to be tested without using a real card. The script is later given to the card supplier, who uses internal merging tools to create commands for each card and uses these in the card production process.

In another method, as is the case with the IBM Smart Card ToolKit, the layout is generated quite easily using a high-level language to create a *layout file*. This is later compiled into the necessary instruction tables using a *layout compiler*. Simple keywords allow different chip processors and operating systems to be specified. At the time of writing, the IBM Smart Card Compiler is unique in the industry and other manufacturers have shown an interest in adopting a similar technique.

The initialization is usually carried out as part of the chip embedding process during card production. This gives added security, since the process is carried out under strictly controlled environments, possibly certified by banking and credit card organizations.

The second step is called *personalization*, where information unique to the cardholder such as name and account number is written to the card. The reason for a two-step process is to provide a technical and cost-efficient process. The personalization process can be a complex and slow operation. It has to load the correct data (for example, name, specific keys) into the smart card maintaining data confidentiality and also needs synchronizing with other card surface information and possibly magnetic strip data. Another benefit of this two-stage process comes from being able to conduct the personalization in more secure premises using cards already initialized. After the personalization process the cards are ready for delivery to the cardholder.[4]

Please refer to Chapter 9, “Smart Card Production” on page 111 for more details about smart card personalization.

8.1.1 File System

This section describes the file system used by the smart card and the features used to secure the data.

The card data is stored in the EEPROM memory in a similar way to a PC hard disk′s file system, which has directories forming a hierarchical tree like structure. At the top of this tree (see Figure 20 on page 89), is a Master File (MF). This file is mandatory. Leading off the Master File, you can have either Dedicated Files (DF) which can be regarded as application directories and Elementary Files (EF), which are used to hold the cardholder′s data and other information such as keys and passwords.

The Dedicated Files logically separate the data and also enables you to specify different security protection levels to the EFs underneath. The files used by the card for management and control purposes are called Internal EFs and the files that hold data accessible by the outside world are called External EFs.

8.1.1.1 File Access Methods

The ISO standards have specified that any file should be accessible by at least one of the following:

- **File Identifier**: A 2-byte hexadecimal code. The MF, for instance, has the file ID 3F00 which is defined by ISO. An EF could have the file ID 2F01.
- **Path Name**: A concatenation of file identifiers (example 3F00A0010011) which is the path from the Master File (3F00) via the Dedicated File (A001) to the Elementary File 0011.
- **Short Identifier**: A short file identifier may be used with an EF which has bits 1 to 5 coded with a number in the range 1 to 30.
- **DF Name**: A DF may have a unique name coded using 1 to 16 bytes. For example EMV specifies that the Payment Systems Directory should have the name ′1.PAY.SYS.DDF01′.

[4] The terms initialization and personalization are used by some card manufacturers, for example IBM. Some others, for example Gemplus, combine both processes under just one name: personalization.

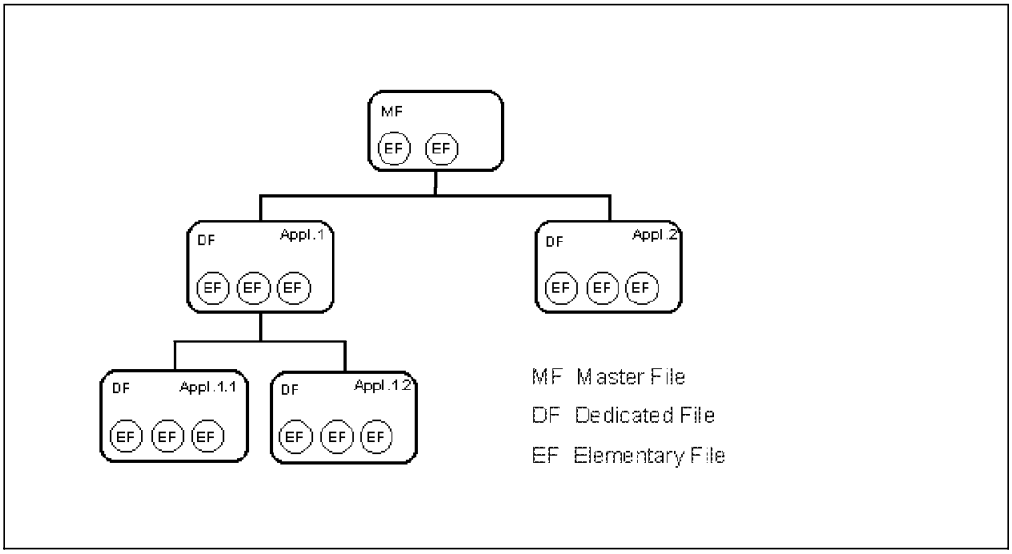


Figure 20. Smart Card File System

8.1.1.2 Elementary File Structure

In the EF, the data can be organized into different file types. The file types are defined by ISO and are shown below:

- Transparent
- Record
  - Linear Fixed
  - Linear Variable
  - Cyclic Fixed

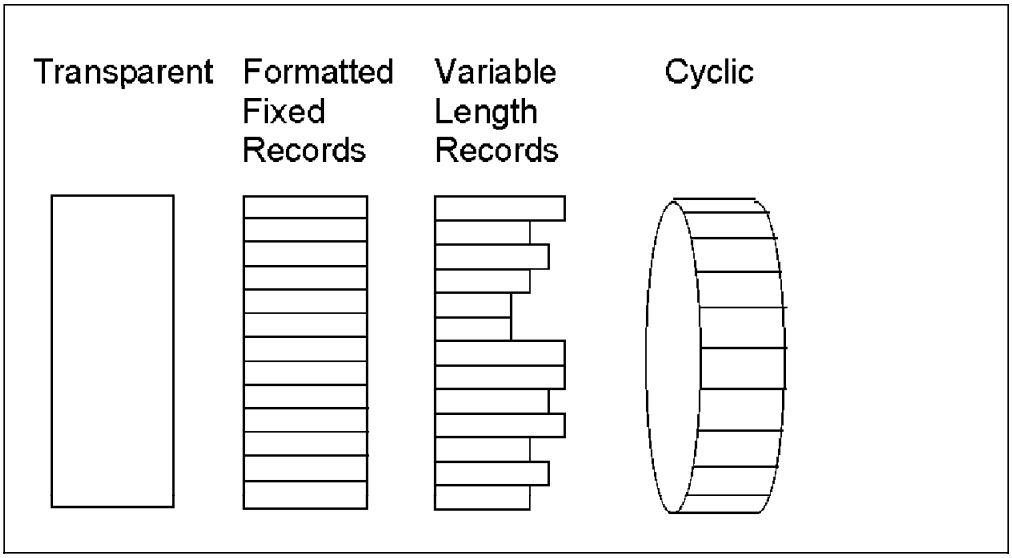


Figure 21. Smart Card File Types

Transparent

A transparent file is used to store information as binary data and the file has no internal structure. The theoretical limit to the data size is 64 KB. This size is much larger than the 8 KB, 16 KB EEPROM size which the current technology

provides. (Also refer to 8.1.2, “Smart Card Profiles” on page 91). In practice the data unit is much smaller, in the order of tens or hundreds of bytes to achieve a compromise between data size and time required to read/write the data.

Instructions used for operation on this type of files are READ_BINARY, UPDATE_BINARY and WRITE_BINARY. The data is accessed using an offset relative to the start of the file.

In the following figure, for example, the employee′s last name would be read by supplying an offset of 10 to the READ_BINARY card instruction.

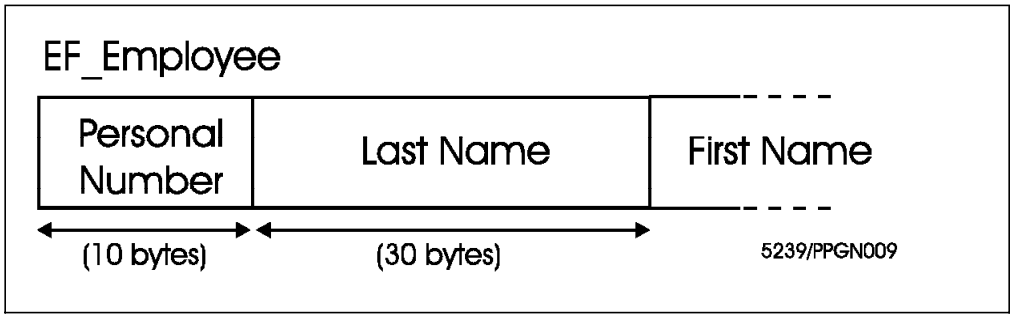


Figure 22. EF Data Storage

A transparent file is typically used for data items such as the cardholder′s name, an account number, personal information, password and keys. These files can also be used to store operating system extension code and executable code called Application Specific Commands (ASC).

Linear Fixed

This type of file consists of fixed-size records, each record consisting of a sequence of bytes. The maximum length of a record is 254 bytes and the sequence starts with record number 1. The maximum number of records is also 254; the index 255 is reserved for future use. A record is accessed in its entirety, and different parameters are available for selecting individual records; for example FIRST, LAST, NEXT or a particular record number. Instructions used are READ_RECORD, UPDATE_RECORD and WRITE_RECORD.

A linear fixed file is typically used to store a cash transaction log, list of applications keys.

Linear Variable

The record format of linear fixed files wastes space when the records are of unequal length. This can be avoided by using a file type of linear variable. Each record now has an additional byte to indicate the record length. Instructions and maximum limits are identical to the linear fixed method.

A linear variable file is typically used to store a list of authorized applications or a list of telephone numbers.

Cyclic Fixed

This is similar in its structure to the linear fixed record layout. A pointer is automatically incremented and points to the last record written to. Once the maximum number of records have been written, the pointer is automatically reset to the top. The record numbering is slightly different in that the record to

which the pointer is currently set is logical record number 1. Instructions and record limits used are identical to the other record access methods described.

Cyclic fixed files are typically used for keeping a list such as hotels visited or a record of last accessed services.

8.1.2 Smart Card Profiles

One provision made by the ISO 7816-4 and CEN726 standards is the so-called Smart Card Profile's, which defines a subset of the file types and applicable instructions.

These profiles are given a letter: Profile M, Profile N and so on. Profile M, for example, has only the transparent and linear fixed file types. The maximum length addressable in the transparent file is limited to 256 bytes and no implicit record selection is possible.

This profiling allowed early smart card chips, which were memory constrained to be ISO compliant. However profiling is no longer considered a cost-effective feature of smart cards today since supporting a fully functional card is no more expensive than supporting a card with a subset of the functions.

8.1.3 Data Structures

The data is stored in the file according to the format required by the application. A static data structure, however, can cause problems if, for example, a data field length needs to be made shorter or longer to what's set when the card was issued.

Problems resulting from such changes frequently cause application breakdown resulting in considerable time, effort and cost to diagnose and correct the problem. One solution to this problem is to use a form of header or tag to describe the data length and the content. This is defined as a standard known as ASN.1 (or Abstract Syntax Notation One). The Basic Encoding Rules (BER) of the ASN.1 is implemented in a structure called TLV (Tag Length Value).

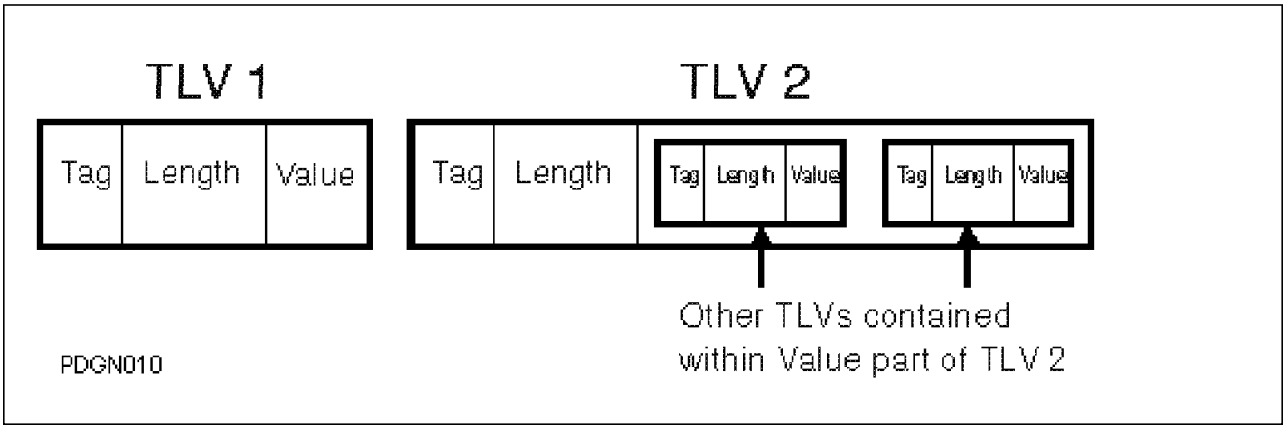


Figure 23. TLV Data Structures

8.1.3.1 TLV Structure

This type of file structure can be used in any of the above file type organizations. The data is accessed using a tag and the adjacent length field. The TLV structure can be used to define more complex structures such as TLVs within another TLVs. An example of where the TLV structure is used is in the EMV specification for payments, as shown in Table 5.

Tag	Length	Value
′9F01′	6	Acquirer Identifier (6 bytes)
′9F42′	2	Application Currency Code (2 bytes)

Table 5. Sample TLV Structure from EMV Specification

8.1.4 File System Security

Security is the single most important consideration for adopting a smart card solution. For this reason, special care must be given to how the data is accessed and who can access the data stored in the smart card.

The file types described above can be selectively protected using *access attributes* with *access conditions*. For example an attribute might be *read* and a condition might be *cardholder verification*. This means that in order to read the file the smart card would require the correct password to be entered.

The basic security mechanisms defined by ISO 7816-4 are:

- Authentication with a password

  The outside world must prove knowledge of the password before being allowed access.

- Authentication with a key

  The outside world must prove knowledge of the key already stored in the smart card before being allowed access.

- Data authentication

  The card attaches a code to the data so that the outside world can ensure that the card is not fraudulent.

- Data encipherment

  The card encrypts the data so that the data confidentiality is maintained.

Each card operating system from smart card manufacturers will have this security implemented using the access attribute and access condition combinations. These operate at the file (MF, DF, EF) level. Usually MF and DF are given a high level of security so that only the card issuers or application providers can manipulate these files (if at all). For example, application providers may be given access to disable their applications, but only with the knowledge of special keys, which are stored in the file hierarchy.

The implementation may be the basic ISO security mechanisms or, for example, as in the case with IBM MFC, with added enhancements which improve security by combining the mechanisms and provide password authentication with data encryption.

The access attributes allowed in the IBM MFC Smart Card Operating System are:

- **(ALW)** Always

- **(NEV)** Never
- **(CHV1)** Cardholder′s Password/PIN
- **(CHV2)** Administrator′s Password/PIN
- **(PRO)** Protected
- **(AUT)** Authenticated
- **(ENC)** Enciphered
- Combinations with CHV1 or CHV2
  - **(CHV1/PRO)**
  - **(CHV1/AUT)**
  - **(CHV1/ENC)**

Note: The acronym in capitals shown above is the notation used in the IBM Smart Card ToolKit layout creation tool.

The above security mechanisms are described in the following sections:

Always (ALW)

As the name suggests, the smart card permits free access to the data for read/update. Typically, data such as cardholder name, a frequent flyer number, or purse amount is granted a read access (but not update) of ″Always″.

Never (NEV)

The highest level of protection is ″Never″, since not even the card issuer is allowed access. A password, for instance, has the password file protected with read access ″Never″. No one outside the smart card can read this data; the code running inside the smart card can, for example, do a cardholder verification. Other files with this level of protection would be the Master File and Dedicated Files.

Cardholder Verification (CHV1/CHV2)

The advantage of a smart card over a magnetic stripe card is that the cardholder′s password or PIN can be stored in the card in a way that it can never be read from the outside. The comparison of the password or PIN, known only by the cardholder, is done within the chip.

Usually, a card has two passwords CHV1 and CHV2 where the second password is used for administrative purposes such as resetting a forgotten or blocked CHV1 password. Data items protected by a CHV requires the external world to supply the correct password or PIN in order to access the data for reading or writing.

Protection (PRO)

To check that the messages exchanged between the smart card and the host application is not altered, a Message Authentication Code (MAC) is appended to the message if the file access condition PRO is specified. The MAC is generated using a function with a key, a random number and the message as input.

Authentication (AUT)

There are two authentication methods: external and internal.

- External Authentication

  The (external) application must prove to the smart card that it is authorized to access/update the data in the smart card, that is,

both must have the same keys. The application authenticates itself by requesting a random number from the smart card and showing that it has encrypted it correctly.

- Internal Authentication (Card Authentication)

  The smart card must prove to the application that it is authorized to send data to the application. The smart card authenticates itself by requesting a random number from the Application and showing that it has encrypted it correctly.

Encryption (ENC)

This command is similar to the PRO command except that the whole message is encrypted.

8.1.4.1 Examples of Applications

We can now look at some examples of where this security applies.

Data Integrity (PRO)

When data is exchanged between the smart card and the outside world, for example using an unsecured or uncontrolled environment like the Internet, it is important to ensure that the data is authentic and remains unchanged. Any attempts at changing sensitive data such as payment confirmation from an electronic purse by criminal elements needs to be protected.

To ensure integrity and authenticity of sensitive data, a (MAC) is used. This is achieved with the PRO access condition.

Data Confidentiality

Secret data can be protected by changing them to an illegible form using cryptographic keys. This is achieved using the ENC access condition.

Digital Signatures

With the use of public key algorithms, the smart card can be used to sign a document using a ″digital signature″. Cryptographic information created with the private key of the sender can be verified by the recipient using the associated public key. The advantage of digital signatures is ″non-repudiation″ where the sender cannot deny that he/she created the message.The digital signature uses the PRO or ENC methods described above.

8.1.5 Security Domains

Since the smart card can contain many different applications from different providers, a method is required to provide ′fire-walling′ between the applications. This is done by storing the application provider′s unique set of keys to define the area of control within the smart card and is given the term Security Domain.

The domain consists of a group of elementary and dedicated files as shown in the diagram below.

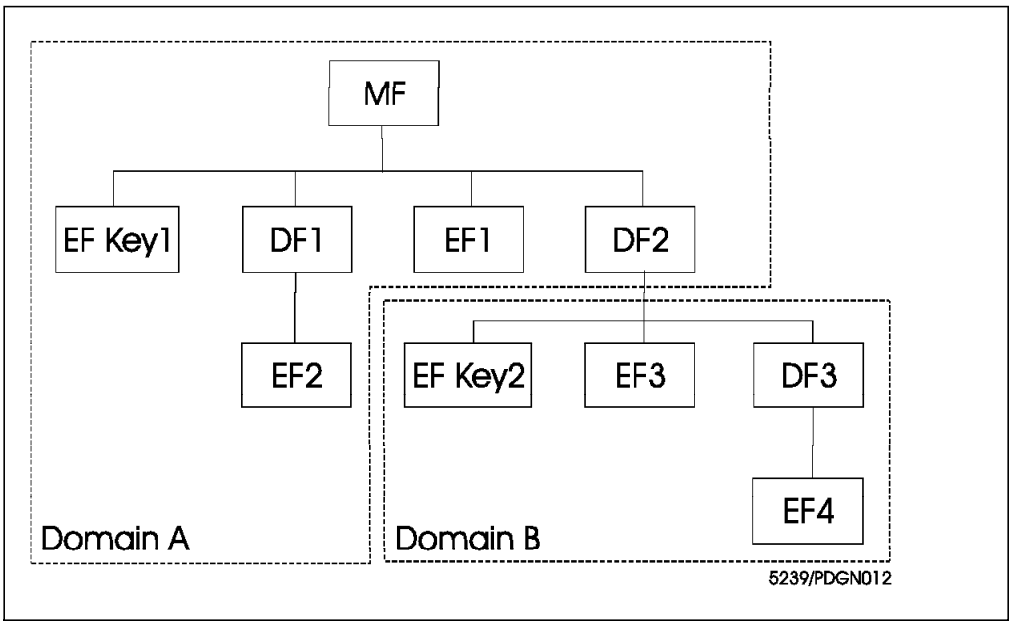


Figure 24. Security Domains

In the object hierarchy given above there are two key domains for the two key files displayed. EF Key1 is the relevant key file for EF1 and EF2. EF Key2 is the relevant key file for EF3 and EF4. So the key domain for EF1 and EF2 is A and the key domain for EF3 and EF4 is B.

8.2 Card Design Process

In the majority of cases, Smart Card Applications will use the card simply to hold small amounts of data with appropriate security around it. This is normally quite adequately provided by the existing card operating system instruction set.

There is the possibility to develop custom operating system (ROM Masks) and ASCs, but these demand a great deal of effort in terms of specialist skills, time and effort with very significant cost implications. The development of the ZKA ROM Mask is a good example of this where the IBM MFC Operating System was customized for the German banks where specific processing was required. The majority of projects, however, do not require such levels of sophistication and complexity.

As a starting point in designing the card layout, we make a list of all the applications that the smart card is required to support, together with any known requirements for future applications. This gives an initial sizing estimate from which applications can be adjusted as required to fit into the smart card. Translating the applications into the smart card hierarchical structure is not too difficult since the Dedicated Files naturally align themselves with the different applications under consideration.

The smart card standards also dictate some EF files, such as card identification, password and key files, which needs to be coded in the root directory (MF). When specifying a layout for the smart card, there is a small overhead associated each time a file is created and this needs to be added to any sizing estimates.

8.2.1 Consideration for Card Design

Defining the data fields in the EFs is more difficult, since we have to take into consideration the performance and the amount of data. Also note that writing to the EEPROM takes a bit longer than reading because in EEPROM, the storage has to be erased before writing. Where the card reader has no mechanism to lock the card while writing is taking place, one needs to keep in mind the action of a user who might remove the card before all the data is written. In this instance the state of the data might end up corrupted. For such possibilities the precaution of adding a status flag or writing a final check-sum byte to the file is included in the design. Another technique especially used in purses is to write a log of the protocols so that if the process is interrupted, corrective action can take place when the card is next used.

8.3 Card Design Checklist

The following checklist is a guideline to the steps required for designing the card layout:

- Define the card data
  - Identify the data elements
  - Check on the EEPROM memory requirements
  - Select the file types to store the data
  - Add mandatory files and files specified by standards
  - Design the file directory structure (tree diagram)

  When allocating files, the sizes of the data elements and TLV structures may need to be taken into consideration and separated according to their frequency of access and update requirements. Note that writing data to the EEPROM usually takes longer than reading the data.
- Define application security requirements
  - MAC/ENC coding for card data transmitted over unsecured networks
- Define card issuing requirements
  - Keys and passwords required for initialization and personalization
  - Data protection and encryption mechanisms for the cardholder data and application data to be loaded onto the card
- Define card administration requirements
  - Design how to reset blocked cards
  - Design how to exchange application keys if required
  - Design the administration of PIN/passwords (for example: change PIN, forgotten PIN, retry attempts)
- Take into account any future requirements

  For example, creating DF(s) to store new application(s).

8.3.1 Answer to Reset (ATR)

The EF_ATR is an Elementary File that supplies the data to the external world in response to a RESET command. This reply is called the ″Answer to Reset″ or ATR. If the card is not in the initialized state (doesn′t have an EF_ATR file), the card normally responds with the ATR information hard coded in the ROM. The content of this file is defined by the ISO 7816 standard. The ATR is composed of a standard 8-byte prefix section, followed by a variable 15-characters historical

bytes defined by the card issuer and a final check character making a total of 24 bytes. The flexibility is therefore available to keep this ATR short where speed of the transaction is important.

In today's environment where terminals and applications have to interface with multi-vendor cards, the ATR performs a key role in addition to negotiating the protocol. In Chapter 10, "Smart Card Programming" on page 119 we describe a technique where the ATR is used to dynamically load the card's unique instruction set. This technique closely follows the open card framework standard.

The prefix header contains interfacing details such as the type of protocol used (example T=0 or 1). The historical characters contain various data elements covering the chip manufacturer's identification code, operating system version and other discretionary data.

8.3.1.1 Defining the ATR for GTT

The card we have chosen for GTT has T=1 protocol and uses the IBM MFC Operating System Version 4.21. The header bytes in the ATR is chip dependent and these have been taken from the manufacturer-supplied data. The format of the ATR has been coded according to the ISO 7816 standard and the EMV specification.

8.3.1.2 Historical Bytes

The ATR historical bytes used in GTT will be defined as proprietary by setting the first byte to 0x7E. This indicates that the second byte identifies the card operating sysem and the third byte indicates the EEPROM level of the card.

```
Header data prescribed by the card operating system
0x17            - length of ATR is 23 bytes (Hex 17)
                      (without the check character)
0x3B            - TS  : Logic Convention  '3B' or '3F'
                        3B = Direct,  High=Logic 1
                        3F = Inverse, High=Logic 0
                       (EMV recommends '3B' convention)
0xEF            - T0  : T=1 indication.
                        'E' means T=1. '6' is for T=0.
                        'F' means 15 historical bytes.
                       ('E' also says, TB1, TC1, TD1 is
                        present to give T=1 characteristics)
0x00            - TB1 : Programming voltage, VPP not required
0xFF            - TC1 : Guard Time between chars (12 etus)
0x81            - TD1 : Further info in TD2 follows (T=1)
0x31            - TD2 : Further info in TA3, TB3 follows (T=1)
0x66            - TA3 : ICC Information Field Size 102 (T=1)
0x45            - TB3 : Block/Char Waiting Time (T=1)
                        '4' block waiting time
                        '5' char waiting time
Historical data defined by the Card Issuer
 0x7E             - Proprietary Historical Bytes.
0x09            - MFC 4.21 Card.
0x38            - 8KB EEPROM
0x47            - 'G'
0x54            - 'T'
0x54            - 'T'
0x20            - Nine blank filler (hex 20) to make up
0x20...0x20       15 bytes of historical data.
0x8B            - TCK : Check character
```

The ATR we defined above is included in the card layout definition shown in Appendix A, “Sample Smart Card Layout for the Case Study” on page 169. The final check character allows the integrity of the ATR data to be checked. The value is calculated so that an exclusive-OR operation of all the ATR bytes from ′T0′ up to and including the ′TCK′ character results in a null value. (In the case of the MFC smart card, the check character is automatically calculated by the card operating system during EEPROM initialization and therefore does not require to be included in the layout definition).

For more details please refer to the *IBM Smart Card ToolKit User′s Guide*, ISO7816 and EMV.

8.3.2 Application Registration

As we mentioned earlier the card′s applications are stored in Dedicated Files (DFs). The identification of this DF with just the 2-byte file identifier (FID) is probably insufficient to meet future demands and may cause problems with duplication among different application providers. The rules governing the definition of the FID is that it is unique within a given card.

A solution to this is specified by ISO where the DFs can have an Application Identifier (AID) in addition to the FID. This AID consists of two parts: a Registered Identifier RID (5 bytes) and optionally a Proprietary Application Identifier Extension (PIX) (0 - 11 bytes). The RID is allocated by a national standards body in each country. The organization requesting an RID has to be a registered company in the country.

```
AID             PIX

DC CC XX XX XX  YY YY YY YY YY YY YY YY YY
|   |
|   └Country Code
|
└Always Hex D to signify this was obtained by
  a national registration authority.
```

For example, an application provider in Germany would contact the following authority to register the application and obtain an Application Identifier.

```
 RID German national registration authority
(c/o GMD, Bruno Struif)
 Rheinstr. 75
 D-64295 Darmstadt
 Germany
```

For our sample case study, we assume that GTT has applied to the USA Standards Body (Country Code 840) and received
the AID components RID:D840000025 and PIX:0x10 to 0x50.
These AID components are used in the card layout definition shown in Appendix A, “Sample Smart Card Layout for the Case Study” on page 169.

8.3.3 Key Management

When a card layout is first designed and tested, we do not use actual card issuer's keys but specify some test keys. When the cards are produced for a live environment or subsequently need updating with new keys, the keys are transported and loaded to the cards using a special set of management keys. The management keys are stored in a special *key management* EF file.

It is a good practice to define and personalize sets of test cards during development and test, which can also be used for software problem determination.

8.3.4 Personalization Keys

When the card needs personalizing with cardholder-specific data (including keys), the card issuer may require that the data is securely encrypted during transport to the card production site. For this purpose, keys are loaded to the card during initialization and these keys (personalization keys) are used by the chip to decrypt the cardholder data before storing it in the EEPROM.

Note: This personalization key is a mechanism used in the IBM MFC card. Other card suppliers may have internal processes operating at the production site to handle data security.

8.3.5 Derived Keys

Keys provide security, but sharing the same set of keys in all cards doesn't inspire confidence. It's better to have unique keys for each cardholder. The technique used to have unique keys is to derive the keys by combining a master key and some feature unique to the cardholder such as an account number or card serial number.

This method allows for example an ATM to calculate the derived key and use this during the transaction.

8.3.6 Accessing Smart Card Data

Accessing the data requires knowledge of the exact location of the data element (file IDs, lengths and data). However, by using the IBM Smart Card ToolKit, an easier and more elegant access method can be achieved using an *alias* name. We can specify the alias name at the time of designing the card layout using abstract labels.

Accessing the smart card data is covered in Chapter 10, "Smart Card Programming" on page 119.

8.3.7 Card Layout for Contactless, Memory and JavaCards

The approach used for storing the data in other card types is different; for instance, there is no CPU in a memory card to create a file and there is no concept of a file system in a JavaCard. The methods which these card types use to handle the data during initialization is discussed next. To assist us in the explanation, we use the process of storing the cardholder's name in the card.

Memory Cards

The process is more straightforward than a contact card since we don't have to worry about a file structure or read/write mechanisms. The memory locations are initialized with the data as part of the chip module creation process.

Contactless Card

Initializing contactless cards is done using a card terminal equipped with the appropriate transmitter device to send the command and data to the card. In contactless cards, there is no concept of a file structure and hence no create file commands are needed. In a combi card, the CPU would process the command in the normal way.

JavaCard

In the JavaCard, the data is encapsulated in an applet object. This means that loading the cardholder′s name is done after down-loading an applet to the card. Then a method in the applet (for example ″updateName″) is invoked to load the cardholder′s name into the JavaCard.

Subsequent applets that need to use the name can invoke this same applet with a method such as ″getName″ and retrieve the cardholder′s name.

8.4 Compliance with International Standards

The specifications set by the standards affect certain files and data content in the card layout. Since there are several standards (ISO 7816, CEN726, EMV, IATA) the use would very much depend on the application area. For instance the use of the EMV Payment Standard for a health card, which carries only medical data may not be applicable. However, the EMV Standard may be a requirement if the health card is also used for prescription payment.

There are some instances where a particular portion from a standard is incorporated because it fulfills a requirement that the other standards have not specified. Again we use the EMV standard as an example where the EMV Directory File provides a neat mechanism that enables a list of card applications to be retrieved. In fact many smart card projects now adopt this feature; the retrieve by application name as defined by EMV is a de-facto standard even in non-payment smart card applications. We have adopted this in our case study.

8.4.1 Defining the Special Elementary Files

These are special files in that they are added to the card layout design in addition to the application directories and files to provide control data for security (password, keys), management (admin keys) and card identification (atr, card_id).

The use of these files and the applicable standards are shown in the table below.

Table 6 (Page 1 of 2). Special Elementary Files

File	EF ID	CEN726	MFC	Description
EF_ASC	9F02		√	Application Specific Commands
EF_ATR	2F01	√	√	Answer to Reset Data
EF_AUT	9F03		√	Keys used for Authentication
EF_CHV1	0000	√	√	Cardholder Password/PIN
EF_CHV2	0001	√	√	Admin Password/PIN

Table 6 (Page 2 of 2). Special Elementary Files				
File	**EF ID**	**CEN726**	**MFC**	**Description**
EF_DIR_MFC	FD01	√	√	Directory of Applications
EF_ICC	0005	√	√	Smart Card Identification
EF_ID	0003	√	√	Card Issuer Identification
EF_KEY_MAN	0011	√	√	Keys to manage Operational Keys
EF_KEY_OP	0001	√	√	Information about Operational Keys
EF_LOG	9F05		√	Logs Operation Errors
EF_NAME	0004	√	√	Name of Cardholder
EF_LANG	2F05	√	√	Language Preference

The EF_DIR_MFC contains an application identifier (ISO, EMV) and the format of this is described in 8.3.2, “Application Registration” on page 98.

8.5 Tools and Techniques

Different card suppliers use different techniques to create the file layout in the card′s EEPROM. In most cases, the card is initialized using a sequence of commands that are similar in format to the standard APDU SELECT, READ and WRITE commands. In this instance the CREATE FILE command is used, but may also need proprietary commands to configure the file control information and management required by the operating system.

Since there are proprietary commands and different file control information, there is no possibility of using one sequence of instructions to initialize for example a Gemplus card and use the same set for Schlumberger, IBM or any other card. The Standards do not lay down any rules regarding the internals of the file system.

At the time of writing, the method employed by most card suppliers require specialized skill to assemble the initialization instruction sequence according to the required file layout. Some manufacturers′ cards do not tolerate any mistakes in the file creation procedure and if this happens the card has to be discarded.

A much simpler and elegant system is available in the IBM Smart Card ToolKit, where the layout is specified in a high-level language and the instruction sequence is output by a layout compiler. We have used this to create the layout for our case study and the process is described next.

8.5.1 The IBM Smart Card ToolKit

The IBM Smart Card ToolKit provides a set of tools to take the card from a conceptual design to a working card and produces a logical card operating system independent mechanism for accessing the data by your applications.

The ToolKit consists of the following:

- Layout compiler (SCTLCOMP)
- Card data access APIs (Agents/Agency)
- Initialization tool (SCTINIT)
- Script processor tool (SCTEXEC)

- Other utilities (SCTBROWS, SCTBUILD, SCTMERGE)

These tools assist you at different stages of the development of smart card layouts as well as smart card applications. In our case study for example, we used the ToolKit in the following way.

Layout Definition

The file structure and data elements that we want to store in the smart card EEPROM (shown later in Table 8 on page 107), is specified in a high-level language layout definition file (″GTT.CLT″). This includes specifying the type of file organization (transparent, record-oriented, cyclic), file names and/or identifiers, access conditions, data loading method (initialization, personalization) etc.

The layout definition file is compiled using the layout compiler SCTLCOMP, which produce a set of files for use during the next stages of smart card development.

The generated files (with the file type extensions) are shown below:

- An card image in binary (.CIB): file for testing
- Agent dictionary (.DIB): to resolve alias names
- Initialization table (.ITB)
- Personalization table (.PTB)
- Initialization script (.ISB): for mass initialization
- Personalization script (.PSB): for mass personalization

The above files are in a binary format, but a readable listing can be generated by using the SCTBROWS tool or other BER-TLV browsing tools, some of which may be available on the Internet as shareware.

Test Card Preparation

During the first stages of card design and application development, a small number of test cards may be needed and we would produce these using the card image file (.CIB) with the initialization tool SCTINIT.

Application Programming

Information about application programming and the concepts of Agent/Agency is covered in Chapter 10, “Smart Card Programming” on page 119.

Smart card Production

During smart card production, we need to prepare a file consisting of the employees data and this data is normally protected with a key (Personalization Key in EF_KEY_PERS file). Each item of data that goes into a smart card file is tagged (TLV format) with the corresponding index that was used during the card layout definition. (Some examples of personal data are shown in our case study card layout in Appendix A, “Sample Smart Card Layout for the Case Study” on page 169).

For more details on smart card production using the files ISB, PSB, etc. and the tools used in this process (SCTEXEC, SCTBROWS, SCTBUILD, SCTMERGE), see Chapter 9, “Smart Card Production” on page 111.

8.5.2 Designing a Card Layout with the ToolKit

The following example shows how the card layout and card chip specific information is specified using a high-level scripting language.

```
ENVIRONMENT
(
   OS TYPE ( MFC_421 )
   PROCESSOR ( CF_68 )
   ......
)

DECLARATIONS
(
   gttsample_key_domain(1)
)

DIRECTORY
(
    MF
    ID(0x3F00)
    ACCESS
    (
     ......
    )
    DIRECTORY
    (
       DF_Health
       Id ( 0xC020 )
       ......

       FILE
       (
          EF_MedicalData
          ID (0xC021)
          SIZE (120)
          ORGANIZATION( TRANSPARENT )
          ......
        )
    )
)
```

Figure 25. Specifying a Card Layout

8.5.3 Creating Layouts for Non-IBM Cards

One of the techniques that can be used as a generic procedure for initializing smart cards of any flavor is to use a script procedure and execute the commands using the script processor SCTEXEC from the ToolKit.

For example, the output from the layout compiler's initialization resembles the following script procedure:

```
SCRIPT
 DESCRIPTION
  CARD_OS ( COS_MFC_V421 )
  TIMESTAMP ("19980614")

 PROCEDURE
  ID ("INITIALIZE")
  CARD_REQUEST
   ARGUMENT
    CONSTANT_PART ('B6200000084D46434F53343138')

   RESULT
    STATUSWORDS ('9000')
```

A similar procedure can be prepared for non-IBM cards using the card manufacturer's card commands and the cards can be prepared using the ToolKit script processor, SCTEXEC.

8.6 Skills Required

The layout design tool makes it easy to create a card layout for a smart card, without resorting to a manual to work out which low-level commands are needed to create a file layout. But there is still a degree of skill and experience with smart cards required to use the tool effectively.

As a guideline the card layout designer would need:

- Knowledge about overall system security design
- Knowledge regarding appropriate use of the smart card file system
- Knowledge of the file system security concepts such as authentication, encryption and their uses
- Knowledge about the process used to initialize and personalize the card, so that the appropriate security measures can be incorporated

In addition, the system architect in charge of designing the overall smart card security strategy would be needed to advise in defining the data access requirements, in addition to the card initialization and personalization needs mentioned above.

Once the layout design is completed, the ToolKit provides the means to prepare cards in small quantities with test data for other component testing and application development steps.

8.7 Case Study

Our case study uses multiple applications in a single card. In this section we have taken each of the applications and gone through our checklist (see8.3, "Card Design Checklist" on page 96) to design the card layout.

Facilities Access:

We wanted to enable facilities access to be available both on the chip and the contactless card. This offers the flexibility of using a different interface in case of problems with installing the contactless reader. For fast read access on the contact card, we put the data in an elementary file in the MF and accessed it using a short file ID. This enables the data to be accessed without an explicit Select File command. The existing magnetic stripe system used the card number and we use the same in order to minimize the disruption to the back end system. For time logging, the employee number is looked up at the host from the transmitted card number.

Cafeteria Purse:

The cafeteria purse is implemented as a closed purse. Only one purse is implemented and this is stored in the card as a separate application with security being controlled by its own key file. This key is also used in all the cash loading machines next to the cafeteria.

The card also has transaction and control logs to ensure that transactions integrity is recorded. To enable the employees to view

the amounts spent in the cafeteria (for the last three days) the transactions are recorded in a cyclic file.

Health Alerts:

The health data is updated using an intranet-based application and any updates are under the control of the user. Read access is permitted for use in medical emergencies. It has been agreed to do a trial on this and get user feedback before implementing additional levels of security. It is agreed that adding higher levels of security would defeat the purpose.

Network Access:

The network access application, which contains the logon profile/URL data is stored as a private data object on the IBM SignCard. The memory taken up is about 100 bytes and this storage will use up part of the 704 bytes we have already allocated for the ctr_pri area (ID=C200).

The Network Access data object can be accessed using the Digital Signature IBMsign API based on the PKCS#11 Cryptoki Version 2.01 standard. For more details of the implementation of the IBM Digital Signature Solution please refer to the redbook *IBM Digital Signature Solution: Concepts and Implementation*, SG24-5283 (available at a later date).

Biometric Access:

The fingerprint identification requires space to store a template of fingerprint data. A fingerprint from both hands are scanned (to allow for fingers temporary out of action) and these are stored in a template size of 400 bytes with a 2-byte prefix denoting the actual size. In this instance we could have used a variable record format instead of the fixed file structure.

Other files

We needed to look at the ATR string that was implemented as a separate EF although we could have just used the default ATR response from the ROM-Mask. The card has two passwords CHV1/CHV2, with CHV2 being used for administrative purposes such as password resets.

We have added the application names to conform to the EMV specification using the layout command to flag the DFs we want to add to the root level EF_DIR_MFC file. Adding the CEN standard files for EF_ICC and MFC specific files completed the layout creation.

Note: Additional free space should probably be specified in each of the application directories. But we have omitted this to show a more accurate picture of the data storage requirements.

Once the card layout was created, we used the layout compiler to produce the output as shown below.

```
E:\SCLayout>SCTLCOMP gtt.clt /CI  /S=.;c:\sctkit\inc;

SCTLComp - IBM SmartCard Layout Compiler version 2.60
(C) Copyright IBM Corp. 1996,1998.
- Licensed Materials - Program Property of IBM - All Rights Reserved.


Errors: 0, Warnings: 0

Free space left on card: 2254 bytes (0x08ce)
```

Figure 26. Using the Layout Compiler

Using this information, the card layout we designed is shown in Figure 27 and the type of files we want to use are shown in Table 7 on page 107.

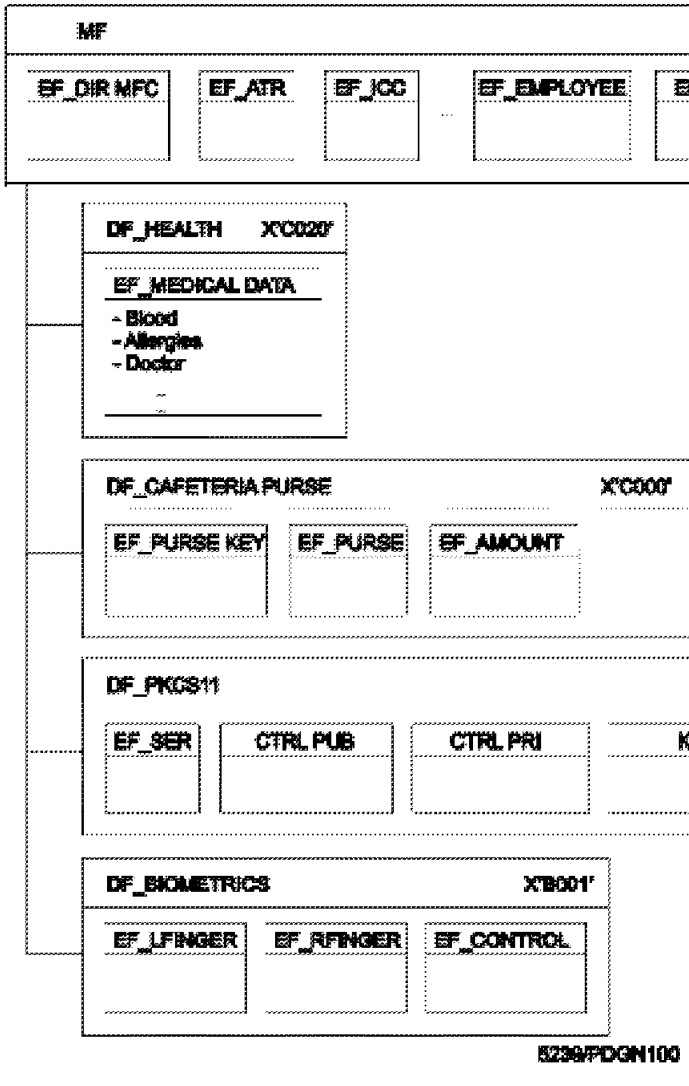


Figure 27. Card Layout (Case Study Applications)

Table 7. Application File Selection				
Application	**Transparent**	**Linear Fixed**	**Linear Variable**	**Cyclic**
Employee Details	√			
Facilities Access	√			
Cafeteria Services	√	√		√
Network Access	√	√		
Biometric Access			√	
Health Alerts	√			
Digital Signature	√			

8.7.1 Sample Memory Requirements

The memory requirements by application and file type are shown below. These figures have been calculated using the layout compiler, which shows the amount of free space left at the end of compiling.

The compiler shows that the amount of EEPROM memory available for the file system is 7860 bytes, after allowing some storage space for the card operating system overhead. The space taken up by the EMV directory entries is shown against the respective DF data column, but the actual storage would be allocated as part of the EF_DIR_MFC file.

Table 8 (Page 1 of 3). Employee Card, Memory Requirements		
Application/File Section	**Data Total Bytes**	**File Management Total Bytes (overhead)**
Control Files (Root)		
Master File		
EF_ATR	25	17
EF_DIR_MFC	10	16
EF_ICC	12	16
EF_ID	21	17
EF_CHV1	23	17
EF_CHV2	23	17
EF_KEY_PERS	16	20
EF_KEY_MAN	16	20
EF_KEY	64	20
EF_AUT	48	20
Sub-Total	**258**	**180**

Table 8 (Page 2 of 3). Employee Card, Memory Requirements

Application/File Section	Data Total Bytes	File Management Total Bytes (overhead)
Employee Details		
EF_Employee	80	16
Sub-Total	**80**	**16**
Facilities Access		
EF_Access	8	16
Sub-Total	**8**	**16**
Health Alerts		
DF_Health	42	18
EF_MedicalData	140	16
• Blood Type	2	
• Allergies	20	
• Doctor	24	
• Phone	20	
• CompanyScheme	1	
• InsuranceCompany	20	
• InsNumber	10	
• RegularMedications	40	
• OrganDonor	1	
Sub-Total	**182**	**34**
Cafeteria Purse		
DF_CafeteriaPurse	48	18
EF_PurseKey	374	20
EF_Purse	36	16
EF_Amount	27	17
EF_LLog	99	21
EF_BLog	111	21
EF_LSeq	22	16
EF_BSeq	22	16
Sub-Total	**739**	**145**
Digital Signature		
DF_Signature	50	18
EF_SER	16	20

Table 8 (Page 3 of 3). Employee Card, Memory Requirements		
Application/File Section	**Data Total Bytes**	**File Management Total Bytes (overhead)**
ctrl_pub	1792	16
ctrl_pri	704	16
EF_KEY_PKA	388	28
Sub-Total	**2936**	**94**
Biometrics		
DF_Biometrics	48	18
• EF_LFinger	402	16
• EF_RFinger	402	16
• EF_Control	100	16
Sub-Total	**952**	**66**
TOTAL	**5143**	**551**

```
EEPROM memory available          = 7948 Bytes
 Operating system overhead         =  116 Bytes
File system overhead             =  551 Bytes
Application data                 = 5143 Bytes
Free Space Left                  = 2254 Bytes
```

8.8 Case Study Sample Card Layout

The layout we designed for GTT, including an example of the cafeteria purse, and digital signature is given in the Appendix A, “Sample Smart Card Layout for the Case Study” on page 169.

Chapter 9. Smart Card Production

This chapter describes the process of taking the card layout we created in Chapter 8, “Card Layout Design” on page 87 and transforming that information to a functional smart card, complete with security keys and the cardholder’s personal data. The process calls for careful planning and preparation, with security a foremost consideration.

There are many parties involved in the production of smart cards, so it should be engineered to deliver an efficient and smooth process, possibly staged at different locations to ensure security is not compromised. We will describe the role that each party plays and the relationship between the different stages of the smart card production.

In this chapter we show how the IBM Smart Card ToolKit can be used for smart card production in our case study. Of course, this is not the only way to send the initialization and personalization data to the card manufacturer; other smart card vendors address this subject using similar techniques.

9.1 Initialization and Personalization Scenario

Before we use the IBM Smart Card ToolKit to create the required files and produce the cards, we should note the key players involved in the smart card production. Some of the terms defined in this section refer to Figure 28 on page 113.

Card Issuer

The card issuer, such as a bank or an enterprise, is responsible for controlling the life cycle of the card. This includes controlling the card production, ensuring the delivery of smart card to cardholders, and informing of them of the PIN (when applicable). This may also be done by a *Trust Center* company who may be subcontracted to carry this task as a paid service. The card issuer is responsible for preparing the personalization data and forwarding it to the personalization site.

Smart Card Application Developer

The application developer designs the smart card layout based on the requirements of the card issuer and the application provider and generates the required files using the layout compiler. To generate layout, files are encrypted using a key (called the transport key) before transmitting them to the card initialization and personalization sites. The key is sent under separate cover.

Card Production Site

The smart card ROM area is written with the specific card operating system (ROM-Mask) at the chip manufacturing site. The card production site is where the chip modules and the plastic cards are assembled. The raw plastic cards that act as the carrier for the chip is milled to mount the chip. Finally the chip module is glued and bonded to the plastic card and the cards are delivered to the initialization site.

Card Initialization Site

At the initialization site, the initialization table or script file is verified using the transport key and the card's file structure with the common data is loaded onto the EEPROM. The personalization key is also loaded at this stage. For large-scale production, the initialization may also take place at the card production site.

Card Personalization Site

Once again, the personalization table or script received from the card issuer/application developer is verified using the transport key (a different key this time) and is merged with the cardholder's data (encrypted with the personalization key) to create a *personalization order*. This personalization order can also consist of other components required to be on the card surface such as the image of the card issuer's logo. Decryption of the card data takes place inside the chip and cardholder data privacy is maintained.

During personalization, the card plastic is printed with the graphical details such as the card issuer's logo, background image and the cardholder's photograph, name and personnel number as specified. For cards with magnetic stripe, the data is coded at this stage.

The personalized cards are checked for visual imperfections and dispatched to the card issuer or directly to the cardholder.

Personalization Order

A data record within a TLV structure, including all personalization data of the cardholder.

Cardholder

The cardholder receives the information from the card issuer regarding the use of the card, terms and conditions and the PIN or password to operate the card. The smart card is received from the card issuer or will be mailed from the personalization site directly.

Note: Loading the cardholder data encrypted with the personalization key, providing a high level of security, may not be present in all smart cards. The feature described here is available in the IBM MFC cards.

9.2 Card Issuance

In Figure 28 on page 113 we show the flow of information during the card issuance. The lady in the top left corner is using the IBM Smart Card ToolKit to create the initialization and personalization tables and scripts. These tables and scripts will be sent to two different sites: the initialization site and the personalization site. In both cases the data will be encrypted before the transport.

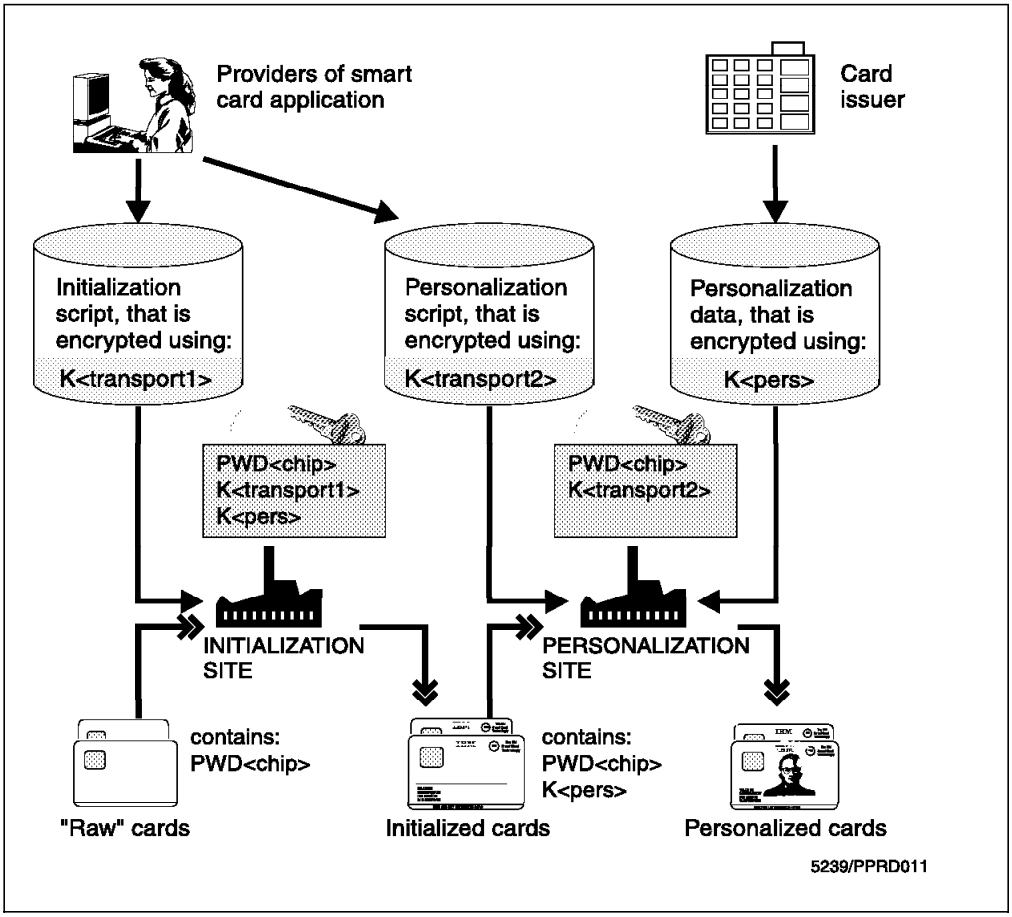


Figure 28. Card Issuance Process Using the IBM Smart Card ToolKit

Figure 28 is mostly self explanatory. We cover this process in more detail in Chapter 12, “Card Management” on page 145. Figure 30 on page 116 takes closer look at the use of the IBM Smart Card ToolKit to create the scripts, showing the different files created with the ToolKit. A similar figure could be done when tables are used instead of scripts.

9.2.1 The Initialization and Personalization Files

We begin with the smart card layout as shown in Appendix A, “Sample Smart Card Layout for the Case Study” on page 169. When the layout was designed there were some fields that could be filled in, such as the ATR and the chip identification information and some fields that were left blank, for example the cardholder’s name. The IBM Smart Card ToolKit enables us to specify these data fields with the INITIAL DATA and PERSONAL DATA keywords. Each different data element that needs personalization is identified by using a 2-byte index field. In the example below the length of the data is 20 bytes and the index is 1.

```
PERSONAL DATA ( INDEX(0X0001 )SITE (20) PROT (MAC(0)))
```

When the layout is compiled, it produces several tables or script files to assist in the smart card production. These files are shown in Table 9 on page 114.

Table 9. Layout Compiler Output Files.			
File	**Compiler Option**	**Default Extension**	**Purpose**
Card Image Binary File	/CI	.CIB	A binary representation of the card EEPROM. Generally used to produce some test cards during development.
Initialization Table Personalization Table	/IT /PT	.ITB .PTB	Tables which can be used by your program to perform initialization or personalization.
Initialization Script Personalization Script	/IS /PS	.ISB .PSB	Scripts which can be used to perform initialization or personalization.

The initialization and personalization tables/scripts can then be used for mass production of cards.

Note: The compiler outputs a table and a script file for each of the initialization and personalization steps. You can use one or the other as explained below.

The use of the table or the script depends on how you want to do the initialization. For example, if you want to write your own program to do the initialization or personalization, then you can use the table format. But if you want to use the program supplied with the ToolKit (the script processor, SCTEXEC) then you must use the script file.

The advantage of using the script processor is that the development effort is lower, but if you need more functionality then it is possible to create your own program. If you write your own program, then you must code the logic in your program to interpret the table and substitute the necessary cardholder data.

If however, you use the script processor then the cardholder data is merged with the personalization script into what is called a personalization order. In order to merge, the cardholder data is formatted into a TLV structured file using the appropriate indices that identify the data elements and using the tool SCTMERGE. The content of these files, being in the TLV format, can be viewed using the SCTBROWS tool.

As shown in Figure 29 on page 115 we can enter agent commands and vendor-specific commands (APDUs).

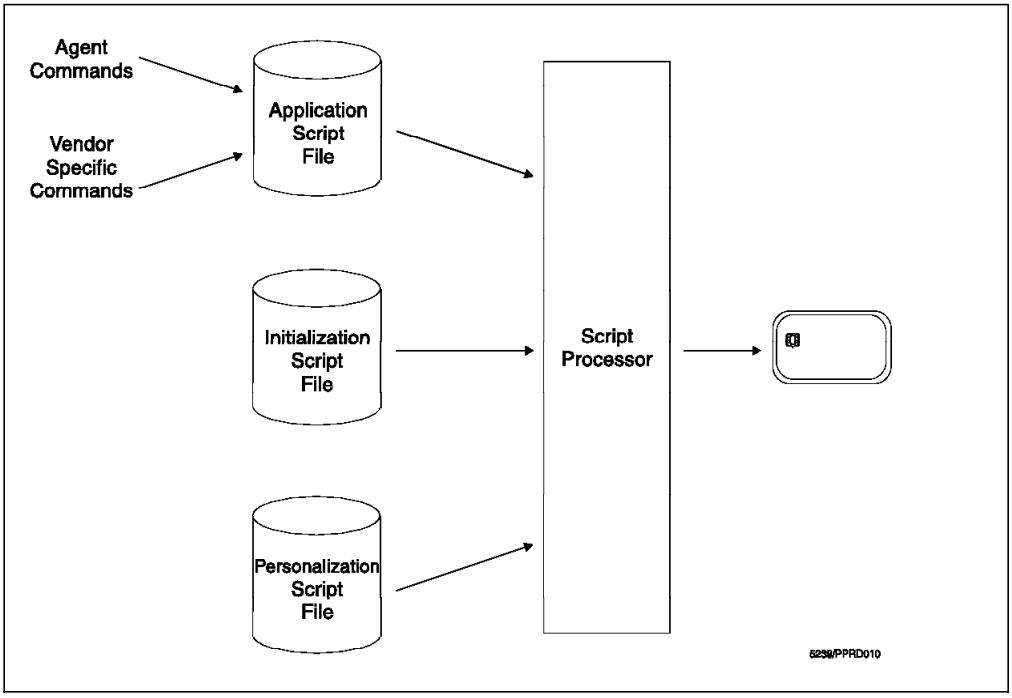


Figure 29. Script Processor

Examples of how to format the cardholder data into a TLV data format is given in the ToolKit documentation. Figure 30 on page 116 shows in detail how the different script files produced by the IBM Smart Card ToolKit are used to initialize and personalize the smart cards.

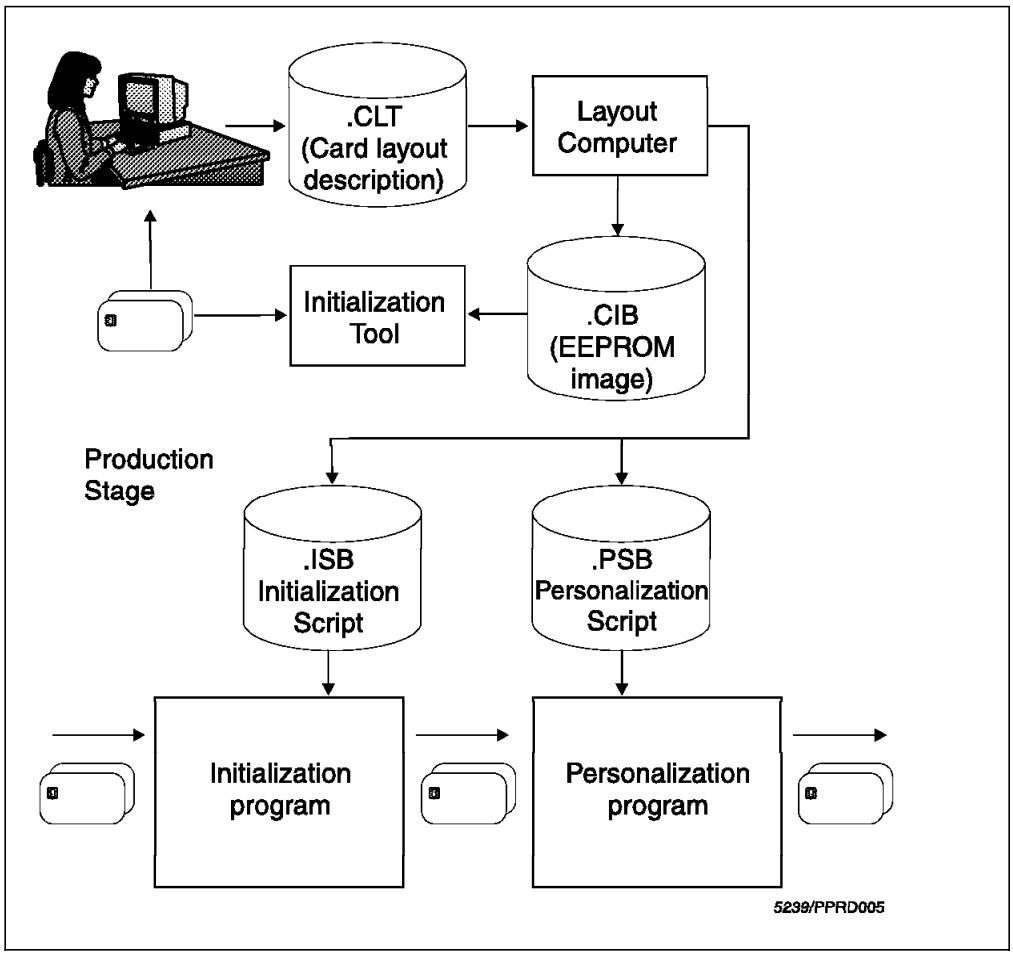


Figure 30. Card Production Process Using the IBM Smart Card ToolKit Scripts

For more information about the personalization process and file information, please refer to *IBM Smart Card ToolKit Personalization Guide*, SC33-7004.

9.3 Card Production Security

We mentioned at the beginning the importance of security during card production. This is implemented with keys which are introduced at various points during the card production. See Figure 28 on page 113.

For example, the cardholder's data is protected using a personalization key when the data is sent to the personalization site. Sensitive data may be encrypted and other data may have a code (MAC) appended to verify the authenticity. The smart card in turn has this same key loaded during the initialization so that the data can be unpacked (decrypted) securely into the EEPROM storage by the chip, ensuring that no one can compromise the data during the personalization step.

In order to understand the security strategy, we will describe some of the characteristics of the passwords and keys used during smart card production.

ROM or Chip Password

The card initialization is a major change to the EEPROM structure and such changes should not be permitted without authorization. The first step in the card initialization is to erase the EEPROM. In some chip

cards, a re-initialization is not allowed but with IBM MFC cards it is permitted. The authorization for this process is that you need to know the ROM password and you need to put the value in the card layout.

Transport Keys

Transport keys are used to protect the file layout during file transfer between different sites. These keys are then used to verify the authenticity of the files before any initialization or personalization.

Master Keys

To guarantee absolute security, the keys used should never be divulged to any one person or even kept in a readable form. The keys are generated by the hardware at a secure computer terminal and the output is split into two or three key parts. (There are special cryptographic adapters which provide this key generation facility.)

The key parts are locked in a safe according to the organization's security procedures and whenever a key is required, the key parts are given to nominated security officers who must separately enter them into the required device or system. The system then merges the key parts in order to work out the master key.

The following example shows the master key split into two key parts. You should find that by using the exclusive OR operation on the two key parts, you will get the master key.

```
Master Key (Hex)  = 96 34 02 8F 3D 77 20 B3
Key Part 1 (Hex)  = 87 2E DD 16 98 72 A7 2C
Key Part 2 (Hex)  = 11 1A DF 99 A5 05 87 9F
```

Derived Keys

The exposure of a key used in smart cards is one that creates a nightmare scenario for any card issuer. But this can be easily circumvented by the use of derived keys. With derived keys, each cardholder's smart card carries a unique set of keys. One method of achieving this is to use a unique identifier from the cardholder's data, such as the card serial number, account number or personnel number and encrypt it using the DES algorithm.

The example below shows the derived key for a cardholder that uses the card serial number:

```
DES Key (Hex)     = 96 34 02 8F 3D 77 20 B3
Serial No (Hex)   = 00 00 00 01 23 45 67 89
ICV (Hex)         = 00 00 00 00 00 00 00 00
Derived Key (Hex) = 93 0A 1D 0F 6A 2A 35 84
```

The ICV (Initial Chaining Vector) is a parameter used by the DES encryption process.

Chapter 10. Smart Card Programming

Smart card programming involves interfacing with two devices: the smart card and the card reader. The majority of toolkits available for smart card programming in the market today simply provide a graphical interface to assist in the capturing of the low-level card commands for later inclusion in the application. As you will see, this kind of development approach is not suited for a multivendor card and reader environment and is very limiting in the scope and flexibility it offers.

We examine the programming issues of using the card commands, which are a set of rather low-level commands, and the problem of having to interface with many different card readers. Once a smart card application is developed, different card issuers adding their applications to the smart card only complicates matters for the developer.

Then we explain how the task of programming has not only been made easier, leading to faster and lower development cost, but also provides vendor independence, helping developers to protect their investment and enabling them participate in the rapidly changing smart card market.

Where appropriate, we have used sample code to demonstrate the cardholder′s name (from our GTT case study) being read from the smart card.

10.1 The Card Instructions

The smart card communications is established through a set of very simple instructions. The procedure requires some basic instructions to the card to select a particular file where the data is (by specifying the path name to the file) and then reading the data. The card operating system always checks the protection attributes of the data and will either reply with the data or request some authorization such as a PIN to verify the card user. Once the smart card has checked the correct PIN entry, the requested data is sent with the next reply.

If we take an example of reading the cardholder′s last name (″Smith″), which is in the file ′A100′ off the Master File ′3F00′, the following sequence of instructions is given to the card.

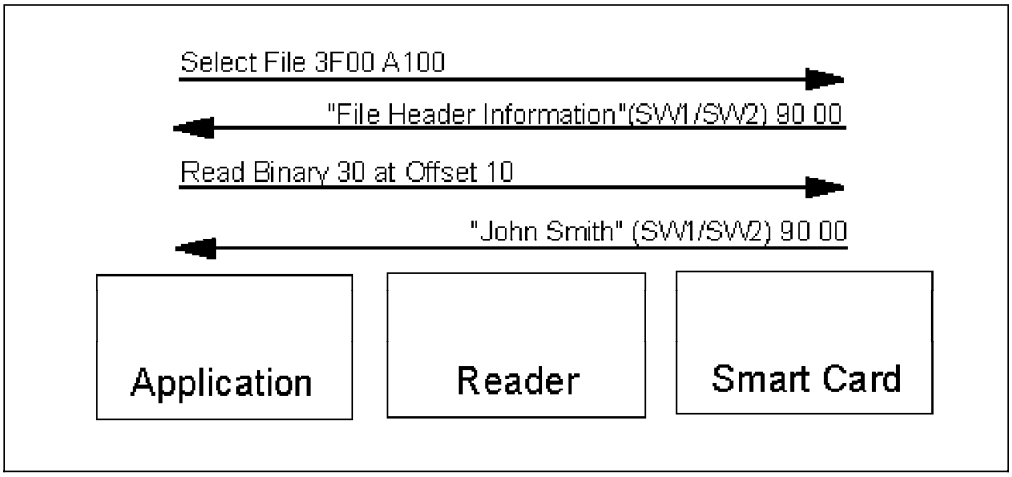


Figure 31. Reading Data from a Smart Card

However, the actual instructions must be coded in hexadecimal (specified by ISO 7816-4) and is called an APDU (Application Protocol Data Unit). The APDU bytes are in a recognizable pattern, which will become very familiar as ″CLA INS P1 P2 Lc Data″ (Class, Instruction, Parameter 1, 2 ,Length and Data).

Note: For more information regarding APDU commands, please refer to Table 11 on page 137.

The above card instructions to read the name, translated to APDU commands is shown in Figure 32. According to the ISO 7816 standard, the hexadecimal instruction code for Select File is ′A4′ and for Read Binary it′s ′B0′.

```
                CLA INS P1  P2  P3  <--- Path --->
SELECT FILE     00  A4  08  00  04  3F  00  A0  04

                                    <---> Data Length
READ BINARY     00  B0  00  00  09
```

Figure 32. Sample APDU Commands from the Application to Smart Card

The response from the smart card is equal to the task, resulting in the following string of bytes for the application programmer to decode.

Note: The return code from smart cards is always at the end of the response in two bytes, which are known as status words (SW1 and SW2). A successful outcome is indicated with the status code ′90 00′.

```
SELECT FILE Response
     ===> 63 0C 00 20 A0 04 00 00 03 3F FF C3 01 00 90 00

READ BINARY Response
     ===> 53 6D 69 74 68 00 00 00 00 90 00
           S  m  i  t  h
```

Figure 33. Sample Response from the Smart Card.

It is usual to program these types of commands by first reading through the smart card command reference obtained from the operating system manufacturer and then writing a wrapper around the command syntax. In theory it should be possible to code using the APDU commands from the ISO standards, but very often there are unique implementations of APDUs by the vendors. The following example illustrates an occurrence of this deviation.

```
Vendor A
 SELECT FILE     00  A4  08  00  04  3F  00  A0  04

Vendor B
 SELECT FILE     20  A4  08  00  04  3F  00  A0  04
```

When programming for these two vendor′s cards, the application has to be made aware of the cards′ operating system (for example, by analyzing the ATR) and

two sets of commands have to be coded. This costs time and effort to the application developer each time a new vendor's card is introduced.

The card instruction must now be sent via a card terminal API, to handle the communication link between the card and the application. An example for a card reader API is shown here:

```
Open the reader interface
   rc = Reader Open ( port );
        .
        .
Send data to the card via the reader interface
   rc = Reader Command ( data send, data receive);
        .
        .
Close the reader interface
   rc = Reader Close ( );
```

Figure 34. Sample Commands to the Card Terminal

The "APDU_send" contains the command and data to the card and the "Receive_Buffer" is filled with the reply from the card. The CTN, DAD, SAD are control information used by the card terminal. Sequences of calls to CT_DATA accomplish other tasks, such applying power to the card or displaying characters on the card terminal display.

These APIs would normally be included in a library (DLL) supplied by the reader manufacturer. However, each reader usually comes with a unique set of APIs. This costs time and effort to the application developer each time a new vendor's card or reader is introduced.

10.2 A Better Approach to Coding Smart Card Applications

There are other drawbacks of coding smart cards and readers by directly interfacing with the devices. In the next few sections, we have listed some other reasons why a better approach is needed for smart card programming.

10.2.1 Card Operating System Dependencies

- An intimate knowledge of the card commands is required
- A lot of time is spent assembling the card commands
- Inability to handle smart cards from different manufacturers

  You cannot always guarantee that a particular APDU command for one smart card works with all cards. There may be subtle differences in the use of the parameters P1, P2 etc. between manufacturers.
- Precise information regarding the smart card data structures must be known to the application
- The application has to keep track of how much data has been read and repeat read/write requests to handle large data items

  The APDU buffer is of finite length and large data items have to be handled with repeated calls using offsets.

- The application has to be modified each time changes are made in the card data structure (for example changes in data field lengths, locations).
- It is time consuming to test the card commands because the application needs to be recompiled each time a command is added

10.2.2 Card Terminal Dependencies

- Different reader manufacturers may have different APIs

  This requires coding separate modules in the application in order to work with different readers.

- Readers cannot be added or removed without changes to the application code
- The application can lock out all the other applications requiring access to the reader

  The second application thinks the reader is not available rather detecting the reader is busy. Each application has to perform an inefficient sequence of initializing and closing the reader before allowing other applications to access the reader.

- The application needs to know the reader features in advance

  The reader may have specific features related to its use, such as a display LCD, audible sounds, keypad, two card slots and indicator LEDs, which ties in the application to a specific reader type.

- Inability to queue requests to the card reader

  This is related to multiple applications accessing a reader.

- Different card detection mechanisms

  Reader drivers may have functions such as informing the application of a card insertion whereas other may need the application to poll the reader.

- Different PIN verification mechanisms
- Some readers may have the facility to verify the PIN within the terminal microcode using the built-in keypad whereas other may require the application to retrieve the PIN from the cardholder.

10.2.3 Card Issuer Dependencies

When cards are issued to customers, the card issuer decides where to place the card-resident applications on the smart card. The same card-resident application might end up in different places on cards distributed by different card issuers.

This means the developer faces the problems of:

- Listing all the applications supported by a smart card
- Locating and selecting an application in a smart card
- Installing and uninstalling applications in a smart card
- Blocking or unblocking applications in a smart card

10.2.4 Solution to the Programming Problems

The issues highlighted above have not gone unnoticed in the industry and have resulted in some innovative concepts and techniques.

In particular these techniques free the developer from having to:

- Know the APDU commands
- Wade through smart card return codes
- Know the details of the smart card layout
- Create lookup tables for smart card data elements
- Maintain position of data read/write offsets within a file
- Write code to handle multivendor cards
- Write code to handle different readers and APIs
- Write control code to handle reader access
- Write code to handle more than one reader type
- Write code for polling readers for card insertion/removal

The following tools/specifications attempt to address these problems:

- IBM Smart Card ToolKit:
  - Smart Card Agent
  - Smart Card Script Processor
  - Smart Card Agency
- PC/SC
- OpenCard Framework (OCF)
- JavaCard[5]

10.3 The IBM Smart Card ToolKit

This section covers the different IBM Smart Card ToolKit components and how they help smart card programmers.

10.3.1 The Smart Card Agent

The Smart Card Agent is an application programming interface (API) designed to free the developer from having to know:

- Where the data is stored in the smart card
- What card commands are needed
- How the data is accessed by card commands

Note: The Smart Card Agent is provided by the IBM Smart Card ToolKit.

[5] JavaCard is not a technique but we show how it manages to avoid some of the difficulties we have highlighted.

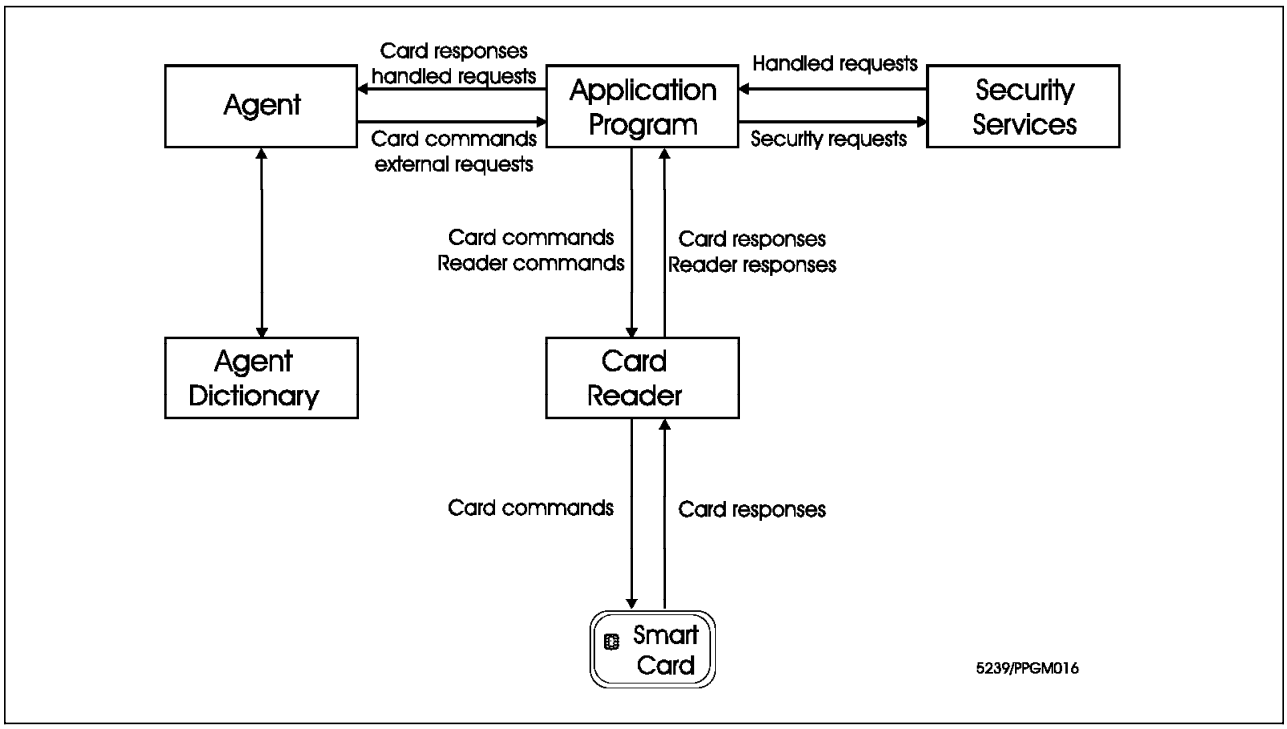


Figure 35. The Smart Card Agent Architecture

Earlier we saw the hexadecimal byte stream (APDU), which needs to be assembled in order to access the card. By delegating the complex task of assembling APDUs to the Agent, the card access logic is drastically simplified and the code is shielded against any changes to the card file and data structure.

It does this by storing the knowledge of the card layout in a Layout Dictionary and the card commands in a small Agent Library (DLL). Since the dictionary also contains information required for accessing the files (for example, the file paths, offsets and sizes), it frees the programmer from a tremendous amount of work having to code this knowledge in the application. When the application wants a data element from the card, it uses an *Alias Name* to look up the dictionary using a simple high level *Agent Command*.

If the card layout changes, only the data definitions change and the Agent and the application is not affected. The dictionary is automatically produced by the IBM Smart Card ToolKit layout compiler, which is described in Chapter 8, “Card Layout Design” on page 87.

The Agent offers much more than just helping with the card commands. It can offer a consulting facility to the application relieving the developer from analyzing the card return codes and having to figure out what needs to be done next.

For example, if the smart card responds with an authentication request, the Agent supplies the appropriate parameters in a buffer area reserved for its use and informs the application program what needs to be done next. This may be a requirement to encrypt a piece of data or prompt the user with a password request. The developer can leave the tedious task of handling the card to the Agent and spend more time concentrating on other parts of the application.

Agents are available for several different smart card operating systems, but have the same application programming interface. The application program retains control over the transmission of card commands to the card and any cryptographic processing.

An example of using the Agent to read the cardholder data is shown in Figure 35 on page 124. Notice that the application no longer needs to assemble SELECT FILE and READ BINARY commands (see Figure 32 on page 120) since these are now prepared by the Agent. When the smart card replies back with a response, the application can invoke the Consult Agent command to get assistance with the next action needed to perform. Many of the tasks necessary for the developer to code have been efficiently implemented by the Agent.

```
Main Program Body

1  SctCreateAgent(...);
2  SctFile2Buffer("GTT.DIB",....);
3  SctAgent(SCT_CMD_READ, "FirstName", ...);
   SctAgent(SCT_CMD_READ, "LastName", ...);
   SctAgent(SCT_CMD_READ, "PersonnelNumber", ...);
   .
   .
   SctAgent(SCT_CMD_UPDATE, "DoctorName", ...);

4  SctDeleteAgent(...);

Agent Consulting Procedure

SctAgent (...)
{
5    SctInstructAgent(...);

     do {
6.         rc = SctConsultAgent(...)
           switch (rc)
           {

7.                  case SCT_RC_CARD_REQUEST:
                   .
8.                  case SCT_RC_IDENTIFICATION_REQUEST:
                   .
               case SCT_RC_AUTHENTICATION_REQUEST:
               case SCT_RC_SIGNATURE_REQUEST:
                   .
           }
     } while(rc);
 }
```

Figure 36. Sample of Agent Commands

Notes:

1 Creating the Agent is done first to reserve work areas and prepare for subsequent calls.

2 The Agent needs to read the dictionary and key tables into work areas.

3 Accessing the data (Read/Update) is done using and SctAgent call. Note that the SctAgent is a procedure implemented as a do-while loop.

4 Deleting the Agent is done at the end to perform a cleanup.

5 The Agent is first Informed of the new card request.

6 Consulting the Agent gives a return code (rc) which tells the application what to do next. If an APDU command is required, the Agent looks up the dictionary and assembles the required instructions most appropriate for accessing the data.

7 An SC_RC_CARD_REQUEST tells the application that the command should be sent to the card reader.

8 An SC_RC_IDENTIFICATION_REQUEST signals the application to prompt the user for an authorization (for example PIN/Password). Other return codes asks the application to provide the necessary logic such as authentication and encryption.

Smart Card Learning and Development Phase

The Agent can also be used as a learning tool to help with understanding the smart card APDU command flow. During the development phase, the Agent can also help you to quickly test out some card command sequences that may be required by your program. Both of these are provided by using the Agent in conjunction with the Script Processor tool (SCTEXEC), which is provided in the IBM Smart Card ToolKit. (Hint: Use the trace option, /T in SCTEXEC).

For more information about the Agent please refer to Table 11 on page 137. We will describe the Script Processor in the next section.

10.3.2 Script Processor

The Agent API or any other card API still requires code to be developed in the application. A way to avoid this coding step is provided by the script processor.

The script processor, as it name implies, executes instructions given in a file (a script). Its like an interpreter, which gives a very fast way of testing smart card commands and sequences of commands without needing to write and compile a program, using the normal practice of code development. For example, this could be used to test a smart card command sequence and check the trace output before writing any code, saving hours of debugging, recompiling and testing effort. When the commands are working as expected, they can be coded in the program.

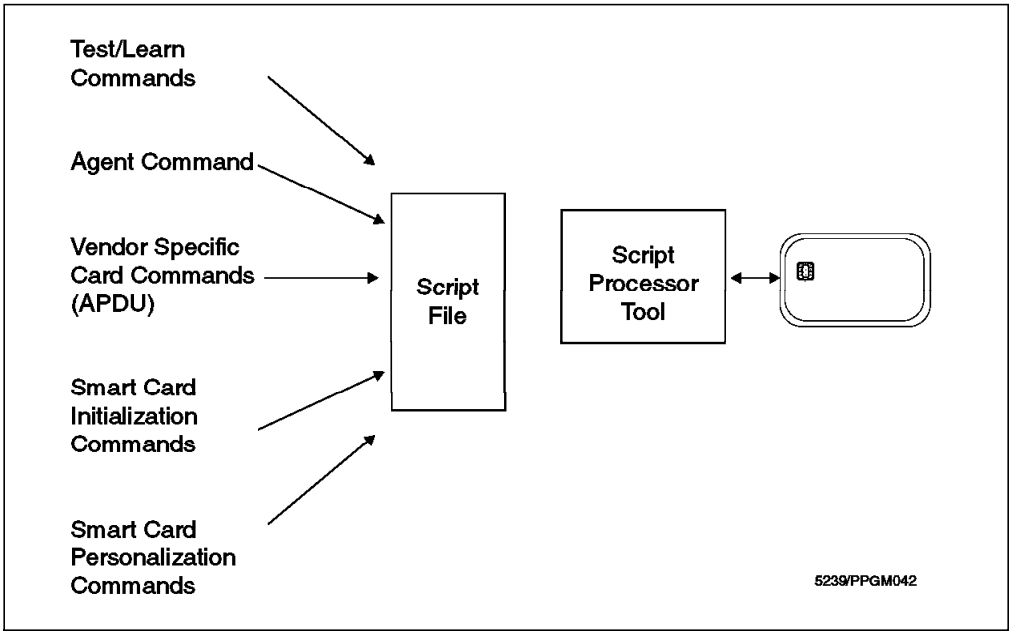


Figure 37. The Script Processing Tool

Sometimes a simple tool for making life easy becomes such a versatile instrument that its applications reach far beyond the original intention. The script processor qualifies for this description. It is now integrated in the IBM Smart Card ToolKit and it can be used to handle anything from testing a few card commands to carrying out a full-scale smart card production.

We mentioned that the aim of a script file was to test card commands and later code the equivalent in your programs. There is no need to bother with that step anymore. You can just have the application call the script processor to execute the command and hand over the required data.

The story of scripts gets better. If a different vendor's card appears in the fuure, you can avoid any changes to your application and just call up a different script file. Calling up the appropriate Agent to handle different cards is what the Agency is all about. The Agency can manage the scripts for you and do a few more nice things along the way, which we describe in the next section.

The power of the script processor has also been incorporated into the OpenCard Framework architecture, which is described later in 10.5, "OpenCard Framework" on page 133.

The following is an example of a script file procedure (named ″READ″) used to read the cardholder's name. We have included comments in the sample code to describe the function of the script commands.

```
SCRIPT
(
   DESCRIPTION
   (
     # -- Some initial environment handling statements
     # -- Note the pointer to the Card Layout Dictionary "GTT"
      COMMENT( "Red Book Example Script GetName.SCB" )
      CARD_OS( COS_MFC_V421 )
      VERSION( 1 )
      CARD_LAYOUT( "GTT" )
   )
   PROCEDURE
   (
     # -- Read Card Holder Name --
     # -- This uses the Dictionary alias name "LastName"
     #    and writes the card data to a variable called "Name"
     ID( "READ")
     AGENT_REQUEST
     (
         PARAMETER( AGENT_COMMAND( CMD_READ )
         ITEM_ALIAS("LastName"))
         RESULT
         (
            BUFFER_ENTRY("Name")
         )
     )
     # -- Now, we can export the variable for the application
     #    to use. In this instance, the "putchar" is a command
     #    in the Script Processor which can print the data.
     EXPORT_REQUEST
     (
          PARAMETER( ID("putchar"))
          ARGUMENT
          (
             CONSTANT_PART("Hello Mr/Ms ")
             BUFFER_PART( ID("Name") )
             CONSTANT_PART("\n")
          )
     )
   )
```

The script processor can:

- Test card Agent API
- Test card commands (APDU)
- Handle different vendors cards
- Initialize cards (using layout compiler output .ISB)
- Personalize cards (using layout compiler output .PSB)

For more information about the Script Processor, please refer to the IBM Smart Card ToolKit manuals listed in Table 11 on page 137. Information about card initialization and personalization is given in Chapter 9, “Smart Card Production” on page 111.

10.3.3 The Smart Card Agency

The Agent we described handles a card and its operating system. Different versions of card operating systems from the same manufacturer would be handled by different Agents. The concept of Agents can be extended to different manufacturers, for example, a Gemplus Agent can be used to handle the Gemplus MPCOS card and an MFC 4.1 Agent can be used to handle an IBM MFC 4.1 card.

The Smart Card Agency takes the same high level interface of the Agent but enables the managing of these different Agents.

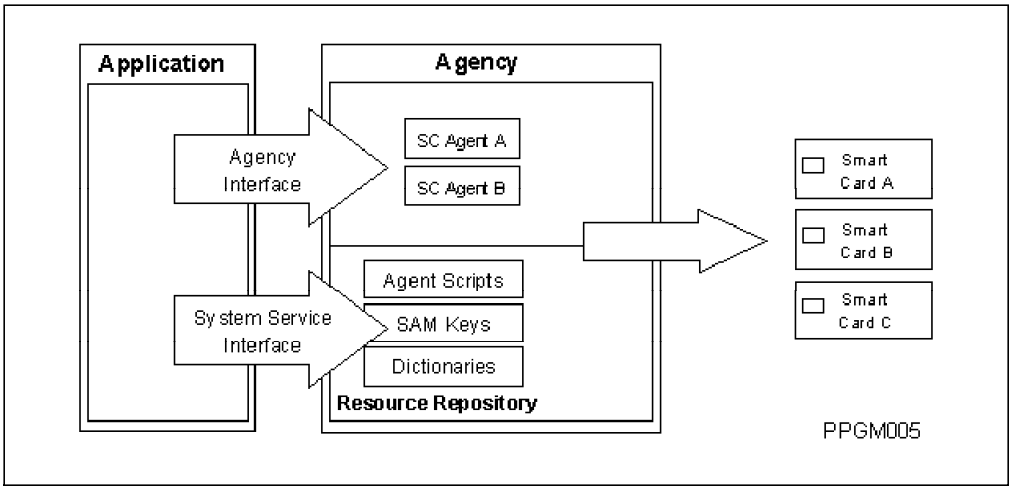


Figure 38. Smart Card Agency

This is how the Agency operates. When a smart card establishes contact with a card terminal, it first responds with the ATR. Since the format of the ATR is more or less standardized, we can collect some parameters to work from any manufacturer's card.

The Agency has a data base (repository), which contains a list of ATRs and a selection of a Card Agent is made for the application based on this ATR. All available local resources (script files, dictionaries, key tables, Agent modules) are registered in the repository.

The Agent operation requires an Agent module (specific to the card) and a dictionary (with the card layout definitions). The Agency makes a selection for these two components using one of the following methods:

The Card Agent is selected using:

- Full ATR (See 8.3.1, "Answer to Reset (ATR)" on page 96)
- Card historical data (from the ATR)
- Operating system version

The Card Dictionary is selected using:

- Required card application name (for example "Loyalty")
- Operating system version

We can dynamically create and delete the entries in this repository using the Agency system services interface. If a new card is introduced to the system, a corresponding entry is added in the repository. To avoid duplications, we can simply place a pointer in one repository to point to another repository. This is

very similar in concept to the Internet making network connections. In a networked environment, all the connected systems would have an Agency.

Let's take the example of reading the cardholder's name from the GTT smart card. All the information required by the application to access the smart card is created in the repository as shown in Figure 39. There are two entries:

1. An "Order" that is invoked by the application to execute procedure "READ" from a script file.
2. The GTT card ATR that tells the Agency which resources the smart card requires.

```
REPOSITORY
(
   DESCRIPTION
   (
       COMMENT("Sample Repository  SamplRep.SCB")
       VERSION(1)
   )
   .
   .
1. REPOSITORY_ENTRY
   (
       ORDER("Welcome")
       PROCEDURE_CALL("READ")
       SCRIPT(RESOURCE("GetName.SCB"\X00))
   )
   .
   .
2. REPOSITORY_ENTRY
   (
       ANSWER_TO_RESET('3B......')
       MODULE(RESOURCE("SCTMFC42.DLL\X00"))
       DICTIONARY( RESOURCE ("GTT.DIB\X00"))
       KEY_TABLE(RESOURCE ("GTT.KTB"))
   )
```

Figure 39. Sample Agency Repository

The Agency operates as follows:

1. The order (Welcome) is logged in the repository. The order specifies the procedure ("READ") to use from the Script file ("GetName.SCB").

2. When a card is inserted, the Agency locates the resources it requires by using this Repository Entry. In this example the ATR ('3B...') is matched with the ATR received from the smart card and the indicated Agent module, dictionary and key table is loaded dynamically. The Agency frees your application from managing this resource.

Once the resources have been loaded by the Agency the order "Welcome" requested by the application is fulfilled.

The Agency can be invoked by your application program or with the SCTEXEC tool.

```
SCTEXEC Welcome  SamplRep.SCB
```

10.3.3.1 Supporting Multivendor Cards

In the above example, the application works fine with vendor A′s card who has got sophisticated tools to generate a smart card dictionary and has card agents.

Supposing we have to link vendor B′s card, who doesn′t have the luxuries of dictionaries and card Agents and doesn′t have enough time and resource to start creating these tools. The Agency offers a helping hand here because it can use scripts directly to issue the APDU commands. In order to handle vendor B′s card, we can compile the instructions required and place them in a different script file. When the Agency detects vendor B′s card (from the ATR), it uses the indicated script file and satisfies the request from the application. The application doesn′t have to know that it is dealing with a different card with a totally different operating system.

The cards need not necessarily support a similar set of ISO standard card commands. The Agency has been successfully deployed where the cards have the MPCOS, STARCOS, ME2000 and MFC operating systems. The Agency is available for a variety of programming languages including C, Java and Visual Basic.

For more information about the Agency please refer to the IBM Smart Card ToolKit manuals listed in Table 11 on page 137.

10.4 PC/SC (Personal Computer/Smart Card) Interface

The above techniques relates to the card and the application program. Programming the application and the reader interface must also be considered in the same way.

The solution to this is to manage the readers separately from the application(s) and provide a common reader API interface. Reader vendors in turn can conform to this API enabling them to readily enter the market.

This is the PC/SC Interface Specification (V1.0) published by the PC/SC Workgroup. More details about this workgroup and the specification is given in 3.5, “PC/SC” on page 32. At the time of writing, PC/SC only addresses the PC Windows 9x/NT 32-bit platforms (PC/SC requires multithreading). Support for 16-bit Windows is not available and would depend on future customer demand.

The architecture of PC/SC is shown in Figure 40 on page 132.

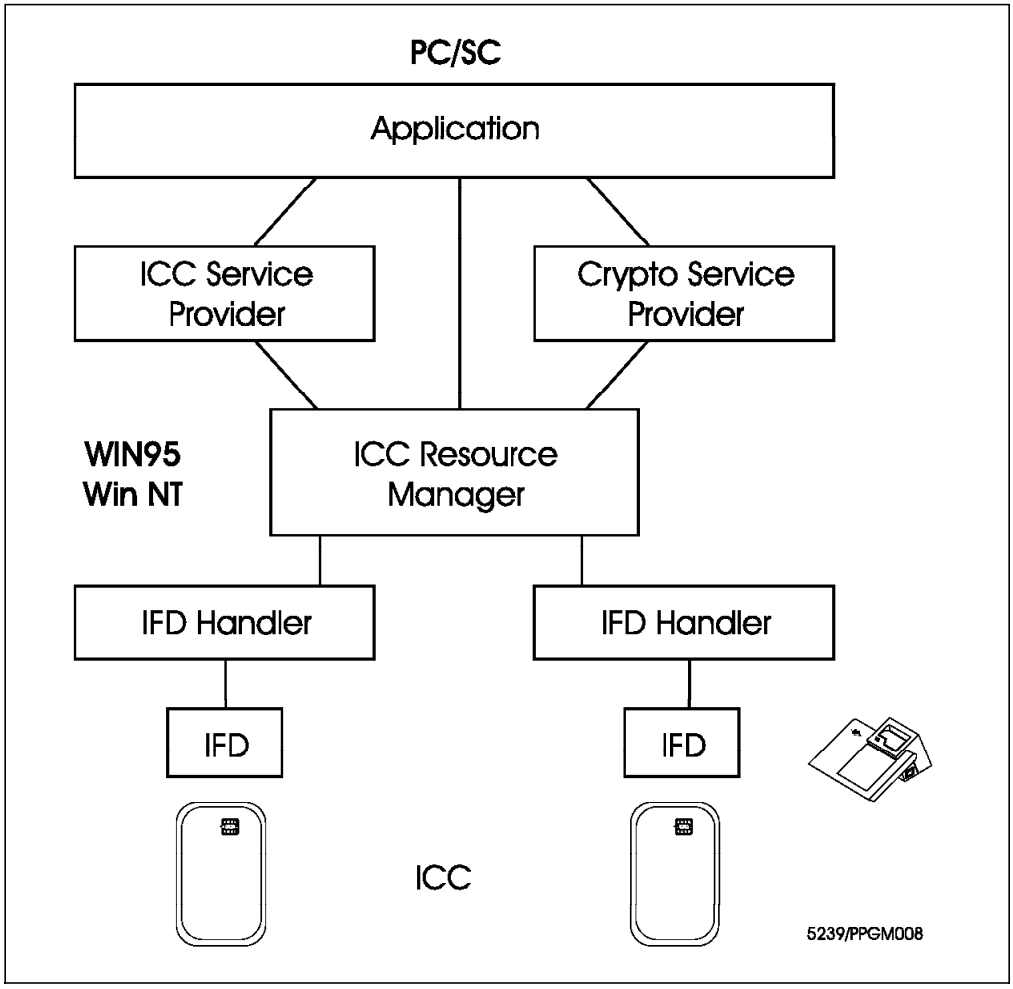


Figure 40. PC/SC Architecture

The PC/SC consists of a Resource Manager, which is responsible for all the readers attached to the workstation, and Service Providers, which are responsible for providing a high level common interface to the smart cards and cryptographic functions. The reader vendors would provide the device drivers (IFD Handlers) in the diagram and the operating system would supply the Service Providers with their smart cards. At the time of writing, PC/SC is strongly associated with the Resource Manager, the reader management interface. The Service Provider Interface is available as a recommended way in which to code the high-level API for software vendors and developers to implement. The Service Provider is dispatched using the ATR (See 8.3.1, “Answer to Reset (ATR)” on page 96).

The reader interfacing problems are solved by the PC/SC specification by providing:

- A high-level common reader programming interface for the application developer
- The programming specification for vendors to conform, in order to construct PC/SC compatible reader device drivers

The following example illustrates the commands used in the PC/SC Interface code:

SCardEstablishContext: command issued by your application to the PC/SC interface, establishes a context for subsequent calls to the PC/SC Resource Manager.

```
SCardEstablishContext(SCARD_SCOPE_USER, ......, hContext);
```

SCardListReaders: command provides a list of available readers within a group.

```
SCardListReaders(hContext, Groups, Readers,...);
```

SCardConnect: command establishes a link to the card in the selected reader. If no card is present this is passed back through a return code.

```
SCardConnect(hContext, ReaderName, SCARD_SHARE_EXCLUSIVE,
                        SCARD_PROTOCOL_T1, hCard,.......);
```

SCardGetAttrib: gets the reader attributes, for example the reader vendor's name.

```
SCardGetAttrib(hCard, SCARD_ATTR_VENDOR_NAME,.....);
```

SCardGetAttrib: gets the card's attributes, in this instance the card's ATR response.

```
SCardGetAttrib(hCard, SCARD_ATR_STRING,.....);
```

SCardTransmit: is an example where you might want to issue a raw APDU command to the card.

```
SCardTransmit(hCard, MY_SCARD, SELECT_3FOO, .....);
```

SCardDisconnect: terminates a previously connected card.

```
SCardDisconnect(hCard, SCARD_EJECT_CARD);
```

SCardReleaseContext: closes an established context to the Resource Manager, freeing up any resources allocated under that context.

```
SCardReleaseContext(hContext);
```

There are many other commands available. For more details, see Table 11 on page 137 for PC/SC reference information and software development kit.

The PC/SC Interface allows a developer to handle any PC/SC compliant reader without the need to include reader-specific APIs.

10.5 OpenCard Framework

The next question is whether we can improve things further for the developer by providing a consistent interface, not only to the card operating system, reader and cryptographic interfaces, but also to the applications on the smart card card.

We could wrap up everything nicely if our application could be freed from having to keep checking for a particular card insertion or trying to monitor what's happenning with the card reader.

Well, you've guessed right. This is the OpenCard Framework.

The OpenCard Framework hides the complexities of smart card programming from the user by clearly encapsulating the technological components. The OpenCard Framework is coded in Java; its target platforms are network computers, Web browsers, or any future platform that runs Java and has to interact with smart cards.

More information about the OpenCard Framework is described in Appendix D, “Overview of OpenCard Framework” on page 185 and information on programming is listed in Table 11 on page 137.

The two most important layers of the OpenCard Framework are:

- The CardService (handles different card operating systems and allows sharing of the smart card)
- And the CardTerminal component (handles different readers; for instance, what to do when a card with a particular ATR is inserted)

All the components in OCF are implemented as Java class libraries. For example, the CardService package is *opencard.core.service* class library and the CardTerminal package is *opencard.core.terminal* class library.

These packages have specific interfaces to supply functions for the application developer. For example, to read the cardholder′s name, we need access to the smart card′s file system, which is provided by the *FileSystemCardService* interface (CardService package). Access to the card reader is provided by the CardTerminalRegistry interface (CardTerminal package), which knows about the addition and removal of card terminals..

The starting point for most applications is perhaps the *SmartCard* class, which presents the physical smart card to the application developer. OCF supports both the application-driven model, for example waitForCard() or the event-driven model, for example, by implementing the event listener interface.

If the event-driven model is used, the application informs the CardTerminalRegistry that it is interested in receiving information about cards being inserted and uses the method addCTListener() method of the CardTerminalRegistry. If the application-driven model is used, the application passes along a CardRequest and specifies what kind of card, which card terminal, whether to use an already inserted card, and even what kinds of CardServices the card should support.

With these basic OCF services, we have written a sample program; see Appendix E, “Sample OCF Code - GTTEmployeeCard.java.” on page 191.

10.6 Programming JavaCards

The traditional smart card applications as you have seen, require knowledge about the data structure stored inside the cards. The techniques we have described in this chapter frees the application programmer from having to maintain this knowledge, which is a step in the right direction. The JavaCard takes this to its ultimate conclusion and places the intelligence about the data inside the smart card itself.

```
Smart Card (Traditional) ---> Please select MF, DF, EF and
                              read "Name" from EF at Offset n.


Smart Card (Java Card)   ---> "Employee" Applet please
                                 give me the "Name".
```

The beauty of developing applications for the JavaCard is that you can do so in your existing development platform. There are already sophisticated application development tools for creating and debugging Java and indeed JavaCard applets from many leading vendors such as Borland, IBM, Microsoft and Sun.

JavaCard applets are written in a similar way to the Java applets which runs on a PC, but due to the limited memory and computing power of a JavaCard only a small subset of the language features are supported. The applets have to be tested and debugged rigorously before being loaded in to the JavaCard and for this reason the testing is done using a JavaCard simulator first.

An example of a JavaCard applet which might be coded to handle the cardholder's name in our case study is given in Appendix C, "Sample JavaCard Applet" on page 183.

The advantage of JavaCard from the application developer point of view is the seamless interface, which means there is no need to learn a new environment.

For more information on JavaCard, see Appendix B, "JavaCard Overview" on page 179. For more programming information, please refer to Table 11 on page 137.

10.7 JavaCard versus OpenCard Framework

Many times a question is asked: What is the difference between OpenCard Framework and JavaCard?

OpenCard Framework is Java in the computer or terminal talking to the smart card. JavaCard is a special, stripped-down version of Java that runs on the Smart Card itself. Java applications running on the PC can use OpenCard to access JavaCard smart cards and standard smart cards. If you want to write a Java applet that should run on the smart cards itself, also know as Cardlets, you have to use a smart card that is compliant to the JavaCard standard.

Another frequently asked question is: Are there any overlaps between JavaCard and OpenCard?

The answer is no. OpenCard Framework and JavaCard fit perfectly together. OpenCard is the ideal host-side application framework for accessing your JavaCard. As for any other Smart Card you need a Card Service, which is supporting the interfaces of your JavaCard applet, to access your Java Card.

10.8 Smart Card Simulators and Testing Tools

Smart Card simulators provide a way for developers or interested parties to learn about smart card and work with the smart card APDU interface without the need for a smart card or card terminal.

The software is installed in the PC and a graphical interface may be provided to create the card file structure and data elements using drag and drop. Data entry fields for hexadecimal numbers allow you to try out and test card instructions using the raw APDU commands.

This type of tool is also frequently supplied by the smart card manufacturers to enable you to create scripts and test their smart cards. A source of references for this type of tool is listed in Table 11 on page 137.

10.9 Programming Languages

Smart card applications are developed using commonly used programming languages such as C, C++. The applications developed using these languages can base the logic on the techniques we described in this chapter.

When programming, consideration must also be given to the availability of specific reader device drivers for the platform, in which the application is required to run on. For instance, the reader may be PC/SC compliant but the PC/SC interface specification at the time of writing is not available for the Windows 3.x 16-bit environment. In this case, the feasibility interfacing with 16-bit device drivers will need to be checked before starting development.

When developing applications for NCs or Web browsers, or using the Java language, the smart card interfacing requirements would naturally point towards using the OpenCard Framework.

10.10 Positioning of Smart Card Programming Methods

The following is an attempt to position the programming methods mentioned in this chapter. We excluded JavaCard since this method refers to programming in the card, while the other techniques aim to facilitate the access to the card.

Table 10. Smart Card Programming Requirement vs Methods

Programming Requirement	IBM Smart Card ToolKit	PC/SC	OCF
Freedom from card commands	√		√
Freedom from card layout	√		√
Freedom from card data structures	√		√
Multi-vendor card support	√		√
Multi-vendor reader support	√	√	√
Multiple reader support	√	√	√
Multi-slot reader support	√		√
Card application management			√

10.11 Smart Card Programming Information

In order to start using the smart card programming techniques in your application we have prepared the following table, which gives you information about where to obtain the IBM Smart Card ToolKit and other information currently available.

Table 11. Smart Card Programming Reference Information

Smart Card Programming Information / Tools	**References**
IBM Smart Card ToolKit • Layout Compiler • Agent • Agency • Script processor • Initialization Tool	IBM SC Home Page - www.chipcard.ibm.com Manuals GC33-7000-01 - Smart Card ToolKit Overview SC33-7001-01 - Smart Card ToolKit Programmer's Reference SC33-7002-01 - Smart Card ToolKit Tools Reference SC33-7003-01 - Smart Card ToolKit User's Guide SC33-7004-01 - Smart Card ToolKit Personalization Guide IBM Multi-Function Card Programmer's Reference • Copies of this book may be ordered via INTERNET: smart@de.ibm.com IBM VNET: SMART at BOEVM4 Mail: IBM Deutchland Entwicklung GmbH Department 7828/7103-03 Postfach 1380 71003 Boeblingen Germany
PC/SC Specification SDK	Workgroup - www.smartcardsys.com/ Microsoft Smart Card SDK MS SC Home Page - www.microsoft.com/smartcard/ Cryptography - www.microsoft.com/security/ MS Dev. Network - www.microsoft.com/msdn/ SmartCardSDK@Discuss.microsoft.com
JavaCard	Sun's JavaCard for Beginners www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html Sun's Products and API download page www.javasoft.com/products/index.html Sun's JavaCard Page java.sun.com/products/javacard/ GemXPresso www.gemplus.com/gemxpresso/index.htm JavaCard Forum www.javacardforum.org
OpenCard Framework	OCF Home Page - www.opencard.org •Architecture Document •White Paper •Programmers Guide •Stock Broker Demo •OCF Reference Implementation Version 1.0
EMV Specification	www.visa.com www.mastercard.com (site search keyword EMV)
APDU Commands	ISO 7816 (www.ansi.org, look in ISO/IEC Catalog search keyword 7816) Card Operating System suppliers EMV, IBM MFC Programming Reference
Smart Card Simulators Smart Card Toolkits	www.scdk.com/scdkptrs.htm

Chapter 11. Project Management

A smart card project is no different from any other IT project. It requires good project management. In this section, we first discuss in general good project management practice, and then we focus on the planning and implementation of the GTT employee card project.

Since the smart card is a new technology, most solutions regarding a smart card implementation will be evolutionary rather than revolutionary.

The scale of the marketing effort varies from case to case. For our case study of an employee card, very little marketing is required, while in the case of an open electronic purse issued by a financial company, marketing can be key factor.

Besides the smart card and smart card acceptance devices, the challenge is in the system integration area.

In general, for a smart card project, the following sequence is recommended:

1. Have a business case in place to justify the expenses, resources, commitment, and investments
2. Obtain an executive sponsorship to endorse the project
3. Hold a joint application development (JAD) session with customer(s) to obtain the detailed requirements and agreed-on implementation alternatives
4. Identify resources
5. Design the overall technical system solution
6. Develop a plan for development and implementation
7. Develop a detailed task list and schedules
8. Pilot a small scale implementation to validate the assumptions, designs, interfaces, and logistics
9. Develop information packages and procedures in marketing and logistic support for the project
10. Roll out the project
11. Get feedbacks and measure the success

11.1 Business Case

As a technology, smart card is now being employed in many areas. However, before a smart card can be a part of a product or solution there are questions that need to be answered. For example:

- Why use a smart card?
- What value does the smart card provide?
- What problems will the smart card solve?
- Will the smart card reduce the operational cost?
- What kind of resources and investment are needed?
- Will the use of smart cards improve the image of the company?

Depending on the business areas, the answers to these questions will vary.

While doing the benefit and cost analysis of using a smart card, we need to be aware that even if most benefits are quantifiable, some may not be. For example, a business case may be built on a non-quantitative value, such as company′s image as a technical leader in the industry, so that the company gains a larger market share. The customer needs to make the final decision.

11.2 Executive Sponsorship

With a good business case, the next step is to secure executive sponsorship. A project has a much better chance of success with a high level executive sponsoring it because the executive can:

- Allocate and approve the funding for the project
- Sell and market the project to the board or higher level of corporate officials
- Set up a strategic and visionary road map for the project
- Set up strategic alliances for the project
- Identify critical success factors

11.3 Requirements Analysis

We must get high-level requirements from customer's meetings or from the customer's RFPs. At the time we prepare an RFP response, we will take the high-level requirements and provide a broad brushed diagram of a draft solution.

Once we have a firm commitment (such as a funding commitment or contract) from the customer, then we will do a Joint Application Development (JAD) for more detailed requirements. The customer's input is needed for making decisions. For example, we will discuss with the customer about the pros and cons of using different standards (such as PS/SC, EMV, etc), the most appropriate operating system etc. Getting the customer involved and informed in the decision-making process for implementation at each phase of the project increases the chances of its success.

For our hypothetical company, in Chapter 7, "Analysis of the Case Study Solution" on page 75 we provide an example of a simple requirement analysis and system design.

11.4 Resource Identification

With a clearer understanding of the detailed requirements we can identify the resources and skills required for the project. An end-to-end smart card project requires skills in many areas such as smart card, acceptance devices, network, databases, and industry operational knowledge and expertise. Depending on the smart card selected, we need the developers to learn and be proficient in the toolkit for the selected smart card. By the same token, the developer also needs to be knowledgeable about the devices which are enabled for handling smart card.

A list required skills follows:

- Project management
- System solution architect
- Programming skill (C, C++, JAVA, VisualAge, etc.)
- Smart card development toolkit
- Database specialist
- Network specialist
- Card management/issuance production
- Marketing
- Training specialist

11.5 Phased Implementation Plan

All the applications cannot be rolled out simultaneously; a phased implementation must be established, based on GTT's priorities and what can be reasonably achieved considering equipment and skills availability. We established a three-phase implementation plan. The starting date of each phase will have to be determined in accordance with the availability of card terminals, skills, application development times, etc.

The use of the new smart card as a badge identifier is set from the very beginning, with the distribution of the cards to the employees. There is no difference between the old badge and the new smart card badge in this respect. All the information printed is the same as in the old badge. On the back side, we use the same magnetic stripe for the building access, cafeteria payment, and time and attendance.

The first application on the chip is the digital signature for e-mail. GTT wants this application to be implemented in phase 1. This should not be a major problem since it does not require any development; GTT will use IBM's Digital Signature Solution (DSS). The main tasks will be to sign a contract with the Certificating Authority (CA), education of the employees on installation, and usage of the card and procurement and installation of the smart card readers.

Health alert is another application for the first phase. Some development is required. Some new card terminals will have to be installed for this application.

In phase 2, we implement some more functions on the employee card:

- The e-purse for payments at the company's cafeteria. The cash loading machines and the new cafeteria's point of sales (POS) will be installed. Programming development is required.
- The fingerprint identification system for access to the secure areas. For this application, we need to install both data capturing and special card terminals. Programming development is required.
- Contactless facility access. For this application we need to install the contactless smart card readers. Little or no programming development is required.

We leave the network access application for phase 3, which requires further studies.

Table 12. GTT Phased Implementation Plan

Applications	**Phase 1**	**Phase 2**	**Phase 3**
Facility Access	Current System	Contactless	Contactless
Cafeteria	Current System	E-Purse	E-Purse
Digital signature	IBM DSS	IBM DSS	IBM DSS
Network Access	--	--	yes
Health Alerts	yes	yes	yes
Biometric	--	yes	yes
Time/Attendance	Current System	Current System	Contactless

11.6 Detailed Tasks Breakdown

A project plan with detailed tasks is beyond the scope of this book. We are only giving an example of very high level tasks as follows:

- Employee Card Design
  - Plastic art work design: location of GTT logo, work location, employee photo, name, employee number, badge number, and employee type (regular employee, contractor, etc.).
  - Smart card application layout design and testing
- Employee database enhancement for digital photo
- Acquiring or developing digital photo data capturing and processing system
- Card management and issuance development
- Testing technical development
- Deployment logistics planning and scheduling
  - Card distribution logistics
  - Equipment installation
  - System integration
- Alliance with a CA
- Download keys and certificate from CA and testing with e-mail
- Biometric system acquiring and development
- Development and testing for health alert application
- Legal aspect, contracts, and licenses
- Information material development
- Validating process and procedure
- Training
- Feedback and measurement

Each of these major tasks can be broken down into subtasks. Resources need to be assigned to each task with an associated schedule.

11.6.1 Pilot Implementation

A smart card project starts with a small-scale pilot to test all the hypotheses, assumptions, issues, components, interfaces and the whole system. When we are satisfied with the results against the success criteria, we will execute the rollout implementation plan.

Table 12 on page 141 shows the general phased implementation. For each phase, we need to do a pilot implementation. We will discuss with GTT to select the best location for a pilot.

Since the first application on the chip is the digital signature, GTT needs to have a relationship with a CA.

During the pilot stage, the card issuance process needs to be fully tested. Please refer to Chapter 13, “System Integration and Testing” on page 157 for details on testing.

Integration and testing with the current system for facility access, cafeteria payment, and time and attendance is another task to be done during the pilot. The process of certificates from CA and signing the mail with the keys and certificate should be checked to make sure the process and procedure is readily available.

Preparing a user marketing and information package for employees is another task that can be performed during the pilot.

11.6.2 Rollout

After a successful pilot, the rollout is more or less a large-scale pilot. However, distribution of hardware and software including installation must be carefully planned and monitored. Mass employee card production and distribution to all locations will be carried out. All the infrastructure should be functional and in place before we launch and roll out the project. Again, all the supporting material, user documentation, etc., should be ready for distribution online or via hard copy.

Celebration of milestone accomplishments such as end of a pilot and at the end of rollout should be also conducted to give encouragement on the continuation of the project.

11.7 Feedback Survey and Installation and Deployment

We need to get feedback of what went right and what went wrong in the project. Past experiences are good for follow-on projects or new projects. A careful design of the questionnaire for feedback or interviews is essential to get the most realistic feedback.

Chapter 12. Card Management

This chapter describes the basic concepts of a Smart Card Management System (SCMS or CMS) and how GTT plans to implement such a system.

Briefly, CMS is the management of the life cycle of a smart card, from issuance to expiration of it, including card replacement in case of loss, theft or physical damage.

The main differences between traditional magnetic stripe card management systems and smart card management systems are outlined below.

The CMS described here implements one of several different approaches to managing the life cycle of a smart card. You will find that different vendors will accomplish the same tasks described in this chapter in a different way or sequence.

The CMS that we describe in this chapter is rather simple, compared, for example, with a CMS of a financial institution that has to deal with millions of cards, different application providers, etc.

12.1 Magnetic Stripe Card Management

GTT has been dealing with magnetic stripe cards for many years. GTT has a structure in place to issue, replace and revoke employees′ cards. GTT wants to be prepared for smart cards now. The main difference between managing magnetic stripe cards and smart cards is that the data stored in a magnetic stripe card rarely changes throughout the whole life of the card, while in a smart card it can be constantly changing. So the challenge is how to replace a lost, stolen or damaged smart card with another that has similar content.[6]

In summary, there is no need for administrative functions that keep track of the content of the magnetic stripe during its life cycle.

The magnetic stripe card can get lost or stolen, or get damaged. In the first case (lost or stolen), the terminals, for instance GTT′s access control readers, need to be programmed in such a way that hot lists can be used to hold the card numbers of all lost and stolen cards, or the readers have to be able to connect to host systems to check on the current version of the hot list. It is then part of the reader application to react to such a lost or stolen card in a predefined way.

Another part of traditional magnetic stripe card management is card issuing, for instance personalizing the plastic and the magnetic stripe, as well as reissuing and expiration.

[6] The magnetic stripe content may change, for example during an ATM transaction, but no attempt will usually be made to replace a card with another that reflects those changes.

12.2 Smart Card Management

Multi-applications smart cards are characterized by holding concurrently several applications which can be provided by different companies or application providers.[7] We can, therefore, distinguish between card issuers and application issuers. In many cases one organization will act in both capacities.

CMS systems assist these application issuers and the card issuer in managing the following objects:

- The smart cards itself
- The applications on the cards
- Cryptographic keys that are stored on the card
- Certificates that are stored on the card

We can divide smart card management into three logical areas:

1. Card management is primarily addressing the need of the card issuer to manage the entire card base and individual cards through their life cycle, for example, adding another cardholder or adding another card to a cardholder or suspending a cardholder from using the card.
2. Application management is supporting the application issuer in managing the entire life cycle of application-specific data on the card. As the data in the smart card is changing, it needs to be managed.

   GTT will need application management, as we will explain later.
3. Key management allows card issuers and application issuers to manage and generate all card- and application-specific cryptographic keys through the entire life cycle of the card and applications. This includes building application keys for individual applications, loading keys on cards, etc. Key Management Systems (KMS) could be operated by trusted third parties (TTP).

[7] This is not GTT's case, since all the applications will be issued in-house.

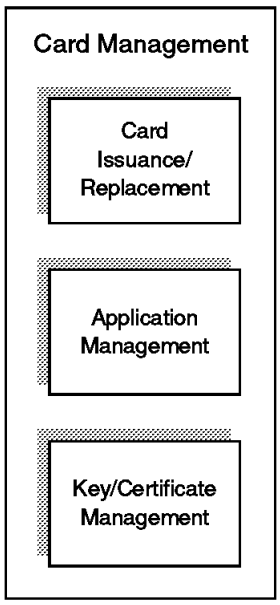


Figure 41. Card Management System Components

As we said earlier, the real difference between magnetic stripe and smart card management systems is the need for online administration of the card, its applications and its keys.

12.3 Card Issuance

Life cycle management begins with the issuing of a card to a cardholder; steps include the initialization of modules and the personalization of the chip and plastic.

The card issuance process process varies widely depending on the number of cards, issuer (financial institution, enterprise, etc). We describe one possible approach, the one adopted by GTT.

The card production process is described in general in Chapter 9, “Smart Card Production” on page 111. In this chapter we describe GTT′s specific approach.

We have to keep in mind that there is a rather small number of smart cards involved in the case study (around 10,000), that the only value involved will be for the cafeteria closed purse, etc. In a case of a great number of cards from a financial institution, more stringent security measures should be taken.

Figure 42 on page 148 shows what activities will be performed at the card manufacturer, card initialization, and card personalization sites, and those carried out at GTT.

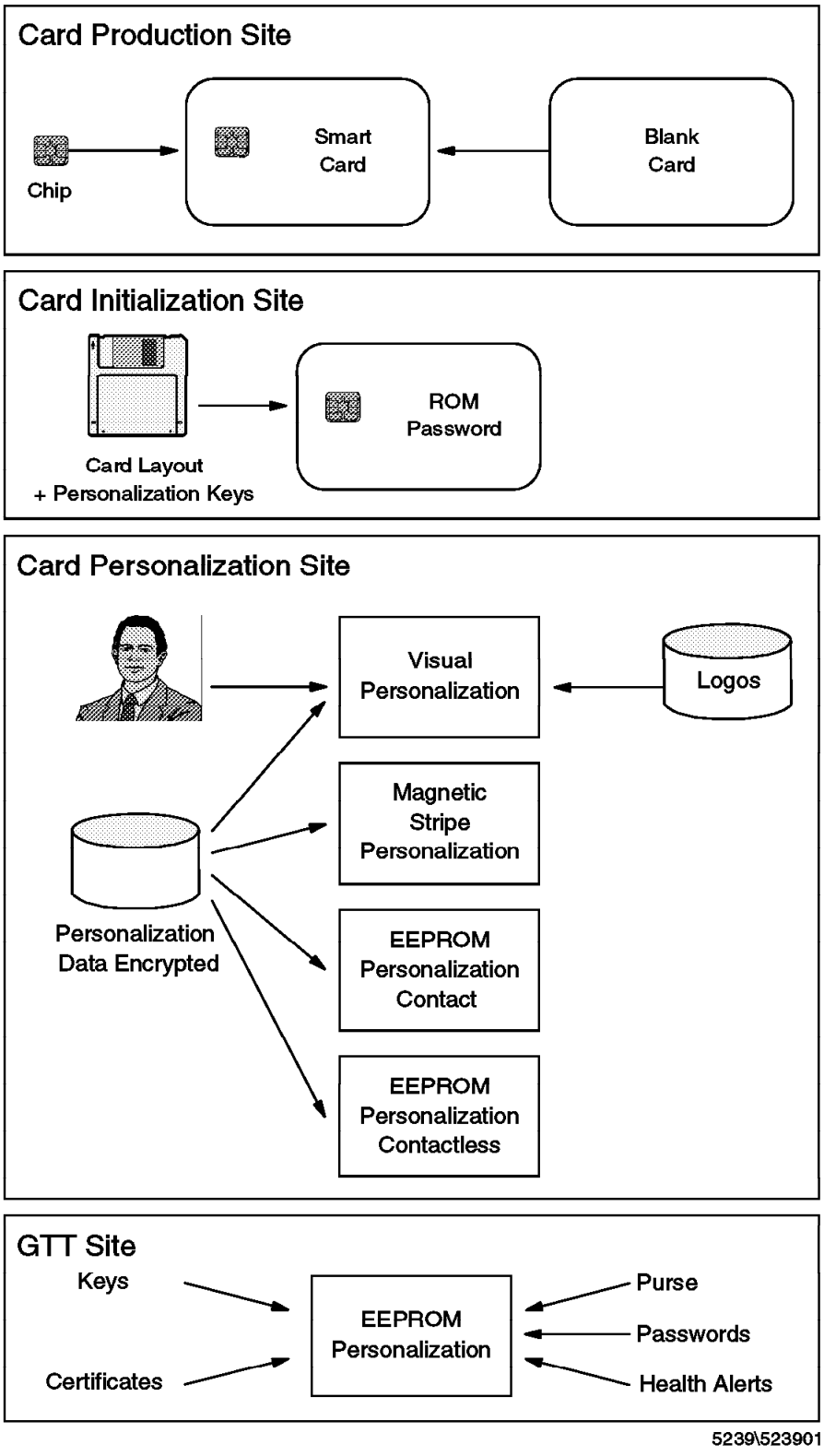


Figure 42. Card Production

We have seen in Chapter 8, “Card Layout Design” on page 87 how the card layout is produced. This card layout contains password(s) that will be used during the personalization process (the personalization data is sent encrypted).

GTT will compress the layout using PKZIP or WINZIP, and assign a password to the zipped file. This file will be stored in a diskette together with a program that can do a checksum of a file. GTT will run this program against the file containing the layout before compressing it, and keep a record of the checksum.

This diskette will be carried to the initialization site by personnel from GTT's security.

At the initialization site, the layout file will be unzipped, and the program that calculates the checksum will be run. This checksum value will be communicated to GTT; if it coincides with the one GTT has, it means that nobody has tampered with the layout file and the initialization process can start.

At a later stage, GTT will send this data to the initialization site online, using, for example, PGP for protecting the privacy of the data.

During the initialization process, the file structure is set and the personalization key(s) are transmitted to the card. The ROM password shown in Figure 42 on page 148 is used to erase the EEPROM before writing data to it, as explained in 9.3, "Card Production Security" on page 116.

This approach has the disadvantage that all the cards will have the same personalization key(s). A better approach would have been to use derived keys (see 8.3.5, "Derived Keys" on page 99 and 9.3, "Card Production Security" on page 116). The derived keys would be generated using the card ID number, which is unique, plus the key(s) included in the layout. GTT may take this approach in a future generation of cards.

Once the cards have been initialized, they are sent to the personalization site. GTT will send to the personalization site the data that has to be printed/written to the card.

This is the most delicate part of the operation; the personal data has to be well protected. In some countries there are some legal implications in handling personal data like photos.

The personalization order will consist of four TLV structures:

1. Printed data: the data that will be visible on the card, such as name, serial number, photo. The photo will be in JPEG format; to give an idea of size, a photo of 200x240 pixels with 16 bits colors can take between 5KB to 8KB.
2. The data for the EEPROM, such as name, serial number, etc. This data will be encrypted using DES with a key of 56 bits. This key is the same as what already is stored in the card after the initialization.
3. The data for the magnetic stripe. The magnetic stripe will use only two of the three tracks. The first track will have the employee number plus an expiration date; this is the data used to pay in the cafeterias.

   The second track will contain a badge number that the actual system uses for access control. This data will not be encrypted.
4. The data for the contactless memory. This is basically the same as the data for the magnetic stripe.

There will also be an index to associate this card with a specific logo. GTT will send a separate file containing the JPEGs of the different logos, with an associated index number.

In order to have a secure transport of these files from GTT to the personalization site, GTT will contract an ISDN line. This line will be set up in such a way that:

- GTT will establish the call
- GTT will be requested to enter a user ID and a password
- The ISDN side at the personalization site will only accept calls from GTT's phone number

Once the personalization orders are complete, GTT will encrypt the whole file using PGP and send it to the personalization site through the ISDN line.

The creation of the personalization orders can be a complex task. GTT will have to merge information from different systems: from the control access system it will be able to match serial number with badge numbers; from the personal department it will get data to match names with badge numbers. Finally the photos will have to be added.

GTT will use a DB2 database to store all this information.

This effort will have two stages: first when all of the cards are produced, and second to keep it up to date, for new hires, lost cards, etc.

A diagram of the proposed solution is shown in Figure 43.

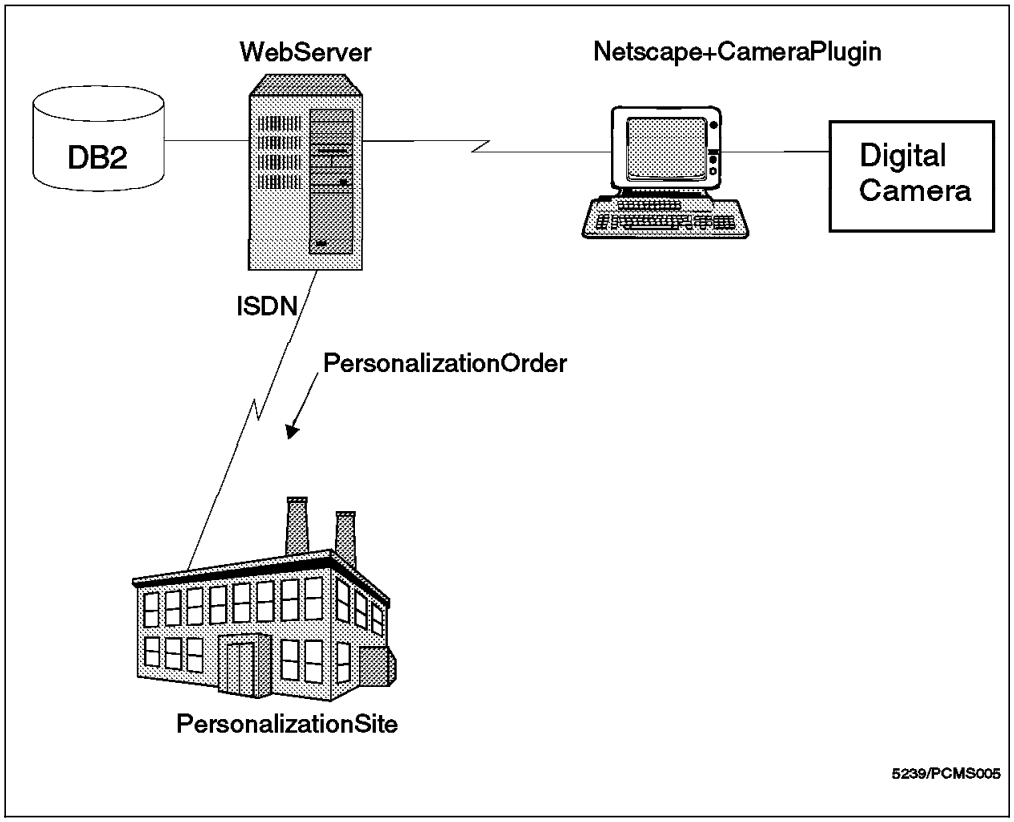


Figure 43. GTT Card Personalization Structure

When a new card is needed, the person in charge of this task will point the Netscape browser to the Web server. The session will be SSL encrypted. All the required data will be entered in at the Netscape station. A picture of the employee will be taken with a digital camera. All this information will be sent to the server.

The server will be located in GTT Germany; the Netscape stations will be located in the different countries where GTT operates. The personalization data site is located in Germany.

After the first batch of cards is produced, the server will send personalization orders to the provider on a weekly basis.

GTT will establish a process to quickly deliver the cards destined to the USA and Malaysia.

GTT will coordinate with the smart card provider to handle appropriately the special characters in some employee's names. It is not expected to be a problem since the cards will be manufactured in Germany, where the special characters for names will appear. Malaysia will use the English character set.

The GTT network for card personalization will look like Figure 44 on page 152:

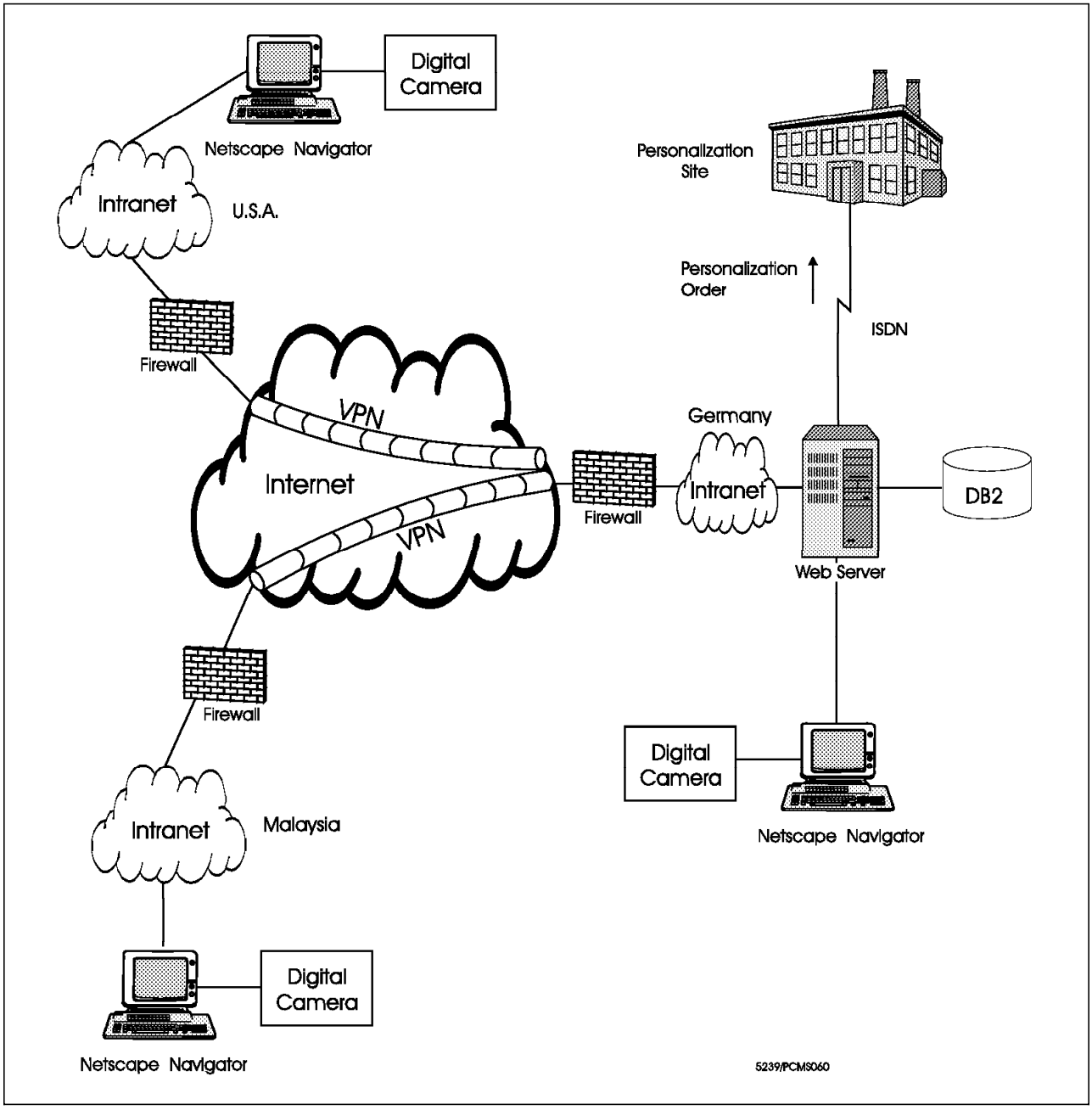


Figure 44. GTT Network for Card Personalization

GTT will define Virtual Private Networks (VPN) to ensure that the transmission of the personalization data is secure. In a VPN the data is encrypted and the keys are changed on a periodic basis. Of course, these VPNs will be used for other purposes too.

As mentioned in Chapter 6, “Case Study” on page 69, GTT may want to issue temporary cards for some of its visitors. These are devices that allow the issuance of smart cards in low volumes. Some vendors of such machines are Datacard Corporation (www.datacard.com), Schlumberger (http://www.slb.com/ms/et/sc_offer/offer_for.html#personalization), CIM from Italy and ADT from Germany. These devices can be attached to a PC to enter the data, including pictures; they do all the personalization of the smart card, including printing and EEPROM writing. It can take up to several minutes to

personalize a single card. GTT will have to decide whether the cost of these devices justifies its acquisition.

You should consider not only the cost of acquiring these machines, but also the cost of maintenance, which could also be high.

12.4 Loading Application Data into the Card

GTT will load the application data into the EEPROM in house, except for the data mentioned above. Let's see how GTT plans to proceed with the different applications.

12.4.1 Digital Signature

Only a fraction of GTT's employees will use digital signature for e-mail. These employees are going to receive a PCMCIA or serial smart card reader. It will be the employee's responsibility to update Netscape Messenger to generate the key pair, get a certificate and handle digital signature with a smart card.

GTT will set up an internal HTTP or FTP site (one in each country) from where the employees will be able to download:

1. The Netscape cryptographic module that handles smart card digital signature
2. Smart card reader driver
3. Instructions

Since GTT will have to get a license for each use of the cryptographic module, the employees that will use it will be given a user ID and password to access the FTP site.

It was discussed whether to go ahead with an FTP site or to give the employees a diskette with the cryptographic module. It was decided in favor of the FTP site approach, so that if an employee is in a business trip and cannot use his/her laptop for any reason, or does not take the laptop with him/her, it would be possible to use any other laptop with Netscape Messenger installed to quickly set up the smart card reader and the cryptographic module. The employee will only need to carry his smart card.

GTT will contract with an Internet Service Provider that allows access to GTT's intranet from the Internet, from any of the countries GTT's employees travel to.

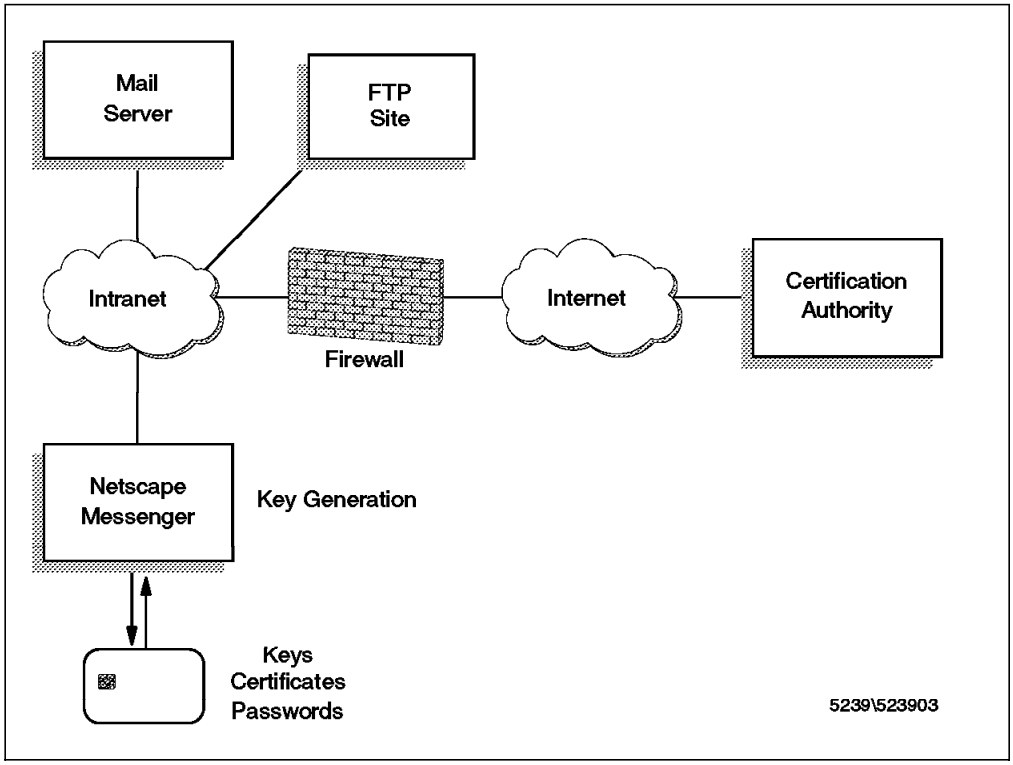


Figure 45. Digital Signature Personalization

Once the employee has installed the reader and the software, he/she will be requested to:

- Create a password for accessing the digital signature files in the smart card.
- Generate the keys. At a first stage, the pair of keys will be generated by Netscape; in the future, they may be generated by the smart card.
- Get a certificate from the CA selected by GTT.
- Store password, certificate and keys in the smart card.

Once this is done, the personalization of the smart card related to the digital signature application is finished.

Note:

This approach of having the employee load the keys and certificates into the smart card is only applicable in cases like GTT's, because:

1. Just a small fraction of the smart cards will need this data.
2. The employees that will perform the task are highly qualified.

In most cases, the best approach is to have the keys and certificates loaded during card issuance. The main advantage is that the load will be done in a trusted environment; a second advantage, not to be underestimated, is that the delicate task of loading the keys and certificates is taken away from the average employee.

In this case there are several approaches:

- GTT would generate the keys, request the certificates and send all this data to the personalization site

- A CA will send this info to the personalization site
- The personalization site will generate the keys and request the certificates
- The keys could be generated by the smart card; this is a very secure but slow process.

See *IBM Digital Signature Solution: Concepts and Implementation*, SG24-5283 (available at a later date) for more details.

12.4.2 Cafeteria Purse

At each cafeteria there will be one or more special terminals that will have a card reader and a slot that accepts money (paper bills or coins). See Figure 16 on page 59 for an example of such a terminal. The smart card purse will be loaded when the employee uses the terminals available at the cafeterias. These terminals will accept paper bills and write the equivalent amount in the card. The employees will also use these terminals to check the balance in the purse. The balance will be displayed in two currencies in Malaysia and Germany: in local currency and in dollars.

12.4.3 Health Alerts

The employees that opt for storing their medical information in the card will do in one of two ways: in their own PCs when they have a card reader installed, or in the terminals installed by GTT for that purpose. These terminals will be located in private areas.

The health alerts PC program will be downloadable from an intranet FTP site.

12.4.4 Biometrics

Only a few employees will need to load their digitized fingerprints in the card. GTT will install special terminals for scanning the fingerprints and storing that data in the card.

12.5 Card Replacement

As mentioned at the beginning of this chapter, the main difference between managing magnetic stripe cards and smart cards is that the data stored in the smart card changes almost every time the card is used. Fortunately for GTT, this is not the case for most of the applications. The only application that changes the smart card EEPROM is the cafeteria purse.

Let's see how GTT plans to act when a card is lost, stolen or damaged.

- Facility access: if the card is lost or stolen, GTT will immediately update the hot list to deny access to the premises if someone intends to use the card.
- Cafeteria purse: GTT will make it clear to the employees that if the card is lost or stolen it is like losing cash. No attempt will be made to try to reconstruct the balance at the time of the loss. GTT will explain to its employees that this is the way that banks running purse schemes proceed when a card is lost.

  GTT assumes that this procedure will be an incentive for the employees to take good care of the card.

- Digital signature: the employee will have to execute the same procedure that he or she used to get the certificates the first time.

  GTT will communicate to the CA the loss of the certificate. GTT will have to establish a policy establishing how the employee will request the new certificate, probably using a security officer.

- Biometrics: The employee will have to proceed as when it was first loaded.

GTT will request a new card from the card issuer; the data required for the new card is the same as that in the original personalization order. GTT may establish a policy that if the employee's photo on record has more than a certain number of years, a new photo will have to be taken.

12.6 New Cards

There are two distinctive cases:

- A new hire

  A new personalization order will have to be created for this case, based on the existing card layout. The new order will be sent to the card supplier and the employee will complete the card personalization as described above.

- An existing employee must change the card logo

  For example, the employee changes location. The personalization order is already available; it will have to be modified to point to the new logo.

  A terminal that accepts two cards will be installed to allow the transfer of the balance of the cafeteria purse from the old card to the new one.

12.7 Card Revocation

The revoked card will be entered in the hot list. If the card is in the hot list, certain readers could retain the card when somebody tries to use it.

If the owner of the card was using digital signature, the CA will be contacted to invalidate the certificate stored on the card.

Chapter 13. System Integration and Testing

Testing is always an important part of a project. In this chapter, we first provide a general summary about the task of testing within a project. Then we will focus on various testing tasks for the GTT employee card project.

As stated in Chapter 11, “Project Management” on page 139, a smart card project is likely to be evolutionary rather than revolutionary. The challenge is in integrating the different components into a system that will perform according to the requirements. The solution involves new developments, as well as the integration of acquired commercial off-the-shelf systems and existing systems.

For the new developments, testing starts at component level. For acquired systems, the testing focuses on the installation and integration of these systems into the total solution. For the existing systems, regression tests must be performed to make sure the new developments will work with the existing systems.

As security is one of the main issues around smart cards, testing should not only check that everything works as designed, but also that no known method can be used by a hacker to commit an illegal act.

Testing is also an iterative process. Test results reporting and problem fixes tracking are all part of the testing process. If there are any fixes, changes, additions of a new piece of hardware or software, then regression tests need to be conducted to make sure no unexpected effects have been introduced by the changes.

In most cases, the developer of the component will test the design and its implementation. The integration and system tests are performed by a different set of people. The integration and system tests take the total system as a whole from the requirements′ point of view. Test scenarios and test cases are generated from the customer′s requirements.

For a smart card project, smart cards and terminals or readers are the two basic new elements. Next are the modifications or enhancements to the existing systems for handling the smart cards. Thus the component testing is performed on the smart cards, readers and the part of the existing system which is changing.

Tests for usability, acceptability, and value proposition of the smart card also need to be performed. A successful project is based not only on its technical merit, but also on the user′s acceptance of the solution. A smart card is acceptable if it is easy to use and provides convenience and value.

Test plans for functional and system testing need to be developed. An outline of such a test plan is presented here as an example. An actual test plan is out of the scope of this redbook.

- Test objectives
- Test assumptions, dependencies, and scope
- Test methodology and responsibility
- Test scenarios and test cases development, entry and exit criteria
- Test execution
- Test reports -- problem and status

- Regression test and status tracking

13.1 Component Testing

For the GTT employee card, the components are as follows:

- Smart card file structure design
- Terminals/readers, and scanners
- Smart card production and issuance
- Digital photo data capturing and processing
- Employee data base enhancements
- Fingerprint data capturing and processing
- Fingerprint door access system
- Health alert data entry and processing
- Several tasks related to digital signature

13.1.1 Smart Card Testing

Although all applications on the chip are not implemented in phase 1, the smart card file structure layout design needs to plan for all the applications. The goal is to initialize the smart card's EEPROM at card production time with place holders for all applications. The place holder will then be populated with data in the future when the infrastructure is ready for these applications.

Once the file structure is designed, test cards need to be produced for testing. A smart card application developer usually needs to make a few iterations before the file structure is finalized. For GTT, we will use IBM Smart Card ToolKit for designing the smart card file structure layout, generating test cards, testing the cards, and generating initialization and personalization scripts for card production.

In Chapter 8, "Card Layout Design" on page 87 and Chapter 10, "Smart Card Programming" on page 119, this process is explained. As shown in those chapters, the IBM Smart Card ToolKit script processor can be used to test the card design.

13.1.2 Terminals/Readers and Scanners Testing

For GTT, we have selected standard off-the-shelf terminals/readers and scanners. The handling and integration of the devices resides in the applications of the system. The testing of these devices is limited to its basic functions as advertised. The smart card readers attached to PCs need to be PC/SC Migration compliant.

13.1.3 Card Production and Issuance Process

As stated in Chapter 9, "Smart Card Production" on page 111 and Chapter 12, "Card Management" on page 145 mass smart card production and issuance are security sensitive processes. It has to be tested that no security holes are left unchecked before sending the smart card data to either the initialization or personalization sites.

13.1.4 Digital Photo Data Capturing and Processing

In order to print the employee photo on the card for visual identification purposes, we need to capture the employee photo and store it digitally in the employee database. Many such systems are available in the market for this process. Digital cameras attached to PCs will need to be installed.

The procedure and the process of capturing the photo, such as having the employee come to the security office for photos during certain office hours, should be planned and tested. For example, if it takes a minute to take a picture of an employee, then the office hours should planned so there will not be a long queue of employees waiting for photo taking.

The photo format (number of pixels and storage format) must be matched with the card printing process.

13.1.5 Employee Database Enhancements

Depending on the GTT′s employee database, we can add the digital photo as part of the database, or we can set up a separate database for photos and then link them at the time we do the card production order.

13.1.6 Fingerprint Data Capturing and Processing

A fingerprint capturing and recording station will be set up at the office of company′s security officer. The employee who requires the fingerprint on the card will go to the security office to register and also to load the fingerprint data onto the employee card.

There are several such systems on the market. The installation of this system, the operation process and the training material for using this system all need to be tested and verified.

13.1.7 Fingerprint Door Access System

A small PC-controlled door system with both smart card reader and fingerprint scanner will be installed at the door. This is probably the most security-sensitive application for GTT. The installation and the operation of this system needs to be tested.

13.1.8 Health Alert Data Entry and Processing

The GTT employee smart card is designed with spaces for the employee to input their health alert information. A data entry program will be developed for employees to enter their health alert information and write on the smart card. Testing of this program and its ability to write data onto the employee′s smart card needs to be performed as well, so that it cannot access other areas of the card′s EEPROM other than the one assigned to this application.

13.1.9 Digital Signature

Those employees who need to sign/verify e-mail will load the public/private keys and the certificate into the smart card. The connectivity to the Internet/Intranet needs to be set up and tested. A version of the browser needs to be loaded and customized in the workstation. The Netscape Communicator V4 or above needs to be installed and customized to use the IBM Digital Signature Solution. An installation procedures document and a user document for employees who need to install this package must be developed and verified.

13.2 Pilot System Integration Testing

A pilot is a small-scale implementation of the whole system. Thus during the pilot stage, all the functionalities need to be tested and verified as a system, not as components.

During the pilot, a portion of the employees will receive the employee card. Components and devices will be installed and deployed at pilot location. The integration system test needs to be conducted before the launch of the pilot.

13.2.1 Functional Integration Test

The following is a list of functions and processes which need to be tested:

- The card ordering process: from the first card issuance to the card loss and re-placement
- The employee photo is visible (in good quality) as picture ID
- The interface to the existing facility access system using the magnetic stripe
- The interface to the existing cafeteria payment system using the magnetic stripe
- The interface with the existing time and attendance system
- The smart card reader device drivers are integrated and communicate with the Netscape and IBM Digital Signature Solution applications
- The internet/intranet connectivity and access to certification authority (CA)
- The download of certificate, private/public keys and processing
- The integration of the signature signing with the e-mail system
- Entering of health alert information and read/write by a PDA
- Capturing of the fingerprint information and store onto the employee card
- Accessing the secure area using the fingerprint

These tests will verify that the employee can use the new card the same way as the old one. The new employee smart card magnetic stripe on the back is encoded with the same information as the existing employee badge for facility access, cafeteria payment and for time and attendance. In addition, this new employee card can be used for digital signing, fingerprint identification, and health alert in phase 1 and e-purse payment, network access, and travel in the future.

13.2.2 Performance Testing

During the pilot, we need to get some performance data, such as the timing of loading keys and certificate, signing e-mail, reading a signed e-mail, volume and e-mail system load for signing the document and receiving mail, timing of the fingerprint matching and the release of the door lock, etc. With this data, we can tune the system for better performance. The time to capture the fingerprint, digitize the photo, and installation and deployment also needs to be obtained for the rollout.

13.2.3 Usability Testing

Usability testing needs to involve the employees, the user of the card. During the pilot, a sample of GTT employees from different professional skills and different positions is to be selected to actually use the card and provide the feedback. From the user feedback, the system can be modified or improved.

13.3 Rollout System Integration Testing

At the end of the pilot, all the components fixes, interface issues, performance problems, procedures, documentations, etc. should be done. The system should be functional as designed. In each of the geographic areas we still need to do the basic functional, usability, and performance tests as in the pilot to make sure the system works in a large-scale environment.

Besides the functional testing, we also need to test the following areas:

- The logistics of card distribution, batch or individually
- National language support for user documents and information packages
- Installation and deployment logistics and schedules
- Training procedures and schedules
- Import and export regulation for the length of the keys
- Different versions of Netscape

Chapter 14. Export and Import

Cryptography allows people to protect their privacy. For some, this means freedom. For others, privacy may mean conspiracy and anarchy. Many countries therefore have controls in place to limit the general availability and use of strong cryptography. The basic controls and regulations around cryptography are introduced in this chapter. Rules applying to smart cards are given emphasis.

Most smart cards must implement some form of cryptography. This might be symmetrical cryptography for MAC generation (see 2.4.1, “Message Authentication Code” on page 19 for more information) or it might be asymmetrical cryptography for digital signatures (see 2.5.1, “Digital Signature” on page 19 for more information). Because smart cards have cryptographic algorithms implemented on the chip or a cryptographic processor, they are subject to import and export regulations. The USA has strict regulations in the export of strong authentication, but other countries also limit the use of strong cryptography, for example France.

This chapter attempts to show examples of when a smart card is and is not subject to export restrictions.

14.1 Strong versus Weak Encryption

The government of the United States has enacted legislation called Export Administration Regulations (EAR) which prohibits export of strategic cryptographic technology from the U.S.A.. EAR defines strategic encryption as asymmetric cryptography (such as RSA) using key sizes over 512 bits and symmetric cryptography (such as DES, IDEA or RC5) using key sizes over 40 bits. Cryptographic algorithms that use keys whose size exceed these values are called ″strong cryptography″. Otherwise they are called ″weak cryptography″. The government is only interested in controlling the export of strong cryptography.

14.2 Cases Subject to Export Control

A cardholder should not have access to a smart card cryptographic function which can be enabled for bulk encryption using a strong encryption algorithm. Bulk encryption is the encryption of arbitrary data for reasons of privacy.

In short, if a smart card can perform two-way encryption (that is send clear text into a smart card, receive encrypted data back out and reverse the process to recover the original clear text) then the smart card may be violating the guidelines set down by the EAR.

If the smart card is to be exported from the United States, it will be necessary to apply for export clearance from the United States government whether the smart card performs weak or strong encryption. However, it is not illegal to sell and use a smart card which performs bulk encryption if it remains within Canada or the United States.

14.3 Cases Not Subject to Export Control

The Export Administration Regulations has relaxed the export controls that used to be in place which were defined by ITAR, an international agreement on arms export limitations drawn up during the Cold War. The relaxed rules means that it is possible to export a smart card from the United States that contains strong cryptography if it adheres to certain guidelines. We will give some examples to illustrate these exceptions.

14.3.1 IBM Agreement

IBM has reached an agreement with the USA government that allows the export of smart cards even if they contains strong cryptographic algorithms or crypto processors. The reason that the government allowed such exports is that the MFC operating system does not allow performing of cryptographic functions that are illegal according to US laws.

IBM negotiated the export of cards using the asymmetric keys of up to 1024 bits (not 2048) and symmetric keys of up to 112 bits.

The rule is straightforward: encryption must be used only for authentication, password and key encryption. No other strong encryption/decryption function or API is allowed.

For example, one of the limitations is that the public key in the smart card cannot be used in asymmetrical encryption when it is larger than 512 bits, as this would be strong encryption. There are no limitations in the use of the private key; 1024 bits keys are allowed, as is used only for authentication.

Another limitation imposed by the MFC operating system is the length of the data that can be encrypted, as we will see below.

The following are examples of what can be done with an IBM MFC card with regard to encryption.

14.3.2 Symmetrical Encryption Examples

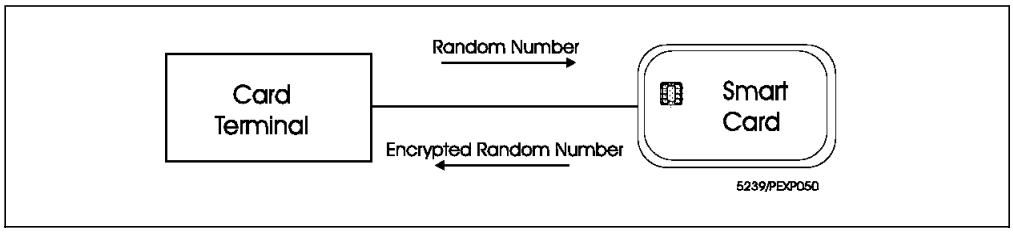


Figure 46. Example of Internal Authentication

Figure 46 illustrates the case where the card terminal wishes to certify that the smart card is genuine. The card terminal will use the Internal Authenticate APDU to perform this certification. See Chapter 8, “Card Layout Design” on page 87 for information about internal authentication. The card terminal will first generate a random number and send it to the smart card. The random number must be no longer than 8 bytes because the smart card cryptographic algorithm is limited to 8 bytes input. The smart card will encrypt the number using a strong symmetrical encryption algorithm such as 56-bit DES or triple DES (112

bits), and pass the encrypted block back to the card terminal, which will verify if the random number was correctly encrypted.

The limitations here are the size of the data that can be encrypted and that there is no ″reverse″ function in the card, which could decrypt the encrypted data.

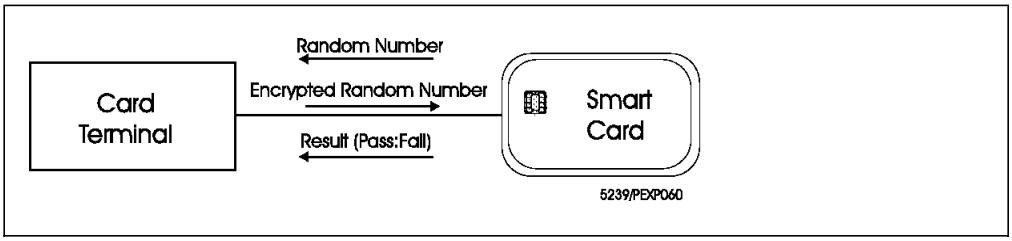


Figure 47. Example of External Authentication

Another example, illustrated in Figure 47, is the External Authenticate APDU which is used to certify that the card terminal into which the smart card has been inserted is genuine. Refer to Chapter 8, “Card Layout Design” on page 87 for information about external authentication. Similar to the internal authentication process, the smart card will generate a random number of up to 8 bytes, the card terminal will encrypt the random number using 56-bit key DES or triple DES 112-bit key and pass the encrypted block to the smart card, which will decrypt the block. Of course, both the smart card and the card terminal must know the encryption key in advance. The smart card sends back a return code indicating success or failure. It does not send back the decrypted clear text. Again, the limitation here is the data length; the decrypted data will not leave the card.

The question here is how, for both examples given above, the card terminal can handle strong cryptography, since it is located in a country outside the USA and subject to export regulations. We assume this card terminal is not installed in a bank, which has exceptions to the rules. How did the cryptographic algorithm get loaded into the card terminal? There are a couple of answers:

- The card terminal got the algorithm from a country outside the USA
- This card terminal could be a machine with two slots to insert two smart cards. One of the smart cards is trusted and remains all the time in the reader and is the one that does the encryption/decryption for the card terminal. The card terminal just passes whatever data comes from one smart card to the other.

  An example of this case could be a physician office; the card that remains in the terminal is the doctor′s, while the patient′s card is the one to be authenticated.

14.3.3 Asymmetrical Encryption Examples

An example using an asymmetrical algorithm such as RSA is the generation and verification of digital signatures.

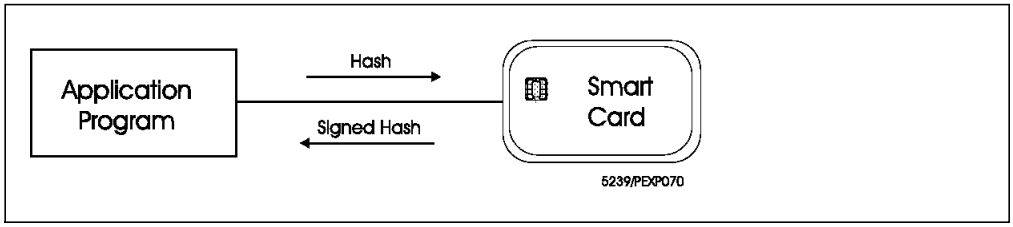


Figure 48. Example of Generating a Digital Signature

Smart cards support the generation of digital signatures using RSA with 1024-bit keys. As mentioned before, only the private key can be used for encryption. Export regulations are not violated (authentication).

Figure 48 illustrates the case where an application program asks a smart card to sign the hash, for example for digital signature. If the SHA algorithm is used to create the hash, the size of it would be 160 bits (20 bytes). MD4 and MD5 algorithms produce 128 bits hash. The MFC operating system will not encrypt data longer than 128 bytes. See Chapter 2, “Smart Card Security” on page 11 for details about digital signatures and hashing algorithms.

The card terminal passes the hash value to the smart card. The smart card returns the signature which is actually an encrypted block of the hash with a strong cryptographic algorithm using the private key stored in the smart card.

There is no violation of any rule, since anybody could decrypt the block using the smart card public key. This is only authentication.

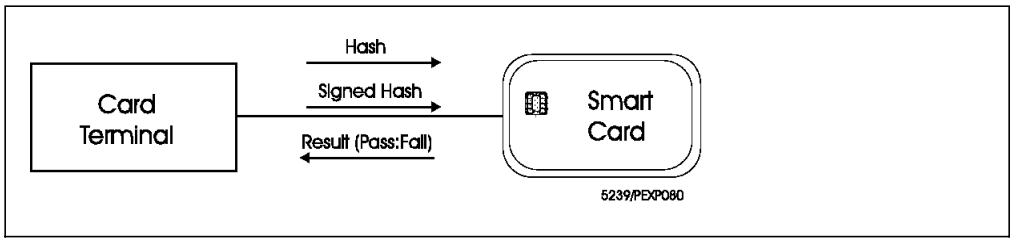


Figure 49. Example of Verifying a Digital Signature

The last example, Figure 49, is the case where the smart card is used to verify a digital signature. The application program will send the signature block and the calculated hash value to the smart card. The smart card will decrypt the signature and internally compare the results of the decryption with the supplied hash value. The smart card returns a result code indicating success or failure back to the application program. No data comes out of the card other than a code indicating if the verification was correct or not.

14.4 Digital Signatures, Netscape and Smart Cards

GTT has selected Netscape Communicator as the mail user agent for all employees. Those employees who need to sign e-mail will obtain a digital certificate from the CA and install it onto their smart card. For employees in the United States, they will be using the non-export version of Netscape Communicator (that is to say, the version with the ability to perform 1024-bit asymmetrical encryption). The employee can choose to generate a key pair of 512, 768 or 1024 bits.

For GTT employees in other countries, they will be using the export version of Netscape Communicator (that is to say, the version with only 512-bit asymmetrical encryption). A key pair generated outside the United States will have a length of 512 bits.

However, we should point out that Communicator has the ability to verify signatures using key sizes up to 1024 bits on both the export and non-export versions. This ability is independent of the smart card. The smart card is also able to handle generation and verification of signatures with key sizes up to 1024 bits but the version of Communicator limits the size of the key, not the ability of the smart card.

E-mail originating from GTT employees in the United States will be signed by Communicator using a 1024-bit key. If that employee were to sign an e-mail and send it to a person within GTT who resides outside the United States, for example in Germany, then the person receiving the e-mail would still be able to verify the e-mail signature.

Similarly, if that person in Germany were to reply to the e-mail and sign the reply, the person in the United States would also be able to verify the originator's signature.

The only consideration that GTT must make when selecting and installing Netscape Communicator company-wide is that Secure Sockets Layer (SSL) can only be done with a maximum key size of 40 bits for symmetrical encryption and 512 bits for asymmetrical authentication.

14.5 Smart Card Toolkits

Application programmers use smart card toolkits to help them design, debug and implement smart card applications. These toolkits sometimes include encryption algorithms, similar to those imbedded in the operating system of the smart card. For this reason, toolkits are also subject to export controls. If the toolkit is free of any strong encryption algorithms, then they may be freely exported from the United States.

For example, the IBM Smart Card ToolKit V2.6 comes in two flavors; exportable and non-exportable. The exportable version does not have any encryption algorithms in it except for those needed to generate Message Authentication Codes. This version may be exported outside the United States.

The non-exportable version includes all the encryption algorithms and requires a license to be exported outside the United States. The non-exportable version can generate encrypted keys for initializing a smart card which the exportable version would not have.

Previous releases of the IBM Smart Card ToolKit did not have an exportable version.

14.6 Conclusion

It is very difficult to give simple guidelines to follow concerning the export of smart cards. There are many rules and many exceptions imposed by the United States government and the governments of other countries. The best advice is to work with the card issuer or the person who sells the smart cards because it is their responsibility to seek conformance with all the appropriate export regulations and advise the customer when and where the smart card violates export and import regulations.

14.7 References

The following Web pages were used for gathering the information in this chapter. The reader is encouraged to read these works for further information about crypto law.

EFF ″Privacy - Crypto - ITAR Export Restrictions″ Archive
(http://www.eff.org/pub/Privacy/ITAR_export/)

RSA′s Cryptography Export FAQ (http://www.rsa.com/PUBS/exp_faq.pdf)

Cryptography Export Controls Archive
(http://ftp.cygnus.com/pub/export/export.html)

Bert-Jaap Koops′ Crypto Law Survey
(http://cwis.kub.nl/~ frw/people/koops/lawsurvy.htm)

UK Export Controls (http://www.dcs.ex.ac.uk/~ aba/ukexport/)

John Young′s Arms Control Essays (http://jya.com/acda.htm)

Electronic Privacy Information Center (http://www.epic.org/)

Appendix A. Sample Smart Card Layout for the Case Study

This appendix contains the card layout produced by the IBM Smart Card ToolKit as explained in Chapter 8, “Card Layout Design” on page 87.

```
/*****************************************************************/
/*                                                               */
/*  GTT.CLT   - Redbook Case Study                               */
/*                                                               */
/*****************************************************************/
ENVIRONMENT
(
   CHIP PASSWORD ( 0x0000000000000000 )/* MFC 4.21/8KB Card     */
   OS TYPE ( MFC_421 )                 /* Card Operating System */
   PROCESSOR ( CF68 )                  /* SGS-THOMSON  8 KB     */
   INIT PROTOCOL ( "GTTINIT.CLT" )     /* Identification String */
   CODE EXTENSION ( "MFC421.CEB" )     /* Code Extension Data   */
)

DECLARATIONS
(
   GTT_KEY_DOMAIN (1)
   GTT_AUT_DOMAIN (2)
   GTT_CHV_DOMAIN (3)
)

DIRECTORY
(
   MF
   ID ( 0x3F00 )
   Access
   (
      Lock   ( Pro(0) )
      Update ( Pro(0) )
      Create ( Pro(0) )
      Delete ( Pro(0) )
   )
   Agent data ( MF  SELECT )
   Key domain (GTT_KEY_DOMAIN)
   Aut domain (GTT_AUT_DOMAIN)
   Chv domain (GTT_CHV_DOMAIN)

/************************************************

      D I R    Select by Application Name (EMV)

 ************************************************/

   EMV DICTIONARY
   (
      EF_DIR_MFC
      ID (0xFD01)
      Organization (variable)
      Access
      (
         Read(Always)
         Update(Pro(0) )
         Append(Pro(0) )
      )
      Agent data (EF_DIR_FIRST      FIRST  )
      Agent data (EF_DIR_NEXT       NEXT   )
      Agent data (EF_DIR_LAST       LAST   )
      Agent data (EF_DIR_PREVIOUS   PREVIOUS )
      Agent data (EF_DIR            SELECT )
   )

/************************************************

      A T R    R e s p o n s e

 ************************************************/
```

```
  File
  (
     EF_ATR
     ID (0x2f01)
     Access
     (
        Read(Always)
        Update(Pro(0) )
     )
     /* Check-sum is calculated by the smart card chip */
     Initial data
     (
         0x173BEF00FF813166457E05384754542020202020202020
     )
     Agent data(EF_ATR)
  )

/************************************************

     C a r d    I d e n t i f i c a t i o n

 ************************************************/
  File
  (
     EF_ICC
     Id (0x0002)
     Access
     (
        Read(Always)
        Update(Never)
     )
     Initial data
     (
        0x00,              /* Clockstop not allowed            */
        0x01020304,        /* IC Card serial number            */
        0x05060708,        /* IC Card manufacturing references */
        0x01,              /* Card personalizer ID             */
        "DE"               /* Country code of embedder         */
        "IB",              /* Char based on name of embedder   */
        "M",               /* Char for other purpose.          */
        0x1234,            /* IC Identifier                    */
        0x99,              /* Card profile                     */
        0x0B               /* Type of selection for MFC 4.21   */
     )
  )

/************************************************

     I D    F i l e

 ************************************************/
  File
  (
     EF_ID
     Id (0x0003)
     Access
     (
        Read(Always)
        Update(Never)
        Invalidate(Pro(0) )
        Rehabilitate(Pro(0) )
     )
     Initial data
     (
        (card_id = 0x01234567890123456789),
        (card_ad = 0x19980701    ),  /* Date of Activation BCD */
        (card_ex = 0x20030630    ),  /* Date of Expiry BCD     */
        (card_nr = 0x01          ),  /* Card Seq. Number BCD   */
        (card_cc = 0x0840        )  /* Country Code(US) ISO3166*/
     )
     Agent data
     ( EF_ID
       agent data ( CardIdentificationNumber length( sizeof(card_id) ) )
       agent data ( CardExpiryDate           length( sizeof(card_ex) ) )
       agent data ( CardDateOfActivation     length( sizeof(card_ad) ) )
```

```
      agent data ( CardSequenceNumber       length( sizeof(card_nr) ) )
      agent data ( CardCountryCode          length( sizeof(card_cc) ) )
    )
  )

/***********************************************

 C a r d   H o l d e r   V e r i f i c a t i o n

 ***********************************************/
  File
  (
     EF_CHV1
     Id (0x0000)
     Access
     (
        Read        (Never )
        Update      (Enc(0))
        Invalidate  (Enc(0))
        Rehabilitate(Enc(0))
     )
     Initial data
     (
        0x01,                       /* CHV is valid            */
        0x00,                       /* CHV is clear/alphanum   */
        0x00,                       /* Key Number if enc. used */
        "password",                 /* Password or PIN value   */
        0x03,                       /* Maximum attempts        */
        0x03,                       /* Remaining attempts      */
        "xxxxxxxx",                 /* Password for unblocking */
        0x05,                       /* Unblock attempts counter*/
        0xFF                        /* Unlimited use of unblock*/
     )
  )

  File
  (
     EF_CHV2
     Id (0x0100)
     Access
     (
        Read        (Never )
        Update      (Enc(0))
        Invalidate  (Enc(0))
        Rehabilitate(Enc(0))
     )
     Initial data
     (
        0x01,                       /* CHV is valid            */
        0x00,                       /* CHV is clear/alphanum   */
        0x00,                       /* Key Number if enc. used */
        "adminpwd",                 /* Password or PIN value   */
        0x03,                       /* Maximum attempts        */
        0x03,                       /* Remaining attempts      */
        "xxxxxxxx",                 /* Password for unblocking */
        0x05,                       /* Unblock attempts counter*/
        0xFF                        /* Unlimited use of unblock*/
     )
  )

/***********************************************

     P e r s o n a l i z a t i o n    K e y

 ***********************************************/

  File
  (
     EF_KEY_PERS
     Id ( 0xFD21 )
     Organization (variable)
     Access
     (
        Read(Never)
        Update(Never)
```

```
      )
      Record
      (
       Initial data
       (
        0x00,                    /* Key Number 0               */
        0x06,                    /* 06-Standard, 07-Triple DES */
        0x03,                    /* Valid tries counter        */
        0x00,                    /* Key version                */
        0xFFFF,                  /* Reserved                   */
        0x0123456789ABCDEF       /* Personalization Key Value  */
       )
      )
   )

/***********************************************

      K e y   M a n a g e m e n t   F i l e

 ***********************************************/

   File
   (
      EF_KEY_MAN
      Id ( 0x0011 )
      Organization (variable)
      Access
      (
         Read(Never)
         Update(Never)
      )
      Record
      (
       Initial data
       (
        0x00,                    /* Key Number 0               */
        0x06,                    /* 06-Standard, 07-Triple DES */
        0x03,                    /* Valid tries counter        */
        0x00,                    /* Key version                */
        0xFFFF,                  /* Reserved                   */
        0x0123456789ABCDEF       /* Personalization Key Value  */
       )
      )
   )

/***********************************************

      G l o b a l     K e y s

 ***********************************************/

   File
   (
      EF_KEY
      Id ( 0x0001 )
      Organization (variable)
      Access
      (
         Read(Never)
         Update(Enc(0))
      )
      Record
      (
       Initial data
       (
        0x00,                    /* Key Number 0               */
        0x06,                    /* 06-Standard, 07-Triple DES */
        0x03,                    /* Valid tries counter        */
        0x00,                    /* Key version                */
        0xFFFF,                  /* Reserved                   */
        0x0123456789ABCDEF       /* Key value                  */
       )
      )
      Record                     /* Key Number 1               */
      (                          /* Key inserted during pers.  */
```

```
      Initial data(0x01060300FFFF)
      Personal data(Index(0xA010) size(8) Protection(MAC(0)) )
    )
    Record                      /* Key Number 2              */
    (
      Initial data(0x02060300FFFF)
      Personal data(Index(0xA020) size(8) Protection(MAC(0)) )
    )
    Record                      /* Key Number 5              */
    (
      Initial data(0x05060300FFFF)
      Personal data(Index(0xA030) size(8) Protection(MAC(0)) )
    )
    Agent data ( EF_KEY  Select )
  )

  File
  (
    EF_AUT
    Id (0x9F03)
    Size ( 3 * ( 6 + 8 + 2 ) )
    Organization (variable)
    Access
    (
       Read (Never)
       Update(Enc(0))
       Append(Enc(0))
    )
    Record
    (
     Initial data
     (
      0x00,                     /* Auth. Key Number 0         */
      0x06,                     /* 06-Standard, 07-Triple DES */
      0x03,                     /* Reserved (Status)          */
      0x00,                     /* Key version                */
      0xFFFF                   /* Reserved                   */
      )
      Personal data             /* Key value is psersonalized */
      (
          Index(20) Size(8) Protection(MAC(0))
      )
    )
    /* Space for two more keys reserved by Size for future use*/
    Agent data ( EF_AUT  Select )
  )

/***********************************************

        E m p l o y e e    D e t a i l s

 ***********************************************/
  FILE
  (
    EF_Employee
    ID (0xA100)
    Size ( 80)
    Access
    (
       Read(Always)
       Update(Never)
    )
    Agent data
    (
        Employee
        Agent data  ( PersonnelNumber    length(10) )
        Agent data  ( LastName           length(30) )
        Agent data  ( FirstName          length(30) )
        Agent data  ( Title              length(10) )
    )
  )


/***********************************************
```

```
     F a c i l i t i e s     A c c e s s

 ***********************************************/
  FILE
  (
     EF_Access
     Short File ID (0x0D)
     Size ( 8)
     Access
     (
        Read(Pro(2))
        Update(Pro(1))
     )
     Agent data ( AccessCode         length(8) )
  )

/***********************************************

     H e a l t h    A p p l i c a t i o n

 ***********************************************/
  Directory
  (
     DF_Health
     Id ( 0xC020 )
     Emv dictionary entry
     (
          Name ( 0xD840000025,      /* Application ID - RID    */
                 0x00000010    )    /* Application ID - PIX    */
          Type ( ADF )
          Label("GTT Health")
          TAG
          (
               0x52,                /* Command to perform      */
               0x07,                /* Command length          */
               0x00,0xA4,           /* CLA, INS                */
               0x00,0x00,0x02,      /* P1, P2, P3              */
               0xC020               /* File ID of application  */
          )
     )
     Access
     (
       Create (Pro(2))
       Delete (Pro(2))
     )
     Agent data ( DF_Health  Select )
     FILE
     (
        EF_MedicalData
        ID (0xC021)
        Size ( 140)

        Access
        (
           Read(Always)
           Update(CHV1)
        )
        Agent data
        (
           Medical
           Agent data  ( BloodType          length(2)  )
           Agent data  ( Allergies          length(20) )
           Agent data  ( Doctor             length(24) )
           Agent data  ( Phone              length(20) )
           Agent data  ( CompanyScheme      length(1)  )
           Agent data  ( InsuranceCompany   length(20) )
           Agent data  ( InsNumber          length(10) )
           Agent data  ( RegularMedications length(40) )
           Agent data  ( OrganDonor         length(1)  )
        )
     )
  )

/***********************************************
```

```
C a f e t e r i a    P u r s e

***********************************************/
  Directory
  (
     DF_Cafeteria
     Id ( 0xC000 )
     Emv dictionary entry
     (
          Name ( 0xD840000025,      /* Application ID - RID    */
                 0x00000050    )    /* Application ID - PIX    */
          Type ( ADF )
          Label("Cafeteria Purse")
          TAG
          (
               0x52,                /* Command to perform      */
               0x07,                /* Command length          */
               0x00,0xA4,           /* CLA, INS                */
               0x00,0x00,0x02,      /* P1, P2, P3              */
               0xC000               /* File ID of application  */
          )
     )
     Access
     (
       Create (Never)
       Delete (Never)
       Invalidate (Pro(1))
       Rehabilitate (Pro(1))
     )
     Agent data ( DF_Cafeteria  Select )

     FILE
     (
         EF_PurseKey
         ID (0x0001)
         Size ( 374 )
         Access
         (
            Read(Never)
            Update(Enc(1))
         )
     )

     FILE
     (
        EF_Purse
        ID (0x7F01)
        Size ( 36 )
        Access
        (
           Read(Aut(1))
           Update(Aut(1))
        )
     )

     FILE
     (
        EF_Amount
        ID (0x7F02)
        Size ( 27 )
        Access
        (
           Read(Aut(1))
           Update(Aut(1))
        )
     )

     File
     (
        EF_LLog
        Id (0x7F03)
        Organization (Cyclic size ( log_length = 33 ))
        size (99 )
        Access
        (
```

```
            Read(Always)
            Update(Always)
         )
         Agent data ( PURSE_LLOG_0 PREVIOUS length( log_length ) )
      )

      File
      (
         EF_BLog
         Id (0x0109)
         Organization (Cyclic size ( log_length = 37 ))
         size (37*3 )
         Access
         (
            Read(Always)
            Update(Always)
         )

         Agent data ( PURSE_BLOG_0  PREVIOUS  length( log_length ) )
      )
      FILE
      (
          EF_LSeq
          ID (0x7F04 )
          Size ( 22   )
          Access
          (
             Read(Never)
             Update(CHV1)
          )
      )
      FILE
      (
          EF_BSeq
          ID (0x7F26 )
          Size ( 22   )
          Access
          (
             Read(Never)
             Update(CHV1)
          )
      )
   )

/***********************************************

 D i g i t a l     S i g n a t u r e

 ***********************************************/
   Directory
   (
      DF_PKCS11
      Id ( 0xC110 )
      Emv dictionary entry
      (
           Name ( 0xD8400000025,      /* Application ID - RID    */
                  0x00000020    )    /* Application ID - PIX    */
           Type ( ADF )
           Label("Digital Signature")
           TAG
           (
                0x52,                /* Command to perform      */
                0x07,                /* Command length          */
                0x00,0xA4,           /* CLA, INS                */
                0x00,0x00,0x02,      /* P1, P2, P3              */
                0xC110               /* File ID of application  */
           )
      )

      Access
      (
        Create (CHV1 )
        Delete (CHV1 )
      )
      Agent data ( DF_Signature  Select )
```

```
    FILE
    (
        EF_SER
        ID (0xC001)
        Size ( 18 )
        Organization (Transparent)
        Access
        (
           Read(Always)
           Update(Never)
        )
    )
    FILE
    (
        ctrl_pub
        ID (0xC100)
        Size ( 1792 )
        Access
        (
           Read(Always)
           Update(Always)
        )
    )
    FILE
    (
        ctrl_pri
        ID (0xC200)
        Size ( 704  )
        Access
        (
           Read(CHV1)
           Update(CHV1)
        )
    )
    FILE
    (
        EF_KEY_PKA
        ID (0xFE00)
        Size ( 360  )
                 /* Content of Header data depends */
                 /* on key stored                  */
        Header data( 0xC20700000000000000 )
        Access
        (
           Read(Never)
           Update(CHV1)
        )
    )
  )

/***********************************************

 B i o m e t r i c   I d e n t i f i c a t i o n

 ***********************************************/
  Directory
  (
     DF_Biometrics
     Id ( 0xB001 )
     Emv dictionary entry
     (
          Name ( 0xD840000025,      /* Application ID - RID    */
                 0x00000040    )    /* Application ID - PIX    */
          Type ( ADF )
          Label("GTT Biometrics")
          TAG
          (
               0x52,                /* Command to perform      */
               0x07,                /* Command length          */
               0x00,0xA4,           /* CLA, INS                */
               0x00,0x00,0x02,      /* P1, P2, P3              */
               0xB001               /* File ID of application  */
          )
     )
```

```
        Access
        (
          Create (Pro(0))
          Delete (Pro(0))
          Invalidate (Never)
          Rehabilitate (Never)
        )
        Agent data ( DF_Biometric  Select )
        FILE
        (
            EF_LFingerPrint
            ID (0xF001)
            Size ( 402 )
            Access
            (
               Read(Always)
               Update(Pro(1))
            )
        )
        FILE
        (
            EF_RFingerPrint
            ID (0xF002)
            Size ( 402 )
            Access
            (
               Read(Always)
               Update(Pro(1))
            )
        )

        FILE
        (
            EF_Control
            ID (0xF003)
            Size ( 100  )
            Access
            (
               Read(Pro(1))
               Update(Pro(1))
            )
        )
     )

) /* end of MF */
```

Appendix B. JavaCard Overview

A JavaCard is a smart card that can load and execute programs written in Java.

This is a new approach with smart cards. The traditional smart card only contains user data files; the programs that execute in the card are already provided by the operating system vendor.

The JavaCard tries to solve some of the shortcomings of the traditional smart cards. We will describe them in the next section.

In this appendix we will look at what a JavaCard is, the advantages a JavaCard has over traditional smart card technologies, and some of the applications for JavaCards.

B.1 Introduction to JavaCard

The JavaCard tries to solve some of the problems that present the traditional ISO smart cards. We will show here a few samples:

- Development of new functions requires insider knowledge

  Suppose that a customer wants to implement new functions in a smart card, for example:

  - A new encryption algorithm
  - A new hashing technique
  - A one-time password scheme

  As we said earlier, the user only defines data files structures in a traditional smart card. Although some of the traditional smart card operating system will allow you to load and run programs in the card, this is a formidable task that only few programmers can do; they usually are the people who develop the operating systems. In an MFC card, these executable files are called ASC.

  This situation changes dramatically when you have a JavaCard. A programmer could code this functions in Java, test it outside the card, and then securely download the new code into the card.

  Although the programs coded to run in a smart card can use only a subset of Java, they provide a great enhancement compared to coding a program for a traditional smart card operating system.

- Downloaded customer functions in native code cannot be controlled

  The execution of native code cannot be controlled by the OS. New code has to be manually inspected before downloading. JavaCard offers increased security in the card. As Java is an interpretative language, all instructions can be checked before executing, preventing an application from performing illegal functions, such as accessing other application′s data.

- Direct data access in files

  Today′s standard ISO multi-application smart cards like the IBM MFC basically provide a secure file system by allowing controlled but direct access to data on the card. The data structures are visible to the business application. A JavaCard would allow you to implement new file access methods.

- Data sharing is not flexible enough

  MFC′s file tree allows data separation and sharing to a certain extent, but changes are hard to implement. This is specially true when new applications are added to the card; the sharing of data that already existed between two applications can be difficult to maintain when a third one is added.

All these benefits come with a price: an interpretative language running in a rather slow smart card CPU with limited memory will impact performance. But we need to keep in mind that chip evolution is doubling the speed of CPU on a frequent basis and in the near future there will be some 32KB EEPROMs.

B.1.1 JavaCard Architecture

A JavaCard is a normal smart card in that it has a CPU, ROM and EEPROM memory, and (possibly) a cryptographic co-processor on the chip. It is compliant with the ISO 7816 standards parts 1, 2 and 3.

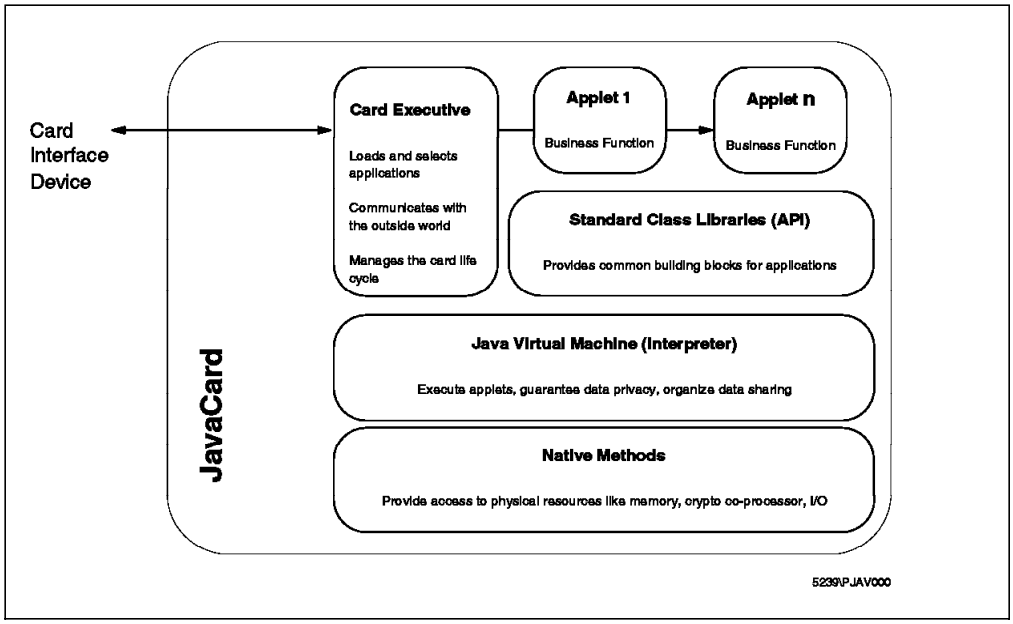


Figure 50. JavaCard structure

Figure 50 is a diagram of the JavaCard architectural elements.

The card operating system consists of a Java virtual machine, a piece of software that can execute programs written in the Java language (Java was invented by Sun Microsystems). The JavaCard virtual machine (VM) runs a special, limited version of Java.

Services, such as communicating with a smart card reader or creating data in memory, are performed for the JavaCard virtual machine via libraries of Java programs called classes. JavaCard classes can be loaded onto the smart card chip in ROM when the chip is manufactured or they can be loaded into EEPROM memory afterwards along with the smart card applications.

We have to keep in mind that a difference between a traditional smart card and a JavaCard is that while in an ISO card all the functions are already present in the card operating system when it is received by the user, a JavaCard is

practically empty. Most of the functions will have be added by loading applets in the card.

B.2 When to Use JavaCard?

We will try to point out some cases in which the use of JavaCards is specially indicated:

- When special data access methods are required.

  These special access methods could be coded in Java and loaded in the card for use of different applications.

- When in a multiapplication card the owners of these applications cannot agree on a file system structure.

  JavaCard allows the different applets to have different file structures.

- Sharing of data between applications.

  As mentioned above, sharing of data in the ISO card's file structure can become cumbersome when new applications are added to the card. This is not the case in a JavaCard.

- The different application providers do not trust each other.

  JavaCard offers strong firewalling between applications.

B.2.1 Securely Downloading Applications

VISA has been very active in the development of the JavaCard. VISA Open Platform (OP) offers several classes for smart card application developers. One of them is a JavaCard class that implements a secured applet downloading mechanism. It is called CardDomain applet. VISA OP guarantees the integrity of JavaCard applications that will be loaded onto a JavaCard after the card has been issued to the cardholder. VISA OP does this code verification through the use of digital signatures.

B.2.2 Examples of JavaCard Applications

At the time of writing, the JavaCard API 2.0 Reference Implementation has only just been released by JavaSoft so there has not been enough time to have real world examples of JavaCards released to the market. These cards and applications are currently under development by companies such as Bull, IBM, Giesecke & Devrient, Schlumberger, Gemplus, De La Rue and so on.

Several companies are offering development kits and evaluation JavaCards. For example, Schlumberger is currently offering a software development kit for their CyberFlex JavaCard (see http://www.cyberflex.slb.com/). Bull is offering the Odyssey card, which complies with the JavaCard 2.0 Reference Specification (see http://www.cp8.bull.net/products/javacara.htm). Giesecke & Devrient markets the C@ppucino JAVA™ card and software development environment for the card (see http://www.gdm.de/products/terminal/software/javacap1.htm).

At the time of publication, there has been one JavaCard pilot announced. In June 1998, Standard Charter Bank in the United States and VISA International Ltd. launched a multifunction JavaCard (http://www.gemplus.com/presse/1998/standard2_uk.htm) with four applications on one smart card:

- Credit card payment

- Loyalty program
- Bank relationship
- Security function for Internet payment

This example will be typical of the types of smart cards that will become prevalent in the market within the next few years. These JavaCards can be customized to the individual cardholder's needs by downloading to the card just those applications that the cardholder requires.

B.2.3 JavaCard Forum

The current owner of the JavaCard API specification is JavaSoft, a division of Sun Microsystems, Inc. In 1997, Schlumberger, Bull and Gemplus created a consortium to promote the use of Java in smart cards. This Consortium is called the JavaCard Forum. The primary purpose of the JavaCard Forum is to promote Java as the preferred programming language for multiple-application smart cards. Their primary activity is the development and recommendation of API specifications to JavaSoft for subsequent inclusion into the specification.

The current membership in the JavaCard Forum is growing. You should check http://www.javacardforum.org for a current listing. More information about JavaCard can be found on the Web at http://java.sun.com/products/javacard/.

Appendix C. Sample JavaCard Applet

The following example shows a very basic JavaCard applet which stores the cardholder's name, mentioned in 10.6, "Programming JavaCards" on page 134. The applet handles reading and updating of the cardholder's name.

```
/*
 *    Sample Java Card Applet:
 *
 *    Purpose : An outline coding example to show the basic
 *              elements of a JavaCard Applet.
 *
 */

package cardlets;
                        /* The javacard.framework package defines
                         * the basic building blocks for JavaCard
                         * programs. It has classes defined to handle
                         * APDUs, runtime and system services.
                         */
import javacard.framework.*;
                        /* The Employee is an instance of an
                         * applet class which extends from
                         * javacard.framework.Applet.
                         */
public class Employee extends javacard.framework.Applet
{
                        /* Instruction bytes for the APDU header */
    public static final byte READ_INS    =(byte) 0xBB ;
    public static final byte UPDATE_INS  =(byte) 0xCC ;
    public byte£‘ apduBuffer ;
    private static byte£‘ name  ;

                        /* The method install() is invoked by JCRE.
                         * The applet calls the register method to
                         * register with the JCRE, which then makes
                         * it visible to the outside world.
                         */
    public static void install (APDU apdu ) throws ISOException
    {
        Employee person=new Employee();
        person.register();
    }
                        /* The method select() is invoked by JCRE
                         * to notify that the applet has been selected.
                         * Any initialization to carry out the APDU
                         * processing is carried out here.
                         */
    public boolean select() throws ISOException
    {
        return TRUE;
    }
                        /* The method process() is invoked by JCRE
                         * to handle incoming APDUs.
                         */
    public void process(APDU apdu) throws ISOException
    {
        apduBuffer = apdu.getBuffer();
        switch( (byte) apduBuffer£ISO.OFFSET_INS‘)
        {
            case READ_INS  : readName();
            case UPDATE_INS: updateName();
            default:
                ISOException.throwIt(ISO.SW_INS_NOT_SUPPORTED);
        }
        return;
    }
                        /* The readName method copies the
                         * cardholder's name to the APDU buffer
                         * and send the data. JCRE takes care of
                         * forming the correct APDU response
                         * with the appropriate status word; in this
```

```
                     * case, normal completion (0x9000).
                     */
    public void readName()
    {
        byte£‘ buffer = apdu.getBuffer;
        util.arrayCopy( name, 0, buffer, 0, name.length);
                     /* Send data equal to length of name
                      * starting at offset 0
                      */
        apdu.setOutgoingAndSend(short)0, (short) name.length);
        return;
    }
                     /* The updateName method expects incoming data
                      * consisting of the cardholder's name
                      * and receives the data starting at offset
                      * ISO.OFFSET_CDATA. (Code to permit only
                      * restricted updating is not shown here).
                      */
    public void updateName()
    {
        byte£‘ buffer = apdu.getBuffer;
        length = buffer£ISO.OFFSET_LC‘ ;
        util.arrayCopy( buffer, ISO.OFFSET_CDATA , name, 0, length);
        return;
    }
}
```

Appendix D. Overview of OpenCard Framework

JavaCard goes a long way to solving the current incompatibilities and restrictions of card operating systems. Even though JavaCard gives us vendor operating system independence, we are not divorced from knowledge of the underlying hardware. Also, we need something to manage different card issuers′ applications on the JavaCard. For example, we need something to list the applications on the card, to install new applications and so on. In essence, we need an interface between the application program and the smart card that:

- Serves as an application platform for smart cards and smart card readers of any flavor
- Provides a standardized, easy to use, high-level API

This is the Java-based OpenCard Framework (OCF) proposed by the OCF Consortium, a collection of major IT manufacturers.

Here is a list of the players in the field of smart cards who would benefit from OpenCard Framework, with a brief description of the problem areas that OCF solves.

Smart Card Reader Vendors

They provide the actual smart card readers (or card terminals in OCF terminology). Each vendor has more or less a broad spectrum of smart card readers, ranging from the very simple ones to more sophisticated models (including perhaps units with a display, a PIN pad, or even a biometric input device). For further information about biometric devices, see 2.7.2, “Biometrics” on page 22.

Unfortunately, the card reader vendors so far have not agreed on a standard programming interface. The few attempts at standardization, such as the European CT API or the G7 health card interface are either of regional importance or have not yet achieved wide acceptance.

Smart Card Operating System Providers

Like the proprietary nature of smart card readers, card operating systems from smart card vendors are also proprietary. There are several competing companies offering competitive card operating systems claiming to comply with the ISO 7816 standard. Unfortunately, the ISO 7816 standard allows for a wide variety of commands and response codes for these card operating systems. Therefore, these competitive operating systems do not work the same way. Recent initiatives, such as JavaCard and MULTOS, are trying to address this issue of card operating system incompatibility.

Smart Card Issuers

In addition to issuing card to consumers, they are also the application providers. The card issuers decide where to place the card-resident applications on the smart card that they issue to their customers. The same card-resident application might end up in different places on cards distributed by different card issuers.

OpenCard Framework addresses these inconsistencies by providing an API that is independent of the smart card inserted into the smart card reader, independent of the card operating system on that card and independent of the card issuer and where that issuer has placed the card-resident application.

The benefits offered by the OpenCard Framework are:

Vendor Independence
Developers can choose cards and readers from different suppliers and are no longer tied to one particular vendor.

Asset Protection
Extensibility of the architecture enables developers to participate in future developments of smart card technology at low cost.

Improved Time-To-Market
Developers profit from shorter development cycles by programming in a high level language, Java, against a high-level API.

Lower Development Costs
Developers save the extra cost of porting their applications to different platforms

Less Development Effort
Smart card providers inherit functionality provided by the framework that reduces their development effort.

D.1 Architecture Overview

We can divide the OpenCard Framework into two main components: The *CardTerminal* component and the *CardService* component; see Figure 51.

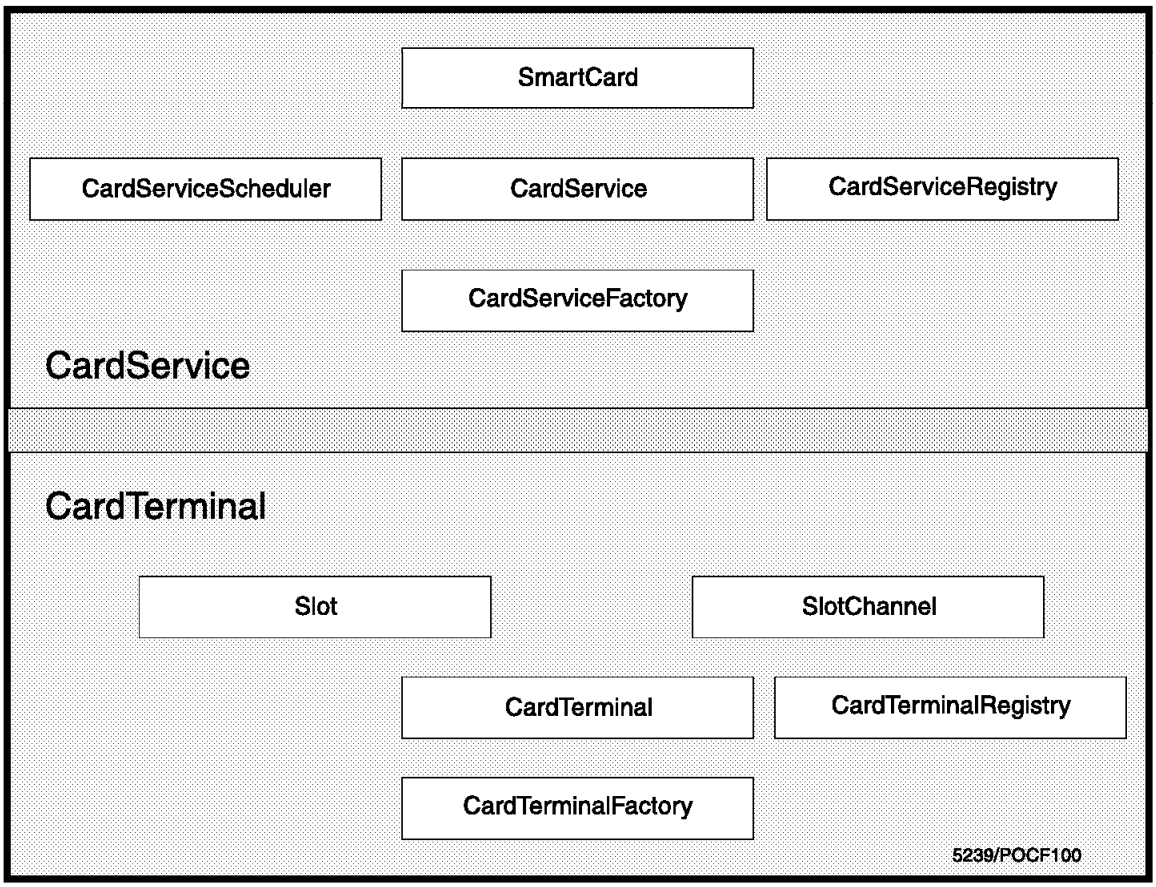


Figure 51. OpenCard Framework Architecture

The *CardTerminal* component contains classes and interfaces that allow you to access card terminals and their slots. Using these classes you can, for example, find out whether a smart card is inserted in a card terminal.

The *CardService* component defines the abstract CardService class: OCF represents smart card functions through card services. Each card service defines a particular (high-level) API that you can use to access some particular smart card function; for example, the file system card service gives you access to the file system of a smart card.

Both of OCF's components — *CardTerminal* and *CardService* — are designed using the abstract factory pattern and the singleton pattern. The objects dealing with the manufacturer-specific details are produced by a factory object which is supplied by the respective manufacturer. To determine which factory to use, OCF deploys a singleton called *registry*. A registry contains the configuration of an OCF component and creates the corresponding factory objects as needed.

We'll take a closer look at both components, the *CardTerminal* and the *CardService* component, in the next two sections.

D.2 Card Terminals

There is a broad range of card terminals on the market, with widely differing functions. Very simple card terminals merely provide basic card input-output (I/O) functionality via a single slot to insert the card. More sophisticated card terminals offer multiple slots or include a PIN pad and a display that can be used to perform cardholder verification. Card terminals can attach to different I/O ports (for example, serial ports and PC Card buses). Card terminals come with the proper driver software for selected computer operating systems. OCF's *CardTerminal* component provides adequate abstractions for the details of the various card terminals.

The classes of the *CardTerminal* component, as illustrated in the lower part of Figure 51 on page 186, serve a dual purpose: the most prominent function is to provide access to physical card terminals and inserted smart cards. This function is encapsulated in the CardTerminal class , the Slot class, and the CardID class.

In OCF, a physical card terminal is represented through the CardTerminal class and the Slot class. The *answer-to-reset* (ATR)[8] is represented in OCF through the CardID class. OCF allows for both static and dynamic configuration. In the case of static configuration, the configuration of card terminals is known *a priori* at system startup time. In the case of dynamic configuration, on the other hand, additional card terminals (for example, PC Card-based terminals) can be added at run-time.

The CardTerminal class is an abstract superclass from which concrete implementations for particular card terminal types are derived. Each CardTerminal object contains one or more Slot objects that represent the physical card slots of that card terminal. Access to a smart card that is inserted in a slot occurs through an exclusive gate object, the SlotChannel object: The CardTerminal class ensures that, at any given point in time, a maximum of one

[8] The ATR is the initial information emitted by a smart card upon being reset; see 8.3.1, "Answer to Reset (ATR)" on page 96

SlotChannel object per slot is instantiated. Thus, once an object has obtained a SlotChannel object, no other object can gain access to the associated smart card until that SlotChannel object has been released.

The CardTerminal class provides methods for checking card presence, obtaining a SlotChannel object, and sending and receiving APDUs. For card terminals offering additional functions — such as a display, a PIN pad, a finger print reader, or other input-output facilities — OCF provides additional interfaces that a CardTerminal can implement.

The second function of the CardTerminal package is a mechanism to add and remove card terminals. The CardTerminalFactory class, and the CardTerminalRegistry object implement this function. Each card terminal manufacturer supporting OCF provides a CardTerminalFactory which ″knows″ about a particular family of card terminals, and the respective CardTerminal classes. The system-wide unique CardTerminalRegistry object keeps track of the installed card terminals — as illustrated in Figure 52

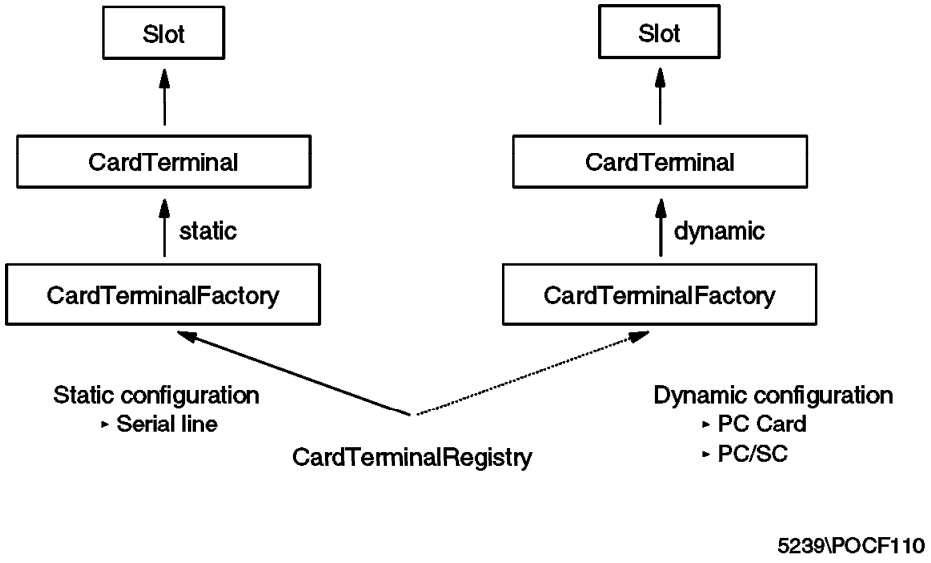


Figure 52. Classes of the CardTerminal Package

OpenCard Framework represents a real card terminal through the CardTerminal class and the Slot class. With OCF, both static and dynamic configurations are possible.

Furthermore, the CardTerminalRegistry object offers methods to register and unregister CardTerminal objects, and to enumerate all installed card terminals.

Both the CardTerminalRegistry and CardTerminal classes generate events and notify the rest of the framework (see Figure 53 on page 189) when:

- A card terminal is added to or removed from the system (CardTerminalRegistry), and
- A smart card is inserted into or removed from a card terminal (CardTerminal).

D.3 Card Services

Card Services are the means by which the OCF makes smart card functions available to the application programmer. Therefore, they are the components that you probably will work with the most.

At the present time, the OpenCard Framework reference implementation defines only a few card service interfaces: the *file system card service* and the *signature card service*. However, the number of standard card services will increase quickly, as it is the intention of the *OpenCard Consortium* to define card services for virtually all smart card functions.

D.3.1 File System Card Service

The electronic storage of information is often based on some sort of file system concept. Not surprisingly, most smart cards in use today offer tamper-proof storage via a file system abstraction.[9]

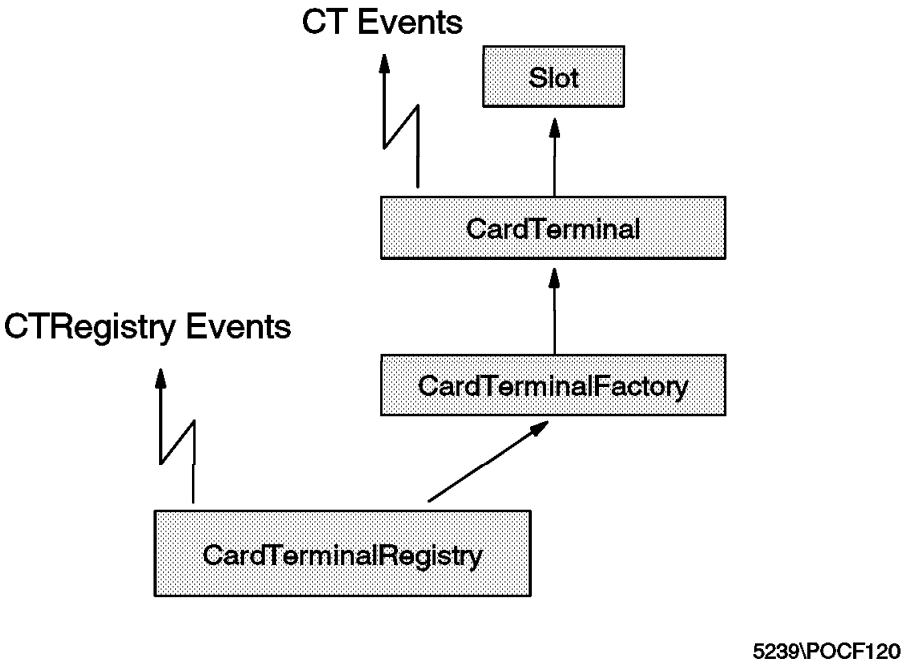


Figure 53. Events of the CardTerminal Component

The CardTerminalRegistry object generates CTRegistryEvents whenever a card terminal is registered or unregistered; by the same token, each CardTerminal object notifies interested parties of card insertion and removal through CardTerminalEvents.

The FileSystemCardService is *OpenCard*'s abstraction for dealing with smart-card based files. It offers a high-level interface modelled after Java's java.io package, making it virtually unnecessary for you to understand the nitty-gritty details of those standards.

[9] *Relational database* mechanisms or *object-based persistence* mechanisms are alternate abstractions for the storage of information and are considered for smart cards, too (see also the ISO standard *ISO 7816-7 Interindustry commands for Structured Card Query Language (SCQL)*

D.3.2 Signature Card Service

Electronic signatures of data rely on cryptographic mechanisms that make use of cryptographic key information to perform their operation. A common scheme for signing data is the asymmetric RSA algorithm which involves a *key pair* consisting of a *private* and a *public* key. Probably the most important application of smart cards is and will be their ability to serve both as a container for key (in particular, private key) information and as a processor for performing the cryptographic signing operation. In other words, the signature generation can be performed without the sensitive key information ever leaving the tamper-proof smart card.

The SignatureCardService is a simple abstraction for the import, verification, and export of RSA key information as well as for the generation and verification of digital signatures generated with the RSA algorithm.

D.3.3 Application Management Card Service

Smart cards can offer multi-function capabilities, for example, a single card can host several applications. Accordingly, smart card applications will have to face situations where cards of the same or different type will host diverse sets of applications. There might even be smart card applications intended for the installation of new or removal of existing card-resident applications or the suspension of certain card-resident applications for a period of time. At the least, a smart card application should be capable of finding out what card-resident applications a particular card hosts and presenting the cardholder a choice from which to select.

In order to address these requirements, card issuers who decide what applications are deployed or planned to be deployed with their cards put additional *meta-information* in the card that card-external applications can access to find out about - and, if necessary, change - the situation in any particular card.

The ApplicationManagementCardService defines a high-level API through which applications can list, select, install, remove, block, and unblock card-resident applications in an issuer independent manner.

Appendix E. Sample OCF Code - GTTEmployeeCard.java.

This is the sample code mentioned in 10.5, “OpenCard Framework” on page 133.

```
//
// Redbook Sample OCF Code - GTTEmployeeCard.java
//

package demos.samples;

import java.io.DataInputStream;
import java.io.DataOutputStream;

import opencard.core.OpenCardConstants;
import opencard.core.service.SmartCard;
import opencard.core.service.CardRequest;
import opencard.core.service.CardServiceException;
import opencard.core.service.CardRequest;
import opencard.core.event.CardTerminalEvent;
import opencard.core.event.CTListener;
import opencard.core.terminal.CardTerminalException;
import opencard.core.terminal.CardTerminalRegistry;
import opencard.core.util.OpenCardClassNotFoundException;
import opencard.core.util.OpenCardLibraryLoadingException;
import opencard.core.util.OpenCardPropertyLoadingException;
import opencard.core.util.OpenCardInitializationException;
import opencard.core.util.HexString;
import opencard.opt.iso.fs.CardFile;
import opencard.opt.iso.fs.CardFileOpenMode;
import opencard.opt.iso.fs.FileSystemCardService;
import opencard.opt.iso.fs.CardFileOutputStream;
import opencard.opt.iso.fs.CardFileInputStream;

import com.ibm.opencard.script.ScriptCardService;
import com.ibm.opencard.iso.fs.*;
import com.ibm.opencard.buffer.TLVBuffer;

/**************************************************************************
* This class encapsulates all card related functionality required by the
* HealthCard Demo.
*
* @version  ¢Id: GTTEmployeeCard.java,v 1.10 1998/04/16 12:22:59 schaeck Exp ¢
* @author Thomas Schaeck (schaeck@de.ibm.com)
**************************************************************************/
public class GTTEmployeeCard implements CTListener
{

  SmartCard     card = null;
  FileSystemCardService filesystemService = null;

  /**************************************************************************
  * Constructs a GTTEmployeeCard object and makes sure that
  * OpenCard is being initialized properly.
  **************************************************************************/
  GTTEmployeeCard () {
    try {
      System.out.println ("GTTEmployeeCard - start OpenCard");
      SmartCard.start ();

      CardTerminalRegistry.getRegistry().addCTListener(this);
      CardTerminalRegistry.getRegistry().createEventsForPresentCards(this);
      System.out.println ("GTTEmployeeCard - OpenCard is up and running");
    } catch (OpenCardPropertyLoadingException e) {
      System.err.println ("OpenCardPropertyLoadingException: ");
      System.err.println (e.getMessage () );
      System.err.println (e.getPossibleCause () );
    } catch (ClassNotFoundException e) {
      System.err.println ("ClassNotFoundException: ");
      System.err.println (e.getMessage () );
    } catch (OpenCardLibraryLoadingException e) {
      System.err.println ("OpenCardLibraryLoadingException: ");
      System.err.println (e.getMessage () );
```

```
    System.err.println (e.getPossibleCause () );
  } catch (OpenCardInitializationException e) {
    System.err.println ("OpenCardInitializationException: ");
    System.err.println (e.getMessage () );
    System.err.println (e.getPossibleCause () );
  }
}

/*****************************************************************************
* React on card removed events sent by OCF: Invalidate card and card service
* @param ctEvent The card inserted event.
*****************************************************************************/
public synchronized void cardRemoved(CardTerminalEvent ctEvent) {
  System.out.println ("GTTEmployeeCard - received CARD_REMOVED event");
  card = null;
  filesystemService = null;
}

/*****************************************************************************
* React on card inserted events sent by OCF: Get new card and card service
* @param ctEvent The card inserted event.
*****************************************************************************/
public void cardInserted(CardTerminalEvent ctEvent) {
  System.out.println ("GTTEmployeeCard - got CARD_INSERTED event");
  try {
    CardRequest cr = new CardRequest(opencard.opt.iso.fs.FileSystemCardService.class);
    card = SmartCard.getSmartCard(ctEvent);
    filesystemService = (FileSystemCardService) card.getCardService(FileSystemCardService.class, true);
  } catch(Exception e) {
    e.printStackTrace();
  }
}

/*****************************************************************************
* Check presence of a GTTEmployee card.
* @return true, if a card is present in a slot, otherwise false
*****************************************************************************/
public boolean GTTEmployeeCardPresent()
{
  return cardPresent() && (filesystemService != null) ;
}

/*****************************************************************************
* Check presence of an (eventually uninitialized) smart card.
* @return true, if a smart card is present in a slot, false otherwise
*****************************************************************************/
public boolean cardPresent()
{
  return (card != null);
}

/*****************************************************************************
* Cleans up GTTEmployeeCard, i.e. in this case shuts down OpenCard.
*****************************************************************************/
void close () {
  System.out.println ("GTTEmployeeCard - stopping OpenCard");
  SmartCard.shutdown ();
}

/*****************************************************************************
* Read the cardholder data from the SmartCard.
*****************************************************************************/
public byte[] getCardHolderData()
{
  try {
    // If no card is present, indicate to application that it must prompt for card
    if (!GTTEmployeeCardPresent())
      return null;

    // mount file system to get access to the root directory
    CardFile root = filesystemService.mount(CardFileOpenMode.BLOCKING);

    // This is the file holding cardholder name and e-Mail address
    CardFile file = new CardFile(root, ":0004");
```

```
      // Create a CardFileInputStream for file
      DataInputStream dis = new DataInputStream(new CardFileInputStream(file));

      // Read in the owner's name
      byte[] cardHolderData = new byte[(int) file.length()];
      System.out.println("reading data");
      dis.read(cardHolderData);

      // Explicitly close the InputStream to yield the smart card to other applications
      dis.close();
      return cardHolderData;
    } catch (Exception e) {
      e.printStackTrace();
      return null;
    }
  }
}
```

Appendix F. Electronic Purse Schemes

As electronic purses are one of the most common smart card applications, in this appendix we describe some that are available from financial institutions worldwide.

An electronic purse is ″an application for effecting prepaid card transactions intended for small amounts″. That definition is taken from the ISO 9992 standard —″Financial Transaction Cards - Messages between the Integrated Circuit Card and Card Accepting Device″.

This statement is not broad enough to describe the range of electronic purse schemes that have been implemented on smart cards. In this appendix, we describe several schemes for handling electronic cash or its equivalent. The three schemes are described with examples of real products that implement each scheme.

F.1.1 What is an Electronic Purse

There are numerous devices that go by the name of electronic purses but which fall outside this definition because they can handle both small and large transaction amounts.

Prepayment instruments on smart cards go by several names:

- Electronic wallet
- Disposable card
- Stored value card
- Prepaid declining balance card
- Cash card
- Electronic purse

All these payment instruments perform basically the same function. They have ″units″ of value which may be tokens or coupons or the electronic equivalent of cash stored on the smart card chip in a secure manner. Those units are deducted from the smart card in exchange for services or items from a merchant.

These electronic prepayment instruments can be sorted into three categories:

- Disposable (for example, Danmønt)
- Re-loadable (for example, VisaCash in UK, Geldkarte)
- Electronic cash (for example, Mondex)

In the remainder of this appendix, we describe each of these categories and give examples of their implementation.

F.2 Disposable Cards

A disposable card is a smart card that is always preloaded with tokens representing a fixed amount and denomination of currency. The card is not personalized to the user. A disposable card is a prepaid, declining-balance smart card because, as these cards are used, the purchase amount is deducted from the remaining balance on the card until it reaches zero. When this happens, the card is discarded because it cannot be recharged. Use of the

disposable card is also anonymous because the transaction is not tied back to the user's bank account and tracked. A telephone card is an example of a disposable card.

Disposable cards are usually issued by the same firm that offers the service being bought by the card. This single issuer/single provider relationship is usually referred to as a "closed system". A closed system can also consist of single issuer/multiple providers and multiple issuers/single provider relationships. A closed system offers little added value to the consumer because the card can only be used within the system for which it was designed and, so, if the user has a need to use different systems, then she must purchase several cards, one for each system. For example, a telephone card purchased in Germany for use on the German public telephone system cannot be used for telephones in France and vice versa.

However, one should not discount the advantages that disposable cards have for the cardholder. They are more convenient to carry around than a pocketful of change and the cardholder pays the exact amount, receiving no change in return. They are therefore safer than cash. For certain services, like public transport, the cardholder obtains faster service because there is no waiting as the attendant or chauffeur counts out change.

The issuer/provider has several streams of revenue coming from the disposable card:

- Since the card is prepaid, the issuer has revenue from cardholder "float" (that is, the issuer receives revenue ahead of the delivery of the service), but there is interest on this prepaid revenue
- There may be establishment fees and usage fees associated with the card.

  For example, when a person makes a call from a public telephone, they must pay a minimum fee for the call (called the antrop value) even for a wrong number.
- There is revenue from unused balances on discarded cards.

In a single issuer/multiple provider relationship, there may also be a revenue stream from merchant fees paid back to the issuer by the various providers for the privilege of using the system.

Three examples of disposable card systems are the Danmønt card in Denmark, the Proton smart card in Belgium and the Geldkarte in Germany.

F.2.1 Danmønt

The Danmønt (Dancoin) card is a disposable smart card which is issued in prepaid amounts of Danish kroner. The Danmønt card can be used wherever payments were previously made with small coins, for example, in parking meters, laundromats, automatic train and bus ticket machines and public telephones. The firm, Dancoin Ltd., would guarantee payment to the vendors who accepted and used the Danmønt system. In return, the banks or merchants would issue the cards to their customers and they, in turn, would use the card at participating merchants.

The Danmønt card terminals can process transactions online or offline. The merchant can select whether the transaction is cleared immediately or whether the transactions are accumulated and processed in a batch.

Danmønt AS was founded in 1991 as a joint venture between PBS (Pengeinstitutternes BetalingsSystemer) and Tele Danmark. Danmønt AS now is a fully owned subsidiary of PBS.

More information about the Danmønt purse scheme can be found at http://www.pbs.dk/pbs_uk/www.danmoent.dk/ and http://www.intellect.com.au/WP/wpaper3.html.

F.2.2 Proton

Banksys, a consortium of Belgian banks, introduced the Proton SVC in August 1994 in two Belgium cities. The initial Proton smart card used a Dallas Semiconductor security module on the smart card that was incompatible with many smart card readers. Merchants required special card terminals which would read the Proton card. The Proton system has since been changed over to use a Bull CP8 smart card and any smart card reader which accepts these card will interoperate with the Proton card.

Banksys has licensed the Proton card all over the world. The Chip Knip card from Interpay B.V. in Holland is based on the Proton system as is Quicklink in Australia, Telekurs in Austria and EXACT in Canada.

F.3 Re-loadable Cards

At the beginning of this appendix, we gave the ISO 9992 definition for an electronic purse. The ISO 9992 standard goes on to define a re-loadable card (or electronic wallet) as ″an application for effecting prepaid card transactions, intended for large amounts″. this sounds similar to the disposable cards described earlier in F.2, “Disposable Cards” on page 195. However, there are significant differences.

First, an electronic wallet is empty when it is issued to the cardholder. Second, they are personalized at issuance and loaded with an initial electronic value. Since the cards are personalized, they are treated similar to cheques. For example, transactions are recorded against the cardholder and tracked in the banking system. Third, the electronic wallet when emptied can be recharged with any value that the cardholder desired. They are not limited to fixed, prepaid amounts.

Like the SVC, there are several streams of revenue to the issuer:

- There are the establishment fees and usage fees to be paid by the cardholder
- Company loyalty programs that may be associated with the card generate revenue
- Customer pays in advance of shopping

F.3.1 GeldKarte

Geldkarte is a smart card that implements both a re-loadable electronic wallet and an electronic cheque for the German banks. As mentioned earlier, the Geldkarte is also a prepaid, declining-balance smart card. The Geldkarte smart card is based upon an IBM Multifunction Card card with a special card operating system. The ZKA (Central Committee of the German Finance Industry) publishes the Geldkarte electronic wallet scheme as an open specification that anyone can

use to implement their own wallet. There are no licensing fees to be paid to the ZKA.

The Geldkarte user would load the wallet with money at a kiosk in the bank. The user has a PIN protection on the wallet when putting money into the wallet but is not required to enter a PIN code when using the card. All online payments are anonymous. The merchant is charged 0,3% of the transaction amount by the bank with a minimum charge of DM 0,02 per transaction. All online transactions are cleared immediately and offline transactions act as a ″guaranteed electronic cheque″. The bank will guarantee payment of the merchant for the amount of the offline transaction.

The Geldkarte project began in 1990 as a ″home banking″ field trial with the cooperation of Giesecke and Devrient GAD (a joint use center of regional cooperative banks), German Telekom and IBM. The project went through further field trials and pilots in several German cities between 1991 and 1996 before being announced in 1997 as a full service offering from the ZKA for all German citizens.

More information about the Geldkarte wallet scheme can be found on the Web at http://www.familaol.de/spezialgk.htm and at http://www.darmstadt.gmd.de/~ seliger/GeldKarte/ectop.html.

F.3.2 Chipper

Chipper is a multi-application smart card with an electronic wallet.

Building on previous experiences with smart cards including the Dutch Student card and the RET card, an SVC for Dutch pay phones, in 1997 KPN Telecom and the Postbank introduced the Chipper card. Consumers can use the cards for home banking, pay phones, parking tickets, public transport and loyalty programs at participating merchants.

The wallet is loaded at any participating bank by securely transferring money from the cardholder′s account to the smart card. Then, when the card is used, the cardholder would insert the smart card into the merchant′s smart card reader. containing a secure application module (SAM). A SAM is nothing more than another smart card securely integrated into the smart card reader and programmed with the terminal′s encryption keys. Money is transferred from the smart card to the SAM. The balance on the smart card is debited by the authorized amount and the value in the SAM is incremented by a similar value. This is known as a secure end-to-end connection. Later, the merchant would perform nightly clearances that would securely transfer electronic funds from the SAM to the bank′s central computing system.

More information about the Chipper wallet scheme can be found at http://www.chipper.com/.

F.4 Electronic Cash

The third system of smart card-based purses is the electronic replacement for cash. The value on the card is an electronic representation of cash.

The main characteristics of this purse scheme are:

- Secure: only the cardholder will be able to use the electronic cash stored in the card
- No change: the exact amount will be withdrawn
- Remote transfer: the purse can be loaded remotely

Electronic cash offers the promise of reduced handling costs, increased security and a larger variety of services than is possible today with cash.

F.4.1 Mondex

Mondex is an electronic payment system developed by National Westminister Bank in the United Kingdom. It became operational in July 1995. It offers its users a payment instrument that is equivalent to cash but with distinct advantages over cash.

Mondex implements a card-to-card transfer mechanism. In fact, all Mondex transactions are always card-to-card transactions. Instead of going through a clearing process as would a credit card or an electronic purse transaction, the Mondex scheme securely transfers cash value between smart cards.

For example, a consumer-to-merchant transaction would proceed as follows. The merchant would install a Mondex-compatible card reader device into which he plugs his Mondex card. All retail purchases would involve plugging the consumer′s Mondex card into the reader and transferring cash value from the consumer′s card to the merchant′s card. The reader records all transactions similar to a point of sale terminal so that the merchant can do totals processing, settlement and reconciliation.

The bank that issued the card has access to only two transactions:

- The transfer of cash value into the consumer′s card when the card is reloaded
- The transfer of cash value from the merchant′s card to his/her bank account.

What makes Mondex better than cash is what it allows users to do:

- Load up, over the telephone or at a participating ATM, five ″pockets″ on the card with any currency the user wishes
- Lock and unlock the Mondex card using a four-digit code chosen by the cardholder
- Obtain a log of the last five transactions on the card
- Make direct card-to-card payments over special Mondex-compatible telephones
- Obtain card balance using a Mondex-compatible keyring balance checker
- No change to deal with

Public system components such as card terminals, balance readers, smart cards, etc. must be approved by Mondex International for use in the system. Mondex partners in participating countries buy a franchise license from Mondex International which gives them full rights to market and operate a Mondex system within that country.

Future implementations of Mondex will run under a card operating system called Multos. Multos is a secure, multi-application interpretative smart card operating system owned by MCI/MCX. Just like JavaCard, it allows a card issuer to combine smart card applications onto a single, physical smart card, which

increases the potential value of the card to the cardholder. Figure 54 on page 200 shows the Multos card operating system and how it handles multiple applications such as the Mondex purse and a loyalty scheme.

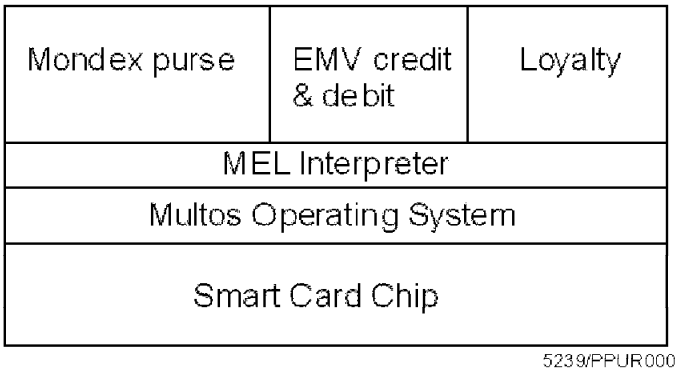


Figure 54. Multos Card Operating System and the Mondex Purse

A consortium of companies promotes Multos smart cards similar to what the JavaCard Forum does for JavaCard (see B.2.3, "JavaCard Forum" on page 182 for information about JavaCard Forum). This consortium is called MAOSCO (http://www.multos.com/200.html). The initial members of MAOSCO include:

- American Express
- Dai Nippon Printing
- Mondex International Ltd.
- Siemens
- Fujitsu
- Hitachi
- Motorola
- MasterCard International
- Keycorp

MAOSCO is responsible for promoting Multos as the operating system of choice for multi-application smart cards, managing the Multos specification and providing Multos licensing and certification services.

Application programmers who wish to develop on the Multos environment must pay a GBP 100,000 Application license fee to MAOSCO. This fee covers the cost of administration and the application development toolset. Card issuers who wish to integrate Multos onto their smart cards must pay a GBP 250,000 Implementation license fee to MAOSCO plus a yearly license fee of GBP 50,000.

Multos is currently implemented on Hitachi smart card hardware by DNP, Siemens smart card hardware by Keycorp, and Motorola smart card hardware by Motorola. There are presently a dozen or so Multos applications written by smart card manufacturers such as Gemplus, Schlumberger, De La Rue, ORGA, Security Plastics and Integrated Card Technology among others.

More information about the Mondex purse scheme can be found on the Web at http://www.mondex.com/mondex/. Further information about Multos and MAOSCO can be obtained over the Web from http://www.multos.com or by sending email to customer.services@multos.com.

Appendix G. Special Notices

This publication is intended to help you to design, plan and implement projects that involve smart cards. The information in this publication is not intended as the specification of any programming interfaces that are provided by the products mentioned in this redbook. See the PUBLICATIONS section of the IBM Programming Announcement for these products for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

DB2
IBM
ThinkPad
VisualAge
400

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix H. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

H.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 205.

- *IBM Digital Signature Solution Concepts and Implementation*, SG24-5283 (available at a later date)

H.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

H.3 Other Publications

These publications are also relevant as further information sources:

- *Smart Card Handbook*, by W. Rankl and W. Effing, published by John Wiley & Sons, Inc.
- *Applied Cryptography*, by Bruce Schneier, published by John Wiley & Sons, Inc.
- *Smart Card ToolKit Overview*, GC33-7000
- *Smart Card ToolKit Programmer's Reference*, SC33-7001
- *Smart Card ToolKit Tools Reference*, SC33-7002
- *Smart Card ToolKit User's Guide*, SC33-7003
- *Smart Card ToolKit Personalization Guide*, SC33-7004

See also 10.11, “Smart Card Programming Information” on page 136.

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com/.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/

- **PUBORDER** — to order hardcopies in the United States
- **Tools Disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLCAT REDPRINT
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

- **REDBOOKS Category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.html). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	Internet
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

Index # 4421 Abstracts of new redbooks
Index # 4422 IBM redbooks
Index # 4420 Redbooks for last six months

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com/
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of Abbreviations

APA	all points addressable
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
PROFS	Professional Office System
AID	Application Identification Data
AIS	Application Issuing System
AMS	Application Management System
ANSI	American National Standards Institute
AP	Application Provider
APDU	Application Protocol Data Unit
API	Application Programming Interface
ASC	Application Specific Command
ATM	Automatic Teller Machine<b>
CA	Certification Authority
Card OS	Card Operating System
CBC	Cipher Block Chaining
CCA	Common Cryptographic Architecture
Cert	Certificate
CertMS	Certificate Management System
CGI	Common Gateway Interface
CHV	Cardholder Verification
CI	Card Issuer
CID	Card Identification Data
CIS	Card Issuing System
CMS	Card Management System
COS	Card Operating System
CPS	Card Personalization Station
CRL	Certificate Revocation List
DAC	Data Authentication Code
DB	Database
DBMS	Database Management System
DC	DataCard
DEA	Data Encryption Algorithm
DES	Data Encryption Standard
DKMS	Distributed Key Management System
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECB	Electronic CodeBook
EEPROM	Electrical Erasable Programmable ROM
EMV	EuroPay, MasterCard, Visa
E-R	Entity-Relationship
FTP	File Transfer Protocol
GSA	Government Services Administration
GSCTIG	Government Smart Card Technical Interoperability Guidelines
GUI	Graphical User Interface
HLD	High Level Design
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IA	Initialization Agency
ICC	Integrated Circuit Card
ICRF	Integrated CRyptographic Feature
ICSF	Integrated Cryptographic Services Facility
IEC	International Electrotechnical Commission
I/F	Interface
IFD	InterFace Device
IMD	IBM Microelectronics Division
ISO	International Organization for Standardization
JDBC	Java Database Connectivity
JVM	Java Virtual Machine
KGK	Key Generation Key
KID	Key Identification Data
KIS	Key Issuing System
KMS	Key Management System
LRA	Local Registration Authority
MAA	Message Authentication Algorithm
MAC	Message Authentication Code

MAOSCO	Multi-Application Operating System Company
MD	Message Digest
MEL	MULTOS Executable Language
MFC	Multi Function Card
MPCOS	Multiple Chip Operating System
MULTOS	MULTi-application Operating System
NC	Network Computer
NLS	National Language Support
OCF	OpenCard Framework
ODBC	Open Database Connectivity
PA	Personalization Agency
PBB	Personalization Building Block
PDS	Personalization Data Set
PGP	Pretty Good Privacy
PIN	Personal Identification Number
PK	Public Key
PKCS	Public Key Cryptographic Standard
PKI	Public Key Infrastructure
PO	Purchase Order
RA	Registration Authority
RAM	Random Access Memory
ROM	Read Only Memory
RSA	Rivest, Shamir, Adeleman Cryptoalgorithm
SAM	Secure Application Module
SC	Smart Card
SCQL	Structured Card Query Language
SCS	Smart Card Solution
SET	Secure Electronic Transaction
SHA	Secure Hash Algorithm
S-HTTP	Secure-HyperText Transfer Protocol
SK	Secret Key
SSL	Secure Socket Layer
TLV	Tag Length Value
TSS	Transaction Security System
TTP	Trusted Third Party
URL	Uniform Resource Locator

Index

E

F

G

H

I

J

K

T

U

V

W

ITSO Redbook Evaluation

Smart Cards: A Case Study
SG24-5239-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
__Customer __Business Partner __Solution Developer __IBM employee
__None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

SG24-5239-00
Printed in the U.S.A.

Smart Cards: A Case Study

SG24-5239-00